Decomposed Prompt Decision Transformer for Efficient Unseen Task Generalization

Hongling Zheng[1] **Li Shen**[2†] **Yong Luo**[1†] **Tongliang Liu**[3] **Jialie Shen**[4] **Dacheng Tao**[5]
[1]Wuhan University [2]Shenzhen Campus of Sun Yat-sen University [3]The University of Sydney
[4]City, University of London [5]Nanyang Technological University
{hlzheng, luoyong}@whu.edu.cn {mathshenli, jialie, dacheng.tao}@gmail.com
tongliang.liu@sydney.edu.au

Abstract

Multi-task offline reinforcement learning aims to develop a unified policy for diverse tasks without requiring real-time interaction with the environment. Recent work explores sequence modeling, leveraging the scalability of the transformer architecture as a foundation for multi-task learning. Given the variations in task content and complexity, formulating policies becomes a challenging endeavor, requiring careful parameter sharing and adept management of conflicting gradients to extract rich cross-task knowledge from multiple tasks and transfer it to unseen tasks. In this paper, we propose the Decomposed Prompt Decision Transformer (DPDT) that adopts a two-stage paradigm to efficiently learn prompts for unseen tasks in a parameter-efficient manner. We incorporate parameters from pre-trained language models (PLMs) to initialize DPDT, thereby providing rich prior knowledge encoded in language models. During the **decomposed prompt tuning phase**, we learn both cross-task and task-specific prompts on training tasks to achieve prompt decomposition. In the **test time adaptation phase**, the cross-task prompt, serving as a good initialization, were further optimized on unseen tasks through test time adaptation, enhancing the model's performance on these tasks. Empirical evaluation on a series of Meta-RL benchmarks demonstrates the superiority of our approach. The project is available at `https://github.com/ruthless-man/DPDT`.

1 Introduction

The purpose of offline reinforcement learning (Offline RL) [1] is to develop a reward-maximizing RL strategy using offline data. This approach is of highly valuable in real-world scenarios where online data collection is expensive, time-consuming, or impractical. Existing offline RL algorithms typically perform well in single task but often struggle for multiple tasks with similar conditions and objectives, as they lack the ability to separate common knowledge from conflicting task gradients. In contrast, humans can leverage knowledge from existing tasks to excel in new ones, which has led to increasing research interest in multi-task reinforcement learning (MTRL) [2, 3, 4]. The goal of MTRL is to develop a universal strategy applicable to tasks with certain similarities, thereby enhancing adaptability and performance in multi-task environments.

Decision transformer (DT) [5] and Prompt-DT [6] introduce the transformer architecture to the field of RL, demonstrating the powerful data modeling capability of sequence offline RL. Additionally, they provide possibilities for integrating advancements [7] from language modeling into MTRL methodologies. Prompt-tuning DT [8] uses a gradient-free method to introduce prompts, retaining context-specific information and catering to specific preferences. These existing sequence-based offline RL methods are primarily trained and tested on the same task or perform fine-tuning using a

†Corresponding authors.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

small portion of labeled test data [9]. As a result, these algorithms often perform poorly when faced with testing tasks that are unseen and unlabeled in a Meta-RL setting [10].

Leveraging MTRL to extract general knowledge offers a promising approach for facilitating cross-task knowledge transfer in Meta-RL scenarios. However, as the number of tasks increases, gradient conflicts become more pronounced, hindering MTRL performance due to unregulated parameter sharing. Additionally, while transformer architectures can capture extensive relationships in offline sequential data, their data-hungry nature means that insufficient training data for RL tasks significantly diminishes model performance. A feasible solution might be to enhance the model's prior knowledge.

To remedy these drawbacks, we propose a prompt-based MTRL method named Decomposed Prompt Decision Transformer (DPDT), inspired by some works in natural language processing where knowledge transfer in multi-task learning scenarios is achieved through prompting strategies [11, 12]. We first employ pre-trained parameters from GPT to initialize a DPDT architecture. Incorporating the parameters of PLMs is motivated by several studies in RL [13, 14, 15]. Leveraging the rich prior knowledge encoded in Pre-trained Language Models (PLMs) effectively addresses the data hunger challenge of transformer architectures, providing ample semantic information for reinforcement learning tasks. Then, we design a two-stage training and testing framework for DPDT. (1) **Decomposed prompt tuning phase:** At this stage, we use prompt decomposition to avoid gradient conflicts between different tasks and to extract common knowledge. Specifically, we decompose the task prompt for each task into a cross-task prompt and a task-specific prompt. The cross-task prompt remains consistent across all training tasks, while the task-specific prompt is tailored to each task's unique characteristics. By isolating the cross-task prompt from the task-specific prompts, the model ensures that updates related to general knowledge do not conflict with those related to specific tasks. Compared to [12], our structured decomposition enables more regulated and harmonious parameter updates, thereby enhancing parameter efficiency and facilitating the extraction of general knowledge more effectively. (2) **Test time adaptation phase:** The cross-task prompt, serving as a strong initialization, is further optimized on unlabeled unseen tasks by incorporating the Test Time Adaptation (TTA) [16] to our model. TTA dynamically adjusts the cross-task prompts during the testing phase based on task characteristics, enhancing the model's adaptability to unseen tasks features.

Our DPDT has been empirically validated in the Meta-RL setting, and the results demonstrate its superiority compared with many recent and competitive counterparts [5, 6, 17]. Furthermore, we conducted ablation experiments covering aspects such as prompt length, scalability, and model variants to establish the superiority of the model. The main contributions of this work are as follows:

- We reconsider the problem of knowledge extraction in MTRL and propose the method of Decomposed Prompt Decision Transformer.
- We propose a two-stage paradigm, which includes decomposing the task prompts for training tasks into cross-task prompts and task-specific prompts, and aligning the cross-task prompts in unlabeled unseen tasks.
- We demonstrate the effectiveness of DPDT through intensive experiments on a broad spectrum of benchmarks, highlighting its competitive performance in Meta-RL scenarios.

2 Related works

In this section, we summarize the most related works as two-fold: offline RL and multitask RL.

Offline RL. In contrast to traditional RL methods [18, 19], offline RL focuses on training models and performing trial-and-error using offline data without environmental interaction to arrive at appropriate strategies. These methods primarily address the issue of out-of-distribution (OOD) through strategies such as constraining the learning policies [20] or bounding the overestimated policy values [21]. The integration of transformer architecture in sequence modeling has emerged as a prominent approach for addressing offline RL tasks [22, 23], further demonstrating the advantages of data-driven policy learning. Decision Transformer (DT) [5] involves encapsulating rewards, states, and actions into triples and training them using autoregressive supervision on offline data. Owing to the transformer's proficiency in capturing and fitting long time series features, it has achieved remarkable results in various offline RL tasks. HDT [17] generalizes new tasks by designing an adaptation module initialized by a hypernetwork. To alleviate the tuning burden while preserving performance, prompt tuning [24, 25] in NLP focuses on optimizing only the input parameters while keeping the majority

of the PLMs parameters frozen. However, integrating prompt tuning into RL field poses a challenge, as RL prompts lack semantic information and are challenging to optimize. While Prompt-DT [6] selects pre-defined expert trajectories combined with inputs to guide model training, it primarily relies on the quality of these trajectories for improvement, rather than enhancing prompts directly. On the other hand, Prompt-tuning DT [8] stands out for introducing prompt-tuning techniques using a gradient-free approach, with the goal of preserving environment-specific details and accommodating specific preferences.

