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

The *Decision Transformer* (DT) has emerged as a powerful paradigm for decision making by framing offline reinforcement learning (RL) as a sequence modeling problem. However, while recent studies have begun extending DTs to online settings, *online finetuning with pure RL gradients* remains largely underexplored—most existing approaches continue to rely primarily on supervised sequence modeling objectives. We identify *hindsight return relabeling*—a component widely used in online DTs—as a key obstacle that, while beneficial for supervised objectives, hinders the performance of importance sampling-based RL algorithms such as PPO and GRPO. In this work, we present a new algorithm that enables online finetuning of Decision Transformers purely with reinforcement learning gradients. Our approach adapts the GRPO framework—originally developed for large language models—to sequential decision making with DTs, introducing several key modifications: 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. Across diverse benchmarks, our method consistently outperforms existing online finetuning baselines such as ODT and ODT+TD3, opening a promising direction toward unifying reinforcement learning and sequence modeling within Decision Transformer-based policies.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) lies at the core of the success of modern foundation models. Large language models (LLMs), in particular, demonstrate remarkable generalization and reasoning capabilities through a simple yet powerful recipe: large-scale pretraining followed by supervised and reinforcement learning-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 into sequential decision making, framing reinforcement learning (RL) as a conditional sequence modeling problem. Unlike conventional RL algorithms, DTs are trained entirely *offline* with a supervised objective over pre-collected trajectories, effectively performing imitation learning (Hussein et al., 2017) conditioned on a desired initial return-to-go (RTG).

The *Online Decision Transformer* (ODT, Zheng et al. (2022)) extends this framework by enabling online finetuning after offline pretraining. During online finetuning, ODT first collects trajectories and then finetunes with *hindsight return relabeling*, which replaces the intended RTG with the actual achieved return to align the RTG distribution of online data with that of the offline dataset. Recent work further augments ODT with TD3 (Fujimoto et al., 2018) gradients to improve performance (Yan et al., 2024). However, existing approaches to online finetuning of DTs remain dominated by supervised objectives: ODT relies purely on supervised loss, while ODT+TD3 only assigns a small weight to the RL gradients. Meanwhile, recent breakthroughs in LLM training reveal that purely reinforcement learning-based finetuning—such as PPO or GRPO—can significantly enhance reasoning and alignment (Shao et al., 2024; Guo et al., 2025; Team, 2025). This motivates a natural and fundamental question:

Can Decision Transformers be finetuned online using pure RL gradients?

[†]Project lead and corresponding author.

To explore this question, we revisit the training paradigm of existing online DT variants and uncover a key challenge. We find that *hindsight return relabeling*, while helpful for supervised objectives, fundamentally conflicts with on-policy RL algorithms that rely on importance sampling. By altering the conditioning variable between rollout and training, it introduces a mismatch in the importance ratio, which destabilizes training and degrades performance (Fig. 1a). Removing this step is therefore a necessary prerequisite for applying importance sampling-based algorithms such as PPO or GRPO to online finetuning of Decision Transformers.

Building on this insight, we develop a new algorithm for online finetuning of DTs *using pure RL gradients*. Specifically, we adapt GRPO—originally designed for LLM reasoning—to sequential decision making, resulting in our method GRPO-DT. Our approach incorporates three key innovations: (i) a sub-trajectory-based optimization objective for fine-grained credit assignment, supported by either environment resetting (Mhammedi et al., 2024) or an auxiliary Q-function; (ii) a sequence-level importance ratio that enhances stability and efficiency; and (iii) an active sampling mechanism that prioritizes uncertain states for exploration. Together, these modifications enable RL-only finetuning of pretrained DTs and achieve state-of-the-art performance across multiple benchmarks. We also adapt PPO for the same setting (PPO-DT), attaining competitive performance where previous PPO-based approaches failed (Yan et al., 2024).

Contributions. We summarize our main contributions as follows:

(i) We identify *hindsight return relabeling* as the key obstacle preventing effective online finetuning of Decision Transformers with importance sampling-based methods such as PPO and GRPO.

(ii) We introduce GRPO-DT, an adaptation of GRPO for Decision Transformers that integrates sub-trajectory optimization, sequence-level importance ratios, and active state sampling, enabling pure-RL finetuning of Decision Transformers.

(iii) We demonstrate through extensive experiments that online finetuning with pure RL gradients—via our GRPO-DT and PPO-DT—achieves new state-of-the-art results across diverse benchmarks.

Paper organization. The remainder of the paper is organized as follows. Section 2 reviews background on MDPs, Decision Transformers, online finetuning, and GRPO. Section 3 introduces our method, and Section 4 presents experimental results. Section 5 discusses related work, and Section 6 concludes our paper.

2 Preliminaries

Markov Decision Process. We formulate the reinforcement learning environment as a *Markov Decision Process* (MDP), defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. 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, and $\gamma \in [0, 1]$ is the discount factor. At each step $h = 1, \ldots, H$, the agent observes state $s_h \in \mathcal{S}$ and selects an action $a_h \in \mathcal{A}$ according to a policy $\pi(a_h \mid s_h)$. The agent then transits to the next state $s_{h+1} \sim P(\cdot \mid s_h, a_h)$ and obtains an immediate reward $r_h = R(s_h, a_h)$. We use $\tau = (s_1, a_1, r_1, \ldots, s_H, a_H, r_H)$ to denote the trajectory of interactions. The learner's objective is to learn a policy π to maximize the expected discounted cumulative reward $\mathbb{E}_{\mathcal{M},\pi}\left[\sum_{h=1}^{H} \gamma^{h-1} r_h\right]$.

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 at step h, denoted as g_h, represents the cumulative reward from step h onward. DT leverages a GPT-style architecture (Radford et al., 2018) to autoregressively learn a deterministic policy from pre-collected trajectories: let K denotes the context length, the DT learns to generate the next action a_h based on past interactions $(g_{h-K+1}, s_{h-K+1}, a_{h-K+1}, \cdots, g_h, s_h)$ of context length K. The model is trained via supervised learning by minimizing the mean squared error (MSE) between the predicted action and the ground-truth action. During evaluation and deployment, the learner specifies a desired initial RTG g_1, 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 DTs with interactively collected data. The offline pretraining stage is identical to DT training. In the online finetuning stage, the DT is deployed into the environment with a desired initial RTG 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 according to the achieved returns. 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 (NLL) 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 loss (with highsight 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 in their training objective by giving a smaller weight to the RL gradients.

