Online Finetuning Decision Transformers with Pure RL Gradients

Junkai Luo
University of California, Riverside
junkail@ucr.edu

Yinglun Zhu[†]
University of California, Riverside
yzhu@ucr.edu

Abstract

Decision Transformers (DTs) have emerged as a powerful framework for sequential decision making by formulating offline reinforcement learning (RL) as a sequence modeling problem. However, extending DTs to online settings with *pure RL gradients* remains largely unexplored, as existing approaches continue to rely heavily on supervised sequence-modeling objectives during online finetuning. We identify hindsight return relabeling—a standard component in online DTs—as a critical obstacle to RL-based finetuning: while beneficial for supervised learning, it is fundamentally incompatible with importance sampling-based RL algorithms such as GRPO, leading to unstable training. Building on this insight, we propose new algorithms that enable online finetuning of Decision Transformers using pure reinforcement learning gradients. We adapt GRPO to DTs and introduce several key modifications, including sub-trajectory optimization for improved credit assignment, sequence-level likelihood objectives for enhanced stability and efficiency, and active sampling to encourage exploration in uncertain regions. Through extensive experiments, we demonstrate that our methods outperform existing online DT baselines and achieve new state-of-the-art performance across multiple benchmarks, highlighting the effectiveness of pure-RL-based online finetuning for Decision Transformers.

1 Introduction

The transformer architecture (Vaswani et al., 2017) lies at the core of modern foundation models. Large language models (LLMs), in particular, have demonstrated remarkable generalization and reasoning capabilities through a simple yet powerful paradigm: large-scale pretraining followed by supervised and reinforcement learning (RL)-based finetuning (Radford et al., 2018; Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023; Comanici et al., 2025). Inspired by this success, the *Decision Transformer* (DT, Chen et al., 2021) introduces transformers to sequential decision making by reframing classical RL as a conditional sequence modeling problem. As an *offline* RL method, DTs are trained with a supervised objective on pre-collected trajectories, effectively performing imitation learning (Hussein et al., 2017) while conditioning on return-to-go (RTG) tokens.

The *Online Decision Transformer* (ODT, Zheng et al., 2022) extends DTs to the online setting by enabling finetuning after offline pretraining. During online finetuning, ODT collects new trajectories and applies *hindsight return relabeling*, replacing intended RTGs with realized returns to align trajectories with their outcomes, mirroring the offline training procedure. Subsequent work further augments ODT with TD3 (Fujimoto et al., 2018) gradients, resulting in ODT+TD3 (Yan et al., 2024). Despite these advances, existing approaches to online finetuning of DTs remain dominated by supervised objectives: ODT relies exclusively on supervised loss, while ODT+TD3 assigns only a small weight to RL gradients.

In contrast, recent progress in LLM finetuning shows that *pure RL* methods—such as Group Relative Policy Optimization (GRPO, Shao et al., 2024)—can substantially improve reasoning and alignment (Guo et al., 2025; Team, 2025). This contrast naturally raises an important question:

Can Decision Transformers be finetuned online using pure RL gradients?

To investigate this question, we revisit the training paradigm of existing online DT methods and identify a fundamental limitation. We show that *hindsight return relabeling* (Zheng et al., 2022), while beneficial for

[†]Project lead and corresponding author.

supervised learning, is fundamentally incompatible with on-policy RL algorithms that rely on importance sampling. Relabeling RTGs used during rollout with returns observed afterward introduces a mismatch in importance ratios, leading to unstable optimization and degraded performance (Fig. 1). Removing hindsight return relabeling is therefore a necessary first step toward applying importance sampling-based algorithms, such as GRPO and PPO, to online finetuning of Decision Transformers.

Building on this insight, we develop a new framework for online finetuning of DTs using *pure RL gradients*. Specifically, we adapt GRPO to Decision Transformers (GRPO-DT) and introduce three key modifications: (i) a sub-trajectory-based optimization objective that enables fine-grained credit assignment, supported by either environment resetting (Mhammedi et al., 2024; Kazemnejad et al., 2025) or an auxiliary Q-function; (ii) a sequence-level importance ratio that improves training stability and efficiency; and (iii) an active sampling strategy that prioritizes uncertain states to enhance exploration. Together, these components enable RL-only finetuning of pretrained DTs and yield new state-of-the-art performance across multiple benchmarks. In addition, we adapt Proximal Policy Optimization (PPO, Schulman et al., 2017) to Decision Transformers (PPO-DT), achieving competitive results where prior PPO-based approaches failed (Yan et al., 2024).

Contributions. Our main contributions are summarized as follows:

(i) We identify hindsight return relabeling—a standard component in existing online DT methods—as a critical obstacle to reinforcement learning-based finetuning: while effective for supervised objectives, it is incompatible with importance sampling-based policy gradient algorithms such as GRPO and PPO.

(ii) We propose GRPO-DT, an adaptation of GRPO for Decision Transformers that enables *pure-RL online finetuning* by integrating sub-trajectory optimization for improved credit assignment, sequence-level importance ratios for enhanced stability and efficiency, and active state sampling for better exploration. These modifications are general and may be of independent interest beyond DTs.

(iii) We conduct extensive experiments and demonstrate that pure RL finetuning—via GRPO-DT and PPO-DT—outperforms existing online DT baselines and achieves new state-of-the-art performance across multiple benchmarks.

Paper organization. The remainder of the paper is organized as follows. Section 2 reviews background and preliminaries. Section 3 presents our methods. Section 4 reports experimental results. Section 5 discusses related work, and Section 6 concludes the paper.

2 Preliminaries

Markov Decision Process. We formulate the reinforcement learning environment as a *Markov Decision Process* (MDP), defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s_{h+1} \mid s_h, a_h)$ is the probability transition function, $R(s_h, a_h)$ is the reward function. At each timestep $h = 1, \dots, H$, the agent observes a state $s_h \in \mathcal{S}$, selects an action $a_h \in \mathcal{A}$ according to a policy $\pi(a_h \mid s_h)$, transitions to the next state $s_{h+1} \sim P(\cdot \mid s_h, a_h)$, and receives a reward $r_h = R(s_h, a_h)$. The goal is to learn a policy π that maximizes the expected cumulative reward $\mathbb{E}_{\mathcal{M},\pi}\left[\sum_{h=1}^{H} r_h\right]$.

Decision Transformer (DT). Decision Transformer (Chen et al., 2021) represents a powerful paradigm for *offline* reinforcement learning, formulating decision making as a sequence modeling problem with pre-collected training trajectories. A DT trajectory consists of three types of tokens: return-to-go (RTG), state, and action, where the RTG $g_h := \sum_{\bar{h}=h}^{H} r_{\bar{h}}$ represents the sum of future reward from step h onward. DT leverages the transformer architecture (Vaswani et al., 2017) to autoregressively learn a policy from pre-collected trajectories. DTs are trained on fixed-length trajectory segments (Chen et al., 2021): let m denotes the context length, the DT learns to generate the next action a_h based on past interactions $((g, s, a)_{h-m+1:h-1}, g_h, s_h) := (g_{h-m+1}, s_{h-m+1}, a_{h-m+1}, \dots, g_h, s_h)$ of context length m. The model is trained via supervised learning by minimizing the mean squared error (MSE) between the predicted action $\pi(a_h \mid ((g, s, a)_{h-m+1:h-1}, g_h, s_h))$ and the ground-truth action a_h. When the context is clear, we use the

shorthand $\pi(a_h \mid s_h, g_h) := \pi(a_h \mid ((g, s, a)_{h-m+1:h-1}, g_h, s_h))$. During evaluation and deployment, the learner specifies a desired initial RTG, since the ground-truth future RTG isn't known in advance, and leverages the DT to autoregressively generate the next action and interact with the environment.

Online finetuning of Decision Transformers. Online Decision Transformer (ODT, Zheng et al., 2022) extends offline DT to the *online* setting by first conducting offline pretraining and then online finetuning with interactively collected data. The offline pretraining stage largely follows the standard DT training procedure. In the online finetuning stage, the DT is deployed into the environment with a desired initial RTG g_{online} to collect new trajectories that gradually replacing old trajectories stored in the replay buffer; the replay buffer is initialized with the offline trajectories. For online collected trajectories, ODT applies *hindsight return relabeling* (Andrychowicz et al., 2017; Ghosh et al., 2019) to relabel the RTG tokens based on the actual achieved RTG g_{actual}. ODT adopts a stochastic Gaussian policy to account for exploration in the online setting. However, similar to offline DT, ODT still learns via a supervised learning objective of minimizing the negative log-likelihood loss.

While ODT improves model performance during online finetuning, recently, Yan et al. (2024) pointed out that ODT fails in settings with medium or low-quality offline data due to the sole use of the supervised learning objective. The supervised learning objective learns $\frac{\partial a}{\partial \mathsf{RTG}}$, i.e., how action changes as the target RTG varies, since DT models actions conditioned on RTGs. However, what actually drives online policy improvement is $\frac{\partial \mathsf{RTG}}{\partial a}$, i.e., how RTG responds to action adjustments, especially when offline pretraining data is not of high-quality; see section 3.1 in Yan et al. (2024) for more details. To enable better online improvements, Yan et al. (2024) propose ODT+TD3, which augments the supervised ODT objective (with hindsight return relabeling) with RL gradients from TD3 (Fujimoto et al., 2018; Fujimoto and Gu, 2021) to guide online exploration and adaptation. However, ODT+TD3 still prioritizes the supervised ODT loss, assigning only a small weight to the RL gradients during optimization.

