Mastering Massive Multi-Task Reinforcement Learning via Mixture-of-Expert Decision Transformer

Yilun Kong [1] **Guozheng Ma** [2] **Qi Zhao** [1] **Haoyu Wang** [1] **Li Shen** [3] **Xueqian Wang** [1] **Dacheng Tao** [2]

Abstract

Despite recent advancements in offline multi-task reinforcement learning (MTRL) have harnessed the powerful capabilities of the Transformer architecture, most approaches focus on a limited number of tasks, with scaling to extremely massive tasks remaining a formidable challenge. In this paper, we first revisit the key impact of task numbers on current MTRL method, and further reveal that naively expanding the parameters proves insufficient to counteract the performance degradation as the number of tasks escalates. Building upon these insights, we propose M3DT, a novel mixture-of-experts (MoE) framework that tackles task scalability by further unlocking the model's parameter scalability. Specifically, we enhance both the architecture and the optimization of the agent, where we strengthen the Decision Transformer (DT) backbone with MoE to reduce task load on parameter subsets, and introduce a three-stage training mechanism to facilitate efficient training with optimal performance. Experimental results show that, by increasing the number of experts, M3DT not only consistently enhances its performance as model expansion on the fixed task numbers, but also exhibits remarkable task scalability, successfully extending to 160 tasks with superior performance.

1. Introduction

Recent developments, such as Decision Transformer (Chen et al., 2021) and Trajectory Transformer (Janner et al., 2021), have reframed offline reinforcement learning (RL) as a sequence modeling problem, showcasing their ability to transform large-scale datasets into potent decision-making agents.

[1]Tsinghua University, China [2]Nanyang Technological University, Singapore [3]Shenzhen Campus of Sun Yat-sen University. Correspondence to: Xueqian Wang <wang.xq@sz.tsinghua.edu.cn>, Li Shen <mathshenli@gmail.com>.

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

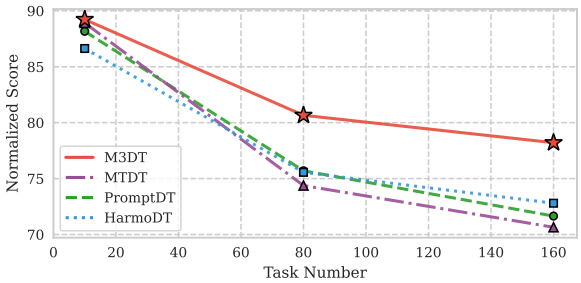

Figure 1: Illustration of a comparative analysis of normalized score across various task numbers. **M3DT achieves unparalleled task scalability.** An in-depth exploration of these results refers to Section 5.

These models also prove valuable for multi-task RL (MTRL) challenges, offering a high-capacity framework capable of accommodating task variances and assimilating knowledge from diverse datasets. Additionally, they pave the way for incorporating innovations from language modeling (Brown et al., 2020) into MTRL methodologies, unlocking new potential for cross-disciplinary advancements.

Drawing inspiration from large language models (LLM), where models harness a remarkable generalization capability to address massive tasks, there's a growing interest in the potential of training RL agent to master increasingly diverse tasks. However, the application of these high-capacity sequential models to massive multi-task RL presents considerable challenges. Firstly, existing approaches exhibit limited scalability with respect to task numbers. With most studies confined to dozens of tasks in Atari or Meta-World (Lee et al., 2022; Hu et al., 2024), when scaled to a larger number of tasks, their performance degrades significantly; while Gato (Reed et al., 2022) extends the research to over 600 tasks, its performance on control tasks remains suboptimal. Secondly, current research either overlooks the impact of parameter scaling, with most studies confined to very small models (Xu et al., 2022), or overly relies on the inherent parameter scalability of the Transformer architecture. Although Gato (Reed et al., 2022) and Multi-Game DT (Lee et al., 2022) have experimentally demonstrated the performance enhancements with expanded model size, a comparison between the increase in model size and the corresponding performance gains suggests that these approaches cannot be considered an efficient method of parameter scaling.

To address the above challenges, we extend the research scenario to encompass 160 simulation control tasks. We begin by investigating the pivotal role of task quantity on model performance and gradient conflicts, followed by an exploration of the impact of model expansion on MTRL. With the evident phenomenon that model performance and gradient conflicts deteriorate as the task scales, our findings indicate that these declines are most pronounced when the task number remains relatively low; once the number of tasks reaches sufficiently large, the performance degradation tends to become gradual and steady. Therefore, one of our key insights emerged when viewed from a reverse perspective: *reducing the learning task numbers to a sufficiently small scale can significantly enhance the performance.* In our study on model scaling, the results surprisingly reveal that simply increasing the model size rapidly hits the performance ceiling. Thus, naively expanding shared parameters to offset the performance degradation from an increasing task numbers proves ineffective. In contrast, expanding parameters while reducing the task numbers results in the most pronounced performance gains. So *how can we really minimize the number of tasks to be learned while efficiently scaling model parameters, thereby maximizing performance?*

Building upon these insights, we propose M3DT, a **M**oE-based **DT** for handling **M**assive **M**ulti-task RL. The overall framework is shown in Figure 4. We make improvements in both the architecture and optimization of the RL agent. Specifically, we introduce mixture-of-experts (MoE) in the DT backbone to achieve parameter separation, allowing the backbone to learn shared knowledge across all tasks, while each expert specializes in learning task-specific knowledge from a distinct small task subset, which greatly simplifies the training of parameters in experts. This also unlocks effective parameter expansion, as the number of experts is the most efficient way to scale models (Fedus et al., 2022). By increasing the number of experts, we can not only introducing a large number of parameters, but reduce the task load on each parameter subset, which mutually reinforce the model performance. Furthermore, M3DT introduces a three-stage training mechanism, sequentially optimize the DT backbone, each expert, and the router, respectively, which allows each module to explicitly learn specialized knowledge without interference, mitigating the severe gradient conflicts encountered when scaling to massive tasks, meanwhile reducing the difficulty of MoE training. Experimental results show that, by increasing the number of experts, on one hand, M3DT can consistently enhance its performance as model expansion on the fixed task scale; on the other hand, it exhibits remarkable task scalability, successfully extending to 160 tasks with superior performance.

In summary, our research makes three significant contributions to the field of MTRL:

1. We rethink the challenges of sequence modeling in MTRL from the perspective of task numbers and model size, analyze the performance degradation and gradient conflicts with increasing task numbers, and identify the limited effectiveness of naively parameter scaling in handling massive multi-task scenarios. (Section 3)
2. Based on the above insights, we propose M3DT, a novel framework that enhances the DT architecture with MoE, explicitly assign the grouped task subsets to each expert through task grouping, and introduce a three-stage training mechanism for training each module without interference. By increasing the number of experts, we unlock the parameter scalability for mastering massive tasks. (Section 4)
3. We demonstrate the superior performance of M3DT through rigorous testing on a broad spectrum of task scales, analyze its functionality through extensive ablation studies, and verify its task scalability and parameter scalability. (Section 5)

2. Preliminary

Offline Reinforcement Learning. The goal of RL is to learn a policy $\pi_\theta(a|s)$ maximizing the expected return $\mathbb{E}[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(s_t,a_t)]$ in a Markov Decision Process (MDP) $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma,d_0)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, environment dynamics $\mathcal{P}(s'|s,a):\mathcal{S}\times\mathcal{S}\times\mathcal{A}\to[0,1]$, reward function $\mathcal{R}:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$, discount factor $\gamma\in[0,1)$, and initial state distribution d_0 (Sutton & Barto, 2018). In the offline setting (Levine et al., 2020), a static dataset $\mathcal{D}=\{(s,a,s',r)\}$, collected by a behavior policy π_β, is provided. Offline RL algorithms learn a policy entirely from this static offline dataset, without any online interactions with the environment.

Multi-Task RL. In multi-task RL, different tasks can have different reward functions, state spaces, and transition functions. Given a specific task $\mathcal{T}\sim p(\mathcal{T})$, a task-specified MDP can be defined as $(\mathcal{S}^\mathcal{T},\mathcal{A}^\mathcal{T},\mathcal{P}^\mathcal{T},\mathcal{R}^\mathcal{T},\gamma,d_0^\mathcal{T})$. Instead of solving a single MDP, the goal is to find an optimal policy that maximizes expected return over all tasks: $\pi^*=\arg\max_\pi\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\mathbb{E}_{a_t\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r_t^\mathcal{T}]$. The static dataset $\mathcal{D}$ is correspondingly partitioned into per-task subsets as $\mathcal{D}=\cup_{i=1}^N\mathcal{D}_i$, where N is the number of tasks.