Multitask RL. Multi-Task Reinforcement Learning (MTRL) [26, 27, 28] aims to address multiple similar reinforcement learning tasks using a unified model. A straightforward approach involves developing a task-conditional multi-task model, akin to those utilized in goal-conditional RL [29] and visual-language grounding [30]. While this method has demonstrated success in certain scenarios, it often encounters challenges stemming from negative interference among tasks. PaCo [31] delves into a compositional structure within the parameter space, distinguishing between task-agnostic and task-specific components. This approach significantly enhances the efficiency and robustness of the MTRL process, leading to more effective training outcomes. Building upon the MTRL paradigm, multi-task prompt [32] aims to acquire transferable, cross-task prompts from multiple tasks, guiding outputs for unseen downstream tasks. Several studies have approached multi-task prompt design from the perspective of prompt decomposition, yielding notable results across various tasks [33, 34].

The most relevant work to ours is Prompt-DT [6], which utilizes carefully selected prompts for Meta-RL tasks training. Our approach differs in (1) employing trainable prompt decomposition to avoid gradient conflicts among multiple tasks, (2) further optimizing general prompts with TTA without using any test data labels, and (3) extending the model architecture by incorporating PLMs for initialization. To the best of our knowledge, we are the first to implement multi-task prompt tuning based on PLMs parameters in the reinforcement learning domain.

3 Preliminary

In this section, we provide several concepts and terminologies that will be used in this work.

3.1 Prompt Decision Transformer

Prompt-DT [6] examines the beneficial effects of incorporating trajectory prompts on the DT [5] in few-shot scenarios. Specifically, the form of trajectory prompts is the same as that of training trajectories, consisting of triplets made up of state s^*, action a^*, and return-to-go $\hat{r}^*$. However, these trajectory prompts are significantly shorter than the training trajectories. During training, for each task i, the trajectory prompt τ^*_{i,K^*} is concatenated with the corresponding training trajectory $\tau_{i,K}$ to form $\tau_i^{input} = \left(\tau^*_{i,K^*}, \tau_{i,K}\right)$, which is then inputted into the model f for training. The concrete form of τ^*_{i,K^*} and $\tau_{i,K}$ are as follows:

$$\tau^\star_{i,K^*} = (\hat{r}^\star_1, s^\star_1, a^\star_1, \ldots, \hat{r}^\star_{K^\star}, s^\star_{K^\star}, a^\star_{K^\star}), \quad \tau_{i,K} = (\hat{r}_1, s_1, a_1, \ldots, \hat{r}_K, s_K, a_K) \tag{1}$$

where K^* represents the number of environment steps stored in the prompt, and K is the nearest steps of the training trajectory. The prediction head associated with the state token s is designed to predict the corresponding action a. It is noteworthy that during training, the model does not predict the actions of the trajectory prompts. The loss function is formulated as follows: $a_{i,m}$ denotes the actual action at the m-th timestep of the i-th task, while $\tau_{i,m-1}$ encompasses all data up to and including the $(m-1)^{th}$ timestep in the training trajectory of the i^{th} task.

$$L_m = \mathbb{E}_{\tau_i^{\text{input}} \sim \mathcal{T}_i}\left[\frac{1}{K}\sum_{m=1}^{K}\left(a_{i,m} - f(\tau^\star_{i,K^\star}, \tau_{i,m-1})\right)^2\right] \tag{2}$$

3.2 Test-Time Adaptation

Test time adaptation (TTA) [16] aims to minimize the gap between training data and testing data distribution during the testing phase. Test-time prompt tuning (TPT) [35] leverages the extensive knowledge in transformer architecture to enhance its generalization capabilities in zero-shot scenarios.

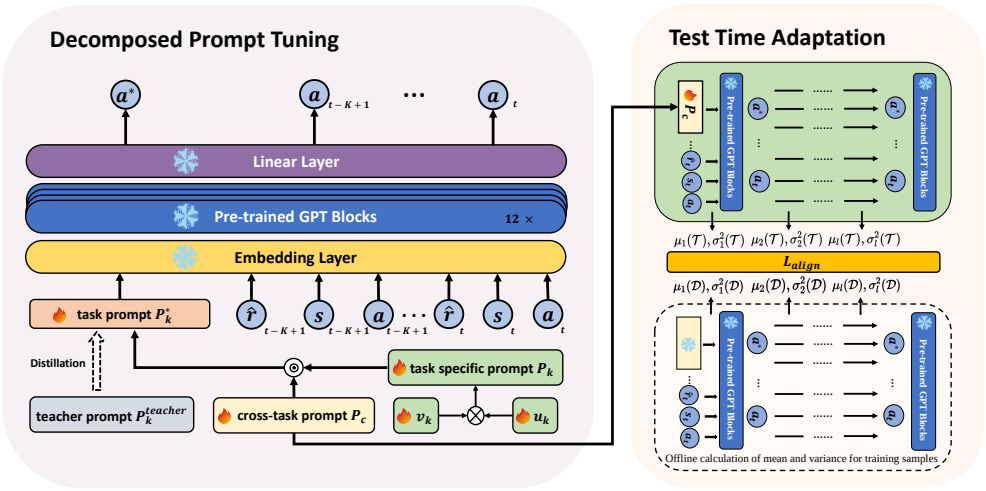

Figure 1: The architecture diagram of DPDT. For simplicity and clarity, the figure only displays the decomposition and integration process during training on a single task. Snowflake icons represent the frozen parts of the model that are not subject to training updates, while flame icons indicate components of the model that remain trainable. **Left: Decomposed Prompt Tuning.** The prompt decomposition is trained on the entire dataset with the assistance of the teacher prompt $p_k^{teacher}$. p_k^* is then combined with the training samples that include K steps, inputted into DPDT, and outputs the corresponding action a. **Right: Test Time Adaptation.** When test samples are fed into the model, we calculate the mean and variance of all samples at each layer and compute the loss by comparing them with the mean and variance of the corresponding layer from the training samples. The losses from all layers are summed to obtain the alignment loss, denoted as L_{align}.

During the inference phase, multiple randomly augmented views are created from the provided testing sample X_{test}. Predictions with entropy below a specified threshold are retained, while other views are filtered out using a confidence selection criterion. The average entropy of the filtered predictions is then used to unsupervisedly update the prompts p. Some methods [36, 37, 38], considering the data instability caused by augmentation techniques, use alignment of feature values in attention layers or feedforward layers to achieve test-time adaptation. Our method falls into this category.

4 Decomposed Prompt Decision Transformer

In this section, we provide a comprehensive description of the proposed DPDT. Task prompts and training data are combined to calculate the loss through action prediction, which is then used to optimize the cross-task and task-specific prompts in a backward pass. Once optimized, the cross-task prompts serve as a good initialization for use in unseen tasks during test-time adaptation. The objective of decomposed prompt tuning is to learn cross-task prompts and task-specific prompts through prompt decomposition. Test-time adaptation provides an alignment approach for the cross-task prompts. Below, we will describe each module of DPDT in detail.

4.1 Decomposed Prompt Tuning

Initialization. Given that Transformers are data-intensive and require pre-training on substantial datasets to achieve satisfactory performance, integrating PLMs from the same architectural family into offline RL is a natural progression. Some existing work has already explored this avenue [13, 39, 40]. Taking inspiration from this, we employ GPT2-SMALL [41] to initialize our DPDT and maintain these parameters frozen throughout training. It is worth noting that, incorporating PLMs into RL is not predicated on the direct applicability of language data to RL tasks. Instead, the advantage lies in leveraging the deep, nuanced representations acquired by PLMs from a variety of datasets. These representations encode a broad spectrum of patterns, relationships, and contexts that can transcend purely linguistic tasks. This taps into the reasoning and few-shot capabilities of language models, addressing challenging scenarios like data scarcity and sparse rewards.