Group Relative Policy Optimization (GRPO). GRPO is a reinforcement learning algorithm initially proposed for large language models (LLMs) finetuning (Shao et al., 2024; Guo et al., 2025). It simplifies Proximal Policy Optimization (PPO, Schulman et al., 2017) by removing the need for a value model to estimate the advantages. Instead, GRPO samples multiple responses per question and uses their average reward as the baseline for advantage calculation. Specifically, for each query $q \sim \Delta(Q)$ sampled from the question distribution $\Delta(Q)$, the model generates a group of G responses $\{o_1, \cdots, o_G\}$ based on policy $\pi_{\theta_{\mathsf{old}}}$. A reward r_i is computed for each response o_i, usually with the help of a reward model. GRPO optimizes the policy model by maximizing the following objective:

$$J_{\mathsf{GRPO}}(\theta) = \mathbb{E}_{q\sim\Delta(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\mathsf{old}}}(\cdot|q)} \\ \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{h=1}^{|o_i|}\min\left(w_{i,h}(\theta)\widehat{A}_i,\ \mathsf{clip}\left(w_{i,h}(\theta), 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_i\right) - \beta D_{\mathsf{KL}}\left(\pi_\theta \parallel \pi_{\theta_{\mathsf{ref}}}\right), \tag{1}$$

where $\widehat{A}_i = \frac{r_i - \mathrm{mean}(\{r_1, r_2, \cdots, r_G\})}{\mathrm{std}(\{r_1, r_2, \cdots, r_G\})}$ denotes the advantage of the i-th rollout, $w_{i,h}(\theta) = \frac{\pi_\theta(o_{i,h}|q, o_{i,<h})}{\pi_{\theta_{\mathsf{old}}}(o_{i,h}|q, o_{i,<h})}$ denotes the importance sampling ratio, and $D_{\mathsf{KL}}(\pi_\theta \parallel \pi_{\theta_{\mathsf{ref}}})$ denotes the KL penalty that prevents large deviations from a reference policy $\pi_{\theta_{\mathsf{ref}}}$. In our implementation, we employ a slow update schedule for the reference policy $\pi_{\theta_{\mathsf{ref}}}$, updating it once every four updates of π_θ.

3 Methods

In Section 3.1, we first analyze a key limitation of adapting Decision Transformers to the online setting using importance-sampling-based reinforcement learning algorithms, and discuss how this limitation can be addressed. Building on these insights, Section 3.2 presents our adaptation of GRPO to Decision Transformers, incorporating several key modifications. We introduce additional extensions in Section 3.3.

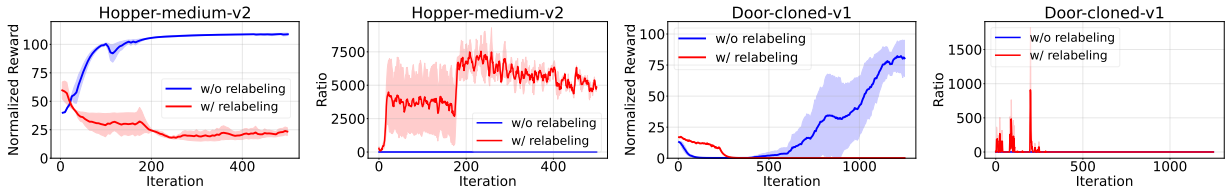


Figure 1: Comparison of GRPO-DT with and without hindsight return relabeling when applied to sampled sub-trajectories. Hindsight return relabeling causes the importance-sampling ratio $\frac{\pi_\theta(a|s,g)}{\pi_{\theta_{\text{old}}}(a|s,g)}$ to become highly unstable, which in turn degrades performance.

3.1 Removing Hindsight Return Relabeling

When deploying DTs, the learner must specify a desired initial RTG, since the ground-truth future RTG is unknown in advance. In Online Decision Transformer (ODT), the learner typically sets a relatively high target RTG g_{online} during rollout to encourage optimistic exploration. During training, a key component of ODT—known as *hindsight return relabeling*—replaces the RTG tokens based on the actual achieved RTG g_{actual} (Zheng et al., 2022).

When augmenting the supervised ODT objective with RL gradients from TD3, ODT+TD3 (Yan et al., 2024) also adopt the hindsight return relabeling step. Yan et al. (2024) further attempt to perform online finetuning of DTs using PPO (Schulman et al., 2017)—an importance sampling-based RL algorithms—but find that PPO gradients lead to poor performance, ultimately reverting to TD3 gradients instead.

While hindsight return relabeling works well under supervised learning objectives (see Fig. 5.4 in Zheng et al. (2022)), we find it incompatible with importance sampling-based RL gradients that rely on the ratio $\frac{\pi_\theta(a|s,g)}{\pi_{\theta_{\text{old}}}(a|s,g)}$, as in PPO and GRPO. The issue arises from a mismatch in the conditioning variable g: the learner conditions rollouts on a high RTG g_{online} for optimistic exploration, yet the achieved RTG g_{actual} is often much lower. If hindsight return relabeling is applied, actions sampled from $\pi_{\theta_{\text{old}}}(a \mid s, g_{\text{online}})$ are later trained as if they were drawn from $\pi_{\theta_{\text{old}}}(a \mid s, g_{\text{actual}})$, producing unreliable importance weights and unstable updates. This inconsistency explains why naive applications of PPO to ODT tend to fail (Yan et al., 2024). As shown in Fig. 1, removing hindsight return relabeling significantly improves stability and overall performance. In simpler environments such as MuJoCo Hopper, relabeling may yield transient gains but eventually leads to collapse, whereas in more complex environments such as Adroit Door, the model fails to learn altogether when relabeling is enabled.

Additionally, hindsight return relabeling requires access to the returns of the entire trajectory, which introduces additional challenges when optimizing over sub-trajectory rollouts (introduced in Section 3.2). In the experiments shown in Fig. 1, we apply hindsight return relabeling to sub-trajectories by rolling out the full trajectory.

3.2 Adapting GRPO to Decision Transformers

We provide an overview of online finetuning Decision Transformers with GRPO in Algorithm 1 (GRPO-DT), which is achieved by optimization on *sub-trajectories* rather than full trajectories as used in the original GRPO formulation (Shao et al., 2024; Guo et al., 2025). At each iteration, the current policy interacts with the environment to collect complete rollouts, from which we sample reset points and generate groups of sub-trajectories for each reset point. The reset points are also *actively selected* based on the variance in the policy action distribution. The algorithm computes advantages for each sub-trajectory according to Eq. (2). These sub-trajectories and their advantages are then used to update the policy with a *sequence-level importance ratio* described in Eq. (3).

Compared to vanilla GRPO, our method introduces three key design modifications to better align GRPO with Decision Transformers. Specifically, (i) we redesign the optimization objective to operate on a group of sub-trajectories rather than full rollouts, enabled by either resetting or learning an extra Q-function; (ii) we compute importance weights at the *sequence level* to better align computed advantages; and (iii) we

Algorithm 1 Online Finetuning Decision Transformers with GRPO (GRPO-DT)

Input: Pretrained policy π_{θ_1}, full trajectory buffer $\mathcal{T}_{\mathsf{replay}}$, sub-trajectory buffer $\mathcal{T}_{\mathsf{sub}}$, number of iterations T, initial RTG g_{online}, number of reset points in a trajectory K, sub-trajectory length L_{traj}, evaluation steps L_{eval}, group size G for GRPO.

for iteration $t = 1, \cdots, T$ **do**
  Roll out a full trajectory τ using the current policy $\pi_{\theta_t}(\cdot \mid s_1, g_{\mathsf{online}})$, conditioned on (randomized) initial state s_1 and RTG g_{online}; update $\mathcal{T}_{\mathsf{replay}}$ with τ. `// Collect full trajectory; FIFO buffer update.`
  Sample a minibatch $\mathcal{B}$ of full trajectories from $\mathcal{T}_{\mathsf{replay}}$ from distribution p with $p(\tau) := \frac{|\tau|}{\sum_{\tau \in \mathcal{T}} |\tau|}$.
  for each full trajectory $\tau \in \mathcal{B}$ **do**
    Sample K reset points $\{(s_k, g_k)\}_{k=1}^{K}$ from action-variance distribution.
    For each reset point (s_k, g_k), generate G sub-trajectories $\{\tau_{k_i}^{\mathsf{sub}}\}_{i=1}^{G}$ of length L_{traj} with the current policy π_{θ_t}; evaluate each sub-trajectory for L_{eval} more steps to get reward $R(\tau_{k_i}^{\mathsf{sub}})$. `// Sub-trajectory generation and evaluation.`
    Compute the advantage $\widehat{A}_{k_i}$ for each sub-trajectory $\tau_{k_i}^{\mathsf{sub}}$ using Eq. (2).
    Update the sub-trajectory buffer $\mathcal{T}_{\mathsf{sub}}$ with $\{(\tau_{k_i}^{\mathsf{sub}}, \widehat{A}_{k_i}, (s_k, g_k))\}_{i=1}^{|G|}$. `// FIFO buffer update.`
  Finetune the current policy with sub-trajectories in $\mathcal{T}_{\mathsf{sub}}$ using the sequence-level importance ratio (Eq. (3)) to get a new policy $\pi_{\theta_{t+1}}$.

Output: Online finetuned policy $\pi_{\theta_{T+1}}$.

incorporate an *active sampling* mechanism that prioritizes uncertain states for optimization. We describe each of these design choices in detail below and also provide abalations to demonstrate their effectiveness.