Prompt Decision Transformer. The integration of Transformer(Vaswani, 2017) in offline RL for sequence modeling has gained prominence in recent years. Prompt-DT (Xu et al., 2022) extends DT by using task-specific prompts to enhance multi-task learning and few-shot generalization. Unlike text-based prompt in NLP (Liu et al., 2023; Kong et al., 2025), Prompt-DT employs short trajectories as prompts, which consist of state, action, and return-to-go tuples $(s^*,a^*,\hat{r}^*)$, providing directed guidance to RL agents with few-shot demonstrations. Each element marked with the superscript $\cdot*$ is relevant to the trajectory prompt. These

trajectory prompts are much shorter than the task's horizon, encompassing essential information to only identify task, yet inadequate for task imitation. During training with offline data, Prompt-DT utilizes $\tau_{i,t}^{input} = (\tau_i^*, \tau_{i,t})$ as input for each task $\mathcal{T}_i$, combining a K^*-step trajectory prompt τ_i^* with a normal K-step trajectory $\tau_{i,t}$. The trajectory is formulated as:

$$\begin{aligned}\tau_{i,t}^{input} = (&\hat{r}_{i,1}^*, s_{i,1}^*, a_{i,1}^*, ..., \hat{r}_{i,K^*}^*, s_{i,K^*}^*, a_{i,K^*}^*, \\ &\hat{r}_{i,t-K+1}, s_{i,t-K+1}, a_{i,t-K+1}, ..., \hat{r}_{i,t}, s_{i,t}, a_{i,t}).\end{aligned} \tag{1}$$

The action a is predicted through a prediction head linked to the state token. The training objective aims to minimize the mean-squared loss:

$$\mathcal{L}_{DT} = \mathbb{E}_{\tau_{i,t}^{input} \sim \mathcal{D}_i} \left[\frac{1}{K} \sum_{m=t-K+1}^{t} (a_{i,m} - \pi(\tau_i^*, \tau_{i,m}))^2 \right]. \tag{2}$$

Prompt-DT has become a widely used backbone in recent research on MTRL.

3. Rethinking DT with MTRL

In this section, we delineate two primary challenges of employing DT in MTRL: the inability to scale with task number and model size, laying the groundwork for the motivation behind our method.

3.1. Limited Scalability of Task Numbers

The complexity of MTRL is significantly amplified with increasing task numbers, largely attributed to escalating gradient conflicts. These conflicts stem from optimizing a shared set of parameters across tasks with differing objectives, leading to compromises in task-specific optimization. To better understand the impact of task numbers, we use Prompt-DT and construct an offline dataset containing 160 tasks based on Meta-World (Yu et al., 2020b), DM Control (Tassa et al., 2018), and Mujoco Locomotion (Todorov et al., 2012), investigating the performance trends as the task number scales from 10 to 160. We measure the model's performance across three benchmarks using normalized scores. The gradient conflict is assessed by the average cosine similarity between the aggregate gradient and the gradients of each task, where a lower similarity indicates a higher degree of gradient conflict. The implementation details are described in Appendix A.

During the expansion of the training tasks, we observe a divergent trend in the performance across different task scales. The results presented in Figure 2 highlight three distinct phenomena: • **Normalized Score:** In the clear trend where the performance degrades with increasing task numbers, the decline is pronounced when task number is relatively low (below 40 tasks), while it becomes much more gradual once the tasks reach a sufficiently large number (above 80 tasks). • **Gradient Conflicts:** The decrease of gradient similarity

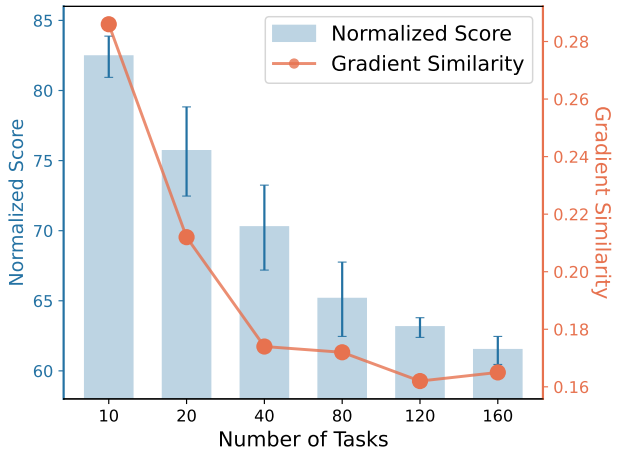

Figure 2: With the number of tasks increases from 10 to 160, both model performance and gradient similarity experience a noticeable decline.

generally aligns with that of model performance, while it drops more rapidly when the task number is low, and, counterintuitively, levels off and shows minimal decline after exceeding 40 tasks. • **Performance Variance:** For each run, we test on different task sets with the same number of tasks, thus the standard deviation of normalized scores reflects the model's robustness to variations in task combinations. For a massive number of tasks (120 to 160), the standard deviation is small, as the variations in task subsets from the total 160 tasks are minimal. For a moderate number of tasks (20 to 80), the standard deviation is large, indicating that different task combinations significantly impact performance, as similar tasks are easier for learning. For few tasks (10), despite greater variability in sampled task subsets, the standard deviation is much smaller than that in moderate tasks, which underscores that when task number is sufficiently low, the learning process is simple enough to be affected by the inter-task relationships.

Thus, a reverse perspective reveals the key insight: *reducing the learning task number, particularly to a sufficiently small scale, can significantly enhance model performance.* When the task number is sufficiently small, challenges such as gradient conflicts and inter-task relationships become effortlessly manageable. However, given the predefined number of tasks must be learned, how to reduce the actual learning task burden?

3.2. Limited Scalability of Model Size in MTRL

Since existing models struggle with large-scale tasks in MTRL, a straightforward approach is to scale up the model size to enhance its capacity. The supervised learning community has convincingly demonstrated that larger networks lead to improved performance, in particular for language models (Kaplan et al., 2020). To investigate whether this trend holds for the DT model in MTRL, we conduct exper-

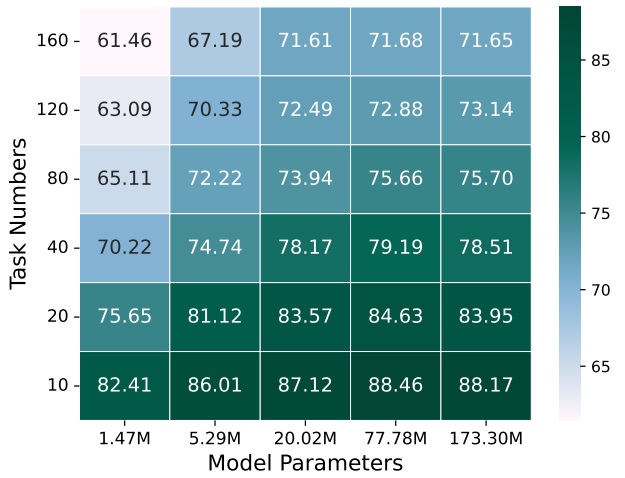

Figure 3: As the model size increases, model performance rapidly reaches its ceiling and ceases to improve thereafter. A more detailed domain-wise analysis can be seen in Appendix B.1.

iments with Prompt-DT, a language model-based method trained in supervised learning for MTRL, of varying sizes across different task scales. Detailed implementation can be found in Appendix A.5. Results in Figure 3 highlight some of the surprising phenomena, which contradict the behaviors typically observed in previous research (Lee et al., 2022; Reed et al., 2022):

In MTRL, increasing the model size of DT swiftly hits the performance ceiling, preventing sustained improvements. The horizontal comparison in Figure 3 shows that expanding the model parameters is effective within a limited range across all task scales; once the model exceeds 20M parameters, further scaling yields no meaningful improvements. This is clearly inefficient, as scalability is a key advantage of the Transformer architecture, and the performance of approaches relying solely on small models becomes increasingly constrained as the number of tasks scales. This leads to another limitation: **Scaling up the model size can *not* effectively mitigate the performance degradation caused by the increase in task numbers.** As shown by the clear decline from bottom-left to top-right in Figure 3, larger task numbers prevent the model from achieving the strong performance seen with fewer tasks, even with increased parameters. A more detailed rethinking of who blocks DT from scaling is presented in Appendix B.2.

Despite these discouraging phenomena, the heatmap reveals the most pronounced performance increase from the top-left to the bottom-right. Building on the insights from the previous section, a natural question arises:

Key Insight: How can we effectively minimize the number of tasks to be learned, while efficiently scaling model parameters, thereby maximizing performance?

4. Methodology: M3DT

From the previous observations and discussions, our key insight is that through task grouping and parameter separation, we can assign much fewer tasks to each parameter subset, leading to exceptional performance. And by expanding the number of parameter subsets while preserving their individual sizes, and simultaneously increasing the number of task groups, we can both reduce the task load on each parameter subset and enhance the model's capacity with more parameters, which enables a scalable solution to handle an increasing number of tasks by expanding model size without compromising performance.

