ARS: Adaptive Reward Scaling for Multi-Task Reinforcement Learning

Myungsik Cho [1] Jongeui Park [1] Jeonghye Kim [1] Youngchul Sung [1]

Abstract

Multi-task reinforcement learning (RL) encounters significant challenges due to varying task complexities and their reward distributions from the environment. To address these issues, in this paper, we propose Adaptive Reward Scaling (ARS), a novel framework that dynamically adjusts reward magnitudes and leverages a periodic network reset mechanism. ARS introduces a history-based reward scaling strategy that ensures balanced reward distributions across tasks, enabling stable and efficient training. The reset mechanism complements this approach by mitigating overfitting and ensuring robust convergence. Empirical evaluations on the Meta-World benchmark demonstrate that ARS significantly outperforms baseline methods, achieving superior performance on challenging tasks while maintaining overall learning efficiency. These results validate ARS's effectiveness in tackling diverse multi-task RL problems, paving the way for scalable solutions in complex real-world applications.

1. Introduction

In recent years, the field of deep reinforcement learning (RL) has achieved remarkable success in addressing complex control problems, such as mastering Atari games (Hafner et al., 2021; Kapturowski et al., 2023; Schwarzer et al., 2023) and advancing locomotion control (Haarnoja et al., 2018b; Fujimoto et al., 2018; Cetin et al., 2022). Despite these achievements, the field remains limited by a task-specific paradigm that demands substantial data and computational resources to train separate policies for each task. Generalizing a single policy to perform effectively across multiple tasks poses a significant challenge, particularly in domains such as robotic control (Kaufmann et al., 2023; Tang et al., 2024), where mastering diverse skills is essential.

[1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea. Correspondence to: Youngchul Sung <ycsung@kaist.ac.kr>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Multi-task reinforcement learning (multi-task RL) (Wilson et al., 2007; Zeng et al., 2018; Yang et al., 2020; Sun et al., 2022; Hendawy et al., 2024; Cho et al., 2024) enables a single policy network to handle multiple tasks with shared parameters (Caruana, 1997), improving data efficiency. However, it faces the challenge of negative transfer, where learning one task disrupts others, destabilizing training and limiting scalability (Sun et al., 2020).

In addition to negative transfer, this paper introduces another critical issue in multi-task RL: the significant variation in reward scales across tasks, which can adversely affect overall performance. To explore this issue, we focus on the impact of reward scaling approach in multi-task scenarios, studied in single-task settings (Henderson et al., 2018; Wu et al., 2018), which involves applying a constant scaling factor to rewards. We observed that the varying magnitudes of rewards across tasks cause biases during training, resulting in overfitting on tasks with highly amplified rewards and a decline in average performance across all tasks.

To address these challenges, we propose Adaptive Reward Scaling (ARS), a novel framework that dynamically adjusts reward scaling factors, ensuring balanced reward magnitudes across diverse tasks. ARS is built upon two core components: (1) adaptive reward scaling that adjusts scaling factors by analyzing the distribution of rewards within each task's experience replay buffer (Mnih et al., 2015), and (2) periodic resetting of network parameters to prevent overfitting and improve convergence. This approach ensures that challenging tasks receive adequate emphasis on their rewards, preventing overfitting to simpler tasks with higher rewards and enhancing overall performance. Experiments conducted on the Meta-World benchmark (Yu et al., 2019), which includes 50 robotic manipulation tasks, demonstrate that ARS significantly outperforms baseline methods. Additionally, ARS integrates seamlessly into existing off-policy algorithms with minimal modifications, making it a practical and effective solution for real-world applications.

The primary contributions of this work are:

- The introduction of reward scale variation of a challenge in multi-task RL, demonstrating how fixed reward scaling can lead to biased training, overfitting, and overall performance degradation

- A novel framework, adaptive reward scaling mechanism with reset strategies, which robustly handles varying reward distributions and enhances the learning process for challenging tasks.
- Empirical validation on the Meta-World benchmark Empirical, demonstrating superior performance and providing detailed insights into the mechanisms driving ARS's effectiveness.

2. Preliminaries

2.1. Multi-Task Reinforcement Learning

A RL problem is typically represented as a Markov decision process (MDP), defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho, \gamma)$. Here, $\mathcal{S}$ represents 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 distribution over initial states, and $\gamma \in [0, 1)$ is the discount factor. At each time step t, the agent receives an observation of the state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to the policy $\pi(a_t \mid s_t)$. The environment returns a reward $r_t = r(s_t, a_t)$ and transitions to the next state s_{t+1} according to the transition probability distribution $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. In single-task RL, the goal is to maximize the expected sum of discounted rewards:

$$J(\pi) = \mathbb{E}_{\tau \sim \rho_\pi}\left[\sum_{t=1}^{H} \gamma^{t-1} r_t\right]. \tag{1}$$

In multi-task RL, the focus shifts to optimizing a policy that achieves high performance across a broad range of tasks. Specifically, multi-task RL contains a set of tasks $\mathcal{C} = \{\mathcal{T}_i\}_{i=1}^N$ and a distribution $p(\mathcal{T})$ over these tasks, where each task $\mathcal{T}_i$ is represented by an MDP $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, r_i, \rho_i, \gamma, H)$. The tasks share common state and action spaces but differ in reward functions, transition dynamics, and initial state distributions. Assuming $p(\mathcal{T})$ is uniform, the goal is to learn a shared policy π that maximizes the average return over all tasks. Formally, the objective is:

$$\max_\pi \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})}[J(\pi, \mathcal{T})], \tag{2}$$

where $J(\pi, \mathcal{T})$ denotes the expected sum of discounted rewards for task $\mathcal{T}$ under policy π.

In this work, we use a shared policy $\pi_\theta(a|s, z)$, which incorporates a task representation z alongside the state s. We represent each task using a one-hot encoding. To train the policy, we adopt Soft Actor-Critic (SAC) (Haarnoja et al., 2018b) as the underlying algorithm.

2.2. Replay Buffer

In off-policy RL, the replay buffer (Mnih et al., 2015) improves sample efficiency and training stability. Denoted as $\mathcal{D}$, it stores transitions (s_t, a_t, r_t, s_{t+1}) collected from the agent's interactions, which are reused during training to stabilize updates to the policy or value function.

To extend this concept to multi-task RL, each task $\mathcal{T}_i$ in the task set $\mathcal{C}$ is assigned a distinct replay buffer $\mathcal{D}_i$. This design isolates experiences for each task, enabling the agent to adapt effectively to the unique dynamics and reward structures of individual tasks.

2.3. Reward Scaling

Reward scaling (Henderson et al., 2018; Wu et al., 2018) is a common preprocessing technique in RL that adjusts reward magnitudes to enhance learning stability and convergence. In single-task RL, the reward signal r_t is transformed using a scaling factor c^{rew}, helping to mitigate the impact of inconsistent reward magnitudes and enabling more stable updates to the value function or policy.

When applied to multi-task RL, the need for reward scaling becomes even more pronounced due to the variation in reward distributions across tasks. To address these differences, task-specific reward scaling is implemented, where each task's rewards are adjusted using unique scaling factors $\{c_i^{\text{rew}}\}_{i=1}^N$. This approach ensures consistency in reward magnitudes across tasks, supporting stable learning.

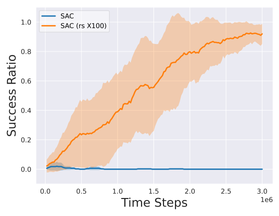
2.4. Parameter Resetting

In single-task deep RL, early learning often leads to overfitting to initial experiences, a phenomenon known as primacy bias, as highlighted by Nikishin et al. (2022). To mitigate this, Nikishin et al. (2022) proposed a method that periodically resets the parameters of the RL agent while preserving the replay buffer. This method leverages the replay buffer's stored experiences to enable the agent to recover quickly from the reset, bypassing the primacy bias. Although resets temporarily reduce performance, the agent rapidly regains its capabilities by focusing on high-quality trajectories acquired later in the learning process, resulting in significant improvements across various environments.

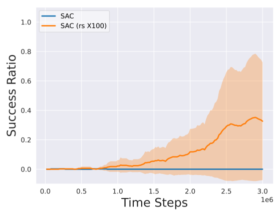
In multi-task deep RL, resetting deep networks offers additional benefits. Beyond counteracting primacy bias, it also addresses biases toward tasks learned early in training. By redistributing focus across tasks, the reset mechanism promotes more balanced learning and enhances performance across the entire task set (Cho et al., 2024).

3. Motivation: Reward Scaling in Deep RL

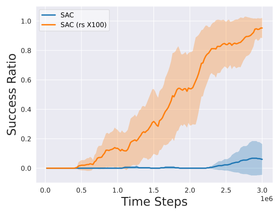
This section introduces a motivating example highlighting the limitations of conventional deep multi-task RL, which assigns an equal reward scale $\{c_i^{\text{rew}} = 1.0\}_{i=1}^N$ to all target tasks. The challenge becomes particularly evident when the



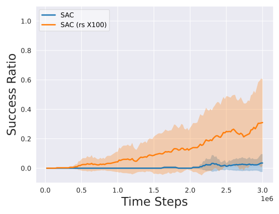
(a) push



(b) pick-place



(c) shelf-place



(d) basketball

Figure 1. Learning curves of four tasks: 'push,' 'pick-place', 'shelf-place' and 'basketball'. With a reward scaling factor of 100, all tasks succeed, whereas without reward scaling, all tasks fail.

