# Prompt Tuning Decision Transformers with Structured and Scalable Bandits

**Finn Rietz**[*]
Örebro University
finn.rietz@oru.se

**Oleg Smirnov**
King AI Labs, Microsoft Gaming
oleg.smirnov@microsoft.com

**Sara Karimi**
King AI Labs, Microsoft Gaming
sara.karimi@king.com

**Lele Cao**
King AI Labs, Microsoft Gaming
lelecao@microsoft.com

## Abstract

Prompt tuning has emerged as a key technique for adapting large pre-trained Decision Transformers (DTs) in offline Reinforcement Learning (RL), particularly in multi-task and few-shot settings. The Prompting Decision Transformer (PDT) enables task generalization via trajectory prompts sampled uniformly from expert demonstrations – without accounting for prompt informativeness. In this work, we propose a bandit-based prompt-tuning method that learns to construct optimal trajectory prompts from demonstration data at inference time. We devise a structured bandit architecture operating in the trajectory prompt space, achieving linear rather than combinatorial scaling with prompt size. Additionally, we show that the pre-trained PDT itself can serve as a powerful feature extractor for the bandit, enabling efficient reward modeling across various environments. We theoretically establish regret bounds and demonstrate empirically that our method consistently enhances performance across a wide range of tasks, high-dimensional environments, and out-of-distribution scenarios, outperforming existing baselines in prompt tuning.

## 1 Introduction

The ability to exploit large amounts of offline data is crucial for training foundation models capable of generalizing across diverse tasks (Radford, 2018; Reed et al., 2022; Brohan et al., 2022). In Reinforcement Learning (RL), a seminal contribution is the Decision Transformer (DT) (Chen et al., 2021), which reframes offline RL (Levine et al., 2020) as a sequence modeling problem, thereby unlocking powerful Transformer architectures for offline RL. DT is particularly well-suited for offline RL because it sidesteps the well-known instabilities of temporal-difference learning with function approximation (Sutton & Barto, 2018), which are exacerbated under distribution shift in the offline setting, by replacing discounted, moving-target value estimates with stationary return-to-go conditioning, autoregressive sequence modeling, and causal attention mechanisms (Chen et al., 2021). The Prompting Decision Transformer (PDT) (Xu et al., 2022) further extends DT from single-task to multi-task settings, enabling large-scale models and generalized pre-training in offline multi-task and few-shot RL (Xu et al., 2022; Mitchell et al., 2021). Analogous to prompting in Large Language Models (LLMs), PDT differentiates tasks through a *stochastic trajectory prompt* prepended to the context, allowing it to identify and model optimal action marginals for each task in the offline dataset. This makes PDT particularly appealing for offline multi-task RL, as it avoids additional learning

[*] Work performed while the author was an intern at King AI Labs (part of Microsoft Gaming). Code available at: https://github.com/king/pdt-bandits

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

on the downstream task while enabling efficient and robust adaptation of the model's behavior at inference time, solely through adjustments of the prompt.

However, PDT samples prompts uniformly, overlooking that not all prompts are equally informative, even in fully observable MDPs. We hypothesize that the sampling of non-informative prompts from expert demonstrations can diminish PDT's ability to differentiate between tasks, thereby leading to performance degradation.

While improving on PDT's uninformed prompt-sampling strategy, prior works on PDT prompt-tuning suffer from key limitations. The approach by Yuan et al. (2024) replaces trajectory prompts with less-expressive goal-conditioning and relies on hindsight relabeling. The generative approaches by Hu et al. (2023, 2024) are not applicable in discrete settings and don't adhere to causal relationships between prompt tokens. Furthermore, all of these works treat prompts as flat, unstructured inputs and operate directly on MDP modalities, which leads to poor scaling with prompt size and state- and action-space sizes.

To address these shortcomings, we introduce a scalable, robust, and computationally efficient bandit-based prompt-tuning method for PDT. Our method exploits the inherent structure of the prompt space to reduce the complexity of prompt selection, transforming it from a combinatorial problem into one that scales linearly with prompt size. Moreover, we show how to leverage the pre-trained PDT as a feature extractor to obtain compact prompt representations that enable efficient deployment even in high-dimensional (e.g., pixel-based) settings. By optimizing prompt selection, our approach boosts downstream task performance without costly weight updates to the underlying Transformer backbone. Our experiments reveal clear performance gains with the proposed method, effectively bridging the gap between pre-training and adaptation.

## 2 Preliminaries

In this section, we introduce key concepts and terminologies that form the foundation of this work.

### 2.1 Problem definition: Offline multi-task RL

The offline multi-task RL problem is formalized similarly to prior works (Xu et al., 2022; Mitchell et al., 2021). The objective is to solve a set of training tasks $\mathcal{T}^{\text{train}}$, with the option to evaluate task generalization capabilities on a holdout test set $\mathcal{T}^{\text{test}}$.

For each task $\mathcal{T}_i \in \mathcal{T}^{\text{train}}$, a dataset $\mathcal{D}_i$ is provided, consisting of trajectories sampled from the corresponding MDP $\mathcal{M}_i = \langle \mathcal{S}_i, \mathcal{A}_i, r_i, d_i, \gamma_i, \mu_i^0 \rangle$. Here, $\mathcal{S}_i$ is the state space, $\mathcal{A}_i$ is the action space, $r_i : \mathcal{S}_i \times \mathcal{A}_i \to \mathbb{R}$ represents the reward function, $d_i : \mathcal{S}_i \times \mathcal{A}_i \times \mathcal{S}_i \to [0, 1]$ defines the discrete-time transition dynamics, $\gamma_i \in (0, 1]$ is the discount factor, and $\mu_i^0$ is the initial state distribution of MDP $i$.

The goal is to learn a generalized policy, $\pi(\mathbf{s}, \psi) \to \mathbf{a}$, capable of solving all tasks in $\mathcal{T}^{\text{train}}$. Here, $\psi$ serves as an auxiliary input to the policy that sufficiently describes a task from $\mathcal{T}$. In the case of PDT, $\psi$ is the *stochastic trajectory prompt* sampled from the demonstration set for the target task. For each task, the optimal generalized policy is expected to maximize the corresponding expected discounted reward objective specific to task $i$.

$$J(\pi, i) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma_i^t r_i(\mathbf{s}_t, \mathbf{a}_t)\right] \tag{1}$$

The expectation is taken over trajectories where actions are sampled from the policy $\pi$, and states are sampled using the transition dynamics $d_i$ of the MDP $\mathcal{M}_i$. In the offline setting, only the pre-collected datasets $\mathcal{D} = \{\mathcal{D}_0, \ldots, \mathcal{D}_n\}$ are available for learning, with no additional data collection allowed. A simulator may be used for policy evaluation, but not for gathering new training data due to the offline nature of the problem.

### 2.2 Prompting Decision Transformer

Chen et al. (2021) reframed RL as a sequence modeling problem, leveraging the capabilities of Transformer architectures to model trajectories. A trajectory in DT is represented as a sequence of triplets $(\hat{r}_t, \mathbf{s}_t, \mathbf{a}_t)$, where $\hat{r}_t = \sum_{t'=t}^{T} r_{t'}$ is the return-to-go, $\mathbf{s}_t$ is the state, and $\mathbf{a}_t$ is the action at time step $t$. DT captures the temporal and causal dependencies between states, actions, and rewards, enabling the model to predict optimal actions directly, without relying on explicit value functions or policies.

Xu et al. (2022) extended DT to a multi-task offline RL setting by introducing stochastic trajectory prompts as task-specific context, enabling the model to identify underlying MDPs. Such a prompt $\rho$ is composed of $J$ trajectory segments, each of length $H$, resulting in a total of $J \times H \times 3$ prompt tokens, as per Equation 2, where the superscript $^\star$ denotes tokens associated with the prompt.

$$\rho = \big(\overbrace{\hat{r}^\star_j, \mathbf{s}^\star_j, \mathbf{a}^\star_j, \dots, \hat{r}^\star_{j+H}, \mathbf{s}^\star_{j+H}, \mathbf{a}^\star_{j+H}}^{\tilde{\tau}_1\text{: segment 1}}, \dots, \overbrace{\hat{r}^\star_k, \mathbf{s}^\star_k, \mathbf{a}^\star_k, \dots, \hat{r}^\star_{k+H}, \mathbf{s}^\star_{k+H}, \mathbf{a}^\star_{k+H}}^{\tilde{\tau}_J\text{: segment } J}\big) \tag{2}$$

$$\mathbf{x} = \big(\rho\big) \odot \big(\overbrace{\hat{r}_{t-L}, \mathbf{s}_{t-L}, \mathbf{a}_{t-L}, \hat{r}_{t-L+1}, \mathbf{s}_{t-L+1}, \mathbf{a}_{t-L+1} \dots, \hat{r}_t, \mathbf{s}_t, \mathbf{a}_t}^{\omega_{L:t}: L \text{ most recent transitions}}\big) \tag{3}$$

For a particular training task $\mathcal{T}_i \in \mathcal{T}^{\text{train}}$, PDT learns to model the sequence in Equation 3 by autoregressively predicting the action tokens, where $\odot$ denotes concatenation. While PDT randomly samples the $J$ segments that constitute the prompt $\rho$ from a set of expert demonstrations $\mathcal{P}$, we propose to optimize segment composition to enhance the downstream task performance with a Multi-Armed Bandit approach.

### 2.3 Multi-Armed Bandits

Multi-Armed Bandits (MABs) provide a framework for optimizing stochastic reward functions, making them an effective tool for tasks like prompt tuning. In the standard MAB setting, at each time step $k$, the agent selects an action (or "arm") $a_k \in \mathcal{A}$, where $\mathcal{A}$ is the set of available arms. Here, we use $k$ to denote bandit time steps, to avoid confusion with the MDPs time $t$. The agent then receives a stochastic reward $r_k \sim R(a_k)$, with the goal of maximizing the cumulative reward $\sum_{k=1}^{K} r_k$ over a time horizon $K$ (Auer et al., 2002). This requires balancing the exploration of arms to gather information about their reward distributions $R(a)$ and exploitation of arms with known high rewards while minimizing cumulative regret, defined as:

$$\text{Regret}(K) = \sum_{k=1}^{K} \left[\max_{a \in \mathcal{A}} \mathbb{E}[R(a)] - \mathbb{E}[r_k]\right] \tag{4}$$

Contextual MABs (CMAB) extend this framework by incorporating side information (or "context") $\mathbf{c}_k \in \mathcal{C}$ which is observed before selecting an arm. The reward distribution is then conditioned on both the arm and the context, $r_k \sim R(a_k \mid \mathbf{c}_k)$. The agent's objective is to learn a policy $\pi : \mathcal{C} \to \mathcal{A}$ that maximizes the expected reward $\mathbb{E}\left[\sum_{k=1}^{K} R(\pi(\mathbf{c}_k) \mid \mathbf{c}_k)\right]$. By leveraging shared features across arms through the context $\mathbf{c}_k$, CMABs enable more efficient learning and better generalization, particularly in settings where arms share intrinsic characteristics (Li et al., 2010).

