Scalable Multi-Objective and Meta Reinforcement Learning via Gradient Estimation

Zhenshuo Zhang, Minxuan Duan, Youran Ye, and Hongyang R. Zhang

Northeastern University, Boston, Massachusetts
Email: {zhang.zhens, duan.mi, ye.you, ho.zhang}@northeastern.edu

December 10, 2025

Abstract

We study the problem of efficiently estimating policies that simultaneously optimize multiple objectives in reinforcement learning (RL). Given n objectives (or tasks), we seek the optimal partition of these objectives into $k \ll n$ groups, where each group comprises related objectives that can be trained together. This problem arises in applications such as robotics, control, and preference optimization in language models, where learning a single policy for all n objectives is suboptimal as n grows. We introduce a two-stage procedure—meta-training followed by fine-tuning—to address this problem. We first learn a meta-policy for all objectives using multitask learning. Then, we adapt the meta-policy to multiple randomly sampled subsets of objectives. The adaptation step leverages a first-order approximation property of well-trained policy networks, which is empirically verified to be accurate within a 2% error margin across various RL environments. The resulting algorithm, PolicyGradEx, efficiently estimates an aggregate task-affinity score matrix given a policy evaluation algorithm. Based on the estimated affinity score matrix, we cluster the n objectives into k groups by maximizing the intra-cluster affinity scores. Experiments on three robotic control and the Meta-World benchmarks demonstrate that our approach outperforms state-of-the-art baselines by 16% on average, while delivering up to 26× faster speedup relative to performing full training to obtain the clusters. Ablation studies validate each component of our approach. For instance, compared with random grouping and gradient-similarity-based grouping, our loss-based clustering yields an improvement of 19%. Finally, we analyze the generalization error of policy networks by measuring the Hessian trace of the loss surface, which gives non-vacuous measures relative to the observed generalization errors.

1 Introduction

Reinforcement learning (RL) is a technique for sequential decision-making in interactive environments, enabling agents to learn from feedback signals. A central challenge in RL is to develop agents that generalize across a wide range of tasks rather than solving a single task in isolation. For instance, a robot needs to master multiple related skills that share an underlying structure [43]. Prior work has studied this problem for two closely related settings. The first, multitask reinforcement learning, seeks to find a single policy that performs well across a given set of objectives (or tasks) [30, 13]. The second, meta-reinforcement learning, aims to learn a meta-initialization that can rapidly adapt to unseen tasks [7, 27]. In both settings, however, the computational cost of evaluating and adapting a shared policy over many competing objectives grows quickly with n.

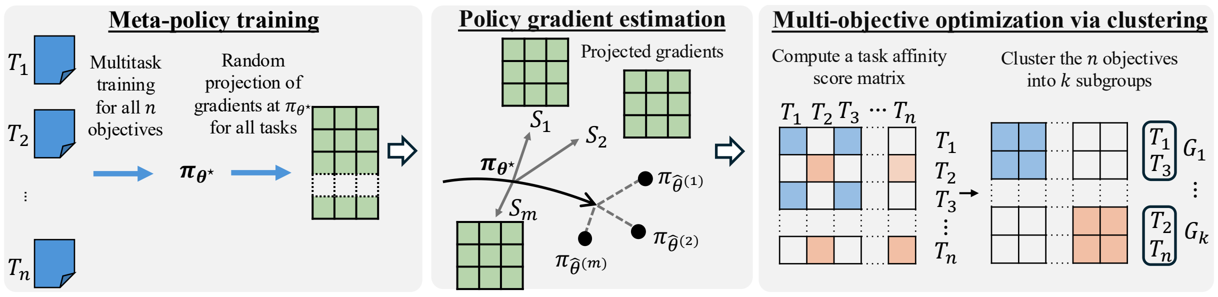

Meta-policy training
T_1
T_2
T_n
Multitask training for all n objectives
Random projection of gradients at $\pi_{\theta^\star}$ for all tasks
$\pi_{\theta^\star}$
Policy gradient estimation
Projected gradients
S_1
S_2
S_m
$\pi_{\theta^\star}$
$\pi_{\hat{\theta}^{(1)}}$
$\pi_{\hat{\theta}^{(2)}}$
$\pi_{\hat{\theta}^{(m)}}$
Multi-objective optimization via clustering
Compute a task affinity score matrix
Cluster the n objectives into k subgroups
T_1 T_2 T_3 $\cdots$ T_n
G_1
G_k

Figure 1: An overview of our approach. *Left:* Run multitask training on all the tasks, $T_1, T_2, \ldots, T_n$, and obtain to a meta-initialization policy $\pi_{\theta^\star}$. Store the projected gradients of a surrogate loss with meta-policy $\theta^\star$ for every transition from $t = 1, 2, \ldots, N$. *Middle:* Estimate policy adaptation performance on m task subsets using projected gradients as features in logistic regression. *Right:* Compute an $n \times n$ task affinity score matrix based on the estimated loss values. Lastly, run a clustering algorithm to group similar objectives, resulting in k subgroups $G_1, \ldots, G_k$, each of which share the same policy within group.

This paper studies the problem of designing a (scalable) multi-objective optimizer for reinforcement learning, with extensions to policy meta-adaptation. Learning from a diverse set of tasks has been shown to improve generalization [33, 4, 41]. However, the main difficulty is modeling task relationships and how different tasks transfer information to one another when trained together in a complex network [38]. Existing approaches have considered measuring the similarity between gradient vectors across tasks to quantify task relationships [42]. Notably, this gradient-similarity-based measure corresponds to pairwise task-affinity scores, whereas in practice it is natural to model how multiple policies interact when trained together [19]. Exhaustively searching over all subsets of $\{1, 2, \ldots, n\}$ is computationally infeasible: evaluating each subset typically requires a full training procedure, and the number of subsets grows exponentially in n [2]. Greedy stepwise selection requires evaluation on $O(n^2)$ subsets, which is again impractical for large n [18, 20].

To overcome these challenges, we introduce a scalable algorithm for estimating policy performance on arbitrary task subsets without full training. Our algorithm starts by learning a single meta-policy across all tasks, then uses a first-order surrogate model—derived from a Taylor approximation around the meta-policy to efficiently estimate the outcome of fine-tuning on any subset. This approach is analogous to a random ensemble [45], which allows us to compute a pairwise task affinity matrix and partition the n objectives into k subgroups using a convex clustering procedure. The resulting groups can be trained separately using any multitask or meta-RL optimizers. This overall approach is illustrated in Figure 1.

We validate our approach through extensive experiments on both Meta-World [43] and robotic control benchmarks [34]. We find that the first-order approximation applied to the outputs of a policy network yields accurate estimates—typically within 2% error margin—while being up to 26× faster than full training. Our algorithm consistently outperforms multitask and meta-RL baselines [40, 30, 31] by 16% on average, and surpasses baseline grouping strategies (such as random grouping or gradient-similarity based grouping) by 23% and 16%, respectively. These results highlight the effectiveness and scalability of our task-affinity estimation approach.

Lastly, we conduct a theoretical analysis of the generalization errors of policy networks. The main technical approach is to compute (and measure) the trace of the Hessian of the loss surface [15, 14, 44]. See Theorem 1 for the precise statement. When evaluated on the above RL environments, we find that the Hessian-based generalization bounds match the scale of empirical generalization errors (see Figure 2).

In summary, the contributions of this paper are threefold. (1) We propose a first-order gradient estimation algorithm to estimate policy performance on any given task subset efficiently. Based on this gradient estimation, we design a scalable algorithm that computes an $n \times n$ task affinity score matrix and partitions n objectives into k clusters via convex relaxation. (2) We conduct extensive experiments to validate the efficiency of our approach for several multi-objective and meta reinforcement learning benchmarks. The code for reproducing our results can be accessed at: https://github.com/VirtuosoResearch/PolicyGradEx. (3) Finally, we show non-vacuous bounds on the generalization error of the loss surface of policy networks via a PAC-Bayes analysis, and report empirical estimates of the Hessian trace.

2 Problem Setup

We consider a Markov decision process in which an agent must learn a set of n tasks $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$. Each task T_i is modeled as a Markov decision process (MDP) $T_i = (\mathcal{S}, \mathcal{A}, P, P_0, r_i, \gamma)$, where all tasks share the same state space $\mathcal{S}$, action space $\mathcal{A}$, transition model P, and initial-state distribution P_0, discount factor γ, but differ in their reward functions r_i. For a policy π_θ with parameters θ, we define its task-specific expected reward as

$$R_i(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \, r_i(s_t, a_t)\right].$$

In multi-objective RL, the goal is to learn a single policy that performs well across all tasks. A natural aggregate objective is the average reward

$$R_{\mathcal{T}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} R_i(\theta).$$

More generally, for any subset $S \subseteq T$, we define

$$R_S(\theta) = \frac{1}{|S|}\sum_{T_i \in S} R_i(\theta).$$

Pairwise task interactions can be characterized through this notion. For example, $R_{\{i,j\}}(\theta)$ captures the joint performance on tasks T_i and T_j, providing a meaningful measure of their affinity. A key challenge in multitask learning is *negative transfer*, where conflicting gradients from two tasks degrade optimization efficiency and final performance [38]. This occurs when

$$R_{\{i,j\}}(\theta) < \left(R_i(\theta) + R_j(\theta)\right)/2,$$

indicating that training the pair together is detrimental compared to training each task separately.

Our goal is to model the underlying relationship between the n tasks, which can be used to group similar tasks or select a small representative subset for meta-learning. However, searching for the optimal partition requires evaluating $R_S(\theta)$ for multiple subsets $S \subseteq \mathcal{T}$, since the number of possible subsets is 2^n. Greedy stepwise selection again requires evaluating $\mathcal{O}(n^2)$ subsets, where each evaluation typically involves a full RL training or adaptation procedure [18, 21]. This is again intractable for large n.

To address this challenge, we introduce **an efficient gradient-estimation algorithm to approximate the loss of any subset trained on a policy network without repeated training**. This enables us to scale up multi-objective RL to a large number of objectives n, as we will describe next.