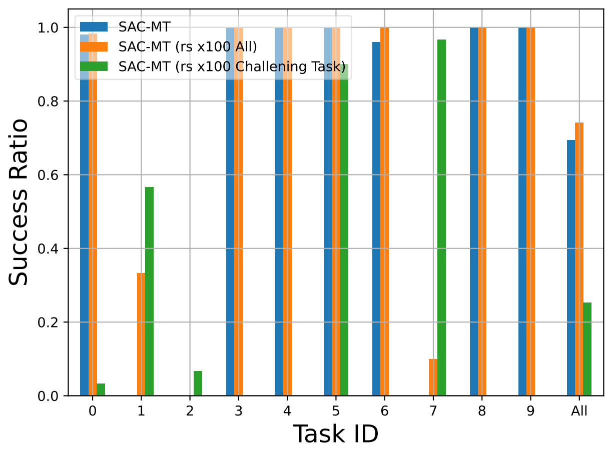


Figure 2. Comparison of the final success rate per task in the MT10 benchmark among SAC-MT, SAC-MT with reward scaling applied to all tasks, and SAC-MT with reward scaling applied only to the challenging tasks: 'push,' 'pick-place,' and 'peg-insert-side.'

target tasks have significantly different reward distributions.

3.1. Reward Scaling in Single-Task RL

Previous works (Henderson et al., 2018; Wu et al., 2018) demonstrate the effectiveness of reward scaling in single-task RL. To investigate its impact within the Meta-World benchmark (Yu et al., 2019), a widely used environment for evaluating multi-task RL, we conducted independent experiments on four challenging tasks—'push,' 'pick-place', 'shelf-place' and 'basketball'. These experiments utilized the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018b), comparing performance across two reward scaling factors—$\{1.0, 100.0\}$, where a reward scaling factor of 1.0 represents the naive SAC method.

As shown in Figure 1, reward scaling method outperforms the non-scaling approach and ensures successful task completion, demonstrating its effectiveness in single-task RL.

3.2. Challenges of Reward Scale in Multi-Task RL

In the previous subsection, we observed that the reward scaling approach effectively supports the training of successful policies in Meta-World. Here, we examine the impact of the reward scaling method on a multi-task RL agent.

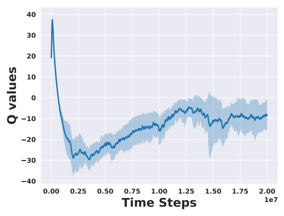
3.2.1. Failures in Challenging Tasks Without the Reward Scaling Approach

We considered the MT10 benchmark, consisting of 10 manipulation tasks in Meta-World, and trained the agent with the SAC-MT algorithm, using one-hot encoding for task identification. Figure 2 (blue) shows the final success ratio per task with SAC-MT. Notably, tasks with IDs 1, 2, and 7 ('push,' 'pick-place,' and 'peg-insert-side') failed to train successfully.

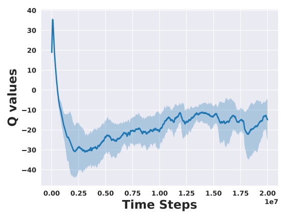
In order to investigate the failure in these challenging tasks with the SAC-MT algorithm, we focused on the magnitude of the initial reward for each task in the MT10 benchmark. We observed that the reward scale for certain challenging tasks is substantially lower compared to others, differing by a factor of $10 \sim 100$. This imbalance negatively affected the training of the Q-network for these tasks, as shown in Figure 3. Specifically, the Q-values for these tasks were negative, even though the rewards from Meta-World are always positive. This highlights the challenges of training the Q-network when reward scales are not sufficiently high. More learning curves of Q-value are provided in Appendix D.1

3.2.2. Fixed Reward Scaling in Multi-Task RL

To further investigate the effects of reward scaling in multi-task RL, we conducted additional two experiments with SAC-MT. In the first, a reward scaling factor of 100 was applied uniformly across all tasks, while in the second, the scaling was selectively applied only to the challenging tasks—'push,' 'pick-place,' and 'peg-insert-side'—leaving the other tasks unscaled. Figure 2 shows the final success ratios achieved with SAC-MT when incorporating reward scaling. Orange represents the scenario where scaling was applied to all tasks, and green denotes the selective scaling approach. In the uniform scaling experiment, Task ID 1 showed progress, but Task IDs 2 and 7 remained unresolved. In contrast, with selective scaling applied only to challenging tasks, Task IDs 1, 2, and 7 were successfully trained.



(a) push



(b) pick-place

Figure 3. Learning curves of Q-values for two tasks: 'push,' and 'pick-place'.

However, this approach caused many other tasks to fail during training, reducing the overall average performance.

This observation highlights a key challenge in multi-task RL: careful reward scaling is essential for handling tasks with diverse reward distributions. Improper scaling can result in overfitting on tasks with highly amplified rewards, failure to learn others, and overall performance decline. To address this challenge, the remainder of this paper introduces a novel approach to effectively adjust reward scaling in multi-task RL, aiming to achieve better overall performance.

Algorithm 1 Adaptive Reward Scaling (ARS)

```
1:  Initialize policy network $\pi_\theta$, $Q$-value network $Q_\psi$
2:  Initialize replay buffer $\mathcal{D}_i$ for each task $\mathcal{T}_i \in \mathcal{C}$
3:  for $t = 1, 2, \ldots, T_{init}$ do
4:      for all $\mathcal{T}_i \in \mathcal{C}$ do
5:          Interact with the environment of $\mathcal{T}_i$ with a random
            policy and store data in $\mathcal{D}_i$
6:      end for
7:  end for
8:  Initialize the reward scaling factors $\{c_i^{rew}\}_{i=1}^N$ using (6)
9:  for $t = T_{init}+1, T_{init}+2, \ldots,$ do
10:     for all $\mathcal{T}_i \in \mathcal{C}$ do
11:         Interact with the environment of $\mathcal{T}_i$ with $\pi_\theta$ and
            store data in $\mathcal{D}_i$
12:     end for
13:     Update $\theta$ and $\psi$, using the data in $\{\mathcal{D}_i\}_{i=1}^N$ and the
        scaling factors $\{c_i^{rew}\}_{i=1}^N$
14:     if $t \;\%\; T_{\text{reset}} == 0$ then
15:         Update $\{c_i^{rew}\}_{i=1}^N$ using (6)
16:         Randomly reinitialize $\theta$ and $\psi$
17:     end if
18: end for
```

4. Adaptive Reward Scaling for Multi-Task Reinforcement Learning

To address the issue raised in the previous section regarding the effective handling of varying reward distributions in multi-task RL, we propose the **A**daptive **R**eward **S**caling (ARS) algorithm. This approach is specifically designed to address imbalances in reward scales across tasks in multi-task RL, thereby improving overall learning efficiency. The ARS framework comprises two key components: a reset mechanism and a dynamic reward scaling strategy, which work together to optimize training performance.

- **History-Based Reward Scaling Strategy**: At the core of ARS is a reward scaling strategy that adjusts reward scales for tasks with lower reward magnitudes, ensuring uniform reward scaling across tasks. This approach leverages a metric derived from the distribution of rewards within each task's experience replay buffer, enabling real-time and adaptive reward scaling.
- **Reset Mechanism**: ARS integrates a reset mechanism to mitigate biases toward the tasks with highly amplified rewards. Through periodic reinitialization of network parameters while maintaining the replay buffer, this approach enhances adaptability and promotes improved performance across diverse tasks.

The overall structure of the proposed ARS framework is presented in Appendix F. ARS employs two networks: a policy distribution π_θ and a state-action value function Q_ψ. Each task $\mathcal{T}_i$ in the task set $\mathcal{C}$ is assigned a separate buffer $\mathcal{D}_i$, maintained independently to store task-specific interactions. The goal is to maximize the sum of each task's objective, i.e., $\sum_{\mathcal{T}i\in\mathcal{C}} J(\pi\theta, \mathcal{T}_i)$. The key components of ARS are further detailed in the following subsections.

4.1. History-Based Reward Scaling Strategy

From the observations in the previous section, assigning higher reward scaling factors to complex tasks is necessary. However, using a fixed reward scaling approach is inadequate as shown in Figure 2, as the reward magnitude for tasks with a high scaling factor progressively increases during training, leading to biases toward those tasks. To address this, the reward for each task $\mathcal{T}_i$ should be adaptively scaled.

This raises the question: "How should the scaling factor for each task $\mathcal{T}_i$ be determined in real-time?" To address this, we examine the training objective in standard off-policy multi-task RL implementations. Given a task set $\mathcal{C} = \{\mathcal{T}_i\}_{i=1}^N$ and the corresponding experience replay buffers $\{\mathcal{D}_i\}_{i=1}^N$, the agent is trained using the following objectives:

$$\ell^\pi(\theta) = \frac{1}{|\mathcal{C}|} \sum_{\mathcal{T}_i\in\mathcal{C}} \mathbb{E}_{s\sim\mathcal{D}_i}\left[\ell^\pi(\theta; s)\right], \tag{3}$$

$$\ell^Q(\psi) = \frac{1}{|\mathcal{C}|} \sum_{\mathcal{T}_i\in\mathcal{C}} \mathbb{E}_{(s,a)\sim\mathcal{D}_i}\left[\ell^Q(\psi; s, a, r, s')\right], \tag{4}$$

where $\ell^\pi(\theta; s)$ and $\ell^Q(\psi; s, a, r, s')$ are the per-sample actor and critic loss functions, respectively.

