SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2026, Vol. 69, Iss. 1, 112105:1–112105:13
https://doi.org/10.1007/s11432-024-4545-1

Prompt tuning with preference ranking for few-shot pre-trained decision transformer

Shengchao HU[1,2], Li SHEN[3*], Ya ZHANG[1,2] & Dacheng TAO[4]

[1]*School of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China*
[2]*Shanghai Artificial Intelligence Laboratory, Shanghai AI Laboratory, Shanghai 200233, China*
[3]*School of Cyber Science and Technology, Shenzhen Campus of Sun Yat-sen University, Shenzhen 518107, China*
[4]*School of Computer and Data Science, Nanyang Technological University, Singapore 639798, Singapore*

Received 17 October 2024/Revised 13 January 2025/Accepted 29 March 2025/Published online 4 January 2026

Abstract Prompt tuning has emerged as a promising method for adapting pre-trained models to downstream tasks or aligning with human preferences. Prompt learning is widely used in natural language processing (NLP) but has limited applicability to reinforcement learning (RL) due to the complex physical meaning and environment-specific information contained within RL prompts. Directly extending prompt-tuning approaches to RL is challenging because RL prompts guide agent behavior based on environmental modeling and analysis, rather than adjusting the prompt format for downstream tasks as widely used in NLP. In this work, we propose the prompt-tuning decision transformer (DT) algorithm to address these challenges by using trajectory segments as prompts to guide RL agents in acquiring environmental information and optimizing prompts via black-box tuning to enhance their ability to contain more relevant information, thereby enabling agents to make better decisions. Our approach involves randomly sampling a Gaussian distribution to fine-tune the elements of the prompt trajectory and using the preference ranking function to find the optimization direction, thereby providing more informative prompts and guiding the agent toward specific preferences in the target environment. Extensive experiments show that with only 0.03% of the parameters learned, Prompt-Tuning DT achieves comparable or even better performance than full-model fine-tuning in few-shot settings. Our research represents a pioneering contribution to the development of prompt-tuning techniques within RL, offering a promising avenue for optimizing large-scale pre-trained RL agents for tasks tailored to specific preferences.

Keywords reinforcement learning, few-shot learning, preference learning, prompt tuning, ranking optimization

Citation Hu S C, Shen L, Zhang Y, et al. Prompt tuning with preference ranking for few-shot pre-trained decision transformer. Sci China Inf Sci, 2026, 69(1): 112105, https://doi.org/10.1007/s11432-024-4545-1

1 Introduction

Pre-trained large-scale models (PLMs) [1–4] have proven to be highly effective for a wide range of tasks due to their high transferability and competitive performance on downstream tasks with limited target data. However, full-model fine-tuning requires updating and storing all the parameters of the PLM, which is memory-intensive and impractical for maintaining a separate set of parameters for each task during inference. Recently, prompt-tuning [5, 6] has emerged as a promising alternative to full-model fine-tuning, allowing for the effective adaptation of pre-trained models to specific downstream tasks and human preferences. By freezing the pre-trained model's parameters and tuning only the prompts, prompt-tuning approaches have demonstrated comparable performance to full-model fine-tuning methods across various model scales and tasks [7–9].

Offline reinforcement learning (offline RL) is a data-driven approach that learns an optimal policy from trajectories collected by a set of behavior policies, without requiring access to the environments. This approach is critical in many settings, where online interactions are expensive or dangerous. However, offline RL struggles with generalization to unseen tasks and adaptation to preferences, as the agent may not find a good policy in the test tasks due to the distribution shift. Recent works address this challenge through offline meta-RL, which leverages the algorithmic learning perspective [10–12]. In contrast, we aim to investigate the power of prompt-tuning methods with PLMs. Nonetheless, unlike natural language processing (NLP) prompts, RL prompts are more complex and contain environment-specific information, which may be vulnerable to the prompt learning process. Additionally, prompt-tuning approaches from NLP cannot be directly applied to RL prompts, as RL prompts guide agent behavior based on environmental modeling and analysis rather than adjusting the prompt format for downstream tasks.

* Corresponding author (email: mathshenli@gmail.com)

Therefore, there is an urgent need to develop novel prompt-tuning techniques specifically tailored to RL that can guide agents toward specific preferences in the target environment.

In this paper, we propose a novel algorithm called prompt-tuning decision transformer (DT) as an approach to tackle the challenge of generalization in offline RL from the perspective of prompt tuning. Our approach leverages trajectory segments as prompts to guide RL agents in acquiring target environmental information and optimizes prompts via black-box tuning to enhance their ability to contain more meaningful information. Prompt-tuning is essential in RL as it enables pre-trained agents to make better decisions by providing more informative prompts. This contrasts with the limitations inherent in straightforward prompt-based adaptation methods [13]: the process of generating high-quality trajectory prompts involves significant investments of time and resources, and the prompt's effectiveness is constrained by the model's input capacity for conditioning prompts [8]. As a result, despite the progress made in prompt-based adaptation, downstream task quality still lags far behind that of full-model fine-tuning methods.

In our prompt-tuning offline RL framework, we first pre-train the agent using offline trajectories from various tasks within the same environment. For each task, the agent learns to predict a target trajectory by conditioning on a trajectory prompt sampled from the same task. During evaluation, the agent is presented with a new task and a small set of new trajectories for fine-tuning the prompt. Our approach perturbs each element of the prompt by randomly sampling from a Gaussian distribution to avoid catastrophic deviations and employs a preference ranking function along with a ranking algorithm to determine the optimization direction. In this experiment, we revisit conventional prompt tuning in the context of RL and demonstrate that our Prompt-Tuning DT outperforms these baselines. Notably, by optimizing only 0.03% of the model parameters, Prompt-Tuning DT achieves performance comparable to full-model fine-tuning and surpasses other parameter-efficient methods. Our work contributes to the advancement of prompt-tuning approaches in RL, providing a promising direction for optimizing PLMs for specific preferences and downstream tasks.

In summary, our main contributions are as follows.

- We propose Prompt-Tuning DT, a memory-efficient alternative to fine-tuning pre-trained agents that achieves comparable performance to full-model fine-tuning methods.
- We present a prompt-tuning RL framework, which leverages a PLM's API to enable streamlined customization for specific preferences with minimal parameter modifications.
- We revisit conventional prompt tuning in the context of RL and demonstrate that Prompt-Tuning DT outperforms these baselines, highlighting its effectiveness as a few-shot learner for generalization in offline meta RL.

2 Related work

Offline RL. Offline RL has emerged as a promising paradigm for learning from fixed, limited datasets consisting of trajectory rollouts from arbitrary policies [14]. However, deploying off-policy RL algorithms directly in the offline setting can be challenging due to the distributional shift problem, which can result in a significant performance drop [15]. To overcome this issue, model-free RL algorithms adopt various strategies, such as constraining the action space of the policy [16] or introducing pessimism to the value function [17], to explicitly correct the distributional mismatch between the behavior policy in the offline data and the optimized policy. In contrast, model-based RL algorithms estimate the environmental reward and transition functions using offline data and require modifications to the RL algorithm to avoid exploiting errors in the estimated model [18, 19].

Offline RL has been increasingly viewed as a sequence modeling task, and transformer-based decision models have been applied to this domain. The objective is to predict the next sequence of actions based on the sequence of recent experiences, which includes state-action-reward triplets. This approach can be trained using supervised learning, which makes it more suitable for offline RL and imitation learning. Several studies have explored the use of Transformers in RL under the sequence modeling paradigm, including Gato [4], Graph DT [20], HarmoDT [21], QT [22], DeMa [23], and the survey works [24, 25]. In this study, we propose a novel approach that is based on Prompt-DT [13] and incorporates prompt-tuning techniques to enhance its performance on downstream target tasks.