3 Our Approach

This section describes our proposed algorithm for multi-objective reinforcement learning. The algorithm consists of two main stages. We first find a meta-policy $\theta^\star$ trained using all tasks. Then, compute a surrogate model to estimate the adapted performance of $\theta^\star$ on a subset of $\{1, 2, \dots, n\}$. This surrogate modeling framework enables efficient estimation of subset combinations without repeated training. In the second stage, we estimate the fine-tuning performance R_{S_i} on m randomly chosen subsets $S_1, S_2, \dots, S_m$ of $\mathcal{T}$, for every $1 \le i \le m$. For each subset, we solve a weighted logistic regression problem to estimate the adaptation parameters $\hat{\theta}_{S_i}$ and then estimate an approximate loss. Importantly, we run this second step using precomputed gradients and functional values at $\theta^\star$ from the first stage. The key technique that enables this gradient estimation is **a first-order approximation property of the policy reward** that we also empirically verify for various RL settings.

3.1 Surrogate Modeling

Our goal is to estimate the output of a policy update starting from the meta-initialization $\theta^\star$. To do so efficiently, we linearize the policy gradient objective around $\theta^\star$ and reformulate the one-step update as a weighted logistic regression problem.

We begin with the standard policy gradient objective used in algorithms such as Proximal Policy Optimization (PPO) [29]:

$$J(\theta) = \mathbb{E}_t\left[r_t(\theta)\hat{A}_t\right], \text{ where } r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^\star}(a_t|s_t)}$$

is the probability ratio and $\hat{A}_t$ is the estimated reward for the state-action pair (s_t, a_t) at time t. We perform a first-order Taylor expansion of the log-probability $\log \pi_\theta(a_t|s_t)$ around the initial parameters $\theta^\star$. Let $\Delta\theta = \theta - \theta^\star$ and define the gradient feature vector

$$g_t = \nabla \log \pi_\theta(a_t|s_t)\big|_{\theta=\theta^\star}.$$

The first-order Taylor expansion of the above log-probability-ratio is given by:

$$\begin{aligned} \log r_t(\theta) &= \log \pi_\theta(a_t|s_t) - \log \pi_{\theta^\star}(a_t|s_t) \\ &= g_t^\top \Delta\theta + \epsilon, \end{aligned} \tag{1}$$

where ϵ refers to the approximation error of the expansion. Provided that $\Delta\theta$ is small relative to $\theta^\star$, we have another approximation as (with an error of $(1 - r_t)^2/2$)

$$r_t(\theta) \approx 1 + \log r_t(\theta). \tag{2}$$

By applying equations (1) and (2) back to $J(\theta)$, we have:

$$J(\theta) \approx \mathbb{E}_t\left[\hat{A}_t(1 + g_t^\top \Delta\theta)\right].$$

Since $\mathbb{E}_t[\hat{A}_t]$ does not depend on $\Delta\theta$, maximizing $J(\theta)$ reduces to maximizing the following which is linear in θ:

$$\mathbb{E}_t[\hat{A}_t g_t^\top (\theta - \theta^\star)].$$

Surrogate loss. Next, we turn the above reward maximization on $J(\theta)$ into a weighted binary classification problem. For every $(s_t, a_t, \hat{A}_t)$, for any $t \in \{1, 2, \dots, N\}$, define:

Algorithm 1 Policy gradient estimation (PolicyGradEx)

```
Input: $m$ random subsets from $\mathcal{T}$, $S_1, S_2, \dots, S_m$
Require: Initial meta-policy parameter $\theta^\star \in \mathbb{R}^p$; Number of episodes $N$; Projection dimension $d$
 1: $\pi_{\theta^\star} \leftarrow$ Setup the meta-policy for all tasks in $\mathcal{T}$
 2: $S_1, S_2, \dots, S_m \leftarrow m$ random subsets of $\mathcal{T}$ with size $\alpha$
 3: $P \leftarrow$ A $p$ by $d$ isotropic Gaussian random projection matrix
 4: for $t = 1, \dots, N$ do
 5:     for $T_i \in \mathcal{T}$ do
 6:         $g_{i,t} \leftarrow P^\top \nabla \pi_{\theta^\star}(a_t|s_t)$
 7:         $w_{i,t} \leftarrow |\hat{A}_t|$
 8:         $y_{i,t} \leftarrow \text{sign}(\hat{A}_t)$
 9:     end for
10: end for
11: for $j = 1, \dots, m$ do
12:     $\Delta\hat{\theta}_d \leftarrow \arg\min_{\Delta\theta \in \mathbb{R}^d} \hat{L}_{S_j}(\theta^\star + P\Delta\theta)$
13:     $\hat{\theta}^{(j)} \leftarrow \theta^\star + P\Delta\hat{\theta}_d$
14:     $\hat{f}(S_j) \leftarrow -\hat{L}_{S_j}(\hat{\theta}^{(j)})$
15: end for
16: Return $\hat{f}(S_1), \dots, \hat{f}(S_m)$
```

- Target binary label $y_t = \text{sign}(\hat{A}_t) \in \{-1, +1\}$, indicating if the action was better or worse than zero.
- Classifier score $z_t = g_t^\top \Delta\theta$, linear score for the update.
- Sample weight $w_t = |\hat{A}_t|$, magnitude of the reward.

This yields a per-sample surrogate loss as follows:

$$\ell(g_t, y_t, w_t; \Delta\theta) = w_t \cdot \log(1 + (-y_t(g_t^\top \Delta\theta))).$$

For a given task subset S, let $\mathcal{D}_S$ denote the set of trajectory samples $(g_{i,t}, y_{i,t}, w_{i,t})$ from all the tasks $T_i \in S$, across all the samples (with an extra index i). The average loss for the subset S is defined by the average loss over all the samples from all the tasks within the subset S:

$$\hat{L}_S(\theta) = \frac{1}{|\mathcal{D}_S|} \sum_{(g,y,w) \in \mathcal{D}_S} \ell(g, y, w; \theta). \tag{3}$$

Minimizing this loss over $\Delta\theta = \theta - \theta^\star$ yields the estimated adaptation for the subset S from the meta-policy $\theta^\star$.

Random projection. Since the gradient vectors $g_{i,t} \in \mathbb{R}^p$ may be high-dimensional (with p in the order of millions), we apply random projections to reduce the dimension down to a few hundred (in practice, $d = 400$ suffices). This projection provably preserves the pairwise similarity between all pairs of gradient vectors via the Johnson–Lindenstrauss Lemma [12]. We sample a random projection matrix $P \in \mathbb{R}^{p \times d}$, where $d \ll p$ and every entry of P is independently drawn from a Gaussian $\mathcal{N}(0, d^{-1})$. We project the high-dimensional gradients to a d-dimensional space as

$$\tilde{g}_{i,t} = P^\top g_{i,t}.$$

Table 1: We empirically find that the approximation error of ϵ is negligible for several interactive RL environments. We report the mean and standard deviation of the approximation error across 10 random subsets. We defer a detailed description of the setup to the experiments section.

Distance	MT10	CartPole	Highway	Lunarlander
0.1%	$0.01_{\pm 0.01}$%	$0.12_{\pm 0.14}$%	$0.02_{\pm 0.02}$%	$0.06_{\pm 0.01}$%
0.5%	$0.43_{\pm 0.73}$%	$0.73_{\pm 0.10}$%	$0.11_{\pm 0.09}$%	$0.03_{\pm 0.02}$%
1.0%	$0.32_{\pm 0.56}$%	$0.98_{\pm 0.65}$%	$2.04_{\pm 0.58}$%	$0.48_{\pm 0.01}$%

Then, we solve a logistic regression problem in the d-dimensional space with $\theta_d \in \mathbb{R}^d$:

$$\widehat{\theta}_d \leftarrow \arg\min_{\theta_d \in \mathbb{R}^d} \frac{1}{|\mathcal{D}_{S_j}|} \sum_{(g,y,w)\in \mathcal{D}_{S_j}} \ell(\tilde{g}, y, w; \theta_d), \tag{4}$$

for any $j = 1, 2, \ldots, m$. Finally, we turn the solution from dimension d back to the original dimension p as

$$\widehat{\theta}^{(j)} = \theta^\star + P\widehat{\theta}_d.$$

The complete description of this surrogate modeling procedure is provided in Algorithm 1.

3.2 Evaluation of First-Order Policy Approximation

A critical assumption is the accuracy of the approximation in equation (1). Here we observe that the approximation error remains negligible for θ around $\theta^\star$. We evaluate this approximation by measuring the relative residual sum of squares (RSS) error across multiple RL environments. Specifically, we compute:

$$\frac{\left(\log \pi_\theta(a_t|s_t) - \log \pi_{\theta^\star}(a_t|s_t) - g_t^\top \Delta\theta\right)^2}{(\log \pi_\theta(a_t|s_t))^2},$$

where $\Delta\theta = \theta - \theta^\star$ and $g_t = \nabla \log \pi_\theta(a_t|s_t)\big|_{\theta=\theta^\star}$.

We begin by verifying that adapted policies remain close to the meta-initialization $\theta^\star$. Using ten randomly sampled subsets, we evaluate the relative distance $\frac{\|\theta-\theta^\star\|_F}{\|\theta^\star\|_F}$, confirming that adaptation stays within a local neighborhood where the approximation is valid.

We report the RSS error measured on MT10 from the MetaWorld benchmark, CartPole, LunarLander, and Highway in Table 1. In these environments, the policy, a four-layer MLP, is trained to perform various control tasks by selecting an action based on its output given the states. We sample 2048 steps per task to obtain the gradients. The results are averaged over 10 randomly sampled subsets of size 5. For a proper initialization $\theta^\star$, the approximation error is less than 2% when the updated policy θ remains close to the initial policy. This verifies that the first-order expansion provides a sufficiently accurate local model for surrogate-based policy estimation. The approximation error increases from less than 2% to up to 10% when the policy's parameter distance from initialization grows to around 5%, defining the boundaries of our method's applicability.

3.3 Task Affinity Grouping

We next quantify the pairwise relationships between tasks by constructing a task affinity matrix $U \in \mathbb{R}^{n\times n}$, where each entry $U_{i,j}$ captures the collaborative behavior between tasks T_i and T_j. Larger values indicate that jointly fine-tuning on the two tasks produces higher surrogate performance.

