Rethinking Decision Transformer via Hierarchical Reinforcement Learning

Yi Ma [1] **Jianye Hao** [1 2] **Hebin Liang** [1] **Chenjun Xiao** [3]

Abstract

Decision Transformer (DT) is an innovative algorithm leveraging recent advances of the transformer architecture in reinforcement learning (RL). However, a notable limitation of DT is its reliance on recalling trajectories from datasets, losing the capability to seamlessly stitch sub-optimal trajectories together. In this work we introduce a general sequence modeling framework for studying sequential decision making through the lens of *Hierarchical RL*. At the time of making decisions, a *high-level* policy first proposes an ideal *prompt* for the current state, a *low-level* policy subsequently generates an action conditioned on the given prompt. We show DT emerges as a special case of this framework with certain choices of high-level and low-level policies, and discuss the potential failure of these choices. Inspired by these observations, we study how to jointly optimize the high-level and low-level policies to enable the stitching ability, which further leads to the development of new offline RL algorithms. Our empirical results clearly show that the proposed algorithms significantly surpass DT on several control and navigation benchmarks. We hope our contributions can inspire the integration of transformer architectures within the field of RL.

1. Introduction

One of the most remarkable characteristics observed in large sequence models, especially Transformer models, is the *in-context learning* ability (Radford et al., 2019; Brown et al., 2020; Ramesh et al., 2021; Gao et al., 2020; Akyürek et al., 2022; Garg et al., 2022; Laskin et al., 2022; Lee et al., 2023). With an appropriate *prompt*, a pre-trained transformer can learn new tasks without explicit supervision and additional parameter updates. *Decision Transformer (DT)* (Chen et al., 2021) is an innovative method that attempts to explore this idea for sequential decision making. Unlike traditional *reinforcement learning (RL)* algorithms, which learn a value function by bootstrapping or computing policy gradient, DT directly learns an autoregressive generative model from trajectory data using a causal transformer (Vaswani et al., 2017; Radford et al., 2019). This approach allows leveraging existing transformer architectures widely employed in language and vision tasks that are easy to scale, and benefitting from a substantial body of research focused on stable training of transformer (Radford et al., 2019; Brown et al., 2020; Fedus et al., 2022; Chowdhery et al., 2022).

DT is trained on trajectory data, $(R_0, s_0, a_0, \ldots, R_T, s_T, a_T)$, where R_t is the *return-to-go*, the sum of future rewards along the trajectory starting from time step t. This can be viewed as learning a model that predicts *what action should be taken at a given state in order to make so many returns*. Following this, we can view the return-to-go prompt as a *switch*, guiding the model in making decisions at test time. If such a model can be learned effectively and generalized well even for out-of-distribution return-to-go, it is reasonable to expect that DT can generate a better policy by prompting a higher return. Unfortunately, this seems to demand a level of generalization ability that is often too high in practical sequential decision-making problems. In fact, the key challenge facing DT is how to improve its robustness to the underlying data distribution, particularly when learning from trajectories collected by policies that are not close to optimal. Recent studies have indicated that for problems requiring the *stitching ability*, referring to the capability to integrate suboptimal trajectories from the data, DT cannot provide a significant advantage compared to behavior cloning (Fujimoto & Gu, 2021; Emmons et al., 2021; Kostrikov et al., 2022; Yamagata et al., 2023; Badrinath et al., 2023; Xiao et al., 2023). This further confirms that a naive return-to-go prompt is not good enough for solving complex sequential decision-making problems.

Recent progress on large language models showed that care-

[1]College of Intelligence and Computing, Tianjin University [2]Huawei, Noah's Ark Lab [3]The Chinese University of Hongkong, Shenzhen. Correspondence to: Jianye Hao <jianye.hao@tju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

fully tuned prompts, either human-written or self-discovered by the model, significantly boost the performance of transformer models (Lester et al., 2021; Singhal et al., 2022; Zhang et al., 2022; Wei et al., 2022; Wang et al., 2022a; Yao et al., 2023; Liu et al., 2023). In particular, it has been observed that the ability to perform complex reasoning naturally emerges in sufficiently large language models when they are presented with a few chain of thought demonstrations as exemplars in the prompts (Wei et al., 2022; Wang et al., 2022a; Yao et al., 2023). Driven by the significance of these works in language models, a question arises: *For RL, is it feasible to learn to automatically tune the prompt, such that a transformer-based sequential decision model is able to learn optimal control policies from offline data?* This paper attempts to address this problem. Our main contributions are:

