Online Finetuning Decision Transformers with Pure RL Gradients

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

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

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 DTs.

1 Introduction

The transformer architecture [49] 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 [36, 8, 33, 1, 10]. Inspired by this success, the *Decision Transformer* (DT) [9] 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 [20] while conditioning on return-to-go (RTG) tokens.

The *Online Decision Transformer* (ODT) [62] 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 [15] gradients, resulting in ODT+TD3 [54]. 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) [44]—can substantially improve reasoning and alignment [16, 47]. This contrast naturally raises an important question:

[†]Project lead and corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

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* [62], while beneficial for 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 (Figure 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 DTs.

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 [31, 22] 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, 42) to Decision Transformers (PPO-DT), achieving competitive results where prior PPO-based approaches failed [54].

Contributions. We make the following contributions:

(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 our pure RL finetuning methods outperform existing online DT baselines and achieve 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, and Section 5 concludes the paper. Related work, additional experiment details and results are deferred to the Appendix due to space limitations.

2 Preliminaries

MDPs and Decision Transformers. We formulate the environment as a Markov Decision Process (MDP) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S}, \mathcal{A}$ denote state and action spaces, $P(s_{h+1} \mid s_h, a_h)$ the transition dynamics, and $R(s_h, a_h)$ the reward function. The objective is to maximize the cumulative reward $\sum_{h=1}^{H} r_h$. Decision Transformer (DT, 9) reformulates this as conditional sequence modeling. Specifically, DT conditions action generation on the *return-to-go* (RTG) $g_h := \sum_{h'=h}^{H} r_{h'}$. Given a context length m, the model takes a trajectory segment $((g, s, a)_{h-m+1:h-1}, g_h, s_h) := (g_{h-m+1}, s_{h-m+1}, a_{h-m+1}, \ldots, g_h, s_h)$ as input to autoregressively predict a_h, typically trained via a mean squared error (MSE) loss. For brevity, we denote this history-conditioned policy as $\pi(a_h \mid s_h, g_h)$. During online evaluation, the agent interacts with the environment conditioned on a desired target RTG (denoted as g_{online}).

Online Finetuning of DTs. Online Decision Transformer (ODT, 62) adapts DTs to online settings by finetuning on iteratively collected data. Crucially, ODT employs *hindsight return relabeling* [2]—replacing the target RTG with the realized return during updates—while strictly adhering to a supervised likelihood objective. However, purely supervised objectives often struggle with low-quality offline data as they fail to explicitly optimize returns [54]. More precisely, the supervised

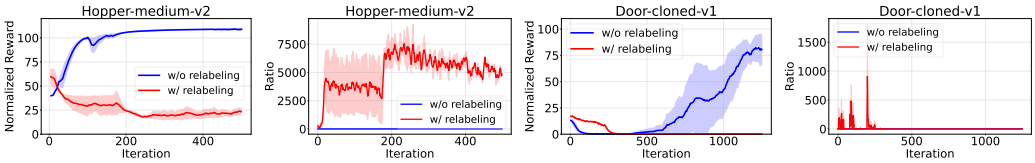


Figure 1: Comparison of GRPO-DT with and without hindsight return relabeling when applied to sampled sub-trajectories. Hindsight return relabeling causes the importance-sampling ratio $\frac{\pi_\theta(a|s,g)}{\pi_{\theta_{\text{old}}}(a|s,g)}$ to become highly unstable, which in turn degrades performance.

learning objective learns $\frac{\partial a}{\partial \text{RTG}}$, i.e., how action changes as the target RTG varies. However, what actually drives online policy improvement is $\frac{\partial \text{RTG}}{\partial a}$, i.e., how RTG responds to action adjustments. To address this, ODT+TD3 [54] augments ODT with auxiliary gradients from TD3 [15]. Nevertheless, ODT+TD3 remains dominated by the supervised loss (assigning only small weights to RL gradients) and relies on hindsight relabeling, which we identify as a source of instability for importance-sampling-based RL.

Group Relative Policy Optimization (GRPO). GRPO is a reinforcement learning algorithm initially proposed for large language models (LLMs) finetuning [44, 16]. It simplifies Proximal Policy Optimization (PPO) [42] 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 $\pi_{\theta_{\text{old}}}$. A reward r_i is computed for each response o_i, usually with the help of a reward model. GRPO uses $\widehat{A}_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}$ as the advantage of the i-th rollout, and $w_{i,h}(\theta) = \frac{\pi_\theta(o_{i,h}|q, o_{i,<h})}{\pi_{\theta_{\text{old}}}(o_{i,h}|q, o_{i,<h})}$ as the importance sampling ratio. The optimization also involves a KL penalty term $D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\theta_{\text{ref}}})$ to prevent large deviations from a reference policy $\pi_{\theta_{\text{ref}}}$.