## 3 Related Work

Recently, there has been a surge in methods across various domains aimed at enhancing the performance and generalization of Transformer-based approaches through automatic prompt tuning.

**Prompting in LLMs**. For LLMs, Chen et al. (2024) introduced InstructZero, which uses Bayesian optimization to explore low-dimensional soft prompt vectors. These vectors are then passed to an open-source LLM to generate instructions for the black-box LLM. Building on this, Lin et al. (2023) proposed INSTINCT, which replaces the Gaussian process in Bayesian optimization with a neural network surrogate, leveraging a neural bandit algorithm to enhance expressivity. Additionally, Shi et al. (2024a) demonstrated that the fixed-budget MAB framework enables learning the optimal prompt within a limited number of LLM evaluations. These methods rely on optimization in continuous spaces, followed by prompt reconstruction, whereas our approach directly operates within the PDT prompt space.

**Multi-task DT**. For RL, Hu et al. (2023) proposed generating prompt candidates for PDT by perturbing initial trajectories with Gaussian noise and applying online or offline feedback to guide a ranking-based optimization. Prompt Diffuser (Hu et al., 2024) similarly frames instruction optimization as a conditional generative modeling task, synthesizing prompts from random noise. In contrast, our method constructs prompts directly from expert demonstrations, which is particularly beneficial in settings like discrete state and action spaces, where noise-based approaches are less effective.

Wang et al. (2024a) introduced hierarchical prompting, using two levels of context: one for encoding task-specific information and another for guiding rollouts via demonstration segments. Hyper-Decision Transformer (Xu et al., 2023) augments DT with a hyper-network for adaptation to novel tasks, but both methods rely on large quantities of demonstration data. Our method, by contrast, selects high-quality prompts from only a few expert trajectories. Other approaches focus on representation learning or adaptation, e.g., Wang et al. (2024b) focus on disentangled world models, while Xie et al. (2023) propose latent-variable conditioned RL and fine-tuning. Our approach avoids updating the transformer backbone and relies solely on test-time prompt optimization.

Finally, MGPO (Yuan et al., 2024) proposes a general framework for online prompt tuning. While sharing a similar goal, we provide a method to identify the best prompt from a demonstration dataset, while MGPO iteratively refines the prompt with data collected during online rollouts.

**LLM for RL**. At the intersection of LLM and RL domains, Yang & Xu (2024) and the closely related LaMo method (Shi et al., 2024b) proposed leveraging a pre-trained LLM as an initializer for PDT, harnessing rich linguistic knowledge to boost performance on unseen tasks. In another work, Zheng et al. (2024) introduced an approach to decompose the prompt into cross-task and task-specific components, ensuring more robust test-time adaptation to unseen tasks. Furthermore, the model is initialized by incorporating parameters from a pre-trained language model to provide it with prior knowledge. Compared to those, our approach avoids reliance on a dedicated LLM, sidestepping the fine-tuning and scalability challenges inherent in such methods.

# 4 Method

We now present our inference time prompt-tuning bandit method for offline multi-task RL. Given a dataset $\mathcal{D} = \{\mathcal{D}_0, \ldots, \mathcal{D}_n\}$ of trajectories for $n$ training tasks $\mathcal{T}^{\text{train}}$, we utilize the original PDT method to learn the generalized policy described in Equation 1. Details of the PDT algorithm and training process are provided by Xu et al. (2022).

An optimal PDT model $\pi^*(\mathbf{x};\theta)$ is assumed to be trained until convergence on $\mathcal{D}$. After training, the goal is to evaluate and enhance the model's performance on a specific task $\mathcal{T}_i$, either from the set of holdout test tasks $\mathcal{T}^{\text{test}}$ or from the training set $\mathcal{T}^{\text{train}}$. We assume access to a small set of demonstrations $\mathcal{P}_i$ for the target task, which serve as a source for sampling prompts, as well as a simulator of the corresponding $\mathcal{M}_i$ to perform online evaluations. Our primary objective is to *identify the stochastic trajectory prompt constructible from $\mathcal{P}_i$ that maximizes performance* when applied to the pre-trained generalized policy.

## 4.1 Bandit-based Prompt Tuning

We propose leveraging a bandit-based approach to efficiently identify the optimal trajectory prompt. A naïve implementation of bandit-based prompt tuning would involve deploying a bandit with one arm for each possible prompt $\rho$ constructible from $\mathcal{P}_i$, i.e., considering all possible combinations of segments among demonstration trajectories. This approach, however, scales extremely poorly, as the number of prompts (and consequently, the size of the bandit problem) grows linearly with the total number of transitions in $\mathcal{P}_i$ and combinatorially with $J$, the number of segments in each prompt. Additionally, treating each prompt as a separate and independent arm disregards prompt similarity, leading to inefficient sample complexity. Thus, we propose to optimize the selection of segments $\tilde{\tau}$ that form the stochastic trajectory prompt $\rho = (\tilde{\tau}_1, \ldots, \tilde{\tau}_J)$ with a special *contextual* MAB algorithm.

As illustrated in Figure 1, at each round $k$, the bandit selects a prompt $\rho_k$ consisting of $J$ segments, balancing exploration and exploitation in the prompt space. The process of constructing the prompt $\rho_k$ through the bandit mechanism is discussed in detail in the next section. We then perform the $k$-th rollout of the PDT $\pi^*(\mathbf{x};\theta)$ in $\mathcal{M}_i$, using the selected prompt $\rho_k$ and the most recent transitions, as per Equation 3. The resulting performance $G_i^k = \sum_{t=0}^{T} r_i(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{a}_t \sim \pi^*(\mathbf{x}_k;\theta)$ serves as

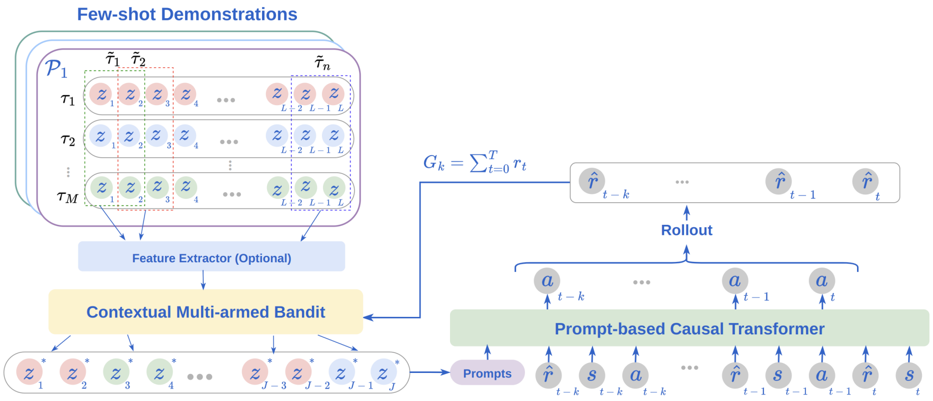

Few-shot Demonstrations
$\mathcal{P}_1$
$\tilde{\tau}_1$ $\tilde{\tau}_2$
$\tilde{\tau}_n$
$\tau_1$
$\tau_2$
$\tau_M$
Feature Extractor (Optional)
Contextual Multi-armed Bandit
$G_k = \sum_{t=0}^{T} r_t$
Rollout
Prompt-based Causal Transformer
Prompts

Figure 1: Overview of our bandit-based prompt-tuning method for multi-task learning with PDT. Each $z_i$ represents a triplet $(\hat{r}_i, \mathbf{s}_i, \mathbf{a}_i)$, each $\tilde{\tau}$ represents a prompt segment, each $\tau$ represents a demonstration trajectory. The bandit explores the demonstration dataset $\mathcal{P}_i$ for the current task $i$ to find the best prompt $\rho^* = (\tilde{\tau}_1^*, \ldots, \tilde{\tau}_J^*)$. The online return $G_k$ achieved by the underlying PDT model at round $k$ and using prompt $\rho_k$ serves as a reward for the bandit.

reward from the bandit's perspective and is stored together with the prompt $\rho_k$ for training the bandit. The algorithmic pseudocode for prompt-tuning with our bandit is provided in the Algorithm 1 in Appendix A.

### 4.2 Scalable and Sample-Efficient Bandit Architecture

To address the aforementioned scalability issues in a naïve bandit, we design a structured bandit architecture that factorizes the problem across prompt segments. Our structured architecture features $J$ arms, one for each segment in the stochastic trajectory prompt. Each arm $j \in \{1, \ldots, J\}$ maintains an *independent* reward model $\phi_j$, which predicts the performance of the underlying PDT when a given segment $\tilde{\tau}$ is placed at position $j$ in the prompt. This decomposition reduces the complexity of the search space from combinatorial to linear in $J$. Exploration can be performed using strategies such as $\epsilon$-greedy, Upper Confidence Bounds (UCB) (Li et al., 2010; Zhou et al., 2020), Thompson Sampling (TS) (Thompson, 1933), or other mechanisms.

To select the prompt $\rho_k$, each model $\phi_j$ predicts the reward for each segment $\{\tilde{\tau}_0, \ldots, \tilde{\tau}_n\} \in \mathcal{P}_i$[2] in its position, $j$, in the prompt, resulting in a prediction matrix $\mathbf{Y} \in \mathcal{R}^{J \times |\mathcal{P}_i|}$. The $J$ segments with the highest predicted performance are then selected by computing $\arg\max$ over the segment dimension of $\mathbf{Y}$, subject to the chosen exploration strategy. This yields the prompt $\rho_k = (\tilde{\tau}_1^k, \ldots, \tilde{\tau}_J^k)$. After the rollout $k$ of the PDT with prompt $\rho_k$ in $\mathcal{M}_i$, all reward models are independently updated on their accumulated $\langle \tilde{\tau}_j^k, G_i^k \rangle$ pairs. The complete algorithmic pseudocode for prompt selection and update with our structured bandit architecture is provided in Algorithms 2 and 3 in Appendix A.