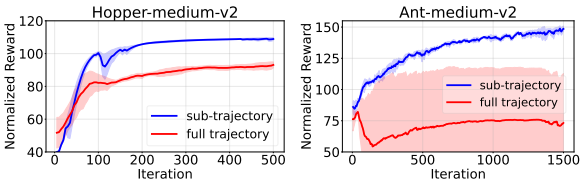
Sub-trajectory rollouts for better credit assignment. In its original formulation for language models, GRPO assigns a single response-level reward to all tokens within the same sequence (Shao et al., 2024; Guo et al., 2025), thereby discarding fine-grained credit assignment. A straightforward adaptation to classical RL environments and tasks would aggregate all stepwise rewards in a rollout and assign this trajectory-level return uniformly to every timestep. However, such a formulation performs poorly in RL environments: as shown in Fig. 2a, the model fails to learn when trained with full trajectories in the Ant-medium-v2 environment. This limitation is expected, as RL tasks—particularly those in continuous control—require more precise credit assignment than language modeling. Whereas tokens in a sentence tend to be coherently correlated, actions in RL can lead to drastically different outcomes (e.g., distinct action choices when navigating a maze).

To address this limitation, we adapt GRPO for Decision Transformers using a *sub-trajectory formulation*. We first select K reset points $\{(s_k, g_k)\}_{k=1}^{K}$ from each full trajectory. For every reset point (s_k, g_k), we generate G sub-trajectories $\{\tau_{k_i}^{\mathsf{sub}}\}_{i=1}^{|G|}$ of length L_{traj} using the current policy π_{θ_t}. To better attribute rewards to each sub-trajectory, we further roll out each one for an additional L_{eval} steps using the expected action (or the most probable action in the discrete case). The reward $r_{k_i}^{\mathsf{sub}}$ for sub-trajectory $\tau_{k_i}^{\mathsf{sub}}$ is defined as the cumulative discounted reward over $L_{\mathsf{traj}} + L_{\mathsf{eval}}$ steps, with a discount factor γ emphasizing rewards obtained near the reset point. For the additional L_{eval} steps, we use the expected action to reduce the variance induced by stochastic action sampling, resulting in more stable sub-trajectory return estimates. We then compute the advantage for each sub-trajectory in its group as:

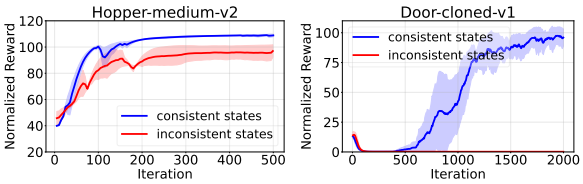
$$\widehat{A}_{k_i} = \frac{r_{k_i}^{\mathsf{sub}} - \mathrm{mean}(\{r_{k_1}^{\mathsf{sub}}, r_{k_2}^{\mathsf{sub}}, \ldots, r_{k_{|G|}}^{\mathsf{sub}}\})}{\mathrm{std}(\{r_{k_1}^{\mathsf{sub}}, r_{k_2}^{\mathsf{sub}}, \ldots, r_{k_{|G|}}^{\mathsf{sub}}\})}. \tag{2}$$

Only the sub-trajectory of length L_{traj} is used for GRPO optimization, while the subsequent L_{eval} steps are used solely for evaluation. The parameter L_{traj} controls the granularity of credit assignment, whereas L_{eval} determines the quality of reward estimation. Empirically, we find that a relatively small L_{traj} combined with a relatively large L_{eval} yields the best performance; see Section 4.3 for detailed ablations on these hyperparameters.

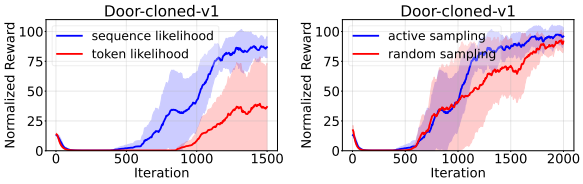
To ensure stable optimization, we enforce *state consistency* by resetting vectorized environments to the same states before generating sub-trajectories within each group. This reset mechanism is essential for convergence,



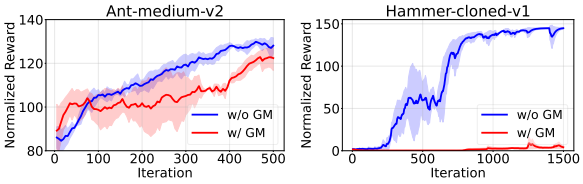
(a) Sub-trajectory vs. full trajectory



(b) Consistent vs. inconsistent states



(c) Sequence vs. token (d) Active vs. random



(e) Removing geometric mean in sequence-level ratio

Figure 2: Ablation studies illustrating the impact of key design choices in GRPO-DT. We compare variants of GRPO-DT with and without several proposed components across multiple environments. (a) Optimizing over sub-trajectories leads to faster convergence and higher final performance compared to updating on full trajectories. (b) Sampling groups from consistent states significantly improves learning stability over mixing inconsistent states. (c) Computing importance ratios using sequence-level likelihoods yields more reliable updates than token-wise likelihoods. (d) Actively selecting reset points based on uncertainty accelerates learning compared to random sampling. (e) Removing geometric mean (GM) normalization improves performance when computing sequence-level importance ratios in the sub-trajectory regime.

as shown in Fig. 2b. In scenarios where environment resetting is infeasible, we train an auxiliary Q-function using TD3 (Fujimoto et al., 2018) to evaluate candidate actions under a shared state (see Section 3.3.1 for details). This Q-function-guided variant also achieves competitive performance, as demonstrated in Section 4.3.

Sequence-level importance ratio. In standard GRPO, importance weights are computed at the token level, reflecting per-step likelihoods. When adapting GRPO to Decision Transformers, we find that computing importance weights at the *sequence level*—that is, over entire sub-trajectories of length L_{traj}—significantly improves model performance. Intuitively, the sequence-level importance ratio $\frac{\pi_\theta(\tau_{k_i}^{\mathsf{sub}}|s_k,g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau_{k_i}^{\mathsf{sub}}|s_k,g_k)}$ is better aligned with the advantage $\widehat{A}_{k_i}$, which is already computed at the sequence level. Empirical validation in Fig. 2c further supports this intuition.

We modify the original GRPO objective to incorporate sequence-level importance weighting. Since the sub-trajectory buffer $\mathcal{T}_{\mathsf{sub}}$ stores sub-trajectory $\tau_{k_i}^{\mathsf{sub}}$ along with its advantage $\widehat{A}_{k_i}$ and reset point (s_k, g_k), we randomly sample sub-trajectories from the sub-trajectory buffer and maximize the following objective:

$$J_{\mathsf{GRPO}}(\theta) = \mathbb{E}_{(\tau_{k_i}^{\mathsf{sub}},\widehat{A}_{k_i},(s_k,g_k))\sim\mathrm{unif}(\mathcal{T}_{\mathsf{sub}})} \min\left(\frac{\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k,g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau_{k_i}^{\mathsf{sub}} \mid s_k,g_k)}\widehat{A}_{k_i},\ \mathsf{clip}\left(\frac{\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k,g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau_{k_i}^{\mathsf{sub}} \mid s_k,g_k)}, 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_{k_i}\right) - \beta D_{\mathsf{KL}}(\pi_\theta \,\|\, \pi_{\theta_{\mathsf{ref}}}). \quad (3)$$

This sequence-level objective yields more stable and sample-efficient optimization, consistent with concurrent findings by Zheng et al. (2025) in language modeling. While Zheng et al. (2025) propose using a geometric mean over sequence-level importance ratios, we find that removing geometric normalization (as in Eq. (3)) leads to superior performance when combined with sub-trajectory optimization using relatively short rollout lengths (Fig. 2e). We hypothesize that this improvement arises because learning without geometric normalization (i) more rapidly suppresses outdated sub-trajectories through clipping, and (ii) permits more aggressive updates for approximately on-policy sub-trajectories.

While maximizing the GRPO objective in Eq. (3), we impose an entropy constraint $H(\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)) \geq \rho$

to encourage exploration. Here, $H(\pi_\theta(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k))$ denotes the average policy entropy computed over the sub-trajectory $\tau^{\mathsf{sub}}_{k_i}$, and ρ is a predefined lower bound. Following Zheng et al. (2022), we relax this hard constraint using a primal-dual formulation $\min_{\kappa \geq 0} \max_\theta J(\theta) + \kappa\big(H(\pi_\theta(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k)) - \rho\big)$, where κ is the dual variable. In practice, this corresponds to augmenting Eq. (3) with an additional entropy term $\kappa H(\pi_\theta(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k))$, while adaptively updating κ during training to ensure that the entropy constraint $H(\pi_\theta(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k)) \geq \rho$ is approximately satisfied.

Active sampling for state selection. During policy rollouts, we observe that certain state-RTG pairs (s_h, g_h) exhibit higher variance in the predicted action distribution $\pi_\theta(\cdot \mid s_h, g_h)$. Since higher variance indicates greater predictive uncertainty, resetting the starting points of sub-trajectory rollouts to such uncertain states can promote more effective exploration and accelerate learning. Motivated by this observation, we introduce an *active sampling* mechanism that biases sub-trajectory selection toward high-uncertainty states.

