It is All Connected: Multi-Task Reinforcement Learning via Mode Connectivity

Ahmed Hendawy[1,2*], **Henrik Metternich**[1], **Jan Peters**[1,2,3,4], **Gabriele Tiboni**[1,5], **Carlo D'Eramo**[1,2,5]

[1]Department of Computer Science, TU Darmstadt, Germany
[2]Hessian Center for Artificial Intelligence (Hessian.ai), Germany
[3]Center for Cognitive Science, TU Darmstadt, Germany
[4]German Research Center for AI (DFKI), Systems AI for Robot Learning
[5]Center for Artificial Intelligence and Data Science, University of Würzburg, Germany

Abstract

Acquiring a universal policy that performs multiple tasks is a crucial building block in endowing agents with generalized capabilities. To this end, the field of Multi-Task Reinforcement Learning (MTRL) proposes sharing parameters and representations among tasks during the learning process. Still, optimizing for a single solution that is able to perform various skills remains challenging. Recent works attempt to address these challenges using a mixture of experts, though this comes at the cost of additional inference-time complexity. In this paper, we introduce STAR, a novel MTRL algorithm that leverages mode connectivity to share knowledge across single skills, while remaining parameter-efficient at deployment time. Particularly, we show that single-task policies can be linearly connected in policy parameter space to the multi-task policy, i.e., task performance is maintained throughout the linear path connecting the two policies. Our experimental evaluation demonstrates that mode connectivity at training time induces implicit regularization in the multi-task policy, surpassing related baselines on MTRL benchmarks MuJoCo and Metaworld. Furthermore, STAR achieves competitive performance even with methods that retain multiple models at inference time.

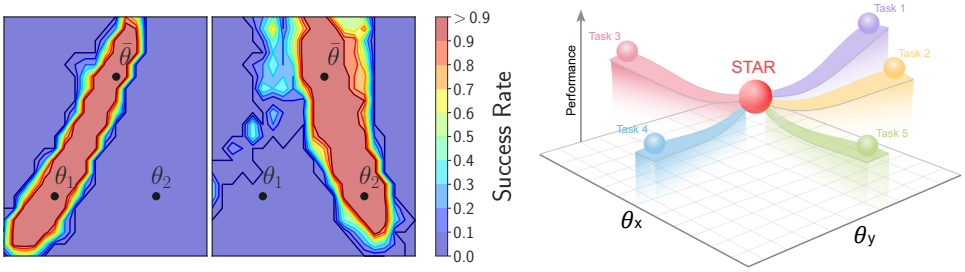
1 Introduction

Multi-Task Reinforcement Learning (MTRL) offers a solution for empowering agents with the ability to handle various tasks, a property that is missing in the traditional Reinforcement Learning (RL) setting, and demonstrated remarkable success across a diverse range of decision-making problems [1–6]. The advantage of MTRL over single-task training is in its focus on finding common knowledge that can be transferred across tasks. Albeit the numerous works that study the benefits of MTRL theoretically [7, 8] and empirically [4–6], still crucial practical challenges remain. For instance, tasks can interfere negatively with each other and slow down the learning process of a good solution that can solve all tasks. Such issue has been highlighted and tackled in prior works [9, 10] from the gradient perspective. Moreover, interference may happen due to high variance in the reward magnitudes across tasks [2], or inconsistent task complexities [11, 12] that distract the learning algorithm to focus on the more salient tasks. In turn, these challenges hinder the widespread adoption of MTRL, and leave single-task training yet a viable alternative in some applications.

Many recent works have adopted the idea of sharing modular knowledge across tasks in the form of a mixture of experts. This direction attempts to alleviate task interference by reducing the number of

*Ahmed Hendawy (ahmed.hendawy@tu-darmstadt.de) is the corresponding author.

18th European Workshop on Reinforcement Learning (EWRL 2025).



(a) Linear mode connectivity between multi-task $\bar{\theta}$ and single-task policies: θ_1 (door open), θ_2 (window open).

(b) STAR illustrative diagram.

Figure 1: (a) Visualization of the task objectives induced after applying STAR on two Metaworld tasks. (b) Illustrative diagram of the STAR topology in a 2D parameter space (θ_x, θ_y). Five single-task policies are connected linearly to the multi-task policy (red node), through linear paths where task-specific performance (differentiated by color) is maintained high.

tasks that share the same model. For instance, [5] introduce a method for learning a shared subspace of abstract policies that can be interpolated to specific tasks. [4] leverage metadata to model task relationships, enabling attention over shared representations produced by a mixture of state encoders. More recently, [6] emphasize the value of sharing diverse representations by enforcing orthogonality among outputs from a mixture of experts. However, a major drawback of these methods is the need to store and utilize the mixture of models during inference time, which defeats the purpose of MTRL to design efficient models for various skills. This limits the potential practical application of these algorithms, as the overall size of the deployed models grows significantly with respect to vanilla MTRL. This raises the question: *how can we benefit from the mixture of experts for alleviating task interference while having a unified, lightweight policy during inference?*

To address this question, we explore a new perspective on the connection between single-task and multi-task policies, focusing on the structure of the parameter space and the possible geometric characteristics of the objective landscape. We are motivated by the study conducted in [13] that demonstrates the existence of a linear path connecting multi-task and continual minima along which the loss function is maintained low. This observation is closely related to a phenomenon known as *mode connectivity* [14–16] which happens when different local minima are connected by a simple path of non-decreasing accuracy. In supervised learning, mode connectivity proved useful in finding better solutions [17], as well as alleviating the catastrophic forgetting in continual learning [13].

Building on the findings in [13], *can we leverage these insights in the MTRL framework to find a fully connected solution where all single-task policies are linearly connected to the multi-task one?* In this work, we design a MTRL algorithm that enforces a linear connectivity property in the parameter space between the multi-task policy and the individual task-specific ones. The objective of our solution is to guide the optimization of the multi-task policy by constraining its parameters to be linearly connected to the single-task policies while maintaining high performance along each linear path. During inference, we end up with a lightweight compact policy that have benefited from single-task policies during training. In Figure 1(a), we show an illustrative example of the outcome of applying our proposed approach to two tasks from Metaworld [18]. Although our approach is efficient during inference, it faces scalability challenges as the number of tasks grows. To address this, we propose an efficient MTRL algorithm that introduces a linear connectivity structure between a unified multi-task policy and a set of group-specific policies. Each group-specific policy serves as a multi-task policy for a cluster of related tasks. This design enables our method to scale effectively, matching the training efficiency of mixture-of-experts approaches while maintaining a *single*, lightweight and efficient multi-task policy at inference time.

To summarize, our contribution is threefold: *(i)* we shed light on a new approach for MTRL through the lens of mode connectivity. We empirically investigate the existence of such solution by proposing a novel MTRL algorithm, called *STAR* (see Figure 1(b)), designed to find a solution in parameter space where all single-task policies are connected to the multi-task policy via linear paths of high-performing policies. *(ii)* We propose a simple yet efficient algorithmic implementation that limits the number of training experts via group-specific policies. *(iii)* Empirically, we demonstrate the

effectiveness of our approach on two benchmarks – MuJoCo [19] and MetaWorld [18] – showing the performance gain over single-model approaches while being more computationally and resource efficient at inference than mixture-of-experts approaches. Additionally, we highlight the benefits of exploiting the learned policy subspace to further enhance performance and robustness when additional resources are available.

2 Related Works

2.1 Multi-Task Reinforcement Learning

MTRL [20] holds the promise of granting agents the capability of tackling multiple decision-making problems. The central aspect of MTRL is sharing any form of knowledge between tasks that captures their similarities and relations, hence introducing an induction bias that is lacking in single-task training. Many works have studied the benefits of MTRL and showed the potential of learning multiple tasks simultaneously [7, 8, 1, 9, 21, 4–6]. Although there is a theoretical benefit, as discussed in [7] for learning multiple tasks jointly, practically, there are challenges that limit the ultimate superiority of MTRL over single-task learning. Task interference is a crucial challenge that negatively affects the MTRL learning process. [9] study the issue from the gradient perspective, hence proposing a gradient manipulation technique that requires projecting the task gradient to the orthogonal direction to the other tasks. Another line of work has focused on sharing knowledge in modular form for reducing the interference. [5] propose to learn a subspace of abstract policies from which task-specific policies are learned. On the other hand, [4] use metadata to capture task relations for attending over shared representations generated by a mixture of state encoders. Recently, [6] study the importance of sharing diverse representations between tasks through orthogonality of the output of a mixture of experts. In this work, we investigate the potential of linearly connecting the multi-task policy to the single-task ones through the lens of mode connectivity.

2.2 Mode Connectivity

Mode connectivity [14, 15, 13] is a research direction that studies the shape of the parameter space and the geometric properties of the loss landscape of neural networks. [14, 15] observe that independently trained neural networks are connected, in the parameter space, by a simple non-linear path along which a low loss is maintained. Importantly, [17] propose an approach to learning a subspace of neural networks from scratch and of different shapes. Moreover, the linear connectivity between modes has been studied in the literature. A linear path can be found when the networks share a few training epochs [16]. [22] note that different minima that rely on pre-trained model's initialization are linearly connected along which no performance barrier exists. Importantly, [13] investigated the connectivity between multi-task and continual learning solutions in the supervised learning setting. It turns out that, clearly, there is a linear connectivity between the different minima obtained from the multi-task and the continual learning regime when the minima share the same initialization. [13] rely on this finding in designing a continual learning approach that constrains the solutions to behave like the multi-task ones through having linear connectivity with the previous tasks. In this work, we also exploit this finding to propose an effective approach for MTRL by forcing the single-task policies to be linearly connected to a common multi-task one. In RL, the mode-connectivity phenomenon has been studied in the literature. [23] learn a subspace of policies for enhancing the robustness to environment changes at test time. In addition, [24] demonstrate the advantage of the subspace of policies in tackling the sequential learning of different tasks under the continual learning regime by extending or keeping the subspace during learning. Motivated by these works, we investigate the MTRL regime of simultaneously learning a set of different tasks. As far as we know, this is the first work that studies the presence mode-connectivity phenomenon in the MTRL setting while showing its implications in designing an effective MTRL algorithm.

3 Preliminaries

In RL, we consider the problem as a Markov Decision Process (MDP) [25, 26], which is defined by a tuple $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho, \gamma >$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition distribution where $\mathcal{P}(s^{'}|s, a)$ is the probability of reaching $s^{'}$ when being in state s and performing action a, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ is the initial state distribution,

and $\gamma \in (0, 1]$ is the discount factor. The agent takes actions using a policy π that maps each state s to a probability distribution over the action space $\mathcal{A}$. The RL objective is to search for a policy π that maximizes the expected discounted return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. In Deep RL, the policy is encoded as a neural network $\pi_\theta(a_t|s_t)$, with parameters θ, that maximizes the objective $J(\pi_\theta) := J(\theta)$.

3.1 Multi-Task Reinforcement Learning

In MTRL, the agent interacts with T different tasks, where each task $c \in \mathcal{C}$ is a different MDP $\mathcal{M}^c =< \mathcal{S}^c, \mathcal{A}^c, \mathcal{P}^c, r^c, \rho^c, \gamma^c >$. The goal of MTRL is to learn a single policy π that maximizes the expected discounted return averaged across all tasks $J(\bar{\theta}) = \mathbb{E}_{c\in\mathcal{C}}\left[J_c(\bar{\theta})\right]$, where $\bar{\theta}$ is the parameter of the multi-task policy. In this work, we follow [4] in defining the MTRL problem as a Block Contextual Markov Decision Process (BC-MDP), which is 5-tuple $< \mathcal{C}, \mathcal{S}, \mathcal{A}, \gamma, \mathcal{M}^{'} >$ where $\mathcal{C}$ is the context space, $\mathcal{S}$ is the true state space, $\mathcal{A}$ is the true action space, while $\mathcal{M}^{'}$ is a mapping function that provides the task-specific MDP components given the context $c \in \mathcal{C}$, $\mathcal{M}^{'}(c) = \{\mathcal{S}^c, \mathcal{A}^c, \mathcal{P}^c, r^c, \rho^c\}$.