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:

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

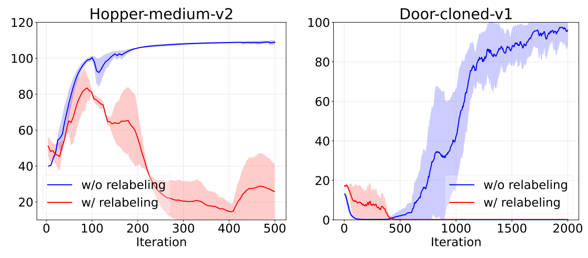
where $\widehat{A}_i = \frac{r_i - \mathrm{mean}(\{r_1, r_2, \cdots, r_G\})}{\mathrm{std}(\{r_1, r_2, \cdots, r_G\})}$ denotes the advantage of the i-th rollout, $w_{i,h}(\theta) = \frac{\pi_\theta(o_{i,h}|q, o_{i,<h})}{\pi_{\theta_{\mathsf{old}}}(o_{i,h}|q, o_{i,<h})}$ denotes the importance ratio, and $D_{\mathsf{KL}}(\pi_\theta \parallel \pi_{\mathsf{old}})$ denotes the KL penalty that prevents the model from deviating too far from policy $\pi_{\theta_{\mathsf{old}}}$.

3 Methods

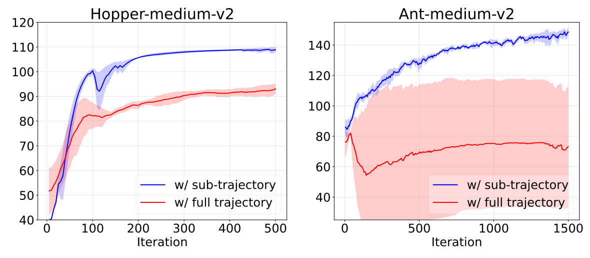
In Section 3.1, we begin by analyzing a key limitation of adapting Decision Transformers to the online setting using importance sampling-based algorithms such as PPO, and discuss how to address them. Building on these insights, Section 3.2 presents our adaptation of GRPO to Decision Transformers, incorporating several key modifications.

3.1 Removing Hindsight Return Relabeling

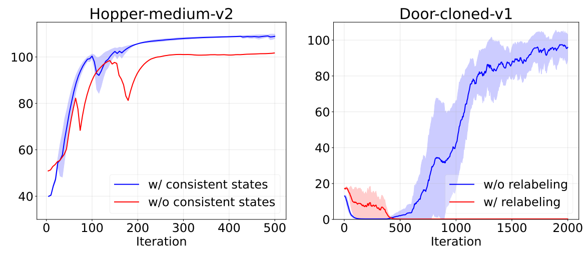
When deploying DTs, the learner must specify a desired initial RTG g_1, since the ground-truth future RTG is unknown in advance. In Online Decision Transformer (ODT), the learner typically sets a relatively high target RTG $g_1 = g_{\mathsf{online}}$ during rollout to encourage optimistic exploration. During training, a key component of ODT—known as *hindsight return relabeling*—replaces the intended RTG g_{online} with the actually achieved RTG g_{actual} (Zheng et al., 2022).



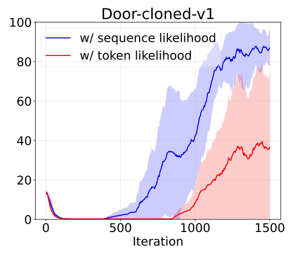
(a) Hindsight return relabeling



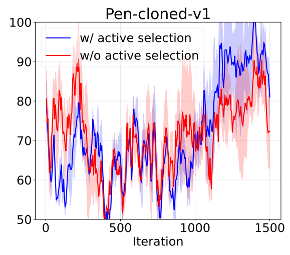
(b) Sub-trajectory and full trajectory



(c) Consistent and inconsistent states



(d) Sequence likelihoods



(e) Active selection

Figure 1: Examples of GRPO with and without some of our key designs. (a) compares reward curves with and without hindsight return relabeling when processing sampled sub-trajectories. (b) compares the learning process of our adapted GRPO (using sub-trajectories) against naive GRPO (using complete trajectories). (c) shows the effect of using consistent states when sampling a group versus not. (d) illustrates the difference in the learning process when computing the importance ratio using sequence likelihood versus token likelihood. (e) compares the learning process with and without active selection for sampling reset points.

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 fundamentally 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 smaller. 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).

To address this, we remove the hindsight return relabeling step in online finetuning of DTs, thereby maintaining consistency between rollout and training distributions. As shown in Fig. 1a, removing hindsight return relabeling significantly improves stability and overall performance. In simpler environments such as Hopper, relabeling may yield transient gains but eventually leads to collapse, whereas in more complex environments such as Door, the model fails to learn altogether when relabeling is enabled.

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*

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

```
Input: Pretrained policy $\pi_{\theta_1}$, full trajectory buffer $\mathcal{T}_{\text{replay}}$, sub-trajectory buffer $\mathcal{T}_{\text{sub}}$, number of iterations $T$,
    initial state $s_1$, initial RTG $g_{\text{online}}$, number of reset points in a trajectory $K$, sub-trajectory length $L_{\text{traj}}$,
    evaluation steps $L_{\text{eval}}$, group size $G$ for GRPO.
1: for iteration $t = 1, \cdots, T$ do
2:     Rollout a full trajectory $\tau$ using the current policy $\pi_{\theta_t}(\cdot \mid s_1, g_{\text{online}})$, conditioned on initial state $s_1$ and
       RTG $g_{\text{online}}$; update $\mathcal{T}_{\text{replay}}$ with $\tau$. // Collect complete policy; buffer updated in a FIFO manner.
3:     Sample a minibatch $\mathcal{B}$ of full trajectories from $\mathcal{T}_{\text{replay}}$ from distribution $p$ with $p(\tau) := \frac{|\tau|}{\sum_{\tau \in \mathcal{T}} |\tau|}$.
4:     for each full trajectory $\tau \in \mathcal{B}$ do
5:         Sample $K$ reset points $\{s_k\}_{k=1}^{K}$ from action-variance distribution.
6:         For each reset point $s_k$, generate $G$ sub-trajectories $\{\tau_{k_i}^{\text{sub}}\}_{i=1}^{G}$ of length $L_{\text{traj}}$ with the current policy
           $\pi_{\theta_t}$; evaluate each sub-trajectory for $L_{\text{eval}}$ more steps to get reward $R(\tau_{k_i}^{\text{sub}})$. // Sub-trajectory
           generation and evaluation.
7:         Compute the advantage $\widehat{A}_{k_i}$ for each sub-trajectory $\tau_{k_i}^{\text{sub}}$ using Eq. (2).
8:         Update the sub-trajectory buffer $\mathcal{T}_{\text{sub}}$ with $\{(\tau_{k_i}^{\text{sub}}, \widehat{A}_{k_i})\}_{i=1}^{|G|}$. // Buffer updated in a FIFO manner.
9:     Finetune the current policy with sub-trajectories in $\mathcal{T}_{\text{sub}}$ using the sequence-level importance ratio
       (Eq. (3)) to get a new policy $\pi_{\theta_{t+1}}$.
Output: Online finetuned policy $\pi_{\theta_{T+1}}$.
```