### 4.3 Regret Analysis

We present a regret bound for our bandit architecture, assuming that the reward for a prompt $\rho = (\tau_1, \ldots, \tau_J)$ is estimated as the average of $J$ *independent* reward models $\phi_j(\tilde{\tau})$. This estimate incurs an approximation error due to unmodeled correlations between segments, which is bounded by a small constant $\varepsilon$.

The prompt-segment independence assumption is motivated by the modular structure of prompts in PDT. Prompt segments correspond to localized behaviors, and many MDPs can be effectively identified by a small number of informative state-action pairs. In such cases, inter-segment interactions are minimal, since the presence of a few discriminative $(\hat{r}, \mathbf{s}, \mathbf{a})$ transitions is sufficient for task identification. Moreover, during PDT pretraining, prompt segments are sampled independently

[2] Note that if $\mathcal{P}_i$ contains $M$ expert trajectories, each of length $L$, and each prompt segment has a length $H$, the total number of segments is given by $|\mathcal{P}_i| = M \times (L - H + 1)$.

from the demonstration pool without constraints on order or co-occurrence. As a result, the model is not encouraged to rely on global interactions between segments, but instead learns to attend to individually informative transitions within each segment. This training setup implicitly promotes invariance to inter-segment dependencies and limits the influence of global prompt structure on downstream behavior. See Section E for an attention weight analysis that further supports this interpretation.

**Theorem 4.1.** *Assume that the reward function $G\colon P^J \to \mathbb{R}$ for a prompt $\rho = (\tilde{\tau}_1, \ldots, \tilde{\tau}_J)$ decomposes as the mean of $J$ independent reward models $\phi_j(\tilde{\tau}_j)$*

$$G(\rho) = \frac{1}{J}\sum_{j=1}^{J} \phi_j(\tilde{\tau}_j) + h(\tilde{\tau}_1, \ldots, \tilde{\tau}_J), \tag{5}$$

*and that the interaction term $h$ is uniformly bounded by $|h(\tilde{\tau}_1, \ldots, \tilde{\tau}_J)| \leq \varepsilon, \quad \forall \tilde{\tau}_j \in P$. Let $\rho^* = (\tilde{\tau}_1^*, \ldots, \tilde{\tau}_J^*)$ denote the optimal prompt, and suppose that for each slot $j$, a bandit algorithm guarantees a slot-specific regret*

$$\mathrm{Regret}_j(K) = \sum_{t=1}^{K} \mathbb{E}\left[\phi_j(\tilde{\tau}_j^*) - \phi_j(\tilde{\tau}_{t,j})\right] \tag{6}$$

*over $K$ rounds. Then the cumulative regret after $K$ rounds is bounded as:*

$$\mathrm{Regret}(K) \triangleq \sum_{t=1}^{K} \mathbb{E}\left[G(\rho^*) - G(\rho_t)\right] \leq \frac{1}{J}\sum_{j=1}^{J} \mathrm{Regret}_j(K) + 2K\varepsilon. \tag{7}$$

**Corollary 4.2.** *Under the same assumptions as Theorem 4.1, suppose each slot-specific reward model $\phi_j$ is learned using a standard contextual bandit algorithm (e.g., UCB, Thompson Sampling, $\epsilon$-greedy) over the set of segments $P$, with $|P|$ segments. Without loss of generality, assume that each $\phi_j$ is bounded in $[0, 1]$ either via normalization of returns or bounded regression targets. If the regret for each $\phi_j$ satisfies $\mathrm{Regret}_j(K) = \mathcal{O}\left(\sqrt{K \log |P|}\right)$, then the total regret is bounded by*

$$\mathrm{Regret}(K) = \mathcal{O}\left(\sqrt{K \log |P|} + K\varepsilon\right). \tag{8}$$

Corollary 4.2 shows that the proposed bandit architecture preserves the sublinear regret bound of standard algorithms, while introducing an additional error term that grows linearly with the number of rounds $K$.

### 4.4 Arm Features

Learning reward models $\phi_j : \tilde{\tau} \to \mathbb{R}$ for raw trajectory segments becomes impractical for MDPs with large state spaces, such as those involving pixel-space observations and high-dimensional actions. The key issue lies in the input size of these reward models, which scales as $H \times (|\mathcal{S}| + |\mathcal{A}| + 1)$, growing linearly with $\mathcal{S}, \mathcal{A}$, and segment length $H$. This limitation can be addressed by integrating a feature extractor $\Psi : \tilde{\tau} \to \mathbb{R}^d$ that encodes raw trajectory segments from the MDP modalities into a latent feature space.

We propose to leverage the pre-trained PDT model as a feature extractor by taking the latent representation of prompt tokens as the embedding for the prompt. This approach not only mitigates the scaling issue but also aligns with the inductive biases of the pre-trained PDT, which is expected to encode meaningful, task-relevant information. The reward models then operate on these fixed-size embeddings rather than raw, flattened segments, significantly improving scalability.

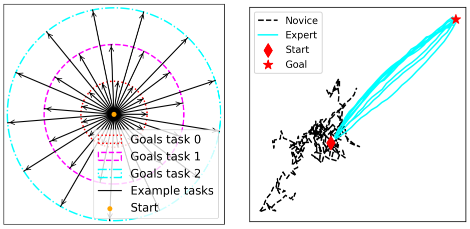
In summary, we introduce a specialized bandit architecture for PDT prompt tuning. By leveraging the structure of the prompt space, we optimize $J$ independent reward models, each corresponding to a segment position in the trajectory prompt, which reduces the search complexity from combinatorial to linear. Additionally, we utilize the pre-trained Transformer backbone to embed prompt segments, allowing the reward model input size to remain fixed even in high-dimensional state and action spaces.

# 5 Experiments

We now present a comprehensive evaluation of our method. We first assess performance across multiple MuJoCo (Todorov et al., 2012) tasks from standard multi-task benchmarks (Yu et al., 2020; Finn et al., 2017). We then analyze prompt-space exploration in a custom 2D environment. Finally, we evaluate the regret behavior of our bandit architecture and compare its scalability to standard algorithms.

## 5.1 Environments, Datasets, Baselines

**Environments**.



(a) Multiple tasks. (b) Trajectories.

Figure 2: `Sparse 2D point` environment: (a) Multiple tasks via parameters $r$ (radius) and $\alpha$ (angle); (b) Expert vs. novice trajectories.

We consider the following environments: (i) `MuJoCo Half Cheetah`: Joint control of a half-cheetah agent. Tasks involve matching varying target velocities. (ii) `MuJoCo Ant`: Joint control of an ant agent. Tasks involve movement along different target directions. (iii) `Meta-world Pick-place`: End-effector control of a simulated Sawyer arm. Tasks require picking objects from varying locations and placing them at distinct target positions. (iv) `Sparse 2D Point`: A planar point agent navigates to a goal position. Tasks are distinguished by varying goal coordinates, distributed on circles with different radii and angles, as depicted in Figure 2a. The agent must issue a `stop` action to end the episode and receive a sparse reward, defined as the negative distance to the goal. A bonus of +10 is awarded for stopping near the target. Crucially, for all settings, the task parameter (target velocity, direction, goal position) is *not* part of the state – it must be inferred from the trajectory prompt. A detailed description of the environments is provided in Appendix B.

**Offline RL dataset.** For each training task $i$, PDT requires a dataset of trajectories $\mathcal{T}_i$ and a dataset of (expert) demonstrations $\mathcal{P}_i$. For the `MuJoCo` tasks, we use datasets and corresponding training and testing tasks from Xu et al. (2022). To generate datasets for the `Sparse 2D point` environment, we employ PPO (Schulman et al., 2017) on 60 tasks with three discrete radii and 20 discrete angles. The first 48 tasks are used for training, and the remaining 12 are reserved for testing generalization. PPO is run for 1M steps on each training task, and the resulting trajectories are stored as $\mathcal{D}_i$. The highest-return trajectories are selected to form $\mathcal{P}_i$. To reduce the computational cost of computing $\arg\max$ over all possible prompt segments, $\mathcal{P}_i$ is limited to 10 randomly sampled high-return trajectories.

**Baselines**. We evaluate our method against the following RL baselines: (i) an optimal policy oracle, either scripted or learned using Conservative Q-Learning (CQL) (Kumar et al., 2020); (ii) a Decision Transformer (DT) trained across multiple tasks without further enhancements; (iii) a vanilla Prompt Decision Transformer (PDT) that samples prompts uniformly at random; (iv) a Gaussian perturbation-based method that performs hill climbing in the prompt space; and (v) the ranking-based prompt tuner from Hu et al. (2023), referred to as "ZORankSGD".

## 5.2 Prompt Tuning Results and Analysis

**Does the proposed method reliably improve the performance of a frozen PDT backbone?** We investigate this question empirically on three MuJoCo environments, where we compare the inference-time performance of a standard PDT, that samples prompts randomly from $\mathcal{P}_i$, to a PDT enhanced by our bandit-based prompt-tuning method. Each method is evaluated over 250 online rollouts on in-distribution training tasks, and we report the mean return from the final 10 rollouts, averaged over three seeds and all tasks. Results are presented in Table 1.