Prompt Decomposition. Given a set of training tasks $S = \{S_1, S_2, \ldots, S_n\}$, our objective is to learn a general prompt P_c that encapsulates common knowledge shared across all tasks in S and can efficiently adapt to unseen tasks. Extracting general task information from tasks with different distributions is often challenging, as gradient conflicts between tasks can lead to suboptimal convergence of information. We adopt prompt decomposition approach to address this issue. As shown in Figure 1, let $P_c \in \mathbb{R}^{l \times s}$ and $P_k \in \mathbb{R}^{l \times s}$. The task prompt P_k^* for the k-th task is obtained by taking the element-wise product of P_c and P_k. The goal of prompt decomposition is to enable efficient knowledge sharing across S, while still allowing each task to maintain its own parameters to encode task-specific knowledge. The P_c aims to acquire general knowledge from S, while the task-specific prompt P_k allows task k to retain its unique knowledge. The task prompts parameterization of the k-th training task is expressed as:

$$P_k^* = P_c \circ P_k = P_c \circ (v_k \otimes u_k). \tag{3}$$

In the specific implementation process, inspired by LORA [42], we further decompose each task-specific prompt into two low-rank vectors $v_k \in \mathbb{R}^{l \times r}$ and $u_k \in \mathbb{R}^{l \times s}$ using a low-rank method. We obtain P_k through vector multiplication. Here l represents the prompt length, r represents the hidden layer dimension, and s represents the prompt dimension. The hyperparameter r is a manually specified low-rank parameter. Its introduction is crucial for designing prompts for all tasks in the dataset, significantly maintaining model superiority while reducing computational load. We use standard normal distribution to initialize P_c, u_k and v_k. Given the DPDT $\mathcal{M}$, the mean squared loss between the model's predicted actions and the true actions is calculated as:

Algorithm 1 Decomposed Prompt Tuning

Input: Training task set S, Offline datasets $\mathcal{D}_{\mathcal{M}}$, Batch size M, Learning rate α, training iterations N, teacher task prompts $p_k^{teacher}$.
Initialize: Initialize a 12-layer, 12-head DPDT $\mathcal{M}$ using GPT2-SMALL, randomly initialize cross-task prompts P_c and low-rank vectors v_k, u_k.

```
for t = 1 to N do
    for k in S do
        Select a trajectory τ that contains M samples in task k.
        Calculate P_k^* by Equation 3.
        Calculate L_MSE and L_dis according to Equations 4 and 5.
        Computed loss function by Equation 6.
        θ ← θ − α∇_θ L_Total.
    end for
end for
```

$$\mathcal{L}_{MSE} = \left(a - \mathcal{M}\left(P_k^\star, \tau\right)\right)^2 \tag{4}$$

Prompt distillation. Due to the lack of explicit constraints, directly implementing prompt decomposition on the multitask dataset S may lead to an overlap in the information learned by P_c and P_k, potentially undermining their ability to capture distinct intended details. We employed knowledge distillation techniques to compel the cross-task and task-specific prompts to learn their respective information. We obtained teacher task prompts $p_k^{teacher}$ for each task k by using traditional prompt-tuning methods individually. During training, the mean squared error is calculated directly between $p_k^{teacher}$ and $p_k^\star$:

$$\mathcal{L}_{dis} = \sum_{k \in |\mathcal{S}|} |p_k^{teacher} - p_k^\star|^2 \tag{5}$$

The total loss function for training task prompts for obtaining a cross-task prompt to be transferred to the target side is then:

$$\mathcal{L}_{Total} = \mathcal{L}_{MSE} + \lambda \mathcal{L}_{dis} \tag{6}$$

where λ is a weight to balance the impact of distillation loss terms. In our experiments, we set λ to 0.5. The overall summary of the multitask training algorithm is presented in Algorithm 1.

4.2 Test Time Adaptation

During the test time adaptation (TTA) phase, we address distribution bias by aligning the distribution of unlabeled test samples with the training samples. For each test task t in the test task set T, we randomly select a subset X of unlabeled test samples, combine them with the cross-task prompts P_c and input them into the model.

Here, we introduce the data collection method for X. The model's testing phase usually occurs in a simulated environment where we predefine our expected reward values $\hat{r}$. The environment

provides the initial state s of the environment, consistent with the settings during inference in prompt DT methods. However, unlike in training tasks where ground-truth labels exist, for action a_1, we assign a value sampled randomly from the action space (which is typically consistent between training and testing tasks). We feed this sequence of Markov chains into the environment, obtaining rewards and the next environment states iteratively, assigning a randomly sampled value to action a_2 in subsequent iterations. This process is repeated $|X|$ times, resulting in data of the form $(\hat{r}0, s_0, a_0, \hat{r}1, s_1, a_1, \ldots, \hat{r}_{|N|}, s_{|N|}, a_{|N|})$.

In each layer of the model, we calculate the alignment loss based on the means and variances of the training and test samples. Our goal is to update the P_c for the given test task through this alignment loss. For each test task t, we denote the distribution of the test samples as $\mathcal{T}$ and the distribution of the training samples as $\mathcal{D}$. Specifically, we calculate the aligned token mean and variance via:

$$\mu_l(\mathcal{T}) = \frac{1}{|X|}\sum_{i=1}^{|X|} H_{l,i}, \quad \sigma_l^2(\mathcal{T}) = \frac{1}{|X|}\sum_{i=1}^{|X|} \left[H_{l,i} - \mu_l(\mathcal{T})\right]^2 \tag{7}$$

Here, $H_{l,i}$ represents the state of the i^{th} sample at the l^{th} hidden layer, while $\mu_l(\mathcal{T})$ and $\sigma_l^2(\mathcal{T})$ denote the mean and variance of all test samples at the l^{th} hidden layer, respectively. Similarly, for each hidden layer of the model, we pre-calculate the statistical measures of mean $\mu_l(\mathcal{D})$ and variance $\sigma_l^2(\mathcal{D})$ of the training samples, which are uniformly sampled across all tasks in the training set, in an offline setting to reduce parallel computing costs, since both training samples and labels are accessible. The formula for calculating the alignment loss function is as follows:

Algorithm 2 Test Time Adaptation

Input: Test samples set X, Cross-task prompts P_c, $\mu_l(\mathcal{D})$, $\sigma_l^2(\mathcal{D})$, The number of layers L.

for $l = 1$ to L **do**
  for i in X **do**
    Calculate $H_{l,i}$ obtained by inputting the concatenation of P_c and i into DPDT.
  end for
end for
for $l = 1$ to L **do**
  Compute $\mu_l(\mathcal{T})$ and $\sigma_l^2(\mathcal{T})$ by Equation 7.
end for
Compute token distribution alignment loss by Equation 8.
Optimize L_{align} to update P_c.

$$L_{\text{align}} = \frac{1}{L}\sum_{l=1}^{L}\left(\|\mu_l(\mathcal{T}) - \mu_l(\mathcal{D})\|_1 + \|\sigma_l^2(\mathcal{T}) - \sigma_l^2(\mathcal{D})\|_1\right). \tag{8}$$

The test time adaptation phase process is illustrated in Algorithm 2.

5 Experiment

In this section, we present an extensive evaluation of our proposed DPDT using widely recognized benchmarks. Additionally, we conduct empirical ablation studies to dissect and understand the individual contributions of the core components of our methodology.