Based on this insight, we introduce M3DT, which includes two key enhancements in both the architecture and optimization of the agent to improve the task and parameter scalability in MTRL. The first enhancement enables efficient parameter separation and expansion to reduce task load, achieved by incorporating an MoE architecture alongside the FFN in Prompt-DT, as detailed in Section 4.1. While relying solely on the advantages of the MoE structure proves ineffective to solve massive MTRL, the second enhancement includes a task grouping process and three distinct training stages that optimize the agent's learning process, especially for ensuring each parameter subset can effectively handle a specific task subset without interference, as outlined in Section 4.2. The overall framework of our method is illustrated in Figure 4.

4.1. Architecture: MoE for reducing Task Load

We propose to use an MoE architecture with scalable experts for massive tasks. The MoE structure is characterized by its composition of N modular experts and a router, $\theta_{\text{MoE}} = \{\theta_1, ..., \theta_N, \theta_r\}$, where each expert θ_i is responsible for a distinct task subset and the router θ_r dynamically assigns weights for these experts without the need for exact task IDs. Accordingly, the parameters of a given expert are updated only by gradients from its corresponding small task subset, fully leveraging the remarkable learning capability for fewer tasks. Meanwhile, by increasing the number of experts, we can further decrease the task load assigned to each expert, thereby effectively alleviating the severe gradient conflicts caused by the overwhelming task numbers and enhancing the performance.

The complete architecture of M3DT is illustrated in the right of Figure 4. We employ the complete Prompt-DT (Xu et al., 2022) as the backbone. As Appendix B.2 have investigated the crucial role of the feed-forward network (FFN) in transformer module in multi-task settings, we incorporate the MoE architecture by augmenting each Transformer block with additional experts and the router alongside the preserved FFN, which can maintain the learned shared knowledge from all tasks in the backbone model. We use networks

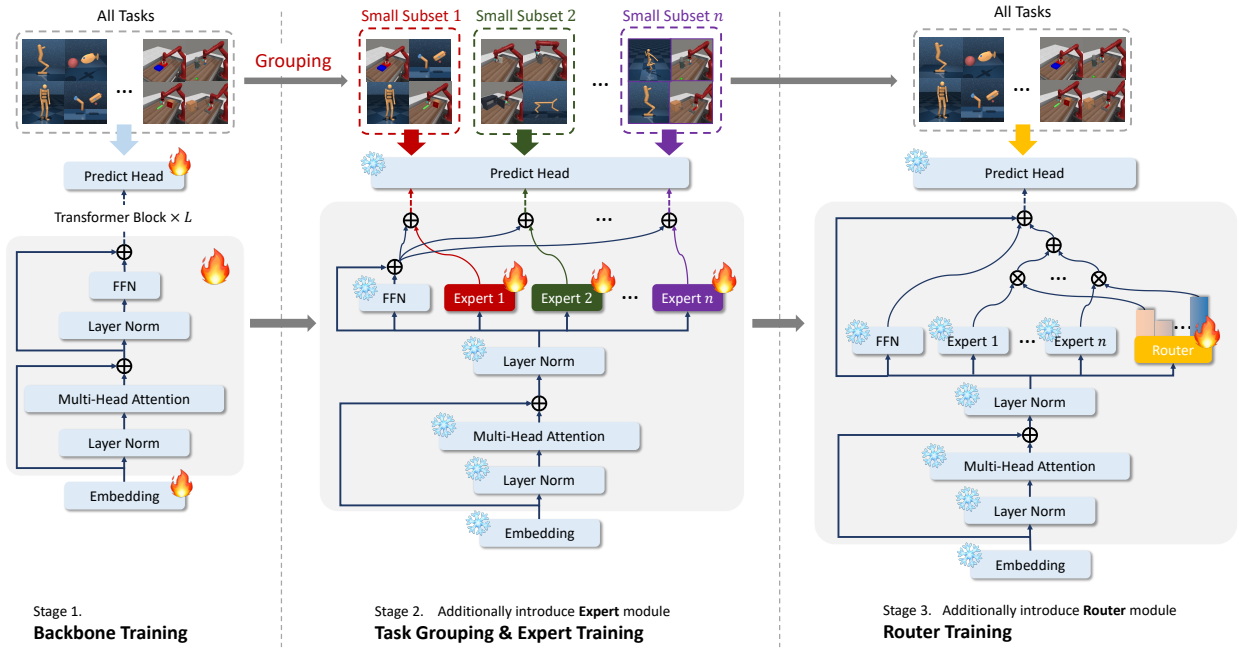

Figure 4: Overview of M3DT. **Stage 1:** We train the PromptDT on all tasks as the backbone. **Stage 2:** Here, we employ task grouping to obtain various small task subsets, and introduce expert module into every transformer block of the backbone, with each expert handling a specific task subset. In this stage, only the experts are optimized. **Stage 3:** We introduce the router to dynamically assign weights to all experts for training across all tasks. Only the router is optimized in this phase.

identical to the FFN module as experts and employ an MLP as the router. Formally, the forward propagation process of the FFN in M3DT can be simplified as follows:

$$f(x) = x + f_{\text{FFN}}(x) + f_{\text{MoE}}(x), \tag{3}$$

$$f_{\text{MoE}}(x) = \sum_{i=1}^{N} Softmax(f_{\theta_r}(x))_i \cdot f_{\theta_i}(x), \tag{4}$$

where f_{FFN} denotes the retained FFN in the backbone, and f_{θ_i} and f_{θ_r} represent the i-th expert and the router, respectively. By this, the MoE and the backbone work in tandem, enabling the agent to dynamically adjust the utilization of different experts for handling diverse types of tasks.

However, the results in Figure 5 suggest that relying solely on the advantage of the MoE is insufficient to handle massive MTRL. Upon end-to-end training of the MoE-enhanced PromptDT, gradient conflicts within the expert modules are even more pronounced than those observed in the MLP layers of PromptDT, and the overall performance improvements remain marginal. These findings underscore the necessity of refining the training paradigm.

4.2. Optimization: Three-Stage Training Mechanism

Considering our unique training purpose, namely the explicit task assignment and independent training for each expert, as well as the inherent challenges of training MoE,

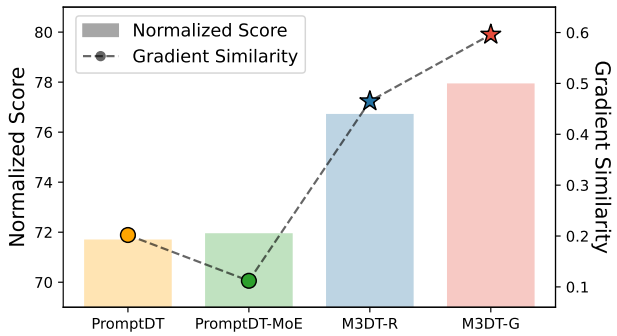

Figure 5: The inherent structural benefits of MoE alone are insufficient, while task grouping and three-stage training paradigm can substantially enhance model performance while alleviating gradient conflicts. The number of experts is configured to 40.

we propose a three-stage training mechanism, with distinct training processes designed for the backbone, experts, and router, each trained separately. The overall training process is illustrated in Figure 4.

Backbone training with minimal gradient conflicts. We first train Prompt-DT on all tasks to capture shared knowledge, enabling it to embed information and predict actions across tasks, which serves as the backbone architecture for M3DT. This training process results in intense gradient conflicts, as illustrated in Figure 2. To gain deeper insight into the progression of gradient conflicts during training, we

Table 1: Comparison of M3DT with baselines of varying sizes across three task scales. M3DT consistently outperforms other baselines and achieves remarkable task scalability, which is attributed to its parameter scalability, as shown in Figure 6.

Task Scale	10 Tasks		80 Tasks		160 Tasks	
Method	Score	Parameters	Score	Parameters	Score	Parameters
MTDT-Small	82.75 ± 2.29	1.47M	66.65 ± 0.43	1.47M	59.19 ± 1.77	1.47M
MTDT-Large	88.92 ± 0.98	173.30M	74.38 ± 1.33	173.30M	70.65 ± 1.67	173.30M
PromptDT-Small	82.41 ± 1.47	1.47M	65.11 ± 2.65	1.47M	61.46 ± 1.78	1.47M
PromptDT-Large	88.17 ± 1.41	173.30M	75.70 ± 1.62	173.30M	71.65 ± 1.13	173.30M
HarmoDT-Small	79.75 ± 1.32	1.47M	60.71 ± 3.91	1.47M	57.27 ± 0.84	1.47M
HarmoDT-Large	86.63 ± 1.28	173.30M	75.56 ± 1.69	173.30M	72.80 ± 2.89	173.30M
M3DT-Random (Ours)	89.09 ± 1.20	47.87M	78.92 ± 1.21	98.37M	76.74 ± 0.94	174.12M
M3DT-Gradient (Ours)	$\mathbf{89.23} \pm 1.12$	47.87M	$\mathbf{80.66} \pm 0.97$	98.37M	$\mathbf{78.21} \pm 0.47$	174.12M

conduct experiments that illustrate how performance and gradient conflicts develop across training iterations in Figure 7 (left). Initially, the model's performance improves rapidly with gradient conflicts escalate from a low initial value, enabling fast knowledge acquisition. However, as training progresses, gradient conflicts rapidly reach the peak, leading to diminishing performance gain. Based on this, we restrict training of the backbone to the early stage before gradient conflicts reach their peak. This strategy allows the model to efficiently learn shared knowledge, ensuring the shared parameters align with the solution space while preventing excessive updates on dominant tasks, thereby minimizing negative interference with conflicting tasks. More discussion is detailed in Section 5.2. This approach provides a solid foundation for the subsequent expert training.

