Reinforcement Learning Gradients as Vitamin for Online Finetuning Decision Transformers

Kai Yan **Alexander G. Schwing** **Yu-Xiong Wang**
University of Illinois Urbana-Champaign
`{kaiyan3, aschwing, yxw}@illinois.edu`
`https://github.com/KaiYan289/RL_as_Vitamin_for_Online_Decision_Transformers`

Abstract

Decision Transformers have recently emerged as a new and compelling paradigm for offline Reinforcement Learning (RL), completing a trajectory in an autoregressive way. While improvements have been made to overcome initial shortcomings, online finetuning of decision transformers has been surprisingly under-explored. The widely adopted state-of-the-art Online Decision Transformer (ODT) still struggles when pretrained with low-reward offline data. In this paper, we theoretically analyze the online-finetuning of the decision transformer, showing that the commonly used Return-To-Go (RTG) that's far from the expected return hampers the online fine-tuning process. This problem, however, is well-addressed by the value function and advantage of standard RL algorithms. As suggested by our analysis, in our experiments, we hence find that simply adding TD3 gradients to the finetuning process of ODT effectively improves the online finetuning performance of ODT, especially if ODT is pretrained with low-reward offline data. These findings provide new directions to further improve decision transformers.

1 Introduction

While Reinforcement Learning (RL) has achieved great success in recent years [55, 31], it is known to struggle with several shortcomings, including training instability when propagating a Temporal Difference (TD) error along long trajectories [14], low data efficiency when training from scratch [67], and limited benefits from more modern neural network architectures [12]. The latter point differs significantly from other parts of the machine learning community such as Computer Vision [17] and Natural Language Processing [11].

To address these issues, Decision Transformers (DTs) [14] have been proposed as an emerging paradigm for RL, introducing more modern transformer architectures into the literature rather than the still widely used Multi-Layer Perceptrons (MLPs). Instead of evaluating state and state-action pairs, a DT considers the whole trajectory as a sequence to complete, and trains on offline data in a supervised, auto-regressive way. Upon inception, DTs have been improved in various ways, mostly dealing with architecture changes [37], the token to predict other than return-to-go [22], addressing the problem of being overly optimistic [46], and the inability to stitch together trajectories [5]. Significant and encouraging improvements have been reported on those aspects.

However, one fundamental issue has been largely overlooked by the community: *offline-to-online RL using decision transformers*, i.e., finetuning of decision transformers with online interactions. Offline-to-online RL [72, 41] is a widely studied sub-field of RL, which combines offline RL learning from given, fixed trajectory data and online RL data from interactions with the environment. By first training on offline data and then finetuning, the agent can learn a policy with much greater data efficiency, while calibrating the out-of-distribution error from the offline dataset. Unsurprisingly, this sub-field has become popular in recent years.

While there are numerous works in the offline-to-online RL sub-field [35, 28, 62], surprisingly few works have discussed the offline-to-online finetuning ability of decision transformers. While there is work that discusses finetuning of decision transformers predicting encoded future trajectory information [64], and work that finetunes pretrained decision transformers with PPO in multi-agent RL [38], the current widely adopted state-of-the-art is the Online Decision Transformer (ODT) [74]: the decision transformer training is continued on online data following the same supervised-learning paradigm as in offline RL. However, this method struggles with low-reward data, as well as with reaching expert-level performance due to suboptimal trajectories [41] (also see Sec. 4).

To address this issue and enhance online finetuning of decision transformers, we theoretically analyze the decision transformer based on recent results [7], showing that the commonly used conditioning on a high Return-To-Go (RTG) that's far from the expected return hampers results. To fix, we explore the possibility of using *tried-and-true RL gradients*. Testing on multiple environments, we find that simply combining TD3 [21] gradients with the original auto-regressive ODT training paradigm is surprisingly effective: it improves results of ODT, especially if ODT is pretrained with low-reward offline data.

Our contributions are summarized as follows:

1) We propose a simple yet effective method to boost the performance of online finetuning of decision transformers, especially if offline data is of medium-to-low quality;

2) We theoretically analyze the online decision transformer, explain its "policy update" mechanism when using the commonly applied high target RTG, and point out its struggle to work well with online finetuning;

3) We conduct experiments on multiple environments, and find that ODT aided by TD3 gradients (and sometimes even the TD3 gradient alone) are surprisingly effective for online finetuning of decision transformers.

2 Preliminaries

Markov Decision Process. A Markov Decision Process (MDP) is the basic framework of sequential decision-making. An MDP is characterized by five components: the state space S, the action space A, the transition function p, the reward r, and either the discount factor γ or horizon H. MDPs involve an *agent* making decisions in discrete steps $t \in \{0, 1, 2, \dots\}$. On step t, the agent receives the current state $s_t \in S$, and samples an action $a_t \in A$ according to its *stochastic* policy $\pi(a_t|s_t) \in \Delta(A)$, where $\Delta(A)$ is the probability simplex over A, or its *deterministic* policy $\mu(s_t) \in A$. Executing the action yields a reward $r(s_t, a_t) \in \mathbb{R}$, and leads to the evolution of the MDP to a new state s_{t+1}, governed by the MDP's transition function $p(s_{t+1}|s_t, a_t)$. The goal of the agent is to maximize the total reward $\sum_t \gamma^t r(s_t, a_t)$, discounted by the discount factor $\gamma \in [0, 1]$ for infinite steps, or $\sum_{t=1}^{H} r(s_t, a_t)$ for finite steps. When the agent ends a complete run, it finishes an *episode*, and the state(-action) data collected during the run is referred to as a *trajectory* τ.

Offline and Online RL. Based on the source of learning data, RL can be roughly categorized into offline and online RL. The former learns from a given finite dataset of state-action-reward trajectories, while the latter learns from trajectories collected online from the environment. The effort of combining the two is called *offline-to-online* RL, which first pre-trains a policy using offline data, and then continues to finetune the policy using online data with higher efficiency. Our work falls into the category of offline-to-online RL. We focus on improving the decision transformers, instead of Q-learning-based methods which are commonly used in offline-to-online RL.

Decision Transformer (DT). The decision transformer represents a new paradigm of offline RL, going beyond a TD-error framework. It views a trajectory τ as a sequence to be auto-regressively completed. The sequence interleaves three types of tokens: **returns-to-go (RTG, the target total return)**, states, and actions. At step t, the past sequence of context length K is given as the input, i.e., the input is $(\text{RTG}_{t-K}, s_{t-k}, a_{t-k}, \dots, \text{RTG}_t, s_t)$, and an action is predicted by the auto-regressive model, which is usually implemented with a GPT-like architecture [11]. The model is trained via supervised learning, considering the past K steps of the trajectory along with the current state and the current return-to-go as the feature, and the sequence of all actions a in a segment as the labels. At evaluation time, a **desired return RTG_{eval}** is specified, since the **ground truth future return RTG_{real}** isn't known in advance.