5.1 Environments and Baselines

Environments. To ensure a fair comparison with existing multi-task offline reinforcement learning algorithms, we conducted verification of DPDT using the MuJoCo [43] and MetaWorld [30] benchmarks, which serve as standard tasks in the domain of sequence offline RL, offering sufficient diversity and representing common challenges in classical RL, such as sparse rewards, complex state spaces, and precise control of robotic systems. Our experiments on the Cheetah-dir, Cheetah-vel, and Ant-dir environments in the MuJoCo benchmark meticulously adhere to the datasets and methodologies outlined in Prompt-DT. These tasks penalize agents for using excessive control signals. In the MetaWorld benchmark, we used the ML10, ML45, MT10 and MT50 environments for Meta-RL. A detailed description of the datasets and the division of training and test tasks is provided in Appendix A.

Baselines. We compared DPDT to the following offline RL baselines: (1) **Multi-task Behaviour Cloning (MT-BC) [44]**: MT-BC optimizes multi-task learning by exclusively simulating trajectories from the original dataset, dispensing with the need for prompts and reward-to-go tokens. This approach emphasizes the utilization of intrinsic task-specific information, adopting a behavior cloning

Table 1: Results for Meta-RL control tasks (zero-shot scenarios). The best mean accumulated returns are highlighted in bold. For each prompt-needing environment, prompts of length K=30 are utilized. Each experiment was run three times to ensure stability and reproducibility of the results. We report the average returns and standard deviations for these three runs (the higher, the better).

	MT-BC [44]	**MT-ORL [5]**	**Soft-Prompt [45]**	**HDT [17]**	**Prompt-DT [6]**	**DPDT-WP**	**DPDT**
Trainable Params	125.5M	125.5M	3.94M	12.94M	125.5M	1.42M	1.42M
Percentage	100%	100%	3%	10.31%	100%	1.14%	1.14%
Cheetah-dir	-24.71$_{\pm 12.04}$	-86.92$_{\pm 15.51}$	-4.21$_{\pm 5.51}$	-45.32$_{\pm 13.22}$	-7.92$_{\pm 2.97}$	11.73$_{\pm 12.8}$	**50.32$_{\pm 11.47}$**
Cheetah-vel	-201.66$_{\pm 30.27}$	-148.24$_{\pm 22.18}$	-171.23$_{\pm 20.58}$	-162.75$_{\pm 20.50}$	-192.38$_{\pm 11.80}$	-143.14$_{\pm 21.40}$	**-139.88$_{\pm 19.65}$**
Ant-dir	131.89$_{\pm 12.96}$	109.21$_{\pm 9.66}$	119.45$_{\pm 14.2}$	115.43$_{\pm 10.22}$	**123.46$_{\pm 10.70}$**	101.49$_{\pm 17.74}$	121.84$_{\pm 8.01}$
MW ML10	256.77$_{\pm 11.93}$	343.16$_{\pm 9.40}$	246.42$_{\pm 24.60}$	292.14$_{\pm 8.21}$	317.31$_{\pm 14.98}$	204.88$_{\pm 28.96}$	**371.01$_{\pm 9.41}$**
MW ML45	287.37$_{\pm 11.38}$	266.744$_{\pm 25.81}$	91.97$_{\pm 14.11}$	274.88$_{\pm 19.74}$	294.55$_{\pm 8.71}$	300.71$_{\pm 15.74}$	**347.21$_{\pm 11.52}$**
MW MT 10	547.83$_{\pm 11.04}$	1064.58$_{\pm 21.70}$	201.23$_{\pm 7.11}$	964.57$_{\pm 15.34}$	1087.54$_{\pm 17.09}$	1015.91$_{\pm 0.74}$	**1317.52$_{\pm 8.22}$**
MW MT 50	582.80$_{\pm 13.48}$	929.74$_{\pm 22.81}$	400.71$_{\pm 26.40}$	820.45$_{\pm 27.19}$	994.63$_{\pm 5.99}$	1131.01$_{\pm 1.17}$	**1559.94$_{\pm 2.49}$**
Average	225.76	354.04	130.62	309.79	373.88	374.66	**518.28**

strategy to streamline the learning process. (2) **Multi-Task Decision Transformer (MT-ORL) [5]**: We train a decision transformer to learn multiple tasks from the training set. To construct the MT-DT, we exclude prompt augmentation present in DPDT, while retaining the rest of the training process identical to that of DPDT. (3) **Soft-Prompt [45]**: Soft-prompt is trained using a universal prompt across all tasks. (4) **Hyper-decision transformer (HDT) [17]**: HDT efficiently adapts DT to new tasks by augmenting them with an adaptation module, whose parameters are initialized by a hyper-network, enabling quick and efficient adaptation with minimal data. (5) **Prompt-DT [6]**: Prompt-DT builds on DT, leveraging trajectory prompts and reward-to-go for multi-task learning and generalization to unseen tasks.

Implementation details. All experiments were carried out on a server with 8 NVIDIA 3090 GPUs, each with 24GB of memory, using PyTorch [46] and Hugging Face Transformers libraries [47]. The experimental hyperparameter configurations are shown in Appendix B. The computer resources utilized by all methods are shown in Table 12.

5.2 Main Results and Analysis

Zero-shot Generalization. In Table 1, we compare the zero-shot generalization ability of DPDT and the baselines to investigate the overall performance of DPDT. For evaluation, we use the average episode cumulative returns in the test task set as the evaluation metric. Additionally, we introduce a variant of DPDT that does not use GPT-SMALL parameters for initialization, referred to as DPDT-WP (DPDT-Without Pretrained). Soft prompts adapt to tasks in a parameter-efficient way. However, training a universal prompt across multiple tasks suffers from significant gradient interference, as demonstrated by the experimental results. The prompt-DT performs well in few-shot scenarios due to its utilization of test data for fine-tuning. However, in downstream tasks where data is scarce, the prompt fails to provide sufficient task-specific guidance to the model, resulting in suboptimal performance. Importantly, our proposed DPDT exhibits significant performance improvements over fine-tuning and prompt-based methods. This vividly demonstrates the distinct advantages offered by our innovative multitask training techniques. It is worth noting that without initialization with PLM, the performance of DPDT-WP is inferior to most methods. We believe this is mainly due to the model lacking sufficient prior knowledge for effective multi-task prompt tuning, resulting in suboptimal performance of DPDT. In Figure 2, we illustrate the accumulated returns curves of DPDT and other baselines across the Cheetah-vel, MW ML45, and MW MT50 environments. Additional curves for other environments can be found in Figure 4. We also conducted experiments on Soft-Prompt and Prompt-DT combined with TTA (shown in Figure 11). Soft-Prompt-TTA showed performance improvements across all tasks, whereas Prompt-DT-TTA experienced performance declines in some tasks. The main reason for this is that Prompt-DT relies on high-quality trajectory data for prompts during testing, and applying TTA on unlabeled data may have adversely affected prompt optimization.

Few-shot Generalization. We explored the performance of DPDT in few-shot scenarios and further investigated whether the prompt decomposition mechanism successfully isolated general knowledge. In this scenario, DPDT does not use TTA to align cross-task prompts P_c. Instead, P_c is fine-tuned

Table 2: Results for Meta-RL control tasks (few-shot scenarios). The best mean accumulated returns are highlighted in bold. For each prompt-needing environment, prompts of length K=30 are utilized. Each experiment was run three times to ensure stability and reproducibility of the results. We report the average returns and standard deviations for these three runs (the higher, the better).