Task grouping and experts individually training. We employ task grouping and then train the experts on smaller task subsets, which not only reduces the task load to achieve better performance, but also mitigates the severe gradient conflicts that arise in shared parameters after a certain stage of training. We begin by proposing two task grouping methods: (1) random grouping: based on the satisfactory results on few tasks as illustrated in Figure 2, we randomly divide all tasks into equally sized subsets for naively reducing the task number to a lower value; (2) gradient-based grouping: we first calculate the agreement vectors (Hu et al., 2024) (as detailed in Appendix C) for each task in the current backbone as a measure of task similarity, and then apply K-means to group these vectors and the corresponding tasks. After obtaining the task subsets, we introduce a dedicated expert for each subset. The expert module is trained on its specific task subset with the backbone parameters frozen, preserving the shared knowledge while allowing the expert to focus on learning task-specific information. By limiting the size of each task subset, we can ensure effective learning on these parameter subsets. Additionally, training each expert independently helps mitigate the issue of imbalanced updates among experts in MoE. Consequently, when dealing with large-scale tasks, we can expand the number of experts and task groups to mitigate the severe performance drop. Figure 5 illustrates that grouping tasks and independently training distinct experts can significantly alleviate gradient conflicts within the expert modules, resulting in notable improvements in the overall performance.

Router training. Finally, we train the router on all tasks, enabling it to dynamically assign weights to different well-trained experts for various tasks based on the hidden states from the backbone, , integrating all sub-policies into a unified strategy without requiring task IDs. In this stage, we freeze the parameters of both the backbone and all experts, allowing only the router to be optimized. This strategy significantly simplifies the training of the MoE while preserving the knowledge already acquired by the model.

5. Experiments and Analysis

In this section, we conduct extensive experiments to answer the following questions: (1) How does M3DT compare to other baselines in the massive multi-task regime? (2) Does M3DT exhibit task scalability and parameter scalability? (3) What makes M3DT effective? The detailed implementations[1], including the baselines and environments, are illustrated in Appendix A.

5.1. M3DT helps Task and Parameter Scalability

In this study, we benchmark M3DT and its variants against baselines on different task scales. We compare the performance of the baselines at their default size and a expanded size. The variants of M3DT considered in this evaluation include **M3DT-Random**, which employ random grouping to obtain task subsets and train experts; **M3DT-Gradient**, which utilizes gradient information for grouping.

As shown in Table 1, M3DT-Random surpasses all other methods at all task scales, achieving a 0.1%, 4.3% and 5.4%

[1]Our code is available at: https://github.com/KongYilun/M3DT

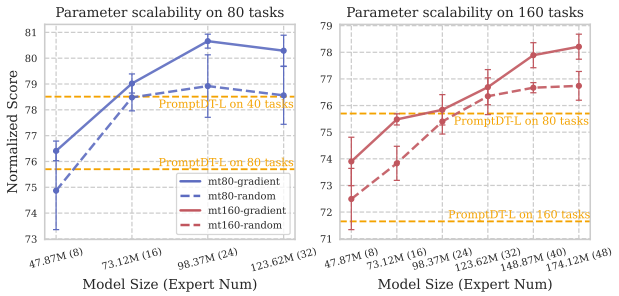

Figure 6: By increasing the number of experts, M3DT effectively unlocks the parameter scalability, further helping to tackle task scalability.

improvement in 10, 80 and 160 tasks, respectively, compared to the best baseline. The advantages of our method become more pronounced as the number of tasks increases. By employing random grouping, M3DT-Random already effectively competes with the current state-of-the-art techniques, highlighting the significant effectiveness of introducing parameter subsets and reducing corresponding task load. Furthermore, M3DT-Gradient enhances the performance by identifying better task groups to mitigate learning complexity, resulting in substantial gains of 6.6% and 7.5% in 80 tasks and 160 tasks, respectively. M3DT also effectively alleviates the performance degradation with increasing task numbers. Compared to the severe performance drop of around 20% observed in other baselines when scaling tasks numbers from 10 to 160, our approach only experienced a 12.3% decline. And M3DT achieves a higher score on 160 tasks than other baselines do on 80 tasks, demonstrating remarkable scalability across task numbers.

The scalability of M3DT in handling massive tasks is attributed to its parameter scalability, which can meanwhile reduce the task load on each parameter subset by increasing the number of experts, thereby enhancing overall performance. We conduct experiments on the number of experts across 80 tasks and 160 tasks, as shown in Figure 6. For 10 tasks, 8 experts already yield excellent results, achieving scores of 89.09 and 89.23 with 47.87M parameters. In 80 and 160 tasks, expanding the parameter size within a certain range through increasing the number of experts can significantly improve model performance, achieving a 11.2% and 11.7% improvement, separately. Since both M3DT-Random and M3DT-Gradient show diminishing performance gains after reaching 40 experts on 160 tasks, we use 40 experts in subsequent experiments unless stated otherwise. Additionally, we analyze the reasons why increasing the number of experts cannot continually improve performance and eventually reaches a performance ceiling in later experiments.

5.2. Further Analysis

Is grouped training important? To investigate whether the success of M3DT is primarily due to the explicit task grouping and three-stage training mechanism, or the inherent advantages of the MoE architecture itself, we conduct a detailed ablation study, as illustrated in Table 2. Despite utilizing the MoE structure, the absence of our meticulously designed training strategy leads to a substantial performance degradation. Training a PromptDT with MoE end-to-end from scratch (i.e. M3DT w/o 3-stage training) only yields results similar to those of a standalone PromptDT with the same parameter scale, while freezing the trained backbone and jointly training all experts and the router on all tasks (i.e. M3DT w/o grouping) results in a worse performance, with a score of 67.34. The comparison of gradient conflicts is shown in Figure 5. In addition, after training the experts in groups, simultaneously fine-tuning them when training the router (i.e. both M3DT-R/G w/o expert freezing) also leads to suboptimal results. This further underscores the validity of our entire framework.

Table 2: Ablation study on different training process of M3DT, which illustrate the effectiveness of our dedicated three-stage training mechanism.

Method	Normalized Score
M3DT-R	76.67 ± 0.29
M3DT-G	77.89 ± 0.47
M3DT w/o 3-stage training	71.90 ± 0.70
M3DT w/o grouping	67.34 ± 0.56
M3DT-R w/o expert freezing	71.89 ± 0.63
M3DT-G w/o expert freezing	71.88 ± 0.62

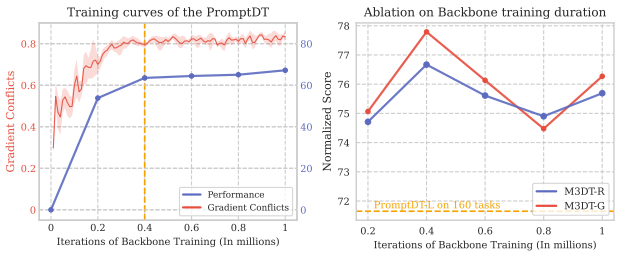

Figure 7: (left) Training curves of the PromptDT, where the early stage exhibits mild gradient conflicts with swift performance gain; (right) The overall performance of M3DT varies with the number of training steps applied to the backbone, with optimal performance occurring just as gradient conflicts reach the peak.