Our bandit-based prompt-tuning method consistently and substantially improves the performance of the frozen PDT backbone, even surpassing the single-task CQL oracle on the `MuJoCo Ant` environment. While PDT performance generally increases with larger prompt sizes (i.e., higher $J$ and $H$), our method yields even greater gains, demonstrating the clear advantage of optimized prompt

| Method | `MuJoCo Half Cheetah` | | `MuJoCo Ant` | | `Meta-world Pick-place` | |
|---|---|---|---|---|---|---|
| | $J = 1, H = 5$ | $J = 2, H = 20$ | $J = 1, H = 5$ | $J = 2, H = 20$ | $J = 1, H = 5$ | $J = 2, H = 2$ |
| CQL oracle | -25.82 ± 12.53 | -25.82 ± 12.53 | 760.54 ± 288.20 | 760.54 ± 288.20 | 535.84 ± 31.02 | 535.84 ± 31.02 |
| PDT, no tuning | -44.75 ± 0.91 | -42.68 ± 1.3 | 694.43 ± 227.04 | 754.22 ± 175.1 | 551.58 ± 26.09 | 535.52 ± 24.86 |
| Hill-climbing | -40.00 ± 26.39 | -29.93 ± 22.84 | 738.56 ± 181.56 | 740.17 ± 189.38 | 555.79 ± 22.72 | 540.15 ± 23.49 |
| ZORankSGD | -43.62 ± 21.14 | -34.77 ± 37.70 | 735.47 ± 180.20 | 731.22 ± 172.48 | 554.26 ± 23.00 | 537.20 ± 25.5 |
| $\epsilon$-greedy$^{\Psi}$ | -42.60 ± 3.77 | -34.12 ± 3.12 | 815.17 ± 182.28 | **819.52 ± 191.39** | 555.35 ± 24.15 | 541.32 ± 22.91 |
| TS$^{\Psi}$ | -38.52 ± 11.21 | -27.62 ± 12.90 | 800.95 ± 184.75 | 791.06 ± 192.75 | **556.87 ± 24.11** | 540.82 ± 22.83 |
| UCB$^{\Psi}$ | -36.55 ± 11.93 | -26.87 ± 14.10 | 751.35 ± 192.25 | 744.55 ± 168.50 | 552.68 ± 24.77 | 538.84 ± 23.14 |
| $\epsilon$-greedy | -76.50 ± 6.68 | -58.76 ± 8.21 | 812.14 ± 176.88 | 816.77 ± 268.42 | 556.22 ± 25.16 | **541.80 ± 23.76** |
| TS | **-33.56 ± 13.48** | **-26.28 ± 10.14** | **835.38 ± 171.25** | 729.65 ± 175.41 | 556.11 ± 24.56 | 541.33 ± 22.79 |
| UCB | -35.72 ± 14.96 | -26.57 ± 10.79 | 732.22 ± 180.25 | 777.29 ± 180.77 | 554.92 ± 26.06 | 538.50 ± 23.64 |

Table 1: Inference time performance (mean ± std) across environments with varying trajectory segments ($J$) and lengths ($H$). Bold values indicate best performance.

| Method | `MuJoCo Half Cheetah` | | `MuJoCo Ant` | | `Meta-world Pick-Place` | |
|---|---|---|---|---|---|---|
| | $J = 1, H = 5$ | $J = 2, H = 20$ | $J = 1, H = 5$ | $J = 2, H = 20$ | $J = 1, H = 5$ | $J = 2, H = 2$ |
| CQL Oracle | -23.8 ± 10.39 | -23.81 ± 10.39 | 508.09 ± 231.90 | 508.09 ± 231.90 | 525.07 ± 60.15 | 525.07 ± 60.15 |
| PDT, no tuning | -64.78 ± 36.91 | -40.95 ± 43.19 | 363.90 ± 105.42 | 360.07 ± 72.36 | 502.8 ± 63.98 | 524.37 ± 39.56 |
| PDT, finetuned | -129.05 ± 65.91 | -39.30 ± 14.66 | 306.29 ± 63.91 | 360.96 ± 138.58 | 495.37 ± 57.87 | 488.17 ± 50.15 |
| Hill-climbing | -53.84 ± 23.56 | -34.39 ± 26.05 | 355.80 ± 135.17 | 344.46 ± 57.21 | **560.92 ± 27.04** | 544.19 ± 28.83 |
| ZORankSGD | -59.85 ± 32.37 | -36.6 ± 19.45 | 383.57 ± 193.35 | 340.68 ± 44.60 | 503.56 ± 66.16 | 538.05 ± 31.46 |
| $\epsilon$-greedy$^{\Psi}$ | -32.61 ± 19.85 | **-23.93 ± 14.14** | 477.24 ± 84.64 | 431.52 ± 43.69 | 531.49 ± 49.86 | 552.11 ± 23.83 |
| TS$^{\Psi}$ | -38.56 ± 21.61 | -31.80 ± 14.20 | 468.76 ± 79.50 | **441.44 ± 80.25** | 549.38 ± 36.12 | 553.12 ± 20.81 |
| UCB$^{\Psi}$ | -42.76 ± 15.77 | -33.27 ± 17.27 | 392.39 ± 51.57 | 346.59 ± 49.50 | 506.43 ± 65.38 | 539.42 ± 31.12 |
| $\epsilon$-greedy | -29.97 ± 21.10 | -25.18 ± 22.17 | **480.85 ± 84.76** | 431.74 ± 44.20 | 530.26 ± 51.61 | 550.06 ± 22.38 |
| TS | **-29.26 ± 21.25** | -34.23 ± 14.51 | 466.11 ± 79.50 | 438.82 ± 125.40 | 551.33 ± 34.52 | **553.34 ± 18.82** |
| UCB | -42.99 ± 15.67 | -35.78 ± 16.83 | 376.68 ± 82.25 | 413.31± 46.31 | 512.25 ± 69.62 | 534.58 ± 35.35 |

Table 2: OOD performance (mean ± std) across environments with varying trajectory segments ($J$) and lengths ($H$). Bold values indicate best performance.

selection over uniform sampling from $\mathcal{P}_i$. When using Transformer-encoded trajectory segments (denoted by $\Psi$), our method achieves comparable performance in terms of return to the unencoded variants, demonstrating that the PDT provides a robust and compact representation for bandit-based prompt tuning. The main benefit of this encoding lies in its fixed-dimensional representation, which makes the approach scalable to high-dimensional settings, as further evidenced by the inference-time results in Appendix 8.

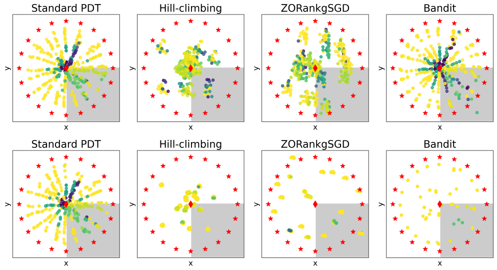
Although hill-climbing and ranking-based prompt-tuning (Hu et al., 2023) can also improve performance, their results are less consistent and typically weaker. We attribute this instability to their reliance on iterative, noise-driven perturbations of initial prompts.

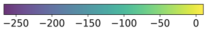
In summary, bandit-based prompt-tuning reliably enhances frozen PDT performance, outperforming existing baselines, particularly when leveraging Transformer-encoded prompt segments.

**Does the bandit-based method improve inference-time performance on OOD tasks?** To assess the benefits of prompt tuning in out-of-distribution (OOD) scenarios, we evaluate the frozen PDT model on OOD tasks from the MuJoCo suite using the same procedure as before. Results are shown in Table 2. We find that our bandit-based method significantly improves PDT's generalization in all tasks, often reaching performance levels comparable to in-distribution tasks (`MuJoCo Half Cheetah`, `Meta-world Pick-Place`), though not always achieving optimal returns. This suggests that the effectiveness of our method extends to OOD tasks, provided the underlying model retains sufficient generalization capability.

In contrast, hill-climbing and ranking-based prompt-tuning (Hu et al., 2024) exhibit reduced robustness in the OOD setting, falling considerably short of the performance gains of our method in `MuJoCo Half Cheetah` and `MuJoCo Ant` environments. We hypothesize that this degradation arises from their reliance on the initial sampled prompt and the increased optimization difficulty faced by the PDT in unfamiliar environments.

Finally, we observe that fine-tuning the PDT backbone solely on target task data for 250 epochs does not reliably improve performance and even degrades performance in some settings. This degradation is hypothesized to stem from the large size of the Transformer backbone, the absence of regularization provided by diverse multi-task training data, and less efficient full-model updates. A similar phenomenon was previously reported by Yuan et al. (2024).



(a) **Top**: First 20 prompts. **Bottom**: Last 20 prompts. Each dot represents the prompt's mean state coordinate, colored according to the achieved return 

. Stars indicate task goals, the diamond marks the initial state, and the shaded region indicates OOD test tasks.

| Method | $J = 1$ | $J = 2$ | $J = 4$ |
|---|---|---|---|
| $\pi^*$ oracle | $10 \pm 0.0$ | $10 \pm 0.0$ | $10 \pm 0.0$ |
| PDT, no tuning | $0.0 \pm 2.1$ | $6.3 \pm 0.8$ | $8.3 \pm 0.6$ |
| Hill-climbing | $5.8 \pm 3.8$ | $7.9 \pm 1.6$ | $6.2 \pm 4.0$ |
| ZORankSGD | $-0.6 \pm 30.9$ | $4.4 \pm 16.7$ | $3.1 \pm 22.1$ |
| $\epsilon$-greedy$^\Psi$ | $9.0 \pm 0.6$ | $9.4 \pm 0.3$ | $9.6 \pm 0.2$ |
| UCB$^\Psi$ | $9.2 \pm 0.5$ | $8.8 \pm 0.9$ | $8.8 \pm 0.9$ |
| TS$^\Psi$ | $9.7 \pm 0.1$ | $9.7 \pm 0.2$ | $9.4 \pm 0.5$ |
| $\epsilon$-greedy | $8.9 \pm 0.5$ | $9.5 \pm 0.3$ | $8.9 \pm 0.7$ |
| UCB | $9.4 \pm 0.5$ | $9.5 \pm 0.3$ | $9.3 \pm 0.4$ |
| TS | $\mathbf{9.9 \pm 0.0}$ | $\mathbf{9.9 \pm 0.0}$ | $\mathbf{9.8 \pm 0.1}$ |

(b) Inference-time performance (mean ± std) in the `Sparse 2D Point` environment. A standard single-task Decision DT achieves $-64.3 \pm 24.1$ average return, underscoring the need for multi-task models like PDT. Results are averaged over training tasks, three seeds, and the final 50 rollouts.

Figure 3: (a) Visualization of prompt selection across tasks in the `Sparse 2D Point` environment. (b) Inference-time performance showing the benefit of prompt tuning over a single-task baseline.

**How do different methods explore the prompt space?** We use the `Sparse 2D Point` environment to qualitatively analyze how different methods explore the prompt space. For interpretability, we consider a PDT with a single prompt segment ($J = 1$), allowing prompts to be visualized via the mean spatial coordinate of their states. Each method is run for 250 rollouts per task, and we visualize the first and last 20 selected prompts in Figure 3a.

The standard PDT samples the prompt space uniformly without refining its selection strategy over time. Hill-climbing gradually discovers high-performing prompts in the local neighborhood of the initial sample by iteratively applying Gaussian noise. ZORankSGD moves prompts toward the task goal, sometimes overshooting it and generating out-of-distribution prompts due to the unconstrained generative approach of Hu et al. (2023).