	MT-ORL [5]	**Soft-Prompt [45]**	**HDT [17]**	**Prompt-DT [6]**	**DPDT-WP**	**DPDT**	**DPDT-F**
Trainable Params	125.5M	3.94 M	12.94 M	125.5M	1.42M	1.42M	125.5M
Percentage	100%	3%	10.31%	100%	1.14%	1.14%	100%
Cheetah-dir	-46.22$_{\pm 3.44}$	940.24$_{\pm 1.08}$	875.23$_{\pm 4.24}$	934.78$_{\pm 5.33}$	946.81$_{\pm 17.24}$	**955.17$_{\pm 8.03}$**	1037.85$_{\pm 5.98}$
Cheetah-vel	-146.64$_{\pm 2.12}$	-41.81$_{\pm 2.10}$	-63.81$_{\pm 6.30}$	-37.80$_{\pm 2.09}$	-48.07$_{\pm 1.85}$	**-30.73$_{\pm 1.88}$**	-29.85$_{\pm 9.46}$
Ant-dir	110.51$_{\pm 2.2}$	379.01$_{\pm 1.75}$	361.49$_{\pm 5.63}$	**411.96$_{\pm 9.28}$**	308.10$_{\pm 5.22}$	384.29 $_{\pm 10.91}$	400.01$_{\pm 9.79}$
MW ML10	421.22$_{\pm 9.21}$	379.82 $_{\pm 14.76}$	467.81$_{\pm 3.07}$	315.07$_{\pm 6.17}$	485.27$_{\pm 19.31}$	**535.52$_{\pm 17.39}$**	670.24$_{\pm 3.88}$
MW ML45	264.14$_{\pm 9.67}$	448.72$_{\pm 11.38}$	477.19$_{\pm 2.16}$	473.34 $_{\pm 4.12}$	519.28 $_{\pm 7.22}$	**579.09$_{\pm 10.42}$**	600.44$_{\pm 17.48}$
Average	120.60	421.204	423.56	419.47	442.27	**484.66**	535.74

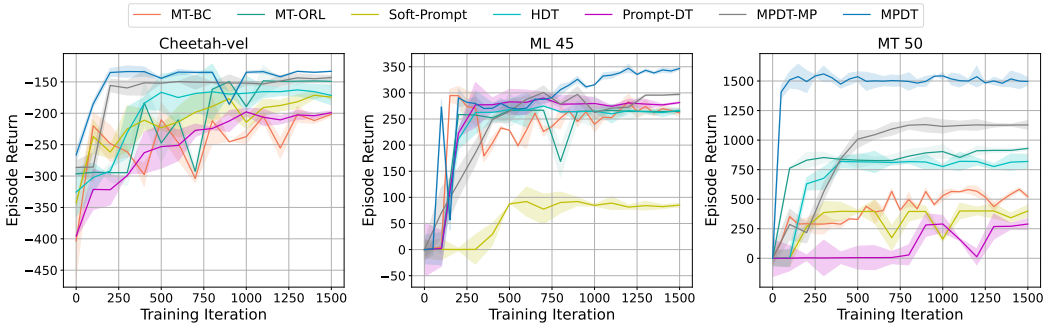

Figure 2: Episodic accumulated returns in three tasks of MTBC, MT-ORL, Soft-Prompt, HDT, Prompt-DT, DPDT-WP and DPDT. Each method is restricted to 1500 rounds of runs in each environment.

directly on a small number of labeled test samples through a self-supervised paradigm. Specifically, we randomly selected only one trajectory from the test dataset for fine-tuning. Other methods adhered to the same amount of fine-tuning data. Furthermore, in few-shot scenarios, we fully fine-tuned DPDT on the complete training and testing data, denoted as DPDT-F. The performance of the DPDT-F method represents the upper bound of all model performances in the current environment. Table 2 shows that the DPDT method, even after fine-tuning, still significantly outperforms or matches the baseline algorithms, demonstrating the effectiveness of prompt decomposition in few-shot environments. Moreover, in some datasets, DPDT approaches the performance of fully fine-tuned models on the test set using only 1.14% of the parameters, as observed in the cheetah-vel environment. It is worth noting that in the Ant-dir environment, the performance of DPDT is slightly inferior to that of Prompt-DT in both zero-shot and few-shot settings. We believe the primary reason for this is the significant domain difference between the language model and environment.

5.3 Further Analysis

In the ablation studies, we conducted research on the components of DPDT as well as the impact of prompt length and model parameters on convergence speed and performance.

Impact of model components. As shown in Table 3, the impact of prompt decomposition was evaluated. Compared to the soft-prompt method in the unseen task settings (first row), substituting it with decomposed prompts P_k and P_c without distillation (third row) resulted in performance improvements across all three tasks. This ablation highlights the significance of the prompt decomposition strategy in DPDT, demonstrating that the shared component adeptly captures the diverse cross-task knowledge essential for enhancing target downstream tasks.

To evaluate the impact of prompt distillation, we trained a standard prompt shared across all training tasks using the same training loss as DPDT. The teacher prompts for each task remained consistent in DPDT. Compared to the basic baseline (first row), incorporating prompt distillation (second row)

Table 3: Ablation: The impact of prompt decomposition, prompt distillation and test time adaptation.

Decomposition	Distillation	TTA	Cheetah-vel	MW ML45	MW MT50
✗	✗	✗	-171.23	91.97	400.71
✗	✔	✔	-163.05	108.01	709.81
✔	✗	✔	-160.10	273.99	1137.39
✔	✔	✗	-167.80	149.21	824.07
✔	✔	✔	**-139.88**	**347.21**	**1559.94**

Table 4: Ablation: The impact of model size. The elements of the triplet represent, in order, the number of transformer blocks, the count of attention heads, and the size of the hidden layers.

Model size	Cheetah-vel	Ant-dir	MW ML45	MW MT50
(3,1,128)	-164.88	129.34	288.14	749.18
(12,12,768)	**-139.88**	121.84	**347.21**	**1559.94**
(24,16,768)	-210.35	**165.99**	292.48	1527.34

resulted in a modest improvement in average performance across the three tasks. This emphasizes that prompt distillation from separately trained source prompts is an effective strategy for acquiring high-quality decomposable prompts.

To compare the impact of using TTA on model performance, we examined the results in the fourth and fifth rows. We found that the use of TTA affects the model's final performance. The reason is intuitive: cross-task prompts provide a good initialization environment for TTA, but relying solely on the general information from cross-task prompts is insufficient for the model to perform well on unseen tasks. Incorporating TTA allows the model to adapt to the specific nuances of each task during testing, resulting in substantial performance gains.

Impact of prompt length. We examined the influence of prompt length on the performance of DPDT by investigating five distinct prompt lengths (3, 6, 30, 60, 90). Specifically, we explored the effect of prompt length variation on the convergence behavior and generalization capability of the model. It is widely recognized that prompt lengths that are excessively short may impede model convergence, while prompt lengths that are overly long can result in slow convergence rates and potential overfitting. Ablation experiments revealed that a prompt length of 30 is optimal. Further increasing the prompt length to 60 or 90, however, results in minor performance fluctuations but increases the convergence time. Therefore, we used a prompt length of 30 for all our experiments.

Impact of model size. We explored the performance of DPDT under three model size configurations. The (3,1,128) configuration uses the pretrained model provided in the original Prompt-DT [6] to initialize DPDT, while the (24,16,768) configuration employs GPT-MIDDLE. Table 4 shows that the size of the pretrained model parameters is correlated with the performance improvement of DPDT. When the model size is expanded to a certain extent (12,12,768), efficient parameter fine-tuning can extract adequate prior knowledge for downstream tasks. However, as the model complexity increases, such as when reaching the size of GPT-MIDDLE, the model exhibits performance improvement on some tasks (Ant-dir) but a decrease in performance on others. This phenomenon can be attributed to the significant gap between the size of the dataset and the complexity of the model. Fine-tuning the model in such scenarios may encounter challenges in appropriately converging for reinforcement learning tasks, potentially leading to overfitting.