Algorithm 2 Clustering related objectives into subgroups using the task affinity score matrix

```
Input: $n$ tasks $\mathcal{T}$; number of desired clusters $k$
Require: Number of subsets $m$ with size $\alpha$; Regularization parameter $\lambda$; Number of episodes $N$; Projected dimension $d$
Output: A disjoint partition of $\{1, 2, \dots, n\}$ into $k$ groups
1: $\theta^\star \leftarrow$ Train a meta-initialization policy with $\mathcal{T}$
2: $S_1, \dots, S_m \leftarrow$ Sample $m$ subsets of size $\alpha$ from $\mathcal{T}$
3: $\hat{f}(S_1), \hat{f}(S_2), \dots, \hat{f}(S_m) \leftarrow$ Apply PolicyGradEx with $(S_1, S_2, \dots, S_m; \theta^\star; N, d)$
4: $U \leftarrow$ An $n \times n$ affinity score matrix via equation (5)
5: $X \leftarrow$ Solve a convex relaxation program (7) with $U$ and regularization parameter $\lambda$
6: $G_1, G_2, \dots, G_k \leftarrow$ Round $X$ into $k$ subgroups
```

A naive approach is to repeatedly train a model for all the $\binom{n}{2}$ subsets, which is impractical. Instead, we sample m random subsets $\{S_1, \dots, S_m\}$ and use the surrogate model to compute their estimated performance $\hat{f}(S_i) = -\hat{L}_{S_i}(\hat{\theta}_i)$, for all $i = 1, 2, \dots, m$ based on Algorithm 1. We then compute an $n \times n$ task affinity score matrix as follows:

$$U_{i,j} = \frac{1}{n_{i,j}} \sum_{1 \le l \le m : \{T_i, T_j\} \in S_l} \hat{f}(S_l), \tag{5}$$

for all i and j in $\{1, 2, \dots, n\}$, where $n_{i,j}$ is the number of sampled subsets containing both T_i and T_j. As a remark, provided that $m = O(n^2)$, then $n_{i,j}$ must be nonzero with high probability, for all i and j in $\{1, 2, \dots, n\}$.

We then cluster tasks by solving a convex optimization problem that maximizes intra-cluster affinity, following a trace-regularized relaxation [1]. This step is very fast since it runs on an n by n matrix with n being at most several hundred, which can typically be solved in just a few seconds. Moreover, this procedure can be repeated with different values of λ to determine the optimal number of clusters k for downstream performance. For brevity, we defer a complete description of the convex relaxation program to the Appendix. The whole procedure is described in Algorithm 2.

4 Experiments

We conduct a comprehensive set of experiments to validate PolicyGradEx and its application across various multi-objective reinforcement learning benchmarks. Our evaluation is designed to answer two key questions: (1) Does our surrogate model accurately and efficiently approximate the outcome of full policy training on various task subsets? (2) Do task groups identified by our algorithm lead to superior performance in downstream multitask and meta-RL evaluations compared to state-of-the-art and heuristic baselines?

Our finding shows that our surrogate modeling approach achieves high accuracy in estimating multitask RL performance, with over 0.73 normalized mutual information relative to the ground-truth, while reducing the number of floating-point operations (FLOPs) by up to 26×. In downstream evaluations, PolicyGradEx outperforms existing multitask optimizers by 19% in multitask RL benchmarks and achieves a 13% improvement in the meta-RL setting. Furthermore, we compare our task affinity grouping algorithm to random grouping and gradient-similarity based grouping, showing that PolicyGradEx achieves a 19% improvement over both. Ablation analysis validates the use of random projections and the choice of k in the algorithm.

Table 2: Normalized Mutual Information (NMI) between estimated and actual clusters, measured on two environments trained with a multi-layer perceptron (MLP). In the last column, the speedup is measured as the ratio of the FLOP count between full training and PolicyGradEx.

# MLP layers	Meta-World	LunarLander	Speedup
2	0.76	0.73	21×
4	0.76	0.73	24×
8	0.76	0.73	26×

4.1 Experimental Setup

Environments. We evaluate our method on two types of benchmarks. First, we use MT10 from the Meta-World benchmark [43], which consists of 10 diverse robotic manipulation tasks. While these tasks share common state and action spaces, their reward functions and dynamics vary (e.g., a goal position in one task may correspond to an object's position in another). Second, we use three classic control environments from Gymnasium [34] and MO-Gymnasium [6]: CartPole, Highway, and LunarLander. For each, we generate a distribution of 10 source tasks by altering key physical parameters (e.g., pole length for CartPole, traffic density for Highway, and gravity for LunarLander).

Baselines. We compare our method against two categories of baselines. (1) Multi-objective RL baselines: These represent standard and state-of-the-art multitask approaches for multitasks training: a single policy trained on all tasks; Soft Modularization [40], which employs a routing-based selection mechanism; PaCo [31], an ensemble method that composes policies from a shared parameter subspace; and CARE [30], which performs inference using contextual information to adapt policies. (2) Grouping baselines: These methods use the same number of groups as our approach but employ different heuristic grouping procedures. This allows us to isolate the benefit of our affinity modeling. We include random grouping (where tasks are randomly assigned to k groups) and gradient-similarity-based grouping (where tasks are clustered based on the cosine similarity of their respective policy gradients).

Implementations. For the MT10 benchmark, we group the $n = 10$ tasks into $k = 3$ subsets and train a separate policy for each subgroup using soft modularization. We measure performance using the average success rate per task.

For the control environments, we form the meta-training set by randomly selecting one from each task group provided by PolicyGradEx. We then use MAML [7] to train a meta-policy on the assigned tasks.

The final performance is measured by the average reward after 200 adaptation steps on 50 unseen target tasks. All experiments are conducted using PyTorch on an Ubuntu server with an Intel Xeon E5-2623 CPU and an NVIDIA Quadro RTX 6000 GPU.

4.2 Results for Loss Function Estimation

We first evaluate both the approximation accuracy and computational cost of PolicyGradEx. We establish ground-truth subset clusters based on task-affinity scores derived from rewards obtained by training policies on different task subsets. We then use Normalized Mutual Information (NMI) to quantify the accuracy of the subsets estimated by PolicyGradEx against this ground truth.

As shown in Table 2, on both MetaWorld and LunarLander, PolicyGradEx identifies subset clusters that achieve over 0.73 similarity to those obtained from full-policy training, while reducing FLOPs by a factor of 26×. By contrast, the NMI under random clustering is approximately 0.2.

Table 3: Comparison of our approach with several baseline methods. We report the average success rate on the Meta-World benchmark and the rewards (after adaptation) for three robotic control environments. As for the meta-RL setting, we several a subset of tasks for which we set as the source transfer tasks. We note that CARE does not apply to the control tasks in the meta-RL setting. We report the mean and standard deviation from five runs.

RL Environment	Meta-World	CartPole	Highway	LunarLander
Multi-task training [43]	$71.3_{\pm 1.2}\%$	$145.9_{\pm 9.0}$	$140.0_{\pm 4.6}$	$53.8_{\pm 14.6}$
Soft modularization [40]	$82.0_{\pm 1.1}\%$	$139.3_{\pm 9.5}$	$141.3_{\pm 3.5}$	$66.1_{\pm 13.0}$
PaCo [31]	$73.1_{\pm 1.1}\%$	$144.5_{\pm 5.2}$	$136.6_{\pm 6.4}$	$62.6_{\pm 11.4}$
CARE [30]	$84.0_{\pm 1.8}\%$	/	/	/
Randomly assign each task into k groups	$58.2_{\pm 6.2}\%$	$144.1_{\pm 10.1}$	$143.4_{\pm 5.4}$	$73.1_{\pm 11.7}$
Gradient-similarity-based grouping [42]	$69.6_{\pm 1.9}\%$	$142.0_{\pm 8.2}$	$135.6_{\pm 7.6}$	$80.8_{\pm 6.9}$
Algorithm 2 (This paper)	$\mathbf{94.0}_{\pm \mathbf{2.8}}\%$	$\mathbf{159.2}_{\pm 3.8}$	$\mathbf{153.5}_{\pm 7.8}$	$\mathbf{82.8}_{\pm 6.9}$

Moreover, the relative error between the estimated task affinity matrix and the ground-truth matrix is at most 0.2.

4.3 Results for Multi-Objective Optimization

We report the evaluation results in Table 3. On the Meta-World benchmark, our approach improves the average success rate by 21% compared to multitask optimizers. It also achieves a 62% improvement over training with random groups and a 35% improvement over training with gradient-similarity-based groups. This observation suggests that random grouping and gradient similarity-based grouping in Meta-World tasks can result in negative transfer. In contrast, PolicyGradEx selects task groups with positive transferability, thereby improving the final success rate.

Next, we evaluate the ability of PolicyGradEx to select a representative task subset for meta-learning in the three control RL environments. Using MAML as the meta-learner, our approach improves the final adapted reward by 7% compared to multitask optimizers.

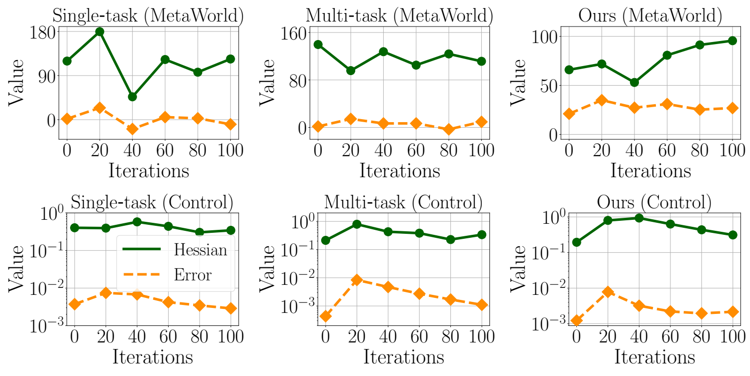
Finally, we compare with two baseline meta-training settings: training on all available source tasks and training on randomly grouped tasks. Our approach achieves a 13% improvement over both baselines and a 9% improvement compared to gradient-similarity-based grouping.

4.4 Generalization Error Measurements via Hessians

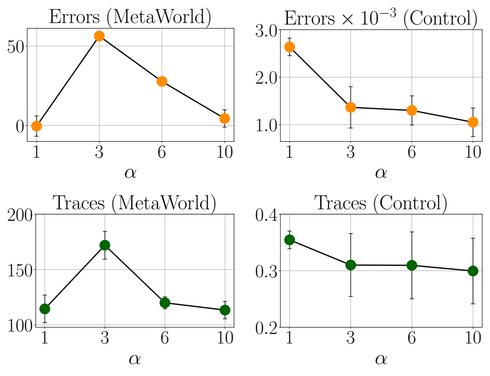
Next, we analyze the generalization behavior. by evaluating a sharpness measure based on the Hessian trace of the test loss for the policy network and the policy gradient loss. A smaller trace value suggests a flatter loss landscape. We implement Hutchinson's estimator and leverage a faster version of this estimator [25] to estimate the Hessian trace of RL models. We compute the policy gradient loss from the target tasks and measure the difference between the training loss and test loss as the generalization error. Results for the largest eigenvalue of the Hessian are similar.