Online Decision Transformer (ODT). ODT has two stages: offline pre-training which is identical to classic DT training, and online finetuning where trajectories are iteratively collected and the policy is updated via supervised learning. Specifically, the action a_t at step t during rollouts is computed by the deterministic policy $\mu^{\text{DT}}(s_{t-T:t}, a_{t-T:t-1}, \text{RTG}_{t-T:t}, T = T_{\text{eval}}, \text{RTG} = \text{RTG}_{\text{eval}})$,[1] or sampled from the stochastic policy $\pi^{\text{DT}}(a_t|s_{t-T:t}, a_{t-T:t-1}, \text{RTG}_{t-T:t}, T = T_{\text{eval}}, \text{RTG} = \text{RTG}_{\text{eval}})$. Here, T is the context length (which is T_{eval} in evaluation), and $\text{RTG}_{\text{eval}} \in \mathbb{R}$ is the target return-to-go. The data buffer, initialized with offline data, is gradually replaced by online data during finetuning.

When updating the policy, the following loss (we use the deterministic policy as an example, and thus omit the entropy regularizer) is minimized:

$$\sum_{t=1}^{T_{\text{train}}} \left\| \mu^{\text{DT}} \left(s_{0:t}, a_{0:t-1}, \text{RTG}_{0:t}, \text{RTG} = \text{RTG}_{\text{real}}, T = t \right) - a_t \right\|_2^2 . \tag{1}$$

Note, T_{train} is the training context length and RTG_{real} is the real return-to-go. For better readability, we denote $\{s_{x+1}, s_{x+2}, \ldots, s_y\}$, $x, y \in \mathbb{N}$ as $s_{x:y}$ (i.e., left *exclusive* and right *inclusive*), and similarly $\{a_{x+1}, a_{x+2}, \ldots, a_y\}$ as $a_{x:y}$ and $\{\text{RTG}_{x+1}, \ldots, \text{RTG}_y\}$ as $\text{RTG}_{x:y}$. Specially, index $x = y$ represents an empty sequence. For example, when $t = 1$, $a_{0:0}$ is an empty action sequence as the decision transformer is not conditioned on any past action.

One important observation: the decision transformer is inherently *off-policy* (the exact policy distribution varies with the sampled starting point, context length and return-to-go), which effectively guides our choice of RL gradients to off-policy algorithms (see Appendix C for more details).

TD3. Twin Delayed Deep Deterministic Policy Gradient (TD3) [21] is a state-of-the-art online off-policy RL algorithm that learns a *deterministic* policy $a = \mu^{\text{RL}}(s)$. It is an improved version of an actor-critic (DDPG [32]) with three adjustments to improve its stability: 1) *Clipped double Q-learning*, which maintains two critics (estimators for expected return) $Q_{\phi_1}, Q_{\phi_2} : |S| \times |A| \to \mathbb{R}$ and uses the smaller of the two values (i.e., $\min(Q_{\phi_1}, Q_{\phi_2})$) to form the target for TD-error minimization. Such design prevents overestimation of the Q-value; 2) *Policy smoothing*, which adds noise when calculating the Q-value for the next action to effectively prevent overfitting; and 3) *Delayed update*, which updates μ^{RL} less frequently than Q_{ϕ_1}, Q_{ϕ_2} to benefit from a better Q-value landscape when updating the actor. TD3 also maintains a set of *target networks* storing old parameters of the actor and critics that are soft-updated with slow exponential moving average updates from the current, active network. In this paper, we adapt this algorithm to fit the decision transformer architecture so that it can be used as an auxiliary objective in an online finetuning process.

3 Method

This section is organized as follows: we will first provide intuition why RL gradients aid online finetuning of decision transformers (Sec. 3.1), and present our method of adding TD3 gradients (Sec. 3.2). To further justify our intuition, we provide a theoretical analysis on how ODT fails to improve during online finetuning when pre-trained with low-reward data (Sec. 3.3).

3.1 Why RL Gradients?

In order to understand why RL gradients aid online finetuning of decision transformers, let us consider an MDP which only has a single state s_0, one step, a one dimensional action $a \in [-1, 1]$ (i.e., a bandit with continuous action space) and a simple reward function $r(a) = (a + 1)^2$ if $a \leq 0$ and $r(a) = 1 - 2a$ otherwise, as illustrated in Fig. 1. In this case, a trajectory can be represented effectively by a scalar, which is the action. If the offline dataset for pretraining is of low quality, i.e., all actions in the dataset are either close to -1 or 1, then the decision transformer will obviously not generate trajectories with high RTG after offline training. As a consequence, during online finetuning, the new rollout trajectory is very likely to be uninformative about how to reach RTG_{eval}, since it is too far from RTG_{eval}. Worse still, it cannot improve locally either, which requires $\frac{\partial \text{RTG}}{\partial a}$. However, the decision transformer *yields exactly the inverse*, i.e., $\frac{\partial a}{\partial \text{RTG}}$. Since the transformer is not invertible (and even if the transformer is invertible, often the ground truth $\text{RTG}(a)$ itself is not), we cannot

[1] ODT uses different RTGs for evaluation and online rollouts, but we refer to both as RTG_{eval} as they are both expert-level expected returns.

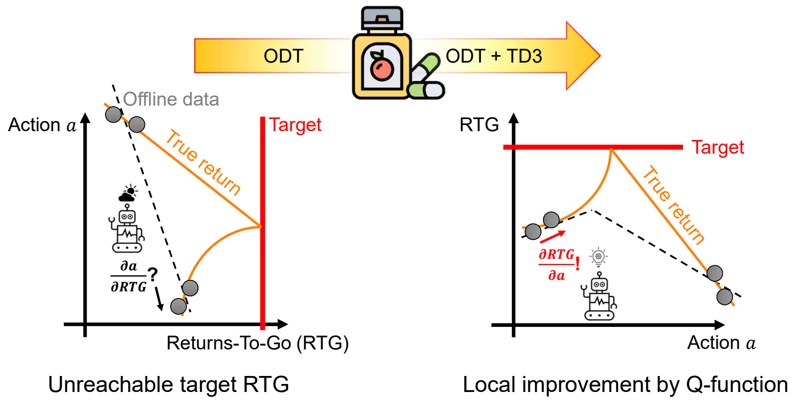


Figure 1: An overview of our work, illustrating why ODT fails to improve with low-return offline data and RL gradients such as TD3 could help. The decision transformer yields gradient $\frac{\partial a}{\partial \text{RTG}}$, but local policy improvement requires the opposite, i.e., $\frac{\partial \text{RTG}}{\partial a}$. Therefore, the agent cannot recover if the current policy conditioning on high target RTG does not actually lead to high real RTG, which is very likely when the target RTG is too far from the pretrained policy and out-of-distribution. By adding a small coefficient for RL gradients, the agents can improve locally, which leads to better performance.