Our bandit method (with $\epsilon$-greedy exploration) initially samples prompts uniformly, capturing both low- and high-performing examples. Over time, it leverages accumulated reward estimates to avoid low-reward regions and increasingly selects segments near the task goal, which provide more informative cues for task identification.

We also report the performance of all methods on the 2D environment tasks with the largest radius in Table 3b. The standard PDT without prompt tuning falls drastically short of the optimal return, due to the overlap between prompt datasets. The performance of the PDT increases with larger prompts, which implies that excessive random sampling suffices to find informative prompts in this environment. Best prompt-tuning results are again achieved by our bandits compared to the hill-climbing and ranking-based prompt-tuning baselines, with Thompson Sampling (TS) performing slightly better than UCB or $\epsilon$-greedy exploration. We find no considerable difference between using Transformer encoded prompts (marked by $\Psi$) or not, which can be attributed to the simplicity of the 2D environment.

### 5.3 Bandit Regret Analysis

We empirically evaluate the proposed bandit-based method with respect to the interaction bound $\varepsilon$, and compare its performance to standard MAB algorithms that learn directly over the full combinatorial space. This evaluation is conducted on a synthetic task carefully designed to reflect key characteristics of PDT prompt tuning. Specifically, the task involves identifying an optimal vector $\mathbf{x}$ of length $J$, where each dimension takes values in $\{1, \ldots, H\}$. The reward function $R(\mathbf{x})$ is defined only for complete vectors and is bounded within $[0, 1]$, making the contribution of individual components unobservable, mirroring the structure of real-world setting. While the synthetic setup is necessarily simplified, it enables controlled investigation of core assumptions underlying our theoretical analysis, particularly the independence of segment-wise reward estimates. We report cumulative regret after $K = 2000$ rounds for varying values of $J$ and $H$, comparing our structured method to standard

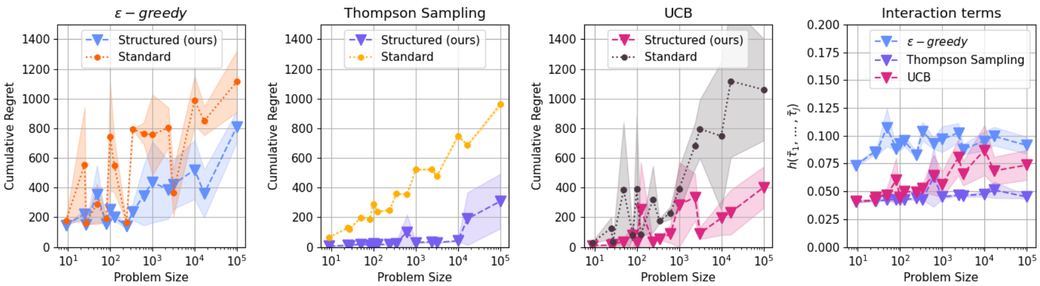


Figure 4: Performance of the structured (ours) and standard MAB methods on a synthetic prompt tuning task. Problem instances are generated by sweeping over $J = \{2, 3, 4, 5\}$ segments and $H = \{3, 5, 7, 10\}$ choices, with problem size reported as $H^J$. Shaded regions indicate one standard deviation around the mean; results are averaged over three random seeds.

$\epsilon$-greedy, UCB, and TS strategies. For our method, we also estimate the neglected interaction term $h(\mathbf{x}_1, \ldots, \mathbf{x}_J) = \left| R(\mathbf{x}) - \frac{1}{J} \sum_j \phi_j(\mathbf{x}_j) \right|$, where $R(\mathbf{x})$ denotes the true reward.

As shown in Figure 4, our method matches the performance of standard, "flat" bandit baselines on small problems and consistently outperforms them as problem size grows. This demonstrates more efficient exploration of the combinatorial space by leveraging its structure. Additionally, the interaction term $h(\mathbf{x}_1, \ldots, \mathbf{x}_J)$ remains low and increases only marginally with problem size, empirically supporting the bounded interaction assumption in Theorem 4.1. These findings reinforce the validity of our method even in settings where interactions between prompt segments exist while global reward must be inferred from segment-level estimates.

## 6 Conclusion

This work introduces a bandit-based prompt-tuning method extending PDT and addressing the limitations of uniform prompt sampling. By leveraging a contextual MAB framework, our approach optimizes the selection of trajectory segments to maximize task performance and enhance adaptation to unseen tasks. Experimental results highlight consistent improvements across diverse tasks, demonstrating the efficacy and robustness of the proposed method in multiple environments. This method not only advances the state of prompt optimization in PDT but also contributes to the broader integration of offline RL and sequence modeling paradigms.

**Limitation and broader impact.** A key limitation is the combinatorial expansion of the search space as the number of demonstrations increases, which renders online exploration impractical. A promising direction is to learn a sampler that pre-selects high-potential segments or clusters of similar prompts to reduce redundancy. While bandit-based tuning enhances OOD performance, pure offline RL struggles with extrapolation. Incorporating meta-learning techniques, such as In-Context RL (Laskin et al., 2022), into PDT pretraining offers a compelling extension of this work.

## References

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, May 2002. ISSN 0885-6125. doi: 10.1023/A:1013689704352. URL `https://doi.org/10.1023/A:1013689704352`.

Nikhil Barhate. Minimal implementation of decision transformer. `https://shorturl.at/YeOpU`, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. InstructZero: Efficient instruction optimization for black-box large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 6503–6518, 2024.

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.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

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

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

Shengchao Hu, Wanru Zhao, Weixiong Lin, Li Shen, Ya Zhang, and Dacheng Tao. Prompt tuning with diffusion for few-shot pre-trained policy generalization. *arXiv preprint arXiv:2411.01168*, 2024.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

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.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Use your instinct: Instruction optimization using neural bandits coupled with transformers. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pp. 7780–7791. PMLR, 2021.

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.

Alec Radford. Improving language understanding by generative pre-training. 2018.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Chengshuai Shi, Kun Yang, Jing Yang, and Cong Shen. Best arm identification for prompt learning under a limited budget. In *ICLR 2024 Workshop on Understanding of Foundation Model*, 2024a.

Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon Shaolei Du, and Huazhe Xu. Unleashing the power of pre-trained language models for offline reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024b.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

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*, pp. 5026–5033. IEEE, 2012.

Zhe Wang, Haozhu Wang, and Yanjun Qi. Hierarchical prompt decision transformer: Improving few-shot policy generalization with global and adaptive. *arXiv preprint arXiv:2412.00979*, 2024a.

Zhi Wang, Li Zhang, Wenhao Wu, Yuanheng Zhu, Dongbin Zhao, and Chunlin Chen. Meta-dt: Offline meta-rl as conditional sequence modeling with world model disentanglement. *Advances in Neural Information Processing Systems*, 37:44845–44870, 2024b.

Zhihui Xie, Zichuan Lin, Deheng Ye, Qiang Fu, Yang Wei, and Shuai Li. Future-conditioned unsupervised pretraining for decision transformer. In *International Conference on Machine Learning*, pp. 38187–38203. PMLR, 2023.

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*, pp. 24631–24645. PMLR, 2022.

Mengdi Xu, Yuchen Lu, Yikang Shen, Shun Zhang, Ding Zhao, and Chuang Gan. Hyper-decision transformer for efficient online policy adaptation. 2023.

Yu Yang and Pan Xu. Pre-trained language models improve the few-shot prompt ability of decision transformer. In *Workshop on Training Agents with Foundation Models at RLC 2024*, 2024.

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*, pp. 1094–1100. PMLR, 2020.

Haoqi Yuan, Yuhui Fu, Feiyang Xie, and Zongqing Lu. Pre-trained multi-goal transformers with prompt optimization for efficient online adaptation. *Advances in Neural Information Processing Systems*, 37:55086–55114, 2024.

Hongling Zheng, Li Shen, Yong Luo, Tongliang Liu, Jialie Shen, and Dacheng Tao. Decomposed prompt decision transformer for efficient unseen task generalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

# Supplementary Material:

## A Prompt-Tuning Algorithm

We provide the full pseudocode of our method. The overall prompt-tuning procedure is outlined in Algorithm 1, prompt selection via the structured bandit is shown in Algorithm 2, and Algorithm 3 details the update of the bandit's internal reward models.

**Algorithm 1** Prompt-Tuning Bandit

```
Input: Pre-trained PDT parameter $\theta$, Simulator $\mathcal{M}_i$, expert demonstrations $\mathcal{P}_i$
Initialize: Bandit parameter $\phi^0$, dataset $\mathcal{B} \leftarrow \{\}$
for bandit steps $k \in K$ do
    Algorithm 2: Select prompt $\rho_k$ for current rollout $k$ using parameters $\phi^k$
    Performance metric $G_i^k = 0$
    for MDP steps $t \in T$ do
        Make PDT input (Equation 3): $\mathbf{x}_k = \rho_k \odot \omega_{L:t}$
        Sample action from PDT: $\mathbf{a}_t \sim \pi^*(\mathbf{x}_k; \theta)$
        Step environment: $r_t, \mathbf{s}_{t+1} \sim \mathcal{M}_i(\mathbf{s}_t, \mathbf{a}_t)$
        Log reward: $G_i^k = G_i^k + r_t$
    end for
    Store data $\mathcal{B} \leftarrow B \cup \langle \rho_k, G_i^k \rangle$
    Algorithm 3: Update $\phi^k$ using $\mathcal{B}$, yielding $\phi^{k+1}$
end for
Return: Final bandit parameter $\phi^K$
```

**Algorithm 2** Prompt Selection

```
Input: Expert demonstrations $\mathcal{P}_i$, bandit parameter $\phi = \langle \phi_1, \cdots, \phi_J \rangle$
Initialize: Prediction matrix $\mathbf{Y} = [\,]$
for segment $\tilde{\tau} \in \mathcal{P}_i$ do
    Predict reward $\hat{\mathbf{y}} = [\phi_1(\tilde{\tau}), \ldots, \phi_J(\tilde{\tau})]$
    Append row to prediction matrix $\begin{bmatrix} \mathbf{Y} \\ \hat{\mathbf{y}} \end{bmatrix}$
end for
Return: $\rho = \mathcal{P}_i[\arg\max(\mathbf{Y}[j,:])]\ \forall j \in \{1, ..., J\}$
```

**Algorithm 3** Bandit Update

```
Input: Bandit dataset $\mathcal{B} = \mathbf{X}, \mathbf{y}$, bandit parameter $\phi = \langle \phi_1, \cdots, \phi_J \rangle$, learning rate $\alpha$
for reward model $j \in J$ do
    Get segments at $j$-th index $\mathbf{X}_j = \mathbf{X}_{[j]}$
    Predict reward $\hat{\mathbf{y}}_j = \phi_j(\mathbf{X}_j)$
    Define $\mathcal{L}(\phi_j) = \text{MSE}(\hat{\mathbf{y}}_j, \mathbf{y})$
    for gradient steps $l = 0, \ldots, L$ do
        $\phi_j^{l+1} = \phi_j^l - \alpha \nabla \mathcal{L}(\phi_j^l)$
    end for
end for
Return: New parameter $\phi = \langle \phi_1^L, \cdots, \phi_J^L \rangle$
```

## B Environment Details

**Sparse 2D point**: This environment involves mixed control of a planar point agent, where the state representation consists of the agent's current 2D coordinates. The agent always begins at $(0, 0)$ with the objective of reaching a (hidden) goal coordinate. It has two continuous actions for movement across the plane and a binary `stop` action. The task requires the agent to navigate to the goal coordinate and appropriately select the `stop` action. A sparse reward is provided upon selecting the `stop` action, proportional to the agent's distance from the goal. When `stop` is selected within close proximity of the goal, the environment provides a reward bonus of 10, which is discounted based on the number of wasted steps. While the goal is not explicitly part of the state, it is implicitly encoded through the reward function, ensuring that each individual task remains a fully observable MDP.