First, we evaluate the Hessian trace and the policy generalization error between single-task training, multitask training on ten tasks, and training on a task group selected by PolicyGradEx. We begin training from the initial policy for 100 iterations, each with 2048 steps.

Figure 2a shows the dynamics of the trace of the Hessian and the generalization errors during training. We find that they both follow a qualitatively similar trend. Next, as shown in Figure 2b, we find that single-task training yields a very low generalization error in the Meta-World benchmark,



(a) Illustration of Hessian traces and generalization errors



(b) Varying subset size α in Algorithm 2

Figure 2: Illustrating the Hessian trace measurements and empirical generalization errors with respect to the policy network. Figure 2a: Showing that the Hessian trace is comparable in scale to the observed generalization errors, tested on Meta-World and a control task. Figure 2b: Showing that in a Meta-World environment, the generalization error reaches the highest when the subset size $\alpha = 3$, suggesting negative transfer among a small set of tasks. In the meta-RL control task, the generalization performance monotonically improves with the addition of more tasks in each group.

indicating that a single policy can solve an individual task. However, when the task count reaches three, the generalization error sharply increases, suggesting that a single policy struggles to perform well across multiple tasks. As the number of tasks increases, the generalization error and the Hessian trace decrease, indicating the positive transfer among tasks. In the control environments, we observe that both the generalization error and the Hessian trace decrease as α increases.

4.5 Ablation Analysis

Given that each task is solvable in the Meta-World benchmark, our goal is to determine the minimum number of groups k required to maximize the success rate. We vary k from 1 to 4, observing average success rates of 94.0% when $k = 3$, compared to 89.5% when $k = 2$ and 95.1% when $k = 4$. Thus, we report the final results with three groups. For the control environments, we vary k from 1 to 5. We find that the success rate stabilizes after k reaches 3, so we report results using three groups.

For the random projection dimension d, we vary it from 200 to 1000 and observe that values beyond 400 yield minimal gains, so we fix $d = 400$.

5 Generalization Error Analysis via Hessians

In this section, we present a generalization error analysis for the RL setting described above. We present a Hessian-based technique to quantify generalization errors, applicable to any kind of policy network used in RL training. We first describe the tools that we will use. Given a data distribution of $\mathcal{D}$, let

$$L(f_W) = \underset{(x,y)\sim\mathcal{D}}{\mathbb{E}}\left[\ell(f_W(x), y)\right]$$

denote the expected loss of an input sample x, y. Let $\hat{L}(f_W)$ denote the empirical loss given n independent samples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ sampled from $\mathcal{D}$:

$$\hat{L}(f_W) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x_i), y_i).$$

Let $\mathcal{W} \subseteq \mathbb{R}^d$ denote the weight space of W. Assume that the weighted Hessian along x is bounded by a fixed constant $\mathcal{H}$ that does not grow with d or n:

$$\mathcal{H} := \sup_{W \in \mathcal{W}} \left(W^\top \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[[\nabla^2 \ell(f_W(x), y)]^+ \right] W \right),$$

where $[\nabla^2 \ell(f_W(x), y)]^+$ means that we truncate the negative eigen-directions of the Hessian to zero. Then, we have the following generalization error bound, which applies uniformly to the hypothesis space $\{f_W : W \in \mathcal{W}\}$.

Theorem 1. *Assume that the loss function ℓ is bounded between 0 and C for a fixed constant $C > 0$ on the data distribution $\mathcal{D}$. Suppose $\ell(f_W(\cdot), \cdot)$ is twice-differentiable in W and the Hessian matrix $\nabla^2 \ell(f_W(\cdot), \cdot)$ is Lipschitz-continuous within the weight space $\mathcal{W}$. Suppose there exists a fixed bound $\mathcal{H}$ on the Hessian trace over any $W \in \mathcal{W}$. Then, for any W in $\mathcal{W}$, any $\epsilon > 0$ small enough, and any small $\delta > 0$, with probability at least $1 - \delta$ over the randomness of n samples, we have:*

$$L(f_W) \le (1 + \epsilon)\hat{L}(f_W) + (1 + \epsilon)\sqrt{\frac{C \cdot \mathcal{H}}{n}} + \epsilon, \tag{6}$$

where $\epsilon = O(n^{-3/4} \log(\delta^{-1}))$ denotes the error term.

The proof of Theorem 1 is based on a PAC-Bayes analysis [15, 44], and the details are in the Appendix. In particular, a new aspect of this result is that we use an anisotropic perturbation in the prior and posterior distributions. We build on a linear PAC-Bayes bound (See Theorem 2 in the Appendix), and optimize the noise distribution to obtain the $\mathcal{H}$ measure.

In Figure 2, we visualize the Hessian trace on the Meta-World and the control tasks. Our results show that the Hessian trace is comparable in scale to the observed generalization errors. In the Meta-World benchmark, the generalization error increases as the number of tasks increases from one to three, suggesting negative transfer. As more tasks are added, the error decreases, indicating positive transfer.

6 Related Work

Multi-objective optimization for reinforcement learning. Early works on multitask learning use hard parameter sharing, where a single neural network backbone is shared among all tasks, with only the final layers or heads being task-specific [43]. A canonical example is the multi-head Soft Actor-Critic (SAC) architecture, which serves as a common baseline [43].

The multitask objective provides implicit regularization across multiple tasks trained together in a shared network [38], which can be formalized in a high-dimensional regression setting [39]. Training the model to learn representations useful across a diverse set of tasks helps prevent overfitting. A deeper analysis of the regularization effect behind multitask RL is an interesting question for future work.

Data attribution and model interpretability. Another line of work involves data modeling, which aims to trace a model's predictions back to its training data. One method is to retrain the model on a different subset of the dataset and estimate the Shapley value of each sample [9]. A commonly used technique is influence functions [16], which use Hessians to approximate the effect of removing a sample. Datamodels [11] finds that a linear regression method can accurately approximate the outcome of deep nets trained with a subset of samples on computer vision tasks. Li et al. [19] introduce a linear surrogate model for multitask learning and analyze the sample complexity of linear surrogate models. TRAK [26] further demonstrates that the approximate solution of linear regression computed with projected gradients delivers comparable results to the original linear model for ImageNet, CLIP, and BERT. Our work contributes to this literature by applying attribution methods to design RL algorithms.

7 Conclusion

This paper introduces an efficient algorithm for multi-objective reinforcement learning. The overall approach works by first training a meta-policy via multitask learning. Then, use a gradient-estimation algorithm to adapt this meta-policy to multiple subsets of training objectives. The key observation is that a policy can effectively adapt across different task subsets through a first-order approximation with a proper initialization. Leveraging this, we propose a surrogate model to approximate the actual training performance of the policy, which enabled us to derive task affinities and subsequently cluster similar tasks into groups. In extensive reinforcement learning tasks, our method consistently improves performance. Lastly, we measure sharpness via Hessians to analyze generalization in multi-objective RL. We hope this work inspires further studies on designing principled methods for multi-objective reinforcement learning and quantifying generalization in policy learning algorithms. Further applying our approach to broader RL settings is another promising direction for future work.

Acknowledgments

Thanks to the anonymous referees for their constructive feedback. This work is partially funded by NSF award IIS-2412008. Any views or opinions expressed herein are solely those of the authors listed, and may differ from those expressed by NSF.

References

[1] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward. "Relax, no need to round: Integrality of clustering formulations". In: *Innovations in Theoretical Computer Science*. 2015, pp. 191–200 (page 7).

[2] R. Caruana. "Multitask learning". In: *Machine Learning* 28.1 (1997), pp. 41–75 (page 2).

[3] Y. Chi, Y. Chen, and Y. Wei. "Statistical and Algorithmic Foundations of Reinforcement Learning". In: *arXiv preprint arXiv:2507.14444* (2025) (page 22).

[4] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. "Quantifying generalization in reinforcement learning". In: *International Conference on Machine Learning*. 2019, pp. 1282–1289 (page 2).

[5] L. Collins, A. Mokhtari, and S. Shakkottai. "Task-robust model-agnostic meta-learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 18860–18871 (page 24).

[6] F. Felten, L. N. Alegre, A. Nowé, A. L. C. Bazzan, E. G. Talbi, G. Danoy, and B. C. d. Silva. "A Toolkit for Reliable Benchmarking and Research in Multi-Objective Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2023 (page 8).

[7] C. Finn, P. Abbeel, and S. Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning*. 2017, pp. 1126–1135 (pages 1, 8, 23, 24).

[8] D. J. Foster and A. Rakhlin. "Foundations of reinforcement learning and interactive decision making". In: *arXiv preprint arXiv:2312.16730* (2023) (page 22).

[9] A. Ghorbani and J. Zou. "Data shapley: Equitable valuation of data for machine learning". In: *International Conference on Machine Learning*. 2019, pp. 2242–2251 (page 12).

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International Conference on Machine Learning*. 2018, pp. 1861–1870 (page 22).

[11] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. "Datamodels: Predicting predictions from training data". In: *International Conference on Machine Learning* (2022) (page 12).

[12] W. B. Johnson and J. Lindenstrauss. "Extensions of Lipschitz mappings into a Hilbert space". In: *Contemporary mathematics* 26.189-206 (1984), p. 1 (page 5).

[13] V. Joshi, Z. Xu, B. Liu, P. Stone, and A. Zhang. "Benchmarking massively parallelized multi-task reinforcement learning for robotics tasks". In: *Reinforcement Learning Journal* (2025) (page 1).

[14] H. Ju, D. Li, A. Sharma, and H. R. Zhang. "Generalization in graph neural networks: Improved pac-bayesian bounds on graph diffusion". In: *International conference on artificial intelligence and statistics*. 2023, pp. 6314–6341 (page 2).

[15] H. Ju, D. Li, and H. R. Zhang. "Robust fine-tuning of deep neural networks with hessian-based generalization guarantees". In: *International Conference on Machine Learning*. 2022, pp. 10431–10461 (pages 2, 11).