In this objective, the reward scaling factor only influences critic training. For the SAC algorithm, the per-sample critic loss for $(s, a, r, s') \in \mathcal{D}_i$ is expressed as

$$\ell^Q(\psi; s, a, r, s') = \\ \Big[\big(r + \gamma(Q_{\hat{\psi}}(s', \pi_\theta(s')) - \alpha \log \pi_\theta(s')\big) - Q_\psi(s, a)\Big]^2 \tag{5}$$

Critic training, therefore, can be viewed as a regression problem, where the targets vary, and the mean reward in the experience replay buffer $\mathcal{D}_i$ plays a crucial role in determining the magnitude of the critic values.

Building on these observations, we define the reward scaling factor c_i^{rew} for each task $\mathcal{T}_i$ as the following equalizer rule:

$$c_i^{\text{rew}} = \frac{max\big(\{\bar{r}_1, \ldots, \bar{r}_N\}\big)}{\bar{r}_i} \tag{6}$$

where $\bar{r}_i$ is the mean reward for task $\mathcal{T}_i$ within its replay buffer $\mathcal{D}_i$ for $i = 1, \cdots, N$. This formulation ensures consistent reward magnitudes across tasks, enabling stable critic learning without bias towards easy tasks. Using this adaptive reward scaling factor c_i^{rew}, the critic loss for $(s, a, r, s') \in \mathcal{D}_i$ is modified as follows:

$$\ell^Q(\psi; s, a, r, s') = \\ \Big[\big(c_i^{\text{rew}} r + \gamma(Q_{\hat{\psi}}(s', \pi_\theta(s')) - \alpha \log \pi_\theta(s')\big) - Q_\psi(s, a)\Big]^2 \tag{7}$$

4.2. Reset Mechanism

A key issue with the naive adaptive reward scaling approach is that the rewards stored in the experience replay buffer fluctuate throughout training due to the changing reward scaling factor. This variability can destabilize critic training, as the regression target in Equation (7) changes frequently. Additionally, it may introduce biases toward tasks with highly amplified reward scaling factors, ultimately leading to suboptimal performance.

To address this issue, we adopt the reset mechanism proposed by Nikishin et al. (2022), which involves resetting the policy parameters θ and Q-value function parameters ψ while preserving the experience replay buffers. Furthermore, we adjust the reward scaling factor c_i^{rew} only prior to resets, ensuring stable critic training throughout the process.

To conclude, our 'History-Based Reward Scaling' strategy adaptively scales rewards across tasks of varying complexities, preventing bias toward simpler tasks with larger raw magnitude of rewards. However, dynamically changing reward scales can destabilize training critic network. To address this, we employ a Reset Mechanism, which resets the policy and critic networks whenever the reward scales are updated, while retaining all historical data in the replay buffers. This approach maintains stable critic learning throughout training, effectively balancing performance across a diverse set of tasks. The main contribution of our work is the novel reward scaling framework, thus any type of multi-task RL method can be used for practical implementation. The overall process of the proposed method is summarized in Algorithm 1.

4.3. Enhancements

Our ARS framework builds upon the SAC-MT algorithm, a simplest baseline for multi-task RL. In the main results, we refer to this combination as ARS. Beyond the default ARS setup, we apply layer normalization (Ba et al., 2016), a well-known technique for stabilizing learning in deep RL (Ball et al., 2023), to both the input and hidden layers of the critic network, and denote this variant as ARS-LN. Building on ARS-LN, we further incorporate Low-Rank Adaptation (LoRA) (Hu et al., 2022) to introduce task-specific parameterization during the later stages of training—specifically, after 75% and 83.3% of the total training steps for MT10 and MT50, respectively. We denote this variant as ARS-LoRA. For efficient adaptation, we use a rank of $r = 8$ for MT10 and $r = 16$ for MT50. Further implementation details regarding the integration of LoRA into ARS are provided in Appendix B.

5. Experiments

Benchmarks. To evaluate the effectiveness of the proposed method on various tasks, we conducted experiments using the Meta-World benchmark (Yu et al., 2019), which includes 50 distinct robotic control tasks involving a Sawyer arm in the MuJoCo environment (Todorov et al., 2012). Our experiments used two setups: MT10 and MT50, which consist of 10 and 50 manipulation tasks, respectively. A detailed description of the benchmarks are provided in Appendix A.

Baselines. We compared the proposed method against the following baseline approaches: 1) SAC with Multi-Task (SAC-MT): A shared policy that uses a one-hot task identification encoding along with the current state as input. 2) SAC with Multi-Task Multi-Head (SAC-MT-MH) (Yu et al., 2019): Similar to SAC-MT but incorporates an independent final layer (multi-head) in the policy network for each task. 3) SAC with Soft Modularization (SAC-Soft-Modular) (Yang et al., 2020): A policy architecture utilizing multiple modules with a soft modularization technique that determines a routing strategy for each task. 4) Gradient Surgery for Multi-Task Learning (PCGrad) (Yu et al., 2020): This method addresses conflicting gradients in multi-task learning by projecting gradients onto a shared subspace. 5) Parameter-Compositional Multi-Task Reinforcement Learning (PaCo)(Sun et al., 2022): A multi-task RL approach

Table 1. Comparison of average and per-task success rates (%) on the Meta-World MT10 benchmark. Refer to Appendix A for task names corresponding to each task ID.

	Task ID										
Algorithm	0	1	2	3	4	5	6	7	8	9	Average
SAC-MT	98±2.4	0±0.0	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	96±2.0	0±0.0	**100±0.0**	**100±0.0**	69.4 ± 0.8
SAC-MT-MH	**100±0.0**	28±21.8	0±0.0	**100±0.0**	98±2.5	**100±0.0**	**100±0.0**	46±21.8	**100±0.0**	**100±0.0**	77.2 ± 11.9
Soft Modular	**100±0.0**	32±14.9	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	12±14.9	**100±0.0**	**100±0.0**	74.4 ± 10.5
PCGrad	94±3.7	0±0.0	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	54± 39.9	**100±0.0**	**100±0.0**	74.8 ± 13.7
PaCo	**100±0.0**	44±25.2	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	80 ± 40	**100±0.0**	**100±0.0**	82.4 ± 14.2
MOORE	**100±0.0**	92.5 ± 7.1	0±0.0	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	67.5 ± 46.5	**100±0.0**	**100±0.0**	86.0 ± 4.8
SMT	96±3.7	62±17.9	34±13.8	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	76±31.9	**100±0.0**	**100±0.0**	86.8 ± 8.6
ARS (Ours)	92.5±11.7	**98.8±3.5**	86.3±15.1	98.8±3.5	**100±0.0**	**100±0.0**	**100±0.0**	96.3±7.4	**100±0.0**	**100±0.0**	97.3 ± 2.1
ARS-LN (Ours)	93.8±7.4	**98.8±3.5**	93.8±9.2	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	98.6 ± 1.5
ARS-LoRA (Ours)	**100±0.0**	97.5±7.1	**98.8±3.5**	**100±0.0**	**100±0.0**	**100±0.0**	**100±0.0**	98.8±3.5	**100±0.0**	**100±0.0**	**99.5 ± 0.8**

that utilizes parameter compositionality by constructing task-specific parameters from a common pool of shared base parameters. 6) Multi-Task Reinforcement Learning with Mixture of Orthogonal Experts (MOORE) (Hendawy et al., 2024): Neural network architecture utilizing mixture-of-experts with gram-schmidt process 7) Scheduled Multi-Task Training for Multi-Task RL (SMT) (Cho et al., 2024): A framework designed to mitigate simplicity bias using a "Hard Tasks First" scheduling scheme combined with a reset mechanism.

Training settings. All methods are trained using 20 million samples in the MT10 benchmark and 100 million samples in the MT50 benchmark. Policy evaluation is based on the success ratio across all tasks, where the success ratio for a specific task is determined by averaging its success rate over 10 episodes with different sampled goals. We report the mean performance along with the standard deviations of policies trained across 8 different random seeds, as summarized in Tables 1 and 2. Details of the hyperparameters used in the experiments can be found in Appendix C.

5.1. Results on Meta-World

The results for two benchmarks, MT10 and MT50, are presented in Table 1 and Table 2, respectively. These findings highlight the effectiveness of the ARS algorithm's innovative strategy for multi-task RL, especially its adaptive reward scaling and reset mechanism. Together, these components drive notable performance improvements across a wide range of tasks in the Meta-World benchmark.

Results on MT10. As shown in the MT10 results in Table 1, our ARS framework with three variants achieves outstanding performance, attaining average success rates of 97.3%, 98.6%, and 99.5%, respectively, which are the highest among all compared methods. Furthermore, ARS-LoRA achieves the best overall performance with a 99.5%

Table 2. Comparison of average bottom-k success rates on the Meta-World MT50 benchmark.