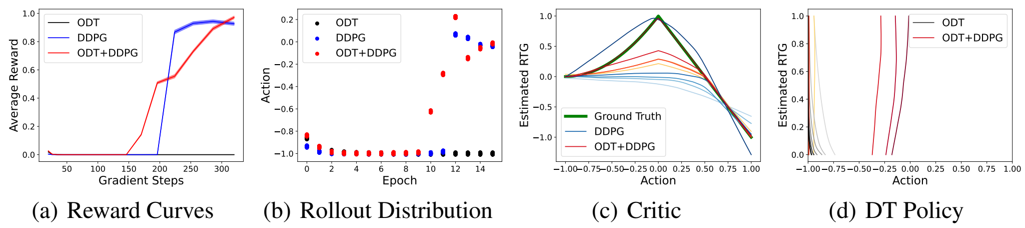


Figure 2: An illustration of a simple MDP, showing how RL can infer the direction for improvement, while online DT fails. Panels (a) and (b) show, DDPG and ODT+DDPG manage to maximize reward and find the correct optimal action quickly, while ODT fails to do so. Panel (c) shows how a DDPG/ODT+DDPG critic (from light blue/orange to dark blue/red) manages to fit ground truth reward (green curve). Panel (d) shows that the ODT policy (changing from light gray to dark) fails to discover the hidden reward peak near 0 between two low-reward areas (near -1 and 1 respectively) contained in the offline data. Meanwhile, ODT+DDPG succeeds in finding the reward peak.

easily estimate the former from the latter. Thus, the hope for policy improvement relies heavily on the generalization of RTG, i.e., policy yielded by high RTG_{eval} indeed leads to better policy without any data as evidence, which is not the case with our constructed MDP and dataset.

In contrast, applying traditional RL for continuous action spaces to this setting, we either learn a value function $Q(s_0, a) : \mathbb{R} \to \mathbb{R}$, which effectively gives us a direction of action improvement $\frac{\partial Q(s_0,a)}{\partial a}$ (e.g., SAC [25], DDPG [32], TD3 [21]), or an advantage $A(s_0, a)$ that highlights whether focusing on action a improves or worsens the policy (e.g., AWR [47], AWAC [40], IQL [28]). Either way provides a direction which suggests how to change the action locally in order to improve the (estimated) return. In our experiment illustrated in Fig. 2 (see Appendix F for details), we found that RL algorithms like DDPG [32] can easily solve the aforementioned MDP while ODT fails.

Thus, adding RL gradients aids the decision transformer to improve from given low RTG trajectories. While one may argue that the self-supervised training paradigm of ODT [74] can do the same by "prompting" the decision transformer to generate a high RTG trajectory, such paradigm is still unable to effectively improve the policy of the decision transformer pretrained on data with low RTGs. We provide a theoretical analysis for this in Sec. 3.3. In addition, we also explore the possibility of fixing this problem using other existing algorithms, such as JSRL [59] and slowly growing RTG (i.e., curriculum learning). However, we found that those algorithms cannot address this problem well. See Appendix G.8 for ablations.

3.2 Adding TD3 Gradients to ODT

In this work, we mainly consider TD3 [21] as the RL gradient for online finetuning. There are two reasons for selecting TD3. First, TD3 is a more robust off-policy RL algorithm compared to other off-policy RL algorithms [54]. Second, the success of TD3+BC [20] indicates that TD3 is a good candidate when combined with supervised learning. A more detailed discussion and empirical comparison to other RL algorithms can be found in Appendix C.

Generally, we simply add a weighted standard TD3 actor loss to the decision transformer objective. To do this, we follow classic TD3 and additionally train two critic networks $Q_{\phi_1}, Q_{\phi_2} : S \times A \to \mathbb{R}$ parameterized by ϕ_1, ϕ_2 respectively. In the offline pretraining stage, we use the following objective for the actor:

$$\min_{\mu^{\text{DT}}} \mathbb{E}_{\tau\sim D}\left[\frac{1}{T_{\text{train}}}\sum_{t=1}^{T_{\text{train}}}\left[-\alpha Q_{\phi_1}(s_t, \mu^{\text{DT}}(s_{0:t}, a_{0:t-1}, \text{RTG}_{0:t}, \text{RTG}=\text{RTG}_{\text{real}}, T=t))+\right.\right.$$
$$\left.\left.\|\mu^{\text{DT}}(s_{0:t}, a_{0:t-1}, \text{RTG}_{0:t}, \text{RTG}=\text{RTG}_{\text{real}}, T=t)-a_t\|_2^2\right]\right]. \quad (2)$$

Here, $\alpha \in \{0, 0.1\}$ is a hyperparameter, and the loss sums over the trajectory segment. For critics Q_{ϕ_1}, Q_{ϕ_2}, we use the standard TD3 critic loss

$$\min_{\phi_1,\phi_2} \mathbb{E}_{\tau\sim D}\sum_{t=1}^{T_{\text{train}}}\left[(Q_{\phi_1}(s_t,a_t)-Q_{\text{min},t})^2+(Q_{\phi_2}(s_t,a_t)-Q_{\text{min},t})^2\right], \quad \text{with}$$
$$Q_{\text{min},t} = r_t + \gamma(1-d_t)\min_{i\in\{1,2\}} Q_{\phi_{i,\text{tar}}}\left(s_t, \text{clip}\left(\mu^{\text{RL}}_{\text{tar}}(z_t)+\text{clip}(\epsilon,-c,c), a_{\text{low}}, a_{\text{high}}\right)\right), \quad (3)$$

where $\tau = \{s_{0:T_{\text{train}}+1}, a_{0:T_{\text{train}}}, \text{RTG}_{0:T_{\text{train}}+1}, d_{0:T_{\text{train}}}, r_{0:T_{\text{train}}}, \text{RTG} = \text{RTG}_{\text{real}}\}$ is the trajectory segment sampled from buffer D that stores the offline dataset. Further, d_t indicates whether the trajectory ends on the t-th step (true is 1, false is 0), Q_{min} is the target to fit, $Q_{\phi_{i,\text{tar}}}$ is produced by the target network (stored old parameter), z_t is the context for "next state" at step t. $\mu^{\text{RL}}_{\text{tar}}$ is the target network for the actor (i.e., decision transformer). For an n-dimensional action, clip(a, x, y), $a \in \mathbb{R}^n, y \in \mathbb{R}^n, z \in \mathbb{R}^n$ means clip a_i to $[y_i, z_i]$ for $i \in \{1, 2, \ldots, n\}$. $a_{\text{low}} \in \mathbb{R}^n$ and $a_{\text{high}} \in \mathbb{R}^n$ are the lower and upper bound for every dimension respectively.