[16] P. W. Koh and P. Liang. "Understanding black-box predictions via influence functions". In: *International Conference on Machine Learning*. 2017, pp. 1885–1894 (page 12).

[17] V. Konda and J. Tsitsiklis. "Actor-critic algorithms". In: *Advances in Neural Information Processing Systems*. Vol. 12. 1999 (page 23).

[18] D. Li, H. Ju, A. Sharma, and H. R. Zhang. "Boosting multitask learning on graphs through higher-order task affinities". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 1213–1222 (pages 2, 3).

[19] D. Li, H. L. Nguyen, and H. R. Zhang. "Identification of negative transfers in multitask learning using surrogate models". In: *Transactions on Machine Learning Research* (2023) (pages 2, 12).

[20] D. Li, A. Sharma, and H. R. Zhang. "Scalable Multitask Learning Using Gradient-based Estimation of Task Affinity". In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 1542–1553 (page 2).

[21] D. Li, Z. Zhang, L. Wang, and H. Zhang. "Scalable Fine-tuning from Multiple Data Sources: A First-Order Approximation Approach". In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. 2024, pp. 5608–5623 (page 3).

[22] D. Li, Z. Zhang, L. Wang, and H. R. Zhang. "Efficient Ensemble for Fine-tuning Language Models on Multiple Datasets". In: *Association for Computational Linguistics* (2025) (page 22).

[23] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu. "Conflict-averse gradient descent for multi-task learning". In: *International Conference on Learning Representations*. 2021 (page 24).

[24] D. McAllester. "A PAC-Bayesian tutorial with a dropout bound". In: *arXiv preprint arXiv:1307.2118* (2013) (page 16).

[25] R. A. Meyer, C. Musco, C. Musco, and D. P. Woodruff. "Hutch++: Optimal stochastic trace estimation". In: *Symposium on Simplicity in Algorithms (SOSA)*. SIAM. 2021, pp. 142–155 (page 9).

[26] S. M. Park, Y. Tsvetkov, J. Z. Költer, and R. Salakhutdinov. "Trak: Attributing model behavior at scale". In: *International Conference on Machine Learning*. 2023, pp. 27457–27481 (page 12).

[27] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. "Efficient off-policy meta-reinforcement learning via probabilistic context variables". In: *International Conference on Machine Learning*. 2019, pp. 5331–5340 (page 1).

[28] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel. "ProMP: Proximal Meta-Policy Search". In: *International Conference on Learning Representations*. 2019 (page 24).

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017) (pages 4, 23).

[30] S. Sodhani, A. Zhang, and J. Pineau. "Multi-task reinforcement learning with context-based representations". In: *International Conference on Machine Learning*. 2021, pp. 9767–9779 (pages 1, 2, 8, 9).

[31] S. Sun, H. Shi, Z. Zhao, X. Wang, and J. Zhou. "PaCo: Parameter-compositional multi-task reinforcement learning". In: *International Conference on Learning Representations*. 2022 (pages 2, 8, 9, 24).

[32] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. "Distral: Robust multitask reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017 (page 24).

[33] S. Thrun and L. Pratt. "Learning to learn". In: Springer, 1998 (page 2).

[34] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis. "Gymnasium: A standard interface for reinforcement learning environments". In: *Advances in Neural Information Processing Systems Datasets and Benchmarks* (2025) (pages 2, 8, 22).

[35] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press, 2018 (page 21).

[36] M. J. Wainwright. *High-Dimensional Statistics: A Non-Symptotic Viewpoint*. Cambridge University Press, 2019 (page 21).

[37] M. J. Wainwright. "Variance-reduced Q-learning is minimax optimal". In: *arXiv preprint arXiv:1906.04697* (2019) (page 22).

[38] S. Wu, H. R. Zhang, and C. Ré. "Understanding and Improving Information Transfer in Multi-Task Learning". In: *International Conference on Learning Representations*. 2020 (pages 2, 3, 11).

[39] F. Yang, H. R. Zhang, S. Wu, C. Re, and W. J. Su. "Precise High-Dimensional Asymptotics for Quantifying Heterogeneous Transfers". In: *Journal of Machine Learning Research* 26.113 (2025), pp. 1–88 (page 11).

[40] R. Yang, H. Xu, Y. Wu, and X. Wang. "Multi-task reinforcement learning with soft modularization". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 4767–4777 (pages 2, 8, 9, 24).

[41] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn. "Meta-learning without memorization". In: *International Conference on Learning Representations* (2020) (page 2).

[42] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. "Gradient surgery for multi-task learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5822–5833 (pages 2, 9, 24).

[43] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning". In: *Conference on Robot Learning*. 2020, pp. 1094–1100 (pages 1, 2, 8, 9, 11).

[44] H. R. Zhang, D. Li, and H. Ju. "Noise Stability Optimization for Finding Flat Minima: A Hessian-based Regularization Approach". In: *Transactions on Machine Learning Research* (2024) (pages 2, 11).

[45] Z. Zhang, Z. Zhang, D. Li, L. Wang, J. Dy, and H. R. Zhang. "Linear-Time Demonstration Selection for In-Context Learning via Gradient Estimation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2025, pp. 16470–16488 (pages 2, 22).

A Clustering Algorithms

We provide further details on using optimization to cluster task groups based on the task affinity matrix. For a partition into m clusters, represented by binary indicator vectors $v_1, \dots, v_m$, the objective, which corresponds to the average density of the m clusters, is given by

$$\sum_{i=1}^{m} \frac{v_i^\top U v_i}{v_i^\top v_i}.$$

However, directly optimizing this combinatorial objective is NP-hard. We therefore introduce a matrix variable

$$X = \sum_{i=1}^{m} \frac{v_i v_i^\top}{v_i^\top v_i}.$$

The objective can then be rewritten as maximizing the inner product $\langle U, X\rangle$. In this formulation, the number of clusters m is equal to both the rank of X. Relaxing the non-convex rank constraint leads to a semi-definite program (SDP) formulation. To avoid specifying the number of clusters m and to automatically determine a suitable number of groups, we penalize the trace of X in the objective. This yields the following regularized optimization problem:

$$\begin{aligned} \max_{X \in \mathbb{R}^{n\times n}} \quad & \langle U, X\rangle - \lambda \cdot \mathrm{Tr}[X] \\ \text{s.t.} \quad & Xe = e, \mathrm{Tr}[X] = k \\ & X \succeq 0, X \geq 0, \end{aligned} \tag{7}$$

where e is the vector of all ones, and $\lambda > 0$ is a hyperparameter that controls the penalty on the number of clusters, effectively balancing clustering quality with the total number of groups.

Once we solve the SDP for the optimal X, we recover the discrete cluster assignments through a rounding procedure. This clustering procedure is efficient to implement in practice; solving the SDP for a typical number of tasks takes several seconds. Thus, the computational cost is negligible compared to the cost of model training.

B Proof of Theorem 1

We provide a high-level illustration of the proof of Theorem 1. There are two main steps in the proof. First, we would like to show that on the original input distribution, the gap between test and training losses remains bounded by the Hessian distance measure. Second, we show that such a statement holds for every transformed distribution of the input distribution. Thus, we reduce the proof of equation (6) to the first step.

Proof sketch. We start with the first step. Let

$$\hat{L}(f_W) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_W(x_i), y_i), \text{ and}$$
$$L(f_W) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\ell(f_W(x), y)\right],$$

denote the training (and test) losses of f_W, given n independent samples from data distribution $\mathcal{D}$. Let $\mathcal{Q}$ denote the *posterior* distribution. Specifically, we consider $\mathcal{Q}$ as being centered at the learned hypothesis W (which could be anywhere within the hypothesis space), given by a Gaussian distribution $\mathcal{N}(W, \Sigma)$, where Σ is a p by p covariance matrix. Given a sample $U \sim \mathcal{N}(0, \Sigma)$, let the perturbed loss be given by

$$\ell_{\mathcal{Q}}(f_W(x), y) = \mathop{\mathbb{E}}_{U} \left[\ell(f_{W+U}(x), y)\right]. \tag{8}$$

Then, let $\hat{L}_{\mathcal{Q}}(W)$ be the averaged value of $\ell_{\mathcal{Q}}(f_W(\cdot), \cdot)$, taken over n empirical samples from the training dataset. Likewise, let $L_{\mathcal{Q}}(W)$ be the population average of $\ell_{\mathcal{Q}}(f_W(\cdot), \cdot)$, in expectation over an unseen data sample from the underlying data distribution.

Having introduced the notations, we start with the PAC-Bayes bound [24], stated as follows.

Theorem 2. *Suppose the loss $\ell(f_W(x), y)$ lies in a bounded range $[0, C]$ given any $x \in \mathcal{X}$ with label y. Let $\mathcal{P}$ and $\mathcal{Q}$ be prior and posterior distributions on the weights W. For any $\beta \in (0,1)$ and $\delta \in (0,1)$, with probability at least $1-\delta$, we hhave:*

$$L_{\mathcal{Q}}(W) \le \frac{1}{\beta}\hat{L}_{\mathcal{Q}}(W) + \frac{C\big(KL(\mathcal{Q}||\mathcal{P}) + \log\frac{1}{\delta}\big)}{2\beta(1-\beta)n}. \tag{9}$$

This result provides flexibility in setting β. Our results will set β to balance the two terms. We will need the KL divergence between the prior $\mathcal{P}$ and the posterior $\mathcal{Q}$ in the PAC-Bayesian analysis. This is stated in the following classical result.

Proposition 3. *Suppose $\mathcal{P} = N(X, \Sigma)$ and $\mathcal{Q} = N(Y, \Sigma)$ are both Gaussian distributions with mean vectors given by $X \in \mathbb{R}^p, Y \in \mathbb{R}^p$, and population covariance matrix $\Sigma \in \mathbb{R}^{p\times p}$. The KL divergence between $\mathcal{P}$ and $\mathcal{Q}$ is equal to*

$$KL(\mathcal{Q}||\mathcal{P}) = \frac{1}{2}(X-Y)^\top \Sigma^{-1}(X-Y).$$

We are interested in the perturbed loss, $\ell_{\mathcal{Q}}(f_W(x), y)$, which is the expectation of $\ell(f_{W+U}(x), y)$ over U. Using a Taylor expansion, we find that

$$\ell_{\mathcal{Q}}(f_W(x), y) - \ell(f_W(x), y) \leq \left\langle \Sigma, \nabla^2 \ell(f_W(x), y) \right\rangle + \epsilon,$$

where Σ is the population covariance matrix of the perturbation, ∇^2 is the Hessian matrix with respect to the weights of f_W, and $\epsilon = C_1(\mathrm{Tr}\,[\Sigma])^{3/2}$. See Lemma 5 for the complete statement of this result.