Meta RL. Meta-learning algorithms [26, 27] enable efficient learning of new tasks by learning the process of learning itself. In the context of meta-RL, the objective is to generalize an agent's knowledge from one task to another. In recent years, several studies have delved into the problem of offline meta-RL. MBML [28] addresses a scenario where task identity is spuriously inferred due to biased datasets and applies the triplet loss to robustify the task inference with reward relabeling. BOReL [29] extends the online Meta-RL method VariBAD [30] to the offline setup, where they assume known reward functions for each task and use reward relabelling to share data across

tasks with shared dynamics. On the other hand, MACAW [10] proposes an offline Meta-RL algorithm based on MAML [31]. Their approach includes an advantage weighting loss and introduces a policy update in the inner loop to theoretically increase the richness of the policy's update and empirically improve adaptation performance and stability. In this study, we investigate an alternative perspective on meta-RL using sequence modeling and prompt engineering, which can achieve comparable or superior performance to traditional methods.

Prompt learning. Prompt learning is a promising methodology in NLP that involves optimizing a small subset of parameters while leaving the main model architecture unchanged. The basic premise of prompt learning involves presenting the model with a cloze test-style textual prompt, which the model is then expected to fill in with the corresponding answer. Autoprompt [6] proposes an automatic prompt search methodology for efficiently finding optimal prompts, while recent advancements in prompt learning have adopted trainable continuous embeddings for prompt representation [5, 8]. Prompt learning has also been applied to the vision-language domain, where introducing continuous prompts into pre-trained vision-language models has led to significant improvements in few-shot visual recognition and generalization performance [32, 33]. While prompt learning reduces the number of tunable parameters, back-propagation through the entire model is still necessary to calculate gradients and update the small subset of parameters. Gradient-free methods have been proposed to optimize continuous [34] or discrete [35] prompts. Despite the great success of prompt-tuning in the fields of NLP and CV, its application in RL has not been thoroughly explored. Therefore, in this study, we propose the Prompt-Tuning DT method that employs gradient-free methods to optimize continuous trajectory prompts with a preference ranking oracle. This approach can be extended to a human-in-the-loop environment, where candidate prompts are ranked manually.

3 Preliminary

In this section, we first define the Markov decision process and provide a concise overview of the core components of our algorithm, namely the decision transformer and ranking optimization. The decision transformer adapts the transformer architecture for offline RL by formulating RL tasks as sequence modeling problems, enabling the development of large-scale RL models. Additionally, we introduce a ranking optimization approach, which leverages ranking data to optimize the model without the need for explicit gradient computation. These algorithms form the basis of our approach illustrated in Section 4.

3.1 Markov decision process

The objective of RL is to learn a policy $\pi_\theta(a \mid s)$ that maximizes the expected cumulative discounted reward $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)\right]$ in a Markov decision process (MDP). An MDP is formally defined as a six-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s' \mid s, a) : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ describes the environment dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and d_0 is the initial state distribution [36]. In the offline setting [14], instead of the online environment, 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 $\mathcal{D}$, without online interactions with the environment.

3.2 Decision transformer

Transformer [37], extensively studied in NLP [2] and CV [38], has also been explored in RL using the sequence modeling pattern [24]. Moreover, recent works from NLP suggest that Transformers pre-trained on large-scale datasets are capable of few-shot or zero-shot learning under the prompt-based framework [1, 39]. Building upon this, Gato [4] and TTP [40] both extend the prompt-based framework to the offline RL setting, constructing pre-trained large-scale agents designed to address multiple tasks concurrently in a zero-shot or few-shot fashion. Both methods are based on the DT [41] which treats learning a policy as a sequence modeling problem. DT introduces the notion of modeling trajectories through state s_t, action a_t, and reward-to-go $\hat{r}_t$ tuples collected at distinct time steps t. The reward-to-go token quantifies the cumulative reward from the current time step to the end of the episode, defined as $\hat{r}_t = \sum_{j=t}^{T} r_j$, where T denotes the maximum number of interactions with the environment. During training with offline collected data, DT processes a trajectory sequence τ_t in an auto-regressive manner which encompasses the most recent K-step historical context,

$$\tau_t = (\hat{r}_{t-K+1}, s_{t-K+1}, a_{t-K+1}, \ldots, \hat{r}_t, s_t, a_t). \tag{1}$$

The prediction head associated with a state token s_t is trained to predict the corresponding action a_t. Regarding continuous action spaces, the training objective is to minimize the mean-squared loss:

$$L_{DT} = \mathbb{E}_{\tau_t \sim \mathcal{D}} \left[\frac{1}{K} \sum_{j=t-K+1}^{t} (a_j - \pi(\tau_t)_j)^2 \right], \tag{2}$$

where $\mathcal{D}$ represents the offline dataset corresponding to the current task $\mathcal{T}$, π denotes the policy induced by the DT, and $\pi(\tau_t)_j$ represents the j-th output generated by the DT.

3.3 Ranking optimization

Black-box optimization, which utilizes a derivative-free framework to optimize the target function, has been extensively studied in the optimization literature for several decades [42–44]. With the rapid development of reinforcement learning with human feedback (RLHF), ranking data, which enables humans to express their personal preferences in a straightforward and intuitive manner [45, 46], has demonstrated great potential for use in various applications, especially those where the exact value of personal information is sensitive, such as healthcare or finance. ZO-RankSGD [47–49] is an effective approach for model optimization that finds the descent direction directly from the ranking information, without the need for knowledge of the gradient of the model or the exact value of the data. Our work focuses on a specific class of ranking oracles that return only the sorted indices of the top-ranked elements. Such oracles are widely regarded as intuitive and natural in human decision-making processes [50], which could be formulated as follows.

Definition 1 ((m,k)-ranking oracle [49])**.** Given a function $f : \mathbb{R}^d \to \mathbb{R}$ and m points $x_1, \ldots, x_m$ to query, an (m,k) ranking oracle $O_f^{(m,k)}$ returns k smallest points sorted in their order. For example, if $O_f^{(m,k)}(x_1, \ldots, x_m) = (i_1, \ldots, i_k)$, then

$$f(x_{i_1}) \leqslant f(x_{i_2}) \leqslant \cdots \leqslant f(x_{i_k}) \leqslant \min_{j \notin \{i_1, \ldots, i_k\}} f(x_j).$$

With the ranking oracle $O_f^{(m,k)}$ and a starting point x, we can query $O_f^{(m,k)}$ with the inputs $x^i = x + \mu\xi_i$, $\xi_i \sim N(0, I_d)$, for $i = 1, \ldots, m$, and μ is a constant. With the directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ constructed from the ranking information of $O_f^{(m,k)}$, where the node set $\mathcal{N} = \{1, \ldots, m\}$ and the directed edge set $\mathcal{E} = \{(i,j) \mid f(x_i) < f(x_j)\}$, the rank-based gradient estimator can be formulated as follows:

$$\tilde{g}(x) = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \frac{x^j - x^i}{\mu} = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} (\xi_j - \xi_i). \tag{3}$$

Then the point can be updated with $x_{\text{new}} = x - \eta\tilde{g}(x)$, where η is the learning rate and $\tilde{g}(x)$ is the estimated gradient. With the help of the preference ranking oracle and ZO-RankSGD algorithm, we are able to optimize the prompt, guiding the agent towards human preferences in the target environment.

4 Prompt-tuning decision transformer

This section introduces prompt-tuning as a memory-efficient alternative to full-model fine-tuning for the pre-trained agents in the context of few-shot policy generalization tasks. We begin by presenting the problem formulation in Subsection 4.1 and subsequently provide a formal definition of our method in Subsection 4.2. The overall procedure of our proposed Prompt-Tuning DT is illustrated in Figure 1.