Specifically, for each trajectory τ, we compute a scalar uncertainty score at each step h by aggregating the diagonal covariance of the action distribution. Given $\pi_\theta(\cdot \mid s_h, g_h) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$ with $\Sigma_\theta = \mathrm{diag}(\sigma^2_{h,1}, \ldots, \sigma^2_{h,d})$. We define $\sigma^2_h := \frac{1}{d}\sum_{i=1}^{d} \sigma^2_{h,i}$, and apply a softmax transformation to obtain the sampling distribution $p_h := \frac{\exp(\sigma^2_h)}{\sum_{h'=1}^{|\tau|} \exp(\sigma^2_{h'})}$. Sub-trajectory reset points are then sampled according to p. This active selection strategy prioritizes learning updates in regions of high uncertainty, leading to faster and more stable convergence (Fig. 2d).

3.3 Extensions

3.3.1 Q-Guided GRPO-DT

In this section, we introduce *Q-guided GRPO-DT*, a variant of Algorithm 1 designed for settings in which environment resetting is infeasible. Q-guided GRPO-DT largely follows the pseudocode in Algorithm 1, except for the reset operation on line 6. Instead of explicitly rolling out sub-trajectories from a reset point (s_h, g_h), we train an auxiliary Q-function Q_{ϕ_t} to evaluate actions sampled from the current policy $\pi_{\theta_t}(\cdot \mid s_h, g_h)$ for advantage computation.

Specifically, at each state-RTG pair (s_h, g_h), we randomly sample a group of G candidate actions $\{a_{h_i}\}_{i=1}^{G} \sim \pi_{\theta_t}(\cdot \mid s_h, g_h)$. Each sampled action is treated as defining a *hypothetical sub-trajectory* that starts from (s_h, g_h) by first executing a_{h_i} and then following the policy π_{θ_t}. Under this interpretation, we evaluate the quality of each action using the Q-function and obtain a scalar reward $R(a_{h_i}) = Q_{\phi_t}(s_h, a_{h_i})$. These rewards are subsequently used to compute the normalized advantage following Eq. (2). The Q-function Q_{ϕ_t} is trained following the TD3 algorithm (Fujimoto et al., 2018).

3.3.2 Adapting PPO to Decision Transformers

In addition to adapting GRPO to DTs, we further extend *Proximal Policy Optimization* (PPO, Schulman et al., 2017) to the Decision Transformer framework, denoted as PPO-DT. Unlike prior attempts that directly apply PPO to DTs (Yan et al., 2024), our approach removes the *hindsight return relabeling* step during online finetuning, as discussed in Section 3.1. This adjustment ensures consistency between the rollout and training objectives, which is crucial for stable optimization under importance sampling. Our formulation also differs from PPO adaptations in multi-agent reinforcement learning that omit return-to-go (RTG) conditioning entirely (Meng et al., 2023), effectively reducing DTs to behavior cloning (Ross and Bagnell, 2010; Hussein et al., 2017). Following Zheng et al. (2022), we perform optimization on sub-trajectories sampled from full trajectories and incorporate an adaptive entropy term to encourage exploration. Since PPO computes advantages at the token level, PPO-DT accordingly employs *token-level importance ratios* instead of sequence-level ratios used in our GRPO-DT adaptation. For training the value network, we leverage generalized advantage estimation (GAE, Schulman et al., 2015) that combine Monte Carlo estimates with temporal-difference bootstrapping, yielding smoother and more stable value function updates. Further implementation details of PPO-DT are provided in Appendix B.2.

4 Experiments

In this section, we empirically evaluate our online finetuning algorithms for Decision Transformers using pure RL gradients. We describe the experimental setup in Section 4.1, present the main results in Section 4.2, and provide additional analyses and ablations in Section 4.3.

4.1 Experimental Setups

Environments and datasets. We evaluate methods on three continuous control and manipulation environments from D4RL (Fu et al., 2020): (i) **MuJoCo** (Todorov et al., 2012) tasks, including *Hopper*, *Walker2d*, and *Ant*, with dense rewards, evaluated on the *medium*, *medium-replay*, and *random* datasets. (ii) **Adroit** manipulation tasks (Rajeswaran et al., 2017), including *Door*, *Hammer*, and *Pen*, evaluated on the *human* and *cloned* datasets. (iii) **AntMaze** (Fu et al., 2020) with sparse goal-reaching rewards (a reward of 1 if success and 0 otherwise), using the *umaze* and *umaze-diverse* datasets. Detailed descriptions of each environment and dataset are provided in Appendix A.

Baselines. We compare our adapted GRPO-DT and PPO-DT against three main baselines: (i) **Online Decision Transformer** (**ODT**; Chen et al. (2021)), the standard online extension of DT that uses a supervised loss as the finetuning objective; (ii) **ODT+TD3** (Yan et al., 2024), the current state-of-the-art method for online finetuning of Decision Transformers; and (iii) **IQL** (Kostrikov et al., 2021), a widely used offline RL algorithm for which we employ its *online variant* in our experiments. For reference, we also report the performance of the offline pretrained **Decision Transformer** (**DT**) without online finetuning. Detailed hyperparameter settings are provided in Appendix B.1.

Metrics. Following D4RL (Fu et al., 2020), we report the normalized final reward for each algorithm, where higher values indicate better performance. All results are averaged over three runs, with standard deviations reported. We additionally present learning curves that track normalized reward throughout training.

For learning curves, the x-axis corresponds to the number of outer learning iterations, i.e., line 1 in Algorithm 1 and line 3 of Algorithm 1 in Zheng et al. (2022). This choice reflects a compromise for unifying different classes of methods. Specifically, IQL, ODT, and ODT+TD3 require nearly two orders of magnitude more gradient updates (and thus more computation) than our methods, GRPO-DT and PPO-DT, while our adapted policy-gradient-based algorithms consume several to tens of times more environment interactions. This difference arises because IQL, ODT, and ODT+TD3 benefit from extensive experience replay, whereas policy gradient methods typically require more online interactions.

To facilitate a fair comparison of asymptotic performance, we train all algorithms for a larger number of iterations than those reported in Yan et al. (2024). Evaluation is conducted after the gradient updates of each iteration. As a result, even at iteration 0, all methods have already undergone several updates, during which their behaviors may diverge and yield different outcomes.

4.2 Main Results

Table 1 reports the normalized returns and standard deviations averaged over three random seeds for each method. Overall, our proposed method GRPO-DT achieves the best performance across the majority of tasks, establishing new state of the arts. PPO-DT performs competitively in several cases but fails in certain environments (e.g., D-C-v1). ODT+TD3 achieves reasonable performance yet is generally outperformed by GRPO-DT. Both ODT and IQL underperform across most benchmarks, particularly on tasks with low-quality pretraining data such as the *random* datasets, as well as on challenging environments like Adroit. It is worth noting that our implementation of ODT+TD3 uses longer training iterations (as described in Section 4.1), leading to better results than those originally reported by Yan et al. (2024).

Learning with random offline data. The first block of Table 1 evaluates model performance when pretrained on *random* offline datasets (Fu et al., 2020) before the onset of online exploration. The offline datasets consist of trajectories collected by an untrained policy and thus contain little meaningful supervision signal,

Table 1: Comparison of average normalized final return of each method across various environments and datasets. The best results are shown in **bold**, and results within 10% of the best are underlined. Environment and task abbreviations are as follows: Ho = Hopper, Wa = Walker2d, An = Ant, D = Door, P = Pen, H = Hammer, U = UMaze, UD = UMaze-Diverse; dataset types: R = Random, M = Medium, MR = Medium-Replay, C = Cloned, H = Human. Each entry is reported as "performance ± standard deviation".

Environments	Datasets	DT	IQL	ODT	ODT+TD3	PPO-DT	GRPO-DT
MuJoCo (random)	Ho-R-v2	2.13	41.02 ± 13.35	30.43 ± 0.01	83.32 ± 8.46	**106.97 ± 0.96**	99.20 ± 3.80
	Wa-R-v2	4.53	22.75 ± 1.56	10.88 ± 0.34	82.95 ± 18.28	**108.69 ± 8.86**	100.25 ± 33.19
	An-R-v2	31.41	58.69 ± 23.03	19.08 ± 3.97	80.58 ± 7.25	107.45 ± 22.83	**120.69 ± 5.47**
	Average	12.69	40.82	20.13	82.28	**107.70**	**106.71**
MuJoCo (medium)	Ho-M-v2	46.46	74.19 ± 20.25	98.02 ± 0.63	101.47 ± 2.29	105.65 ± 5.43	**108.81 ± 0.85**
	Ho-MR-v2	38.12	96.97 ± 2.16	87.73 ± 0.59	107.94 ± 2.29	**109.60 ± 1.63**	83.61 ± 20.75
	Wa-M-v2	47.95	103.45 ± 1.37	76.49 ± 0.78	103.27 ± 5.95	109.49 ± 9.04	**158.34 ± 3.75**
	Wa-MR-v2	56.24	103.00 ± 2.65	74.21 ± 2.41	102.80 ± 2.68	117.45 ± 14.79	**137.36 ± 5.64**
	An-M-v2	86.22	118.18 ± 2.42	90.71 ± 0.03	131.56 ± 0.41	139.84 ± 0.95	**147.51 ± 2.44**
	An-MR-v2	84.30	117.51 ± 0.82	83.63 ± 0.87	120.01 ± 2.94	117.95 ± 2.54	**142.05 ± 3.32**
	Average	59.88	102.21	85.13	111.18	116.66	**129.61**
Adroit	D-C-v1	0.14	46.72 ± 0.30	1.26 ± 1.02	79.98 ± 5.62	0.19 ± 0.00	**96.41 ± 7.59**
	D-H-v1	3.28	11.27 ± 0.44	8.76 ± 3.87	79.73 ± 4.37	**94.12 ± 3.99**	89.33 ± 10.12
	P-C-v1	43.31	62.11 ± 13.25	16.24 ± 5.12	109.86 ± 6.27	27.14 ± 0.24	**111.15 ± 2.61**
	P-H-v1	33.10	24.94 ± 1.48	19.84 ± 7.42	77.18 ± 7.42	9.92 ± 5.00	**85.11 ± 6.08**
	H-C-v1	0.65	4.87 ± 3.10	1.32 ± 0.06	119.95 ± 2.45	130.60 ± 2.81	**140.45 ± 1.93**
	H-H-v1	1.08	1.04 ± 1.56	0.91 ± 0.22	120.93 ± 2.18	129.23 ± 2.18	**132.64 ± 12.56**
	Average	13.59	25.15	8.06	97.93	65.2	**109.18**
AntMaze	U-v2	50.00	91.21 ± 2.14	89.27 ± 3.73	**99.64 ± 0.20**	0.00 ± 0.00	96.07 ± 0.53
	UD-v2	65.00	0.00 ± 0.00	63.81 ± 1.64	**99.42 ± 0.43**	47.00 ± 4.00	97.70 ± 2.67
	Average	57.5	45.61	76.54	**99.53**	23.50	96.89

serving as a standard stress test for offline-to-online adaptation and model robustness to poor pretraining quality (Yan et al., 2024).