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 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 reinforcement learning 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. 1b, the model fails to learn when trained with full trajectories in the Ant-medium-v2 environment. This limitation is expected, as reinforcement learning 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 reset points from each full trajectory. For every reset point s_k, we generate G sub-trajectories $\{\tau_{k_i}^{\text{sub}}\}_{i=1}^{|G|}$ of length L_{traj} using the current policy π_{θ_t}. To better evaluate the quality of actions taken along each sub-trajectory, we then continue its rollout for an additional L_{eval} steps by executing the mean action (or the most-likely action in the discrete case). The cumulative discounted rewards $\{r_{k_i}^{\text{sub}}\}_{i=1}^{|G|}$ obtained over these $L_{\text{traj}} + L_{\text{eval}}$ steps are used to compute normalized advantages for GRPO:

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

Here, only the sub-trajectory of length L_{traj} is directly used for GRPO optimization, while the subsequent L_{eval} steps serve purely for evaluation. The parameter L_{traj} controls the granularity of credit assignment, and L_{eval} determines the quality of reward estimation. In practice, we find that using a small L_{traj} and a large L_{eval} yields the best performance; see Section 4.3 for ablations on these hyperparameters.

To ensure stable optimization, we enforce *state consistency* by resetting vectorized environments to the same initial states before generating sub-trajectories within each group. This reset mechanism is essential for convergence, as shown in Fig. 1c. In scenarios where environment resetting is infeasible, we train an auxiliary Q-function following TD3 (Fujimoto et al., 2018) to evaluate candidate actions under a shared state (see Appendix A.4 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—significantly improves model performance (Fig. 1d). Intuitively, the sequence-level importance ratio $\frac{\pi_\theta(\tau^{\mathsf{sub}}_{k_i}|s_k,g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau^{\mathsf{sub}}_{k_i}|s_k,g_k)}$ is better aligned with the advantage $\widehat{A}_{k_i}$, which is already defined at the sequence level, unlike the token-level ratio used in Eq. (1). Empirical validation in Fig. 1d further supports this intuition.

To incorporate sequence-level weighting into GRPO for Decision Transformers, we modify the original formulation in Eq. (1). For each reset point s_k with return-to-go g_k, let $\{\tau^{\mathsf{sub}}_{k_i}\}_{i=1}^{|G|}$ denote the group of sub-trajectories generated from s_k, and $\{\widehat{A}_{k_i}\}_{i=1}^{|G|}$ their corresponding advantages. The optimization objective becomes:

$$J_{\mathsf{GRPO}}(\theta) = \frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k)}\widehat{A}_{k_i},\, \mathsf{clip}\left(\frac{\pi_\theta(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau^{\mathsf{sub}}_{k_i} \mid s_k, g_k)}, 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_{k_i}\right) - \beta D_{\mathsf{KL}}(\pi_\theta \,\|\, \pi_{\mathsf{old}})\,. \tag{3}$$

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 a geometric mean over sequence-level importance ratios, we find that the formulation in Eq. (3) achieves superior performance when integrated with Decision Transformers.

Active sampling for restting points. During policy rollout, we observe that certain state s_t exhibit high variance in the predicted action distribution $\pi_\theta(\cdot \mid s_t, g_t)$. Such variance indicates uncertainty in the policy's behavior and suggests these states would benefit from additional exploration. We therefore introduce an *active sampling* mechanism that biases sub-trajectory sampling toward high-uncertainty states. For each full trajectory, we apply a softmax transformation to the per-step action variance $\sigma_t^2 := \mathsf{Var}(\pi_\theta(\cdot \mid s_t, g_t))$ to form a sampling distribution p with $p_t := \frac{\exp(\sigma_t^2)}{\sum_{k=1}^{|\tau|}\exp(\sigma_k^2)}$, and sample reset points according to p_t for sub-trajectory generation. This simple active selection mechanism prioritizes learning updates on the most uncertain regions of the trajectory, yielding faster and more stable convergence (Fig. 1e).

4 Experiments

In this section, we aim to answer three questions:

(i) How does our GRPO-DT (Algorithm 1) perform compared with existing algorithms?

(ii) Do pure RL gradients provide better signals compared with methods that prioritize supervised loss during DT online finetuning?

(iii) How does each component in our method affect the performance?

The model architecture and hyperparameter setting can be found in Appendix A.3.1.

4.1 Experimental Setups

Tasks and datasets. We evaluate on three types of continuous control and manipulation benchmarks from D4RL (Fu et al., 2020). (1) *Gym locomotion* (*Hopper*, *Walker2d*, *Ant*) (Todorov et al., 2012) with dense rewards, using *medium*, *medium-replay*, and *random* datasets. (2) *AntMaze* (v2) with sparse goal-reaching rewards (success = 1, else 0), using *umaze* and *umaze-diverse* datasets. (3) *Adroit* manipulation tasks (v1) (Rajeswaran et al., 2017) including *Door*, *Hammer*, and *Pen*, evaluated on *human* and *cloned* datasets. The *random* datasets consist of offline trajectories of low quality while others are of medium quality. Details of each environment are provided in Appendix A.1.