Is early stopping of backbone training truly effective? As illustrated in Figure 7 (left), the gradient conflicts in the PromptDT backbone progressively escalate with training duration, eventually reaching a peak, after which the performance improvements become notably sluggish. To assess the efficacy of early stopping, which is employed to mitigate these severe gradient conflicts at the expense

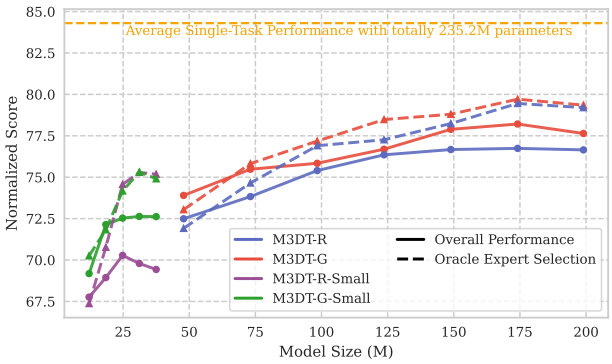

Figure 8: Performance curves of M3DT with increasing model size (i.e., the number of experts) across different base sizes, where each point denotes the addition of 8 experts.

of some performance gains, we conduct a comparison of M3DT using PromptDT trained for varying steps as backbone. The results are depicted in Figure 7 (right). M3DT consistently achieves strong performance across various training durations, with early stopping at 400k steps, just as gradient conflicts peak, yielding the most optimal results. This demonstrates the robustness of M3DT in the initial training phase. Training for only 200k steps prevents the backbone from fully learning the shared knowledge, causing the parameters to deviate from the optimal solution space, resulting in suboptimal performance. On the other hand, continuing training beyond 400k steps, when gradient conflicts are extremely severe, also leads to a decline in M3DT's performance. This is likely due to the parameters become overfitted to tasks whose gradients dominate, causing them to deviate further from the solution space of other conflicting tasks. Introducing expert modules at this time for grouped task learning also falls victim to this issue, as they unable to learn the specific knowledge on these conflicting tasks, thus failing to alleviate the performance drop.

What affects parameter expansion? In this study, we introduce: **Oracle Expert Selection:** manually selecting each test task's corresponding expert based on Task ID, individually evaluating their performance on each task, calculate the averaged results as the upper-bound performance of M3DT, as task IDs are unavailable in overall method evaluation and real-world applications; **Small:** a scaled-down version of M3DT, where the width of backbone, experts, and router are all proportionally reduces. As shown in Figure 8, the main factors influencing the parameter scalability of M3DT are threefold: (1) Model width: With the number of experts increasing, M3DT-Small achieves significant performance improvements with modest growth in parameters. However, due to the constrains of the small model width on the capacity of each module, its scalability to the number of experts is poor, reaching the performance ceiling at only 24 experts. As a result, its parameter scalability is inherently constrained. (2) Router: The performance gap between

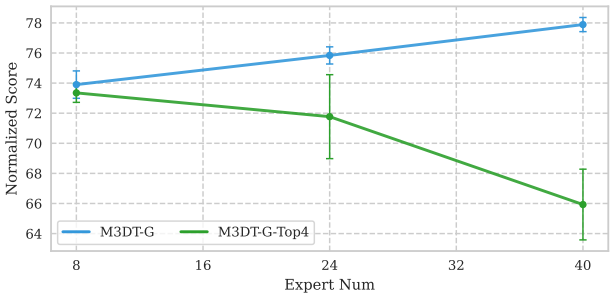

Figure 9: Top-K routing selects the router's top-k outputs, applies softmax to obtain probabilities, and computes the weighted sum of the corresponding expert outputs. M3DT with Top-4 router fails to scale with the number of experts.

the dashed and solid lines reflects the performance loss attributed to the router. When the number of experts is small, the router can easily allocate weights across the few experts, resulting in minimal performance loss or even better results. As the number of experts increases, the difficulty of assigning weights to the experts grows, resulting in a progressively larger performance gap, which peaks when the expert performance continues to improve while the overall performance plateaus or even drops. Thus, it is reasonable and tolerable that the framework's overall performance may be lower than the ideal case of perfect expert switch, as increasing the number of experts inherently raises the difficulty of optimal weight allocation. (3) Backbone + Expert: Expert performance also tends to plateau when the number of experts becomes sufficiently large. This is primarily due to the shared knowledge learned by the backbone across all tasks is limited, which restricts further performance improvements, regardless of the expert's capabilities. Additionally, when the number of experts is large enough and each expert already faces a sufficiently small task subset, further increasing the number of experts yields diminishing returns in reducing the task load.

Router design. We compare the Top-K routing (Shazeer et al., 2017) strategy to investigate whether reducing the number of activated experts can enhance the scalability of expert number. In this experiment, we employed Top-4 routing, and the results are presented in Figure 9. Although our proposed three-stage training mechanism significantly simplifies the training of MoE, Top-4 router fails to scale the number of experts, and performance deteriorates as expert number increases. This is attributed to improper routing load balancing, where certain routes are excessively optimized, combined with the instability in router training induced by sparsity. This finding is consistent with prior work demonstrating the difficulty in scaling up deep RL networks with Top-K router (Obando-Ceron et al., 2024). While it is possible some losses (Riquelme et al., 2021; Mustafa et al., 2022) may result in better Top-K performance, this finding suggests that M3DT benefits from having a weighted combination of all experts from all task subsets.

6. Related Work

Multi-Task RL Multi-task RL aims to learn a shared policy for a diverse set of tasks. One of the most straightforward approaches to MTRL is to formulate the multi-task model as task-conditional sequence modeling (Xu et al., 2022; Reed et al., 2022; Lee et al., 2022; He et al., 2023; Hu et al., 2024). Some methods also focus on handling gradient conflicts among different tasks (Yu et al., 2020a; Chen et al., 2020; Liu et al., 2021). On the other hand, some methods employed a dedicated shared structure to leverage the shared knowledge (D'Eramo et al., 2024; Yang et al., 2020; Sun et al., 2022). While most of these methods address tens or even less tasks, we focus on scaling to a significantly larger number of tasks, achieving task scalability.

Mixture-of-Experts MoEs have recently helped scaling language models up to trillions of parameters thanks to their modular nature (Lepikhin et al., 2020; Fedus et al., 2022). MoEs also help performance in multi-task settings (Fan et al., 2022; Ye & Xu, 2023; Dou et al., 2024). There have been few works exploring MoEs in RL for single (Akrour et al., 2021; Obando-Ceron et al., 2024) and multi-task learning (Hendawy et al., 2023; Huang et al., 2024). While M3DT explicitly assign tasks to specific experts and train each expert independently to handle multi-task problems.

7. Limitation

We primarily focus on the challenges in MTRL, analyze the scalability of task numbers and parameter size , and propose a paradigm that increases model parameters to achieve task scalability. However:

(1) we did not focus on fine-grained network architecture design, such as the expert and router modules. Further optimization of these components could potentially improve performance while reducing model size.

(2) While we leave held-out task generalization and continual learning unexplored, our method's modular parameterization and group-wise learning naturally supports solutions like held-out task adaptation via expert learning, dynamic router-based skill composition for held-out tasks, and forgetting mitigation mechanisms for the router for past tasks. Since generalization and continual learning are highly critical in RL, resolving them presents promising future works.

(3) Scaling experts with task count raises inference costs. While we prioritize algorithm performance above all else, we experimented with activating Top-k experts with fixed k to control inference overhead, but it degrades the performance. A more tailored Top-k gating mechanism, better aligned with our three-stage training paradigm, could enhance the practical applicability of our method.

8. Conclusion

In this study, we first delve into the challenges of sequence modeling in MTRL, analyze the performance degradation and gradient conflicts with increasing task numbers, and identify the limited effectiveness of naively scaling parameter in MTRL. Based on these insights, we introduce M3DT, a novel approach designed to unlock the parameter scalability for handling massive tasks. By employing task grouping and MoE, M3DT significantly reduces the task load assigned to each parameter subset while enhancing the overall performance through the expansion of experts. We propose a three-stage training mechanism that allows explicit task assignment to each expert, enabling sequential training of different modules without interference. Our empirical evaluations across diverse task scales underscore M3DT's superior performance compared to existing baselines, establishing its state-of-the-art effectiveness in MTRL scenarios.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

This work is supported by STI 2030-Major Projects (No. 2021ZD0201405), Shenzhen Basic Research Project (Natural Science Foundation) Basic Research Key Project (NO. JCYJ20241202124430041), National Research Foundation, Singapore, under its NRF Professorship Award No. NRF-P2024-001.

References

Akrour, R., Tateo, D., and Peters, J. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6795–6806, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. Just pick a sign: Opti-

Mastering Massive Multi-Task Reinforcement Learning via Mixture-of-Expert Decision Transformer

mizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33: 2039–2050, 2020.

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. Sharing knowledge in multi-task deep reinforcement learning. *arXiv preprint arXiv:2401.09561*, 2024.

Dou, S., Zhou, E., Liu, Y., Gao, S., Shen, W., Xiong, L., Zhou, Y., Wang, X., Xi, Z., Fan, X., et al. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024.

Fan, Z., Sarkar, R., Jiang, Z., Chen, T., Zou, K., Cheng, Y., Hao, C., Wang, Z., et al. M^3vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–28457, 2022.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hansen, N., Su, H., and Wang, X. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

He, H., Bai, C., Xu, K., Yang, Z., Zhang, W., Wang, D., Zhao, B., and Li, X. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36:64896–64917, 2023.