Under this challenging setup, methods that rely primarily on supervised objectives struggle to improve during online finetuning: ODT fails to make meaningful progress. IQL, while leveraging temporal-difference learning, suffers from inaccurate value estimation and limited policy improvement. ODT+TD3 achieves moderate performance by augmenting supervised objectives with RL gradients, yet remains substantially weaker than our GRPO-DT and PPO-DT, which rely purely on reinforcement learning gradients. By directly optimizing reward-driven RL signals, our methods effectively recover from poor initialization and achieve the best overall performance in this low-quality pretraining regime.

Learning with medium-quality offline data. The last three blocks of Table 1 report results obtained with medium-quality pretraining data before online exploration, representing realistic offline-to-online adaptation scenarios. On the **MuJoCo** tasks, our adapted methods, GRPO-DT and PPO-DT, achieve the highest overall returns. ODT+TD3 and IQL remain competitive, while ODT exhibits reasonable but inferior performance.

In the more challenging **Adroit** environment—where both state and action spaces are relatively high-dimensional—policies are prone to degradation or collapse during finetuning. Under these conditions, ODT and IQL fail to improve beyond their pretrained performance, whereas our adapted GRPO-DT consistently achieves strong results, obtaining the best performance on 5 out of 6 datasets. ODT+TD3 remains competitively on some Adroit tasks but is outperformed by our GRPO-DT across all datasets. PPO-DT achieves strong results on certain tasks but shows limited improvement in others.

Finally, in the **AntMaze** environment with sparse, goal-reaching rewards (a reward of 1 for success and 0 otherwise), ODT+TD3 achieves the best performance, while our GRPO-DT follows closely—achieving returns within 4% of the best score. All other methods fail to make meaningful progress under this sparse-reward setting.

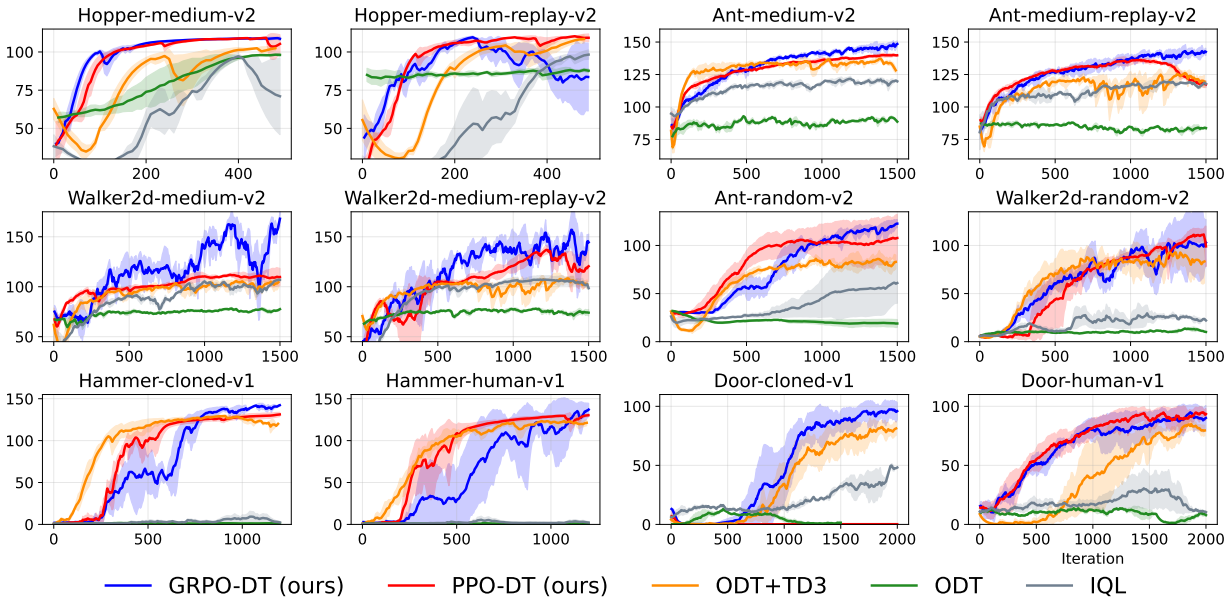


Figure 3: Performance comparison across different RL environments. Our proposed method GRPO-DT achieves the best performance across the majority of tasks. PPO-DT performs competitively in most cases but fails in certain environments. ODT+TD3 achieves overall decent performance but is generally outperformed by GRPO-DT. Both ODT and IQL consistently underperform across most environments.

Computational and practical advantages over ODT+TD3. Beyond achieving superior performance (as shown in Table 1 and Fig. 3), our adapted GRPO-DT and PPO-DT also offer notable computational and practical advantages over ODT+TD3.

(i) **Fewer gradient updates.** Sub-trajectory rollouts enable finer-grained credit assignment and more accurate gradient estimation, allowing our methods to learn effectively with far fewer updates. For example, each iteration of our approach performs only 8×256 gradient updates, compared to roughly 256×300 updates required by ODT+TD3 (and ODT), representing a substantial reduction in compute cost.

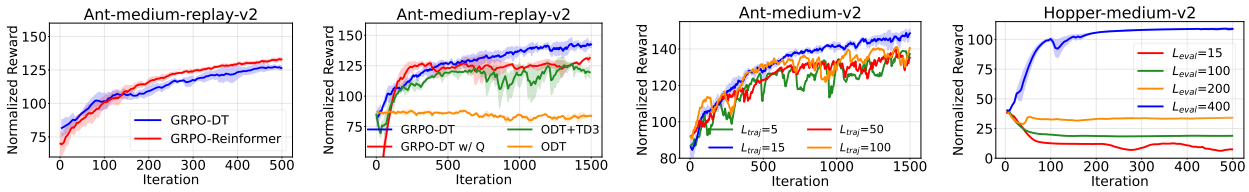
(ii) **Seamless compatibility with pretrained DT-style models.** Our method can directly finetune any pretrained DT-style model with minimal modification (see Section 4.3 for experiments on other models), whereas ODT+TD3 requires *modifying the offline pretraining objective* to integrate RL gradients and jointly train auxiliary Q-functions—significantly reducing its flexibility and scalability.

(iii) **Simpler implementation and improved stability.** Unlike ODT+TD3, which relies on auxiliary critic networks, our approach introduces no additional networks. This critic-free design simplifies implementation, improves training stability, and facilitates reproducibility across different environments and pretrained models.

4.3 Additional Analyses and Ablations

We provide additional analyses and ablation studies in this section. Empirical evidence supporting the key design choices in Algorithm 1 is presented in Fig. 1 and Fig. 2, including the effects of removing hindsight return relabeling, using sub-trajectory rollouts, using consistent states for rollouts, adopting sequence-level importance ratios, applying active sampling for state selection, and removing geometric mean normalization for sequence-level importance ratios.

Online finetuning beyond DTs. To assess the applicability of our GRPO-DT beyond the standard Decision Transformer, we evaluate it on the *Reinformer* model (Zhuang et al., 2024). Reinformer integrates



(a) GRPO on Reinformer (b) Q-guided GRPO-DT (c) Ablation on L_{traj} (d) Ablation on L_{eval}

Figure 4: Additional analyses and ablations. (a) Performance of GRPO-DT across different Decision Transformer models, including the standard DT and Reinformer, demonstrating that GRPO-DT generalizes beyond the standard DT. (b) Comparison between reset-based GRPO-DT and Q-guided GRPO-DT when environment resetting is infeasible; Q-guided GRPO-DT maintains strong performance even without resetting. (c) Ablation on sub-trajectory length L_{traj}: both overly short and overly long sub-trajectories degrade performance, highlighting the importance of an appropriate sub-trajectory length. (d) Ablation on sub-trajectory evaluation length L_{eval}: insufficient evaluation steps lead to unstable training or model collapse, while larger L_{eval} yields more reliable performance.

return maximization into supervised sequence modeling by leveraging expectile regression to predict higher RTGs, which then guide action selection. As shown in Fig. 4a, our GRPO-DT also performs effectively on Reinformer, suggesting that our proposed GRPO-DT generalize beyond standard DTs.