The environment offers a continuous task space, parameterized by the angle and radius, for arbitrary goal locations on the 2D plane. We discretize the task space by using three discrete radii, $(0.9, 1.9, 2.9)$, and 20 discrete angles $(0.0 \cdot \pi, 0.1 \cdot \pi, \dots, 1.9 \cdot \pi)$, instead of sampling continuous task parameters. This is primarily done to separate datasets for different tasks, since PDT requires expert demonstration for each training task. To separate these $3 \cdot 20 = 60$ tasks into the training set $\mathcal{T}^{\text{train}}$ and testing set $\mathcal{T}^{\text{test}}$, all tasks with an angles greater than $1.5 \cdot \pi$ (independently of the radius) are treated as testing task and are not part of the training set. This split yields 48 training tasks and 12 testing tasks. Spatially, the test set is indicated by the shaded area in Figure 3.

**MuJoCo Half Cheetah:** This task involves continuous joint control of a planar half-cheetah agent. The state and action spaces have dimensions 20 and 7, respectively. Tasks vary by target velocity, and the reward is proportional to the deviation from this target. Although the target velocity is not part of the observable state, it is implicitly encoded via the reward, ensuring that each task defines a fully observable MDP. There are 40 tasks in total, 35 are used for training, and 5 are used for testing. For more environment details, see (Duan et al., 2016; Xu et al., 2022).

**MuJoCo Ant:** As in Xu et al. (2022), this task requires continuous control of an ant agent via eight joint actuators. The 27-dimensional state space includes positions and velocities of the agent's body. Tasks involve moving in different directions, with a reward based on velocity along the target heading. There are 50 tasks in total, 45 are used for training, and 5 are used for testing. For more environment details, see Duan et al. (2016); Xu et al. (2022).

**Meta-World Pick-Place:** Following Yu et al. (2020), this task involves Cartesian control of a simulated Sawyer robot's end-effector. The 4-dimensional action space includes 3D position deltas and gripper torque. The 39-dimensional state space encodes gripper and object positions, and quaternions, but excludes the goal location. The benchmark consists of 50 task variations with different placement goals; 5 are held out during training for evaluation.

# C Training Details

All experiments were conducted on an instance equipped with an NVIDIA T4 GPU with 8GB of memory, utilizing the PyTorch library (Paszke et al., 2019). Pre-training the multi-task PDT backbone took approximately 5 hours. Performing 250 online rollouts and prompt tuning took approximately 30 to 120 minutes per task, depending on environment and algorithm.

Details of the hyperparameters for the DT and PDT models are provided in Table 3, while those for the bandit model are listed in Table 4. We used standard values for each hyperparameter instead of performing an expensive hyperparameter optimization. Environment-specific hyperparameters are reported separately in Table 5. The implementations of DT and PDT were adopted from Barhate (2022) with minimal modifications to integrate seamlessly with the CMAB prompt-tuning framework. The MuJoCo experiments were performed by integrating our prompt-tuning bandits into the official PDT repository Xu et al. (2022). For the hill-climbing and ZORankSGD (Hu et al., 2023) baselines, we sample an initial prompt randomly from the expert demonstration dataset for the target task, then apply Gaussian noise and perform hill climbing. To encourage convergence to a final prompt, we linearly anneal the scale of the exploration noise from 1.1 to 0.1 over the course of the 250 online rollouts.

| Hyperparameter | Value |
|---|---|
| Number of transformer blocks | 3 |
| Number of attention heads | 1 |
| Embedding dimension | 128 |
| Transformer activation function | GELU |
| MLP activation function | ReLU |
| MLP hidden layers | 2 |
| MLP width | 128 |
| Batch size (per task) | 8 |
| Learning rate | 1e-4 |
| Learning rate decay weight | 1e-4 |
| Optimizer | AdamW |

Table 3: Hyperparameter values for PDT and DT models.

| Hyperparameter | Value |
|---|---|
| Batch size | All data |
| Learning rate | 1e-3 |
| Evaluation trials | 250 |
| $\epsilon$ in $\epsilon$-greedy | 0.1 |
| $c$ (exploration parameter) in UCB | 3 |
| Reward model type | MLP |
| MLP hidden layers | 2 |
| MLP width | 16 |
| MLP activation function | ReLU |
| Optimizer | Adam |

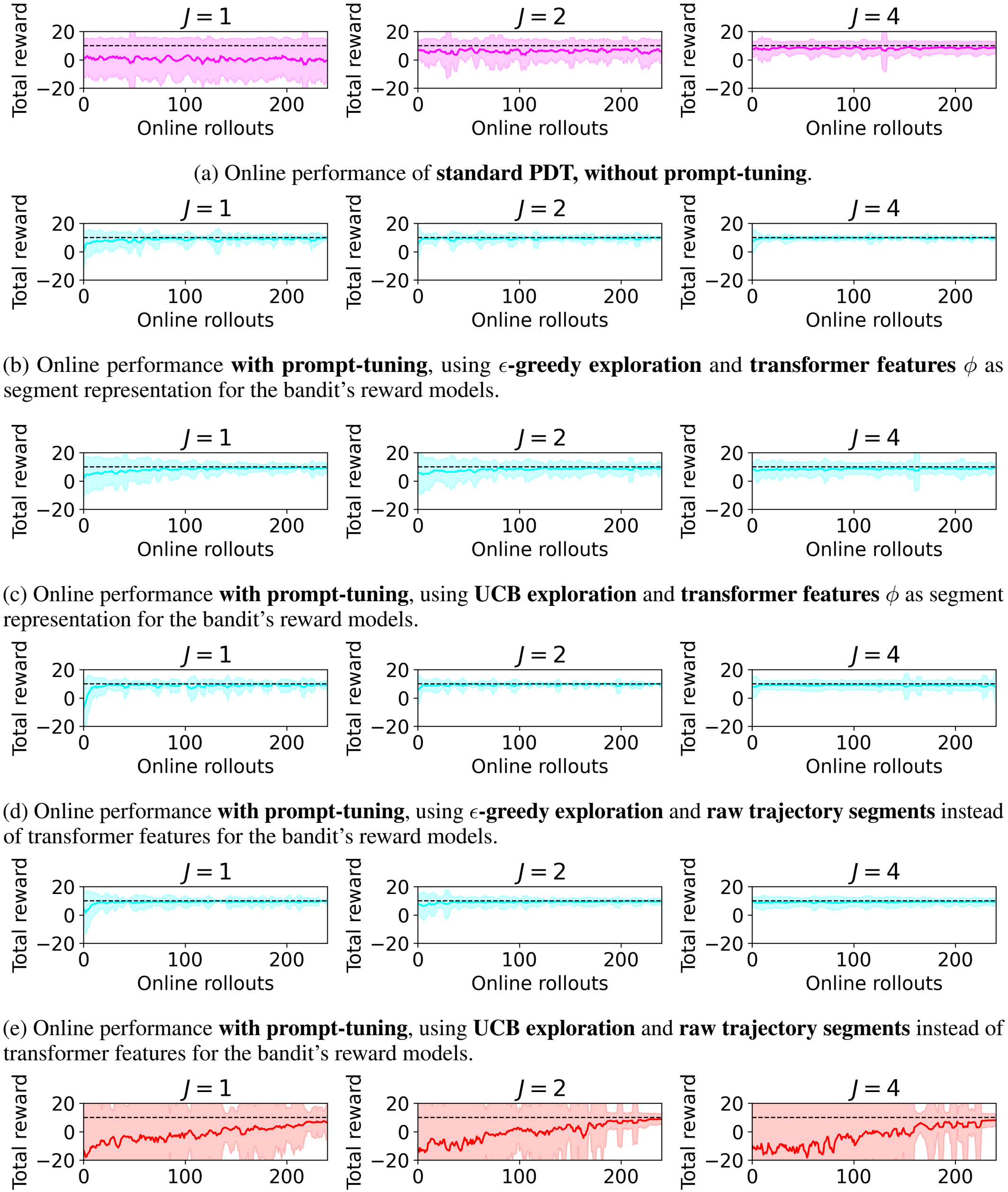
Table 4: Hyperparameters of Contextual Multi-armed Bandit

| Environment | Target Return | Prompt Length $(J \times H)$ | Context Length |
|---|---|---|---|
| 2D-Point-Sparse | 10 | $(1 \times 3), (2 \times 3), (4 \times 3)$ | 5 |
| Cheetah-vel | 0 | $(1 \times 5), (2 \times 20)$ | 20 |

Table 5: Environment-specific hyperparameters of DT and PDT.

# D Additional Results

## D.1 Online Rollouts in `Sparse 2D point`



(a) Online performance of **standard PDT, without prompt-tuning**.

(b) Online performance **with prompt-tuning**, using **$\epsilon$-greedy exploration** and **transformer features** $\phi$ as segment representation for the bandit's reward models.