3 Methods

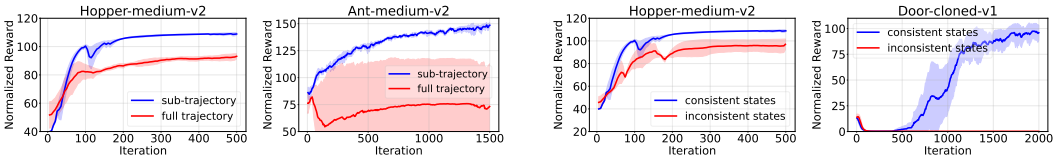
In Section 3.1, we first analyze a key limitation of adapting Decision Transformers to the online setting using importance-sampling-based reinforcement learning algorithms, and discuss how this limitation can be addressed. Building on these insights, Section 3.2 presents our adaptation of GRPO to Decision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

3.1 Removing Hindsight Return Relabeling

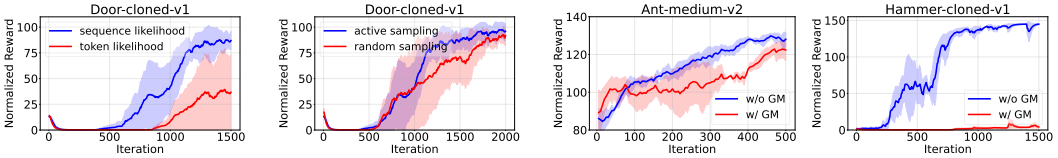
When deploying DTs, the learner must specify a desired initial RTG, since the ground-truth future RTG is unknown in advance. In Online Decision Transformer (ODT), the learner typically sets a relatively high target RTG g_{online} during rollout to encourage optimistic exploration. During training, a key component of ODT—known as *hindsight return relabeling*—replaces the RTG tokens based on the actual achieved RTG g_{actual} [62].

When augmenting the supervised ODT objective with RL gradients from TD3, ODT+TD3 [54] also adopts the hindsight return relabeling step. Yan et al. [54] further attempt to perform online finetuning of DTs using PPO [42]—an importance sampling-based RL algorithm—but find that PPO gradients lead to poor performance.

While hindsight return relabeling works well under supervised learning objectives (see Fig. 5.4 in Zheng et al. [62]), 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 [54]. As shown in Figure 1, removing hindsight return relabeling significantly improves stability and overall performance. In simpler environments such as



(a) Sub-trajectory vs. full trajectory (b) Consistent vs. inconsistent states



(c) Sequence vs. token (d) Active vs. random (e) Removing geometric mean

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.

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.

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 [44, 16]. 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 Equation (1). These sub-trajectories and their advantages are then used to update the policy with a *sequence-level importance ratio* described in Equation (2).

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 ablations 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 [44, 16], 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 Figure 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 DTs 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:

$$\widehat{A}_{k_i} = \frac{r_{k_i}^{\mathsf{sub}} - \mathrm{mean}(\{r_{k_1}^{\mathsf{sub}}, r_{k_2}^{\mathsf{sub}}, \ldots, r_{k_{|G|}}^{\mathsf{sub}}\})}{\mathrm{std}(\{r_{k_1}^{\mathsf{sub}}, r_{k_2}^{\mathsf{sub}}, \ldots, r_{k_{|G|}}^{\mathsf{sub}}\})}. \tag{1}$$

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, as shown in Figure 2b. In scenarios where environment resetting is infeasible, we train an auxiliary Q-function using TD3 [15] to evaluate candidate actions under a shared state (see Appendix D.2 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 Figure 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 objective Equation (2). This sequence-level objective yields more stable and sample-efficient optimization, consistent with concurrent findings by Zheng et al. [61] in language modeling.[1]

$$J_{\mathsf{GRPO}}(\theta) = \mathbb{E}_{(\tau_{k_i}^{\mathsf{sub}}, \widehat{A}_{k_i}, (s_k,g_k))\sim \mathrm{unif}(\mathcal{T}_{\mathsf{sub}})}$$
$$\min\left(\frac{\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)}\widehat{A}_{k_i},\ \mathsf{clip}\left(\frac{\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)}{\pi_{\theta_{\mathsf{old}}}(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)}, 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_{k_i}\right) - \beta D_{\mathsf{KL}}\left(\pi_\theta \,\|\, \pi_{\theta_{\mathsf{ref}}}\right). \tag{2}$$

While Zheng et al. [61] propose using a geometric mean over sequence-level importance ratios, we find that removing geometric normalization (as in Equation (2)) leads to superior performance when combined with sub-trajectory optimization using relatively short rollout lengths (Figure 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 Equation (2), we impose an entropy constraint $H(\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)) \geq \rho$ to encourage exploration. Here, $H(\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k))$ denotes the average policy entropy computed over the sub-trajectory $\tau_{k_i}^{\mathsf{sub}}$, and ρ is a predefined lower bound. Following Zheng et al. [62], we relax this hard constraint using a primal-dual formulation $\min_{\kappa\geq 0}\ \max_\theta\ J(\theta) + \kappa\big(H(\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)) - \rho\big)$, where κ is the dual variable. In practice, this corresponds to augmenting Equation (2)

[1] Concurrent work by Zheng et al. [61] also uses sequence-level importance ratios, in the context of LLM post-training. Our setting and overall method differ, and we discuss the relationship and a key design distinction in Appendix A.

with an additional entropy term $\kappa H(\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k))$, while adaptively updating κ to ensure that the entropy constraint $H(\pi_\theta(\tau_{k_i}^{\mathsf{sub}} \mid s_k, g_k)) \geq \rho$ is approximately satisfied.