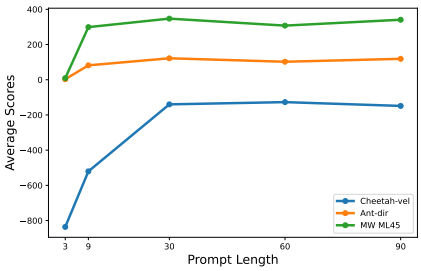

Figure 3: Ablation: The effect of prompt length on DPDT's zero-shot generalization ability.

Impact of data quality. The quality of data used for fine-tuning cross prompts does indeed affect the final model performance. We conducted experiments focusing on data quality. In the Cheetah-vel and ML45 environments, we differentiated the quality of datasets into expert, medium, random, and mixed datasets. Each dataset consists of 200 time steps, which aligns with the setup for few-shot scenarios relative to the size of the training set. As shown in Table 5, we found that models fine-tuned

using expert datasets perform the best, which aligns with our intuition. Additionally, the performance of models fine-tuned on mixed datasets is close to that on expert datasets, suggesting implicitly that the DPDT method can extract information from suboptimal datasets to ensure model performance.

Impact of adaptation method. In addition to only utilizing cross-task prompts P_c for TTA in zero-shot scenarios, we also investigated (1) combining cross-task prompts P_c with the average of all task-specific prompts P_k from the training set for TTA, (2) freezing the cross-task prompts P_c, we initialized a new task-specific prompt combined with the cross-task prompts for TTA and (3) randomly selecting one P_k from a training task and combining it with P_c for TTA. However, we found that all of these initialization methods resulted in suboptimal outcomes, shown in Table 10.

Table 5: Ablation: The impact of data quality.

	Cheetah-vel	ML45
expert datasets	-30.10	586.84
medium datasets	-41.73	502.64
random datasets	-935.66	37.91
mixed datasets	-30.73	579.09

Impact of learning rate in prompt decomposition. As shown in Table 6, we present experimental results on the ML45 dataset where different learning rates were applied to P_c and P_k in prompt decomposition. We observed optimal performance when both had the same learning rate. We speculate that this occurs because, over training iterations, both prompts converge to their optimal values, and differing learning rates disrupt their joint convergence, leading to poorer performance under similar runtime conditions.

Table 6: Ablation: The impact of learning rate in prompt decomposition.

	lr_{P_c}=1e-2	lr_{P_c}=1e-3	lr_{P_c}=1e-4
lr_{P_k}=1e-2	310.74	307.36	311.40
lr_{P_k}=1e-3	198.17	350.99	338.21
lr_{P_k}=1e-4	204.94	104.07	347.21

Impact of low-rank parameter r. Table 7 presents the results of our ablation experiments focusing on the low-rank parameter r of prompt decomposition for Cheetah-vel and MW ML45. DPDT is relatively insensitive to selecting hyperparameters, a potential advantage of our work. As observed, the model's performance varies with different values of r. These findings suggest that the performance of DPDT remains stable across different values of r. This characteristic can be advantageous, as it allows users to implement the model without extensive tuning, streamlining the deployment process while still achieving competitive results across diverse tasks.

Table 7: Ablation: The impact of low-rank parameter r.

	Cheetah-vel	MW ML45
r=1	-139.88	347.21
r=4	-138.08	344.10
r=10	-135.51	350.33

6 Conclusion, Limitation and Broader Impact

We have introduced a novel approach, the Decomposed Prompt Decision Transformer (DPDT), aimed at efficient generalization to unseen tasks. Through the utilization of PLMs for parameter initialization and the implementation of parameter-efficient multi-task prompt tuning techniques, we have successfully extracted cross-task general knowledge and further fine-tuned it on previously unseen tasks. Our experiments across various Meta-RL environments demonstrated the effectiveness of our components, achieving superior performance with significantly fewer task-specific parameters compared to fully fine-tuned methods. This approach offers a robust framework for future research to further explore and optimize multi-task learning and generalization capabilities.

Limitation. Currently, our work primarily involves using large language models to initialize DPDT. While we've utilized parameter-efficient techniques to fine-tune the model and mitigate inter-domain variances, focusing on optimizing these differences could potentially enhance model performance.

Broader Impact. Overall, the application of PEFT methods based on PLMs can help users obtain high-quality reinforcement learning decision models at minimal cost. However, this approach may lead to the misuse of language models when users are unaware of the inter-domain differences, potentially resulting in unforeseen negative outcomes in the RL decision-making process.

Acknowledgements

This work is supported by the STI 2030-Major Projects (No. 2021ZD0201405), the National Natural Science Foundation of China (Grant No. U23A20318 and 62276195), the Fundamental

Research Funds for the Central Universities (No. 2042024kf0039), the Science and Technology Major Project of Hubei Province under Grant 2024BAB046, and the Innovative Research Group Project of Hubei Province under Grant 2024AFA017. Tongliang Liu is partially supported by the following Australian Research Council projects: FT220100318, DP220102121, LP220100527, LP220200949, IC190100031. Dr. Tao's research is partially supported by NTU RSR and Start Up Grants. The numerical calculations in this paper have been done on the supercomputing system in the Supercomputing Center of Wuhan University.

References

[1] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[2] Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Efficient multi-task and transfer reinforcement learning with parameter-compositional framework. *IEEE Robotics and Automation Letters*, 2023.

[3] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36, 2024.

[4] Shengchao Hu, Ziqing Fan, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Harmodt: Harmony multi-task decision transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.18080*, 2024.

[5] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[6] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR, 2022.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Prompt-tuning decision transformer with preference ranking. *arXiv preprint arXiv:2305.09648*, 2023.

[9] Shengchao Hu, Li Shen, Ya Zhang, Yixin Chen, and Dacheng Tao. On transforming reinforcement learning with transformers: The development trajectory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[10] Suyoung Lee and Sae-Young Chung. Improving generalization in meta-rl with imaginary tasks from latent dynamics mixture. *Advances in Neural Information Processing Systems*, 34:27222–27235, 2021.

[11] Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attentional mixtures of soft prompt tuning for parameter-efficient multi-task knowledge sharing. *arXiv preprint arXiv:2205.11961*, 3, 2022.

[12] Tu Thanh Vu, Daniel Matthew Cer, Noah Constant, Brian David Lester, and Rami Al-Rfou. Frozen model adaptation through soft prompt transfer, January 18 2024. US Patent App. 17/863,840.

[13] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[14] Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon S Du, and Huazhe Xu. Unleashing the power of pre-trained language models for offline reinforcement learning. *arXiv preprint arXiv:2310.20587*, 2023.

[15] Yao Wei, Yanchao Sun, Ruijie Zheng, Sai Vemprala, Rogerio Bonatti, Shuhang Chen, Ratnesh Madaan, Zhongjie Ba, Ashish Kapoor, and Shuang Ma. Is imitation all you need? generalized decision-making with dual-phase training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16221–16231, 2023.

[16] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022.

[17] Mengdi Xu, Yuchen Lu, Yikang Shen, Shun Zhang, Ding Zhao, and Chuang Gan. Hyper-decision transformer for efficient online policy adaptation. *arXiv preprint arXiv:2304.08487*, 2023.

[18] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[19] Hao-nan Wang, Ning Liu, Yi-yun Zhang, Da-wei Feng, Feng Huang, Dong-sheng Li, and Yi-ming Zhang. Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 21(12):1726–1744, 2020.