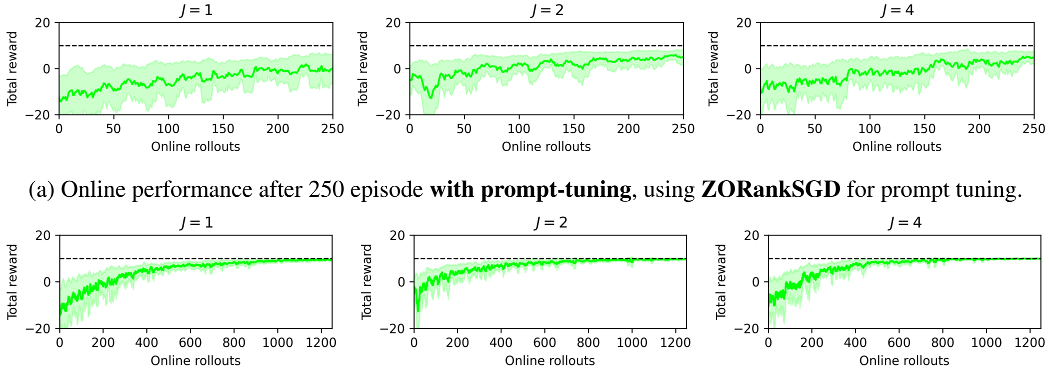
(c) Online performance **with prompt-tuning**, using **UCB exploration** and **transformer features** $\phi$ as segment representation for the bandit's reward models.

(d) Online performance **with prompt-tuning**, using **$\epsilon$-greedy exploration** and **raw trajectory segments** instead of transformer features for the bandit's reward models.

(e) Online performance **with prompt-tuning**, using **UCB exploration** and **raw trajectory segments** instead of transformer features for the bandit's reward models.

(f) Online performance **with prompt-tuning**, using **Gaussian noise** and **hill climbing** to optimize prompt selection.

Figure 5: Plots of data used in Table 3b. The dashed line marks the optimal return at +10, the shaded area corresponds to 1 standard deviation around the mean.



(a) Online performance after 250 episode **with prompt-tuning**, using **ZORankSGD** for prompt tuning.

(b) Online performance after 1500 episodes **with prompt-tuning**, using **ZORankSGD** for prompt tuning.

Figure 6: Plots of ZORankSGD prompt-tuning as reported in Table 3b. The dashed line marks the optimal return at +10. The shaded area corresponds to one standard deviation around the mean. ZORankSGD requires the online evaluation of $m = 5$ prompt perturbations to compute one prompt update. Results using a budget of 250 online rollouts (and 50 prompt improvement steps) are shown in subfig (a). Results using 1250 online rollouts (and 250 prompt improvements) are shown in subfig (b).

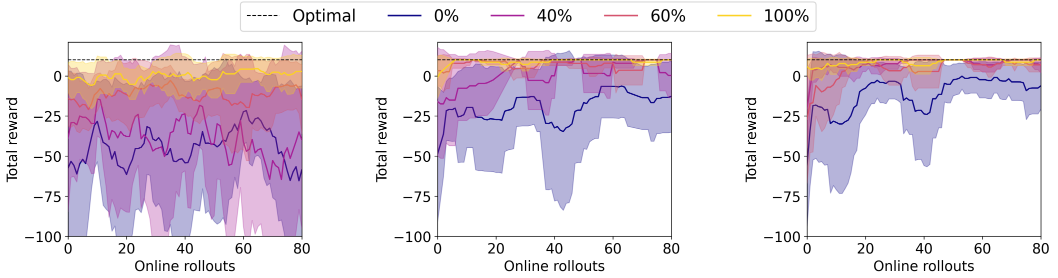
### D.2 Prompt Quality Experiment for the `Sparse 2D point` Environment

**Can prompt-tuning exploit non-expert datasets?** While the PDT model with a random prompt selection strategy relies on access to expert demonstrations, our method demonstrates greater robustness to the quality of demonstration data. To examine this, we generate additional prompt datasets $\{\mathcal{P}_i^{\%j}\}, j \in \{0, 10, \dots, 100\}$, which combine $j\%$ of expert data along with novice demonstrations, selected from trajectories in the bottom 5th percentile of task returns (see Figure 2b).

When sampling prompts from these mixed datasets, the bandit-based prompt optimization significantly enhances the PDT model's robustness and improves its performance, as shown in Table 6. This approach reduces reliance on pure expert demonstrations by learning to identify the optimal prompt from any arbitrary mixture dataset.

| Expert percentage $j\%$ | No tuning | $\epsilon$-greedy | UCB |
|---|---|---|---|
| 0% | -39.4 ± 16.8 | -12.1 ± 19.4 | -3.8 ± 10.0 |
| 10% | -40.8 ± 14.1 | 4.4 ± 3.4 | 7.3 ± 1.9 |
| 20% | -49.6 ± 32.8 | 5.5 ± 3.4 | 9.4 ± 0.5 |
| 30% | -50.1 ± 27.4 | 6.5 ± 1.3 | **9.8 ± 0.3** |
| 40% | -41.4 ± 26.4 | 2.1 ± 6.4 | 8.6 ± 2.0 |
| 50% | -12.9 ± 4.0 | 5.1 ± 4.4 | 8.7 ± 0.8 |
| 60% | -14.1 ± 5.0 | 7.6 ± 1.5 | 8.5 ± 1.2 |
| 70% | -28.6 ± 17.0 | 8.4 ± 0.9 | 7.7 ± 1.3 |
| 80% | -12.3 ± 11.1 | 8.6 ± 1.0 | **9.8 ± 0.3** |
| 90% | 0.9 ± 0.5 | 8.6 ± 0.9 | 8.6 ± 1.0 |
| 100% | **0.7 ± 1.7** | **9.7 ± 0.4** | 9.7 ± 0.4 |

Table 6: Without prompt-tuning, PDT's performance deteriorates with the percentage of expert trajectories in $\mathcal{P}_i$. Our prompt-tuning method is robust with respect to the percentage of expert data and achieves near-optimal performance with as little as 10% expert demonstrations. Results are averaged over three seeds for a single training task.



(a) Baseline PDT, no tuning (b) $\epsilon$-greedy prompt tuning (c) UCB prompt tuning

Figure 7: Prompt quality experiment. Color indicates percentage of expert demonstrations in $\mathcal{P}_i$. Without prompt-tuning, PDT's performance degrades and is roughly proportional to the percentage of expert demonstrations in the dataset. Our prompt-tuning method quickly recovers and converges to near-optimal performance by finding the high-performance prompts in the mixture dataset. The shaded region denotes 1 standard deviation around the mean, averaged over three random seeds for a single training task.

# E Segment Independence Attention Analysis

The key assumption based on which we design our structured bandit architecture is that prompt segments contribute independently and additively to prompt informativeness. This assumption implies that PDT can identify the target tasks by attending to key $(\hat{r}, \mathbf{s}, \mathbf{a})$-pairs in the prompt, as opposed to deriving task identity from the interaction effects between prompt segments, or by attending to the prompt globally. This assumption is crucial, because it allows us to decompose the prompt tuning problem into a bandit that maintains $J$ reward models, one for each prompt segment, which results in a reduction of the size of the exploration problem from combinatorial in $J$ to linear in $J$. The basis for the assumption is that a) many MDPs can be characterized by their optimal state-action marginal (i.e., by key $(\mathbf{s}, \mathbf{a})$-pairs) and b) that PDT's pre-training paradigm discourages reliance on segment interaction or arrangement, since prompt segments are sampled uniformly at random during pre-training, not by enforcing order or other constraints between segments. This training setup implicitly encourages invariance to permutation and inter-segment dependencies, and thus, we believe that this limits the influence of global prompt structure on the model's behavior.

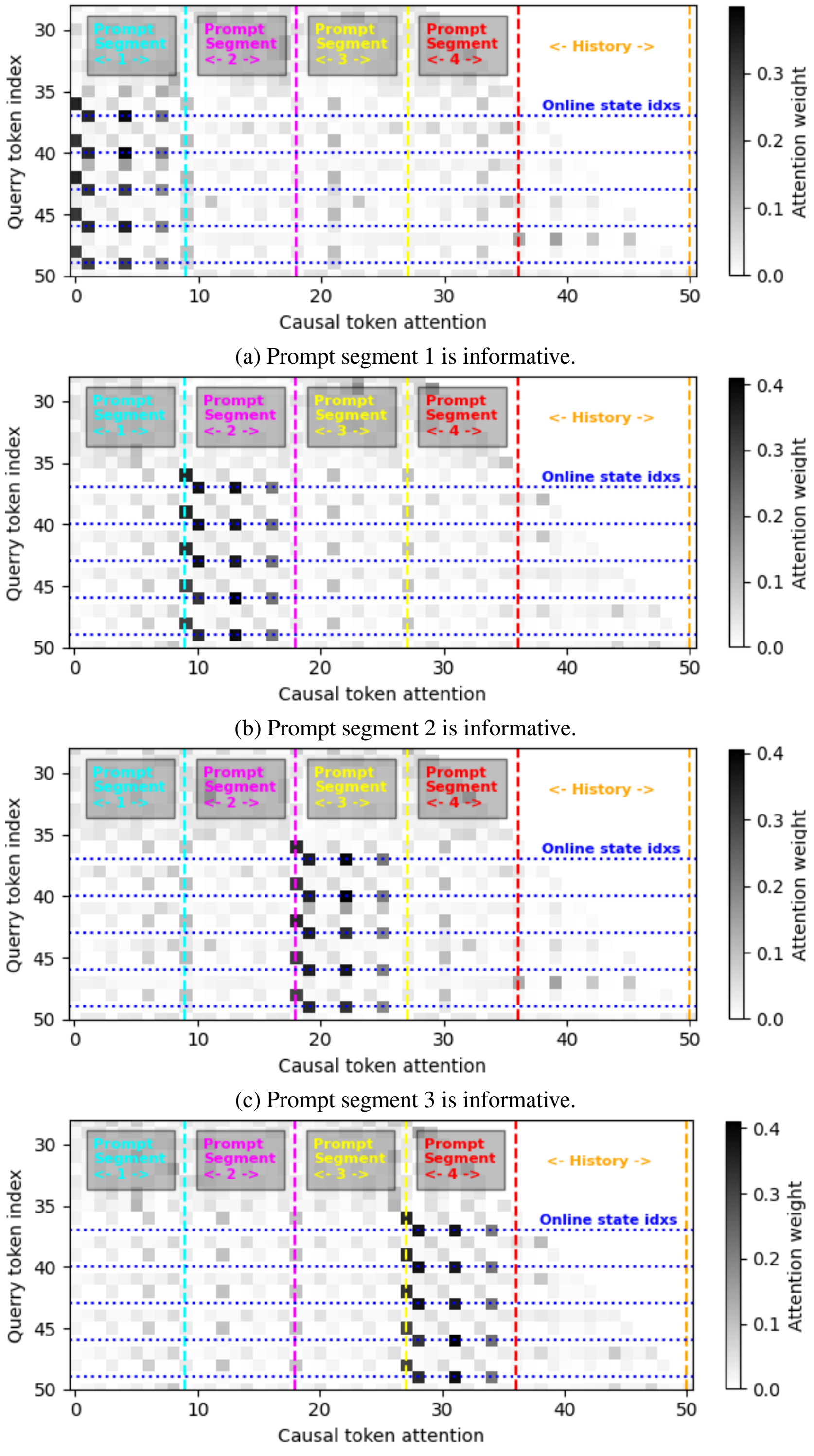
To provide additional support for this interpretation, we conduct an analysis of PDT's attention weight on our 2D environment. Here, for each task, we construct prompts that contain exactly one segment that is highly informative for task identification, based on our domain knowledge, while the remaining $J - 1$ segments carry little information for discriminating between tasks. We then performed rollouts using different permutations of these segments as prompts. The performance of PDT remained robust under those permutations with $6.27 \pm 0.34$ return on average over tasks, prompts, and model configurations ($J = 2, H = 3$ and $J = 4, H = 3$), which implies prompt segment permutation invariance in PDT. We visualize a representative PDT attention mask for different prompt permutations in Figure 8 to confirm qualitatively that PDT indeed identifies tasks by attending mostly to the individual, informative prompt segment.