In addition, the reference policy $\pi_{\theta_{\text{ref}}}$ is periodically updated from the current policy, preventing the KL regularization from constraining the policy to a potentially poor initialization.

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 $\Sigma_\theta = \text{diag}(\sigma_{h,1}^2, \ldots, \sigma_{h,d}^2)$. We define $\sigma_h^2 := \frac{1}{d}\sum_{i=1}^{d} \sigma_{h,i}^2$, and apply a softmax transformation to obtain the sampling distribution $p_h := \frac{\exp(\sigma_h^2)}{\sum_{h'=1}^{|\tau|} \exp(\sigma_{h'}^2)}$. 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 (Figure 2d).

3.3 Extensions

We further extend our framework to broader settings.

- First, for scenarios where environment resetting is infeasible, we introduce *Q-guided GRPO-DT*, which replaces explicit sub-trajectory rollouts with value estimates from an auxiliary Q-function trained via TD3 [15] (Appendix D.2).
- Second, to demonstrate the generality of eliminating hindsight return relabeling, we adapt PPO [42] to Decision Transformers (**PPO-DT**). Unlike prior works [54], PPO-DT uses token-level importance ratios without relabeling to ensure consistent optimization objectives (Appendix D.3).

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 [14]: (i) **MuJoCo** [48] tasks, including *Hopper*, *Walker2d*, and *Ant*, with dense rewards, evaluated on the *medium*, *medium-replay*, and *random* datasets. (ii) **Adroit** manipulation tasks [38], including *Door*, *Hammer*, and *Pen*, evaluated on the *human* and *cloned* datasets. (iii) **AntMaze** [14] 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 C.

Baselines. We compare our adapted GRPO-DT and PPO-DT against four main baselines: (i) **Online Decision Transformer** (**ODT**; Zheng et al. [62]), the standard online extension of DT that uses a supervised loss as the finetuning objective; (ii) **ODT+TD3** [54], the current state-of-the-art method for online finetuning of Decision Transformers; (iii) **IQL** [23], a widely used offline RL algorithm for which we employ its *online variant* in our experiments; and (iv) **SAC-DT**, which finetunes the pretrained DTs using pure Soft Actor-Critic (SAC) [17] gradients without a supervised learning objective. For reference, we also report the performance of the offline pretrained **Decision Transformer** (**DT**) without online finetuning. Detailed hyperparameter settings are provided in Appendix E.1.

Metrics. Following D4RL [14], we report the normalized final reward for each algorithm, where higher values indicate better performance. All results are averaged over three runs, with standard

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	SAC-DT	ODT	ODT+TD3	PPO-DT	GRPO-DT
MuJoCo (random)	Ho-R-v2	2.13	41.02 ±13.35	57.77 ±7.13	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	11.95 ±2.80	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	8.31 ±0.11	19.08 ±3.97	80.58 ±7.25	107.45 ±22.83	**120.69 ±5.47**
	Average	12.69	40.82	26.01	20.13	82.28	**107.70**	106.71
MuJoCo (medium)	Ho-M-v2	46.46	74.19 ±20.25	85.22 ±5.20	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	71.15 ±2.37	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	67.54 ±3.78	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	47.97 ±5.84	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	114.38 ±0.11	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	114.90 ±0.36	83.63 ±0.87	120.01 ±2.94	117.95 ±2.54	**142.05 ±3.32**
	Average	59.88	102.21	83.53	85.13	111.18	116.66	**129.61**
Adroit	D-C-v1	0.14	46.72 ±0.30	6.34 ±3.95	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	2.92 ±1.23	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	19.38 ±0.46	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	-2.00 ±0.27	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	0.56 ±0.47	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	1.07 ±0.82	0.91 ±0.22	120.93 ±2.18	129.23 ±2.18	**132.64 ±12.56**
	Average	13.59	25.15	4.71	8.06	97.93	65.20	**109.18**
AntMaze	U-v2	50.00	91.21 ±2.14	0.00 ±0.00	89.27 ±3.73	**99.64 ±0.20**	0.00 ±0.00	96.07 ±0.53
	UD-v2	65.00	0.00 ±0.00	94.40 ±1.41	63.81 ±1.64	**99.42 ±0.43**	47.00 ±4.00	97.70 ±2.67
	Average	57.50	45.61	47.20	76.54	**99.53**	23.50	96.89

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. [62]. 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 optimization compute) 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. Replay-based baselines are therefore more sample-efficient, while our methods require less optimization compute; see Appendix G for end-to-end wall-clock time comparison.