- We present a generalized framework for studying decision-making through sequential modeling by connecting it with *Hierarchical Reinforcement Learning* (Nachum et al., 2018): a high-level policy first suggests a prompt for the current state, a low-level policy subsequently generates an action conditioned on the given prompt. We show DT can be recovered as a special case of this framework.
- We investigate when and why DT fails in terms of stitching sub-optimal trajectories. To overcome this drawback of DT, we investigate how to jointly optimize the high-level and low-level policies to enable the stitching capability. This further leads to the development of two new algorithms for offline RL. The joint policy optimization framework is our key contribution compared to previous studies on improving transformer-based decision models (Yamagata et al., 2023; Wu et al., 2023; Badrinath et al., 2023).
- We provide experiment results on several offline RL benchmarks, including locomotion control, navigation and robotics, to demonstrate the effectiveness of the proposed algorithms. Additionally, we conduct thorough ablation studies on the key components of our algorithms to gain deeper insights into their contributions. Through these ablation studies, we assess the impact of specific algorithmic designs on the overall performance.

2. Preliminaries

2.1. Offline Reinforcement Learning

We consider Markov Decision Process (MDP) determined by $M = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$ (Puterman, 2014), where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces. The discount factor is given by $\gamma \in [0, 1)$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ defines the transition dynamics[1]. Let $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$ be a trajectory. Its *return* is defined as the discounted sum of the rewards along the trajectory: $R = \sum_{t=0}^{T} \gamma^t r_t$. Given a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, we use $\mathbb{E}^\pi$ to denote the expectation under the distribution induced by the interconnection of π and the environment. The *value function* specifies the future discounted total reward obtained by following policy π,

$$V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \Big| s_0 = s \right], \tag{1}$$

There exists an *optimal policy* π^* that maximizes values for all states $s \in \mathcal{S}$.

In this work, we consider learning an optimal control policy from previously collected offline dataset, $\mathcal{D} = \{\tau_i\}_{i=0}^{n-1}$, consisting of n trajectories. Each trajectory is generated by the following procedure: an initial state $s_0 \sim \mu_0$ is sampled from the initial state distribution μ_0; for time step $t \geq 0$, $a_t \sim \pi_{\mathcal{D}}$, $s_{t+1} \sim P(\cdot|s_t, a_t), r_t = r(s_t, a_t)$, this process repeats until it reaches the maximum time step of the environment. Here $\pi_{\mathcal{D}}$ is an *unknown behavior policy*. In offline RL, the learning algorithm can only take samples from $\mathcal{D}$ without collecting new data from the environment (Levine et al., 2020).

2.2. Decision Transformer

Decision Transformer (DT) is an extraordinary example that bridges sequence modeling with decision-making (Chen et al., 2021). It shows that a sequential decision-making model can be made through minimal modification to the transformer architecture (Vaswani et al., 2017; Radford et al., 2019). It considers the following trajectory representation that enables autoregressive training and generation:

$$\tau = \left(\widehat{R}_0, s_0, a_0, \widehat{R}_1, s_1, a_1, \ldots, \widehat{R}_T, s_T, a_T \right). \tag{2}$$

Here $\widehat{R_t} = \sum_{i=t}^{T} r_i$ is the *return-to-go* starting from time step t. We denote $\pi_{\mathrm{DT}}(a_t|s_t, \widehat{R}_t, \tau_t)$ the DT policy, where $\tau_t = (s_0, a_0, \widehat{R}_0, \ldots, s_{t-1}a_{t-1}, \widehat{R}_{t-1})$[2] is the sub-trajectory before time step t. As pointed and verified by Lee et al. (2023), τ_t can be viewed as as a *context* input of a policy, which fully takes advantages of the in-context learning ability of transformer model for better generalization (Akyürek et al., 2022; Garg et al., 2022; Laskin et al., 2022).

DT assigns a desired return-to-go R^0, together with an initial state s_0 are used as the initialization input of the model. After executing the generated action, DT decrements the desired return by the achieved reward and continues this

[1] We use $\Delta(\mathcal{X})$ to denote the set of probability distributions over $\mathcal{X}$ for a finite set $\mathcal{X}$.

[2] We define τ_0 the empty sequence for completeness.