To demonstrate the impact on aiding the exploration of a decision transformer, in this work we choose the simplest form of a critic, which is reflective, i.e., only depends on the current state. This essentially makes the Q-value an *average* of different context lengths sampled from a near-uniform distribution (see Appendix D for the detailed reason and distribution for this). The choice is based on the fact that training a transformer-based value function estimator is quite hard [45] due to increased input complexity (i.e., noise from the environment) which leads to reduced stability and slower convergence. In fact, to avoid this difficulty, many recent works on Large Language Models (LLMs) [13] and vision models [48] which finetune with RL adopt a policy-based algorithm instead of an actor-critic, despite a generally lower variance of the latter. In our experiments, we also found such a critic to be much more stable than a recurrent critic network (see Appendix G for ablations).

During online finetuning, we again use Eq. (2) and Eq. (3), but always use $\alpha = 0.1$ for Eq. (2).

While the training paradigm resembles that of TD3+BC, our proposed method improves upon TD3+BC in the following two ways: 1) **Architecture**. While TD3+BC uses MLP networks for single steps, we leverage a decision transformer, which is more expressive and can take more context into account when making decisions. 2) **Selected instead of indiscriminated behavior cloning.** Behavior cloning mimics all data collected without regard to their reward, while the supervised learning process of a decision transformer prioritizes trajectories with higher return by conditioning action generation on higher RTG. See Appendix G.9 for an ablation.

3.3 Why Does ODT Fail to Improve the Policy?

As mentioned in Sec. 3.1, it is the goal of ODT to "prompt" a policy with a high RTG, i.e., to improve a policy by conditioning on a high RTG during online rollout. However, beyond the intuition provided in Sec. 3.1, in this section, we will analyze more formally why such a paradigm is unable to improve the policy given offline data filled with low-RTG trajectories.

Our analysis is based on the performance bound proved by Brandfonbrener et al. [7]. Given a dataset drawn from an underlying policy β and given its RTG distribution P_β (either continuous or discrete), under assumptions (see Appendix E), we have the following *tight* performance bound for a decision transformer with policy $\pi^{\text{DT}}(a|s, \text{RTG}_{\text{eval}})$ conditioned on RTG_{eval}:

$$\text{RTG}_{\text{eval}} - \mathbb{E}_{\tau=(s_1,a_1,\ldots,s_H,a_H)\sim\pi^{\text{DT}}(a|s,\text{RTG}_{\text{eval}})}[\text{RTG}_{\text{real}}] \leq \epsilon\left(\frac{1}{\alpha_f}+2\right)H^2. \tag{4}$$

Here, $\alpha_f = \inf_{s_1} P_\beta(\text{RTG}_{\text{real}} = \text{RTG}_{\text{eval}}|s_1)$ for every initial state s_1, $\epsilon > 0$ is a constant, H is the horizon of the MDP.[2] Based on this tight performance bound, we will show that **with high probability, $\frac{1}{\alpha_f}$ grows *superlinearly* with respect to RTG_{eval}**. If true, then the RTG_{real} term (i.e., the actual return from online rollouts) must decrease to fit into the tight bound, as RTG_{eval} grows.

To show this, we take a two-step approach: First, we prove that the probability mass of the RTG distribution is concentrated around low RTGs, i.e., *event probability* $\Pr_\beta\left(\text{RTG} - \mathbb{E}_\beta(\text{RTG}|s) \geq c|s\right)$ for $c > 0$ decreases superlinearly with respect to c. For this, we apply the Chebyshev inequality, which yields a bound of $O\left(\frac{1}{c^2}\right)$. However, without knowledge on $P_\beta(\text{RTG}|s)$, the variance can be made arbitrarily large by high RTG outliers, hence making the bound meaningless.

Fortunately, we have knowledge about the RTG distribution $P_\beta(\text{RTG}|s)$ from the collected data. If we refer to the maximum RTG in the dataset via $\text{RTG}_{\beta\text{max}}$ and if we assume all rewards are non-negative, then all trajectory samples have an RTG in $[0, \text{RTG}_{\beta\text{max}}]$. Thus, with adequate prior distribution, we can state that with high probability $1-\delta$, the probability mass is concentrated in the low RTG area. Based on this, we can prove the following lemma:

Lemma 1. *(Informal) Assume rewards $r(s, a)$ are bounded in $[0, R_{max}]$,[3] and $RTG_{eval} \geq RTG_{\beta max}$. Then with probability at least $1-\delta$, we have the probability of event $\Pr_\beta$ bounded as follows:*

$$\Pr_\beta\left(RTG_{eval} - V^\beta(s) \geq c|s\right) \leq O\left(\frac{R_{max}^2 T^2}{c^2}\right), \tag{5}$$

where δ depends on the number of trajectories in the dataset and prior distribution (see Appendix E for a concrete example and a more accurate bound). $V^\beta(s)$ is the value function of the underlying policy $\beta(a|s)$ that generates the dataset, for which we have $V^\beta(s) = \mathbb{E}_\beta(RTG|s)$.

The second step uses the bound of probability mass $\Pr_\beta(\text{RTG} \geq c|s)$ to derive the bound for α_f. For the discrete case where the possibly obtained RTGs are finite or countably infinite (note, state and action space can still be continuous), this is simple, as we have

$$P_\beta\left(\text{RTG} = V^\beta(s) + c|s\right) = \Pr_\beta\left(\text{RTG} = V^\beta(s) + c|s\right) \leq \Pr_\beta\left(\text{RTG} \geq V^\beta(s) + c|s\right). \tag{6}$$

Thus $\alpha_f = \inf_{s_1} P_\beta(\text{RTG}|s_1)$ can be conveniently bounded by Lemma 1. For the continuous case, the proof is more involved as probability density $P_\beta(\text{RTG}|s)$ can be very high on an extremely short interval of RTG, making the total probability mass arbitrarily small. However, assuming that $P_\beta(\text{RTG}|s)$ is Lipschitz when $\text{RTG} \geq \text{RTG}_{\beta\text{max}}$ (i.e., RTG area not covered by dataset), combined with the discrete distribution case, we can still get the following (see Appendix E for proof):

Corollary 1. *(Informal) If the RTG distribution is discrete (i.e., number of possible different RTGs are at most countably infinite), then with probability at least- $1-\delta$, $\frac{1}{\alpha_f}$ grows on the order of $\Omega(RTG_{eval}^2)$ with respect to RTG_{eval}. For continuous RTG distributions satisfying a Lipschitz continuous RTG density p_β, $\frac{1}{\alpha_f}$ grows on the order of $\Omega(RTG_{eval}^{1.5})$.*

Here, $\Omega(\cdot)$ refers to the big-Omega notation (asymptotic lower bound).