	Average Bottom-k Success Ratio (%)				
Algorithm	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 50$
SAC-MT	0.0 ± 0.0	0.0 ± 0.0	14.1 ± 3.1	34.5 ± 2.5	47.6 ± 2.0
SAC-MT-MH	0.0 ± 0.0	0.0 ± 0.0	14.1 ± 1.7	34.0 ± 2.0	47.2 ± 1.7
Soft Modular	0.0 ± 0.0	1.8 ± 3.7	23.7 ± 12.3	42.6 ± 9.5	54.1 ± 7.6
PCGrad	0.0 ± 0.0	0.0 ± 0.0	21.0 ± 12.9	39.9 ± 10.7	51.9 ± 8.5
PaCo	0.0 ± 0.0	4.6 ± 8.2	26.1 ± 15.0	44.6 ± 11.2	55.6 ± 9.1
MOORE	0.0 ± 0.0	15.9 ± 7.0	41.2 ± 5.0	55.3 ± 3.9	64.2 ± 3.1
SMT	0.0 ± 0.0	8.0 ± 8.9	26.8 ± 13.1	45.0 ± 9.9	56.0 ± 8.0
ARS (Ours)	9.1 ± 6.6	26.3 ± 6.1	45.3 ± 4.7	57.5 ± 3.6	65.9 ± 2.9
ARS-LN (Ours)	21.0 ± 11.4	49.1 ± 6.5	64.3 ± 6.2	72.9 ± 4.7	78.3 ± 3.7
ARS-LoRA (Ours)	**29.3 ± 7.7**	**58.4 ± 4.5**	**72.0 ± 3.0**	**79.0 ± 2.2**	**83.2 ± 1.8**

success rate, successfully solving all tasks with at least 97% accuracy. This marks a significant advancement over existing methods and a major breakthrough as the first to solve the MT10 benchmark using scratch multi-task RL training. Notably, ARS framework performs exceptionally well on challenging tasks, such as task ID 1 ('push'), 2 ('pick-place'), and 7 ('peg-insert-side'). These results highlight the effectiveness of ARS's adaptive reward scaling combined with the reset mechanism, which emphasizes addressing difficult tasks and accelerating progress during the early stages of training.

Results on MT50. The MT50 results in Table 2 highlight the consistent effectiveness of ARS, which outperforms other methods, particularly in the average bottom-k success ratios, as suggested by Cho et al. (2024). In particular, in the most challenging subset of tasks (bottom-10), where all baseline methods fail, ARS variants achieve average success rates of 9.1%, 21.0%, and 29.3%, respectively—demonstrating a clear advantage on difficult tasks. Moreover, the use of Layer Normalization and LoRA fine-tuning remarkably improves performance from 65.9% to

Table 3. Results of incorporating the ARS framework into existing multi-task RL methods.

		Multi-Task Algorithm			
Benchmark	ARS	SAC-MT	PCGrad	Soft Modular	MOORE
MT10	w/ ARS	97.3±2.1	95.9±2.6	**98.8±1.3**	98.0±0.4
	w/o ARS	69.4±0.8	74.8±13.7	74.4±10.5	86.0±4.8
MT50	w/ ARS	65.9±2.9	72.4±2.6	75.5±3.6	**84.6±1.6**
	w/o ARS	47.6±2.0	51.9±8.5	54.1±7.6	64.2±3.1

78.3% and 83.2%, respectively, in terms of average success rate across all tasks. This marks a significant advancement over existing methods and represents a major milestone as the first instance of successfully solving diverse tasks in the MT50 benchmark through scratch multi-task RL training. Note that while the performance of MOORE (Hendawy et al., 2024) appears competitive with our basic ARS's performance, this is not a fair comparison, as the MOORE method uses approximately 4× more parameters than ours. To examine the effects of network capacity, we conducted an ablation study on the number of parameters in Section 5.4. With a bigger network, ARS-LN achieves **88.9%** average success rate for the MT50 benchmark. Notably, ARS-LN successfully solves all tasks with a success rate of at least 45%, except for two tasks: assembly and disassemble, resulting in an average success rate of 92.3% across 48 tasks.

To further investigate the advantages of mastering diverse tasks, we analyze the Effective Solvable Task Ratio (ESTR) on the MT50 benchmark. The ESTR is defined as:

$$ESTR = \frac{\left|\{\mathcal{T}_i \in \mathcal{C} \mid \text{EvalFinalSR}(\mathcal{T}_i) \geq \delta\}\right|}{N} \tag{8}$$

where δ is the success ratio threshold, and $\text{EvalFinalSR}(\mathcal{T}_i)$ represents the final average success ratio for a task $\mathcal{T}_i$. A higher ESTR indicates that the agent successfully solves more tasks. We compute the ESTR for threshold values $\delta \in \{0.1, 0.3, 0.5, 0.7\}$. The threshold δ specifies the criteria for a task to be considered solvable, with tasks achieving an average success ratio equal to or above δ classified as solvable. As δ increases, the criteria become stricter, causing the ESTR metric to decrease accordingly.

As shown in the ESTR results in Figure 4, our ARS framework demonstrates exceptional performance across all threshold values $\delta \in \{0.1, 0.3, 0.5, 0.7\}$, indicating that ARS successfully solves more tasks than other baselines. Notably, ARS significantly outperforms competing methods as δ decreases, highlighting its ability to address a broader range of tasks, including those with lower success ratios. In contrast, the baseline methods tend to focus on only a subset of tasks, failing to tackle more challenging ones effectively.

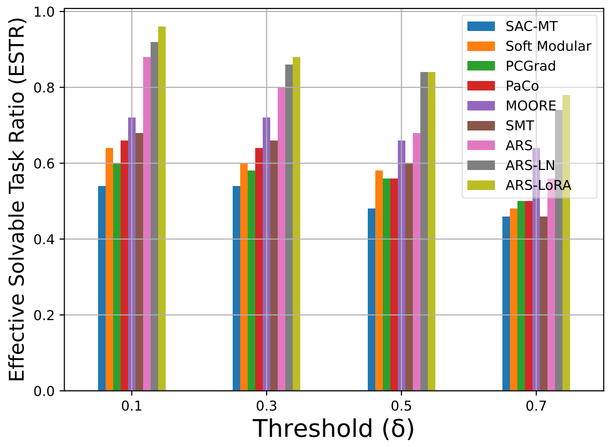


Figure 4. Comparison of the Effective Solvable Task Ratio (ESTR) with respect to the threshold δ in the MT50 benchmark.

5.2. Analysis of the Reward Scaling Approach

In this section, we assess the effectiveness of the proposed reward scaling method by analyzing changes in the reward scaling factor c_i^{rew} for each task $\mathcal{T}_i$. Figure 5 shows the learning curve of the reward scale factors $\{c_i^{\text{rew}}\}$ on a logarithmic scale for the MT10 benchmark. Initially, the agent assigns higher scaling factors to three difficult tasks including 'push,' 'pick-place,' and 'peg-insert-side' due to their low initial reward magnitudes. Over time, these factors gradually become more uniform, as shown in Figure 5. This indicates that as tasks with high scaling factors are successfully learned, their mean rewards increase, leading to a reduction in the scaling factor.

5.3. Integration of the ARS Framework with Off-Policy Multi-Task RL Methods

The proposed ARS framework is a universal solution that can be applied to any off-policy multi-task reinforcement learning (RL) method. To evaluate its applicability, we incorporated the dynamic reward scaling and reset mechanisms of ARS into various off-policy multi-task approaches, including PCGrad, Soft Modular, and MOORE.

Table 3 presents the performance of these algorithms combined with the ARS framework (training curves are provided in Appendix D.3). The results demonstrate that the ARS framework consistently improves all evaluated multi-task RL algorithms on both the MT10 and MT50 benchmarks, highlighting its broad compatibility and effectiveness. In particular, the integration of ARS boosts the average success rate by at least 20% across all off-policy MTRL methods on MT50. Notably, MOORE (Hendawy et al., 2024) integrated with ARS achieves the highest average success rate of 84.6% on MT50 benchmark, a performance level never

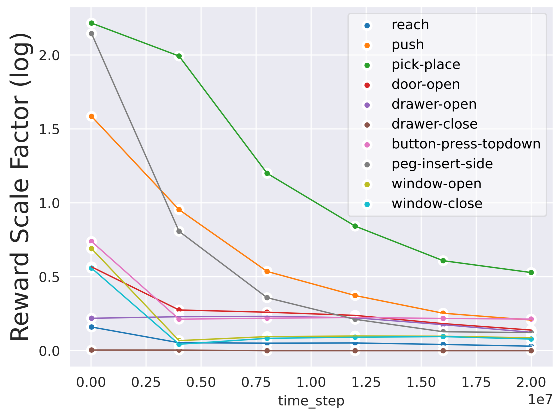


Figure 5. Learning curve of the reward scale factor c_i^{rew} on a logarithmic scale for each task $\mathcal{T}_i$ within the MT10 benchmark. Complex tasks such as 'pick-place,' 'push,' and 'peg-insert-side' have larger reward scaling factors compared to others.

before reached by other multi-task RL approaches.