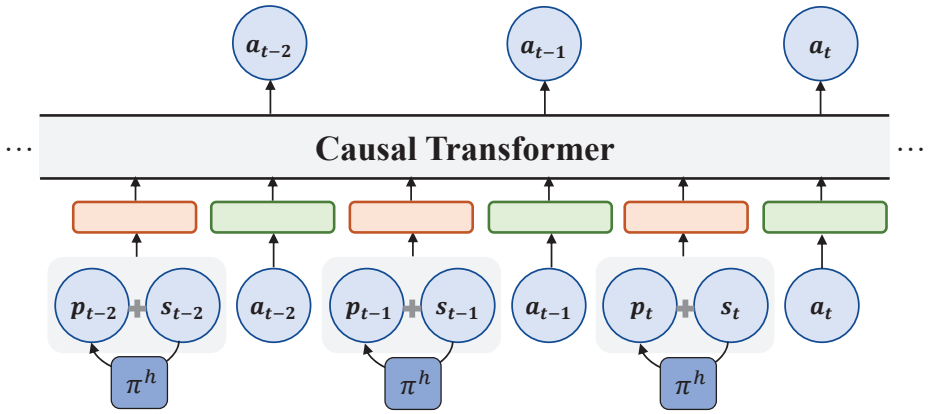

Figure 1. ADT architecture. The high-level policy generates prompts that inform the low-level policy to make decisions. We concatenate prompts with states instead of treating them as separate tokens. Embeddings of tokens are fed into a causal transformer that predicts actions auto-regressively.

process until the episode reaches termination. Chen et al. (2021) argues that the conditional prediction model is able to perform policy optimization without using dynamic programming. However, recent works observe that DT often shows inferior performance compared to dynamic programming based offline RL algorithms when the offline dataset consists of sub-optimal trajectories (Fujimoto & Gu, 2021; Emmons et al., 2021; Kostrikov et al., 2022).

3. Autotuned Decision Transformer

In this section, we present *Autotuned Decision Transformer (ADT)*, a new transformer-based decision model that is able to stitch sub-optimal trajectories from the offline dataset. Our algorithm is derived based on a general hierarchical decision framework where DT naturally emerges as a special case. Within this framework, we discuss how ADT overcomes several limitations of DT by automatically tuning the prompt for decision making.

3.1. Key Observations

Our algorithm is derived by considering a general framework that bridges transformer-based decision models with hierarchical reinforcement learning (HRL) (Nachum et al., 2018). In particular, we use the following hierarchical representation of policy

$$\pi(a|s) = \int_{\mathcal{P}} \pi^h(p|s) \cdot \pi^l(a|s,p)dp\,, \tag{3}$$

where $\mathcal{P}$ is a set of prompts. To make a decision, the high-level policy π^h first generates a prompt $p \in \mathcal{P}$, instructed by which the low-level policy π^l returns an action conditioned on p. DT naturally fits into this hierarchical decision framework. Consider the following value prompting mechanism. At state $s \in \mathcal{S}$, the high-level policy generates a real-value prompt $R \in \mathbb{R}$, representing *"I want to obtain R return starting from s."*. Informed by this prompt, the low-level policy responses an action $a \in \mathcal{A}$, *"Ok, if you want to obtain return R, you should take action a now."*. This is exactly what DT does. It applies a dummy high-level policy which initially picks a target return-to-go prompt and subsequently decrement it along the trajectory. The DT low-level policy, $\pi_{\text{DT}}(\cdot|s,R,\tau)$, learns to predict which action to take at state s in order to achieve return R given the context τ.

To better understand the failure of DT given sub-optimal data, we re-examine the illustrative example shown in Figure 2 of Chen et al. (2021). The dataset comprises random walk trajectories and their associated per-state return-to-go. Suppose that the DT policy π_{DT} perfectly memorizes all trajectory information contained in the dataset. The return-to-go prompt in fact acts as a *switch* to guide the model to make decisions. Let $\mathcal{T}(s)$ be the set of trajectories starting from s stored in the dataset, and $R(\tau)$ be the return of a trajectory τ. Given $R' \in \{R(\tau), \tau \in \mathcal{T}(s)\}$, π_{DT} is able to output an action that leads towards τ. Thus, given an *oracle return* $R^*(s) = \max_{\tau \in \mathcal{T}(s)} R(\tau)$, it is expected that DT is able to follow the optimal trajectory contained in the dataset following the switch.