To facilitate a fair comparison, we train all algorithms for a larger number of iterations than those reported in Yan et al. [54]. 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. PPO-DT performs competitively in several cases but fails in certain environments. ODT+TD3 achieves reasonable performance yet is generally outperformed by GRPO-DT. ODT, IQL and SAC-DT all underperform across most benchmarks, particularly on tasks with low-quality pretraining data, 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. [54].

Learning with random offline data. The first block of Table 1 evaluates model performance when pretrained on *random* offline datasets [14] before online exploration. The offline datasets

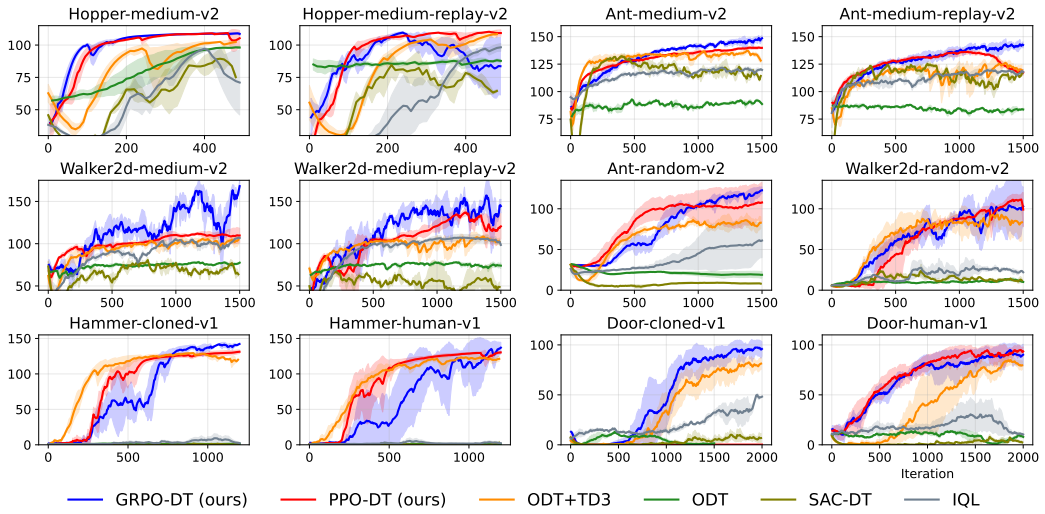


Figure 3: **Performance comparison across different RL environments.** Our proposed method GRPO-DT achieves the best performance across the majority of tasks. PPO-DT performs competitively in most cases but fails in certain environments. ODT+TD3 achieves overall decent performance but is generally outperformed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

consist of trajectories collected by an untrained policy and thus contain little meaningful supervision signal, serving as a standard stress test for offline-to-online adaptation and model robustness to poor pretraining quality [54].

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 TD 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 methods, which rely purely on reinforcement learning gradients. By directly optimizing reward-driven RL signals, our methods effectively recover from poor initialization in this low-quality pretraining regime.

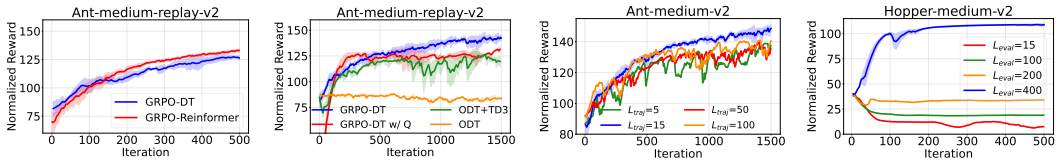
Learning with medium-quality offline data. The last three blocks of Table 1 report results obtained with medium-quality pretraining data before online exploration, representing realistic offline-to-online adaptation scenarios. On the **MuJoCo** tasks, our adapted methods, GRPO-DT and PPO-DT, achieve 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 after pretraining, whereas our adapted GRPO-DT consistently achieves strong results, obtaining the best performance on most 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. Other methods perform well on at most one of the two datasets: IQL and ODT on umaze, and SAC-DT on umaze-diverse.

Computational and practical advantages over ODT+TD3. Beyond achieving superior performance (as shown in Table 1 and Figure 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,



(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.

compared to roughly 256×300 updates required by ODT+TD3 (and ODT), substantially reducing optimization 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 conduct additional experiments to validate the generality and robustness of our framework.

Generality and Flexibility. We verify that GRPO-DT extends beyond standard DTs by effectively finetuning the *Reinformer* model [65], as shown in Figure 4a. Furthermore, for scenarios where environment resetting is infeasible, our **Q-guided variant** (using an auxiliary Q-function) maintains competitive performance without requiring explicit sub-trajectory rollouts (Figure 4b), demonstrating the method's applicability to diverse settings. We further validate the Q-guided variant on additional environments, with results provided in Appendix E.4 (Figure 5 (a) and (b)).