3.2 Mode Connectivity

Following [17], we can find a subspace of neural networks of high-accuracy and diverse solutions by learning the corresponding parametrization. The subspace is parameterized by a set of neural networks (anchors). In the case of a line, we define it by the parameters of two neural networks, $\theta_1, \theta_2 \in \mathbb{R}^d$, where d is the number of dimensions of the parameters space. We learn the anchors by optimizing for having high objective value for every parameter θ on the line,

$$\theta = \alpha\theta_1 + (1 - \alpha)\theta_2, \tag{1}$$

where $\alpha \in [0, 1]$ is the convex combination weight.

To identify mode connectivity along a line, we adapt the measure, named error barrier height, introduced in [16] to RL as shown:

$$\mathcal{B} := \frac{1}{2}(J(\theta_1) + J(\theta_2)) - \min_\alpha J(\alpha\theta_1 + (1 - \alpha)\theta_2), \tag{2}$$

where $\mathcal{B}$ is the performance barrier height which should be close to zero in case of linear connectivity.

4 Methodology

Inspired by the findings in [13] for supervised learning, we aim to investigate mode connectivity among policy networks in the context of MTRL. To this end, our approach attempts to find a set of single-task policies where each is linearly connected—in policy parameter space—to the multi-task policy. Next, we dive into the proposed method, describing the mathematical formulation and the algorithmic details.

4.1 Problem Formulation

In Deep RL, we aim to optimize for a parameter vector $\theta \in \mathbb{R}^d$ that parameterizes a policy $\pi(a|s, \theta)$ such that the discounted return for a specific problem is maximized. On the other hand, MTRL targets learning a universal policy that can perform multiple tasks. In the proposed framework, we simultaneously search for the single-task policies and the multi-task policy through optimizing for their corresponding parameters $\Theta = \left\{\theta_1, \ldots, \theta_T, \bar{\theta}\right\}$ that belong to a class of solutions $\Theta \in \Omega$, where there is a linear path that connects every single-task policy θ_c to the multi-task policy $\bar{\theta}$ along which high performance is maintained. This is achieved by optimizing the following constrained MTRL optimization problem:

$$\begin{aligned} \text{argmax}_{\Theta\in\Omega}\, \mathbb{E}_{c\in\mathcal{C}} \quad & \left[J_c(\theta_c) + J_c(\bar{\theta})\right] \\ \text{s.t.} \quad & \mathcal{B}_c := \frac{1}{2}(J_c(\theta_c) + J_c(\bar{\theta})) - \min_\alpha J_c(\alpha\theta_c + (1 - \alpha)\bar{\theta}) = 0, \quad \forall c \in \mathcal{C}, \end{aligned} \tag{3}$$

where the single-task and the multi-task policies are optimized to maximize the MTRL objective while satisfying a set of constraints of performance barrier heights maintained zero on every linear path connecting single-task and multi-task policies to enforce mode connectivity.

This constrained optimization problem can be simplified for practical reasons as optimizing for a modified MTRL objective:

$$\text{argmax}_{\Theta \in \Omega} \, \mathbb{E}_{c \in \mathcal{C}, \alpha \sim \mathcal{U}(0,1)} \, [J_c(\alpha \theta_c + (1-\alpha)\bar{\theta})], \tag{4}$$

where α is sampled from a uniform distribution $\mathcal{U}(0,1)$. Note that when $\alpha = 0$ is sampled, we optimize for the MTRL policy. In contrast, sole single-task policies can be found by setting $\alpha = 1$.

4.2 STAR: Multi-Task Reinforcement Learning via Mode Connectivity

Given the general formulation, we present a novel algorithm, named *STAR*, which can be built on top of any off-policy actor-critic RL algorithm. In this work, we use Soft Actor-Critic (SAC) [27] as the base RL method. It is worth mentioning that the name STAR is inspired by the STAR-topology in networking due to the similarity in its shape to the target solution.

Policy Update. Following Eq. 4, we optimize for having a linear path that connects each single-task policy with the multi-task one. In this context, connectivity implies that performance is maximized and maintained high for all policies across the linear subspace. To achieve this, we minimize the actor loss at the anchors (single-task and multi-task), as well as along the line by uniformly sampling the random convex combination weight(s) α for each task. In turn, we solve the following optimization problem:

$$\text{argmin}_{\Theta \in \Omega} \, \mathbb{E}_{c \in \mathcal{C}, \mathcal{D}_c \sim \mathcal{R}_c} [\underbrace{\mathcal{L}^c_{\text{actor}}(\theta_c, \mathcal{D}_c)}_{\text{Single-Task}} + \underbrace{\mathcal{L}^c_{\text{actor}}(\bar{\theta}, \mathcal{D}_c)}_{\text{Multi-Task}} + \underbrace{\mathbb{E}_{\alpha \sim \mathcal{U}(0,1)} [\mathcal{L}^c_{\text{actor}}(\alpha \theta_c + (1-\alpha)\bar{\theta}, \mathcal{D}_c)]}_{\text{Mode Connectivity}}], \tag{5}$$

where $\mathcal{D}_c$ is a batch of transitions for task c sampled from a replay buffer of the corresponding task. In this work, the actor loss follows the loss function used in SAC.

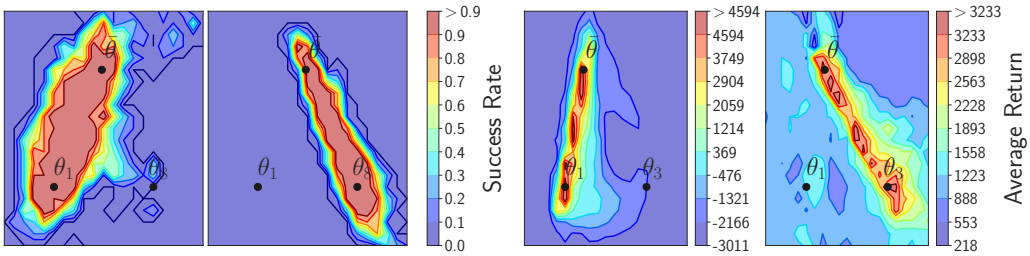
Q Update. For each task c, we model a separate critic function $Q^c_\phi(s, a, \alpha)$ defined as a neural network with a learnable parameter ϕ. The critic function models the state-action value function of every policy on the line. We learn the parameter ϕ of each critic function by minimizing the critic loss considering the state-action value function of the single-task policy ($\alpha = 1$), the multi-task policy ($\alpha = 0$), and uniformly sampled convex combination weight(s) α for modeling the interpolated policies on the line:

$$\text{argmin}_{\phi} \, \mathbb{E}_{c \in \mathcal{C}, \mathcal{D}_c \sim \mathcal{R}_c} [\underbrace{\mathcal{L}^c_{\text{critic}}(\phi, \alpha = 1, \mathcal{D}_c)}_{\text{Single-Task}} + \underbrace{\mathcal{L}^c_{\text{critic}}(\phi, \alpha = 0, \mathcal{D}_c)}_{\text{Multi-Task}} + \underbrace{\mathbb{E}_{\alpha \sim \mathcal{U}(0,1)} [\mathcal{L}^c_{\text{critic}}((\phi, \alpha, \mathcal{D}_c))]}_{\text{Mode Connectivity}}]. \tag{6}$$

Sharing a task-specific critic model across all policies on the line is an algorithmic component that effectively models the state-action value functions of the policies on the line and proves crucial for enforcing a linear subspace of high-performing solutions.

The linear connectivity is enforced by optimizing for the multi-task and the single-task policies, as well as a number of randomly selected policies on the line interpolated via the convex combination parameter α. The number of the sampled weights is a hyper-parameter of our method which can be tuned for each MTRL setting. Our ablation study shows that linear connectivity may be ensured even when sampling a single weight for each optimization update, hence reducing the computational overhead required (see Appendix for details). It is worth mentioning that, when computing the loss for multiple interpolated policies on the line, we evenly split the data batch among them for maintaining consistent training speed. Nevertheless, the loss is computed on the full batch for the single-task and multi-task anchors.

Exploration. One of the pros of building such linear paths, is having a continuous subspace of policy parameterizations which can be used for exploration. In our approach, we leverage this flexibility by moving on the line of each task with a constant predetermined step at every episode. At each episode, we then generate a complete rollout and store it in the replay buffer for training. We are motivated by prior works [23, 24] as well as our empirical study that shows the performance gain when sharing data between the single-task policy, multi-task policy, and the various policies on the line.



(a) Metaworld: Reach θ_1, Peg Insert Side θ_8 (b) Mujoco: Ant θ_1 and Hopper θ_3

Figure 2: Linear mode connectivity study showing how different single-task policies θ_c are connected to a multi-task policy $\bar{\theta}$ after applying STAR on Metaworld and Mujoco. In each sub-figure, the heatmaps on the left and right display the objective landscapes for the first and second tasks referenced in the caption, respectively.

4.3 Mode Connectivity Analysis

We investigate both the existence of the intended solution and the effectiveness of our approach in discovering it by empirically evaluating STAR in two distinct MTRL scenarios—Mujoco [19] and Metaworld—[18] with various numbers of tasks, 4 and 10, respectively. In both settings, STAR successfully converges to a fully connected solution in which each single-task policy is linearly connected to the multi-task policy via a path that maintains high performance throughout. In Fig. 2, we visualize a subset of this solution by projecting the task objective landscape onto a plane defined by two single-task policies and the multi-task policy. As shown, each single-task policy is connected to the multi-task policy by a trajectory of consistently high-performing policies. Please refer to the appendix for visualizations of the complete solution.

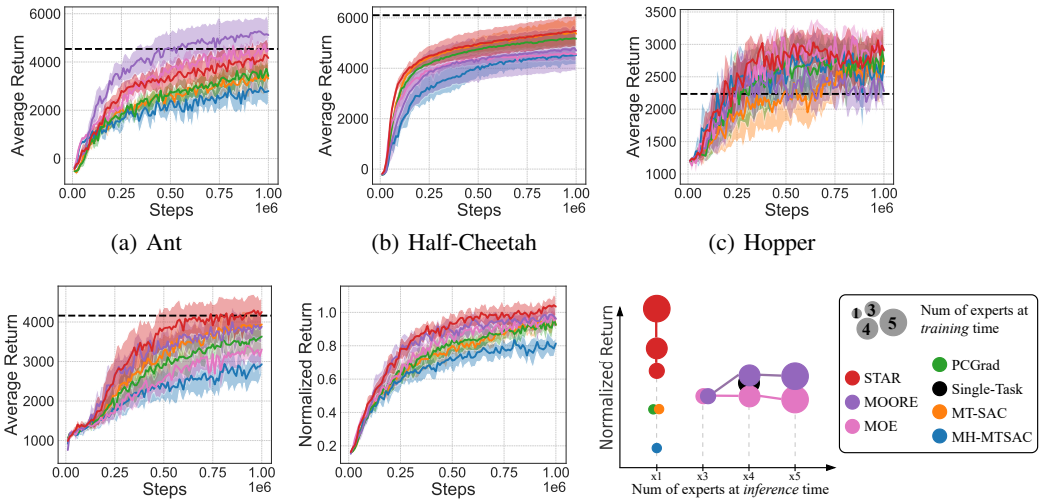
4.4 Implementation Details

Without loss of generality, we add flexibility to the practical implementation of the algorithm by allowing for *grouping* of the single-task policies. More precisely, we allow STAR to conveniently reduce the number of training experts required by distributing the tasks across predefined clusters, and learning single expert policies for each group—as in standard MTRL. The global multi-task policy is then linearly connected to each expert. The number of experts $k = 2, ..., T + 1$ represents the number of task groups in addition to the global multi-task policy. This can yield up to $T + 1$ training policies as in the original STAR configuration, which we refer to as STAR (*full*) throughout our experiments. The overall STAR algorithm combines the training effectiveness of mixture-of-experts approaches with the inference efficiency of single-model methods. Nevertheless, notice that our JAX-based [28] implementation allows for efficient and parallelized training regardless of the hyperparameter k. Refer to the Appendix for an ablation on various grouping strategies.