[20] Dibya Ghosh, Anurag Ajay, Pulkit Agrawal, and Sergey Levine. Offline rl policies should be trained to be adaptive. In *International Conference on Machine Learning*, pages 7513–7530. PMLR, 2022.

[21] Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8753–8760, 2022.

[22] Shengchao Hu, Ziqing Fan, Chaoqin Huang, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Q-value regularized transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.17098*, 2024.

[23] Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022.

[25] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. *arXiv preprint arXiv:2303.02861*, 2023.

[26] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

[27] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, pages 9767–9779. PMLR, 2021.

[28] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. *arXiv preprint arXiv:2405.08550*, 2024.

[29] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

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

[31] Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Paco: Parameter-compositional multi-task reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21495–21507, 2022.

[32] Roei Herzig, Ofir Abramovich, Elad Ben Avraham, Assaf Arbelle, Leonid Karlinsky, Ariel Shamir, Trevor Darrell, and Amir Globerson. Promptonomyvit: Multi-task prompt learning improves video transformers using synthetic scene data. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6803–6815, 2024.

[33] Tianxiang Sun, Zhengfu He, Qin Zhu, Xipeng Qiu, and Xuan-Jing Huang. Multitask pre-training of modular prompt for chinese few-shot learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11156–11172, 2023.

[34] Sheng Shen, Shijia Yang, Tianjun Zhang, Bohan Zhai, Joseph E Gonzalez, Kurt Keutzer, and Trevor Darrell. Multitask vision-language prompt tuning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5656–5667, 2024.

[35] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *Advances in Neural Information Processing Systems*, 35:14274–14289, 2022.

[36] Shuai Wang, Daoan Zhang, Zipei Yan, Jianguo Zhang, and Rui Li. Feature alignment and uniformity for test time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20050–20060, 2023.

[37] Sanghun Jung, Jungsoo Lee, Nanhee Kim, Amirreza Shaban, Byron Boots, and Jaegul Choo. Cafa: Class-aware feature alignment for test-time adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19060–19071, 2023.

[38] Jameel Abdul Samadh, Mohammad Hanan Gani, Noor Hussein, Muhammad Uzair Khattak, Muhammad Muzammal Naseer, Fahad Shahbaz Khan, and Salman H Khan. Align your prompts: Test-time prompting with distribution alignment for zero-shot generalization. *Advances in Neural Information Processing Systems*, 36, 2024.

[39] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[40] Yujin Tang, Wenhao Yu, Jie Tan, Heiga Zen, Aleksandra Faust, and Tatsuya Harada. Saytap: Language to quadrupedal locomotion. *arXiv preprint arXiv:2306.07580*, 2023.

[41] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[42] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[44] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[45] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[47] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Appendix

The appendix is organized into several sections, each providing additional insights and details related to different aspects of the main work.

A Detailed Environment

- **Cheetah-dir**: There are two tasks in Cheetah-dir with goal directions as forward and backward, respectively. The cheetah agent is rewarded with high velocity along the goal direction. The training and testing set are equal, and both contain the two tasks.
- **Cheetah-vel**: There are 40 tasks in Cheetah-vel with different goal velocities. The target velocities are uniformly sampled from the interval [0,3]. The agent is penalized with l2 errors to the target velocity. We hold out 5 tasks to construct the testing set and train with the remaining 35 tasks.
- **Ant-dir**: There are 50 tasks in Ant-dir with different goal directions uniformly sampled in 2D space. The 8-joints ant is rewarded with high velocity along the goal direction. We sample 5 tasks for testing and leave the rest for training.
- **Meta-World ML10**: In Meta-World ML10, the task is to control a Sawyer robot's end-effector to reach a target position in 3D space. The agent directly controls the XYZ location of the end-effector. Each task has a different goal position. We train in 10 tasks and test in unseen 3 tasks.
- **Meta-World ML45**: In Meta-World ML45, each task has a different goal position. We train in 45 tasks and test in unseen 5 tasks.
- **Meta-World MT10**: Meta-World MT10 comprises 10 distinct robot manipulation tasks with shared dynamics for training and 3 unseen robot manipulation tasks for testing. These tasks exhibit greater variability compared to standard meta-learning environments, posing greater challenges for extracting generalizable knowledge.
- **Meta-World MT50**: The task design of Meta-World MT50 is similar to MT10, with the key difference being an expansion of the training tasks to 45, while the number of unseen test tasks remains at 5.

Adhering to the experimental setup of Prompt-DT, we illustrate the task distribution for both training and testing sets in each dataset, as detailed in Table 13. Our experiments are meticulously crafted in accordance with the specifications outlined in this table.

B Hyperparameters configuration

We show the hyperparameter of DPDT and other baslines in Table 8 and Table 9.

Table 8: Common Hyperparameters configuration of DPDT and DPDT-WP.

Hyperparameters	Value
Pretraining model	GPT2-small
K (length of context)	20
Prompt Length	30
training batch size for each task	16
number of evaluation episodes for each task	5
learning rate	1e-4
learning rate decay weight	1e-4
number of layers	12
number of attention heads	12
embedding dimension	768
activation	ReLU
r	1

Table 9: Common Hyperparameters configuration of MT-BC,MT-DT, Soft-prompt, HDT and Prompt-DT.

Hyperparameters	Value
K (length of context P)	20
training batch size for each task	16
number of evaluation episodes for each task	5
learning rate	1e-4 (2e-5 for Prompt-DT)
learning rate decay weight	1e-4 (1e-5 for Prompt-DT)
number of layers	12
number of attention heads	12
embedding dimension	768
activation	ReLU

C Supplementary experiment

Table 10: Ablation: The impact of adaptation method. (1) Combining cross-task prompts P_c with the average of all task-specific prompts P_k from the training set for TTA, (2) freezing the cross-task prompts P_c, we initialized a new task-specific prompt combined with the cross-task prompts for TTA and (3) randomly selecting one P_k from a training task and combining it with P_c for TTA.

Method	Cheetah-vel	MW ML45	MW MT50
(1)	-148.50	332.80	1482.75
(2)	-171.75	320.30	1418.50
(3)	-159.26	325.02	1079.82
DPDT	**-139.88**	**347.21**	**1559.94**

Table 11: Results for Meta-RL control tasks (zero-shot scenarios).

	Soft-Prompt-TTA [45]	**Prompt-DT-TTA [6]**	**DPDT**
Cheetah-dir	$2.91_{\pm 0.74}$	$8.33_{\pm 0.41}$	$50.32_{\pm 11.47}$
Cheetah-vel	$-160.51_{\pm 10.13}$	$-204.57_{\pm 8.36}$	$-139.88_{\pm 19.65}$
Ant-dir	$119.14_{\pm 10.72}$	$120.07_{\pm 2.67}$	$121.84_{\pm 8.01}$
MW ML10	$251.37_{\pm 6.17}$	$316.74_{\pm 12.05}$	$371.01_{\pm 9.41}$
MW ML45	$301.74_{\pm 18.55}$	$299.47_{\pm 8.97}$	$347.21_{\pm 11.52}$
MW MT 10	$541.99_{\pm 10.46}$	$1027.80_{15.58}$	$1317.52_{\pm 8.22}$
MW MT 50	$519.78_{\pm 20.97}$	$1134.72_{\pm 7.92}$	$1559.94_{\pm 2.49}$
Average	226.35	386.08	518.28

Table 12: Computer resources (memory, time of execution)