Hyperparameter Sensitivity. We analyze the impact of sub-trajectory formulation. Results in Figure 4c indicate that the sub-trajectory length L_{traj} requires a balance: excessively long trajectories destabilize training, while very short ones provide weak signals due to homogeneity. Conversely, increasing the evaluation length L_{eval} generally yields more reliable reward estimates and better performance (Figure 4d). We further extend both sensitivity analyses to the more challenging Door-cloned-v1 environment in Appendix E.4, where similar trends hold.

PPO-DT Failure Analysis. PPO-DT performs poorly on Door-cloned. To understand this result, we conduct additional runs and examine critic accuracy and policy-update statistics in failed and successful runs. The diagnostics suggest that poor initial rollouts and inaccurate critic estimates can form a feedback loop that prevents effective policy improvement, highlighting a limitation of critic-dependent advantage estimation. We present the diagnostic experiments and detailed analysis in Appendix E.3.

We additionally test whether our method preserves the long-horizon RTG-conditioned control ability of DTs. Results show that higher RTGs still induce better returns after finetuning, and performance gains persist beyond the early stage. Full results and analysis are provided in Appendix E.2.

5 Conclusion

In this work, we conducted a systematic study of online finetuning Decision Transformers using *pure RL 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.

Acknowledgments and Disclosure of Funding

YZ acknowledges support from NSF IIS 2425006.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL `https://arxiv.org/abs/1606.01540`.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

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

[19] 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`.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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*, 2025.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

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

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

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

[43] Burr Settles. Active learning literature survey. 2009.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

[60] 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.

[61] 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.

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

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

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

[65] 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 Related Work

Transformers for RL. With the transformer architecture [49] becoming the dominant paradigm in deep learning—most notably in language [36, 8] and vision [11]—a growing body of work has explored transformer-based approaches for reinforcement learning. In this paradigm, exemplified by Decision Transformers [9], 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 [21, 50, 53, 65]. While initially developed for offline RL, several recent works have extended these methods to the online setting, namely offline-to-online RL [25, 58, 32]. However, existing online finetuning approaches for Decision Transformers either rely purely on supervised sequence-modeling objectives [62, 52], or continue to prioritize the supervised loss by assigning only a small weight to reinforcement learning gradients [54]. 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*.

RL for transformers. Reinforcement learning has also emerged as a powerful approach for aligning and enhancing large language models [33, 5, 24] as well as multimodal models [27, 45]. A broad spectrum of algorithms has been explored, ranging from online policy gradient methods such as PPO and its variants [33, 29, 22], to offline reinforcement learning methods based on IQL and its variants [46, 35], as well as preference-based policy optimization approaches such as DPO and its extensions [37, 13]. More recently, substantial attention has been devoted to GRPO and its variants [44, 16, 26, 57], 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 by Zheng et al. [61], which appeared online in July 2025 and uses sequence-level importance ratios for LLM post-training. In contrast, our work addresses a different problem setting and incorporates sequence-level importance ratios as one component of a broader method. Our approach was developed independently, with an initial version made publicly available in September 2025, and differs from Zheng et al. [61] in a key design choice.

Specifically, while Zheng et al. [61] 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 (Figure 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 [43], where uncertainty-based methods are known to exponentially improve sample complexity compared to standard supervised learning [18, 34, 63, 64], and have demonstrated strong empirical performance with deep neural networks [3, 40, 60]. More recently, the core principles of active learning have been applied to reinforcement learning [55], large language models [6, 51], and multimodal models [59].

B Limitations and Future Work

Our method has several limitations. First, sub-trajectory rollouts assume access to environment resets. Resets are available in many domains, including perfect-information games, LLM reasoning, and simulator-based RL [31, 55], but they are not always feasible and may be difficult to implement in real-world applications. Our Q-guided variant (Appendix D.2) removes this requirement, but its performance depends on the quality of the learned Q-function. Second, our method uses more environment interactions than replay-based baselines, and long rollouts slow down training, especially when evaluation episodes are lengthy. Third, it may face challenges in sparse-reward or very long-horizon tasks, and it can require environment-specific hyperparameter tuning.

Future work includes developing reset-free strategies, improving robustness to simulator mismatch and the sim-to-real gap, scaling to more complex domains, and combining our approach with stronger architectures and exploration techniques to further improve performance and generalization.

C Environment and Dataset Details

We consider three continuous control and manipulation environments from D4RL [14]: *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** [48], 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-replay* 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 are 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.

Adroit 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 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 $\pm$ 12.27
Hopper-medium-replay-v2	402 000	14.98 $\pm$ 16.32
Hopper-random-v2	999 906	1.19 $\pm$ 1.16
Walker2d-medium-v2	999 995	62.09 $\pm$ 23.83
Walker2d-medium-replay-v2	302 000	14.84 $\pm$ 19.48
Walker2d-random-v2	999 997	0.01 $\pm$ 0.09
Ant-medium-v2	999 946	80.30 $\pm$ 35.82
Ant-medium-replay-v2	302 000	30.95 $\pm$ 31.66
Ant-random-v2	999 930	6.36 $\pm$ 10.07

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** [14], 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 latter 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.[2]

D Methodological Extensions

D.1 Detailed Algorithm

In this section, we present the complete training procedure of GRPO-DT in Algorithm 1. The algorithm proceeds iteratively. In each iteration, the agent first interacts with the environment to collect full trajectories conditioned on the optimistic return-to-go g_{online} (line 2). Instead of updating directly on these full trajectories, we employ a sub-trajectory optimization strategy. Specifically, we sample reset points from the collected data based on action variance to encourage exploration (line 5).