5 Experimental Results

In this section, we empirically demonstrate the advantage of using our approach on two MTRL benchmarks, namely MuJoco [19], and Metaworld [18]. Moreover, we present a series of ablation studies for an in-depth analysis of the different components of our method, besides, showing the robustness of the subspace in handling different changes in the environment as highlighted in [23].

Baselines. We evaluate our approach against a set of related single-model and mixture-of-experts baselines. *(i)* **SAC** [27]: representing the single-task performance; *(ii)* **Multi-Task SAC (MTSAC)**: adaptation of SAC to MTRL setting where we concatenate the context (as one-hot encoding) to the state representation fed to the policy; *(iii)* **Multi-Head MTSAC (MH-MTSAC)**: SAC with a Shared-Bottom architecture [29]; *(iv)* **PCGrad** [9]: a MTRL algorithm that handles task-interference through projecting gradients; *(v)* **Mixture Of ORthogonal Experts (MOORE)** [6]: a recent state-of-the-art MTRL algorithm that enforces diversity across a set of shared experts. *(vi)* **MOE** [6]: same architecture as MOORE but without the orthogonality constraints on the representations. On top, we have two more baselines in Metaworld that utilize pretrained embeddings of the task instructions:



(a) Ant (b) Half-Cheetah (c) Hopper

(d) Walker (e) All (Performance) (f) All (Performance vs. Efficiency)

Figure 3: The performance of STAR (Multi-task policy) and the related baselines on the MT4 of MuJoCo, showing the Average Return on the four environments while reporting the normalized Average Return relative to the single-task performance as an overall metric (e). The black dashed line represents the final single-task performance of SAC on each environment. We report the mean and the 95% confidence interval across 10 seeds. For STAR, the number of sampled α weights is 1. For MOE, MOORE, and STAR, the number of experts is 3. We finally explore the performance-efficiency trade-off for various number of training experts in (f), highlighting the number of experts retained at inference time by each method.

(vii) **FiLM** [30]: is a task-conditioned policy using the FiLM module, *(viii)* **CARE** [4]: is an attention-based approach that learns a mixture of state encoders conditioned on metadata of the tasks.

5.1 MuJoCo

For MuJoCo benchmark, the MTRL setting involves four different tasks: Ant, Half-Cheetah, Hopper, and Walker. The environments have different state and action dimensionality, hence we perform a padding process to the state and action spaces complying with the BC-MDP definition in Sec. 3.1.

In Figure 3, we compare the performance of the multi-task policy of STAR against both solo model and mixture-of-experts baselines across various MuJoCo environments. Overall, STAR consistently strikes a strong balance across tasks, outperforming all baselines on average. While MOORE surpasses STAR on the Ant environment (Figure 3(a)), it struggles to maintain competitive performance elsewhere, likely due to being overly influenced or distracted by a single environment. On Walker (Figure 3(d)), our approach shows a noticeable performance gap to the closest baseline while exceeding the single-task performance. Finally, Figure 3(e) shows that STAR achieves the highest average performance across environments, while utilizing a single model during inference.

In Figure 3(f), we take a closer look at the training *effectiveness* and inference *efficiency*. The plot shows the final performance of each method, along with the number of experts used during training and inference. Single-model approaches underperform, as they rely solely on a single expert during training. Mixture-of-experts methods improve performance but at the cost of increased inference complexity. In contrast, STAR offers a more favorable trade-off—*effectively* leveraging multiple experts during training while retaining the *efficiency* of single-model inference. Notably, STAR's performance scales with the number of training experts, unlike MOORE and MOE, which saturate.

5.2 Metaworld

To assess the scalability of STAR and its ability to discover solutions that satisfy the mode connectivity property, we benchmark our approach on the Meta-World suite [18], a widely adopted benchmark for MTRL comprising a diverse set of robotic manipulation tasks. Specifically, we focus on the MT10-

Table 1: The performance of STAR and the related baselines on Metaworld MT10 with random goals. We then average the success rate across all 10 tasks, computed after evaluating on all 50 goals. We report the mean and the standard deviation of the metric across 10 seeds. For STAR, we consider the performance of the multi-task policy. The number of sampled α weights is 4.

Method	Success Rate (%) ↑	Train → Test Parameters (M) ↓
SAC [18]	85.6±4.8	3.40 → 3.40
MTSAC [18]	70.4±4.8	0.34 → 0.34
MH-MTSAC [18]	75.6±6.4	0.37 → 0.37
PCGrad [9]	73.5±7.6	0.34 → 0.34
FiLM [30]	51.9±9.5	0.45 → 0.45
CARE [4]	67.4±6.5	0.51 → 0.51
MOE [6]	77.7±9.6	1.38 → 1.38
MOORE [6]	89.7±0.3	1.38 → 1.38
STAR (ours)	85.3±5.8	1.38 → 0.34

rand setting, which challenges a single robot to perform 10 distinct skills, each with randomized goal positions. Evaluating on Metaworld provides a rigorous test of STAR's ability to adhere to the mode connectivity property, thereby guiding the multi-task policy toward regions of high performance in parameter space.

In Table 1, we present the final success rates of all baselines after 20 million training steps (2 million per task), along with their respective parameter counts during both training and inference. As the results show, STAR exhibits a good trade-off between performance and inference-time efficiency. By effectively leveraging a mixture of models during training, enforcing the linear mode connectivity property, STAR improves the performance of the multi-task policy relative to single-model approaches. While MOORE achieves higher overall success rates, it incurs a significant overhead by requiring all expert models to be stored, accessed, and activated during inference—a limitation that STAR overcomes with minimal compromise in performance.

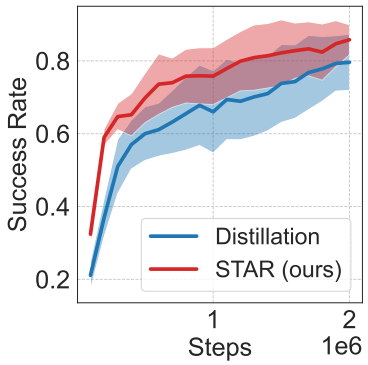


Figure 4: Average success rate of STAR vs. a Distillation approach on Metaworld MT10-rand scenario.

5.3 Relation to Distillation

The objective of our approach is to leverage a mixture of experts during training to guide the multi-task policy parameters toward regions of high performance.To achieve this, we impose constraints that enforce the existence of linear paths in parameter space—connecting each expert to the multi-task policy—along which performance remains consistently high. A similar objective was explored in prior work, notably Distral [1], which learns a set of single-task policies that distill their knowledge into a centralized policy. This shared policy, in turn, regularizes the training process by serving as a prior. This comparison raises an important question: *Can we distill the experts' knowledge directly, instead of relying on mode connectivity?*

To answer this question, we implemented a Distral-inspired baseline built on our framework, where knowledge is distilled from a mixture of experts—rather than from single-task policies—by minimizing the Kullback–Leibler (KL) divergence between the multi-task policy and each expert. This replaces the role of enforcing the mode connectivity property. In Figure 4, we compare STAR against this distillation baseline on the MT10-rand scenario from Meta-World. STAR outperforms the baseline in both sample efficiency and final performance, highlighting its effectiveness. These results suggest that STAR offers a compelling alternative to distillation in MTRL, potentially achieving a similar goal by *implicitly distilling expert knowledge through the lens of mode connectivity*.

Table 2: Study on different inference strategies for STAR and STAR (*full*) conducted on Mujoco MT4, and Metaworld MT10 with random goals. We average the performance metric across all tasks, computed at the end of training. We report the mean and the standard deviation of the metric across 10 seeds. We report the normalized average return for Mujoco and the success rate in Metaworld. The number of sampled α weights is 1 and 4 on Mujoco and Metaworld, respectively.

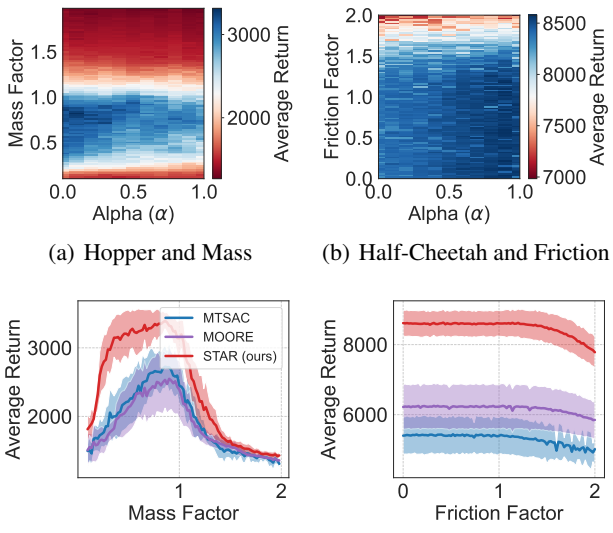
Inference Strategies	Mujoco		Metaworld	
	STAR	STAR (*full*)	STAR	STAR (*full*)
Group/ Single-Task	1.01±0.08	1.18±0.09	85.1±5.8	85.6±7.7
Midpoint	1.03±0.07	1.22±0.06	85.6±5.7	85.9±7.7
Multi-Task	1.04±0.08	1.21±0.08	85.3±5.8	85.8±7.7
Max-on-Line	1.09±0.07	1.27±0.06	85.9±5.8	86.1±7.7

5.4 Beyond the Multi-task Policy

We finally explore novel possibilities in leveraging the mode connectivity induced by STAR at training time. Particularly, we study how moving across the linear manifold of each task affects the overall performance and its robustness to dynamics variations.

STAR for novel inference strategies. In practice, performance naturally exhibits slight variations along the linear subspace induced by STAR, as mode connectivity can only be promoted via sample estimates. We notice that, when resources permit, higher performance can be achieved at inference time by searching for the best-performing policy across the linear subspace connecting the multi-task policy with the underlying task-specific expert—referred to as *Max-on-Line*. In Table 2 we report the performance for a collection of inference strategies that leverage the star-shaped manifold, conducted on Mujoco and Metaworld. We conclude that the multi-task policy offers a favorable trade-off between performance and efficiency, with only a modest performance gap compared to more resource-intensive strategies.

STAR for robustness. We further explore whether policies across the linear path between the single-task and multi-task anchors may exhibit different robustness capabilities, despite the similar performance. In Figure 5, we present two instances of this study where we change the nominal body mass in the Hopper environment by a factor in range $[0.1, 2.0]$, in addition to the scaling of the nominal value of the ground friction coefficient in the Half-Cheetah environment by a factor between $[0.01, 2.0]$. We illustrate how the policies on the line handle varying dynamic parameters. For instance, as we move towards the multi-task policy, our approach demonstrates high robustness capabilities with respect to the mass changes, especially in the low mass range as shown in Figure 5(a). On the other hand, in Figure 5(b), the subspace demonstrates more robustness to the changes of the friction coefficients as we move to the single-task policy end ($\alpha = 1$). Comparing our approach to a single-model approach MTSAC and a mixture-of-expert one MOORE in Figures 5(c) and 5(d), we surely notice the edge of the subspace for being robust to different changes to various environments. In Appendix, we include the complete robustness study conducted on all Mujoco tasks.



(a) Hopper and Mass (b) Half-Cheetah and Friction

(c) Hopper and Mass (d) Half-Cheetah and Friction

Figure 5: Robustness analysis.

6 Conclusion and Discussion

In this work, we explored mode connectivity in MTRL and empirically showed that linear paths in parameter space can link the multi-task policy to each single-task policy, preserving high performance throughout. Leveraging this insight, we introduced STAR, a scalable and efficient MTRL algorithm that enforces linear mode connectivity between a global multi-task policy and a mixture of expert policies. This formulation steers the multi-task policy toward regions of high performance that satisfy the connectivity constraint. Empirically, STAR achieves a strong balance between performance and inference-time efficiency—an advantage over existing mixture-of-experts methods in MTRL. Furthermore, we showed that the learned subspace supports flexible inference strategies—when resources permit—and demonstrates robustness to environmental variations at test time.