Q-guided GRPO-DT without resetting. In scenarios where environment resetting is infeasible, we train an auxiliary Q-function using TD3 (Fujimoto et al., 2018) and apply GRPO-DT with Q-guided advantages (see Section 3.3.1 for details). This variant also reduces sample complexity, as the sub-trajectory generation and evaluation step (line 6 in Algorithm 1) is no longer required. As shown in Fig. 4b, the Q-guided version of GRPO-DT maintains strong performance even without environment resetting.

Ablation on sub-trajectory length. In our method, each sub-trajectory serves as the fundamental unit for advantage computation and credit assignment, making its length L_{traj} a critical hyperparameter. Empirical results in Fig. 4c show that increasing the sub-trajectory length destabilizes training and degrades performance. Conversely, very short sub-trajectories can also lead to suboptimal results. This is likely because short segments sampled from similar state distributions are overly homogeneous, providing limited diversity and weaker learning signals for RL finetuning.

Ablation on sub-trajectory evaluation length. For each sub-trajectory, we extend the rollout by L_{eval} additional evaluation steps to compute rewards. In our experiments, L_{eval} ranges from 30 to 400 depending on the environment (see Table 6). We conduct an ablation study on the evaluation length L_{eval} to examine its impact. As shown in Fig. 4d, increasing L_{eval} provides more reliable estimates of sub-trajectory quality, leading to consistent improvements in model performance.

5 Related Work

Transformers for RL. With the transformer architecture (Vaswani et al., 2017) becoming the dominant paradigm in deep learning—most notably in language (Radford et al., 2018; Brown et al., 2020) and vision (Dosovitskiy et al., 2021)—a growing body of work has explored transformer-based approaches for reinforcement learning. In this paradigm, exemplified by Decision Transformers (Chen et al., 2021), RL is formulated as a sequence modeling problem, where models condition on past states, actions, and returns-to-go to autoregressively predict future actions. Leveraging the strong sequence modeling capability of transformers, DT-style methods have demonstrated competitive or state-of-the-art performance across a range of benchmarks (Janner et al., 2021; Wang et al., 2022; Yamagata et al., 2023; Zhuang et al., 2024). While initially developed for offline RL, several recent works have extended these methods to the online setting—often referred to as offline-to-online RL (Lee et al., 2022; Yu and Zhang, 2023; Nakamoto et al., 2023). However, existing online

finetuning approaches for Decision Transformers either rely purely on supervised sequence-modeling objectives (Zheng et al., 2022; Xie et al., 2023), or continue to prioritize the supervised loss by assigning only a small weight to reinforcement learning gradients (Yan et al., 2024). In contrast, our work complements this line of research by presenting the first approach to online finetuning of Decision Transformers using *pure reinforcement learning gradients*, enabling principled policy optimization without reliance on supervised objectives.

RL for transformers. Reinforcement learning has also emerged as a powerful approach for aligning and enhancing large language models (Ouyang et al., 2022; Bai et al., 2022; Lee et al., 2023) as well as multimodal models (Liu et al., 2025b; Shen et al., 2025). A broad spectrum of algorithms has been explored, ranging from online policy gradient methods such as PPO and its variants (Ouyang et al., 2022; Luong et al., 2024; Kazemnejad et al., 2025), to offline reinforcement learning methods based on IQL and its variants (Snell et al., 2022; Qi et al., 2024), as well as preference-based policy optimization approaches such as DPO and its extensions (Rafailov et al., 2023; Ethayarajh et al., 2024). More recently, substantial attention has been devoted to GRPO and its variants (Shao et al., 2024; Guo et al., 2025; Liu et al., 2025a; Yu et al., 2025), which have been successfully applied to large language and multimodal models to improve reasoning performance. In this work, we extend GRPO and PPO to the online finetuning of Decision Transformers in classical reinforcement learning environments. Along the way, we also introduce a sub-trajectory variant of GRPO that significantly improves credit assignment over standard GRPO, and which may be of independent interest beyond the settings considered in this paper.

Additional related work. During the preparation of this paper, we became aware of a concurrent work on sequence-level importance ratios by Zheng et al. (2025), which appeared online in July 2025. While our approach shares the same core idea, our sequence-level importance ratio technique was developed independently; an initial version of our work was made publicly available in September 2025. Our method differs from Zheng et al. (2025) in a key design choice. Specifically, while Zheng et al. (2025) propose using a geometric mean over sequence-level importance ratios, we remove this geometric normalization when performing sub-trajectory-level optimization. Empirically, we find that this modification yields superior performance when combined with relatively short rollout lengths (Fig. 2e). We hypothesize that this improvement arises because learning without geometric normalization (i) more rapidly suppresses outdated sub-trajectories via clipping, and (ii) enables more aggressive updates for approximately on-policy sub-trajectories.

Our active state sampling technique is inspired by classical active learning (Settles, 2009), where uncertainty-based methods are known to exponentially improve sample complexity compared to standard supervised learning (Hanneke, 2014; Puchkin and Zhivotovskiy, 2021; Zhu and Nowak, 2022a,b), and have demonstrated strong empirical performance with deep neural networks (Ash et al., 2020; Saran et al., 2023; Zhang et al., 2024). More recently, the core principles of active learning have been applied to reinforcement learning (Yin et al., 2023), large language models (Bhatt et al., 2024; Wang et al., 2025), and multimodal models (Zhang and Zhu, 2025).

6 Conclusion

In this work, we conducted a systematic study of online finetuning Decision Transformers using *pure reinforcement learning gradients*. We identify hindsight return relabeling as a key obstacle that hinders the effective application of importance sampling-based policy gradient methods to online DT finetuning. Building on this insight, we propose a principled solution by adapting GRPO to the Decision Transformer framework. Our approach integrates sub-trajectory optimization for improved credit assignment, sequence-level importance ratios for enhanced stability and efficiency, and active state sampling for better exploration, collectively enabling pure-RL online finetuning of pretrained DTs. We further demonstrate that PPO can also be adapted to online DT finetuning. Extensive experiments show that our methods outperform existing online DT baselines and achieve new state-of-the-art performance across multiple benchmarks. Overall, our findings highlight the viability and effectiveness of pure RL optimization for Decision Transformers and open new avenues for scaling DT-based agents through RL-driven finetuning.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Gantavya Bhatt, Yifang Chen, Arnav M Das, Jifan Zhang, Sang T Truong, Stephen Mussmann, Yinglun Zhu, Jeffrey Bilmes, Simon S Du, Kevin Jamieson, et al. An experimental design framework for label-efficient supervised finetuning of large language models. *Findings of the Association for Computational Linguistics*, 2024.

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

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.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Steve Hanneke. Theory of active learning. *Foundations and Trends in Machine Learning*, 7(2-3), 2014.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

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

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025a.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 2024.

Linghui Meng, Muning Wen, Chenyang Le, Xiyun Li, Dengpeng Xing, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, Yaodong Yang, et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20(2):233–248, 2023.

Zak Mhammedi, Dylan J Foster, and Alexander Rakhlin. The power of resets in online reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12334–12407, 2024.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Nikita Puchkin and Nikita Zhivotovskiy. Exponential savings in agnostic active learning through abstention. In *Conference on learning theory*, pages 3806–3832. PMLR, 2021.

Jianing Qi, Hao Tang, and Zhigang Zhu. Verifierq: Enhancing llm test time compute with q-learning-based verifiers. *arXiv preprint arXiv:2410.08048*, 2024.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

Akanksha Saran, Safoora Yousefi, Akshay Krishnamurthy, John Langford, and Jordan T Ash. Streaming active learning with deep neural networks. In *International Conference on Machine Learning*, pages 30005–30021. PMLR, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Burr Settles. Active learning literature survey. 2009.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, 2025.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 34748–34761, 2022.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

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*, pages 38187–38203. PMLR, 2023.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pages 38989–39007. PMLR, 2023.

Kai Yan, Alex Schwing, and Yu-Xiong Wang. Reinforcement learning gradients as vitamin for online finetuning decision transformers. *Advances in Neural Information Processing Systems*, 37:38590–38628, 2024.

Dong Yin, Sridhar Thiagarajan, Nevena Lazic, Nived Rajaraman, Botao Hao, and Csaba Szepesvari. Sample efficient deep reinforcement learning via local planning. *arXiv preprint arXiv:2301.12579*, 2023.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In *International Conference on Machine Learning*, pages 40452–40474. PMLR, 2023.

Jiancheng Zhang and Yinglun Zhu. Towards multimodal active learning: Efficient learning with limited paired data. *arXiv preprint arXiv:2510.03247*, 2025.

Jifan Zhang, Yifang Chen, Gregory Canal, Arnav Mohanty Das, Gantavya Bhatt, Stephen Mussmann, Yinglun Zhu, Jeff Bilmes, Simon Shaolei Du, Kevin Jamieson, et al. Labelbench: A comprehensive framework for benchmarking adaptive label-efficient learning. *Journal of Data-centric Machine Learning Research*, 2024.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022.

Yinglun Zhu and Robert Nowak. Active learning with neural networks: Insights from nonparametric statistics. *Advances in Neural Information Processing Systems*, 35:142–155, 2022a.

Yinglun Zhu and Robert Nowak. Efficient active learning with abstention. *Advances in Neural Information Processing Systems*, 35:35379–35391, 2022b.

Zifeng Zhuang, Dengyun Peng, Jinxin Liu, Ziqi Zhang, and Donglin Wang. Reinformer: Max-return sequence modeling for offline rl. *arXiv preprint arXiv:2405.08740*, 2024.

Table 2: MuJoCo dataset size (in terms of total #steps) and normalized final reward statistics.