Baselines. In our experiments, we mainly compare our adapted GRPO-DT and PPO-DT with three baselines: Online Decision Transformer (**ODT**) (Chen et al., 2021), the widely adopted online version of Decision Transformer with supervised loss as online finetuning objective; **ODT+TD3** (Yan et al., 2024), the current state-of-the-art method for online finetuning of Decision Transformer; **IQL** (Kostrikov et al., 2021), a popular offline algorithm which also has an online variant.

Metrics. We use the normalized average reward of 3 random seeds according to D4rl's statistic (Fu et al., 2020) where higher rewards represent better performance. Meanwhile, we also present the learning curves which shows the change of the normalized rewards with respect to the training iterations. When presenting the curves, we set the x-coordinate to be the number of iteration. This variable is the *iteration* from line 3 of the Algorithm. 1 from ODT Zheng et al. (2022) paper. Note that conventional x-axis metrics, such as the number of online transitions (indicating sample efficiency) and the number of gradient updates (indicating computational cost), are not suitable for our setting. For gradient updates, ODT/ODT+TD3 requires nearly two orders of magnitude more updates per iteration compared to our methods GRPO-DT and PPO-DT; for online interactions, however, our adapted policy gradient methods consume several to tens of times more samples than ODT/ODT+TD3.[1] Hence, neither metric provides a fair comparison.

PPO-DT implementation. Our PPO-DT implementation follows the practice of CleanRL (Huang et al., 2022). Unlike prior work that applies PPO to multi-agent reinforcement learning (MARL) tasks with Decision Transformer (Meng et al., 2023), we train the critic using λ-returns rather than discounted Monte Carlo returns. Specifically, λ-returns combines multi-step returns with temporal-difference bootstrapping to balance bias and variance in value estimation. In addition, we store the action probabilities at sampling time instead of recomputing them during training.

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. 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 more challenging domains 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 low-quality offline data. The first block of Table 1 presents results obtained when pretraining on low-quality offline data. We observe that both our proposed methods, GRPO-DT and PPO-DT, perform substantially better than all other baselines on the *random* datasets. These datasets contain trajectories collected by an untrained random policy, which provide little meaningful supervision. As a result, pretraining on such data can introduce detrimental biases, often causing purely supervised methods to collapse or converge to suboptimal behaviors. In contrast, our adapted GRPO-DT and PPO-DT demonstrate strong robustness under these challenging conditions, achieving superior performance despite poor initialization. Meanwhile, ODT—relying solely on supervised learning signals—struggles to escape local optima, and IQL exhibits similar limitations.

Learning with medium-quality offline data. The remaining sections of Table 1 report results obtained when pretraining on medium-quality offline data. For the **Gym locomotion** tasks, our adapted methods,

[1]This is consistent with the conventional idea that temporal-difference (TD) learning is more sample efficient than policy gradients due to bootstrapped TD updates (Sutton et al., 1998).

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

		DT	IQL	ODT	ODT+TD3	PPO	GRPO-DT
Mujuco (random)	Ho-R-v2	1.98	42.73 (13.66)	30.43 (0.01)	83.32 (8.46)	**106.97 (0.96)**	99.20 (3.80)
	Wa-R-v2	4.59	15.92 (3.54)	10.88 (0.34)	82.95 (18.28)	**108.69 (8.86)**	100.25 (33.19)
	An-R-v2	30.38	59.65 (23.26)	19.08 (3.97)	80.58 (7.25)	107.45 (22.83)	**120.69 (5.47)**
	Average	12.32	39.43	20.13	82.28	**107.70**	**106.71**
Mujuco (medium)	Ho-M-v2	63.1	61.49 (33.33)	98.02 (0.63)	101.47 (2.29)	105.65 (5.43)	**108.81 (0.85)**,
	Ho-MR-v2	29.76	98.36 (0.62)	87.73 (0.59)	**107.94 (2.29)**	**109.60 (1.63)**	83.61 (20.75)
	Wa-M-v2	70.78	102.28 (1.04)	76.49 (0.78)	103.27 (5.95)	109.49 (9.04)	**158.34 (3.75)**
	Wa-MR-v2	58.06	104.27 (3.64)	74.21 (2.41)	102.80 (2.68)	117.45 (14.79)	**137.36 (5.64)**
	An-M-v2	90.58	118.18 (2.42)	90.71 (0.03)	131.56 (0.41)	139.84 (0.95)	**147.51 (2.44)**
	An-MR-v2	78.15	117.51 (0.82)	83.63 (0.87)	120.01 (2.94)	117.95 (2.54)	**142.05 (3.32)**
	Average	65.07	100.35	85.13	111.175	116.66	**129.61**
Adroit	D-C-v1	4.97	0.10 (0.06)	1.26 (1.02)	79.98 (5.62)	0.19 (0.00)	**96.41 (7.59)**
	D-H-v1	9.30	17.18 (0.75)	8.76 (3.87)	79.73 (4.37)	**94.12 (3.99)**	89.33 (10.12)
	P-C-v1	75.02	63.09 (14.38)	16.24 (5.12)	**109.86 (6.27)**	27.14 (0.24)	**111.15 (2.61)**
	P-H-v1	95.23	24.94 (1.48)	19.84 (7.42)	77.18 (7.42)	9.92 (5.00)	**85.11 (6.08)**
	H-C-v1	1.80	9.56 (8.13)	1.32 (0.06)	119.95 (2.45)	130.60 (2.81)	**140.45 (1.93)**
	H-H-v1	1.01	0.74 (0.37)	0.91 (0.22)	120.93 (2.18)	129.23 (2.18)	**132.64 (12.56)**
	Average	31.22	19.27	8.06	97.93	65.2	**109.18**
Antmaze	U-v2	16.00	91.21 (2.14)	89.27 (3.73)	**99.64 (0.20)**	0.00 (0.00)	96.07 (0.53)
	UD-v2	38.00	0.00 (0.00)	63.81 (1.64)	**99.42 (0.43)**	47.00 (4.00)	97.70 (2.67)
	Average	27	45.61	76.54	**99.53**	23.50	96.89

GRPO-DT and PPO-DT, achieve the highest returns, while ODT+TD3 remains competitive and ODT performs reasonably well. In **Adroit**, where both the state and action spaces are substantially larger and more complex, policies are prone to degradation or collapse during finetuning. Under these challenging conditions, ODT and IQL fail to improve upon their pretrained performance, whereas our adapted GRPO consistently attains higher returns, demonstrating strong exploration and stability. ODT+TD3 achieves competitive performance on several Adroit environments but does not match the robustness of our approach across the board. Similarly, PPO-DT performs strongly on certain tasks yet fails to improve in others For the **AntMaze** domain, where rewards are sparse, ODT+TD3 achieves the best performance, while our adapted GRPO remains competitive. Other methods fail to make meaningful progress in this setting.