5.4. Ablation Studies

First, We conducted ablation studies to evaluate the key components of ARS: (1) the adaptive reward scaling scheme and (2) the reset mechanism. To assess their contributions, we performed experiments where each component was omitted individually, denoted using the notation "w/o." Specifically, the 'ARS w/o reward scaling' corresponds to the naive resetting framework proposed by (Nikishin et al., 2022).

Table 4 shows the average success ratio for the task set configurations in the MT10 benchmark (training curves provided in Appendix D.2)). Here, $\mathcal{C}_{\text{easy}}$ represents the set of seven tasks with the highest success ratios, while the remaining tasks constitute the difficult set, $\mathcal{C}_{\text{difficult}}$. Removing the reset mechanism or reward scaling results in performance drops of 27.5% and 27.0%, respectively. The absence of the reset mechanism significantly reduces the performance of the easy task set, underscoring its importance for stable training. On the other hand, reward scaling notably enhances the performance of the difficult task set, demonstrating its effectiveness in addressing varying reward distributions and ensuring balanced learning across tasks. These observations highlight the significance of both components in the effectiveness of the ARS framework.

Next, to verify the effectiveness of the proposed reward scaling method, we conducted experiments using alternative scaling approaches including mean reward, standard deviation of reward, and normalization. The scaled rewards based on standard deviation ($\hat{r}_i^{\sigma}$) and normalization ($\hat{r}_i^{\text{normal}}$) for each task $\mathcal{T}_i$ are defined as $\hat{r}_i^{\sigma} = \frac{r}{\sigma(r_i)}$ and $\hat{r}_i^{\text{normal}} = \frac{r-\bar{r}_i}{\sigma(r_i)}$,

Table 4. Ablation study on the components of ARS.

	Algorithm		
Task set	ARS w/o reset	ARS w/o reward scaling	ARS
$\mathcal{C}_{\text{easy}}$	85.9±11.0	**100±0.0**	99.6±1.0
$\mathcal{C}_{\text{difficult}}$	32.1±15.6	0.8±1.5	**91.7±7.3**
$\mathcal{C}$	69.8 ±8.1	70.3 ± 0.5	**97.3±2.1**

Table 5. Ablation study on the reward scaling method.

	Reward Scaling Method		
Task set	Normalization	Standard Deviation	Mean Reward(Ours)
$\mathcal{C}_{\text{easy}}$	85.9 ± 26.0	99.1±2.0	**99.6±1.0**
$\mathcal{C}_{\text{difficult}}$	75.0 ± 33.3	82.9±22.0	**91.7±7.3**
$\mathcal{C}$	82.4 ± 27.9	94.3±7.1	**97.3±2.1**

respectively. Table 5 presents the average success ratio for each scaling metric. The results show that scaled-up based on the mean rewards consistently outperforms other approaches, highlighting the effectiveness of our method.

Finally, we conducted an ablation study to investigate the effect of network capacity. While the MOORE (Hendawy et al., 2024) model integrated with ARS achieves the highest performance on the MT50 benchmark among multi-task methods, a key contributing factor is its larger network size. Each network in MOORE contains approximately 2.0M parameters, whereas our main result in Table 2 uses only 0.5M. To assess the impact of model size, we increased the hidden unit size in our architecture to 512, 800, and 1024. The configuration with 800 hidden units has a parameter count comparable to that of the MOORE model. As shown in Table 6, the performance of ARS improves consistently with increased capacity, whereas SAC-MT (w/o ARS) does not. This highlights ARS as a scalable and effective method for leveraging larger models through reset and dynamic reward scaling. Remarkably, ARS-LN with 1024 hidden units achieves the highest average success rate of 88.9% across all evaluated MTRL methods. Furthermore, despite having similar network capacity, ARS-LN with 800 hidden units outperforms the MOORE model integrated with ARS. This performance gain demonstrates the effectiveness of the layer normalization technique, even though its adoption is significantly simpler compared to the complex model design of MOORE. Additional ablation studies are provided in Appendix E.

6. Related Works

Multi-task RL. Multi-task learning has become a key area in machine learning, focusing on algorithms that perform well across various tasks. In the context of RL, it aims

Table 6. Ablation study on the network capacity.

	Network Capacity: Hidden Layer Dimensions			
Algorithm	$[400, 400, 400, 400]$	$[512, 512, 512, 512]$	$[800, 800, 800, 800]$	$[1024, 1024, 1024, 1024]$
ARS-LN	78.3±3.7	84.2±1.7	88.0±2.9	**88.9±2.6**
ARS	65.9±2.9	73.6±4.4	75.1±3.8	78.7±2.9
SAC-MT	47.6±2.0	48.8±2.6	47.2±1.7	45.5±2.4

to develop models capable of handling diverse tasks (Wilson et al., 2007; Pinto & Gupta, 2017; Zeng et al., 2018; Hausman et al., 2018).

Various approach for Multi-taks RL Addressing negative transfer is crucial for the success of multi-task RL, and various strategies have been developed to tackle this issue. (i) Distillation methods leverage policy distillation to merge knowledge from multiple tasks into a unified model but often require task-specific networks, increasing resource demands(Rusu et al., 2016a; Parisotto et al., 2016; Teh et al., 2017). (ii) Modular networks assign distinct modules to tasks with task-specific routing, enabling strategic parameter sharing to reduce interference and negative transfer (Rusu et al., 2016b; Devin et al., 2017; Andreas et al., 2017; Haarnoja et al., 2018a; Yang et al., 2020; Sun et al., 2022; Hendawy et al., 2024). (iii) Gradient-based approaches analyze task gradients to address conflicts but face challenges due to gradient noise and variability (Zhang & Yeung, 2013; Chen et al., 2018; Hu et al., 2019; Yu et al., 2020). (iv) Explicit policy-sharing methods share behaviors or policies across tasks to obtain good samples from different tasks, enabling knowledge transfer without sharing parameters (Zhang et al., 2023; He et al., 2024). (v) Task scheduling methods use a scheduling framework to prioritize and train more effective tasks earlier in the training process of diverse tasks. (Cho et al., 2024).

In contrast, our method focuses on the reward function. The varying reward distributions make training challenging, so we introduce an adaptive reward scaling scheme to enable stable and efficient training in multi-task RL.

Resetting Deep RL Agent In deep RL, primacy bias, or overfitting to early experiences, has been addressed through periodic resets of agent parameters while retaining the replay buffer (Nikishin et al., 2022). Extending this concept, D'Oro et al. (2023) Subsequent advancements include reset frequency modulation (D'Oro et al., 2023), ensemble-based resets (Kim et al., 2023), and refined mechanisms achieving human-level Atari performance (Schwarzer et al., 2023).

Our method extends these ideas to a multi-task RL setting, where resets not only address primacy bias but also mitigate biases toward tasks learned earlier in training.

Reward Scaling in Deep RL Early studies showed that reward scaling is one of the key factors for stabilizing training: Henderson et al. (2018) demonstrated that SAC and DDPG are sensitive to this choice, while Wu et al. (2018) proposed *Automatic Reward Scaling* (ANS), which adjusts a global factor based on return statistics. In multi-task RL, PopArt (Hessel et al., 2019) introduced *scale-invariant updates* by normalizing critic targets through an adaptive affine transformation, enabling training across 57 Atari games with different reward scales. However, PopArt only normalizes the value head and overlooks the broader impact of reward scaling on deep RL training, which remains critical for solving difficult tasks.

Our *ARS* differs in that it equalizes the *mean* reward across tasks using a simple, parameter-free rule based on replay buffer statistics (Eq. 6), automatically assigning higher scaling factors to harder tasks. As demonstrated in Sections 5.1–5.2 and E.3, this lightweight framework consistently outperforms PopArt-style and other normalization methods on MT10 and MT50.

7. Conclusion

In this work, we introduced Adaptive Reward Scaling (ARS), a novel framework designed to tackle the critical challenges caused by varying reward distributions. By employing a history-based reward scaling strategy, ARS dynamically adjusts reward magnitudes to balance training focus across diverse tasks. The integration of a reset mechanism further enhances ARS by mitigating biases introduced by early learned tasks, ensuring improved adaptability and convergence. Together, these innovations enable ARS to achieve state-of-the-art performance on challenging benchmarks such as Meta-World, demonstrating its effectiveness in handling diverse and complex task sets. Future work could extend ARS to other multi-task RL algorithms and more diverse domains, as well as refining its reward scaling and reset strategies for greater efficiency. Investigating alternative approaches to task representation and value normalization, alongside scaling ARS to multi-agent or hierarchical RL frameworks, presents exciting directions for further research. By addressing key limitations in multi-task RL, ARS contributes significantly to advancing the field, paving the way for more scalable and robust solutions to real-world problems.

Impact Statement

A major challenge in reinforcement learning (RL) is generalization, where a policy trained for one task often struggles to perform effectively on different tasks. Addressing this issue is crucial for training policies that can handle multiple related tasks effectively. In this paper, we introduced a novel reward scaling framework for multi-task RL that improves overall performance. Our approach contributes to the advancement of RL in real-world applications requiring the ability to solve multiple similar tasks.

Acknowledgments

This work was supported in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2022-II220124, Development of Artificial Intelligence Technology for Self-Improving Competency-Aware Learning Capabilities, 50%) and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2022-II220469, Development of Core Technologies for Task-oriented Reinforcement Learning for Commercialization of Autonomous Drones, 50%)

References

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 166–175. PMLR, 2017.

Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Ball, P. J., Smith, L. M., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1577–1594. PMLR, 2023.

Caruana, R. Multitask learning. *Mach. Learn.*, 28(1):41–75, 1997.

Cetin, E., Ball, P. J., Roberts, S. J., and Çeliktutan, O. Stabilizing off-policy deep reinforcement learning from pixels. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2784–2810. PMLR, 2022.

Chen, Z., Badrinarayanan, V., Lee, C., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 793–802. PMLR, 2018.

Cho, M., Park, J., Lee, S., and Sung, Y. Hard tasks first: Multi-task reinforcement learning through task scheduling. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pp. 2169–2176. IEEE, 2017.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P., Bellemare, M. G., and Courville, A. C. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 6244–6251. IEEE, 2018a.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018b.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. In *9th International Conference on Learning Representations, ICLR 2021,* *Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. A. Learning an embedding space for transferable robot skills. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

He, J., Li, K., Zang, Y., Fu, H., Fu, Q., Xing, J., and Cheng, J. Efficient multi-task reinforcement learning with cross-task policy guidance. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Hendawy, A., Peters, J., and D'Eramo, C. Multi-task reinforcement learning with mixture of orthogonal experts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3207–3214. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11694.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task deep reinforcement learning with popart. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 3796–3803. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33013796.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Hu, H., Dey, D., Hebert, M., and Bagnell, J. A. Learning anytime predictions in neural networks via adaptive loss balancing. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 3812–3821. AAAI Press, 2019.

Kapturowski, S., Campos, V., Jiang, R., Rakicevic, N., van Hasselt, H., Blundell, C., and Badia, A. P. Human-level atari 200x faster. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. Champion-level drone racing using deep reinforcement learning. *Nat.*, 620(7976):982–987, 2023. doi: 10.1038/S41586-023-06419-4.

Kim, W., Shin, Y., Park, J., and Sung, Y. Sample-efficient and safe deep reinforcement learning via reset deep ensemble agents. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10-16, 2023*, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P., and Courville, A. C. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847. PMLR, 2022.

Parisotto, E., Ba, L. J., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Pinto, L. and Gupta, A. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pp. 2161–2168. IEEE, 2017.

Rusu, A. A., Colmenarejo, S. G., Gülçehre, Ç., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016a.

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

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Sun, L., Zhang, H., Xu, W., and Tomizuka, M. Paco: Parameter-compositional multi-task reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21495–21507, 2022.

Sun, X., Panda, R., Feris, R., and Saenko, K. Adashare: Learning what to share for efficient deep multi-task learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Martín-Martín, R., and Stone, P. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8, 2024.

Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4496–4506, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 5026–5033. IEEE, 2012.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pp. 1015–1022. ACM, 2007.

Wu, Y.-H., Sun, F.-Y., Chang, Y.-Y., and Lin, S.-D. Ans: adaptive network scaling for deep rectifier reinforcement learning models. *arXiv preprint arXiv:1809.02112*, 2018.

Yang, R., Xu, H., Wu, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2019.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., and Funkhouser, T. A. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pp. 4238–4245. IEEE, 2018.

Zhang, G., Jain, A., Hwang, I., Sun, S., and Lim, J. J. Efficient multi-task reinforcement learning via selective behavior sharing. *CoRR*, abs/2302.00671, 2023. doi: 10.48550/ARXIV.2302.00671.

Zhang, Y. and Yeung, D. A regularization approach to learning task relationships in multitask learning. *ACM Trans. Knowl. Discov. Data*, 8(3):12:1–12:31, 2013.

A. MT10 and MT50 Environments

To evaluate multi-task reinforcement learning methods, we adopt the robotic manipulation tasks provided by the Meta-World benchmark suite (Yu et al., 2019). Meta-World contains 50 distinct tasks and defines two standard benchmarks for multi-task RL: MT10 and MT50, comprising 10 and 50 tasks, respectively. The MT10 benchmark consists of the following 10 tasks with their corresponding task ID numbers: (1) reach, (2) push, (3) pick-place, (4) door-open, (5) drawer-open, (6) drawer-close, (7) button-press-topdown, (8) peg-insert-side, (9) window-open, and (10) window-close.

Performance is measured using the success rate per task. An episode is considered successful if the agent completes the task at least once during the episode, following common evaluation protocols in prior work (Yu et al., 2020; Haarnoja et al., 2018b; Sun et al., 2022; Hendawy et al., 2024; Cho et al., 2024). In addition, we adopt modified versions of MT10 and MT50 as our default evaluation setting, where each task is trained with randomly sampled goal positions and evaluated across 10 randomized goal configurations, consistent with prior studies (Yang et al., 2020; Sun et al., 2022; Hendawy et al., 2024).

B. Low-Rank Adaptation for Multi-Task RL

Low-Rank Adaptation (LoRA) (Hu et al., 2022) injects a learnable, low-rank update into a frozen weight matrix so that only a small number of additional parameters are trained while the original backbone remains intact. In our experiments we attach LoRA adapters to the *actor* and *critic* multilayer perceptrons (MLPs) used in ARS-LN.

Parameterisation. For any linear layer with weights $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ we introduce two trainable matrices $A \in \mathbb{R}^{d_{\text{out}} \times r}$ and $B \in \mathbb{R}^{r \times d_{\text{in}}}$ of rank $r \ll \min(d_{\text{out}}, d_{\text{in}})$. The effective weight becomes

$$W = W_0 + \alpha \frac{AB}{r}, \tag{9}$$

where α is a scaling constant set to the LoRA rank (8 for MT10, 16 for MT50). Gradients flow only through A and B; W_0 is frozen throughout.

Adapter placement. We wrap the input projection and all hidden linear layers of both the actor and critic with LoRA. The final output heads are left untouched to preserve output variance accumulated during pre-LoRA training.

Activation schedule. To avoid destabilizing early training, LoRA adapters are *disabled* (i.e. $A = B = 0$) for the first $0.75 \times T$ training steps on MT10 and $0.833 \times T$ steps on MT50, where T is the total number of gradient updates reported in Table 8. From that point onwards the adapters are enabled and trained jointly with the actor and critic optimizer.

Parameter overhead. The additional learnable parameters introduced by LoRA are $r(d_{\text{out}} + d_{\text{in}})$ per adapted layer. With $r = 8$ (MT10) and $r = 16$ (MT50) this overhead amounts to less than 3% of the original network size, providing efficient task-specific specialization without significant memory or compute cost.

C. Hyperparameters

In this section, we provide the hyperparameters for ARS used in the MT10 and MT50 experiments in Table 7, along with some general hyperparameters used across ARS and the baselines in Table 8. Specifically, for the Soft Modular method, we adopt the deep structure proposed in (Yang et al., 2020), consisting of 4 layers with 4 modules and a hidden unit size of 128.

Table 7. ARS specific hyperparameters.

Hyperparameters	**MT10**	**MT50**
number of Reset	4	6

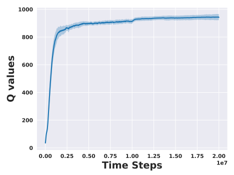
Table 8. General Multi-task RL hyperparameters.

Hyperparameters	**MT10**	**MT50**
training steps	2×10^7	1×10^8
number of reset (n_{reset})	4	6
replay buffer size per task	1×10^6	5×10^5
episode length	500	
optimizer	Adam (Kingma & Ba, 2015)	
batch size per task	100	
learning rate (all networks)	3e-4	
activation for critic	Tanh	
activation for actor	ReLU	
discount factor (γ)	0.99	
MLP hidden layer size	[400, 400, 400, 400]	
target network update period	1	
tau(τ)	5e-3	

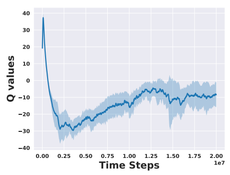
D. Additional Results

D.1. Learning Curves of Q-Values with the SAC-MT Algorithm

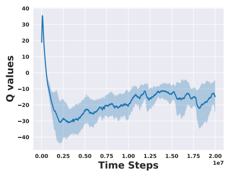
In addition to Figure 3 in Section 3.2, we plot additional learning curves of Q-values for all tasks in the MT10 benchmark. As shown in Figure 6, the training of the Q-network struggles with the more challenging tasks—'push,' 'pick-place,' and 'peg-insert-side.' where their Q-values remain negative.