4.1 Problem formulation

In our few-shot evaluation experiments, our objective is to align the output of the PLM with human preferences using a restricted number of offline trajectories and limited oracle calls, all accomplished in a parameter-efficient manner. To better quantitatively evaluate our method, we adopt high cumulative reward, a widely-used indicator in the field of RL, as a representation of human preference and conduct experiments on few-shot generalization tasks, which involve training the agent on a set of tasks using offline data and evaluating its ability to generalize to new tasks.

There are two distinct sets of tasks, denoted as $\mathcal{T}^{train}$ and $\mathcal{T}^{test}$, ensuring that there is no overlap between them ($\mathcal{T}^{train} \cap \mathcal{T}^{test} = \emptyset$). This arrangement requires the model to perform well on tasks with goals that lie outside

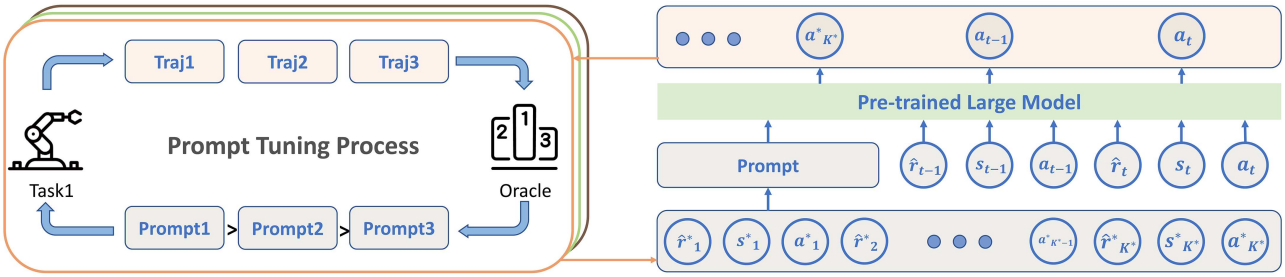

Figure 1 (Color online) Application of Prompt-Tuning DT. At each iteration, the PLM generates different trajectories (e.g., Traj1, Traj2, Traj3) for the current task based on different prompts (e.g., Prompt1, Prompt2, Prompt3) and the most recent K-step history. These prompts are generated by perturbing the initial prompt using a random Gaussian distribution. The generated trajectories are then ranked based on a specific property using a preference ranking oracle, and the ranking information is leveraged to update the prompt.

the training range, thereby necessitating the ability to generalize to out-of-distribution tasks. Each task $\mathcal{T}_i$ in the training set $\mathcal{T}^{train}$ is associated with a corresponding dataset $\mathcal{D}_i$, which consists of pre-collected trajectories obtained using an unknown behavior policy π_i. For a test task $\mathcal{T}_i \in \mathcal{T}^{test}$, there are two possible approaches to adapt to the new domain. One approach involves updating the model parameters using task-specific offline data $\mathcal{P}_i$, which is usually much smaller than the training dataset $|\mathcal{P}_i| << |\mathcal{D}_i|$. Alternatively, one can incorporate task-specific prompts derived from $\mathcal{P}_i$ to mitigate the issue of distribution shift, although such approaches are generally considered inferior to fine-tuning methods [1]. Our method combines the advantages of both approaches to fine-tune prompts.

4.2 Deep black-box tuning

Trajectory prompts contain only the necessary information to aid in task identification while being insufficient for the agent to imitate. Therefore, the prompt length K^* should not be too long; in our experiment, we set $K^* = 5$. Each trajectory prompt contains multiple tuples of state s^*, action a^* and reward-to-go $\hat{r}^*$, denoted as $(s^*, a^*, \hat{r}^*)$, following the representation in [13, 41]. Each element with superscript $\cdot^*$ is associated with the trajectory prompt, which can be formulated as

$$\tau^* = (\hat{r}_1^*, s_1^*, a_1^*, \ldots, \hat{r}_{K^*}^*, s_{K^*}^*, a_{K^*}^*). \tag{4}$$

In contrast to the prompt learning approach typically employed in NLP, where a cloze test-style textual prompt is presented to the model for filling in the corresponding answer, the trajectory prompt utilized in the decision transformer consists of tokens that have unique representations and physical interpretations. These tokens are carefully crafted to represent essential components of RL tasks, including the state, action, and return-to-go. The state token encapsulates the environmental information of the agent at a given position and is usually represented by a high-dimensional vector. On the other hand, the action token exhibits significant variations across dimensions, with specific values corresponding to distinct actions. Moreover, the return-to-go token serves to denote the expected reward that we aim for the agent to attain. Given these distinct characteristics of RL prompts, directly applying prompt-tuning approaches from NLP becomes challenging: RL prompts are specifically tailored to guide agent behavior by leveraging environmental modeling and analysis, rather than primarily focusing on adjusting the prompt format as in NLP prompt learning.

We utilize the ZO-RankSGD optimization approach to update the trajectory prompt. This method avoids explicit gradient computation and eliminates the necessity for an intricate understanding of the particular structure of the PLM. Given the initial trajectory prompt τ^*, we concatenate one trajectory segment as a unit and add a standard Gaussian distribution to it to avoid catastrophic deviations:

$$\begin{aligned} x_0 &= \hat{r}_1^* \,||\, s_1^* \,||\, a_1^* \,||\, \cdots \,||\, \hat{r}_{K^*}^* \,||\, s_{K^*}^* \,||\, a_{K^*}^*, \\ x_0^n &= x_0 + \mu\xi_n, \quad \xi_n \sim N(0, I_{d_x}), \end{aligned} \tag{5}$$

where $||$ means concatenation, $\hat{r}_i^* \in \mathbb{R}^{d_r}, s_i^* \in \mathbb{R}^{d_s}, a_i^* \in \mathbb{R}^{d_a}$, and $d_x = (d_r + d_s + d_a) \times K^*$.

For the ranking function f, we propose two preference ranking functions tailored to different RL environments (offline and online): the offline loss function and the online reward function. For the offline setting, where we have access to a set of trajectories $\mathcal{P}$ collected in advance, we utilize the MSE loss between the true action and predicted

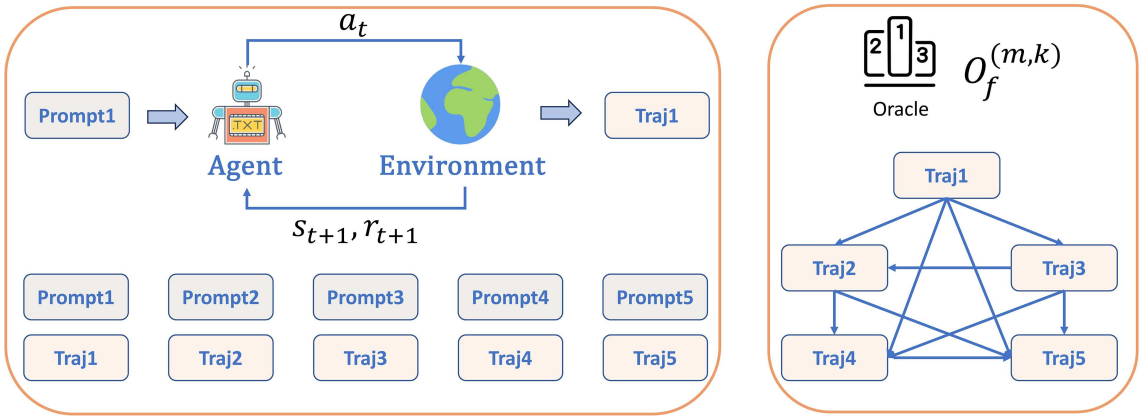

Figure 2 (Color online) The workflow of our proposed method. Given an initial prompt, the agent is first guided to generate a corresponding trajectory. Based on the trajectory, along with an appropriate ranking function f and oracle $O_f^{(m,k)}$, we return the k smallest prompts, sorted in ascending order. In this figure, the DAG of the trajectories is used to obtain the corresponding ranking of prompts: $O_f^{(5,3)}$(prompt1, prompt2, prompt3, prompt4, and prompt5) = (1, 3, 2). These rankings are then used to estimate the gradient for updating the prompt in subsequent steps.