Hendawy, A., Peters, J., and D'Eramo, C. Multi-task reinforcement learning with mixture of orthogonal experts. *arXiv preprint arXiv:2311.11385*, 2023.

Hu, S., Fan, Z., Shen, L., Zhang, Y., Wang, Y., and Tao, D. Harmodt: Harmony multi-task decision transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.18080*, 2024.

Huang, S., Zhang, Z., Liang, T., Xu, Y., Kou, Z., Lu, C., Xu, G., Xue, Z., and Xu, H. Mentor: Mixture-of-experts network with task-oriented perturbation for visual reinforcement learning. *arXiv preprint arXiv:2410.14972*, 2024.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kong, Y., Mao, H., Qi, Z., Zhang, B., Ruan, J., Shen, L., Chang, Y., Wang, X., Zhao, R., and Tao, D. QPO: Query-dependent prompt optimization via multi-loop offline reinforcement learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=bqMJToTkvT.

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35: 27921–27936, 2022.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Mustafa, B., Riquelme, C., Puigcerver, J., Jenatton, R., and Houlsby, N. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems*, 35:9564–9576, 2022.

Obando-Ceron, J., Sokar, G., Willi, T., Lyle, C., Farebrother, J., Foerster, J., Dziugaite, G. K., Precup, D., and Castro, P. S. Mixtures of experts unlock parameter scaling for deep rl. *arXiv preprint arXiv:2402.08609*, 2024.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34: 8583–8595, 2021.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

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

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., and Gan, C. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pp. 24631–24645. PMLR, 2022.

Yang, R., Xu, H., Wu, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

Ye, H. and Xu, D. Taskexpert: Dynamically assembling multi-task representations with memorial mixture-of-experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21828–21837, 2023.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33: 5824–5836, 2020a.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020b.

A. Experimental Details

A.1. Detailed Environments

We consider a total of 160 continuous control tasks from 3 task domains: Meta-World (Yu et al., 2020b), DMControl (Tassa et al., 2018), Mujoco Locomotion (Todorov et al., 2012). This section provides an exhaustive introduction of the tasks considered, including their observation and action dimensions, and the calculation of normalized score. Our goal is not to propose a new benchmark with 160 tasks, but rather to use a sufficiently large number of tasks to explore the impact of task quantity.

A.1.1. Meta-World

The Meta-World benchmark encompasses a diverse array of 50 distinct manipulation tasks, unified by shared dynamics. These tasks involve a Sawyer robot engaging with a variety of objects, each distinguished by unique shapes, joints, and connective properties. The complexity of this benchmark lies in the heterogeneity of the state spaces and reward functions across tasks, as the robot is required to manipulate different objects towards varying objectives. The robot operates with a 4-dimensional fine-grained action input at each timestep, which controls the 3D positional movements of its end effector and modulates the gripper's openness. The state space is unified into 39 dimension. In its original configuration, the Meta-World environment is set with fixed goals, a format that somewhat limits the scope and realism of robotic learning applications. To address this and align with recent advancements in the field, as noted in works by (Yang et al., 2020; Sun et al., 2022; He et al., 2023; Hu et al., 2024), we have modified all tasks to incorporate a random-goal setting. For the offline dataset, we follow the works (He et al., 2023; Hu et al., 2024) and utilize their dataset with the near-optimal trajectories, which consists of the experience from random to expert (convergence) in SAC-Replay (Haarnoja et al., 2018). The primary metric for evaluating performance in this benchmark is the average success rate across all tasks, providing a comprehensive measure of the robotic system's adaptability and proficiency in varied task environments. We directly use the success rate on each task as its normalized score.

A.1.2. DMControl

The tasks in DMControl involve significantly more diverse embodiments, state spaces, action spaces, and reward functions, which greatly increases their complexity. We consider a total of 30 continuous control tasks in the DMControl domain, including 19 original DMControl tasks and 11 new (custom) tasks created specifically for M3DT benchmarking, following the work (Hansen et al., 2023). We directly use the dataset collected by (Hansen et al., 2023), and we only use the first 2,000 trajectories for each task to ensure consistency in dataset size with other tasks. We list all used DMControl tasks in Table 3. For evaluation, we linearly scale the original reward range of [0,1000] to [0,100], using it as our normalized score.

A.1.3. Mujoco Locomotion

In this paper, we also employ a diverse array of meta-RL control tasks to construct a dataset with a sufficient number of tasks for exploring the challenges of MTRL when confronted with a large number of tasks. We directly utilize the datasets proposed by Xu et al. (2022). The tasks are detailed as follows:

- **Cheetah-vel** (Finn et al., 2017): It defines 40 unique tasks, each associated with a specific goal velocity, uniformly distributed between 0 and 3 m/s. The agent's performance is assessed based on the l2 error relative to the target velocity, with a penalty for deviations. These 40 tasks share a unified state space of 20 and an action space of 6. Based on the reward ranges of these environments, we linearly map the return values within the interval [-100, -30] to the normalized range of [0, 100] as our normalized scores, while returns outside this range are directly capped at 0 or 100, respectively.

- **Ant-dir** (Rothfuss et al., 2018): We also use 40 tasks in this domain, each with a goal direction uniformly sampled in a two-dimensional plane. The agent, an 8-jointed ant, is incentivized to attain high velocity in the designated direction. The state space for these tasks has a dimensionality of 27, and the action space consists of 8 dimensions. We linearly map the return values within the interval [0, 500] to the normalized range of [0, 100], while returns outside this range are directly capped at 0 or 100, respectively, to calculate our normalized scores.

By using normalized scores, we can align tasks with initially inconsistent evaluation metrics, enabling us to assess the model's ability to simultaneously tackle multiple tasks. For the calculation of average normalized score, we assign equal weight to each of the 160 tasks and compute the average of normalized scores.

Table 3: DMControl tasks used in this paper.

Task	Observation dim	Action dim	New?
Acrobot Swingup	6	1	N
Cartpole Balance	5	1	N
Cartpole Balance Sparse	5	1	N
Cartpole Swingup	5	1	N
Cartpole Swingup Sparse	5	1	N
Cheetah Jump	17	6	Y
Cheetah Run	17	6	N
Cheetah Run Back	17	6	Y
Cheetah Run Backwards	17	6	Y
Cheetah Run Front	17	6	Y
Cup Catch	8	2	N
Cup Spin	8	2	Y
Finger Spin	9	2	N
Finger Turn Easy	12	2	N
Finger Turn Hard	12	2	N
Fish Swim	24	5	N
Hopper Hop	15	4	N
Hopper Hop Backwards	15	4	Y
Hopper Stand	15	4	N
Pendulum Spin	3	1	Y
Pendulum Swingup	3	1	N
Reacher Easy	6	2	N
Reacher Hard	6	2	N
Reacher Three Easy	8	3	Y
Reacher Three Hard	8	3	Y
Walker Run	24	6	N
Walker Run Backwards	24	6	Y
Walker Stand	24	6	N
Walker Walk	24	6	N
Walker Walk Backwards	24	6	Y

To address the inconsistency in state and action spaces across tasks, we zero-pad all states and actions to their largest respective dimensions (i.e. 39 and 8, respectively), and mask out invalid action dimensions in predictions made by the policy during both training and inference.

A.2. Baselines

We compare our proposed M3DT with the following DT-based baselines.

- **MTDT**: We extend the DT architecture (Chen et al., 2021) to learn from multitask data. Specifically, MTDT concatenates an embedding z and a state s as the input tokens, where z is the encoding of task ID. In evaluation, the reward-to-go and task ID are fed into the Transformer to provide task-specific information. Leveraging the scalability of the Transformer architecture, we compare the performance of this method at both its default size (1.47M) and expanded size (173.30M).
- **PromptDT** (Xu et al., 2022): PromptDT built on DT aims to learn from multi-task data and generalize the policy to unseen tasks. It leverages short task trajectories as prompts to guide the model in identifying the current task. PromptDT generates actions based on the trajectory prompts and reward-to-go. We compare the performance of this method at both its default size (1.47M) and expanded size (173.30M).
- **HarmoDT** (Hu et al., 2024): HarmoDT built on PromptDT aims to lean a specific mask for each task, effectively shielding the model's parameters that conflict most with the task, thereby mitigating the severe gradient conflicts in

MTRL. During evaluation, HarmoDT also generates actions based on the trajectory prompts and reward-to-go. We compare the performance of this method at both its default size (1.47M) and expanded size (173.30M).