Limitations. While we evaluate our approach on two distinct benchmarks, including MuJoCo tasks with varying morphologies, we acknowledge that further investigation into the relationship between mode connectivity and task similarity is important. Our method assigns an expert to groups of tasks, and we include an ablation study on this clustering in the Appendix. Exploring alternative clustering strategies, including learning-based approaches, presents a promising direction for future research.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) (Project: 01IS22078). This work was also funded by Hessian.ai through the project 'The Third Wave of Artificial Intelligence – 3AI' by the Ministry for Science and Arts of the state of Hessen. The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project b187cb. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683. Calculations for this research were also conducted on the Lichtenberg highperformance computer of the TU Darmstadt and the Intelligent Autonomous Systems (IAS) cluster at TU Darmstadt.

References

[1] 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*, 2017.

[2] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. *CoRR*, abs/1809.04474, 2018.

[3] Yash Chandak, Shantanu Thakoor, Zhaohan Daniel Guo, Yunhao Tang, Remi Munos, Will Dabney, and Diana L Borsa. Representations and exploration for deep reinforcement learning using singular value decomposition. In *International Conference on Machine Learning*, pages 4009–4034. PMLR, 2023.

[4] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, 2021.

[5] Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Paco: Parameter-compositional multi-task reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=LYXTPNWJLr`.

[6] Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *The Twelfth International Conference on Learning Representations*, 2024.

[7] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

[8] Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. Sparse multi-task reinforcement learning. In *Advances in Neural Information Processing Systems*, 2014.

[9] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020.

[10] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

[11] Myungsik Cho, Jongeui Park, Suyoung Lee, and Youngchul Sung. Hard tasks first: Multi-task reinforcement learning through task scheduling. In *Forty-first International Conference on Machine Learning*.

[12] Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Efficient multi-task and transfer reinforcement learning with parameter-compositional framework. *IEEE Robotics and Automation Letters*, 8(8):4569–4576, 2023.

[13] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *arXiv preprint arXiv:2010.04495*, 2020.

[14] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[15] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[16] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[17] Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In *International Conference on Machine Learning*, pages 11217–11227. PMLR, 2021.

[18] 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*, 2019.

[19] 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*, pages 5026–5033. IEEE, 2012.

[20] Nelson Vithayathil Varghese and Qusay H Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9):1363, 2020.

[21] Ruihan Yang, Huazhe Xu, YI WU, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems*, 2020.

[22] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

[23] Jean-Baptiste Gaya, Laure Soulier, and Ludovic Denoyer. Learning a subspace of policies for online adaptation in reinforcement learning. *arXiv preprint arXiv:2110.05169*, 2021.

[24] Jean-Baptiste Gaya, Thang Doan, Lucas Caccia, Laure Soulier, Ludovic Denoyer, and Roberta Raileanu. Building a subspace of policies for scalable continual learning. *arXiv preprint arXiv:2211.10445*, 2022.

[25] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

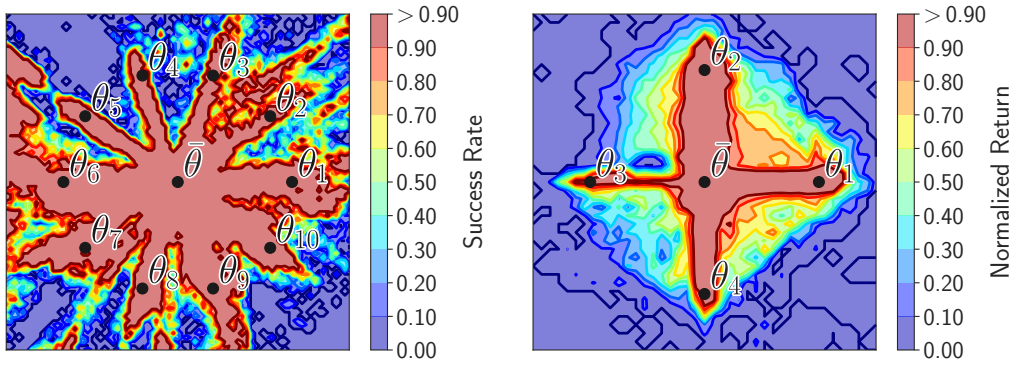
[26] Martin L Puterman. Markov decision processes: Discrete stochastic dynamic programming. *Journal of the Operational Research Society*, 1995.

[27] 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*, 2018.

[28] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

[29] Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

[30] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. *CoRR*, abs/1709.07871, 2017.



(a) STAR topology of Metaworld MT10

(b) STAR topology of MuJoco MT4

Figure 6: Visualization of the STAR topology in case of Metaworld MT10 and MuJoco MT4 scenarios. Each is a contour plot of the objective landscape of all tasks summed and clipped to 1, computed on a 2D parameter space resulted from projecting each 2D plane which contains a pair of task policies and the multi-task one. These contour plots are the realization of the illustrative diagram of the STAR topology in Figure 1(b)

A Experimental Results on Metaworld MT50

To evaluate the scalability of STAR, we tested our approach on the MT50 benchmark from Metaworld, which comprises 50 challenging yet related robotic manipulation tasks. As in the MT10 setup, we consider a more complex variant—MT50-rand—in which both goal and object positions are randomized. We benchmark STAR against the same set of baselines as in the MT10 experiment, excluding PCGrad due to limited computational resources. Table 3 presents the final performance of each approach, reported as the success rate, along with the number of parameters (in millions) used during training and inference. SAC outperforms all multi-task reinforcement learning (MTRL) approaches by leveraging 17M parameters—dedicating a separate policy to each task. Conversely, MOORE [6] achieves the highest performance among MTRL methods while utilizing around 1.5M parameters during both training and inference. Despite its strong performance, MOORE presents practical limitations due to the need to deploy all expert models at inference. In contrast, STAR offers a substantial performance gain over single-policy baselines like MTSAC by leveraging a mixture of experts during training and preserving a lightweight multi-task policy at test time. This demonstrates STAR's ability to effectively trade off a modest drop in accuracy for substantial gains in deployment efficiency.

Table 3: The performance of STAR and related baselines on Metaworld MT50 with random goals (MT50-rand). We report the final success rate, computed after 2 million training steps per task and averaged across all 50 tasks. We report the mean and the standard deviation of the metric across 10 seeds. For STAR, we consider the performance of the multi-task policy. In addition, the number of sampled α weights used during training is 4. Moreover, the number of experts is 4 for STAR, MOORE, and MOE.

Method	Success Rate (%) ↑	Train → Test Parameters (M) ↓
SAC [18]	75.0±3.6	17.0 → 17.0
MTSAC [18]	53.2±2.4	0.36 → 0.36
MH-MTSAC [18]	49.6±1.7	0.50 → 0.50
FiLM [30]	47.3±2.4	0.48 → 0.48
CARE [4]	50.8±2.1	0.57 → 0.57
MOE [6]	59.7±3.4	1.51 → 1.51
MOORE [6]	69.6±2.2	1.51 → 1.51
STAR (ours)	60.3±3.1	1.44 → 0.36

B Additional Empirical Analyses

In this section, we present the complete results from experiments discussed in the main paper as well as other studies and findings that could not fit because of the limited space.

B.1 Clustering strategies

As discussed in Section 4.4, the original STAR, referred to as STAR ($full$), suffers from limited scalability due to its reliance on connecting individual single-task policies with a multi-task policy. This design leads to a linear growth in the number of parameters as the number of tasks increases. To address this limitation, we propose replacing the single-task policies with a mixture of group-specific multi-task policies. Each group comprises a subset of tasks, with a dedicated policy trained to handle that subset. In this work, we cluster tasks into groups using one of the following strategies:

- **Naive**: Tasks are grouped sequentially based on their task IDs. Given T tasks and $k-1$ groups, the first $\left\lfloor \frac{T}{k-1} \right\rfloor$ tasks are assigned to the first group, and so on, with any remaining tasks allocated to the final group. Here, k denotes the total number of policies, including the $k-1$ group-specific policies and one global multi-task policy at the center of the star topology.
- **Random**: Tasks are randomly assigned to groups for each seed while maintaining approximately balanced group sizes.
- **Manual**: Groups are manually constructed based on prior knowledge of task relationships.

Since the MuJoCo scenario includes only four tasks, we adopt the *random* clustering strategy for all corresponding experiments. In contrast, we apply the *manual* clustering strategy in both the MT10 and MT50 scenarios of Metaworld. For instance, an illustrative diagram highlighting the clusters used in the MT10 scenario is presented in Figure 7.

To assess the impact of different grouping strategies on performance, we evaluate our approach using each strategy on the Metaworld MT50 benchmark. Table 4 presents the final success rate after 2 million training steps per task. While the observed performance differences across strategies are relatively modest, results indicate a trend toward improved outcomes when more thoughtful grouping is applied. These findings motivate future investigation into automated, learning-based clustering approaches that leverage task-related information, such as language instructions.

Additionally, we include the performance of various inference strategies on the MT50 benchmark to complement the results shown in Table 2. As observed, the multi-task policy achieves performance comparable to other inference methods, offering a good trade-off between performance and inference-time efficiency. In contrast, selecting the best-performing task-specific policy yields the highest success rates but comes at a higher computational and resource cost.

Table 4: Study on different inference strategies for STAR conducted on Metaworld MT50 with random goals. In addition, we show the performance of STAR with different clustering strategies. We average the success rate across all 50 tasks, computed at the end of training (after 2 million training steps per task). We report the mean and the standard deviation of the metric across 10 seeds. The number of sampled α weights is 4. The number of experts k is 4 (3 group-specific policies and a global multi-task policy).

Inference Strategies	Clustering Strategies		
	Naive	Random	Manual
Group	57.7±2.7	59.1±3.2	59.5±3.4
Midpoint	58.5±3.1	60.0±3.2	60.3±3.5
Multi-Task	58.5±2.4	59.9±3.4	60.3±3.1
Max-on-Line	59.1±2.7	60.5±3.2	61.2±3.2

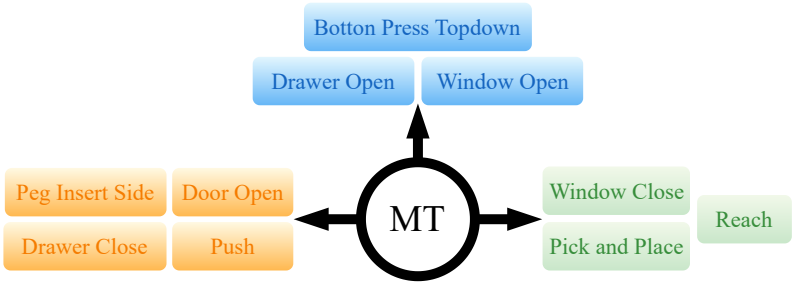


Figure 7: An illustration of the manual clusters used for the Metaworld MT10 scenario. STAR ($k = 4$) learns a mixture of 3 group-specific policies, one policy per group, in addition to the global multi-task policy at the center of the star topology.

B.2 Ablation on algorithmic components

Our approach, STAR, aims to couple the training of a mixture of policies with a central multi-task policy by enforcing the mode connectivity (MC) property. Two key components contribute to forming a star topology that achieves high performance along each linear path connecting the multi-task policy to a group-specific policy. In Section 4, we highlight those components which are: how the state-action value function is modeled for each path and how rollouts are collected per task.

For critic design, we model the state-action value function along each linear path using a dedicated critic network. Specifically, STAR employs $k - 1$ critic networks—one for each group—while STAR ($full$) uses a separate critic per task. This design allows more accurate modeling of value functions in the shared subspace, thereby enhancing the quality of the learned paths. The following section examines different design choices for the critic model.

Additionally, our approach introduces another form of coupling during data collection. We leverage the full set of policies on the line, including both the multi-task and group-specific policies, for exploration and rollout generation. This strategy enriches the collected transitions for each task by introducing diverse behaviors from multiple policies.