action as the preference ranking function:

$$f(x_0^n) = \mathbb{E}_{\tau_t^{\text{offline}} \sim \mathcal{P}} \left[\frac{1}{K} \sum_{j=t-K+1}^{t} (a_j - \pi(x_0^n, \tau_t^{\text{offline}})_j)^2 \right]. \tag{6}$$

While for the online setting, where we can interact with the simulator, we consider the episode accumulated reward obtained by the model during online interactions as the preference ranking function, which is represented as follows:

$$f(x_0^n) = -\mathbb{E}_{\tau^{\text{online}}} \left[\sum_{j=1}^{T} \mathcal{R}(s_j, \pi(x_0^n, \tau_j^{\text{online}})_j) \right], \tag{7}$$

where τ^{online} represents the trajectory collected from online interactions, and T denotes the maximum number of interactions with the environment.

Note that since the function is optimized to the minimum, we need to add a minus sign in front of (7). In both cases, the selection of the preference ranking function aims to adapt to the human preference for high cumulative reward, which also serves as a widely used metric for evaluating a pre-trained model's performance. Then the ranking oracle $O_f^{(m,k)}$ simply returns the order of these values, which is subsequently utilized for computing the gradient estimator (as illustrated in Figure 2).

Human judgment can also be employed as an oracle to rank these trajectories based on individual preferences. However, this study does not delve into comprehensive experiments within human-in-the-loop settings, leaving this aspect for future investigations. In this context, we primarily showcase the algorithm's potential in a human-in-the-loop framework from a design perspective. (1) Ranking information possesses a unique appeal to humans as it offers a straightforward and intuitive means to express personal preferences without the need for exact scores or ratings, making our approach user-friendly. (2) The forward-forward fine-tuning strategy proves advantageous in terms of conserving GPU memory, which is significance for deployment on devices with limited resources. (3) Ranking-based approaches avoid intricate understanding of the PLM's structure, and leverage the PLM's API enables effective prompt fine-tuning in alignment with human preferences. Collectively, these attributes render our method well-suited for human-in-the-loop environments. The primary objective of this article is to establish the method's feasibility and efficiency.

We summarize the entire procedure of prompt-tuning in Algorithm 1. Prompt-Tuning DT employs an approximate gradient calculation to adapt the pre-trained agent to specific preferences. Gaussian noise is introduced to the initial prompt, driving Prompt-Tuning DT to discover a more expressive prompt tailored to the target tasks. There are two options available for the ranking function. The offline loss function requires pre-collected datasets from the target tasks in $\mathcal{T}^{test}$, while the online reward function assumes interaction with a simulator for the target tasks in $\mathcal{T}^{test}$. After E iterations of the fine-tuning process, we utilize the optimized result to initialize the prompt τ^* at the onset of the evaluation stage and update the recent history τ with streaming collected data.

Algorithm 1 Prompt-tuning DT.

Require: Initial prompt τ_0^*, stepsize η, fine-tune iterations E, maximal iteration in test T, smoothing parameter μ;
// fine-tune the prompt
Given the test task $\mathcal{T}$ with corresponding few-shot trajectories $\mathcal{P}$;
Construct the initial point from prompt: $x_0 = \hat{r}_1^* \,||\, s_1^* \,||\, a_1^* \,||\, \cdots \,||\, \hat{r}_{K^*}^* \,||\, s_{K^*}^* \,||\, a_{K^*}^*$;
for $e = 1$ to E **do**
  Sample m i.i.d. random vectors $\{\xi_{(e,1)}, \ldots, \xi_{(e,m)}\}$ from $N(0, I_{d_x})$;
  Query the ranking function to obtain the exact value with offline loss function (6) or online reward function (7) with input $\{x_{e-1}^n\}_{n=1}^m$, where $x_{e-1}^n = x_{e-1} + \mu\xi_{(e,n)}$;
  Construct the corresponding DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ as described in Subsection 3.3;
  Compute the gradient estimator by using: $g_e = \frac{1}{|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}}(\xi_{(e,j)} - \xi_{(e,i)})$;
  $x_e = x_{e-1} - \eta g_e$;
end for
// inference with the learned prompt x_E
Sample the initial state s_0 and return-to-go $\hat{r}_0$ and construct $\tau_0 = \hat{r}_0||s_0$;
Initialize $R = 0$;
for $t = 0, \ldots, T$ **do**
  Sample action a_t based on $\pi(x_E, \tau_t)_t$;
  Receive next state s_{t+1} and reward r_{t+1} and construct the newly trajectory $\tau_{t+1} = \tau_t||a_t||\hat{r}_t - r_{t+1}||s_{t+1}$;
  $R = R + r_{t+1}$;
end for
Output R.

4.3 Theoretical analysis

To establish the convergence properties of our algorithm, we first introduce the underlying assumptions and subsequently present the formal convergence results of the proposed method.

Assumption 1 (Assumptions on the function f)**.** The ranking function f satisfies the following conditions.

(1) f is twice continuously differentiable.
(2) f is L-smooth, meaning that $\|\nabla^2 f(x)\| \leqslant L$.
(3) f is lower bounded by a value f^*, i.e., $f(x) \geqslant f^*$ for all x.

Theorem 1 (Convergence guarantee [49])**.** After running Algorithm 1 for E iterations and taking $\eta = \sqrt{\frac{1}{d_x E}}$ and $\mu = \sqrt{\frac{d_x}{C_{d_x}^2 E}}$, where C_{d_x} is some constant that only depends on d_x, we have

$$\mathbb{E}\left[\min_{e\in\{1,\ldots,E\}} \|\nabla f(x_{e-1})\|\right] = \mathcal{O}\left(\sqrt{\frac{d}{E}}\right). \tag{8}$$

Theorem 1 guarantees that the expected minimum norm of the gradient over E iterations converges at a rate of $\mathcal{O}(\sqrt{\frac{d}{E}})$. The rate indicates that the algorithm achieves sublinear convergence with respect to the number of iterations, which is consistent with other zeroth-order optimization methods [49, 51].

5 Experiment

We perform experiments to assess the performance of Prompt-Tuning DT by using the episode accumulated reward as the evaluation metric. Our experimental evaluation seeks to answer the following research questions. (1) How do conventional prompt-tuning methods perform in the offline meta-RL setting? (2) Can our prompt-tuning approach achieve comparable performance to full-model fine-tuning with limited ranking oracle calls? (3) What is the impact of the fine-tuning dataset size on the effectiveness of the prompt-tuning approach? (4) How do the quality and quantity of prompts affect the performance of the prompt-tuning approach? (5) How does the hyper-parameter influence the effectiveness of the prompt-tuning approach?

5.1 Datasets and tasks

In this study, we assess the performance of our proposed approach on several datasets that are used in meta-RL [10, 31, 52, 53], namely Cheetah-dir, Cheetah-vel, Ant-dir, Meta-World (MV) reach-v2, and Meta-World-10-Reward. The objectives of these tasks are distinct, with Cheetah-dir and Ant-dir incentivizing high velocity in the goal direction, Cheetah-vel penalizing deviations from the target velocity using l_2 errors, the MV reach-v2 task requiring the robot's end-effector to reach a designated position in 3D space, and Meta-World-10-Reward

encompassing 10 tasks with varying reward and transition functions, testing the agents' adaptability to more complex meta-environments.