4 Experiments

In this section, we aim to address the following questions: **a)** Does our proposed solution for decision transformers indeed improve its ability to cope with low-reward pretraining data. **b)** Is improving

[2]Eq. (4) is very informal. See Appendix E for a more rigorous description.

[3]Note we use "max" instead of "βmax" as this is a property of the environment and not the dataset.

what to predict, while still using supervised learning, the correct way to improve the finetuning ability of decision transformers? **c)** Does the transformer architecture, combined with RL gradients, work better than TD3+BC? **d)** Is it better to combine the use of RL and supervised learning, or better to simply abandon the supervised loss in online finetuning? **e)** How does online decision transformer with TD3 gradient perform compared to other offline RL algorithms? **f)** How much does TD3 improve over DDPG which was used in Fig. 2?

Baselines. In this section, we mainly compare to six baselines: the widely recognized state-of-the-art DT for online finetuning, Online Decision Transformer (**ODT**) [74]; **PDT**, a baseline improving over ODT by predicting future trajectory information instead of return-to-go; **TD3+BC** [20], a MLP offline RL baseline; **TD3**, an ablated version of our proposed solution where we use TD3 gradients only for decision transformer finetuning (but only use supervised learning of the actor for offline pretraining); **IQL** [28], one of the most popular offline RL algorithms that can be used for online finetuning; **DDPG [32]+ODT**, which is the same as our approach but with DDPG instead of TD3 gradients (for ablations using SAC [25], IQL [28], PPO [52], AWAC [40] and AWR [47], see Appendix C). Each of the baselines corresponds to one of the questions a), b), c), d), e) and f) above.

Metrics. We use the normalized average reward (same as D4RL's standard [19]) as the metric, where higher reward indicates better performance. If the final performance is similar, the algorithm with fewer online examples collected to reach that level of performance is better. We report the reward curve, which shows the change of the normalized reward's mean and standard deviation with 5 different seeds, with respect to the number of online examples collected. The maximum number of steps collected is capped at 500K (for mujoco) or 1M (for other environments). We also report evaluation results using the rliable [3] library in Fig. 7 of Appendix B.

Experimental Setup. We use the same architecture and hyperparameters such as learning rate (see Appendix F.2 for details) as ODT [74]. The architecture is a transformer with 4 layers and 4 heads in each layer. This translates to around 13M parameters in total. For the critic, we use Multi-Layer Perceptrons (MLPs) with width 256 and two hidden layers and ReLU [1] activation function. Specially, for the random dataset, we collect trajectories until the total number of steps exceeds 1000 in every epoch, which differs from ODT, where only 1 trajectory per epoch is collected. This is because many random environments, such as hopper, have very short episodes when the agent does not perform well, which could lead to overfitting if only a single trajectory is collected per epoch. For fairness, we use this modified rollout for ODT in our experiments as well. Not doing so does not affect ODT results since it does generally not work well on random datasets, but will significantly increase the time to reach a certain number of online transitions. After rollout, we train the actor for 300 gradient steps and the critic for 600 steps following TD3's delayed update trick.

4.1 Adroit Environments

Environment and Dataset Setup. We test on four difficult robotic manipulation tasks [49], which are the Pen, Hammer, Door and Relocate environment. For each environment, we test three different datasets: expert, cloned and human, which are generated by a finetuned RL policy, an imitation learning policy and human demonstration respectively. See Appendix F.1 for details.

Results. Fig. 3 shows the performance of each method on Adroit before and after online finetuning. TD3+BC fails on almost all tasks and often diverges with extremely large Q-value during online finetuning. ODT and PDT perform better but still fall short of the proposed method, TD3+ODT. Note, IQL, TD3 and TD3+ODT all perform decently well (with similar average reward as shown in Tab. 2 in Appendix B). However, we found that TD3 often fails during online finetuning, probably because the environments are complicated and TD3 struggles to recover from a poor policy generated during online exploration (i.e., it has a *catastrophic forgetting* issue). To see whether there is a simple fix, in Appendix G.7, we ablate whether an action regularizer pushing towards a pretrain policy similar to TD3+BC helps, but find it to hinder performance increase in other environments. IQL is overall much more stable than TD3, but improves much less during online finetuning than TD3+ODT. ODT can achieve good performance when pretrained on expert data, but struggles with datasets of lower quality, which validates our motivation. DDPG+ODT starts out well in the online finetuning stage but fails quickly, probably because DDPG is less stable compared to TD3.

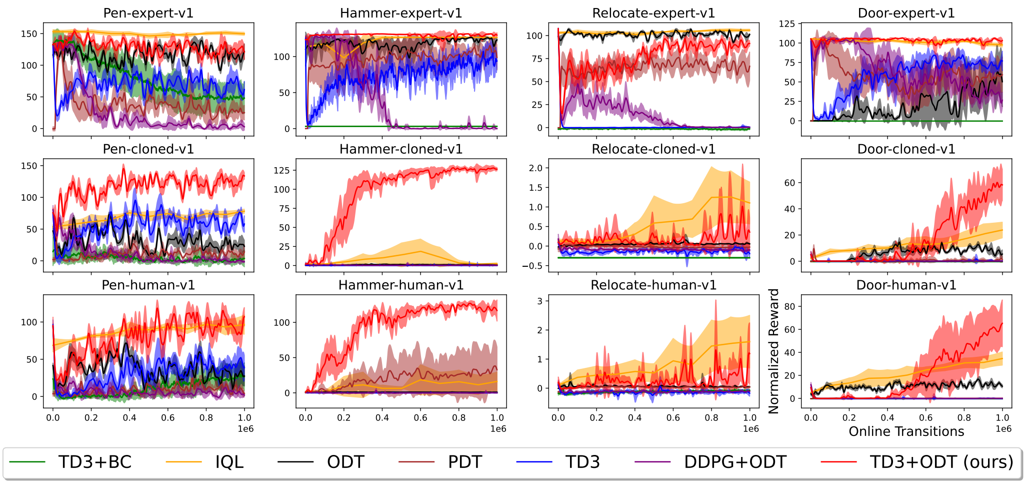


Figure 3: Results on Adroit [49] environments. The proposed method, TD3+ODT, improves upon baselines. Note that TD3, IQL, and TD3+ODT all perform decently at the beginning of online finetuning, but TD3 fails while TD3+ODT improves much more than IQL during online finetuning.