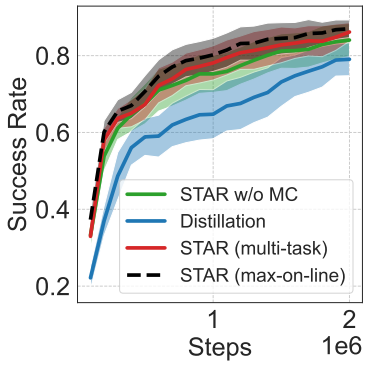


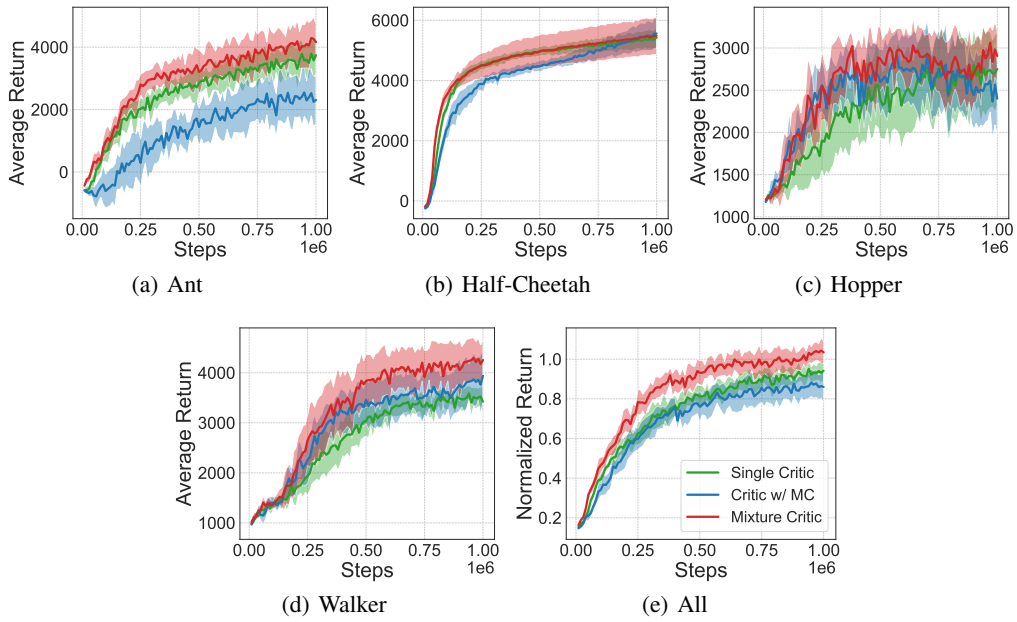
Figure 8: Comparison of STAR, STAR w/o MC, and the Distillation baseline on the Metaworld MT10-rand benchmark. We report the average success rate across all tasks during 2 million training steps per task, with mean and the 95% confidence interval computed over 20 seeds. All methods evaluate the performance of the multi-task policy. For STAR, we also report the performance using the max-on-line inference strategy. The number of experts is set to 4; for STAR, 4 α values are sampled along each linear path. The distillation baseline uses a KL-divergence regularization weight of $\lambda = 0.5$.

To evaluate the contribution of these two components, we conduct an ablation study on the Metaworld MT10 benchmark, comparing STAR with and without mode connectivity. As shown in Figure 8, a performance gain is observed for STAR w/o MC due to our critic design and shared rollouts. When mode connectivity is enforced, performance further improves due to two factors: the constraint of maintaining high performance across the linear paths, and the use of multiple intermediate policies for data collection. This setup helps position the multi-task policy in a high-performing region of the parameter space. Furthermore, when applying the max-on-line inference strategy—selecting the best policy along the line per task—we observe an additional performance gain due to task-specific specialization.

Figure 8 also compares our method to a distillation baseline. Surprisingly, distilling expert knowledge into the multi-task policy via KL divergence (see Equation 7) yields degraded performance, despite leveraging the same critic and shared rollout components. This result suggests that direct distillation may introduce harmful regularization or insufficient transfer when performed online under the given task diversity.

$$\mathcal{L}_{\text{distill-actor}} = \mathcal{L}_{\text{actor}} + \lambda \, \mathbb{E}_{g \in \mathcal{G}} \left[D_{\text{KL}}(\pi_{\bar{\theta}} \,||\, \pi_{\theta_g}) \right], \tag{7}$$

where $\mathcal{G}$ is the cluster space, and g is the cluster identifier.



(a) Ant (b) Half-Cheetah (c) Hopper

(d) Walker (e) All

Figure 9: Study on the influence of the critic design choice on the performance of multi-task policy of STAR, conducted on the MT4 of MuJoCo, showing the Average Return on the four environments while reporting the normalized Average Return relative to the single-task performance as an overall metric (e). We report the mean and the 95% confidence interval across 10 seeds. The number of sampled α weights is 1. The number of experts k is 3.

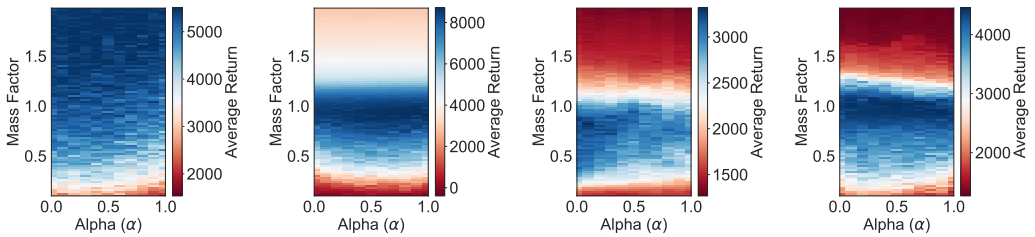
B.3 Critic design ablation

In this section we examine how the choice of critic architecture shapes the performance of STAR's multi-task policy. We evaluate three design choices:

- **Single critic**: A single neural network $Q(s, a, c, \alpha)$ estimates the state–action value function for all policies in the star topology. Task context c and convex-combination coefficient α are concatenated to the state–action input.
- **Critic w/ MC**: We employ k separate critics—one per task group (task in case of STAR ($full$) plus one for the multi-task policy—and impose *mode connectivity* constraints on their parameters, mirroring the constraint used for the policies themselves.
- **Mixture critic**: Each of the $k - 1$ linear paths connecting a group-specific policy to the multi-task policy is assigned its own critic model that is conditioned on task context c and combination weight α.

Figure 9 summarizes the results on the MuJoCo MT4 benchmark, with learning curves for individual tasks (Figures 9(a)–9(d)) and for the average across all environments (Figure 9(e)). The study demonstrates the influence of the critic choice on the performance of the multi-task policy of STAR.

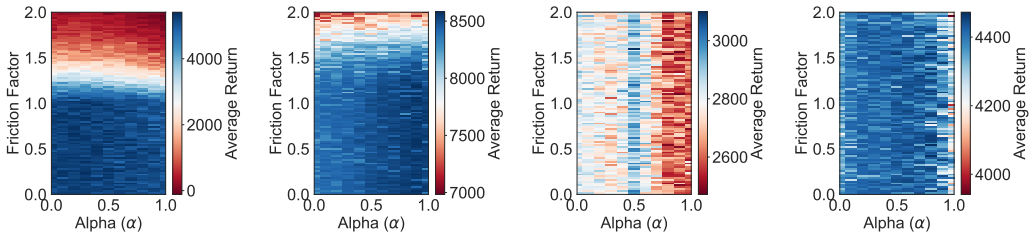
The findings indicate that a single critic lacks the capacity to model the various value functions of the policies in the star topology. Moreover, imposing mode connectivity on Q-functions hurts learning—most clearly on the Ant task (Figure 9(a))—potentially because (i) Q-function landscapes do not exhibit the same connectivity as policy landscapes, and/or (ii) bootstrap targets create a non-stationary objective. To our knowledge, mode connectivity for Q-functions has not been explored previously in the literature, motivating future work. By contrast, allocating a dedicated critic to each linear path consistently delivers the best performance, underscoring the importance of this design choice in STAR.



(a) Ant (b) Half-Cheetah (c) Hopper (d) Walker

Figure 10: Robustness analysis of STAR on the four Mujoco environments with respect to changes in *mass*. Each heatmap shows the performance across the linear subspace with respect to changes in the dynamics parameter. We report the mean of the Average Return metric across 10 seeds. The number of experts is 5 (STAR (*full*)) and the number of sampled α weights is 1

.



(a) Ant (b) Half-Cheetah (c) Hopper (d) Walker

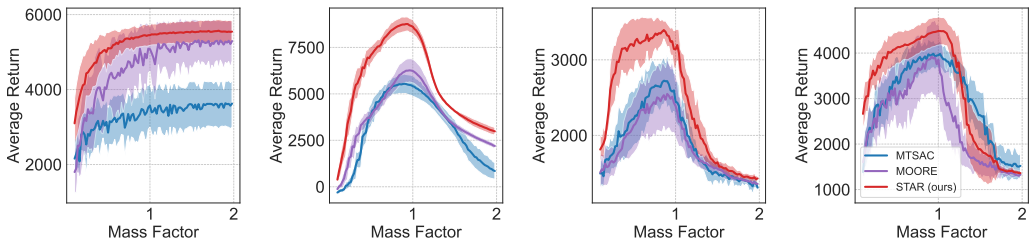
Figure 11: Robustness analysis of STAR on the four Mujoco environments with respect to changes in *friction*. Each heatmap shows the performance across the linear subspace with respect to changes in the dynamics parameter. We report the mean of the Average Return metric across 10 seeds. The number of experts is 5 (STAR (*full*)) and the number of sampled α weights is 1

B.4 Robustness analysis

To complete our discussion on the robustness of the learned subspace, we provide the full results from our study. Figures 10 and 11 show heatmaps depicting policy performance along the interpolation line under variations in mass and friction, respectively, across four Mujoco environments.

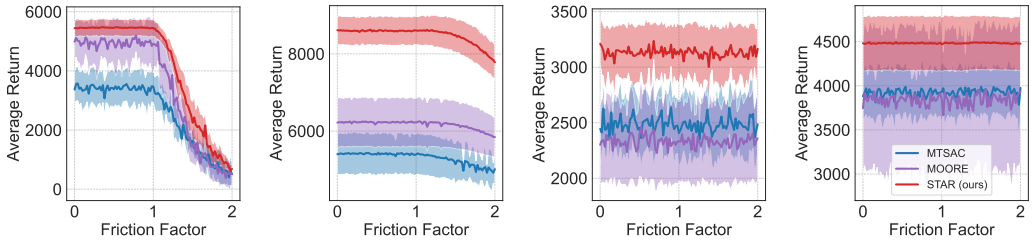
Additionally, we compare the robustness of our approach, STAR, by reporting the maximum performance along the interpolation line against the MOORE and MTSAC baselines. The corresponding quantitative results under varying mass and friction conditions are presented in Figures 12 and 13.

All methods are trained on the nominal tasks where the *mass* and *friction* factors are set to 1. The analysis demonstrates how different methods behaves when changes happen to dynamic parameters at inference time.