A.3. Task Selection

To mitigate the impact of task selection and combination on performance, we use different task combinations for each run seed in scenarios involving 10, 20, 40, 80, and 120 tasks. To fairly analyze the effects solely caused by task quantity, we maintain consistent average task difficulty across various task scales and seeds. Specifically, we train 160 separate PromptDT models for each of the 160 tasks and compute the score for each task, which serves as the task difficulty for that task. When selecting tasks for different task scales, we ensure that the average task difficulty of the chosen task set aligns with that of the complete set of 160 tasks. We use the 1.47M PromptDT model to compute the task difficulty and derive the task sets for all task scales. Subsequently, all experiments are conducted using the same task set to eliminate potential bias.

In summary, to achieve methodological rigor in task selection, we follow three key criteria:

- **Performance Balance**: Based on single-task normalized score, the selected task subsets maintain average scores comparable to the full set of 160 tasks, to eliminate the influence of task difficulty;
- **Domain Ratio Preservation**: We make the subset composition at different task scales approximately maintains the original domain ratio (MW:DMC:Ant:Cheetah=5:3:4:4), in order to mitigate domain-specific biases;
- **Subset Diversity**: We employ distinct task subsets for each run with the same task scale to maximize coverage of the full task pool, eliminating spurious correlations between specific tasks.

A.4. Gradient Simalarity

We compute the gradients for each task and calculate the mean gradient across all tasks. The **gradient similarity** is defined as the average cosine similarity between this mean gradient and all tasks' gradients. During training, we record this metric every 1e4 steps to obtain a similarity curve. After smoothing to the stabilized phase of training, we use the plateau value as our final gradient similarity. **Gradient conflict** is calculated as (1 - gradient similarity).

A.5. Implementation of Model Expansion

In this experiment, we do not invest significant effort into exploring the optimal structure for each baseline model at different parameter scales. Instead, we adopt their default number of layers and attention heads, only increasing the width of the models, i.e. the dimension of their hidden states. Although previous research (Kaplan et al., 2020) have shown that the model structure has only a marginal impact on performance when the parameter size remains constant, we conduct experiments with PromptDT to verify whether our expanding approach is reasonable in the context of MTRL. The results are shown in Figure 10, where d_{model} denotes the width of the model, n_{layer} and n_{head} denote the number of layers and attention heads, respectively. In MTRL, the model performance is also weakly depends on the model architecture. We summarize the specific structures of each scaled model in Table 4.

Table 4: DMControl tasks used in this paper.

Parameters	Layers	Attention Heads	Model Width
1.47M	6	8	128
5.29M	6	8	256
20.02M	6	8	512
77.78M	6	8	1024
173.30M	6	8	1536

A.6. Experimental Setups, Hyper-parameters and Resources

In this section, we introduce the implementation for M3DT. We employ the PromptDT with 5.29M parameters as the backbone of M3DT, with the structure outlined in Table 4. The structures of our introduced experts are identical to that of the FFN in the backbone. We employ a 5-MLP as our router. The padded input and output dimensions are 39 and 8,

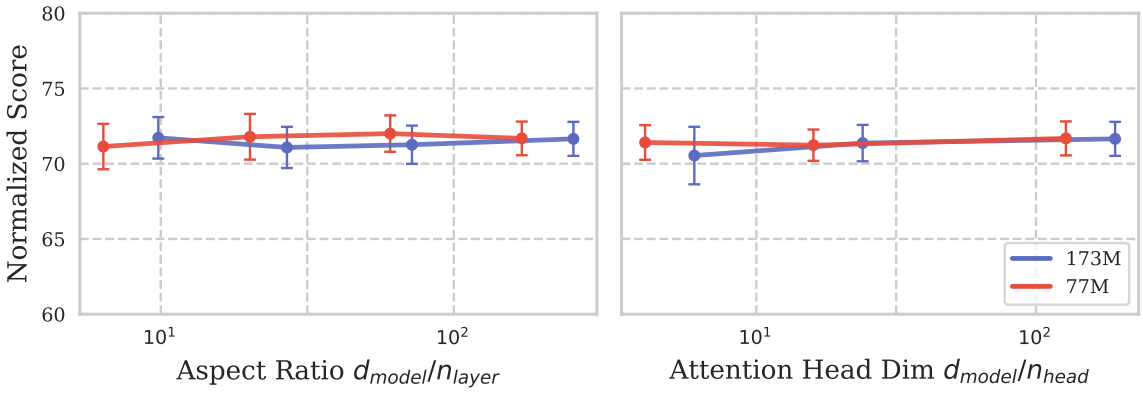

Figure 10: Performance depends very mildly on model shape when the total number of parameters is held fixed.

respectively, as illustrated in A.1. All experiment in this paper are run with 3 seeds. The specific model parameters and hyper-parameters utilized in our training process are outlined in Table 5. We use NVIDIA GeForce RTX 4090 to train and evaluate each model except HarmoDT-Large, while it is trained and evaluated on NVIDIA A100 40G due to its substantial resource requirements.

Table 5: Hyper-parameters of M3DT in our experiments.

Parameter	Value
Number of layers	6
Number of attention heads	8
Hidden dimension	256
Number of experts	[8,16,24,32,40,48]
Nonlinearity function	ReLU
Batch size	16
Prompt length K	20
Dropout	0.1
Learning rate	1.0e-4
Optimizer	Adam
Total rounds	1e6
-Backbone training rounds	4e5
-Expert training rounds	2e5
-Router training rounds	4e5

B. More Experiments and Further Analysis

B.1. Domain-wise Analysis of the Scalability of Task Numbers and Model Size

We separately report the scores in respective domains to eliminate the potential conclusion bias caused by varying difficulty levels across domains. The model is still trained on mixed tasks from all four domains (MW + DMC + Ant + Cheetah). The Total number on the y-axis denotes the overall trained tasks from the four domains, while the Domain number in parentheses indicates the number of tasks belonging to the specific domains in the training set, and the score represents the average performance across these tasks. **Both task scalability and parameter scalability exhibit the same trends as demonstrated in the original paper**.

For the parameter scalability, the performances on MW+DMC exhibit the same performance patterns as observed in Fig. 3. For Ant+Cheetah, PromptDT achieves near-optimal performance at original size when trained on limited tasks, while also shows the same pattern of rapidly reaching performance saturation with increased parameters under large-scale tasks. These collectively confirm that including Ant and Cheetah as 80 tasks does not introduce bias to our parameter scalability conclusion.

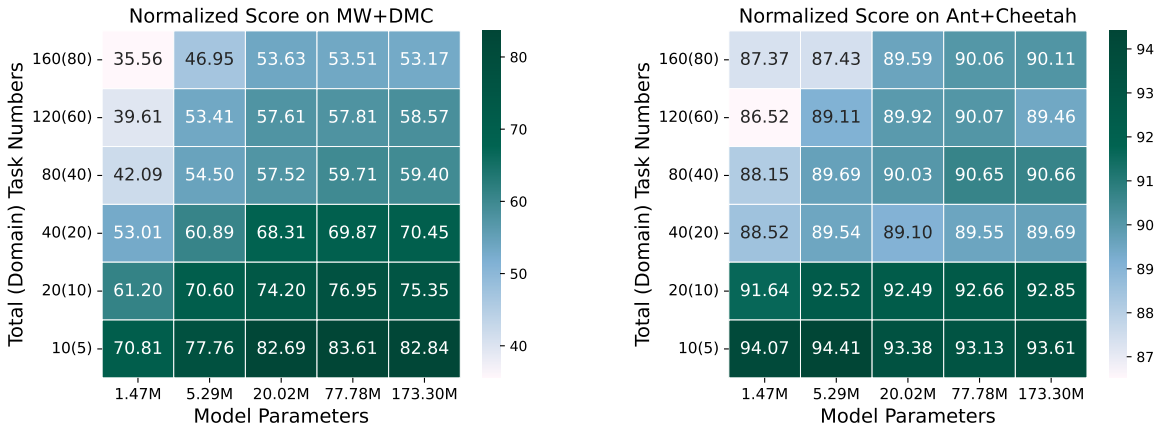

Figure 11: The task scalability and parameter scalability exhibit the same trends as demonstrated in Section 3.

From the perspective of task scalability, with the criteria of Domain Ratio Consistency ensuring proportional task sampling across different scales, both MW+DMC and Ant+Cheetah scores exhibit identical scaling patterns: sharp drop from 10 to 80 tasks and plateau from 80 to 160 tasks, which further confirms our conclusion.

B.2. Who blocks DT from scaling in MTRL

To further systematically investigate the limiting factors in DT model scaling, we conduct module-wise dimensional expansion to analyze each component's impact on MTRL performance. The results are illustrated in Figure 12.

Through layer-wise analysis of gradient similarity across Transformer blocks, we identify significantly stronger gradient conflicts in MLP compared to Attention, as shown in Figure 12a. And Figure 12b and Figure 12c indicate that separately expanding the dimensions of either MLP or attention layers demonstrates measurable efficacy in mitigating gradient conflicts within their respective modules, albeit to varying degrees. Additionally, as seen in Figure 12d, independently enhancing the dimensionality of either the MLP or attention layers can increase the overall gradient similarity of the model.