Furthermore, we quantify attention to each prompt segment by computing the mean of the token attention score of tokens in each prompt segment. We then sum the per-timestep mean segment attention scores, and find that in 99% of rollouts conducted during this analysis, PDT assigned the highest attention to the informative segment. Additionally, we manually set attention weights for specific prompt segments to zero, and find that PDT's performance remains stable when uninformative segments are masked out, while it degrades drastically when the informative segment is removed, as per Table 7.

| Full prompt | Only informative segment | Only uninformative segments |
|---|---|---|
| $6.27 \pm 0.34$ | $5.88 \pm 0.23$ | $-34.06 \pm 4.49$ |

Table 7: PDT performance with different attention scopes.

These results suggest that PDT behavior is largely driven by the most informative segment, and that inter-segment interactions are negligible in practice, which supports and justifies our assumption and bandit architecture. However, we also observed instances where PDT attended to all prompt segments rather uniformly, despite $J - 1$ segments not being useful for task discrimination as per our domain knowledge. We hypothesize that these instances were due to slight overfitting to the prompt dataset, and see the design of regularization that prevents PDT from exploiting such nuances in the prompt dataset as important future work.



(a) Prompt segment 1 is informative.

(b) Prompt segment 2 is informative.

(c) Prompt segment 3 is informative.

(d) Prompt segment 4 is informative.

Figure 8: Attention weights from the first transformer block (after softmax), darker values indicate higher attention scores. The first 20 rows have been cropped to improve readability. The blue horizontal lines indicate the indices of state tokens at times $t - H, t - H + 1, \ldots, t$ from which the action at the corresponds $t$ is predicted. It can be seen that PDT attends mostly to the informative prompt token, independently of it's position or neighboring segments.

| Method | `MuJoCo Ant` ($J=1, H=5$) | `MuJoCo Ant` ($J=2$, H=20) |
|---|---|---|
| PDT (full-model fine-tuning) | $\sim$ 28m | $\sim$ 38m |
| Hill climbing | $\sim$ 16m | $\sim$ 24m |
| ZORankSGD | $\sim$ 52m | $\sim$ 120m |
| TS$^{\Psi}$ | $\sim$ 28m | $\sim$ 34m |
| TS | $\sim$ 33m | $\sim$ 172m |

Table 8: Wall-clock inference times with different methods.

## F Wall-clock inference time

Table 8 reports representative wall-clock inference times for different methods on the `MuJoCo Ant` environment. Comparing our bandit architecture with PDT-encoded prompt representations (denoted by $^{\Psi}$) to its raw unencoded counterpart highlights the efficiency benefits of using the PDT as a feature extractor. When operating on PDT-encoded prompts, the Thompson Sampling bandit maintains low inference times even for large prompts ($J = 2, H = 20$), since the PDT-encoded representation has a fixed dimension $\mathbb{R}^d$ determined by the PDT's token embedding space, independent of the prompt length. In contrast, raw unencoded prompts lead to input sizes that scale as $H \times (|\mathcal{S}| + |\mathcal{A}| + 1)$ for each segment, substantially increasing inference time due to the higher cost of updating the reward models.

## G Proof of Theorem 4.1

**Theorem.** *Assume that the reward function $G\colon P^J \to \mathbb{R}$ for a prompt $\rho = (\tilde{\tau}_1, \ldots, \tilde{\tau}_J)$ decomposes as the mean of $J$ independent reward models $\phi_j(\tilde{\tau}_j)$:*

$$G(\rho) = \frac{1}{J}\sum_{j=1}^{J} \phi_j(\tilde{\tau}_j) + h(\tilde{\tau}_1, \ldots, \tilde{\tau}_J),$$

*and that the interaction term is uniformly bounded by $|h(\tilde{\tau}_1, \ldots, \tilde{\tau}_J)| \le \varepsilon, \quad \forall \tilde{\tau}_j \in P$. Let $\rho^* = (\tilde{\tau}_1^*, \ldots, \tilde{\tau}_J^*)$ denote the optimal prompt, and suppose that for each slot $j$, a bandit algorithm guarantees a slot-specific regret*

$$\mathrm{Regret}_j(K) = \sum_{t=1}^{K} \mathbb{E}\left[\phi_j(\tilde{\tau}_j^*) - \phi_j(\tilde{\tau}_{t,j})\right]$$

*over $K$ rounds. Then the cumulative regret after $K$ rounds is bounded as:*

$$\mathrm{Regret}(K) \triangleq \sum_{t=1}^{K} \mathbb{E}\left[G(\rho^*) - G(\rho_t)\right] \le \frac{1}{J}\sum_{j=1}^{J} \mathrm{Regret}_j(K) + 2K\varepsilon.$$

*Proof.* We begin with the decomposition of the reward function. The instantaneous regret at round $t$ is

$$G(\rho^*) - G(\rho_t) = \left(\frac{1}{J}\sum_{j=1}^{J} \phi_j(\tau_j^*) + h(\tau_1^*, \ldots, \tau_J^*)\right) - \left(\frac{1}{J}\sum_{j=1}^{J} \phi_j(\tau_{t,j}) + h(\tau_{t,1}, \ldots, \tau_{t,J})\right).$$

Rearranging terms yields:

$$G(\rho^*) - G(\rho_t) = \frac{1}{J}\sum_{j=1}^{J}\Big(\phi_j(\tau_j^*) - \phi_j(\tau_{t,j})\Big) + \Big[h(\tau_1^*, \ldots, \tau_J^*) - h(\tau_{t,1}, \ldots, \tau_{t,J})\Big].$$

By the bounded interaction assumption, we have:

$$\Big|h(\tau_1^*, \ldots, \tau_J^*) - h(\tau_{t,1}, \ldots, \tau_{t,J})\Big| \le 2\varepsilon.$$

Thus, taking expectations and summing over $t = 1$ to $K$, the cumulative expected regret satisfies

$$\begin{aligned}\text{Regret}(K) &= \sum_{t=1}^{K} \mathbb{E}\left[G(\rho^*) - G(\rho_t)\right] \\ &\leq \sum_{t=1}^{K} \left[ \frac{1}{J} \sum_{j=1}^{J} \mathbb{E}\Big( \phi_j(\tau_j^*) - \phi_j(\tau_{t,j}) \Big) + 2\varepsilon \right] \\ &= \frac{1}{J} \sum_{j=1}^{J} \sum_{t=1}^{K} \mathbb{E}\left[ \phi_j(\tau_j^*) - \phi_j(\tau_{t,j}) \right] + 2K\varepsilon.\end{aligned}$$

By definition, the slot-specific cumulative regret is given by

$$\text{Regret}_j(K) = \sum_{t=1}^{K} \mathbb{E}\left[ \phi_j(\tau_j^*) - \phi_j(\tau_{t,j}) \right],$$

so we obtain the final bound:

$$\text{Regret}(K) \leq \frac{1}{J} \sum_{j=1}^{J} \text{Regret}_j(K) + 2K\varepsilon.$$

□

## 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: The contributions outlined in the abstract and introduction are consistent with the prompt-tuning bandit method Section 4 the evaluations in Section 5.

   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 discuss the main limitations of our method, and potential remedies, in Section 6. Potential limitations in the evaluation procedure are discussed in Section 5.

   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: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Our theoretical result is the regret bound of our bandit architecture in Theorem 4.1. We provide a proof for this regret bound in Section G in the Appendix and state all assumptions.

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: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the complete pseudocode of our method in Section A in the Appendix. We provide a detailed description of our custom `Sparse 2D point` environment in Section B. Training details and hyperparameters are stated in Section C in the Appendix.

   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: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide our codebase with instructions for reproducing each experiment at `https://github.com/king/pdt-bandits`.

   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: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We state the split between in-distribution training tasks and out-of-distribution testing tasks for each environment in Section B in the Appendix. Hyperparameters are stated in Section C.

   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 report all results in terms terms of the mean $\pm$ one standard deviation, either in tabular form or as confidence intervals in plots.

   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 report the hardware specification and approximate execution time for our experiments in Section C.

   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: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and adhere to the NeurIPS Code of Ethics. We experiment mainly on publically available meta RL datasets (Yu et al., 2020; Duan et al., 2016), additional experiments were conducted on a simulated 2D environment and on a synthetic bandit benchmark.

   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: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: We are contributing a foundational method to improve the performance of PDT models and can therefore not estimate the true societal impact. However, we don't see any application or deployment with negative societal impact resulting directly from this work.

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: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This work does not release data or models with high risk of misuse.

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 for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All used assets (public datasets, algorithm implementations) are cited appropriately.

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. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The source code released along with our paper is properly documented and contains the license terms.

    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: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work did not involve crowdsourcing.

    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. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work did not involve human or animal subjects.

    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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This work does not involve any LLM components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.