Based on the above expansion, next, we apply the PAC-Bayes bound from equation (9) to an L-layer transformer neural network f_W parameterized by W. We note that the KL divergence between the prior and posterior distributions, which are both Gaussian, is equal to $\left\langle \Sigma^{-1}, vv^\top \right\rangle$, where v is the difference between the initialized and the trained weights.

Next, we combine the above Hessian-based bound and KL divergence in the PAC-Bayes bound. Let ∇_+^2 denote the truncated Hessian matrix where we set the negative eigenvalues of ∇^2 to zero. We have that

$$\langle \Sigma, \nabla_W^2[\ell(f_W(x), y)] \rangle \leq \langle \Sigma, \nabla_+^2[\ell(f_W(x), y)] \rangle. \tag{10}$$

Substituting into (9), and minimizing over β and Σ, we will derive an upper bound on the generalization error (between $L(f_W)$ and $\hat{L}(f_W)$) equal to

$$\alpha := \sup_{W \in \mathcal{W}} \sup_{(x,y)\sim\mathcal{D}} \frac{\sqrt{W^\top \left[\tilde{\nabla}_+^2 \ell(f_W(x), y)\right] W}}{\sqrt{n}}, \tag{11}$$

where $\mathcal{W}$ is the support of the distribution from which the data is drawn.

We will use a Taylor expansion to bound the perturbed loss, as follows.

Claim 4. *Let f_W be twice-differentiable, parameterized by weight vector $W \in \mathbb{R}^p$. Let $U \in \mathbb{R}^p$ be another vector with dimension p. For any W and U, the following identity holds*

$$\ell(f_{W+U}(x), y) = \ell(f_W(x), y) + U^\top \nabla \ell(f_W(x), y) + \frac{1}{2} U^\top [\nabla^2 \ell(f_W(x), y)] U + R_2(\ell(f_W(x), y)),$$

where $R_2(\ell(f_W(x), y)))$ is a second-order error term in a Taylor expansion of $\ell \circ f_W$ around W.

Proof. The proof follows from the fact that $\ell \circ f_W$ is twice-differentiable. By the mean value theorem, there must exist η between W and $U + W$ such that

$$R_2(\ell(f_W(x), y)) = U^\top \Big(\nabla^2[\ell(f_\eta(x), y)] - \nabla^2[\ell(f_W(x), y)] \Big) U.$$

This completes the proof of the claim.

Based on the above result, we provide a Taylor's expansion for $\ell_{\mathcal{Q}}$ minus ℓ.

Lemma 5. *In the setting of Theorem 1, suppose each parameter is perturbed by an independent random variable drawn from $N(0, \Sigma)$. Let $\ell_{\mathcal{Q}}(f_W(x), y)$ be the loss averaged over the noise. There is a value C_1 that does not depend on n and $1/\delta$ such that*

$$\ell_{\mathcal{Q}}(f_W(x), y) - \ell(f_W(x), y) \leq \frac{1}{2} \left\langle \Sigma, \nabla^2[\ell(f_W(x), y)] \right\rangle + C_1(\mathrm{Tr}\,[\Sigma])^{\frac{3}{2}}. \tag{12}$$

Proof. We take the expectations over U of both sides of the equation in Claim 4. The result becomes

$$\mathop{\mathbb{E}}_{U}\left[\ell(f_{W+U}(x),y)\right] = \mathop{\mathbb{E}}_{U}\left[\ell(f_W(x),y) + U^\top \nabla \ell(f_W(x),y) + \frac{1}{2}U^\top \nabla^2[\ell(f_W(x),y)]U + R_2(\ell(f_W(x),y))\right].$$

Then, we use the perturbation distribution to calculate

$$\ell_{\mathcal{Q}}(f_W(x),y) = \mathop{\mathbb{E}}_{U}\left[\ell(f_W(x),y)\right] + \mathop{\mathbb{E}}_{U}\left[U^\top \nabla \ell(f_W(x),y)\right] + \frac{1}{2}\mathop{\mathbb{E}}_{U}\left[U^\top \nabla^2[\ell(f_W(x),y)]U\right] + \mathop{\mathbb{E}}_{U}\left[R_2(\ell(f_W(x),y))\right].$$

Since $\mathbb{E}[U]=0$, the first-order term vanishes. The second-order term becomes

$$\mathop{\mathbb{E}}_{U}\left[U^\top[\nabla^2\ell(f_W(x),y)]U\right] = \langle \Sigma, \nabla^2[\ell(f_W(x),y)]\rangle. \tag{13}$$

The expectation of the error term $R_2(\ell(f_W(x),y))$ be

$$\begin{aligned}
\mathop{\mathbb{E}}_{U}\left[R_2(\ell(f_W(x),y))\right] &= \mathop{\mathbb{E}}_{U}\left[U^\top\left(\nabla^2[\ell(f_\eta(x),y)] - \nabla^2[\ell(f_W(x),y)]\right)U\right]\\
&\le \mathop{\mathbb{E}}_{U}\left[\|U\|_2^2 \cdot \left\|\nabla^2[\ell(f_\eta(x),y)] - \nabla^2[\ell(f_W(x),y)]\right\|_F\right]\\
&\lesssim \mathop{\mathbb{E}}_{U}\left[\|U\|_2^2 \cdot C_1\|U\|_2\right]\\
&\lesssim C_1\left(\mathbb{E}\left[U^\top U\right]\right)^{\frac{3}{2}} = C_1(\mathrm{Tr}\,[\Sigma])^{\frac{3}{2}}.
\end{aligned}$$

To obtain the first inequality in the last line, we have used that η lies on the line between W and $W+U$, so that $\|\eta\| \le \|U\|$, and that the Hessian is Lipschitz. Thus, the proof is complete.

The last piece we will need is the uniform convergence of the Hessian, which uses the fact that the Hessian is Lipschitz continuous.

Lemma 6. *In the setting of Theorem 1, there exist values C_2, C_3 that do not grow with n and $1/\delta$, such that for any $\delta > 0$, with probability at least $1-\delta$ over the randomness of the n training examples, we have*

$$\left\|\frac{1}{n}\sum_{i=1}^n \nabla^2[\ell(f_W(x_i),y_i)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2[\ell(f_W(x),y)]\right]\right\|_F \le \frac{C_2\sqrt{\log(C_3 n/\delta)}}{\sqrt{n}}. \tag{14}$$

The proof will be deferred to Section B.0.1. With these results ready, we will provide proof of the Hessian-based generalization bound.

Now, we present the full proof of Theorem 1.

Proof of Theorem 1. First, we separate the gap between $L(W)$ and $\frac{1}{\beta}\hat{L}(W)$ into three parts:

$$L(W) - \frac{1}{\beta}\hat{L}(W) = L(W) - L_{\mathcal{Q}}(W) + L_{\mathcal{Q}}(W) - \frac{1}{\beta}\hat{L}_{\mathcal{Q}}(W) + \frac{1}{\beta}\hat{L}_{\mathcal{Q}}(W) - \frac{1}{\beta}\hat{L}(W).$$

By Lemma 5, we can bound the difference between $L(W)$ and $L_{\mathcal{Q}}(W)$ by the Hessian trace plus an error:

$$\begin{aligned}
L(W) - \frac{1}{\beta}\hat{L}(W) \le &- \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{1}{2}\langle\Sigma, \nabla^2[\ell(f_W(x),y)]\rangle\right] + C_1(\mathrm{Tr}\,[\Sigma])^{\frac{3}{2}} + \Big(L_{\mathcal{Q}}(W) - \frac{1}{\beta}\hat{L}_{\mathcal{Q}}(W)\Big)\\
&+ \frac{1}{\beta}\Big(\frac{1}{n}\sum_{i=1}^n \frac{1}{2}\langle\Sigma, \mathrm{Tr}\left[\nabla^2[\ell(f_W(x_i),y_i)]\right]\rangle + C_1(\mathrm{Tr}\,[\Sigma])^{\frac{3}{2}}\Big).
\end{aligned}$$

After rearranging the terms, we find

$$L(W) - \frac{1}{\beta}\hat{L}(W) \leq \underbrace{- \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{1}{2}\langle\Sigma, \nabla^2[\ell(f_W(x), y)]\rangle\right] + \frac{1}{n\beta}\sum_{i=1}^{n}\frac{1}{2}\langle\Sigma, \nabla^2[\ell(f_W(x_i), y_i)]\rangle}_{E_1}$$
$$+ \frac{1+\beta}{\beta}C_1 \mathrm{Tr}\left[\Sigma\right]^{\frac{3}{2}} + \underbrace{L_{\mathcal{Q}}(W) - \frac{1}{\beta}\hat{L}_{\mathcal{Q}}(W)}_{E_2}. \tag{15}$$

We analyze E_1 by separating it into two parts

$$E_1 = \frac{1}{\beta}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\langle\Sigma, \nabla^2[\ell(f_{\hat{W}}(x_i), y_i)]\rangle - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\frac{1}{2}\langle\Sigma, \nabla^2[\ell(f_W(x), y)]\rangle\right]\right) \tag{16}$$
$$+ \frac{1-\beta}{2\beta}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\langle\Sigma, \nabla^2\ell(f_W(x), y)\rangle\right]. \tag{17}$$

We can use the uniform convergence result of Lemma 6 to bound the term in (16), leading to:

$$\frac{1}{2\beta}\left(\frac{1}{n}\sum_{i=1}^{n}\langle\Sigma, \nabla^2\ell(f_W(x_i), y_i)\rangle - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\langle\Sigma, \nabla^2\ell(f_W(x), y))\rangle\right]\right)$$
$$\leq \frac{\sigma^2}{2\beta}\cdot\sqrt{p}\cdot\left\|\frac{1}{n}\sum_{i=1}^{n}\nabla^2[\ell(f_W(x_i), y_i)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2[\ell(f_W(x), y)]\right]\right\|_F \qquad \text{(by Cauchy-Schwarz)}$$
$$\leq \frac{\sigma^2\sqrt{p}\cdot C_2\sqrt{\log(C_3 n/\delta)}}{2\beta\sqrt{n}}. \tag{18}$$