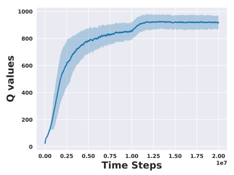
(a) reach



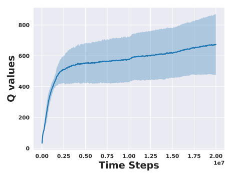
(b) push



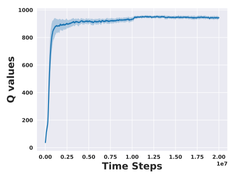
(c) pick-place



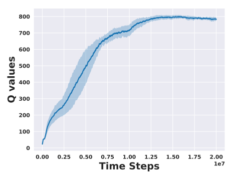
(d) door-open



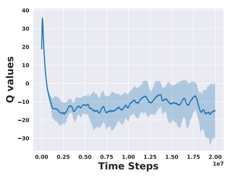
(e) drawer-open



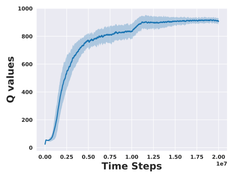
(f) drawer-close



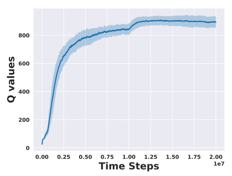
(g) button-press-topdown



(h) peg-insert-side



(i) window-open

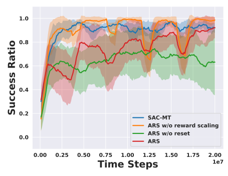


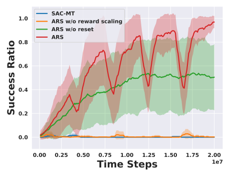
(j) window-close

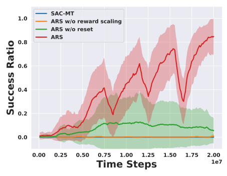
Figure 6. Learning curves of Q-values with SAC-MT for all tasks in MT10 benchmark

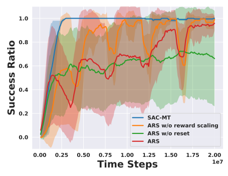
D.2. Learning Curves of Average Success Ratios on MT10 Based on Key ARS Components

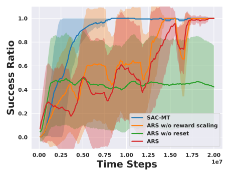
Figures 7 and 8 show the average success rates of ARS and its two variants (without reward scaling and without reset) in MT10 experiments. 'ARS w/o reset' exhibits greater variation and instability due to biases from amplified rewards, whereas 'ARS w/o reward scaling' is more stable but underperforms on challenging tasks, such as 'push,' 'pick-place,' and 'peg-insert-side,' due to low reward magnitudes, as shown in Figure 7. The full ARS outperforms both variants, achieving improved stability and success rates, with at least 0.93 average success across all random seeds.

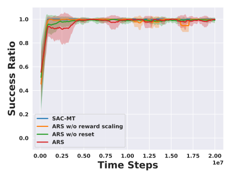

(a) reach

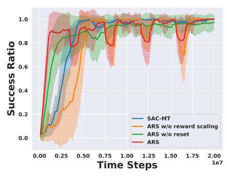

(b) push

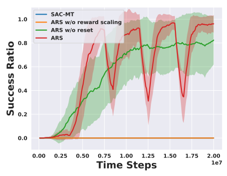

(c) pick-place

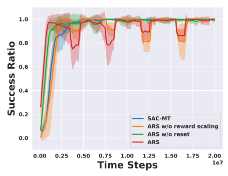

(d) door-open


(e) drawer-open


(f) drawer-close


(g) button-press-topdown


(h) peg-insert-side


(i) window-open

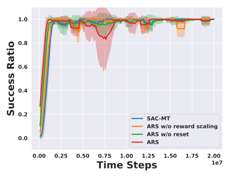

(j) window-close

Figure 7. Learning curves of the success ratio averaged over all tasks in the MT10 benchmark, based on the key ARS components: (1) the adaptive reward scaling scheme and (2) the reset mechanism.

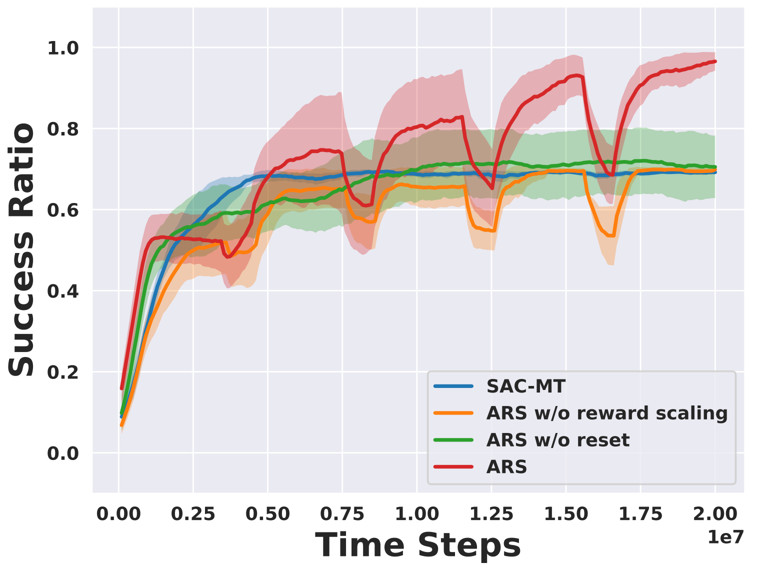


Figure 8. Learning curves of average success ratios across tasks in the MT10 benchmark, based on the key ARS components: (1) the adaptive reward scaling scheme and (2) the reset mechanism.

D.3. Learning Curves of Average Success Ratios on MT50

Figure 9 shows that all off-policy multi-task RL methods incorporating our ARS framework surpass the previous state-of-the-art (SOTA) performance (Cho et al., 2024) on the MT50 benchmark, indicated by the dotted line. Even the simplest method, SAC-MT, achieves exceptional results, emphasizing the remarkable effectiveness of the proposed framework. Furthermore, our approach achieves at least the previous SOTA performance using only half the samples, i.e., 50 million.

Figure 10 show that the performance of ARS improves with capacity, whereas SAC-MT (w/o ARS) does not. This confirms ARS as an effective approach for scaling model size and improving performance via reset and dynamic reward scaling.

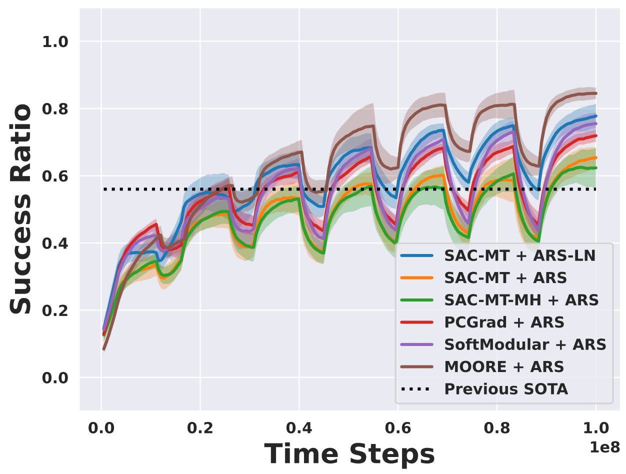


Figure 9. Learning curves of average success ratios across tasks in the MT50 benchmark using off-policy multi-task RL methods incorporating the ARS framework. The dashed line indicates the previous state-of-the-art performance (Cho et al., 2024).

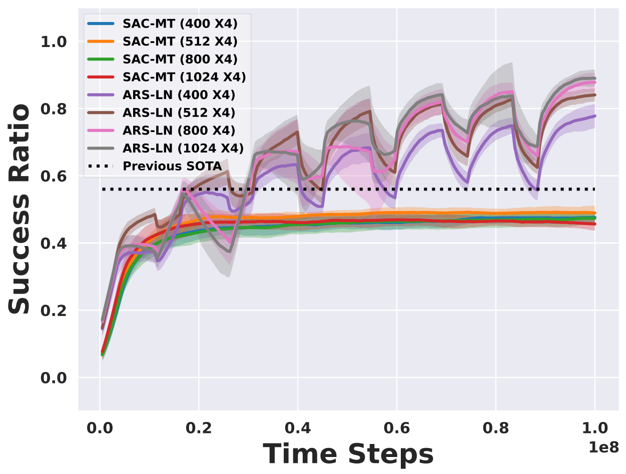


Figure 10. Learning curves of average success ratios across tasks in the MT50 benchmark based on the network capacity.

E. Additional Ablation Studies

E.1. Ablation Study on the Hyperparameter n_{reset}

Incorporating our ARS framework into other off-policy multi-task RL algorithms introduces a unique hyperparameter: the number of resets (n_{reset}). To evaluate its impact, we conducted ablation studies on n_{reset}. Figure 11 shows the average success ratios for different n_{reset} values. The results show that performance remains robust across various values, though lower n_{reset} generally achieves better performance. This trend is likely due to the reduced frequency of model updates with higher n_{reset}, resulting in insufficient updates.