Nonetheless, the results in Figure 12d suggest that despite separate parameter expansion, the model's overall gradient conflict still converges to a fairly high value. In terms of model performance, while an initial expansion of the MLP yields some improvement, continued enlargement results in performance degradation. On the other hand, expanding the Attention module directly causes a performance drop, suggesting that the default-sized Attention already possesses sufficient representation modeling capability. Given that the MLP is a key component for enhancing performance and mitigating gradient conflicts, but merely expanding its parameters yields limited benefits, what approach should we take?

B.3. Domain-wise Main Results

We also separately report the normalized scores of our method and baselines in respective domains in Table 6 and Table 7. M3DT shows more superior performance on complex tasks (MW+DMC).

B.4. Training Time Overhead and Computational Cost

Our experiments are conducted on RTX 4090 GPU. In the first stage, we train PromptDT-5M for 4e5 steps, which takes approximately 5.2 hours. In the second stage, since each expert trains independently on its dedicated task group, all experts train in parallel. We record the training time of the overall second stage as training a single expert for approximately 1.8 hours. In the third stage, we train the router for 4e5 steps, which takes 17.2 hours. The primary reason for the long training time is our code implementation's lack of parallel computation for experts. Code optimization may reduce this training time. Thus, the total training time of our method amounts to approximately **24.2 hours**. In comparison, training both PromptDT-173M and MTDT-173M requires similar durations of around **21.3 hours**, whereas training HarmoDT-173M demands **95.6 hours**.

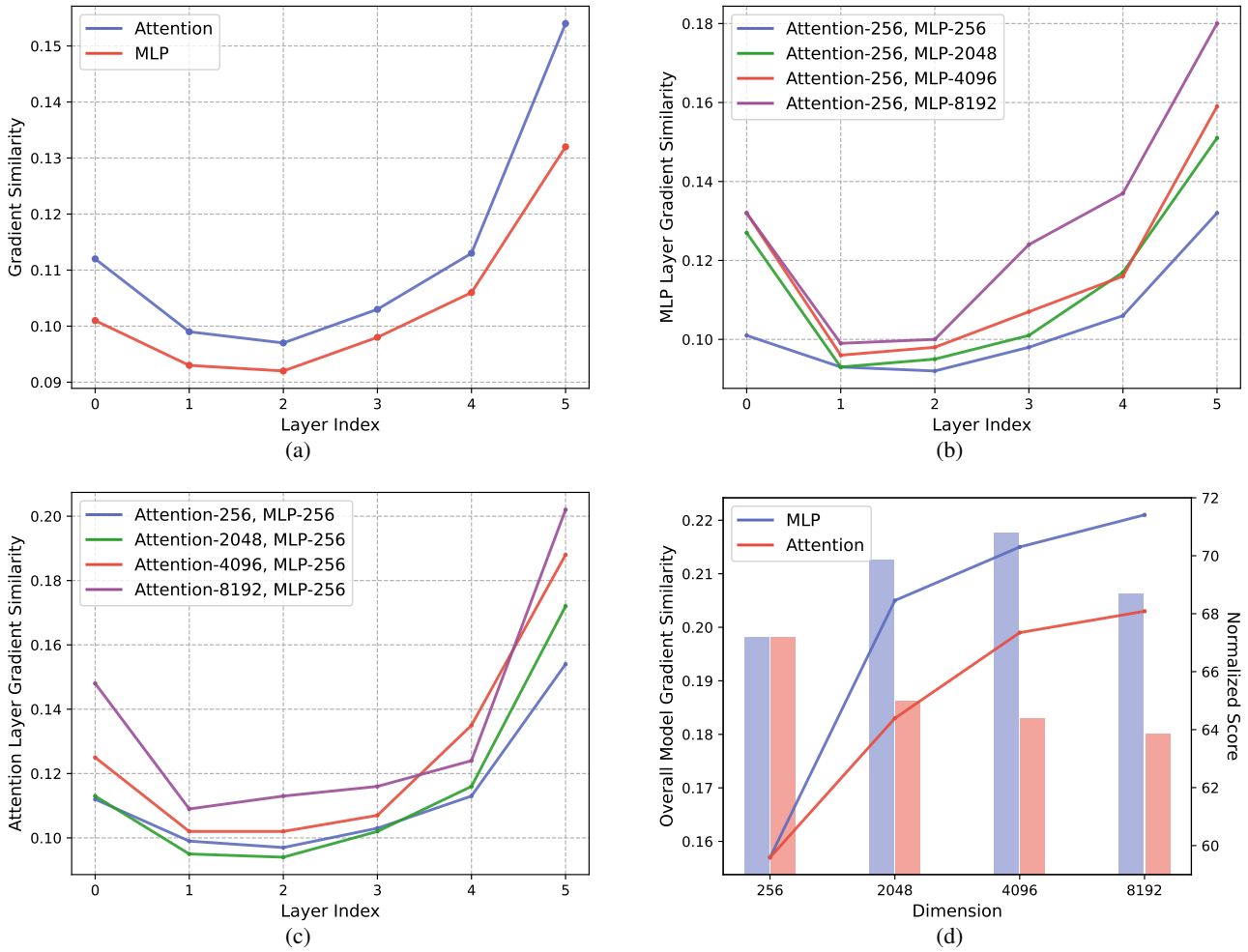

Figure 12: (a) MLP layers exhibit more severe gradient conflicts compared to Attention layers within each Transformer layer. (b) Expanding the dimension of MLP can mitigate gradient conflicts in the MLP layers. (c) Expanding the dimension of Attention also helps alleviate gradient conflicts in the Attention layers. (d) Despite dimensional expansion across different modules, the model's overall gradient conflicts converge to a severe level, with limited practical performance gains.

B.5. Expert design.

Our proposed MoE architecture replaces the FFN with an MoE in each transformer block. This is based on what is common practice when adding MoEs to transformer architectures, but is by no means the only way to utilize MoEs. Here we investigate a variant: **Big:** Each expert is a full transformer architecture, where the embedding layer and prediction layer are shared in the backbone. However, M3DT-Gradient-Big only results in a normalized score of 76.53, compared to M3DT-Random scored 77.89, which confirms our intuition that employing normal MoE performs better.

C. Agreement Vector

This section elucidates the utilization of the Agreement Vector, as proposed by Hu et al. (2024), as a metric for task grouping. For each task $\mathcal{T}_i$, the agreement score vector is defined as follows: $A(\mathcal{T}_i) = g_i \odot \frac{1}{N} \sum_{i=1}^{N} g_i$, where g_i denotes the gradient of the parameters calculated from task $\mathcal{T}_i$ and N denotes the total number of tasks. This vector reflects the gradient similarity between the task-specific and the average gradients, and further indicate the task similarity.

Table 6: Normalized score on MW+DMC

Task Scale	10 Tasks		80 Tasks		160 Tasks	
Method	Score	Parameters	Score	Parameters	Score	Parameters
MTDT-Small	70.97	1.47M	45.89	1.47M	35.34	1.47M
MTDT-Large	84.32	173.30M	59.01	173.30M	50.86	173.30M
PromptDT-Small	70.81	1.47M	42.09	1.47M	35.46	1.47M
PromptDT-Large	81.24	173.30M	59.40	173.30M	53.07	173.30M
HarmoDT-Small	65.13	1.47M	34.35	1.47M	28.05	1.47M
HarmoDT-Large	80.55	173.30M	59.91	173.30M	56.81	173.30M
M3DT-Random (Ours)	84.75	47.87M	66.93	98.37M	62.48	174.12M
M3DT-Gradient (Ours)	**84.77**	47.87M	**70.52**	98.37M	**65.90**	174.12M

Table 7: Normalized score on Ant+Cheetah

Task Scale	10 Tasks		80 Tasks		160 Tasks	
Method	Score	Parameters	Score	Parameters	Score	Parameters
MTDT-Small	94.09	1.47M	86.82	1.47M	83.04	1.47M
MTDT-Large	93.89	173.30M	89.30	173.30M	90.43	173.30M
PromptDT-Small	**94.47**	1.47M	88.15	1.47M	87.37	1.47M
PromptDT-Large	93.61	173.30M	90.66	173.30M	90.11	173.30M
HarmoDT-Small	93.83	1.47M	86.31	1.47M	86.54	1.47M
HarmoDT-Large	92.61	173.30M	**90.75**	173.30M	90.19	173.30M
M3DT-Random (Ours)	93.34	47.87M	90.55	98.37M	**90.97**	174.12M
M3DT-Gradient (Ours)	93.52	47.87M	90.41	98.37M	90.53	174.12M