There are several issues. *First*, the oracle return R^* is not known. The initial return-to-go prompt of DT is picked by hand and might not be consistent with the one observed in the dataset. This requires the model to generalize well for unseen return-to-go and decisions. *Second*, even though R^* is known for all states, memorizing trajectory information is still not enough for obtaining the stitching ability as R^* only serves a lower bound on the maximum achievable return. To understand this, consider an example in Figure 2 with two trajectories $a \to b \to c$, and $d \to b \to e$. Suppose that e leads to a return of 10, while c leads to a return of 0. In

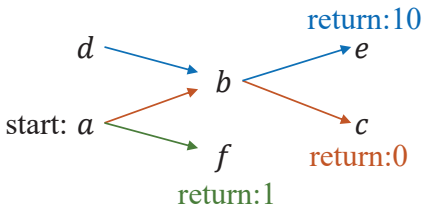

Figure 2. Illustrative example.

this case, using 10 as the return-to-go prompt at state b, DT should be able to switch to the desired trajectory. However, the information "leaning towards c can achieve a return of 10" does not pass to a during training, since the trajectory $a \to b \to e$ does not exist in the data. If the offline data contains another trajectory that starts from a and leads to f with a mediocre return (e.g. 1), DT might switch to that trajectory at a using 10 as the return-to-go prompt, missing a more promising path. Thus, making predictions conditioned on return-to-go alone is not enough for policy optimization. Some form of information backpropagation is still required.

3.2. Algorithms

ADT jointly optimizes the hierarchical policies to overcomes the limitations of DT discussed above. An illustration of ADT architecture is provided in Figure 1. Similar to DT, ADT applies a transformer model for the low-level policy. Instead of (2), it considers the following trajectory representation,

$$\tau = (p_0, s_0, a_0, p_1, s_1, a_1, \dots, p_T, s_T, a_T) \ . \tag{4}$$

Here p_i is the prompt generated by the high-level policy $p_i \sim \pi^h(\cdot|s_i)$, replacing the return-to-go prompt used by DT. That is, for each trajectory in the offline dataset, we relabel it by adding a prompt generated by the high-level policies for each transition. Armed with this general hierarchical decision framework, we propose two algorithms that apply different high-level prompting generation strategy while sharing a unified low-level policy optimization framework. We learn a high-level policy $\pi_\omega \approx \pi^h$ with parameters ϕ, and a low-level policy $\pi_\theta \approx \pi^l$ with parameters θ. Here 'joint training' is used to indicate both the prompt input to the low-level policy and the low-level policy itself are trained, while in DT the prompt is obtained via manual prompt instead of well-trained policy. In practice, we train the high-level policy and the low-level policy in a sequential manner.

3.2.1. Value-prompted Autotuned Decision Transformer

Our first algorithm, *Value-promped Autotuned Decision Transformer (V-ADT)*, uses scalar values as prompts. But unlike DT, it applies a more principled design of value prompts instead of return-to-go. V-ADT aims to answer two questions: what is the maximum achievable value starting from a state s, and what action should be taken to achieve such a value? To answer these, we view the offline dataset $\mathcal{D}$ as an *empirical MDP*, $M_\mathcal{D} = \{\mathcal{S}_\mathcal{D}, \mathcal{A}, P_\mathcal{D}, r, \gamma\}$, where $\mathcal{S}_\mathcal{D} \subseteq \mathcal{S}$ is the set of observed states in the data, $P_\mathcal{D}$ is the transition, which is an empirical estimation of the original transition P (Fujimoto et al., 2019). The optimal value of this empirical MDP is

$$V^*_\mathcal{D}(s) = \max_{a:\pi_\mathcal{D}(a|s)>0} r(s,a) + \gamma \mathbb{E}_{s' \sim P_\mathcal{D}(\cdot|s,a)} \left[V^*_\mathcal{D}(s')\right] \ . \tag{5}$$

Let $Q^*_\mathcal{D}(s,a)$ be the corresponding state-action value. $V^*_\mathcal{D}$ is known as the *in-sample optimal value* in offline RL (Fujimoto et al., 2018; Kostrikov et al., 2022; Xiao et al., 2023). Computing this value requires to perform dynamic programming without querying out-of-distribution actions. We apply Implicit Q-learning (IQL) to learn $V_\phi \approx V^*_\mathcal{D}$ and $Q_\psi \approx Q^*_\mathcal{D}$ with parameters ϕ, ψ (Kostrikov et al., 2022). Details of IQL are presented in the Appendix. We now describe how V-ADT jointly optimizes high and low level policies to facilitate stitching.