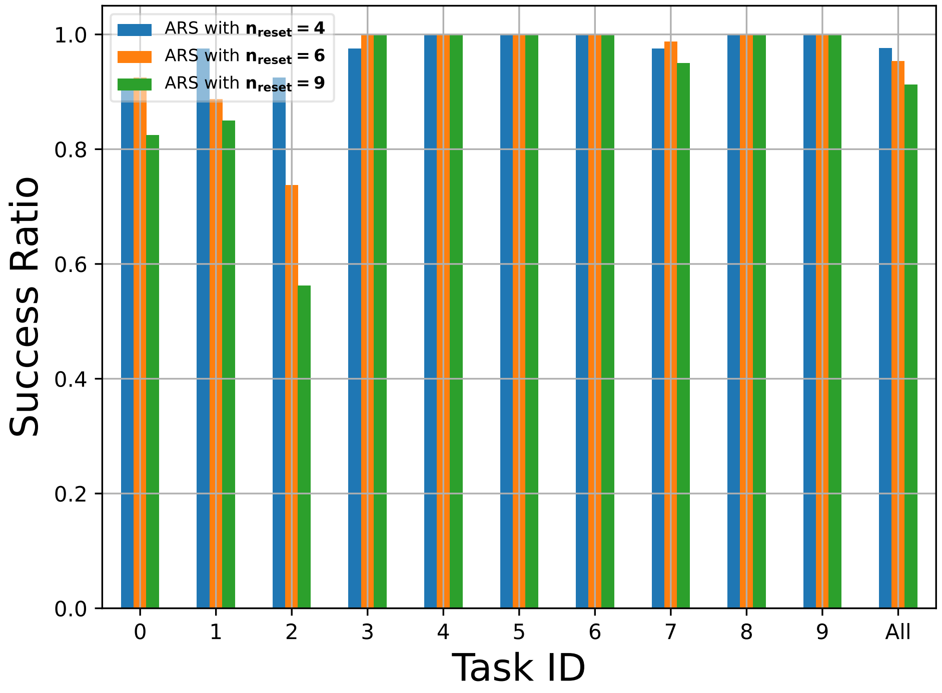


Figure 11. Comparison of the final success rate per task in the MT10 benchmark based on the number of resets (n_{reset}).

E.2. Ablation Study on the Presence of Reset Mechanisms in Actor and Critic Networks

In this subsection, we conduct an ablation study to analyze the impact of reset mechanisms in the actor and critic networks. Table 9 shows the average success ratio based on the reset strategy, showing the advantage of resetting both the actor and critic networks.

Table 9. Ablation study on the Reset Mechansim

	Reset Strategy		
Benchmark	No reset	Reset Critic Only	Reset Actor & Critic
MT10	69.4±0.8	96.3±2.3	**97.3±2.1**

E.3. Comparison with PopArt

Prior work, PopArt (Hessel et al., 2019), identifies the issue of varying reward scales and addresses it by introducing scale-invariant updates, which normalize critic targets using an adaptive affine layer. To evaluate the effectiveness of our proposed adaptive reward scaling (ARS) method against PopArt's scale-invariant updates, we integrate the latter into the SAC-MT algorithm and compare it with the ARS framework. We vary the update frequency of the scale-invariant updates over $1, 10, 100, 500$. As shown in Table 10, our ARS consistently outperforms PopArt on the MT10 benchmark across all frequencies, demonstrating the effectiveness of the ARS framework.

Table 10. Results of average ratio (%) on Meta-World MT10 for PopArt Variants

	Update Frequency for PopArt				
	1	10	100	500	ARS
MT10	67.1 ± 12.8	73.3 ± 7.6	75.0 ± 6.3	71.5 ± 7.8	**97.3 ± 2.1**

E.4. Ablation Study on the Horizon Length

In the default setup of the MetaWorld-v2 (Yu et al., 2019) environments, the horizon length is set to 500, which was used consistently across all our experiments. However, we noticed that the experiments reported in the PaCo and MOORE papers (Sun et al., 2022; Hendawy et al., 2024) were conducted using a horizon length of 150, making a direct comparison between the reported results and our ARS method inappropriate. To address this, we reported the results of PaCo and MOORE with the horizon length set to 500 in the main results (Table 1 & 2 in Section 5.1).

In addition to the results with a horizon length of 500, we compare the proposed method, ARS-LN, with two baselines (PaCo and MOORE) on both the MT10 and MT50 benchmarks using a shorter horizon length of 150. As shown in Table 11, ARS-LN significantly outperforms both baselines, achieving the highest average success rates of ***98.4%*** on MT10 and ***91.0%*** on MT50, respectively. Notably, the scaling law observed with increased network capacity (as described in Section 5.4) still holds in the horizon-150 setting. Moreover, ARS-LN even achieves slightly higher performance with a horizon length of 150 compared to the 500-horizon setting.

Table 11. Comparisons of average ratio (%) of the Meta-World MT10 and MT50 benchmark with Horizon Legnth 150

	ARS-LN (Hidden Layer Dimensions)				
	$[400, 400, 400, 400]$	$[800, 800, 800, 800]$	$[1024, 1024, 1024, 1024]$	PaCo	MOORE
MT10	**98.4 ± 1.5**	-	-	85.4±4.5	88.7 ± 5.6
MT50	83.3±4.3	89.8±1.6	**91.0±2.4**	57.3±1.3	72.9±3.3

F. Overview of the ARS framework

This section provides an overview of the proposed ARS framework. The orange box highlights its key components, including history-based reward scaling, supported by the reset mechanism (green box). It also illustrates that ARS can be seamlessly integrated into any off-policy multi-task RL algorithm with minimal effort.

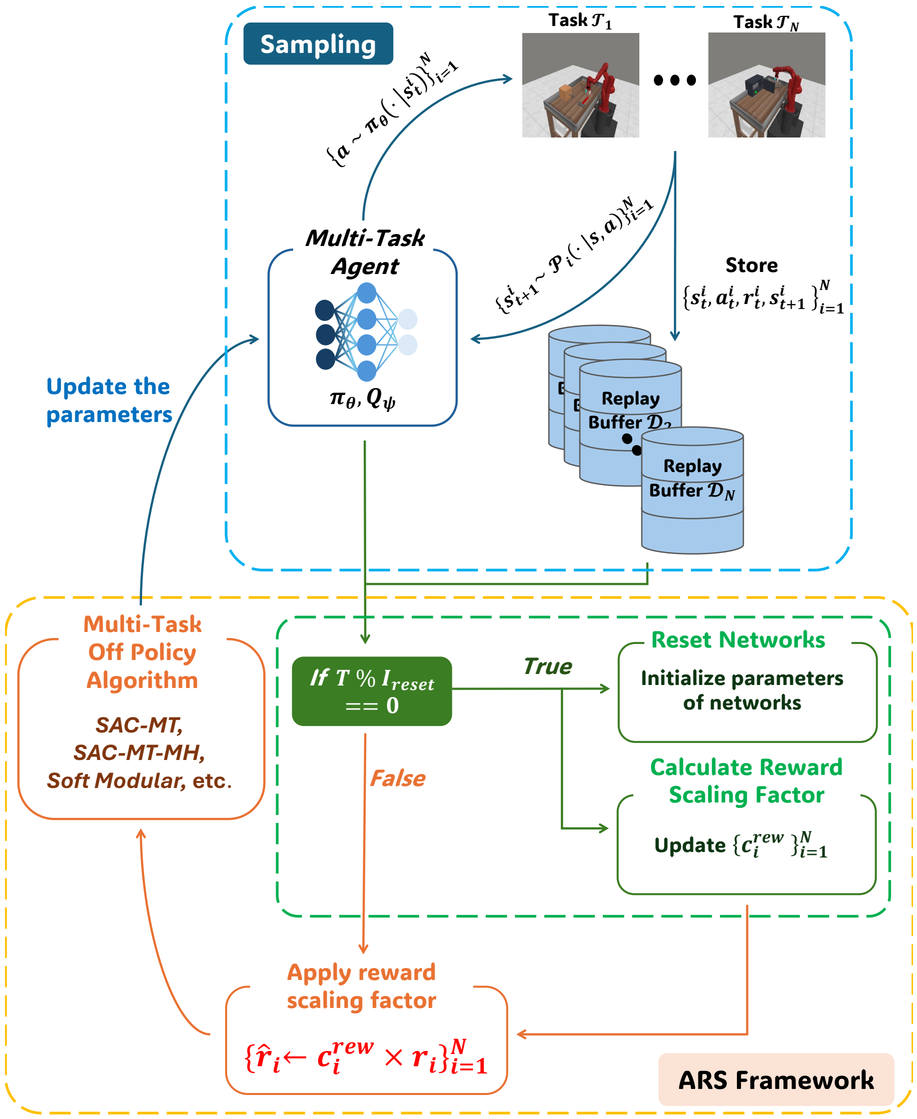

Sampling
Task $\mathcal{T}_1$
Task $\mathcal{T}_N$
$\{a \sim \pi_\theta(\cdot \mid s_t^i)\}_{i=1}^N$
Multi-Task Agent
π_θ, Q_ψ
$\{s_{t+1}^i \sim \mathcal{P}_i(\cdot \mid s, a)\}_{i=1}^N$
Store
$\{s_t^i, a_t^i, r_t^i, s_{t+1}^i\}_{i=1}^N$
Replay Buffer $\mathcal{D}_2$
Replay Buffer $\mathcal{D}_N$
Update the parameters
Multi-Task Off Policy Algorithm
SAC-MT, SAC-MT-MH, Soft Modular, etc.
If $T \,\%\, I_{reset} == 0$
True
False
Reset Networks
Initialize parameters of networks
Calculate Reward Scaling Factor
Update $\{c_i^{rew}\}_{i=1}^N$
Apply reward scaling factor
$\{\hat{r}_i \leftarrow c_i^{rew} \times r_i\}_{i=1}^N$
ARS Framework

Figure 12. Overview of the ARS framework. The orange box illustrates the history-based reward scaling, supported by a reset mechanism.