We adopt the dataset construction and settings from [10] for the meta-RL control tasks considered in this study. Specifically, the datasets comprise the full replay buffer of Soft Actor-Critic [54] for Cheetah-dir, Ant-dir, and Meta-World-10-Reward, and TD3 [55] for Cheetah-vel. Expert trajectories are collected for MV reach-v2 [53] using scripted expert policies provided in the environment. By evaluating our approach on these diverse tasks, we can assess its performance and generalization capabilities across different scenarios.

5.2 Baselines

We assess the few-shot generalization capabilities of Prompt-Tuning DT by comparing it against five baseline methods across meta-RL control tasks. Our approach begins with pre-training a PLM on the training tasks $\mathcal{T}^{train}$ using the DT architecture and the DT loss (2). This pre-trained PLM is then directly applied for inference on the test tasks $\mathcal{T}^{test}$ without any prompt and further adaptation, denoted as "PLM" in Table 1:

$$\mathbb{E}_{\tau^{\text{online}}}\left[\sum_{j=1}^{T}\mathcal{R}(s_j,\text{PLM}(\tau_j^{\text{online}})_j)\right], \tag{9}$$

where τ^{online} represents the trajectory collected from online interactions, and T denotes the maximum number of interactions with the environment. We explore four additional few-shot learning methods.

• **PDT (prompt decision transformer) [13].** This method collects prompts from the target tasks $\mathcal{T}^{test}$ and incorporates these prompts into the input to assist the PLM in better adapting to the target tasks, a process known as straightforward prompt-based adaptation without any further update process:

$$\mathbb{E}_{\tau^{\text{online}}}\left[\sum_{j=1}^{T}\mathcal{R}(s_j,\text{PLM}(\tau_{\text{prompt}}^{*},\tau_j^{\text{online}})_j)\right], \tag{10}$$

where τ_{prompt}^{*} denotes the prompt trajectory, sampled from the corresponding offline dataset $\mathcal{P}$.

• **Soft-Prompt.** This method treats collected prompts as soft embeddings and uses a separate optimizer to fine-tune these embeddings, similar to common practices in NLP [8]. It adds an additional update step for the learnable prompt:

$$\max_{h}\mathbb{E}_{\tau^{\text{online}}}\left[\sum_{j=1}^{T}\mathcal{R}(s_j,\text{PLM}(h,\tau_j^{\text{online}})_j)\right], \tag{11}$$

where h is initialized with the prompt trajectory τ_{prompt}^{*}. The update follows the DT loss (2) based on few-shot trajectories $\mathcal{P}$, using a gradient-based method (AdamW [56]).

• **Adaptor.** This method employs a parameter-efficient technique, previously explored in HDT [57]. Adaptors are integrated into each decoder module of the PLM, and during inference, only the adaptors are fine-tuned to improve the PLM's adaptation to the target tasks:

$$\max_{\theta^{*}}\mathbb{E}_{\tau^{\text{online}}}\left[\sum_{j=1}^{T}\mathcal{R}(s_j,\text{PLM}(\tau_{\text{prompt}}^{*},\tau_j^{\text{online}}|\theta,\theta^{*})_j)\right], \tag{12}$$

where θ^{*} represents the newly introduced adaptor parameters, following [57]. The update is based on the DT loss from few-shot trajectories, using the AdamW optimizer.

• **Full-Model-FT.** This approach serves as an upper bound for fine-tuning techniques in scenarios with access to full data [5]:

$$\max_{\theta}\mathbb{E}_{\tau^{\text{online}}}\left[\sum_{j=1}^{T}\mathcal{R}(s_j,\text{PLM}(\tau_{\text{prompt}}^{*},\tau_j^{\text{online}}|\theta)_j)\right], \tag{13}$$

where θ represents the parameters of the PLM. The update is based on the DT loss from few-shot trajectories, using the AdamW optimizer.

Table 1 Results for meta-RL control tasks. The best mean scores are highlighted in bold. Each environment has prompts of length $K^* = 5$ and limited fine-tuned datasets. Scores are normalized so that 100 represents an expert policy. Our methods outperform other parameter-efficient methods on all tasks and even achieve comparable performance with the Full-Model-FT method.

Algorithm	PLM	PDT	Soft-Prompt	Adaptor	PTDT-offline	PTDT-online	Full-Model-FT
Random prompt initialization							
Cheetah-dir	-3.8 ± 0.3	94.7 ± 0.0	$\mathbf{95.5 \pm 0.3}$	74.5 ± 2.0	$\mathbf{95.5 \pm 0.0}$	95.1 ± 0.6	93.3 ± 1.0
Cheetah-vel	7.1 ± 0.9	44.2 ± 0.1	44.6 ± 0.1	19.7 ± 4.6	$\mathbf{61.2 \pm 3.2}$	60.5 ± 7.9	44.0 ± 9.8
Ant-dir	24.7 ± 1.4	61.7 ± 0.2	66.5 ± 0.4	75.3 ± 6.6	75.3 ± 3.9	$\mathbf{78.7 \pm 0.2}$	77.5 ± 1.7
MW reach-v2	45.5 ± 1.0	42.1 ± 5.8	41.8 ± 5.8	0.3 ± 0.1	$\mathbf{54.0 \pm 2.2}$	49.9 ± 4.4	43.9 ± 13.0
Average	18.4	60.7	62.1	41.2	**71.5**	71.0	64.7
Expert prompt initialization							
Cheetah-dir	-3.8 ± 0.3	94.6 ± 0.5	$\mathbf{95.5 \pm 0.1}$	75.5 ± 0.9	$\mathbf{95.5 \pm 0.1}$	95.4 ± 0.1	93.6 ± 0.7
Cheetah-vel	7.1 ± 0.9	86.0 ± 1.4	86.4 ± 1.2	27.1 ± 4.5	$\mathbf{87.8 \pm 0.4}$	87.5 ± 0.1	81.9 ± 0.5
Ant-dir	24.7 ± 1.4	71.3 ± 0.3	72.7 ± 0.3	76.8 ± 6.6	75.5 ± 0.4	74.5 ± 0.3	$\mathbf{84.2 \pm 1.6}$
MW reach-v2	45.5 ± 1.0	50.9 ± 6.6	50.9 ± 6.6	0.3 ± 0.1	$\mathbf{56.5 \pm 1.0}$	53.1 ± 3.0	54.2 ± 8.6
Average	18.4	75.7	76.4	43.1	**78.8**	77.6	78.5
Meta-World-10-Reward with expert prompt initialization							
Coffee-push-v2	11.9 ± 4.3	19.4 ± 6.3	32.7 ± 12.6	0.7 ± 0.3	35.6 ± 11.0	35.0 ± 9.5	$\mathbf{54.6 \pm 12.8}$
Door-open-v2	18.2 ± 1.5	28.7 ± 2.0	26.4 ± 5.1	29.0 ± 12.4	30.0 ± 3.4	29.5 ± 4.4	$\mathbf{48.3 \pm 13.0}$
Handle-pull-v2	5.4 ± 3.3	20.2 ± 7.9	25.2 ± 7.4	16.0 ± 12.1	$\mathbf{33.1 \pm 10.9}$	33.0 ± 12.2	25.0 ± 18.7
Average	11.8	22.7	28.1	15.2	32.9	32.5	**42.6**

Our approach encompasses two distinct variations: Prompt-Tuning DT with an offline loss function (PTDT-offline) and Prompt-Tuning DT with an online reward function (PTDT-online). To maintain fairness in the comparison, all offline methodologies are confined to utilizing an equivalent quantity of samples $\mathcal{P}_i$ sourced from the target task $\mathcal{T}^{test}$. While PTDT-online involves interaction with a simulator, we meticulously regulate the number of interactions to guarantee access to new trajectories of comparable magnitudes. Note that all methods utilize the same PLM to ensure an equitable comparison. By aligning these experimental setups, we establish a robust foundation for an unbiased assessment of the methods' performance, thereby enhancing the validity of our findings.