Advantages over previous methods. Beyond final performance, our adapted GRPO offers several practical advantages over existing approaches. First, unlike methods that rely on an auxiliary critic, our approach introduces no additional networks, making it substantially simpler to implement. Second, by leveraging accurate gradient estimation through sub-trajectory sampling, our method is more computationally efficient and requires far fewer gradient updates per iteration. For instance, our approach performs only 8×256 gradient updates per iteration, compared to approximately 256×300 updates required by ODT or ODT+TD3, representing a significant reduction in compute cost. Finally, our method can finetune any pretrained DT-style model with minimal modification (see Appendix A.5 for experiments), whereas prior approaches such as ODT+TD3 often require *modifying the offline pretraining loss* to incorporate RL gradients and jointly training an auxiliary Q-function. This dependency prevents them from directly finetuning already pretrained models, limiting their flexibility and scalability.

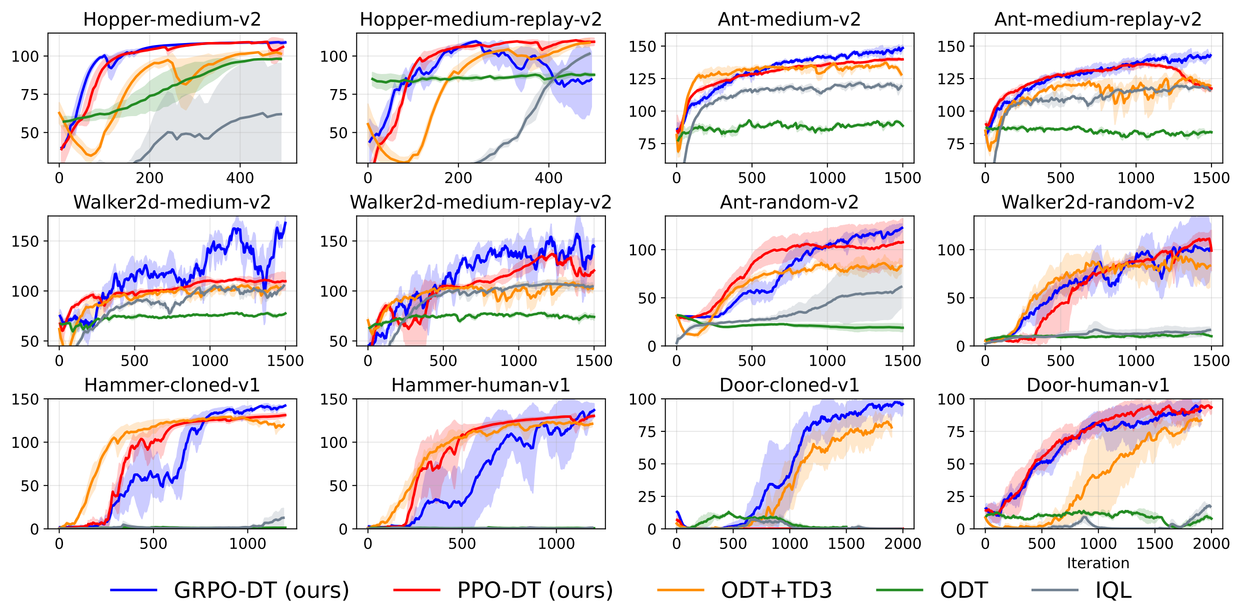


Figure 2: Performance comparison across different RL environments. Our proposed method GRPO-DT achieves the best performance across the majority of tasks. PPO-DT performs competitively in most cases but fails in certain environments. ODT+TD3 achieves overall decent performance but is generally outperformed by GRPO-DT. Both ODT and IQL consistently underperform across most environments.

4.3 Analyses and Ablations

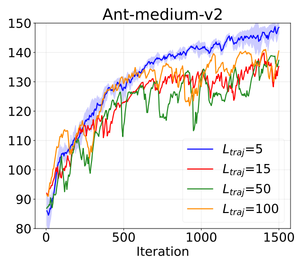
Ablation on sub-trajectory length. Sub-trajectory in our method represents the unit for assigning advantage. Thus its length is crucial to our algorithm. Empirical results in Fig. 3a confirm that increasing sub-trajectory length destabilizes training and leads to inferior outcomes. However, excessively short sub-trajectories, while stable, also yield sub-optimal results. This is likely because very short trajectories sampled from the same state distribution are overly homogeneous, limiting their ability to provide informative learning signals for RL finetuning.

Ablation on sub-trajectory evaluation steps. For each sub-trajectory, we extend the rollout with L_{eval} additional evaluation steps ranging from 30 to 400, depending on the environment. As illustrated in Fig. 3b, longer evaluation steps enable more reliable assessment of sub-trajectory quality and as a result consistently improve model performance.

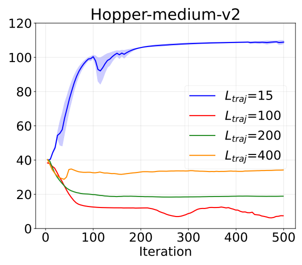
Q-function-guided GRPO-DT without resetting. In scenarios where resetting the environment is infeasible, we instead train an additional Q-function using TD3 and apply GRPO-DT with Q-function-guided advantages (detailed in Algorithm 2 in Appendix A.4). As shown in Fig. 3c, this Q-function-guided variant still achieves decent performance without state resetting.