Method	Batch Size (each)	GPU usage	Wall Time
MT-BC	16	10.4GB	$\approx$ 5-6 hours
MT-ORL	16	10.4GB	$\approx$ 1 day
Soft-Prompt	16	20.1GB	$\approx$ 5 hours
HDT	16	40.8GB	$\approx$ 2 hours
Prompt-DT	16	10.9GB	$\approx$ 4 hours
DPDT-WP	16	20.5GB	$\approx$ 2-3 hours
DPDT	16	20.5GB	$\approx$ 2-3 hours

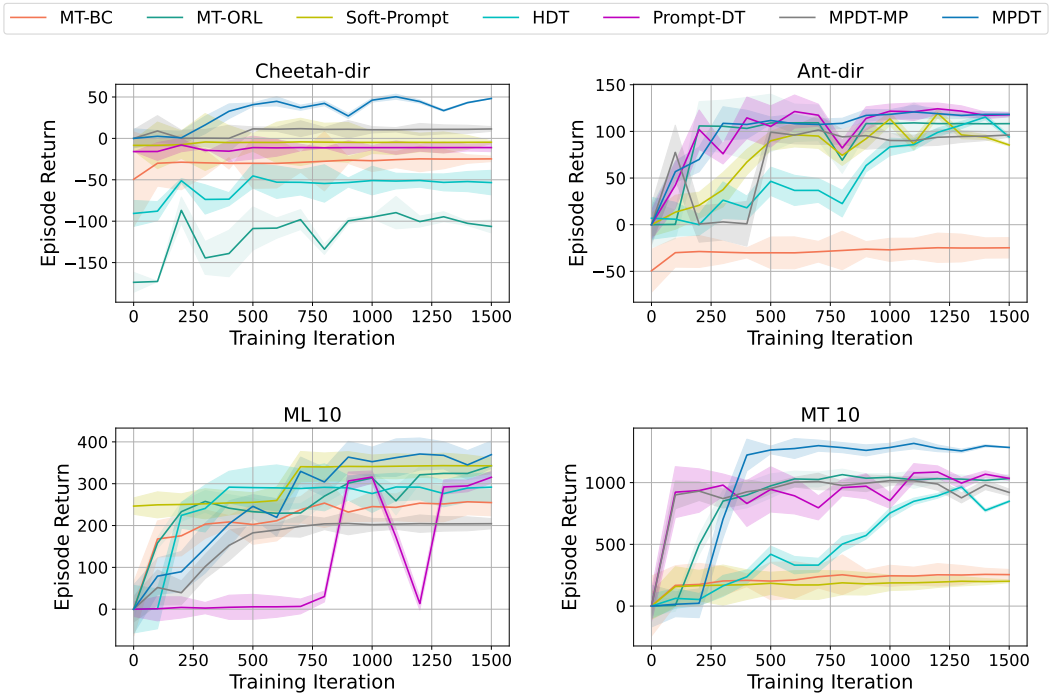

Figure 4: Episodic accumulated returns in four unseen tasks of MTBC, MT-ORL, Soft-Prompt, HDT, Prompt-DT, DPDT-WP and DPDT. Each method is restricted to 1500 rounds of runs in each environment.

Table 13: Training and testing task indexes when testing the generalization ability in unseen tasks.

Cheetah-dir	
Training set of size 2	$[0, 1]$
Testing set of size 2	$[0.1]$
Cheetah-vel	
Training set of size 35	$[0-1, 3-6, 8-14, 16-22, 24-25, 27-39]$
Testing set of size 5	$[2, 7, 15, 23, 26]$
Ant-dir	
Training set of size 45	$[0-5, 7-16, 18-22, 24-29, 31-40, 42-49]$
Testing set of size 5	$[6, 17, 23, 30, 41]$
Meta-World ML10	
Training set of size 10	$[0, 9, 19, 29, 33, 36, 39, 40, 48, 49]$
Testing set of size 3	$[11, 24, 41]$
Meta-World ML45	
Training set of size 45	$[0-10, 12-16, 18-24, 26-35, 37-40, 42-49]$
Testing set of size 5	$[11, 17, 25, 36, 41]$
Meta-World MT10	
Training set of size 10	["assembly-v2", "button-press-topdown-v2", "coffee-push-v2", "dial-turn-v2", "disassemble-v2", "door-open-v2", "hand-insert-v2", "drawer-open-v2","box-close-v2","push-wall-v2"]
Testing set of size 3	["peg-unplug-side-v2","hammer-v2","handle-press-v2"]
Meta-World MT50	
Training set of size 45	["basketball-v2", "bin-picking-v2", "button-press-topdown-v2", "button-press-v2", "coffee-button-v2", "coffee-pull-v2", "coffee-push-v2", "dial-turn-v2", "disassemble-v2", "door-close-v2", "door-lock-v2", "door-open-v2", "hand-insert-v2", "drawer-close-v2", "drawer-open-v2", "faucet-close-v2", "handle-press-v2", "handle-pull-side-v2", "handle-pull-v2", "lever-pull-v2","peg-insert-side-v2", "pick-place-wall-v2", "pick-out-of-hole-v2", "reach-v2", "push-back-v2", "push-v2", "pick-place-v2", "plate-slide-v2", "plate-slide-back-v2", "plate-slide-back-side-v2", "soccer-v2", "push-wall-v2", "shelf-place-v2", "sweep-into-v2", "sweep-v2", "window-open-v2", "window-close-v2","assembly-v2","button-press-topdown-wall-v2", "hammer-v2","peg-unplug-side-v2", "reach-wall-v2", "stick-push-v2", "stick-pull-v2", "box-close-v2"]
Testing set of size 5	["plate-slide-side-v2", "handle-press-side-v2", "buttonpress-wall-v2", "door-unlock-v2", "faucet-open-v2"]

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We introduce the scope of this paper in the abstract and the Introduction section, and summarize the contribution in the Introduction section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We state the limitation in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: If you are including theoretical results, did you state the full set of assumptions of all theoretical results, and did you include complete proofs of all theoretical results?

   Answer: [NA]

Justification: Not applicable.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: If the contribution is a dataset or model, what steps did you take to make your results reproducible or verifiable?

   Answer: [Yes]

   Justification: Yes, we provide a detailed experimental description in 5.1. All the datasets, code, and model checkpoints are publicly available.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: If you ran experiments, did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)?

Answer: [Yes]

Justification: All the datasets, code, and model checkpoints are publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: If you ran experiments, did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?

   Answer: [Yes]

   Justification: Both in main paper and Appendices A and B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide standard deviations in numerical form in Tables 1 and Tables 2. Additionally, Figure 2 includes standard deviations represented as shaded areas.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the information in Table 12.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Have you read the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines` and ensured that your research conforms to it?

   Answer: [Yes]

   Justification: This paper does not involve human subjects, and we have not used any personal data.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: If appropriate for the scope and focus of your paper, did you discuss potential negative societal impacts of your work?

    Answer: [Yes]

    Justification: We discuss the broader impact in Section 6

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Do you have safeguards in place for responsible release of models with a high risk for misuse (e.g., pretrained language models)?

    Answer: [NA]

    Justification: The paper uses standard benchmark datasets and does not involve any high-risk data or models. The pre-trained models and datasets are publicly available.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses**

    Question: If you are using existing assets (e.g., code, data, models), did you cite the creators and respect the license and terms of use?

    Answer: [Yes]

    Justification: We use standard benchmark datasets, which are properly credited and have open licenses.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **Assets**

    Question: If you are releasing new assets, did you document them and provide these details alongside the assets?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: If you used crowdsourcing or conducted research with human subjects, did you include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **IRB Approvals**

    Question: Did you describe any potential participant risks and obtain Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your institution), if applicable?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.