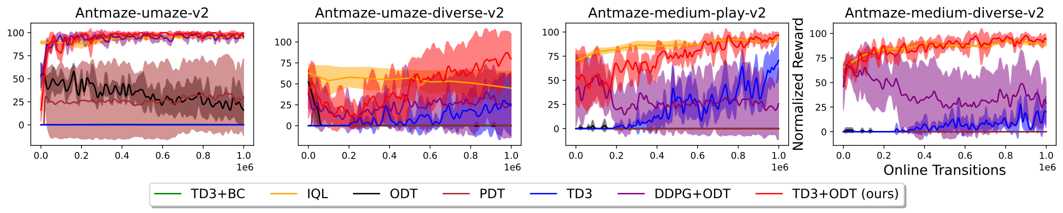


Figure 4: Reward curves for each method in Antmaze environments. IQL works best on the large maze, while our proposed method works the best on the medium maze and umaze. DDPG+ODT works worse than our method and IQL but much better than the rest of the baselines, which again validates our motivation that adding RL gradients to ODT is helpful.

4.2 Antmaze Environments

Environment and Dataset Setup. We further test on a harder version of the Maze2D environment in D4RL [19] where the pointmass is substituted by a robotic ant. We study six different variants, which are umaze, umaze-diverse, medium-play, medium-diverse, large-play and large-diverse.

Results. Fig. 4 lists the results of each method on umaze and medium maze before and after online finetuning (see Appendix C for reward curves and Appendix B for results summary on large antmaze). TD3+ODT works the best on umaze and medium maze, and significantly outperforms TD3. This shows that RL gradients alone are not enough for offline-to-online RL of the decision transformer. Though TD3+ODT does not work on large maze, we found that IQL+ODT works decently well. However, we choose TD3+ODT in this work because IQL+ODT does not work well on the random datasets. This is probably because IQL aims to address the Out-Of-Distribution (OOD) estimation problem [28], which makes it better at utilizing offline data but worse at online exploration. See Appendix C for a detailed discussion and results. DDPG+ODT works worse than TD3+ODT but much better than baselines except IQL.

4.3 MuJoCo Environments

Environment and Dataset Setup. We further test on four widely recognized standard environments [58], which are the Hopper, Halfcheetah, Walker2d and Ant environment. For each environment, we study three different datasets: medium, medium-replay, and random. The first and second one contain trajectories of decent quality, while the last one is generated with a random agent.

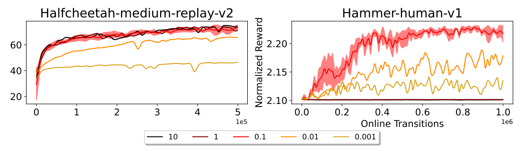
	TD3+BC	IQL	ODT	PDT	TD3	DDPG+ODT	TD3+ODT (ours)
Ho-M-v2	60.24(+4.4)	44.72(-21.3)	**97.84(+48.69)**	74.43(+72.21)	88.98(+29.25)	41.7(-13.18)	89.07(+25.97)
Ho-MR-v2	**99.07(+33.33)**	62.76(-7.63)	83.29(+63.17)	84.53(+82.23)	93.72(+55.66)	32.36(+9.9)	95.65(+65.89)
Ho-R-v2	8.36(-0.35)	20.42(+12.36)	29.08(+26.92)	35.9(+34.67)	75.68(+73.69)	25.12(+23.14)	**76.13(+74.15)**
Ha-M-v2	51.29(+2.73)	37.12(-10.35)	42.27(+19.23)	39.35(+39.55)	70.9(+29.59)	55.69(+14.71)	**76.91(+35.3)**
Ha-MR-v2	56.5(+13.07)	49.97(+6.84)	41.45(+26.77)	31.47(+31.8)	69.87(+40.59)	53.71(+24.91)	**73.27(+43.98)**
Ha-R-v2	44.78(+31.12)	47.85(+40.3)	2.15(-0.09)	0.74(+0.9)	**68.55(+66.3)**	34.56(+32.31)	59.35(+57.1)
Wa-M-v2	85.34(+3.49)	65.55(-15.12)	75.57(+18.47)	63.37(+63.3)	90.49(+24.74)	2.01(-69.54)	**97.86(+27.08)**
Wa-MR-v2	83.28(+0.0)	95.99(+28.78)	77.2(+12.46)	54.49(+54.18)	**100.88(+32.54)**	1.04(-60.59)	100.6(+42.54)
Wa-R-v2	6.99(+5.86)	10.67(+4.96)	14.12(+9.82)	15.47(+15.32)	**69.91(+66.31)**	2.91(-2.47)	57.86(+53.27)
An-M-v2	129.11(+7.11)	110.36(+14.26)	88.1(-0.51)	52.08(+48.47)	125.67(+37.55)	10.81(-75.52)	**132.0(+41.42)**
An-MR-v2	129.33(+41.03)	113.16(+24.24)	85.64(+4.49)	36.92(+32.41)	**133.58(+51.17)**	4.05(-87.7)	130.23(+52.08)
An-R-v2	67.89(+33.47)	12.28(+0.97)	24.96(-6.44)	14.88(+10.38)	63.47(+32.02)	4.93(-26.55)	**71.69(+40.31)**
Average	68.52(+14.6)	55.9(+7.8)	55.14(+18.58)	41.97(+40.44)	87.64(+44.95)	22.87(-19.22)	**88.38(+46.59)**

Table 1: Average reward for each method in MuJoCo environments before and after online finetuning. The best performance for each environment is highlighted in bold font, and any result $> 90\%$ of the best performance is underlined. To save space, the name of the environments and datasets are abbreviated as follows: for the environments Ho=Hopper, Ha=HalfCheetah, Wa=Walker2d, An=Ant; for the datasets M=Medium, MR=Medium-Replay, R=Random. The format is "final(+increase after finetuning)". The proposed solution performs well.

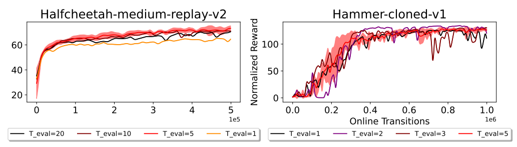
Results. Fig. 6 shows the results of each method on MuJoCo before and after online finetuning. We observe that autoregressive-based algorithms, such as ODT and PDT, fail to improve the policy on MuJoCo environments, especially from low-reward pretraining with random datasets. With RL gradients, TD3+BC and IQL can improve the policy during online finetuning, but less than a decision transformer (TD3 and TD3+ODT). In particular, we found IQL to struggle on most random datasets, which are well-solved by decision transformers with TD3 gradients. TD3+ODT still outperforms TD3 with an average final reward of 88.51 vs. 84.23. See Fig. 6 in Appendix B for reward curves.