Dataset	Size	Normalized Reward
Hopper-medium-v2	999 906	44.32 ± 12.27
Hopper-medium-replay-v2	402 000	14.98 ± 16.32
Hopper-random-v2	999 906	1.19 ± 1.16
Walker2d-medium-v2	999 995	62.09 ± 23.83
Walker2d-medium-replay-v2	302 000	14.84 ± 19.48
Walker2d-random-v2	999 997	0.01 ± 0.09
Ant-medium-v2	999 946	80.30 ± 35.82
Ant-medium-replay-v2	302 000	30.95 ± 31.66
Ant-random-v2	999 930	6.36 ± 10.07

A Environment and Dataset Details

We consider three continuous control and manipulation environments from D4RL (Fu et al., 2020): *MuJoCo*, *Adroit*, and *AntMaze*. In total, we conduct experiments on 8 different tasks spanning 17 datasets with varying offline data quality. We provide detailed descriptions of each environment, task, and dataset below.

MuJoCo environments. The first environment is **MuJoCo** (Todorov et al., 2012), including locomotion tasks *Hopper*, *Walker2d*, and *Ant* with dense reward signals. We evaluate these tasks using the *medium*, *medium-replay*, and *random* datasets, where the *medium* dataset contains trajectories generated by a policy early-stopped at medium-level performance, the *medium-reply* dataset contains trajectories sampled in the training process of the medium policy, and the *random* contains trajectories collected by a random policy. Dataset size and normalized reward statistics of each offline dataset is provided in Table 2. We discuss each task below.

- **Hopper.** Hopper is a single-legged locomotion task where the agent controls three joints to make the robot hop forward while maintaining stability. The action space is a 3-dimensional continuous vector, corresponding to torques applied at the joints, each bounded within $[-1, 1]$. The observation space has 11 dimensions, consisting of positional and velocity information. At each timestep, the reward is a combination of a healthy reward, a forward progress reward, and a control cost penalty proportional to the squared magnitude of the action.
- **Walker2d.** Walker2d is a 2-dimensional bipedal walking robot task where the agent controls six joints to make the robot walk forward steadily. The action space is a 6-dimensional continuous vector, corresponding to torques applied at hinge joints, each bounded within $[-1, 1]$. The observation space has 17 dimensions, consisting of positional and velocity information. At each timestep, the reward is a combination of a healthy reward, a forward progress reward, and a control cost penalty proportional to the squared magnitude of the action.
- **Ant.** Ant is a 3-dimensional locomotion task where the agent controls an 8-joint quadruped to move forward while maintaining balance. The action space is a 8-dimensional continuous vector, corresponding to torques applied at hinge joints, each bounded within $[-1, 1]$. The observation space has 105 dimensions consisting of positional, velocity, and external contact force information. At each timestep, the reward is a combination of a healthy reward, a forward progress reward, a control cost penalty proportional to the squared magnitude of the action, and an external contact force penalty proportional to the squared magnitude of contact force.

Adriot environment. The second environment is **Adroit**, including challenging manipulation tasks *Door*, *Hammer*, and *Pen*. We evaluate these tasks using the *human* and *cloned* datasets, where the *human* dataset contains a small amount of human demonstrations, and the *cloned* dataset contains the mixture of trajectories collected from a behavior cloning policy and human demonstrations. Dataset size and normalized reward statistics of each offline dataset is provided in Table 3. We discuss each task below.

Table 3: Adroit dataset size (in terms of total #steps) and normalized final reward statistics.

Dataset	Size	Normalized Reward
Pen-cloned-v1	499 886	108.63 ± 122.43
Pen-human-v1	4800	202.69 ± 154.48
Hammer-cloned-v1	999 872	8.11 ± 23.35
Hammer-human-v1	10 948	23.80 ± 33.86
Door-cloned-v1	999 939	12.29 ± 18.35
Door-human-v1	6504	28.35 ± 13.88

Table 4: AntMaze dataset size (in terms of total #steps) and normalized final reward statistics.

Dataset	Size	Normalized Reward
Antmaze-umaze-v2	998 573	86.14 ± 34.55
Antmaze-umaze-diverse-v2	999 000	3.48 ± 18.32

- **Door.** The Door task involves a robotic hand-arm system that learns to unlatch and open a door. The action space is a 28-dimensional continuous vector representing joint angular positions (scaled to $[-1, 1]$). The 39-dimensional observation space contains information about the angular position of the finger joints, the pose of the palm of the hand, as well as state of the latch and door. The dense reward combines distance, hinge-alignment, and velocity penalties with positive rewards for door hinge displacement.
- **Hammer.** The Hammer task involves a robotic hand-arm system to pick up a hammer and drive a nail into a board. The action space is a 26-dimensional continuous vector representing joint angular positions (scaled to $[-1, 1]$). The 46-dimensional observation space contains information about the angular position of the finger joints, the pose of the palm of the hand, the pose of the hammer and nail, and external forces on the nail. The dense reward combines distance and velocity penalties with positive rewards for lifting the hammer and driving the nail.
- **Pen.** The Pen task involves a robotic hand-arm system to manipulate a pen into a target orientation. The action space is a 24-dimensional continuous vector representing joint angular positions (scaled to $[-1, 1]$). The 45-dimensional observation space contains information about the angular position of the finger joints, the pose of the palm of the hand, as well as the pose of the real pen and target goal. The dense reward combines distance and orientation penalties and a penalty for dropping the pen, with bonuses for precise alignment and stable control.

AntMaze environment. The third environment is **AntMaze** (Fu et al., 2020), including navigation tasks in mazes with sparse goal-reaching rewards. We evaluate on the *umaze* and *umaze-diverse* tasks, where the former has fixed starting and ending points and the later has random starting/ending points. The umaze environment in AntMaze places an ant quadruped in a U-shaped maze. The action space is a 8-dimensional continuous vector, corresponding to torques applied at hinge joints, each bounded within $[-1, 1]$. The observation space is a goal-aware dictionary, consisting of a 27-dimensional "observation" vector (positions and velocities of the Ant body parts), a 2-dimensional desired goal vector (coordinates of the desired position), and a 2-dimensional achieved goal vector (coordinates of the current position). The reward is sparse, granting a value of 1 when the ant reaches the target position and 0 otherwise. Dataset size and normalized reward statistics of both offline dataset is provided in Table 4.[1]

[1]Following ODT+TD3's (Yan et al., 2024) practice, we remove all 1-step trajectories in the dataset.

Table 5: Common architecture and training hyperparameters used across all environments. The policy network is parameterized as a Transformer, while the value network is parameterized as an MLP.

Hyperparameters for policy network (Transformer)	Value
Number of layers	4
Number of attention heads	4
Embedding dimension	512
Activation function	SiLU (Elfwing et al., 2018)
Optimizer	LAMB (You et al., 2019)
Dropout	0.1 in pretraining, disabled in finetuning
Gradient norm clip	0.5
Weight decay	1×10^{-4}
KL coefficient β	1×10^{-3}
Target entropy ρ	$-\mathsf{dim}(\mathcal{A})$
Hyperparameters for value network (MLP)	**Value**
Number of layers	2
Embedding dimension	256 for MuJoCo, 512 for others
Activation function	SiLU (Elfwing et al., 2018)
Optimizer	AdamW (Loshchilov and Hutter, 2017)

B Experimental Details

B.1 Hyperparameters

In this section, we describe the hyperparameters used in our experiments. For the ODT+TD3 and ODT baselines, we use the codebase and default hyperparameters provided by Yan et al. (2024). For the IQL baseline, we largely follow the implementation of Yan et al. (2024), but set the number of pretraining steps to match those used by our methods and other baselines to ensure a fair comparison.

Table 5 summarizes the common architectural and training hyperparameters used by our algorithms across all environments, which largely follow the practices of Yan et al. (2024). Following Zheng et al. (2022) and Yan et al. (2024), we do not use positional embeddings. Additional algorithm-specific details are provided below.

For GRPO-DT, we collect 1 full trajectory per iteration in MuJoCo and AntMaze, and 5 full trajectories per iteration in Adroit. The full-trajectory replay buffer stores up to 32 trajectories. During resetting, we sample 16 trajectories from the buffer, select 4 reset points per trajectory, and construct groups of size 8, resulting in 512 sub-trajectories per iteration. The sub-trajectory buffer stores up to 2048 sub-trajectories. When forming groups of sub-trajectories, we discard those whose raw rewards are within an additive margin Δ_r of the group-average reward, in order to provide stronger optimization signals. Environment-specific hyperparameters for GRPO-DT are reported in Table 6.

For Q-guided GRPO-DT, we conduct experiments on Ant-medium-replay-v2 (Fig. 4b) and largely follows the practice of Yan et al. (2024) for training the Q-functions and our GRPO-DT for training the policy network. To improve training stabilities, we change the policy learning rate to $\mathsf{lr}_{\mathsf{policy}} = 1 \times 10^{-5}$ and the initial entropy dual variable to $\kappa_{\mathsf{initial}} = 1 \times 10^{-2}$.

For PPO-DT, we collect 8 full trajectories per iteration in MuJoCo and AntMaze, and either 8 or 16 full trajectories per iteration in Adroit (see Table 7). The full-trajectory replay buffer stores 4 times the number of trajectories collected per iteration. Following the practice of ODT+TD3, we apply LayerNorm (Ba et al., 2016) to the value network in Adroit and AntMaze to improve training stability. Environment-specific hyperparameters for PPO-DT are reported in Table 7.

B.2 Implementation Details of PPO-DT

In this section, we present the implementation of our PPO-DT (Algorithm 2), which adapts the classical Proximal Policy Optimization (PPO, Schulman et al., 2017) to Decision Transformers (Chen et al., 2021).