High-Level policy V-ADT considers $\mathcal{P} = \mathbb{R}$ and adopts a deterministic policy $\pi_\omega : \mathcal{S} \to \mathbb{R}$, which predicts the in-sample optimal value $\pi_\omega \approx V^*_\mathcal{D}$. Since we already have an approximated in-sample optimal value V_ϕ, we let $\pi_\omega = V_\phi$. This high-level policy offers two key advantages. *First*, this approach efficiently facilitates information backpropagation towards earlier states on a trajectory, addressing a major limitation of DT. This is achieved by using $V^*_\mathcal{D}$ as the value prompt, ensuring that we have precise knowledge of the maximum achievable return for any state. Making predictions conditioned on $R^*(s)$ is not enough for policy optimization, since $R^*(s) = \max_{\tau \in \mathcal{T}(s)} R(\tau)$ only gives a lower bound on $V^*_\mathcal{D}(s)$ and thus would be a weaker guidance (see Section 3.1 for detailed discussions). *Second*, the definition of $V^*_\mathcal{D}$ exclusively focuses on the optimal value derived from observed data and thus avoids out-of-distribution returns. This prevents the low-level policy from making decisions conditioned on prompts that require extrapolation.

Low-Level policy Directly training the model to predict the trajectory, as done in DT, is not suitable for our approach. This is because the action a_t observed in the data may not necessarily correspond to the action at state s_t that leads to the return $V^*_\mathcal{D}(s_t)$. However, the probability of selecting a_t should be proportional to the value of this action. Thus, we use *advantage-weighted regression* to learn the low-level policy (Peng et al., 2019; Kostrikov et al., 2022; Xiao et al.,

2023): given trajectory data (4) the objective is defined as

$$\mathcal{L}(\theta) = -\sum_{t=0}^{T} \exp\left(\frac{\mathcal{A}_{\text{low}}}{\alpha}\right) \cdot \log \pi_\theta(a_t|s_t, \pi_\omega(s_t)) \,. \quad (6)$$

where $\mathcal{A}_{\text{low}} = Q_\psi(s_t, a_t, \pi_\omega(s_t)) - V_\phi(s_t, \pi_\omega(s_t))$ and $\alpha > 0$ is a hyper-parameter. The low-level policy takes the output of high-level policy as input. This guarantees no discrepancy between train and test value prompt used by the policies. We note that the only difference of this compared to the standard maximum log-likelihood objective for sequence modeling is to apply a weighting for each transition. One can easily implement this with trajectory data for a transformer. In practice we also observe that the tokenization scheme when processing the trajectory data affects the performance of ADT. Instead of treating the prompt p_t as a single token as in DT, we find it is beneficial to concatenate p_t and s_t together and tokenize the concatenated vector. We provide an ablation study on this in Section 5.2.3. This completes the description of V-ADT.

3.2.2. Goal-prompted Autotuned Decision Transformer

In HRL, the high-level policy often considers a latent action space. Typical choices of latent actions includes *sub-goal* (Nachum et al., 2018; Park et al., 2023), *skills* (Ajay et al., 2020; Jiang et al., 2022), and *options* (Sutton et al., 1999; Bacon et al., 2017; Klissarov & Machado, 2023). We consider goal-reaching problem as an example and use sub-goals as latent actions, which leads to our second algorithm, *Goal-promped Autotuned Decision Transformer (G-ADT)*. Let $\mathcal{G}$ be the goal space[3]. The goal-conditioned reward function $r(s, a, g)$ provides the reward of taking action a at state s for reaching the goal $g \in \mathcal{G}$. Let $V(s, g)$ be the universal value function defined by the goal-conditioned rewards (Nachum et al., 2018; Schaul et al., 2015). Similarly, we define $V^*_{\mathcal{D}}(s, g)$ and $Q^*_{\mathcal{D}}(s, a, g)$ the in-sample optimal universal value function. We also train $V_\phi \approx V^*_{\mathcal{D}}$ and $Q_\psi \approx Q^*_{\mathcal{D}}$ to approximate the universal value functions. We now describe how G-ADT jointly optimizes the policies.