5 Related Work

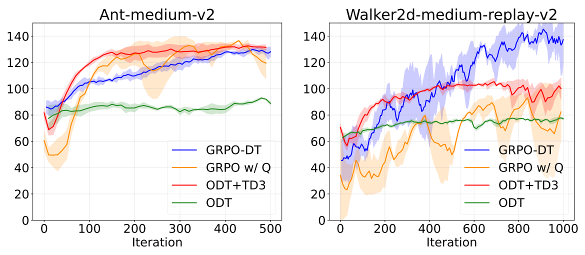
Transformers for RL. With transformers becoming the dominant architecture in both CV and NLP, a growing number of transformer-based approaches have been proposed in the RL community (Lin et al., 2023; Chen et al., 2022; Yuan et al., 2024). Owing to their strong capability in modeling sequential dependencies (Parisotto and Salakhutdinov, 2021), transformers are naturally suited for reinforcement learning when formulated as a sequence modeling problem (Chen et al., 2021; Janner et al., 2021; Wang et al., 2022). In this paradigm, models typically condition on past states, actions, and returns to autoregressively predict future



(a) Ablation on L_{traj}



(b) Ablation on L_{eval}



(c) Q-function-guided GRPO-DT

Figure 3: Panel (a) shows ablation on sub-trajectory length L_{traj}. Both longer and shorter sub-trajectory length lead to inferior results. Panel (b) shows ablation on evaluation steps L_{eval}. Inadequate evaluation steps lead to model collapse. Panel (c) shows training with our variant described in Algorithm 2. It achieves decent results.

actions. However, such approaches rely on offline datasets and often suffer from issues of data scarcity and out-of-distribution problem. This motivates the offline pretraining followed by online finetuning paradigm. Nevertheless, existing works either treat supervised objectives as the primary training signal when tuning transformers online (Zheng et al., 2022; Yan et al., 2024), rely on Q-learning rather than transformer-based architectures (Lee et al., 2022; Zheng et al., 2023; Song et al., 2022; Yu and Zhang, 2023; Nair et al., 2020), or are situated in MARL settings (Meng et al., 2023). In contrast, our work focuses on online finetuning of offline-pretrained decision-making transformers using purely RL-based gradients.

RL for transformers. Reinforcement learning has also emerged as a powerful technique for aligning and enhancing large language models (LLMs) (Ouyang et al., 2022; Lee et al., 2023). A wide spectrum of algorithms has been explored, ranging from policy gradient methods such as PPO, to off-policy methods like Implicit Language Q-Learning (ILQL) (Snell et al., 2022) and VerifierQ (Qi et al., 2024), as well as reward-model-free methods such as DPO (Rafailov et al., 2023) and KTO (Ethayarajh et al., 2024). More recently, novel algorithms such as GRPO and approaches like ReFT (Luong et al., 2024) have been proposed to further improve the reasoning ability of LLMs. RL methods have also been applied to transformer-based multi-modal models (Liu et al., 2025; Shen et al., 2025). However, the strategies designed for training LLMs cannot be directly transferred to finetuning Decision Transformers, as decision-making tasks fundamentally differ from language generation in terms of environment dynamics, reward distributions, and optimization objectives. To this end, our work adapts RL algorithms widely adopted in LLMs, specifically GRPO and PPO, to the context of finetuning Decision Transformers.

6 Conclusion

We present a systematic study on applying pure reinforcement learning gradients for online finetuning of Decision Transformers. We identify *hindsight return relabeling* as the central obstacle preventing the use of importance sampling-based policy gradients and propose a principled solution by adapting GRPO to the DT framework. Our adapted algorithm incorporates sub-trajectory optimization, sequence-level importance ratios, and active state sampling, enabling effective online finetuning of pretrained DTs. We further extend PPO to this setting and show that pure RL gradients substantially enhance DT performance across a wide range of benchmarks.

Our findings highlight the promise of unifying reinforcement learning and sequence modeling under a shared transformer framework. We believe this work opens up new directions for scaling Decision Transformers through pure RL optimization, bridging advances in language modeling and reinforcement learning toward more capable and adaptable decision making policies.

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.

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

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.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

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.

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.

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.

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.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. *arXiv preprint arXiv:2310.08566*, 2023.

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*, 2025.

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.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

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.

Emilio Parisotto and Ruslan Salakhutdinov. Efficient transformers in reinforcement learning using actor-learner distillation. *arXiv preprint arXiv:2104.01655*, 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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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.

Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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.

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.

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.

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.

Weilin Yuan, Jiaxing Chen, Shaofei Chen, Dawei Feng, Zhenzhen Hu, Peng Li, and Weiwei Zhao. Transformer in reinforcement learning for decision-making: a survey. *Frontiers of Information Technology & Electronic Engineering*, 25(6):763–790, 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.

Han Zheng, Xufang Luo, Pengfei Wei, Xuan Song, Dongsheng Li, and Jing Jiang. Adaptive policy learning for offline-to-online reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11372–11380, 2023.

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

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.

A Appendix

A.1 Environmental and Dataset Details

Our experiments cover three types of continuous control and manipulation benchmarks from D4RL (Fu et al., 2020). The first type includes the Gym locomotion environments *Hopper*, *Walker2d*, and *Ant* (all in v2) (Todorov et al., 2012), which provide dense reward signals. For these environments, we evaluate on the *medium*, *medium-replay*, and *random* datasets. The *medium* dataset consists of trajectories generated by a policy trained to roughly one-third the performance of an expert policy. The *medium-replay* dataset is constructed from the replay buffer of a policy trained to medium-level performance, and the *random* dataset consists of trajectories generated by an untrained random policy. The second type focuses on sparse-reward goal-reaching problems in the *AntMaze* domain (v2). Here, the agent controls an Ant robot to reach a designated target location, receiving a reward of 1 upon success and 0 otherwise. We use the *umaze* and *umaze-diverse* datasets. The third type includes robotic manipulation tasks from the *Adroit* benchmark (v1) (Rajeswaran et al., 2017), including *Door*, *Hammer*, and *Pen*. These are high-dimensional tasks with challenging dynamics. We experiment with both the *human* and *cloned* datasets, where the former is collected from human teleoperation and the latter from behavior cloning policies.