[2]Following ODT+TD3's [54] practice, we remove all 1-step trajectories in the dataset.

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.

1: **for** iteration $t = 1, \cdots, T$ **do**
2: 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.`
3: Sample a minibatch $\mathcal{B}$ of full trajectories from $\mathcal{T}_{\mathsf{replay}}$ from distribution p with $p(\tau) := \frac{|\tau|}{\sum_{\tau \in \mathcal{T}} |\tau|}$.
4: **for** each full trajectory $\tau \in \mathcal{B}$ **do**
5: Sample K reset points $\{(s_k, g_k)\}_{k=1}^{K}$ from action-variance distribution.
6: 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.`
7: Compute the advantage $\widehat{A}_{k_i}$ for each sub-trajectory $\tau_{k_i}^{\mathsf{sub}}$ using Equation (1).
8: 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.`
9: **end for**
10: Finetune the current policy with sub-trajectories in $\mathcal{T}_{\mathsf{sub}}$ using the sequence-level importance ratio (Equation (2)) to get a new policy $\pi_{\theta_{t+1}}$.
11: **end for**

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

From each reset point, we generate a group of sub-trajectories and evaluate them for an additional L_{eval} steps to obtain robust reward estimates (line 6). Finally, the policy is updated by maximizing the sequence-level objective using the computed group relative advantages (line 10).

D.2 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 Equation (1). The Q-function Q_{ϕ_t} is trained following the TD3 algorithm [15].

D.3 Adapting PPO to Decision Transformers

In addition to adapting GRPO to DTs, we further extend *Proximal Policy Optimization* (PPO, 42) to the Decision Transformer framework, denoted as PPO-DT. Unlike prior attempts that directly apply PPO to DTs [54], 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 [30], effectively reducing DTs to behavior cloning [39, 20]. Following Zheng et al. [62], 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

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 (Equation (3)).
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 Equation (4) to get an updated $\pi_{\theta_{t+1}}$; finetune the value network based on Equation (5) to get an updated $V_{\phi_{t+1}}$.
7: **end for**

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

estimation (GAE) [41] that combine Monte Carlo estimates with temporal-difference bootstrapping, yielding smoother and more stable value function updates.

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:

$$\widehat{A}_h = \sum_{\ell=0}^{T-h-1} (\gamma\lambda)^\ell \Big(r_{h+\ell} + \gamma V_{\phi_t}(s_{h+\ell+1}) - V_{\phi_t}(s_{h+\ell}) \Big), \tag{3}$$

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. [19], we further normalize the advantages across the batch. We then update the parameter by maximizing the following objective:

$$J_{\mathsf{PPO}}(\theta) = \frac{1}{|\mathcal{B}_{\mathsf{sub}}||\tau_k^{\mathsf{sub}}|} \sum_{\tau_k^{\mathsf{sub}} \in \mathcal{B}_{\mathsf{sub}}} \sum_{i=1}^{|\tau_k^{\mathsf{sub}}|}$$
$$\min\left(\frac{\pi_\theta(a_{k_i} \mid s_{k_i}, g_{k_i})}{\pi_{\theta_{\mathsf{old}}}(a_{k_i} \mid s_{k_i}, g_{k_i})} \widehat{A}_{k_i},\ \mathsf{clip}\left(\frac{\pi_\theta(a_{k_i} \mid s_{k_i}, g_{k_i})}{\pi_{\theta_{\mathsf{old}}}(a_{k_i} \mid s_{k_i}, g_{k_i})}, 1-\varepsilon, 1+\varepsilon \right) \widehat{A}_{k_i} \right) - \beta D_{\mathsf{KL}}(\pi_\theta \,\|\, \pi_{\theta_{\mathsf{ref}}}) \,. \tag{4}$$

Similar to GRPO-DT, we augment Equation (4) 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:

$$\mathcal{L}_{\mathsf{value}} = \mathbb{E}_{s_h}\left[(V_\phi(s_h) - V_{\mathsf{target}}(s_h))^2 \right]. \tag{5}$$

E Experimental Details

E.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. [54]. For the IQL baseline, we largely follow the implementation of Yan et al. [54], but set the number of pretraining steps to match those used by our methods and other baselines to ensure a fair comparison.

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 [12]
Optimizer	LAMB [56]
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 [12]
Optimizer	AdamW [28]

For SAC-DT, we follow the SAC-based online finetuning implementation of Yan et al. [54], adjusting its online training schedule and related hyperparameters to match our experimental setting.