Table 6: Environment-specific hyperparameters for GRPO-DT. n_{batch} denotes the training batch size. c_{offline} and c_{online} denote the context lengths used in the offline and online stages, respectively. g_{online} denotes the initial return-to-go for online exploration. L_{traj} denotes the sub-trajectory rollout length, and L_{eval} denotes the number of additional evaluation steps. γ is the discount factor used to compute sub-trajectory rewards. ε denotes the GRPO clipping hyperparameter, and Δ_r denotes the additive margin used for sub-trajectory filtering. $\mathsf{lr}_{\mathsf{policy}}$ denotes the learning rate for the policy network, and $\kappa_{\mathsf{initial}}$ denotes the initial value of the entropy dual variable for online finetuning, which is adaptively updated during training.

Environments	n_{batch}	c_{offline}	c_{online}	g_{online}	L_{traj}	L_{eval}	γ	ε	Δ_r	$\mathsf{lr}_{\mathsf{policy}}$	$\kappa_{\mathsf{initial}}$
Ho-M/MR-v2	256	20	1	7200	15	400	0.995	0.2	2.0	5e-5	0.20
Ho-R-v2	256	20	1	7200	15	400	0.995	0.2	2.0	5e-5	0.20
Wa-M/MR-v2	256	20	1	10000	15	400	0.995	0.3	2.0	5e-5	0.04
Wa-R-v2	256	20	1	10000	15	400	0.995	0.3	2.0	5e-5	0.20
An-M/MR-v2	256	20	1	12000	15	200	0.995	0.3	2.0	5e-5	0.04
An-R-v2	256	20	1	12000	15	200	0.995	0.3	2.0	5e-5	0.20
D-C-v1	512	5	1	3000	10	100	0.99	0.3	0.5	3e-5	0.10
D-H-v1	512	5	1	3000	10	100	0.99	0.3	0.4	3e-5	0.04
P-C-v1	512	5	1	6000	3	30	0.99	0.3	0	3e-5	0.02
P-H-v1	512	5	1	6000	3	30	0.99	0.3	0	3e-5	0.02
H-C-v1	512	5	5	4000	10	100	0.99	0.3	0	3e-5	0.05
H-H-v1	512	5	5	4000	10	100	0.99	0.3	0.8	3e-5	0.05
U-v2	256	5	1	2	10	200	1.0	0.2	0	5e-5	0.05
UD-v2	256	1	5	2	10	200	1.0	0.2	0	5e-5	0.05

In each iteration, we sample K_{PPO} complete trajectories. For each trajectory τ and each time step h, we compute the advantage via generalized advantage estimation (GAE, Schulman et al., 2015):

$$\widehat{A}_h = \sum_{\ell=0}^{T-h-1} (\gamma\lambda)^\ell \Big(r_{h+\ell} + \gamma V_{\phi_t}(s_{h+\ell+1}) - V_{\phi_t}(s_{h+\ell}) \Big), \tag{4}$$

where V_{ϕ_t} denote the learned value function, $\gamma \in [0,1]$ denotes the discount factor and $\lambda \in [0,1]$ controls the bias-variance trade-off of GAE. To update the policy network, we first randomly sample sub-trajectory $\tau_k^{\mathsf{sub}} = (s_{k_1}, a_{k_1}, g_{k_1}, \cdots, s_{k_L}, a_{k_L}, g_{k_L})$ of length $L = c_{\mathsf{train}}$ to form a batch $\mathcal{B}_{\mathsf{sub}}$; following the practice in Huang et al. (2022), we further normalize the advantages across the batch. We then update the parameter by maximizing the following objective:

$$\begin{aligned} J_{\mathsf{PPO}}(\theta) = & \frac{1}{|\mathcal{B}_{\mathsf{sub}}||\tau_k^{\mathsf{sub}}|} \sum_{\tau_k^{\mathsf{sub}} \in \mathcal{B}_{\mathsf{sub}}} \sum_{i=1}^{|\tau_k^{\mathsf{sub}}|} \\ & \min\left(\frac{\pi_\theta(a_{k_i} \mid s_{k_i}, g_{k_i})}{\pi_{\theta_{\mathsf{old}}}(a_{k_i} \mid s_{k_i}, g_{k_i})} \widehat{A}_{k_i},\ \mathsf{clip}\left(\frac{\pi_\theta(a_{k_i} \mid s_{k_i}, g_{k_i})}{\pi_{\theta_{\mathsf{old}}}(a_{k_i} \mid s_{k_i}, g_{k_i})}, 1-\varepsilon, 1+\varepsilon \right) \widehat{A}_{k_i} \right) - \beta D_{\mathsf{KL}}(\pi_\theta \,\|\, \pi_{\theta_{\mathsf{ref}}}) . \end{aligned} \tag{5}$$

Similar to GRPO-DT, we augment Eq. (5) with an additional entropy term $\kappa H(\pi_\theta(\tau_k^{\mathsf{sub}} \mid s_k, g_k))$, while adaptively updating κ during training to ensure that the entropy constraint $H(\pi_\theta(\tau_k^{\mathsf{sub}} \mid s_k, g_k)) \geq \rho$ is approximately satisfied.

The value function is updated by minimizing the square loss with respect to value target $V_{\mathsf{target}(s_h)} = \widehat{A}_h + V_{\phi_t}(s_h)$ on states present in the sampled sub-trajectories:

$$\mathcal{L}_{\mathsf{value}} = \mathbb{E}_{s_h}\left[(V_\phi(s_h) - V_{\mathrm{target}}(s_h))^2 \right]. \tag{6}$$

Table 7: Environment-specific hyperparameters for PPO-DT. n_{batch} denotes the training batch size. c_{offline} and c_{online} denote the context lengths used in the offline and online stages, respectively. g_{online} denotes the initial return-to-go for online exploration. K_{PPO} denotes the number of online full trajectories rollouts per iteration. $\mathsf{lr}_{\mathsf{policy}}$ and $\mathsf{lr}_{\mathsf{value}}$ denote the learning rates for the policy and value networks, respectively. $\kappa_{\mathsf{initial}}$ denotes the initial value of the entropy dual variable for online finetuning, which is adaptively updated during training. Across all environments, we set the PPO clipping hyperparameter $\varepsilon = 0.2$ and the GAE hyperparameters $(\gamma, \lambda) = (0.99, 0.95)$.

Environments	n_{batch}	c_{offline}	c_{online}	g_{online}	K_{PPO}	$\mathsf{lr}_{\mathsf{policy}}$	$\mathsf{lr}_{\mathsf{value}}$	$\kappa_{\mathsf{initial}}$
Ho-M/MR-v2	256	20	1	7200	8	5e-5	1e-3	0.02
Ho-R-v2	256	20	1	7200	8	5e-5	1e-3	0.04
Wa-M/MR-v2	256	20	1	10000	8	5e-5	1e-3	0.02
Wa-R-v2	256	20	1	10000	8	5e-5	1e-3	0.20
An-M/MR-v2	256	20	1	12000	8	5e-5	1e-3	0.02
An-R-v2	256	20	1	12000	8	5e-5	1e-3	0.02
D-C-v1	512	5	1	3000	16	3e-5	2e-4	0.002
D-H-v1	512	5	1	3000	16	3e-5	2e-4	0.002
P-C-v1	512	5	1	6000	8	3e-5	2e-4	0.04
P-H-v1	512	5	1	6000	8	3e-5	2e-4	0.04
H-C-v1	512	5	5	4000	16	3e-5	2e-4	0.005
H-H-v1	512	5	5	4000	16	3e-5	2e-4	0.005
U-v2	256	5	1	2	8	5e-5	1e-3	0.02
UD-v2	256	1	5	2	8	5e-5	1e-3	0.02

Algorithm 2 Online Finetuning Decision Transformers with PPO (PPO-DT)

Input: Pretrained policy π_{θ_1}, value network V_{ϕ_1}, full trajectory buffer $\mathcal{T}_{\mathsf{replay}}$, number of iterations T, initial return-to-go g_{online}, number of trajectories collected per iteration n_{PPO}, sub-trajectory length $\mathsf{CL}_{\mathsf{train}}$.

1: **for** iteration $t = 1, \cdots, T$ **do**

2: Roll out K_{PPO} trajectories using the current policy $\pi_{\theta_t}(\cdot \mid s_1, g_{\mathsf{online}})$, conditioned on initial (randomized) state s_1 and RTG g_{online}; update $\mathcal{T}_{\mathsf{replay}}$. `// Collect full trajectories; FIFO buffer update.`

3: For each trajectory $\tau \in \mathcal{T}_{\mathsf{replay}}$, compute advantage for each step based on GAE (Eq. (4)).

4: Sample a batch $\mathcal{B}$ of full trajectories from $\mathcal{T}_{\mathsf{replay}}$ from distribution $p(\tau) = \frac{|\tau|}{\sum_{\tau \in \mathcal{T}_{\mathsf{replay}}} |\tau|}$.

5: For each trajectory $\tau \in \mathcal{B}$, randomly sample one sub-trajectory τ^{sub} of length $L = c_{\mathsf{train}}$ to form the sub-trajectory batch $\mathcal{B}_{\mathsf{sub}}$. Normalize the advantages across batch $\mathcal{B}_{\mathsf{sub}}$.

6: Finetune the current policy based on Eq. (5) to get an updated $\pi_{\theta_{t+1}}$; finetune the value network based on Eq. (6) to get an updated $V_{\phi_{t+1}}$.

Output: Online finetuned policy $\pi_{\theta_{T+1}}$.