A.1.1 Mujuco Environments

We conduct our experiment on three Mujuco environments:

- **Hopper.** Hopper is a Mujuco-based single-legged locomotion task where the agent controls three joints to make the robot hop forward while maintaining stability. The action space is 3-dimensional continuous, 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 survival bonus, forward progress, and a control cost penalty proportional to the squared magnitude of the action. Episodes terminate when the agent falls or reaches the maximum horizon (default 1000 steps).
- **Walker2d.** Walker2D is a 2D 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 (torques in $[-1, 1]$) applied to hinge joints. The observation space has 17 dimensions. At each timestep, the agent receives a reward composed of (i) a "healthy" survival bonus, (ii) a forward progress reward proportional to the displacement in the x-direction, and (iii) a control cost penalty proportional to the magnitude of the action. Episodes terminate if the robot becomes unhealthy (e.g. torso height out of range, non-finite states) or reaches the maximum horizon.
- **Ant.** The Ant task is a 3-dimensional locomotion problem where the agent controls an 8-joint quadruped to move forward while maintaining balance. The action space is an 8-dimensional continuous vector (typically bounded in $[-1, 1]$). The observation space comprises the robot's positional and velocity state (and sometimes contact observations). Each timestep the agent receives a reward combining a forward-progress term (displacement in the x-axis), a control cost penalty (proportional to the squared action magnitude), and often an alive bonus. Episodes terminate when the ant falls or the time horizon (default 1000) is reached.

The size and normalized return of each offline dataset is presented in Table 2.

A.1.2 Adroit Environment

We choose three Adroit environments to experiment:

- **Door.** The Door task requires a 28-DoF hand-arm system to unlatch and open a door. The action space is 28-dimensional continuous, with each joint command scaled to $[-1, 1]$ The observation space has 39 dimensions, including joint states, latch status, and relative positions between the hand and handle. The dense reward combines distance penalties, velocity regularization, and bonuses for increasing door hinge displacement, encouraging successful door opening.

Table 2: The size and normalized rewards of offline dataset used in Mujuco environment.

Dataset	Size	Normalized Reward
Hopper-medium-v2	999906	44.32±12.27
Hopper-medium-replay-v2	402000	14.98±16.32
Hopper-random-v2	999906	1.19±1.16
Walker2d-medium-v2	999995	62.09±23.83
Walker2d-medium-replay-v2	302000	14.84±19.48
Walker2d-random-v2	999997	0.01±0.09
Ant-medium-v2	999946	80.30±35.82
Ant-medium-replay-v2	302000	30.95±31.66
Ant-random-v2	999930	6.36±10.07

- **Hammer.** The Hammer task involves a 28-DoF robotic hand-arm system (a 24-DoF ShadowHand plus a 4-DoF arm) that must pick up a hammer and drive a nail into a board. The action space is 26-dimensional continuous, representing joint commands (scaled into $[-1, 1]$. The observation space is 46-dimensional, encoding joint states, poses of the hammer and nail, and forces on the nail. The reward combines terms for progress in driving the nail (hinge displacement or insertion depth), penalties on control magnitude, and distance-based cost.
- **Pen.** The Pen task requires a 24-degree-of-freedom robotic hand to manipulate a pen into a target orientation. The action space is 24-dimensional continuous, with joint commands scaled to $[-1, 1]$ for each actuator. The observation space is 45-dimensional, including joint states, pen pose, and the goal orientation. The reward is composed of a negative penalty proportional to the Euclidean distance between the pen and target, an orientation similarity term (dot product between real and target orientation), proximity bonuses when both distance and angular alignment are sufficiently tight, and a dropping penalty if the pen falls.

The corresponding offline dataset quality can be found in Table 3.

Table 3: The size and normalized rewards of offline dataset used in Adroit environment.

Dataset	Size	Normalized Reward
Pen-cloned-v1	499886	108.63± 122.43
Pen-human-v1	4800	202.69± 154.48
Hammer-cloned-v1	999872	8.11± 23.35
Hammer-human-v1	10948	23.80± 33.36
Door-cloned-v1	999939	12.29± 18.35
Door-human-v1	6504	28.35± 13.88

A.2 Antmaze Environment

The Umaze environment in Antmaze places an Ant quadruped in a U-shaped maze. The action space is 8-dimensional continuous, with torques in $[-1, 1]$. The observation space is a goal-aware dictionary: a 27-dimensional "observation" vector (positions and velocities of the Ant body parts), plus 2D achieved goal and desired goal vectors indicating the Ant's torso position and the target goal in the plane. The reward provide is sparse: 0 if the ant hasn't reached its final target position, and 1 if the ant is in the final target position (the ant is considered to have reached the goal if the Euclidean distance between both is lower than 0.5 m). The quality of the offline datasets used is presented in Table 4.

Table 4: The size and the average and standard deviation of the normalized reward of the Antmaze datasets used in our experiments.

Dataset	Size	Normalized Reward
Antmaze-Umaze-v2	998573	86.14± 34.55
Antmaze-Umaze-Diverse-v2	999000	3.48± 18.32

A.3 Experimental Details

A.3.1 Hyperparameters

Table 5 shows the hyperparameters that are common across all our experiments and Table 6 summarizes the domain-specific hyperparameters for each environment and dataset for GRPO. For antmaze-environment, following ODT+TD3's (Yan et al., 2024) practice, We remove all 1-step trajectories, because the size of the replay buffer for decision transformers is controlled by the number of trajectories, and antmaze dataset contains a large number of 1-step trajectories due to its data generation mechanism (immediately terminate an episode when the agent is close to the goal, but do not reset the agent location). And we did not add positional embedding as suggested by ODT (Zheng et al., 2022).

For GRPO, we collect 1 complete trajectory fpr replay buffer per iteration in Mujoco and Antmaze environments and 5 complete trajectories each iteration in Adroit environments. The buffer size for the complete trajectories is 32. When doing resetting, we sample 16 trajectories from the complete trajectories buffer. We choose four reset points for each trajectory and the group size for each trajectory is 8. This results in 512 sub-trajectories per iteration. The buffer size for this sub-trajectories is 2048.

For PPO, we collect 8 trajectories for Mujuco and Antmaze environment and 16 trajectories for Adroit each iteration. The buffer size is 4 times of the number of trajectories collected per iteration. Following ODT+TD3's practice, we add Layernorm (Ba et al., 2016) to the critic of PPO in Adroit and Antmaze environment to stabilize training process.