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. [54]. Following Zheng et al. [62] and Yan et al. [54], 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 (Figure 4b), Pen-cloned-v1 (Figure 5a) and Hopper-medium-v2 (Figure 5b) and largely follows the practice of Yan et al. [54] for training the Q-functions and our GRPO-DT for training the policy network. To improve training stability, 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 [4] to the value network in Adroit and AntMaze to improve training stability. Environment-specific hyperparameters for PPO-DT are reported in Table 7.

E.2 RTG-conditioned Long-horizon Control

A central property of Decision Transformers is that the policy is conditioned on a desired return-to-go (RTG). Therefore, after online finetuning, it is important to verify that the model does not simply collapse into an RTG-insensitive policy, but still responds meaningfully to different target RTGs. To this end, we evaluate both pretrained and finetuned checkpoints under multiple target RTGs. We also decompose the achieved rewards into different trajectory stages to examine whether the performance improvement is concentrated only at the beginning of the trajectory or persists over longer horizons.

Table 8 reports the returns obtained under low, medium, and high target RTGs. After finetuning, higher target RTGs generally induce stronger performance, indicating that the finetuned policy retains RTG-conditioned behavior. For example, in Door-cloned-v1, the finetuned model improves

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. $\text{lr}_{\text{policy}}$ denotes the learning rate for the policy network, and κ_{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	$\text{lr}_{\text{policy}}$	κ_{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

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. $\text{lr}_{\text{policy}}$ and lr_{value} denote the learning rates for the policy and value networks, respectively. κ_{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}	$\text{lr}_{\text{policy}}$	lr_{value}	κ_{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

from 2508.8 under the low RTG to 3012.2 under the high RTG. Similar trends are observed in Pen-human-v1 and AntMaze-umaze, where high RTGs lead to better final outcomes after finetuning.

Table 9 further decomposes rewards across trajectory stages. The results show that the gains after finetuning are not limited to the early stage. In Door-cloned-v1 and Pen-human-v1, improvements are observed in both the middle and late stages, suggesting that the policy does not merely exploit short-term rewards but maintains long-horizon control. In AntMaze-umaze, the improvement appears mainly in the final stage, which is consistent with the sparse goal-reaching reward structure of AntMaze.

Overall, these results provide additional evidence that GRPO-DT improves online performance while preserving the long-horizon RTG-conditioned control ability of Decision Transformers.

Table 8: Rewards under different target RTGs. Higher RTGs generally induce better outcomes after finetuning, suggesting that the finetuned policy preserves RTG-conditioned control.

Env	Metric	Model	Low RTG	Mid RTG	High RTG
Door-cloned-v1	Return	Pretrained	45.6	302.2	401.5
		Finetuned	2508.8	2881.4	3012.2
Pen-human-v1	Return	Pretrained	1898.6	2104.8	1867.6
		Finetuned	2310.3	2767.4	2919.7
AntMaze-umaze	Return	Pretrained	0.67	0.50	0.57
		Finetuned	0.17	0.80	0.97
	Length	Pretrained	391.3	458.0	423.3
		Finetuned	623.2	306.2	231.4

Table 9: Reward decomposition across trajectory stages. Improvements after finetuning are not concentrated only in early stages, but also appear in middle and late stages.

Env	Return	Model	Stage 1	Stage 2	Stage 3
Door-cloned-v1	3000	Pretrained	-17.3	-12.4	431.1
		Finetuned	334.1	1338.9	1339.1
Pen-human-v1	6000	Pretrained	83.6	874.1	910.0
		Finetuned	571.2	1105.9	1242.6
AntMaze-umaze	2	Pretrained	0.0	0.0	0.57
		Finetuned	0.0	0.0	0.97

E.3 PPO Failure Analysis.

To investigate why PPO-DT performs poorly on Door-cloned, we repeated training with additional random seeds. Most runs remained near random performance, while a small subset succeeded. We compare a representative failed run and a successful run, tracking return, critic explained variance (EV), the fraction of importance ratios outside the PPO clipping interval $[0.8, 1.2]$, and ratio width (95th minus 5th percentile). Higher EV indicates better critic fit, while the latter two metrics characterize policy-update stability.

Table 10: PPO-DT diagnostics on Door-cloned. Each entry reports early $\rightarrow$ late training values.

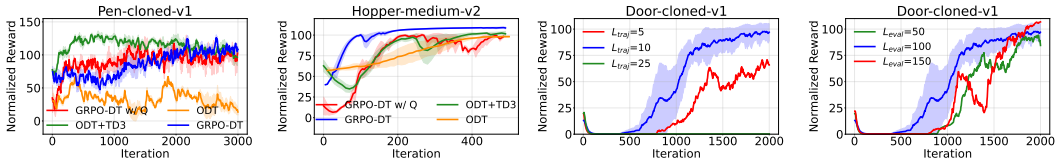
Run	Return	Critic EV	Clipped fraction	Ratio width
Failed	$-0.272 \rightarrow -0.280$	$0.228 \rightarrow 0.290$	$43.0\% \rightarrow 93.8\%$	$0.922 \rightarrow 4.85 \times 10^8$
Successful	$-0.265 \rightarrow 10.696$	$0.346 \rightarrow 0.892$	$48.7\% \rightarrow 36.4\%$	$1.020 \rightarrow 0.808$