5.3 Main results

We perform a comparative analysis between Prompt-Tuning DT and the parameter-efficient baseline methods to assess their few-shot generalization abilities and evaluate the tuning efficiency of Prompt-Tuning DT in relation to the full-model fine-tuning approach. We use the episode accumulated reward as the evaluation metric in the test task set $\mathcal{T}^{test}$. Note that we present two prompt initialization settings: the random prompt, gathered from a random policy, and the expert prompt, acquired from the expert policy. Given the challenge of generalizing to varying reward and transition functions in the Meta-World-10-Reward environment, we limit our evaluation to the expert prompt in this setting. The main results are normalized and presented in Table 1, which showcases the few-shot performance of various algorithms.

The outcomes from the PLM underscore the significance of prompts, as PLM struggles to adapt to target tasks in zero-shot scenarios. During the random prompt initialization setting, PDT effectively leverages pre-collected prompts by incorporating them into the PLM input, resulting in substantial improvements. The adaptor also exhibits enhanced performance over the PLM by introducing supplementary fine-tuning adaptors within decoder modules. However, its efficacy is hampered, particularly in the MW reach-v2 and cheetah-vel environments, likely due to limited target datasets $\mathcal{P}$. Both Soft-Prompt and our proposed approach undertake further fine-tuning of the prompt itself. While Soft-Prompt treats the prompt as an embedding and optimizes it using the AdamW optimizer, it achieves better results than PDT but lags behind our approach. This discrepancy can be attributed to the intricate and environment-specific nature of RL prompts, which necessitate meticulous updates. Our approach demonstrates effectiveness in both offline and online settings, yielding notable performance improvements that surpass the benchmark of Full-Model-FT, which serves as a primary reference for evaluating the efficacy of our approach.

Under the expert prompt initialization setting, characterized by strong prior knowledge, all baseline approaches exhibit substantial enhancements in comparison to their counterparts in the random initialization setting. Moreover, the relative improvement of our method, compared with other approaches, diminishes under the expert prompt regime. This reduction can be attributed to the strong prior condition introduced by expert trajectories, which

Table 2 Ablation study on the impact of parameters m and μ in our PTDT-offline method, initialized with random prompts. Experiments are conducted in the Ant-dir environment, with results averaged over three independent runs and normalized for comparison. In each run, only one parameter is varied while the other is held constant to ensure a robust evaluation.

	$\mu = 1$					
	$m = 10$	$m = 20$	$m = 30$	$m = 50$	$m = 80$	$m = 100$
PTDT-offline	71.0 ± 5.8	73.0 ± 4.0	75.3 ± 3.9	72.9 ± 3.8	73.8 ± 2.3	74.0 ± 2.2
	$m = 30$					
	$\mu = 0.5$	$\mu = 1$	$\mu = 5$	$\mu = 10$	$\mu = 15$	$\mu = 20$
PTDT-offline	72.6 ± 2.2	75.3 ± 3.9	70.9 ± 3.8	64.0 ± 4.0	60.2 ± 4.3	59.2 ± 4.4

Table 3 Ablation study examining the effect of prompt length on prompt-tuning methodologies, initialized with random prompts. Experiments are conducted in the Ant-dir environment, with results averaged over three independent runs and normalized for comparison. The proposed methods demonstrate consistently strong performance across a wide range of prompt lengths.

	Length				
	2	5	10	15	20
PDT	53.2 ± 4.1	61.7 ± 0.2	57.1 ± 3.1	56.2 ± 3.3	54.7 ± 4.0
Soft-Prompt	57.5 ± 3.0	66.5 ± 0.4	61.1 ± 2.1	60.2 ± 2.3	59.2 ± 2.7
PTDT-offline	45.7 ± 2.1	75.3 ± 3.9	78.1 ± 2.7	76.2 ± 2.2	76.0 ± 2.4

constrains the extent of improvement across methods. Nevertheless, despite this limitation, our approach consistently outperforms all other parameter-efficient methods. Furthermore, our method achieves outcomes on par with Full-Model-FT. Collectively, these outcomes accentuate the effectiveness of our prompt-tuning techniques across both random and expert prompt initialization scenarios.

When contrasting random and expert initialization conditions, it is crucial to highlight that the ultimate performance of our PTDT-offline method, when initialized randomly, has not only matched but, in certain instances, exceeded the outcomes attained through expert initialization (in comparison with PTDT-expert). This phenomenon can be attributed to the high efficiency of our method, which is capable of rapidly steering randomly initialized prompts into the proximity of expert-level prompts.

In the Meta-World-10-Reward setting, the reward and transition functions differ between the training and test tasks, making fine-tuning increasingly important for assessing the methods' generalization capabilities. In this context, fine-tuning only the prompt or the adaptor may not be sufficient to generalize to complex new tasks, where full-model fine-tuning achieves the best performance. Nevertheless, our approach consistently demonstrates effectiveness across all test tasks among parameter-efficient methods, highlighting the robustness of our method.

5.4 Ablation

Random search. Considering the random search can lead to a lot of variability in the performance of the algorithm, we further investigate the impact of the number m and variance μ of Gaussian random variables. The results are shown in Table 2. When we increase the number of samples m during each update, the algorithm explores a larger set of possible directions to evaluate the performance, leading to a more accurate gradient estimation (variance decrease). However, as m increases, the burden on the oracle, which needs to provide ranking information for the m samples, also grows. On the other hand, increasing the variance of the Gaussian distribution μ also allows the algorithm to explore a broader range of potential directions for performance evaluation. However, a higher variance of the Gaussian μ also introduces larger variability in the gradient estimation, which may not consistently guarantee performance improvement and can potentially have a detrimental effect on the optimization process.

Prompt length. We investigate the impact of prompt length on prompt-tuning methods, considering its influence on both the number of tuning parameters in the approach and the speed of inference. It is crucial to strike a balance between the richness of information provided by the prompt and the effectiveness of the prompt-tuning process. The results are shown in Table 3. Our primary focus is on investigating Soft-Prompt and PTDT-offline. We employ PDT as the baseline, which does not involve additional fine-tuning processes. The augmentation of the prompt does not uniformly lead to enhanced generalization performance for both PDT and Soft-Prompt. Our method also achieves peak performance at a prompt length of 10, after which a slight decline is observed. This indicates that excessively long prompts may introduce redundancy or noise, adversely impacting performance. Nevertheless, our method demonstrates strong performance across a wide range of prompt lengths, underscoring its effectiveness in refining prompts, even when managing a larger number of tuning parameters.

Sample efficiency. We explore the impact of progressively increasing the number of fine-tuning samples on the performance of fine-tuning approaches, aiming to understand the prompt-tuning methods' dependence on the

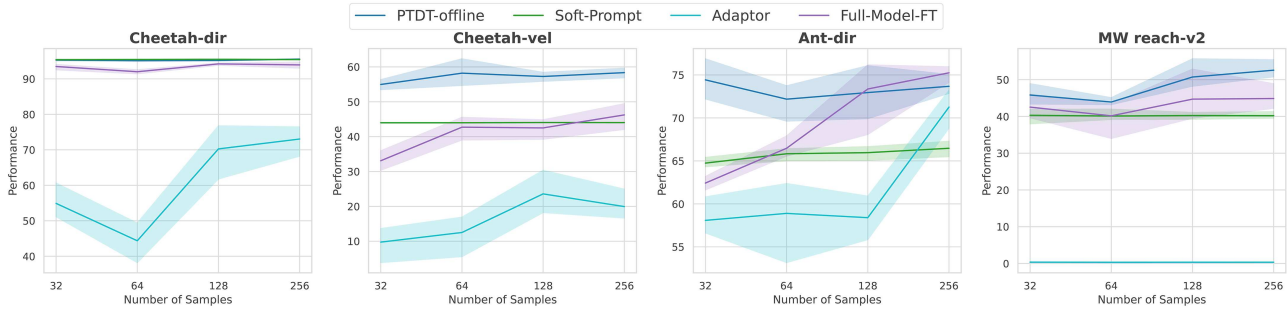

Figure 3 (Color online) Comparison between different fine-tuning approaches when different numbers of training samples are available. Each plot represents the results averaged over 3 independent seeds. The x-axis indicates the training sample size, while the y-axis represents the evaluation metric (with higher values indicating better performance). Our method consistently demonstrates superior performance across different training sample sizes, highlighting its effectiveness.