High-Level policy G-ADT considers $\mathcal{P} = \mathcal{G}$ and uses a high-level policy $\pi_\omega : \mathcal{S} \to \mathcal{G}$. To find a shorter path, the high-level policy π_ω generates a sequence of sub-goals $g_t = \pi_\omega(s_t)$ that guides the learner step-by-step towards the final goal. We use a sub-goal that lies in k-steps further from the current state, where k is a hyper-parameter of the algorithm tuned for each domain (Badrinath et al., 2023; Park et al., 2023). In particular, given trajectory data (4), the high-level policy learns the optimal *k-steps jump* using the recently proposed Hierarchical Implicit Q-learning (HIQL) algorithms (Park et al., 2023):

$$\mathcal{L}(\phi) = -\sum_{t=0}^{T} \exp\left(\frac{\mathcal{A}_{\text{high}}}{\alpha}\right) \log \pi_\omega(s_{t+k}|s_t, g) \,.$$

$$\mathcal{A}_{\text{high}} = \sum_{t'=t}^{k-1} \gamma^{t'-t} r(s_{t'}, a_{t'}, g) + \gamma^k V_\phi(s_{t+k}, g) - V_\phi(s_t, g).$$

Low-Level policy The low-level policy in G-ADT learns to reach the sub-goal generated by the high-level policy. G-ADT shares the same low-level policy objective as V-ADT. Given trajectory data (4), it considers the following

$$\mathcal{L}(\theta) = -\sum_{t=0}^{T} \exp\left(\frac{\mathcal{A}_{\text{low}}}{\alpha}\right) \cdot \log \pi_\theta(a_t|s_t, \pi_\omega(s_t)) \,,$$

where $\mathcal{A}_{\text{low}} = Q_\psi(s_t, a_t, \pi_\omega(s_t)) - V_\phi(s_t, \pi_\omega(s_t))$. Note that this is exactly the same as (6) except that the advantages $\mathcal{A}_{\text{low}}$ are computed by universal value functions. G-ADT also applies the same tokenization method as V-ADT by first concatenating $\pi_\omega(s_t)$ and s_t together. This concludes the description of the G-ADT algorithm.

4. Discussions

Types of Prompts Xu et al. (2022) introduces Prompt-DT, which leverages the sequential modeling ability of the Transformer architecture, using expert trajectory prompts as task-specific guides to adapt to unseen tasks without extra finetuning. Reed et al. (2022) have delved into the potential scalability of transformer-based decision models through prompting. They show that a causal transformer, trained on multi-task offline datasets, showcases remarkable adaptability to new tasks through fine-tuning. The adaptability is achieved by providing a sequence prompt as the input of the transformer model, typically represented as a trajectory of expert demonstrations. Unlike such expert trajectory prompts, our prompt can be seen as a latent action generated by the high-level policy, serving as guidance for the low-level policy to inform its decision-making process.

Comparison of other DT Enhancements Several recent works have been proposed to overcome the limitations of DT. Correia & Alexandre (2022) employs a dual transformer architecture to design Hierarchical DT (HDT), where a high-level mechanism selects sub-goal states from demonstration data to guide a low-level controller in task completion to improve DT. Yamagata et al. (2023) relabelled trajectory data by replacing return-to-go with values learned by offline RL algorithms. Badrinath et al. (2023) proposed to use sub-goal as prompt, guiding the DT policy to find shorter path in navigation problems. Wu et al. (2023) learned maximum achievable returns, $R^*(s) = \max_{\tau \in \mathcal{T}(s)} R(\tau)$, to boost the stitching ability of DT at decision time. Liu & Abbeel

[3]The goal space and state space could be the same (Nachum et al., 2018; Park et al., 2023)

(2023) structured trajectory data by relabelling the target return for each trajectory as the maximum total reward within a sequence of trajectories. Their findings showed that this approach enabled a transformer-based decision model to improve itself during both training and testing time. Compared to these previous efforts, ADT introduces a principled framework of hierarchical policy optimization. Our practical studies show that the joint optimization of high and low level policies is the key to boost the performance of transformer-based decision models.

5. Experiment

We investigate three primary questions in our experiments. *First*, how well does ADT perform on offline RL tasks compared to prior DT-based methods? *Second*, is it essential to auto-tune prompts for transformer-based decision model? *Third*, what is the influence of various implementation details within an ADT on its overall performance? We refer readers to Appendix A for additional details and supplementary experiments.