The results in Table 10 suggest a feedback loop behind PPO-DT's failures. For most seeds, the initially weak policy collects nearly random trajectories, which provide little useful information for critic training. The inaccurate critic then produces noisy token-level GAE estimates, leading to poor policy updates and another batch of low-quality trajectories. As the Door environment is challenging, it's difficult for the model to explore high-reward actions. Consistent with this explanation, the failed run retains low critic EV, while its clipped-sample fraction rises to 93.8% and its ratio width reaches the numerical cap. In the few successful seeds, early trajectories contain sufficient useful behavior for the critic to improve, after which more accurate advantages lead to progressively better trajectories.

GRPO-DT is not critic-dependent and provides a more reliable learning signal when the pretrained policy and initial trajectories are poor. The difficulty is further amplified for PPO-DT in sparse-reward AntMaze, where informative critic targets are even harder to obtain.

E.4 Additional Ablation Results

Q-guided GRPO-DT. To further evaluate the applicability of Q-guided GRPO-DT without environment resetting, we extend the experiments in Section 4.3 to two additional environments. As shown in Figure 5(a,b), the Q-guided variant achieves meaningful policy improvement in these settings,



(a) Q-guided GRPO-DT on Pen-cloned-v1 (b) Q-guided GRPO-DT on Hopper-medium-v2 (c) Ablation on L_{traj} (d) Ablation on L_{eval}

Figure 5: Additional ablation study results. (a)(b) Additional evaluations of Q-guided GRPO-DT on two environments, showing that the reset-free variant remains competitive beyond the case presented in the main text. (c) Additional sensitivity analysis on sub-trajectory length L_{traj} in Door-cloned-v1. (d) Additional sensitivity analysis on evaluation horizon L_{eval} in Door-cloned-v1.

providing further evidence of its effectiveness. Its performance may depend on the quality of the learned Q-function, particularly in more challenging environments.

Hyperparameter sensitivity. We further extend the sensitivity analyses of L_{traj} and L_{eval} to the more challenging Door-cloned-v1 environment. Figure 5(c,d) illustrates the effects of these two parameters on training performance. These additional experiments complement the original results in Section 4.3 and provide further evidence of the importance of appropriate sub-trajectory lengths and sufficiently informative evaluation horizons.

F Broader Societal Impacts

This work studies online finetuning of Decision Transformers with pure reinforcement learning gradients, which may improve the efficiency and stability of sequential decision-making agents. Potential applications include robotics, autonomous control, and resource allocation, where better online adaptation can reduce engineering and computational cost. However, more capable adaptive agents may also introduce risks when rewards are misspecified or when online exploration is used in safety-critical settings. Our experiments are limited to simulated benchmarks, and real-world deployment would require careful reward design, safety constraints, and human oversight.

G Computational Resources

All experiments were run on a single NVIDIA RTX A6000 GPU, with four runs executed concurrently on the same GPU, which is how we ran all our experiments. Under this setup and our longer training schedule, each ODT+TD3 run requires approximately 24 hours on MuJoCo and 40 hours on Adroit, compared with 36 and 42 hours respectively for GRPO-DT. Hopper and AntMaze are relatively lightweight, requiring around 12 hours for our methods and 8 hours for ODT+TD3. Training accounts for 75-85% of ODT+TD3's runtime but only around 5% of GRPO-DT's, whose runtime is dominated by environment sampling and sub-trajectory evaluation. Since environment interaction dominates GRPO-DT's runtime, faster or more parallel simulators (e.g., GPU-accelerated physics engines) could substantially reduce its wall-clock time.

H Dataset and Algorithm Licenses

Our code is developed upon multiple algorithm repositories and environment testbeds.

Algorithm repositories. We implement our method based on the Online Decision Transformer [62] repository, which is licensed under CC BY-NC 4.0. We also refer to ODT+TD3 [54] when running baselines. To the best of our knowledge, the ODT+TD3 repository does not specify an explicit license; therefore, we only use it for baseline reproduction and do not redistribute its code.

Environment testbeds and datasets. We use OpenAI Gym [7], MuJoCo [48], and D4RL [14] as environment testbeds, which have an MIT license, an Apache-2.0 license, and an Apache-2.0 license respectively.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims are summarized in abstract and introduction, with the key contributions listed in Section 1. The corresponding technical details are presented in Section 3, and the main empirical results are reported in Section 4.2, including Table 1 and Figure 3.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Appendix B.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: See Section 3, Appendix C, Appendix D and Appendix E.1.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The code is released at: `https://github.com/otis704/DT-w-GRPO`.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Appendix E.1

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: See Section 4.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: See Appendix G.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research in the paper fully conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: See Appendix F.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: See Appendix H.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [Yes]

    Justification: Our code is publicly available in the repository linked in Item 5, with documentation covering environment setup and instructions for training and evaluation.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.