Ablations on α. Fig. 5 (a) shows the result of using different α (i.e., RL coefficients) on different environments. We observe an increase of α to improve the online finetuning process. However, if α is too large, the algorithm may get unstable.

Ablations on evaluation context length T_{eval}. Fig. 5 (b) shows the result of using different T_{eval} on halfcheetah-medium-replay-v2 and hammer-cloned-v1. The result shows that T_{eval} needs to be balanced between more information for decision-making and potential training instability due to a longer context length. As shown in the halfcheetah-medium-replay-v2 result, T_{eval} too long or too short can both lead to performance drops. More ablations are available in Appendix G.



(a) Ablations on α



(b) Ablations on T_{eval}

Figure 5: Panel (a) shows ablations on RL coefficient α. While higher α aids exploration as shown in the halfcheetah-medium-replay-v2 case, it may sometimes introduce instability, which is shown in the hammer-human-v1 case. Panel (b) shows ablations on T_{eval}. T_{eval} balances training stability and more information for decision-making.

5 Related Work

Online Finetuning of Decision Transformers. While there are many works on generalizing decision transformers (e.g., predicting waypoints [5], goal, or encoded future information instead of return-to-go [22, 5, 57, 36]), improving the architecture [37, 16, 53, 65] or addressing the overly-optimistic [46] or trajectory stitching issue [63]), there is surprisingly little work beyond online decision transformers that deals with online finetuning of decision transformers. There is some loosely related literature: MADT [31] proposes to finetune pretrained decision transformers with PPO. PDT [64] also studies online finetuning with the same training paradigm as ODT [74]. QDT [66] uses an offline RL algorithm to re-label returns-to-go for offline datasets. AFDT [76] and STG [75] use decision transformers offline to generate an auxiliary reward and aid the training of online RL algorithms. A few works study in-context learning [33, 34] and meta-learning [60, 30] of decision transformers, where improvements with evaluations on new tasks are made possible. However, none of the papers above focuses on addressing the general online finetuning issue of the decision transformer.

Transformers as Backbone for RL. Having witnessed the impressive success of transformers in Computer Vision (CV) [17] and Natural Language Processing (NLP) [11], numerous works also studied the impact of transformers in RL either as a model for the agent [45, 38] or as a world model [39, 50]. However, a large portion of state-of-the-art work in RL is still based on simple Multi-Layer Perceptrons (MLPs) [35, 28]. This is largely because transformers are significantly harder to train and require extra effort [45], making their ability to better memorize long trajectories [42] harder to realize compared to MLPs. Further, there are works on using transformers as feature extractors for a trajectory [37, 45] and works that leverage the common sense of transformer-based Large Language Model's for RL priors [10, 9, 70]. In contrast, our work focuses on improving the new "RL via Supervised learning" (RvS) [7, 18] paradigm, aiming to merge this paradigm with the benefits of classic RL training.

Offline-to-Online RL. Offline-to-online RL bridges the gap between offline RL, which heavily depends on the quality of existing data while struggling with out-of-distribution policies, and online RL, which requires many interactions and is of low data efficiency. Mainstream offline-to-online RL methods include teacher-student [51, 6, 59, 72] and out-of-distribution handling (regularization [21, 29, 62], avoidance [28, 23], ensembles [2, 15, 24]). There are also works on pessimistic Q-value initialization [69], confidence bounds [26], and a mixture of offline and online training [56, 73]. However, all the aforementioned works are based on Q-learning and don't consider decision transformers.

6 Conclusion

In this paper, we point out an under-explored problem in the Decision Transformer (DT) community, i.e., online finetuning. To address online finetuning with a decision transformer, we examine the current state-of-the-art, online decision transformer, and point out an issue with low-reward, sub-optimal pretraining. To address the issue, we propose to mix TD3 gradients with decision transformer training. This combination permits to achieve better results in multiple testbeds. Our work is a complement to the current DT literature, and calls out a new aspect of improving decision transformers.

Limitations and Future Works. While our work theoretically analyzes an ODT issue, the conclusion relies on several assumptions which we expect to remove in future work. Empirically, in this work we propose a simple solution orthogonal to existing efforts like architecture improvements and predicting future information rather than return-to-go. To explore other ideas that could further improve online finetuning of decision transformers, next steps include the study of other environments and other ways to incorporate RL gradients into decision transformers. Other possible avenues for future research include testing our solution on image-based environments, and decreasing the additional computational cost compared to ODT (an analysis for the current time cost is provided in Appendix H).

Acknowledgments

This work was supported in part by NSF under Grants 2008387, 2045586, 2106825, MRI 1725729, NIFA Award 2020-67021-32799, the IBM-Illinois Discovery Accelerator Institute, the Toyota Research Institute, and the Jump ARCHES endowment through the Health Care Engineering Systems Center at Illinois and the OSF Foundation.

References

[1] Agarap, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *ICML*, 2020.

[3] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021.

[4] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Badrinath, A., Flet-Berliac, Y., Nie, A., and Brunskill, E. Waypoint transformer: Reinforcement learning via supervised learning with intermediate targets. In *NeurIPS*, 2023.

[6] Bastani, O., Pu, Y., and Solar-Lezama, A. Verifiable reinforcement learning via policy extraction. In *NeurIPS*, 2018.

[7] Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., and Bruna, J. When does return-conditioned supervised learning work for offline reinforcement learning? In *NeurIPS*, 2022.

[8] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[9] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[10] Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., et al. Do as i can, not as i say: Grounding language in robotic affordances. In *CoRL*, 2023.

[11] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[12] Chebotar, Y., Vuong, Q., Hausman, K., Xia, F., Lu, Y., Irpan, A., Kumar, A., Yu, T., Herzog, A., Pertsch, K., et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *CoRL*, 2023.

[13] Chen, C., Wang, X., Jin, Y., Dong, V. Y., Dong, L., Cao, J., Liu, Y., and Yan, R. Semi-offline reinforcement learning for optimized text generation. In *ICML*, 2023.

[14] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021.

[15] Chen, X., Wang, C., Zhou, Z., and Ross, K. Randomized ensembled double q-learning: Learning fast without a model. In *ICLR*, 2021.

[16] David, S. B., Zimerman, I., Nachmani, E., and Wolf, L. Decision s4: Efficient sequence-based rl via state spaces layers. In *ICLR*, 2022.

[17] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[18] Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? In *ICLR*, 2022.

[19] Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[20] Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. In *NeurIPS*, 2021.

[21] Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.

[22] Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. In *ICLR*, 2022.

[23] Garg, D., Hejna, J., Geist, M., and Ermon, S. Extreme q-learning: Maxent rl without entropy. In *ICLR*, 2023.

[24] Ghasemipour, S. K. S., Schuurmans, D., and Gu, S. S. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *ICML*, 2021.