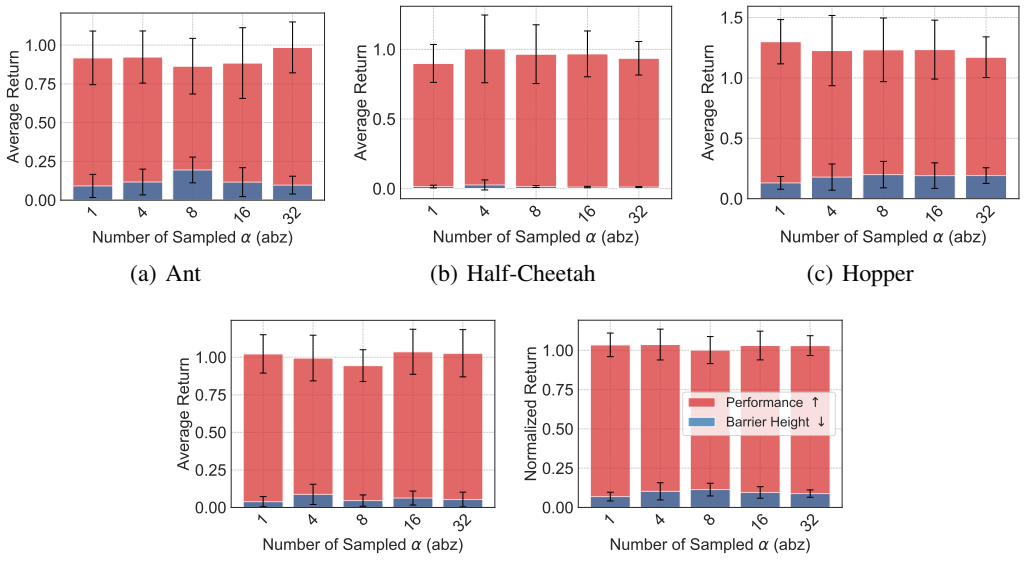
(a) Ant (b) Half-Cheetah (c) Hopper (d) Walker

Figure 12: Robustness analysis, comparing STAR to MOORE and MTSAC baselines on the four Mujoco environments with respect to changes in *mass*. We report the mean and the 95% confidence interval of the Average Return metric across 10 seeds. For STAR and MOORE, the number of experts is 5. For STAR, the number of sampled α weights is 1.



(a) Ant (b) Half-Cheetah (c) Hopper (d) Walker

Figure 13: Robustness analysis, comparing STAR to MOORE and MTSAC baselines on the four Mujoco environments with respect to changes in *friction*. We report the mean and the 95% confidence interval of the Average Return metric across 10 seeds. For STAR and MOORE, the number of experts is 5. For STAR, the number of sampled α weights is 1.



(a) Ant (b) Half-Cheetah (c) Hopper

(d) Walker (e) All

Figure 14: A sensitivity analysis on the effect of the number of sampled α weights on the performance of the multi-task policy of STAR (in red). In addition, we include the mode connectivity metric (in blue), performance barrier height. The results considers all four MuJoco environments as well as the overall relative performance to the single-task SAC. We report the mean and the standard deviation of the performance metric across 10 different runs. The number of experts k employed during training is 3.

B.5 Sensitivity analysis on the number of sampled α

In this section, we investigate how the frequency of sampling the convex combination weight α during training affects the performance of our method. The number of sampled α values serves as a hyperparameter that influences the discovery of linear subspaces connecting each group-specific (single-task) policy to the multi-task policy. When sampling multiple α values per training step, the training batch is split accordingly to allow parallel updates, accelerating the overall learning process.

Figure 14 presents the results of this study on the MuJoCo MT4 benchmark. It reports the final performance of STAR's multi-task policy across individual environments as well as the average performance across all tasks, with respect to varying the number of α weights. To further assess the quality of the learned subspaces, we also report the Performance Barrier Height—a metric introduced in Section 3 that quantifies mode connectivity.

The results reveal that sampling even a single α per training step is sufficient to achieve strong multi-task performance along with well-connected solutions (as indicated by a low barrier height). While modest fluctuations in both performance and connectivity metrics are observed across different α sampling frequencies, these differences are minimal. This indicates that our method is relatively robust to this hyperparameter, reducing the need for fine-grained tuning.

B.6 Linear mode connectivity analysis

As described in Section 4.3, we visualize the objective landscapes of various tasks using contour plots defined over a 2D plane that includes two single-task policies and the shared multi-task policy. These plots illustrate the linear mode connectivity between each pair of single-task policies and the common multi-task policy.

For instance, to construct each contour plot for the parameter vectors θ_1, θ_2, and $\bar{\theta}$. We first define two basis vectors: $u = \theta_2 - \theta_1$ and $v = \bar{\theta} - \theta_1$. We then orthogonalize v with respect to u using the transformation: $v \leftarrow v - \cos(u, v) \cdot u$. A mapping function $p(x, y) = \theta_1 + u \cdot x + v \cdot y$ translates each point in the Cartesian 2D space to a point in the parameter space. The performance metric (e.g., success rate or average return) is evaluated at each mapped point to generate the final contour plot.

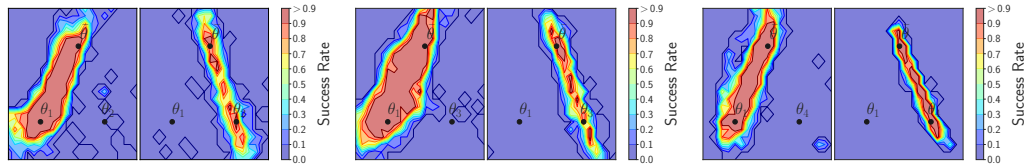
Figure 2(a) illustrates a representative example of our analysis on the MT10 Metaworld benchmark, using a single seed that achieved 100% success. The example focuses on the Reach and Peg Insert Side tasks. Comprehensive contour plots for all other task pairs in MT10, excluding redundant combinations, are provided in Figures 15–23. It is worth noting that the minor performance fluctuations and the imperfect connectivity observed along some linear paths are primarily attributable to the resolution of the 2D evaluation grid, which is set to 21×21.

Similarly, Figure 2(b) showcases an example from the MuJoCo MT4 benchmark, illustrating linear mode connectivity between the Ant and Hopper tasks. The full set of contour plots for all remaining task pairs in this benchmark is provided in Figures 24–29. Additionally, in the same figures, we include contour maps generated using STAR without mode connectivity (STAR w/o MC). These results reveal significant disconnectivity between the single-task policies and the multi-task policy, emphasizing the critical role of learning intermediate policies along linear paths by sampling interpolation weights α during training.

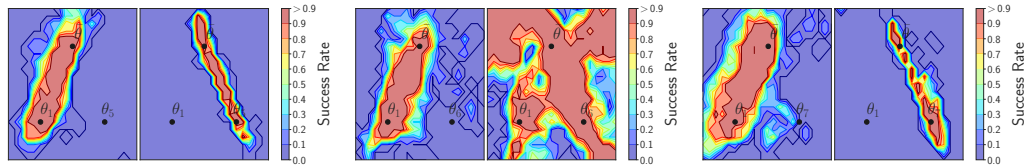
The presented individual contour plots aim to show the accurate objective landscapes for a pair of tasks. To have a high level view of the final solution, Figure 6 illustrates a compressed contour plot for Metaworld MT10 and Mujoco MT4 in Figures 6(a) and 6(b), respectively. Each plot includes all single-task policies and the multi-task one while presenting the linear paths connecting them along which high performance is maintained. To improve visualization fidelity, we use higher-resolution 2D evaluation grids: 81×81 for Figure 6(a) and 41×41 for Figure 6(b).

The parameter space that is used to generate the plots is created by dissecting the plane around $\bar{\theta}$ using equally spaced points around the unit circle. This yields one triangular section per policy. On to each section of this space we then project a local parameter subspace that is constructed using the center policy $\bar{\theta}$ and the midpoints $\frac{\theta_c+\theta_{c+1}}{2}$, $\frac{\theta_c+\theta_{c-1}}{2}$ of the task c and the adjacent tasks along the circle. This ensures that adjacent slices always intersect, in order to avoid any discontinuities in the heatmap. Consequently, the linear connector for each task is located in the center of each slice.

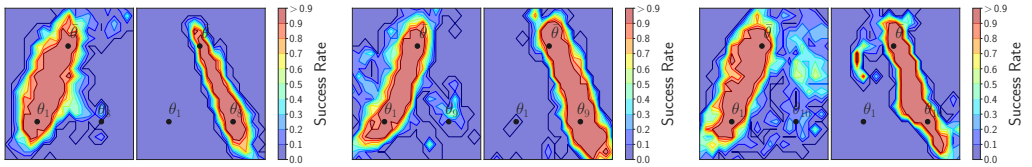
The parameters computed for each point on the grid are then evaluated on all tasks. For Metaworld, we obtain the success rate for each task in $[0, 1]$, whereas in MuJoco we normalize the return of each task relative to the single-task performance to be in the same interval. Instead of computing the mean over all tasks, we sum up the individual values and then clip them to 1. Without this clipping, the multitask performance would drop as we move further away from the central policy. Although the star would still be visible, we decided to go with the clipped variant to exaggerate the star shape's visibility in the heatmap.



(a) reach-v2 and push-v2 (b) reach-v2 and pick-place-v2 (c) reach-v2 and door-open-v2

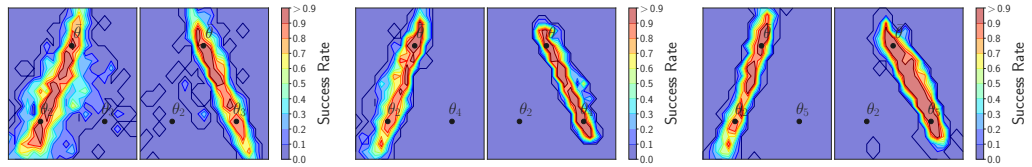


(d) reach-v2 and drawer-open-v2 (e) reach-v2 and drawer-close-v2 (f) reach-v2 and button-press-topdown-v2

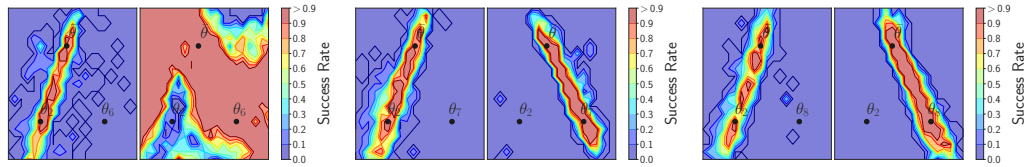


(g) reach-v2 and peg-insert-side-v2 (h) reach-v2 and window-open-v2 (i) reach-v2 and window-close-v2

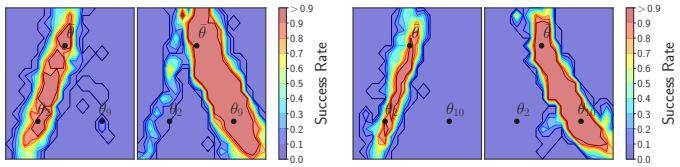
Figure 15: Contour plots representing the task objective landscape of the *reach-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.



(a) push-v2 and pick-place-v2 (b) push-v2 and door-open-v2 (c) push-v2 and drawer-open-v2

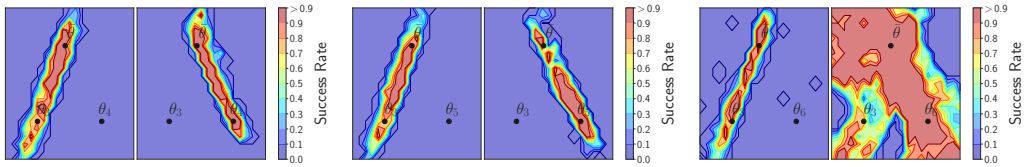


(d) push-v2 and drawer-close-v2 (e) push-v2 and button-press-topdown-v2 (f) push-v2 and peg-insert-side-v2

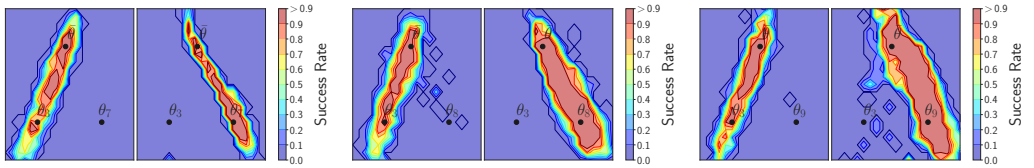


(g) push-v2 and window-open-v2 (h) push-v2 and window-close-v2

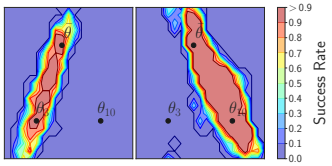
Figure 16: Contour plots representing the task objective landscape of the *push-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.