In particular, the second step also uses the fact that the Hessian is a symmetric p by p matrix. As for equation (17), we have that

$$\frac{1-\beta}{2\beta}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\langle\Sigma, \nabla^2\ell(f_W(x), y)\rangle\right] \leq \frac{1-\beta}{2\beta}\langle\Sigma, \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2_{+}\ell(f_W(x), y)\right]\rangle,$$

Combined with equation (18), we have shown that

$$E_1 \leq \frac{\sigma^2\sqrt{p}\cdot C_2\sqrt{\log(C_3 n/\delta)}}{2\beta\sqrt{n}} + \frac{1-\beta}{2\beta}\langle\Sigma, \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2_{+}\ell(f_W(x), y)\right]\rangle. \tag{19}$$

Next, for E_2, we use the PAC-Bayes bound of Theorem 2. In particular, we set the prior distribution $\mathcal{P}$ as the distribution of U and the posterior distribution $\mathcal{Q}$ as the distribution of $W + U$. Thus,

$$E_2 \leq \frac{C\big(KL(\mathcal{Q}||\mathcal{P}) + \log(\delta^{-1})\big)}{2\beta(1-\beta)n} \leq \frac{C\Big(\frac{1}{2}W^\top\Sigma^{-1}W + \log(\delta^{-1})\Big)}{2\beta(1-\beta)n}. \tag{20}$$

Combining equations (15), (19), (20), we claim that with probability at least $1 - 2\delta$, we must have:

$$L(W) - \frac{1}{\beta}\hat{L}(W) \leq \frac{\sigma^2\sqrt{p}\cdot C_2\sqrt{\log(C_3 n/\delta)}}{2\beta\sqrt{n}} + \frac{1+\beta}{\beta}C_1\mathrm{Tr}\left[\Sigma\right]^{3/2} + \frac{C\log\frac{1}{\delta}}{2\beta(1-\beta)n} \tag{21}$$
$$+ \frac{CW^\top\Sigma^{-1}W}{4\beta(1-\beta)n} + \frac{1-\beta}{2\beta}\langle\Sigma, \mathbf{H}_W\rangle, \tag{22}$$

where

$$\mathbf{H}_W = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[[\nabla^2 \ell(f_W(x), y)]^+\right].$$

We next choose Σ and $\beta \in (0,1)$ to minimize the above bound. In particular, we set

$$\Sigma = \sqrt{\frac{C}{2(1-\beta)^2 n \left\|W\right\|^2}} \mathbf{H}_W{}^{-\frac{1}{2}} W W^\top \tag{23}$$

so that the two terms in equation (22) are equal to each other:

$$\begin{aligned}\frac{1-\beta}{\beta}\cdot\sqrt{\frac{C}{2(1-\beta)^2 n \left\|W\right\|^2}}\langle \mathbf{H}_W{}^{\frac{1}{2}} W, W\rangle &\le \frac{1}{\beta}\sqrt{\frac{C}{2n\left\|W\right\|^2}}\left\|\mathbf{H}_W{}^{\frac{1}{2}} W\right\|\left\|W\right\| \\ &= \frac{1}{\beta}\sqrt{\frac{C}{2n}}\left\|W^\top \mathbf{H}_W W\right\| \\ &\le \frac{1}{\beta}\sqrt{\frac{C\mathcal{H}}{2n}}.\end{aligned}$$

By plugging in σ to equation (21) and re-arranging terms, the gap between $L(W)$ and $\beta^{-1}\hat{L}(W)$ can be bounded as:

$$L(W) - \frac{1}{\beta}\hat{L}(W) \le \frac{1}{\beta}\sqrt{\frac{C\mathcal{H}}{n}} + \frac{C_2\sqrt{2p\log(C_3 n/\delta)}}{2\beta\sqrt{n}}\sigma^2 + \frac{1+\beta}{\beta}C_1 \operatorname{Tr}\left[\Sigma\right]^{3/2} + \frac{C}{2\beta(1-\beta)n}\log\frac{1}{\delta}.$$

Let $\beta = 1/(1+\epsilon)$ and so that $\epsilon = (1-\beta)/\beta$. We find that

$$\begin{aligned}&L(W) \le (1+\epsilon)\hat{L}(W) + (1+\epsilon)\sqrt{\frac{C\mathcal{H}}{n}} + \xi, \text{ where}\\ &\xi = \frac{C_2\sqrt{2p\log(C_3 n/\delta)}}{2\beta\sqrt{n}}\sigma^2 + \left(1+\frac{1}{\beta}\right)C_1 \operatorname{Tr}\left[\Sigma\right]^{3/2} + \frac{C}{2\beta(1-\beta)n}\log\frac{1}{\delta}.\end{aligned}$$

Notice that ξ is of order

$$O(n^{-\frac{3}{4}} + n^{-\frac{3}{4}} + \log(\delta^{-1})n^{-1}) = O(\log(\delta^{-1})n^{-\frac{3}{4}}).$$

Therefore, we have shown that with probability at least $1-\delta$,

$$L(f_W) \le (1+\epsilon)\hat{L}(f_W) + (1+\epsilon)\sqrt{\frac{C\mathcal{H}}{n}} + O(n^{-3/4}). \tag{24}$$

The proof is now complete.

B.0.1 Proof of Lemma 6

In this section, we provide the proof of Lemma 6, which shows the uniform convergence of the loss Hessian.

Proof of Lemma 6. Let C, $\epsilon > 0$, and let

$$S = \{W \in \mathbb{R}^p : \left\|W\right\|_2 \le C\}.$$

There exists an ϵ-cover of S with respect to the ℓ_2-norm with at most $\max\left(\left(\frac{3C}{\epsilon}\right)^p, 1\right)$ elements; see, e.g., Example 5.8 [36]. Let $T \subseteq S$ denote this cover. Recall that the Hessian $\nabla^2[\ell(f_W(x), y)]$ is C_1-Lipschitz for all $(W+U) \in S, W \in S$. Then we have

$$\left\|\nabla^2[\ell(f_{W+U}(x), y)] - \nabla^2[\ell(f_W(x), y)]\right\|_F \leq C_1 \left\|U\right\|_2.$$

For $\delta, \epsilon > 0$, define the event

$$E = \left\{\forall\ W \in T, \left\|\frac{1}{n}\sum_{i=1}^n \nabla^2[\ell(f_W(x_i), y_i)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2[\ell(f_W(x), y)]\right]\right\|_F \leq \delta\right\}.$$

By the matrix Bernstein inequality [35], we have

$$\Pr[E] \geq 1 - 4 \cdot |\mathcal{N}| \cdot p \cdot \exp\left(-\frac{n\delta^2}{2\alpha^2}\right). \tag{25}$$

Next, for any $W \in S$, we can pick some $W + U \in T$ such that $\left\|U\right\|_2 \leq \epsilon$. We have

$$\left\|\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2[\ell(f_{W+U}(x), y)]\right] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2[\ell(f_W(x), y)]\right]\right\|_F \leq C_1 \left\|U\right\|_2 \leq C_1\epsilon$$

$$\left\|\frac{1}{n}\sum_{j=1}^n \nabla^2[\ell(f_{W+U}(x_j), y_j)] - \frac{1}{n}\sum_{j=1}^n \nabla^2[\ell(f_W(x_j), y_j)]\right\|_F \leq C_1 \left\|U\right\|_2 \leq C_1\epsilon.$$

Therefore, for any $W \in S$, we obtain:

$$\left\|\frac{1}{n}\sum_{j=1}^n \nabla^2[\ell(f_W(x_j), y_j)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2[\ell(f_W(x), y)]\right]\right\|_F \leq 2C_1\epsilon + \delta.$$

Set $\epsilon = \delta/(2C_1)$ so that on the event E,

$$\left\|\frac{1}{n}\sum_{j=1}^n \nabla^2[\ell(f_W(x_j), y_j)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\nabla^2[\ell(f_W(x), y)]\right]\right\|_F \leq 2\delta.$$

The event E happens with a probability of at least:

$$1 - 4|T|p \cdot \exp\left(-\frac{n\delta^2}{2\alpha^2}\right) = 1 - 4p \cdot \exp\left(\log|T| - \frac{n\delta^2}{2\alpha^2}\right).$$

Now, we have $\log|T| \leq p\log(3B/\epsilon) = p\log(6CC_1/\delta)$. If we set

$$\delta = \sqrt{\frac{4p\alpha^2\log(3\tau CC_1 n/\alpha)}{n}}, \tag{26}$$

so that $\log(3\tau CC_1 n/\alpha) \geq 1$ (because $n \geq \frac{e\alpha}{3C_1}$ and $\tau \geq 1$), then we find

$$\begin{aligned}
p\log(6CC_1/\delta) - n\delta^2/(2\alpha^2) =& p\log\left(\frac{6CC_1\sqrt{n}}{\sqrt{4p\alpha^2\log(3\tau CC_1 n/\alpha)}}\right) - 2p\log\left(3\tau CC_1 n/\alpha\right) \\
=& p\log\left(\frac{3CC_1\sqrt{n}}{\alpha\sqrt{p\log(3\tau CC_1 n/\alpha)}}\right) - 2p\log\left(3\tau CC_1 n/\alpha\right) \\
\leq& p\log\left(3\tau CC_1 n/\alpha\right) - 2p\log\left(3\tau CC_1 n/\alpha\right) \quad (\tau \geq 1, \log(3\tau CC_1 n/\alpha) \geq 1) \\
=& -p\log\left(3\tau CC_1 n/\alpha\right) \leq -p\log(e\tau). \qquad (3CC_1 n/\alpha \geq e)
\end{aligned}$$

Therefore, with a probability at least

$$1 - 4|\mathcal{N}|p \cdot \exp(-n\delta^2/(2\alpha^2)) \geq 1 - 4p(e\tau)^{-p}, \tag{27}$$

we have

$$\left\| \frac{1}{n} \sum_{j=1}^{n} \nabla^2[\ell(f_W(x_j), y_j)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\nabla^2[\ell(f_W(x), y)]\right] \right\|_F \leq \sqrt{\frac{16p\alpha^2 \log(3\tau C C_1 n/\alpha)}{n}}.$$

Denote $\delta' = 4p(e\tau)^{-p}$, $C_2 = 4\alpha\sqrt{p}$, and $C_3 = 12pCC_1/(e\alpha)$. With probability at least $1 - \delta'$, we have

$$\left\| \frac{1}{n} \sum_{i=1}^{n} \nabla^2[\ell(f_W(x_i), y_i)] - \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\nabla^2[\ell(f_W(x), y)]\right] \right\|_F \leq C_2 \sqrt{\frac{\log(C_3 n/\delta')}{n}}.$$

This completes the proof of Lemma 6.