Table 5: The common hyperparameters in our experiments

Hyperparameters	Value
Number of layers	4
Number of attention heads	4
Embedding dimension	512
Actor Optimizer	LAMB (You et al., 2019)
Dropout	0.1 when pretraining, disabled when finetuning
Nonlinearity function	SiLU (Elfwing et al., 2018)
Weight decay	0.0001
Gradient norm clip	0.5
Target entropy	-dim($\mathcal{A}$)
PPO Critic layer	2
PPO Critic hidden size	256 for Mujoco, 512 for others
PPO Critic activation	SiLU
PPO Critic Optimizer	AdamW (Loshchilov and Hutter, 2017)
PPO discount factor γ	0.99

A.4 Q-function-guided GRPO-DT

In this section we introduce GRPO with Q, an action-level variant of our method designed for settings where environment resets are infeasible. Instead of generating multiple sub-trajectories from the same state, our method samples a group of actions under the current policy for each visited state and evaluates them with an auxiliary Q-function. The resulting Q-values are normalized to provide advantages, which are then used to optimize the policy via the GRPO objective. Meanwhile, the Q-function is updated following standard TD3

Table 6: The hyperparameters that we use to finetune DT in each domain, where T_{train} and T_{eval} stands for context length for training and evaluation, γ is the discount factor, lr_a represents learning rate for the actor, L_{traj} and L_{eval} represent sub-trajectory length and evaluation steps for each sub-trajectory respectively, ε is Clipping threshold,ε_{GRPO} is the minimum deviation of a sub-trajectory's raw reward from the mean reward of its group, ETPR is the initial entropy temperature for online finetuning.

Environ	BS	T_{train}	T_{eval}	RTG	γ	lr_a	L_{traj}	L_{eval}	ε	ε_{GRPO}	ETPR
Ho-M(R)	256	20	1	7200	0.995	5e-5	15	400	0.2	2.0	0.20
Ho-R	256	20	1	7200	0.995	5e-5	15	400	0.2	2.0	0.20
Wa-M(R)	256	20	1	10000	0.995	5e-5	15	400	0.3	2.0	0.04
Wa-R	256	20	1	10000	0.995	5e-5	15	400	0.3	2.0	0.20
An-M(R)	256	20	1	12000	0.995	5e-5	15	200	0.3	2.0	0.04
An-R	256	20	1	12000	0.995	5e-5	15	200	0.3	2.0	0.20
D-C	512	5	1	3000	0.99	3e-5	10	100	0.3	0.5	0.10
D-H	512	5	1	3000	0.99	3e-5	10	100	0.3	0.4	0.04
P-C	512	5	1	6000	0.99	3e-5	3	30	0.3	0	0.02
P-H	512	5	1	6000	0.99	3e-5	3	30	0.3	0	0.02
H-C	512	5	5	4000	0.99	3e-5	10	100	0.3	0	0.05
H-H	512	5	5	4000	0.99	3e-5	10	100	0.3	0.8	0.05
U	256	5	1	2	1.0	5e-5	10	200	0.2	0	0.05
UD	256	1	5	2	1.0	5e-5	10	200	0.2	0	0.05

practice. This design preserves the core idea of group-based policy optimization while eliminating the need for environment reset.

Algorithm 2 Q-function-guided GRPO-DT (action-level variant)

```
Input: Pretrained policy $\pi_\theta$, trajectory buffer $\mathcal{T}_{\mathsf{replay}}$, auxiliary Q-function $Q_\phi$, total rounds $T$, group size $G$, discount factor $\gamma$.
1: for round $t = 1, \cdots, T$ do
2:     Rollout trajectory $\tau$ using current policy $\pi_\theta(\cdot|s, g)$; update $\mathcal{T}_{\mathsf{replay}}$ with $\tau$. // Trajectory collection with FIFO buffer update.
3:     Sample a minibatch $\mathcal{G}$ from $\mathcal{T}_{\mathsf{replay}}$ with probability $p(\tau) \propto |\tau|$.
4:     for each $\tau \in \mathcal{G}$ do
5:         For each state $s_h$ in $\tau$, sample $G$ actions $\{a_{h,i}\}_{i=1}^{G} \sim \pi_\theta(\cdot|s_h, g_h)$.
6:         Evaluate each sampled action with $Q_\phi(s_h, a_{h,i})$.
7:         Normalize scores $\{Q_\phi(s_h, a_{h,i})\}$ to obtain advantages $\{\widehat{A}_{h,i}\}$. // Action-level evaluation with Q-function.
8:         Update policy $\pi_\theta$ using GRPO objective with advantages $\{\widehat{A}_{h,i}\}$.
9:         Update $Q_\phi$ following TD3-style critic learning.
```

A.5 Training with Other Architectures

To evaluate the generality of our algorithm, we further apply it to other DT-style architectures. *Reinformer* (Zhuang et al., 2024) is a max-return sequence modeling approach for offline reinforcement learning. It integrates the RL objective of return maximization into supervised sequence modeling by using expectile regression to predict the in-distribution maximum return, which then guides optimal action generation. This method enhances trajectory stitching capability and achieves state-of-the-art performance among sequence models on the D4RL benchmark, particularly on tasks requiring learning from suboptimal data. The training process of applying our adapted GRPO to this architecture is presented in Fig. 4.

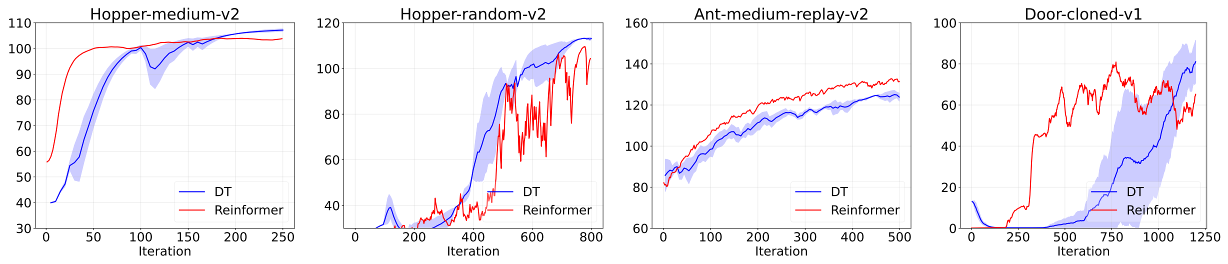


Figure 4: Applying our GRPO-DT to different architectures: standard DT vs. *Reinformer*.