(a) pick-place-v2 and door-open-v2 (b) pick-place-v2 and drawer-open-v2 (c) pick-place-v2 and drawer-close-v2

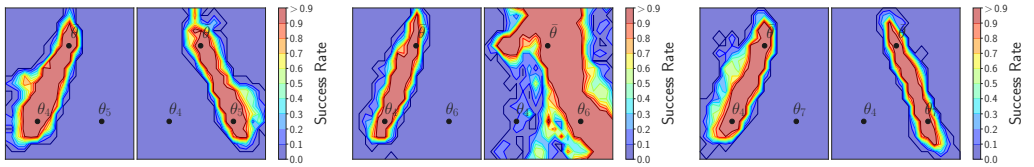


(d) pick-place-v2 and button-press-topdown-v2 (e) pick-place-v2 and peg-insert-side-v2 (f) pick-place-v2 and window-open-v2

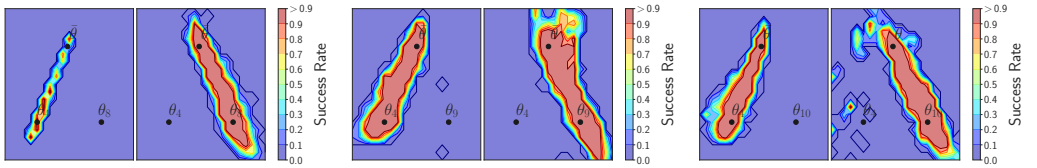


(g) pick-place-v2 and window-close-v2

Figure 17: Contour plots representing the task objective landscape of the *pick-place-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.

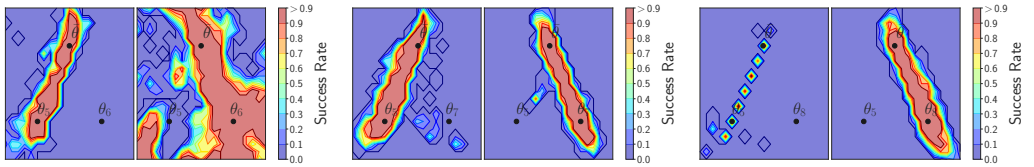


(a) door-open-v2 and drawer-open-v2 (b) door-open-v2 and drawer-close-v2 (c) door-open-v2 and button-press-topdown-v2



(d) door-open-v2 and peg-insert-side-v2 (e) door-open-v2 and window-open-v2 (f) door-open-v2 and window-close-v2

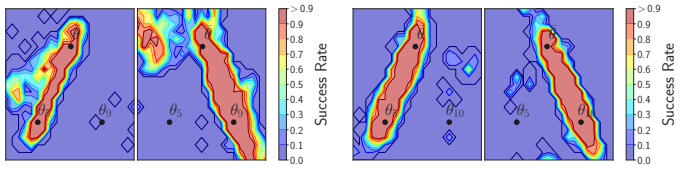
Figure 18: Contour plots representing the task objective landscape of the *door-open-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.



(a) drawer-open-v2 and drawer-close-v2

(b) drawer-open-v2 and button-press-topdown-v2

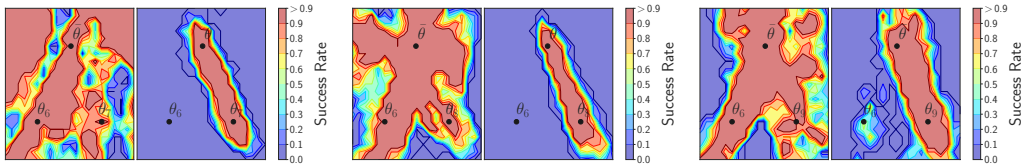
(c) drawer-open-v2 and peg-insert-side-v2



(d) drawer-open-v2 and window-open-v2

(e) drawer-open-v2 and window-close-v2

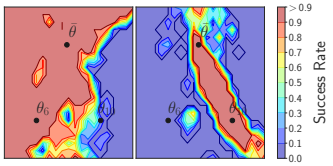
Figure 19: Contour plots representing the task objective landscape of the *drawer-open-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.



(a) drawer-close-v2 and button-press-topdown-v2

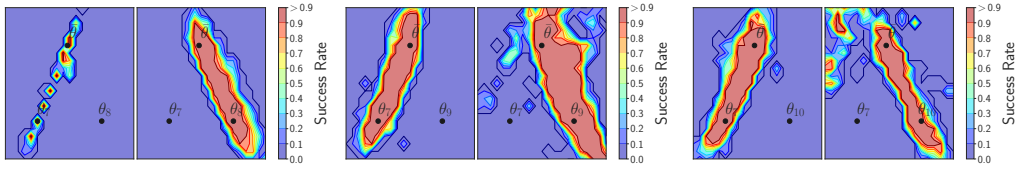
(b) drawer-close-v2 and peg-insert-side-v2

(c) drawer-close-v2 and window-open-v2



(d) drawer-close-v2 and window-close-v2

Figure 20: Contour plots representing the task objective landscape of the *drawer-close-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.

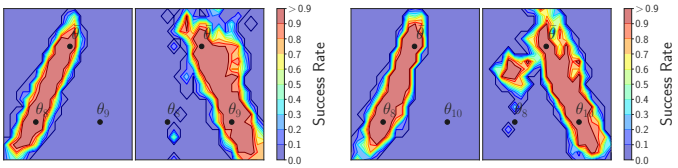


(a) button-press-topdown-v2 and peg-insert-side-v2

(b) button-press-topdown-v2 and window-open-v2

(c) button-press-topdown-v2 and window-close-v2

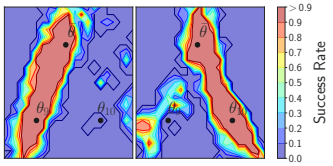
Figure 21: Contour plots representing the task objective landscape of the *button-press-topdown-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.



(a) peg-insert-side-v2 and window-open-v2

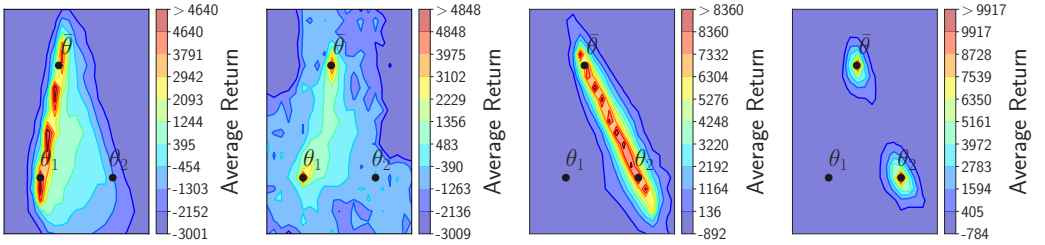
(b) peg-insert-side-v2 and window-close-v2

Figure 22: Contour plots representing the task objective landscape of the *peg-insert-side-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.



(a) window-open-v2 and window-close-v2

Figure 23: Contour plots representing the task objective landscape of the *window-open-v2* task and the remaining tasks from the MT10 scenario as a result of applying our approach, STAR.



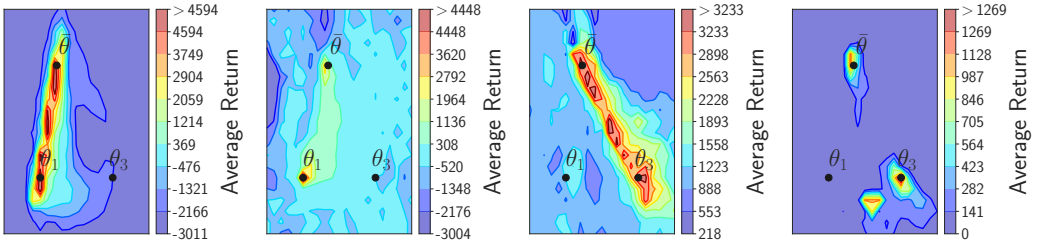
(a) Ant and STAR

(b) Ant and STAR w/o MC

(c) Half-Cheetah and STAR

(d) Half-Cheetah and STAR w/o MC

Figure 24: Contour plots representing the task objective landscapes of *Ant* and *Half-Cheetah* as a result of applying STAR and STAR w/o MC, respectively.



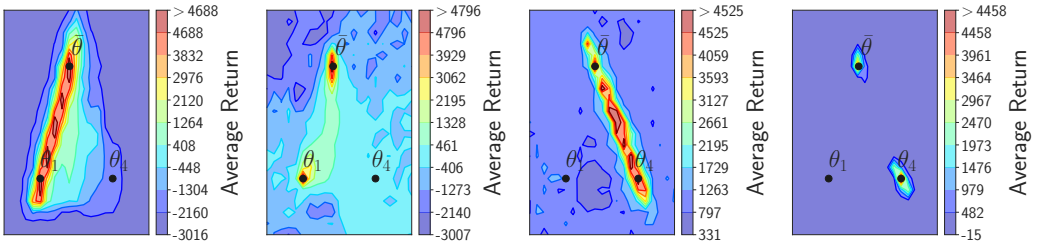
(a) Ant and STAR

(b) Ant and STAR w/o MC

(c) Hopper and STAR

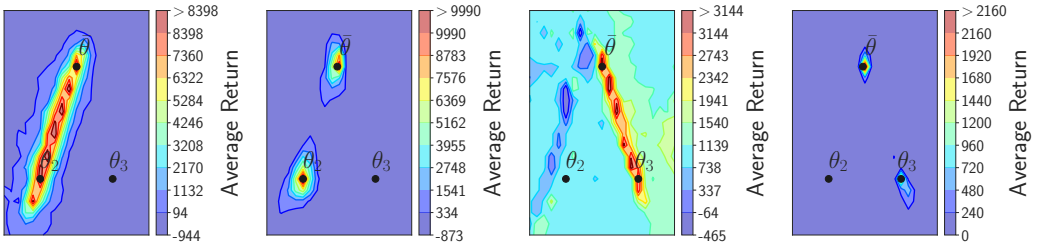
(d) Hopper and STAR w/o MC

Figure 25: Contour plots representing the task objective landscapes of *Ant* and *Hopper* as a result of applying STAR and STAR w/o MC, respectively.



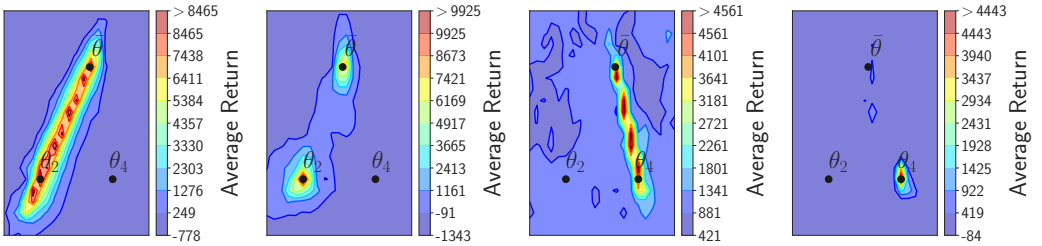
(a) Ant and STAR (b) Ant and STAR w/o MC (c) Walker and STAR (d) Walker and STAR w/o MC

Figure 26: Contour plots representing the task objective landscapes of *Ant* and *Walker* as a result of applying STAR and STAR w/o MC, respectively.



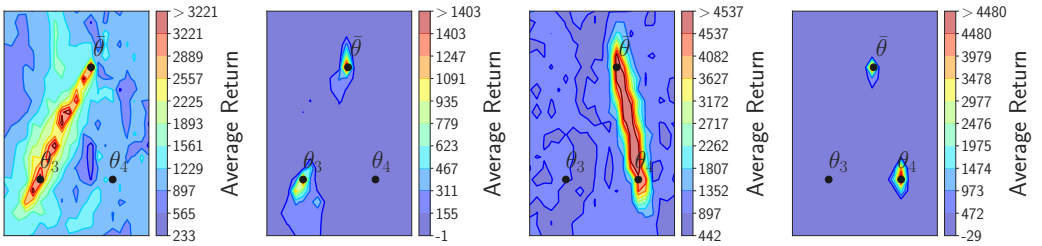
(a) Half-Cheetah and STAR (b) Half-Cheetah and STAR w/o MC (c) Hopper and STAR (d) Hopper and STAR w/o MC

Figure 27: Contour plots representing the task objective landscapes of *Half-Cheetah* and *Hopper* as a result of applying STAR and STAR w/o MC, respectively.