[25] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[26] Hong, J., Kumar, A., and Levine, S. Confidence-conditioned value functions for offline reinforcement learning. In *ICLR*, 2023.

[27] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[28] Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[29] Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, 2020.

[30] Lee, J., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. In-context decision-making from supervised pretraining. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[31] Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. Multi-game decision transformers. In *NeurIPS*, 2022.

[32] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N. M. O., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[33] Lin, L., Bai, Y., and Mei, S. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. In *ICLR*, 2024.

[34] Liu, H. and Abbeel, P. Emergent agentic transformer from chain of hindsight experience. In *ICML*, 2023.

[35] Lyu, J., Ma, X., Li, X., and Lu, Z. Mildly conservative q-learning for offline reinforcement learning. In *NeurIPS*, 2022.

[36] Ma, Y., Xiao, C., Liang, H., and Hao, J. Rethinking decision transformer via hierarchical reinforcement learning. *arXiv preprint arXiv:2311.00267*, 2023.

[37] Mao, H., Zhao, R., Chen, H., Hao, J., Chen, Y., Li, D., Zhang, J., and Xiao, Z. Transformer in transformer as backbone for deep reinforcement learning. In *AAMAS*, 2024.

[38] Meng, L., Wen, M., Yang, Y., Le, C., Li, X., Zhang, W., Wen, Y., Zhang, H., Wang, J., and Xu, B. Offline pre-trained multi-agent decision transformer: One big sequence model tackles all smac tasks. *arXiv preprint arXiv:2112.02845*, 2021.

[39] Micheli, V., Alonso, E., and Fleuret, F. Transformers are sample efficient world models. In *ICLR*, 2023.

[40] Nair, A., Dalal, M., Gupta, A., and Levine, S. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[41] Nakamoto, M., Zhai, Y., Singh, A., Mark, M. S., Ma, Y., Finn, C., Kumar, A., and Levine, S. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. In *NeurIPS*, 2023.

[42] Ni, T., Ma, M., Eysenbach, B., and Bacon, P.-L. When do transformers shine in rl? decoupling memory from credit assignment. In *NeurIPS*, 2023.

[43] Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. In *ICML*, 2018.

[44] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[45] Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., et al. Stabilizing transformers for reinforcement learning. In *ICML*, 2020.

[46] Paster, K., McIlraith, S., and Ba, J. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. In *NeurIPS*, 2022.

[47] Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[48] Pinto, A. S., Kolesnikov, A., Shi, Y., Beyer, L., and Zhai, X. Tuning computer vision models with task rewards. In *ICML*, 2023.

[49] Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *RSS*, 2018.

[50] Robine, J., Höftmann, M., Uelwer, T., and Harmeling, S. Transformer-based world models are happy with 100k interactions. In *ICLR*, 2023.

[51] Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., Leibo, J. Z., Kuttler, H., Zisserman, A., Simonyan, K., et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

[52] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[53] Shang, J., Kahatapitiya, K., Li, X., and Ryoo, M. S. Starformer: Transformer with state-action-reward representations for visual reinforcement learning. In *ECCV*, 2022.

[54] Sharif, A. and Marijan, D. Evaluating the robustness of deep reinforcement learning for autonomous and adversarial policies in a multi-agent urban driving environment. In *QRS*, 2021.

[55] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *Science*, 2018.

[56] Song, Y., Zhou, Y., Sekhari, A., Bagnell, J. A., Krishnamurthy, A., and Sun, W. Hybrid rl: Using both offline and online data can make rl efficient. In *ICLR*, 2023.

[57] Sudhakaran, S. and Risi, S. Skill decision transformer. *Foundation Models for Decision Making Workshop at NeurIPS*, 2022.

[58] Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

[59] Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., Bennice, M., Fu, C., Ma, C., Jiao, J., et al. Jump-start reinforcement learning. In *ICML*, 2023.

[60] Wang, Z., Wang, H., and Qi, Y. J. T3gdt: Three-tier tokens to guide decision transformer for offline meta reinforcement learning. In *Robot Learning Workshop: Pretraining, Fine-Tuning, and Generalization with Large Scale Models in NeurIPS*, 2023.

[61] Wołczyk, M., Cupiał, B., Ostaszewski, M., Bortkiewicz, M., Zając, M., Pascanu, R., Kuciński, Ł., and Miłoś, P. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem. In *ICML*, 2024.

[62] Wu, J., Wu, H., Qiu, Z., Wang, J., and Long, M. Supported policy optimization for offline reinforcement learning. In *NeurIPS*, 2022.

[63] Wu, Y.-H., Wang, X., and Hamaya, M. Elastic decision transformer. In *NeurIPS*, 2023.

[64] Xie, Z., Lin, Z., Ye, D., Fu, Q., Wei, Y., and Li, S. Future-conditioned unsupervised pretraining for decision transformer. In *ICML*, 2023.

[65] Xu, M., Lu, Y., Shen, Y., Zhang, S., Zhao, D., and Gan, C. Hyper-decision transformer for efficient online policy adaptation. In *ICLR*, 2023.

[66] Yamagata, T., Khalil, A., and Santos-Rodriguez, R. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *ICML*, 2023.

[67] Yan, K., Schwing, A., and Wang, Y.-X. Ceip: Combining explicit and implicit priors for reinforcement learning with demonstrations. In *NeurIPS*, 2022.

[68] You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. In *ICLR*, 2020.

[69] Yu, Z. and Zhang, X. Actor-critic alignment for offline-to-online reinforcement learning. In *ICML*, 2023.

[70] Yuan, H., Zhang, C., Wang, H., Xie, F., Cai, P., Dong, H., and Lu, Z. Plan4MC: Skill reinforcement learning and planning for open-world Minecraft tasks. In *Foundation Models for Decision Making Workshop at NeurIPS 2023*, 2023.

[71] Yue, Y., Lu, R., Kang, B., Song, S., and Huang, G. Understanding, predicting and better resolving q-value divergence in offline-rl. *NeurIPS*, 2024.

[72] Zhang, H., Xu, W., and Yu, H. Policy expansion for bridging offline-to-online reinforcement learning. In *ICLR*, 2023.

[73] Zheng, H., Luo, X., Wei, P., Song, X., Li, D., and Jiang, J. Adaptive policy learning for offline-to-online reinforcement learning. In *AAAI*, 2023.

[74] Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In *ICML*, 2022.

[75] Zhou, B., Li, K., Jiang, J., and Lu, Z. Learning from visual observation via offline pretrained state-to-go transformer. In *NeurIPS*, 2023.

[76] Zhu, D., Wang, Y., Schmidhuber, J., and Elhoseiny, M. Guiding online reinforcement learning with action-free offline pretraining. *arXiv preprint arXiv:2301.12876*, 2023.