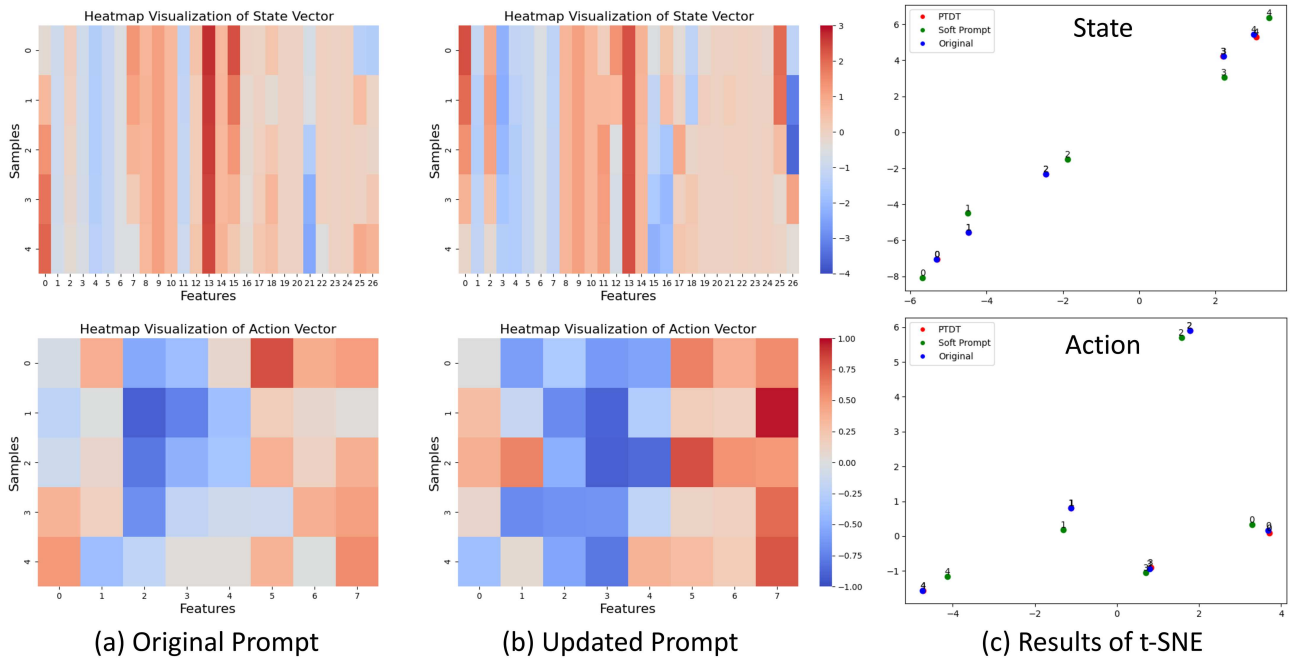

Figure 4 (Color online) Ablation study on prompt visualization conducted in the Ant-dir environment with expert prompt initialization. For this visualization, we generate state vectors and action vectors from five different samples (y-axis). (a) corresponds to the original prompts. (b) represents the prompts after fine-tuning using the PTDT-offline method. (c) presents a t-SNE visualization, where the dimensionality of the states and actions of these prompts is reduced and the final results are displayed in a two-dimensional graph.

quantity of fine-tuning samples. Figure 3 illustrates the performance trends of these methods on the Cheetah-dir, Cheetah-vel, Ant-dir, and MW reach-v2 environments. Prompt-tuning methods (PTDT, Soft-Prompt) exhibit consistent performance across varying sample sizes, whereas model-tuning methods (Adaptor, Full-Model-FT) exhibit incremental improvements as the number of samples increases.

Furthermore, it is worth noting that unlike the observed phenomenon in NLP [5, 58], the performance of these fine-tuning approaches does not exhibit a monotonically increasing trend as the amount of fine-tuning data increases. In all environments, a downward inflection point is observed in the performance curve as the number of samples increases. This phenomenon can be attributed to the presence of "bad samples" in the training dataset, which negatively impact the fine-tuning process and may lead to performance degradation. As the dataset size increases, the occurrence of such "bad samples" becomes more frequent, preventing performance from consistently improving with additional data.

5.5 Visualization

In this subsection, we provide additional visualizations to enhance the intuitiveness of our experimental results in the ablation study. We present heatmaps in Figure 4 to compare the original prompts (a) with the prompts after

fine-tuning (b). These heatmaps visualize the state and action vectors for five samples (0–4 on the y-axis). The heatmaps highlight notable changes in specific dimensions resulting from the prompt-tuning process. Specifically, when examining the state vector, we observe a consistent trend of low values in dimensions such as the 7th and 15th. A similar pattern is also evident in the action vector. While these figures may not provide directly interpretable content, they offer valuable insights into the dimensions that significantly influence the final performance.

Beyond the heatmaps, we include t-SNE visualizations of the state and action vectors, as depicted in Figure 4(c). These visualizations include the original prompt alongside the updated prompts generated by the Soft-Prompt method and our proposed method. By facilitating an exploration of the high-level distribution of prompts, these visualizations offer additional insights into the adaptation process. The t-SNE results indicate that the prompts obtained through our method exhibit minimal deviations from the original prompts, maintaining a high degree of overlap with the original distribution. In contrast, prompts fine-tuned using the Soft-Prompt method deviate significantly from the original distribution. Despite these differences, both methods yield improvements over the original prompts. These findings suggest that aligning with the structure of the original distribution may be a more effective strategy for prompt tuning, particularly when there is no predefined "optimal prompt". By maintaining proximity to a known "good" prompt distribution, our approach leverages the favorable properties of the original prompts while avoiding drastic deviations that may not consistently lead to improved performance.

6 Conclusion and future work

In this paper, we introduce the Prompt-Tuning DT, a novel algorithm that aligns with human preferences in the target environment. By optimizing prompts without back-propagation, Prompt-Tuning DT offers a memory-efficient alternative to fine-tuning PLMs. Furthermore, our prompt-tuning offline RL framework using trajectory prompts allows for effective adaptation to new tasks with minimal parameter optimization and a small number of trajectories. Through extensive experiments, our approach achieves performance on par with full-model fine-tuning and surpasses alternative parameter-efficient methods.

Our work contributes to the advancement of prompt-tuning approaches in RL, providing a promising direction for optimizing pre-trained large-scale RL agents for specific preferences and downstream tasks. Our approach demonstrates the potential of prompt-tuning methods in RL settings and opens up avenues for future research in developing tailored prompt-tuning techniques for RL agents. We envision that prompt-tuning approaches will continue to play a crucial role in enhancing the generalization and adaptability of RL agents in real-world scenarios.

Acknowledgements This work was supported by STI 2030—Major Projects (Grant No. 2021ZD0201405) and Shenzhen Basic Research Project (Natural Science Foundation) Basic Research Key Project (Grant No. JCYJ20241202124430041).