(a) Half-Cheetah and STAR (b) Half-Cheetah and STAR w/o MC (c) Walker and STAR (d) Walker and STAR w/o MC

Figure 28: Contour plots representing the task objective landscapes of *Half-Cheetah* and *Walker* as a result of applying STAR and STAR w/o MC, respectively.



(a) Hopper and STAR (b) Hopper and STAR w/o MC (c) Walker and STAR (d) Walker and STAR w/o MC

Figure 29: Contour plots representing the task objective landscapes of *Hopper* and *Walker* as a result of applying STAR and STAR w/o MC, respectively.

Algorithm 1 STAR

Input: $\{\theta_c\}_{c=1}^{|\mathcal{C}|}$ (single-task), $\bar{\theta}$ (multi-task), Q_ϕ (subspace critic).
Initialize: $\{\mathcal{R}_c\}_{c=1}^{|\mathcal{C}|}$ (replay buffers with random samples).
for $e = 1$ **to** E **do**
  Sample $\alpha \sim \mathcal{U}(0, 1)$ for exploration.
  Set $\theta_\alpha^c \leftarrow \alpha\, \theta_c + (1 - \alpha)\, \bar{\theta} \quad \forall c \in \mathcal{C}$.
  for $i = 1$ **to** H **do**
    Collect and store (s, a, r, s') in $\mathcal{R}_c$ by sampling $a \sim \pi_{\theta_\alpha^c}(.|s, c) \quad \forall c \in \mathcal{C}$.
    Sample $\mathcal{D}_c \sim \mathcal{R}_c\ \forall c \in \mathcal{C}$.
    Sample $\alpha \sim \mathcal{U}(0, 1)$ for training.
    Compute SAC MTRL objective: $J(\theta_1, ..., \theta_{|\mathcal{C}|}, \bar{\theta}, \phi)$ following Equations 4,5,6.
    Update subspace critic: $\phi \leftarrow \phi - \lambda \nabla_\phi J(..., \phi)$.
    Update single-task policies: $\theta_c \leftarrow \theta_c - \lambda \nabla_{\theta_c} J(..., \theta_c) \quad \forall c \in \mathcal{C}$.
    Update multi-task policy: $\bar{\theta} \leftarrow \bar{\theta} - \lambda \nabla_{\bar{\theta}} J(..., \bar{\theta})$
  end for
end for

C Algorithm Box and Code Snippet

This section highlights key implementation details of our approach, STAR. First, we present the main algorithmic workflow in Algorithm 1, which outlines the essential steps for obtaining a fully connected solution. While the algorithm corresponds specifically to STAR ($full$), we omit the suffix for brevity.

As detailed in Section 4.4, our implementation is built using JAX [28]. Code Snippet 1 showcases a core component of STAR: the efficient interpolation of expert parameters based on a given combination weight α. To enable this, we replace the standard Dense layer with a custom SubspaceDense layer that supports parallelized computation. Leveraging the linearity property, we note that a convex combination of weights and biases is mathematically equivalent to the convex combination of their corresponding activations. Therefore, in SubspaceDense, we first compute the activations from multiple Dense layers in parallel, all using the same input x and then combine the resulting outputs y according to the convex weight α.

Listing 1: Subspace Dense Layer

```
class SubspaceDense(nn.Module):
    features: int
    n_experts: int = 3
    kernel_init: Callable = default_init()
    dtype: Optional[Dtype] = jnp.float32

    @nn.compact
    def __call__(self, x: jnp.ndarray, alpha: jnp.ndarray) -> jnp.
        ndarray:

        y = nn.vmap(
            nn.Dense,
            in_axes=None,
            out_axes=-1,
            variable_axes={"params": 0},
            split_rngs={"params": True},
            axis_size=self.n_experts,
        )(
            self.features,
            kernel_init=self.kernel_init,
            dtype=self.dtype,
        )(x)

        y = y @ alpha[..., None]

        return jnp.squeeze(y, -1)
```

D Implementations Details

In this section, we highlight the hyperparameters used for the two MTRL setting studied in this work, namely Mujoco [19], and Metaworld [18].

D.1 MuJoco

For Mujoco [19], we consider four different continuous control environments: Ant, Half-Cheetah, Hopper, and Walker. They differ in the dimensionality of the state and action spaces. This requires a padding process to the maximum possible dimensionality among them, $\mathcal{S} \in \mathbb{R}^{27}$ and $\mathcal{A} \in \mathbb{R}^{8}$ for the state space and action space, respectively.

In this work, we use SAC [27] as the RL underlying algorithm. In particular, we adapt SAC to the MTRL setting as done in prior works [18, 4]. In Table 5, we report the general hyperparameters of the learning setting, the baselines, and our approach, STAR.

D.2 Metaworld

Metaworld [18] is a collection of various robotic manipulation tasks, each involving interaction with one or two objects. The state space includes several elements: the 3D position of the end effector, a normalized value indicating the gripper's openness, the 3D position of the first object, its quaternion representation (4D), and the 3D position along with the quaternion of the second object (set to zero if not required). Additionally, two consecutive data frames are stacked together, along with the 3D goal position, resulting in a 39-dimensional state space. Conversely, the action space remains consistent across tasks, defining the 3D displacement of the end effector and the normalized torque exerted by the gripper. To evaluate our approach, we conduct benchmarks on the MT10 and MT50 scenarios. In line with [21, 5], we introduce randomization to the goal or object positions across all tasks, referring to this variant as MT10-rand and MT50-rand. In Table 6, we report the general hyperparameters of the learning setting, the baselines, and our approach, STAR.

Hyperparameter	**Value**
Horizon	1000
Discount factor γ	0.99
Number of environments T	4
Steps per environment	1 step per 1 environment
Number of epochs	100
Steps per epoch (Evaluation Frequency)	1×10^4
Total Training Steps	1×10^6 steps per 1 environment
Train frequency	1
Number of test episodes	5
SAC [27] Hyperparameters	
Batch Size	512 per 1 environment
Critic Loss	Mean Squared Error
Number of Critic Networks	2
SAC Temperature	Automatic Tuned and Disentangled
Target Entropy	-dim($\mathcal{A}$)
Optimizer	Adam
Learning rate for Actor, Critic, Alpha	3×10^{-4}, 3×10^{-4}, 1×10^{-4}
Policy standard [min, max]	$[e^{-10}, e^2]$
Soft target interpolation	5×10^{-3}
Minimum Replay Buffer Size	3000
Replay Buffer Capacity	1×10^6 per 1 environment
MTSAC [18] and PCGrad [9] Hyperparameters	
Actor Network $\pi(.\|s, c)$	[256, 256, dim($\mathcal{A}$) $\times$ 2)]
Critic Network $Q(s, a, c)$	[256, 256, 1]
Activation function	{Actor: ReLU, Critic: ReLU}
MH-MTSAC [18] Hyperparameters	
Actor Network $\pi(.\|s, c)$	[256, 256]
Actor Output Layer	Multi-Head (Shared Bottom), $T \times$ [dim($\mathcal{A}$) $\times$ 2]
Critic Network $Q(s, a, c)$	[256, 256]
Critic Output Layer	Multi-Head (Shared Bottom), $T \times [1]$
Activation function	{Actor: ReLU, Critic: ReLU}
MOORE and MOE [6] Hyperparameters	
Number of Experts k	3
Actor Network $\pi(.\|s, c)$	$k \times [256, 256]$
Actor Output Layer	Multi-Head (Shared Bottom), $T \times$ [dim($\mathcal{A}$) $\times$ 2]
Critic Network $Q(s, a, c)$	$k \times [256, 256]$
Critic Output Layer	Multi-Head (Shared Bottom), $T \times [1]$
Activation function	{Actor: ReLU, Critic: ReLU}
STAR Hyperparameters	
Number of Sampled α	1
Number of Experts k	3 (2 Group-specific + Multi-Task Policy)
Actor Network $\pi_{\theta_\alpha}(.\|s, c)$	$k \times [256, 256,$ dim($\mathcal{A}$) $\times$ 2]
Critic Network $Q(s, a, c, \alpha)$	$k - 1 \times [256, 256, 1]$
Activation function	{Actor: ReLU, Critic: ReLU}

Table 5: MuJoco Hyperparameters.

Hyperparameter	**Value**
Horizon	150
Discount factor γ	0.99
Number of environments T	$\{\text{MT10} : 10, \text{MT50} : 50\}$
Steps per environment	1 step per 1 environment
Total Training Steps	2×10^6 steps per 1 environment
Train frequency	1
Number of test episodes	10
SAC [27] Hyperparameters	
Batch Size	128 per 1 environment
Critic Loss	Mean Squared Error
Number of Critic Networks	2
SAC Temperature	Automatic Tuned and Disentangled
Target Entropy	-dim($\mathcal{A}$)
Optimizer	Adam
Learning rate for Actor, Critic, Alpha	3×10^{-4}, 3×10^{-4}, 1×10^{-4}
Policy standard [min, max]	$[e^{-10}, e^2]$
Soft target interpolation	5×10^{-3}
Minimum Replay Buffer Size	1500
Replay Buffer Capacity	$\{\text{MT10} : 1 \times 10^6, \text{MT50} : 1 \times 10^5\}$ per 1 environment
MTSAC [18] and PCGrad [9] Hyperparameters	
Actor Network $\pi(.\|s, c)$	[400, 400, 400, dim($\mathcal{A}$) $\times$ 2)]
Critic Network $Q(s, a, c)$	[400, 400, 400, 1]
Activation function	{Actor: ReLU, Critic: ReLU}
MH-MTSAC [18] Hyperparameters	
Actor Network $\pi(.\|s, c)$	[400, 400, 400]
Actor Output Layer	Multi-Head (Shared Bottom), $T \times [\text{dim}(\mathcal{A}) \times 2]$
Critic Network $Q(s, a, c)$	[400, 400, 400]
Critic Output Layer	Multi-Head (Shared Bottom), $T \times [1]$
Activation function	{Actor: ReLU, Critic: ReLU (Tanh for MT50)}
FiLM [30] Hyperparameters	
Task Encoder Network	$[100, 50, 50, 50, 6]$
State Encoder	$[50, 50, 50]$
Actor Network $\pi(.\|s, c)$	$[400, 400, 400, \text{dim}(\mathcal{A}) \times 2]$
Critic Network $Q(s, a, c)$	$[400, 400, 400, 1]$
Activation function	{Actor: ReLU, Critic: Tanh}
CARE [4] Hyperparameters	
Task Encoder Network	$[100, 50, 50, 50, 50]$
Number of Experts (State Encoders) k	$\{\text{MT10} : 6, \text{MT50} : 10\}$
State Encoders	$k \times [50, 50, 50]$
Actor Network $\pi(.\|s, c)$	$[400, 400, 400, \text{dim}(\mathcal{A}) \times 2]$
Critic Network $Q(s, a, c)$	$[400, 400, 400, 1]$
Activation function	{Actor: ReLU, Critic: Tanh}
MOORE and MOE [6] Hyperparameters	
Number of Experts k	4
Actor Network $\pi(.\|s, c)$	$k \times [400, 400, 400]$
Actor Output Layer	Multi-Head (Shared Bottom), $T \times [\text{dim}(\mathcal{A}) \times 2]$
Critic Network $Q(s, a, c)$	$k \times [400, 400, 400]$
Critic Output Layer	Multi-Head (Shared Bottom), $T \times [1]$
Activation function	{Actor: ReLU, Critic: ReLU (Tanh for MOE on MT50)}
STAR Hyperparameters	
Number of Sampled α	4
Number of Experts k	4 (3 Group-specific + Multi-Task Policy)
Actor Network $\pi_{\theta_\alpha}(.\|s, c)$	$k \times [400, 400, 400, \text{dim}(\mathcal{A}) \times 2]$
Critic Network $Q(s, a, c, \alpha)$	$k - 1 \times [400, 400, 400, 1]$
Activation function	{Actor: ReLU, Critic: ReLU (Tanh for MT50)}

Table 6: Metaworld Hyperparameters.