Discussions. The minimax optimal sample complexity to find an ϵ-optimal Q-function in tabular γ-discounted MDPs is known to be $\tilde{O}\left(SA(1-\gamma)^{-3}\epsilon^{-2}\right)$ [3]. This rate is achieved by model-based algorithms. Standard synchronous Q-learning, however, is sub-optimal, requiring $\tilde{O}\left(SA(1-\gamma)^{-4}\epsilon^{-2}\right)$ samples. This gap in horizon dependency was later closed by variance reduction techniques. For instance, a variance-reduced Q-learning algorithm [37] achieves the minimax optimal rate, matching the lower bound up to logarithmic factors. See a recent monograph for extensive references [8]. It is an interesting open question to study the sample complexity of Q-learning for multi-objective reinforcement learning. Another interesting question is to explore gradient estimation [22, 45] for designing ensemble methods in meta-RL. These are left for future work.

C Omitted Experimentals

Environments. We summarize the RL environments: MT10, CartPole, Highway, LunarLander in Table 4. We adopt the Gymnasium library [34] for environment implementation.

RL algorithms. In the Meta-World experiments, we employ Soft Actor-Critic (SAC) [10] as the policy gradient algorithm. SAC is an off-policy, actor-critic deep reinforcement learning algorithm based on the maximum entropy reinforcement learning framework. The core idea is to encourage exploration by maximizing a trade-off between the expected return and the policy's entropy.

The SAC algorithm learns three functions: A policy π_θ and two critic functions Q_{ψ_1} and Q_{ψ_2}. We train the critic networks by minimizing the mean squared Bellman error:

$$L(\psi_i) = \hat{\mathbb{E}}_{(s_t, a_t, r_t, s_{t+1}, d_t)\sim\mathcal{D}} \left[\frac{1}{2}\left(Q_{\psi_i}(s_t, a_t) - y(r_t, s_{t+1}, d_t)\right)^2\right]. \tag{28}$$

The target value y is shared between both critics and is computed as:

$$y(r_t, s_{t+1}, d_t) = r_t + \gamma(1 - d_t)\left(\min_{j=1,2} Q_{\psi_{j,\text{target}}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\theta(a_{t+1} \mid s_{t+1})\right), \tag{29}$$

with $a_{t+1} \sim \pi_\theta(\cdot \mid s_{t+1})$. The target network parameters $\psi_{j,\text{target}}$ is updated by

$$\psi_{j,\text{target}} \leftarrow \tau\psi_j + (1-\tau)\psi_{j,\text{target}}.$$

Table 4: A summary of RL environments tested in our experiments.

Env.	**Goal**	**State**	**Action**	**Reward**
MT10	Execute robotic manipulation tasks	$\mathbb{R}^{39}$, representing the state of the robot and the target objective.	4 continuous actions: controlling end-effector movement and gripper actuation.	A sparse reward for moving the objective to its goal position.
CartPole	Balance a pole attached to a cart	$\mathbb{R}^{4}$, representing the cart's position, velocity, the pole's angle, and angular velocity.	2 discrete actions: applying a push to the cart (left or right).	A reward of +1 for each timestep the pole remains upright.
Highway	Avoid collisions while driving on a highway	$\mathbb{R}^{5\times 5}$, representing the kinematic states of the ego vehicle and four nearby vehicles.	5 discrete actions: change lane left/right, maintain lane, accelerate, decelerate.	A dense reward for maintaining high speed, with a large penalty for collisions.
LunarLander	Land a spacecraft safely on the ground	$\mathbb{R}^{8}$, representing the lander's coordinates, velocities, angle, and leg contact status.	4 discrete actions: control the main engine and side thrusters.	A shaped reward for approaching the landing pad, plus a large bonus for a soft landing.

Next, the actor parameters θ are trained to maximize the policy's expected return and entropy:

$$L_{\text{SAC}}(\theta) = \mathbb{E}_{\hat{s}_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)} \left[\alpha \log \pi_\theta(a_t \mid s_t) - \min_{j=1,2} Q_{\psi_j}(s_t, a_t) \right] \right] \tag{30}$$

In the control environments, we employ Proximal Policy Optimization (PPO) [29] as our policy gradient algorithm. PPO follows the design of the actor-critic algorithm [17], where the core optimization objective is the clipped surrogate policy loss. This is combined with a value function loss and an entropy as the optimization objective.

PPO involves tackling the following objective:

$$L_{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] - c_1 \mathbb{E}_t \left[(V_\theta(s_t) - \hat{V}_t)^2 \right] + c_2 \mathbb{E}_t \left[\mathcal{H}[\pi_\theta](s_t) \right], \tag{31}$$

where

$$r_t(\theta_t) = \frac{\pi_{\theta_t}(a_t|s_t)}{\pi_{\theta_{t-1}}(a_t|s_t)}$$

is the probability ratio between the new policy and the previous policy, $\hat{A}_t$ is the advantage estimation computed from the value function $V_\theta(s_t)$, and $\mathcal{H}[\pi_\theta](s_t)$ is the entropy of the policy.

In the meta-RL setting, we use the MAML algorithm [7], which consists of a two-step optimization process. In the inner loop adaptation, MAML computes an adapted policy for each source task T_i from the trajectory $\mathcal{D}^i_\theta$ sampled by the interaction with the meta policy θ and the corresponding environment:

$$\theta_i' = \theta - \alpha \nabla_\theta L_{T_i}(\pi_\theta). \tag{32}$$

In the outer loop meta-update, MAML then aggregates the data from the trajectory sampled by the interaction with the task-specific policy ϕ_1^i and the corresponding environment:

$$\theta \leftarrow \theta + \beta \cdot \nabla_\theta \sum_{T_i \in \mathcal{T}} L_{T_i}(\pi_\theta). \tag{33}$$

Baseline descriptions. PaCo (Parameter Compositional MTRL) introduces a highly flexible framework where task-specific parameters are explicitly composed rather than separated [31]. In PaCo, the parameter vector for a given task is constructed as a linear combination of a shared, task-agnostic basis of parameter vectors and a learned, task-specific compositional vector. Other notable architectures, such as Soft Modularization (SM) and those based on Mixture-of-Experts (MoE), explore ways to dynamically route information and learn task-specific components in a unified network [40, 31].

Simultaneous optimization of a shared set of parameters under competing demands of various tasks frequently leads to destructive interference, where joint training results in worse performance on a given task than training it alone [32, 42]. The gradients from different tasks can point in conflicting directions, which destabilizes learning [42, 23].

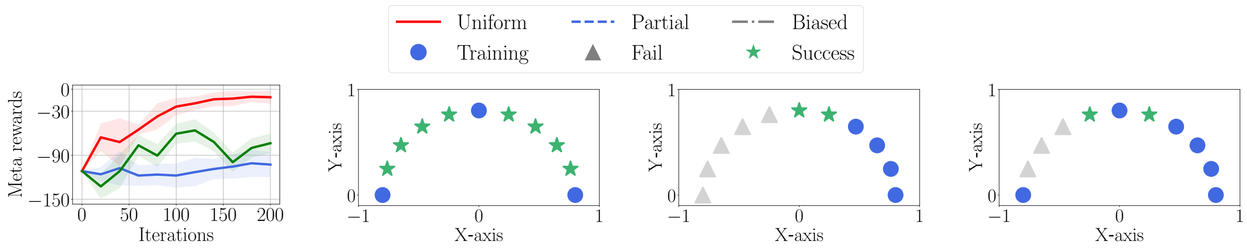
Meta-reinforcement learning. Meta reinforcement learning may be viewed as a specific scenario of meta learning. MAML [7] is a foundational work that introduces a bi-level optimization structure. In the inner loop, the policy is adapted to specific source tasks. In the outer loop, the initial meta-policies are updated by differentiating with respect to inner-loop adaptation, optimizing for the expected performance of the adapted policy across a distribution of source tasks. Building on this, ProMP [28] introduces a low variance curvature estimator to stabilize training by assigning weight to each sample in trajectories. Tr-MAML [5] focuses on the most challenging task and introduces a min-max framework to minimize the maximum loss in the set.

Implementation details. In the Meta-World benchmark, we use training steps of 10^7, a buffer size of 10^6, a batch size of 1280, and a learning rate of 3×10^{-4} with the Adam optimizer.

In the CartPole, Highway, and LunarLander environments, we use the following settings: 100 meta-training steps, 2048 training steps for each inner loop policy update, and a batch size of 64.

Additional experiments. We find that the shifted distribution of the source task $\mathcal{T}$ affects the generalization on the unseen task. We conduct a straightforward experiment in a 2D navigation environment to empirically investigate the influence of the source task distribution on meta reinforcement learning. In this environment, the agent starts at the origin $(0, 0)$ and navigates towards a goal within a $[-1, 1] \times [-1, 1]$ plane. All goals are located on a circle centered at the origin with a fixed radius of 0.9. An episode is considered a success if the agent's final distance to the goal is within a threshold of 0.1. The agent is guided by a reward function defined as the negative Euclidean distance to the goal, with an additional bonus upon success.

For the experiment, we maintain a uniform distribution for the target tasks, where goals can appear anywhere on the circle. We then analyze the agent's adaptation performance after being meta-trained on source tasks drawn from three different distributions. The training curves are illustrated in Figure 3a. We find that meta-training under uniform task distribution consistently outperforms that under partial and biased task distribution. In detail, we find that:



(a) Meta training curve (b) Uniform sampling (c) Partial sampling (d) Biased sampling

Figure 3: We illustrate the meta-training curve and the corresponding meta task distribution of the navigation environment. The agent is meta-trained to navigate a training set of goals, and then tested on a distinct set of unseen test goals with a few adaptation steps.

1. In Figure 3b, when the source task distribution is uniform and covers the four directions of the target space, the agent can easily adapt to any target task.
2. In Figure 3c, when the source tasks do not cover the target tasks, the policy overfits to the source distribution and fails to generalize.
3. In Figure 3d, when the source distribution covers the target space but is heavily biased, the policy struggles to adapt to target tasks in the sparsely represented regions.