Supporting information Appendixes A–E. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. Adv Neural Inform Process Syst, 2020, 33: 1877–1901
2 Devlin J, Chang M W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. ArXiv:1810.04805
3 Lee K H, Nachum O, Yang M S, et al. Multi-game decision transformers. Adv Neural Inform Process Syst, 2022, 35: 27921–27936
4 Reed S, Zolna K, Parisotto E, et al. A generalist agent. ArXiv:2205.06175
5 Li X L, Liang P. Prefix-tuning: optimizing continuous prompts for generation. ArXiv:2101.00190
6 Shin T, Razeghi Y, Logan IV R L, et al. Autoprompt: eliciting knowledge from language models with automatically generated prompts. ArXiv:2010.15980
7 Liu X, Zheng Y, Du Z, et al. Gpt understands, too. ArXiv:2103.10385
8 Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning. ArXiv:2104.08691
9 Zhong Z, Friedman D, Chen D. Factual probing is [MASK]: learning vs. learning to recall. ArXiv:2104.05240
10 Mitchell E, Rafailov R, Peng X B, et al. Offline meta-reinforcement learning with advantage weighting. In: Proceedings of the International Conference on Machine Learning, 2021. 7780–7791
11 Nichol A, Achiam J, Schulman J. On first-order meta-learning algorithms. ArXiv:1803.02999
12 Rajeswaran A, Finn C, Kakade S M, et al. Meta-learning with implicit gradients. In: Proceedings of Advances in Neural Information Processing Systems, 2019
13 Xu M, Shen Y, Zhang S, et al. Prompting decision transformer for few-shot policy generalization. In: Proceedings of the International Conference on Machine Learning, 2022. 24631–24645
14 Levine S, Kumar A, Tucker G, et al. Offline reinforcement learning: tutorial, review, and perspectives on open problems. ArXiv:2005.01643
15 Fujimoto S, Meger D, Precup D. Off-policy deep reinforcement learning without exploration. In: Proceedings of the International Conference on Machine Learning, 2019. 2052–2062
16 Kumar A, Fu J, Soh M, et al. Stabilizing off-policy q-learning via bootstrapping error reduction. In: Proceedings of Advances in Neural Information Processing Systems, 2019
17 Kumar A, Zhou A, Tucker G, et al. Conservative Q-learning for offline reinforcement learning. Adv Neural Inform Process Syst, 2020, 33: 1179–1191
18 Kidambi R, Rajeswaran A, Netrapalli P, et al. MoReL: model-based offline reinforcement learning. Adv Neural Inform Process Syst, 2020, 33: 21810–21823
19 Yu T, Thomas G, Yu L, et al. MOPO: model-based offline policy optimization. Adv Neural Inform Process Syst, 2020, 33: 14129–14142

20 Hu S, Shen L, Zhang Y, et al. Graph decision transformer. ArXiv:2303.03747
21 Hu S, Fan Z, Shen L, et al. HarmoDT: harmony multi-task decision transformer for offline reinforcement learning. In: Proceedings of the International Conference on Machine Learning, 2024. 19182–19197
22 Hu S, Fan Z, Huang C, et al. Q-value regularized transformer for offline reinforcement learning. In: Proceedings of the International Conference on Machine Learning, 2024. 19165–19181
23 Dai Y, Ma O, Zhang L, et al. Is Mamba compatible with trajectory optimization in offline reinforcement learning? Adv Neural Inform Process Syst, 2024, 37: 51474–51502
24 Hu S, Shen L, Zhang Y, et al. On transforming reinforcement learning with transformers: the development trajectory. Proc IEEE Trans Pattern Anal Mach Intell, 2024, 46: 8580–8599
25 Li W, Luo H, Lin Z, et al. A survey on transformers in reinforcement learning. ArXiv:2301.03044
26 Wang X, Ji Z, Yu Y, et al. Model attention expansion for few-shot class-incremental learning. Proc IEEE Trans Image Process, 2024, 33: 4419–4431
27 Liu J, Ji Z, Pang Y, et al. NTK-guided few-shot class incremental learning. ArXiv:2403.12486
28 Li J, Vuong Q, Liu S, et al. Multi-task batch reinforcement learning with metric learning. Adv Neural Inform Process Syst, 2020, 33: 6197–6210
29 Dorfman R, Shenfeld I, Tamar A. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. Adv Neural Inform Process Syst, 2021, 34: 4607–4618
30 Zintgraf L, Shiarlis K, Igl M, et al. VariBAD: a very good method for Bayes-adaptive deep RL via meta-learning. ArXiv:1910.08348
31 Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the International Conference on Machine Learning, 2017. 1126–1135
32 Zhou K, Yang J, Loy C C, et al. Learning to prompt for vision-language models. Int J Comput Vis, 2022, 130: 2337–2348
33 Zhou K, Yang J, Loy C C, et al. Conditional prompt learning for vision-language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022. 16816–16825
34 Sun T, Shao Y, Qian H, et al. Black-box tuning for language-model-as-a-service. In: Proceedings of the International Conference on Machine Learning, 2022. 20841–20855
35 Prasad A, Hase P, Zhou X, et al. GrIPS: gradient-free, edit-based instruction search for prompting large language models. ArXiv:2203.07281
36 Sutton R S, Barto A G. Reinforcement Learning: an Introduction. Cambridge: MIT Press, 2018
37 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Advances in Neural Information Processing Systems, 2017
38 Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16×16 words: transformers for image recognition at scale. ArXiv:2010.11929
39 Liu P, Yuan W, Fu J, et al. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Comput Sur, 2023, 55: 1–35
40 Jain V, Lin Y, Undersander E, et al. Transformers are adaptable task planners. In: Proceedings of the Conference on Robot Learning, 2023. 1011–1037
41 Chen L, Lu K, Rajeswaran A, et al. Decision transformer: reinforcement learning via sequence modeling. Adv Neural Inform Process Syst, 2021, 34: 15084–15097
42 Frazier P I. A tutorial on Bayesian optimization. ArXiv:1807.02811
43 Nesterov Y, Spokoiny V. Random Gradient-Free Minimization of Convex Functions. Found Comput Math, 2017, 17: 527–566
44 Loshchilov I, Hutter F. CMA-ES for hyperparameter optimization of deep neural networks. ArXiv:1604.07269
45 Bai Y, Jones A, Ndousse K, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. ArXiv:2204.05862
46 Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback. Adv Neural Inform Process Syst, 2022, 35: 27730–27744
47 Cai H Q, McKenzie D, Yin W, et al. A one-bit, comparison-based gradient estimator. Appl Comput Harmon Anal, 2022, 60: 242–266
48 Bergou E H, Gorbunov E, Richtárik P. Stochastic three points method for unconstrained smooth minimization. SIAM J Optim, 2020, 30: 2726–2749
49 Tang Z, Rybin D, Chang T H. Zeroth-order optimization meets human feedback: provable learning via ranking oracles. ArXiv:2303.03751
50 Keeney R L. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge: Cambridge University Press, 1993
51 Wang Y, Du S, Balakrishnan S, et al. Stochastic zeroth-order optimization in high dimensions. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2018. 1356–1365
52 Rothfuss J, Lee D, Clavera I, et al. ProMP: proximal meta-policy search. ArXiv:1810.06784
53 Yu T, Quillen D, He Z, et al. Meta-world: a benchmark and evaluation for multi-task and meta reinforcement learning. In: Proceedings of the Conference on Robot Learning, 2020. 1094–1100
54 Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the International Conference on Machine Learning, 2018. 1861–1870
55 Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proceedings of the International Conference on Machine Learning, 2018. 1587–1596
56 Loshchilov I, Hutter F, et al. Fixing weight decay regularization in Adam. ArXiv:1711.05101
57 Xu M, Lu Y, Shen Y, et al. Hyper-decision transformer for efficient online policy adaptation. ArXiv:2304.08487
58 Gu Y, Han X, Liu Z, et al. PPT: pre-trained prompt tuning for few-shot learning. ArXiv:2109.04332