Benchmarks and Baseline Algorithms We leverage datasets across several domains including Gym-Mujoco, AntMaze, and FrankaKitchen from the offline RL benchmark D4RL (Fu et al., 2020). We compare the performance of ADT with several representative baselines including (1) *offline RL methods*: TD3+BC (Fujimoto & Gu, 2021), CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2022); (2) *valued-conditioned methods*: Decision Transformer (DT) (Chen et al., 2021), Q-Learning Decision Transformer (QLDT) (Yamagata et al., 2023) and Elastic Decision Transformer (EDT) (Wu et al., 2023); (3) *goal-conditioned methods*: HIQL (Park et al., 2023), RvS (Emmons et al., 2021), Hierarchical Decision Transformer (HDT) (Correia & Alexandre, 2022) and Waypoint Transformer (WT) (Badrinath et al., 2023). All the baseline results except for QLDT are obtained from (Badrinath et al., 2023) and (Park et al., 2023) or by running the codes of CORL repository (Tarasov et al., 2022). For HIQL, we present HIQL's performance with the goal representation in Kitchen and that without goal representation in AntMaze, as per our implementation in ADT, to ensure fair comparison. QLDT and the transformer-based actor of ADT are implemented based on the DT codes in CORL, with similar architecture. Details are given in Appendix. The critics and the policies to generate prompts used in ADT are re-implemented in PyTorch following the official codes of IQL and HIQL. In all conducted experiments, five distinct random seeds are employed. Results are depicted with 95% confidence intervals, represented by shaded areas in figures and expressed as standard deviations in tables. The reported results of ADT in tables are obtained by evaluating the final models. Note that as HDT reports the best score, to ensure fair comparison, we report best normalized scores for both HDT and ADT in Table 4.

Implementation of ADT The implementations of ADT is based on CORL repository (Tarasov et al., 2022). A key different between the implementation of ADT and DT is that we follow the way in (Badrinath et al., 2023) that we concatenate the (scaled) prompt and state, then the concatenated information and the action are treated as two tokens per timestep. In practice, we first train the high-level policy of ADT, then train the low-level policy. For each time of evaluation, we run the algorithms for 10 episodes for MuJoCo datasets, 50 episodes for Kitchen datasets, and 100 episodes for AntMaze datasets. Codes for reproducing our results are provided here. Detailed settings of other hyperparameters are provided in Appendix A.2.

5.1. Main Results

Tables 1 and 2 present the performance of two variations of ADT evaluated on offline datasets. ADT significantly outperforms prior transformer-based decision making algorithms. Compared to DT and QLDT, two transformer-based algorithms for decision making, V-ADT exhibits significant superiority especially on AntMaze and Kitchen which require the stitching ability to success. Meanwhile, Table 2 shows that G-ADT significantly outperforms WT, an algorithm that uses sub-goal as prompt for a transformer policy. We note that ADT enjoys comparable performance with state-of-the-art offline RL methods. For example, V-ADT outperforms all offline RL baselines in Mujoco problems. In AntMaze and Kitchen, V-ADT matches the performance of IQL, and significantly outperforms TD3+BC and CQL. Table 2 concludes with similar findings for G-ADT.

5.2. Ablation Studies

5.2.1. Effectiveness of Prompting

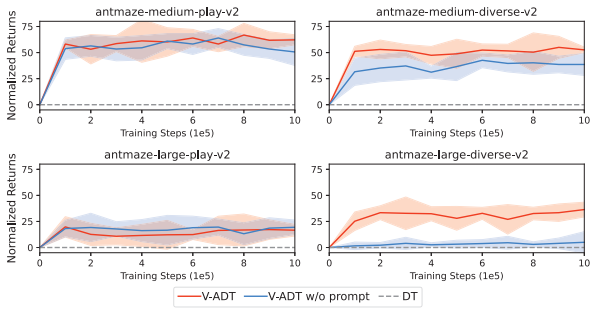

Figure 3. Learning curves of V-ADT with and without value prompt. The value prompt significantly boosts the performance in harder diverse datasets.

In Section 3.1 we discuss an illustrative example showing

Table 1. Average normalized scores of V-ADT, value-conditioned (DT and QLDT), and value-based RL methods. The methods on the right of the vertical line are DT-based methods. The top scores among all DT-based methods are highlighted in **bold**.

Environment	TD3+BC	CQL	IQL	DT	QLDT	EDT	V-ADT
halfcheetah-medium-v2	48.3 ± 0.3	44.0 ± 5.4	47.4 ± 0.2	42.4 ± 0.2	42.3 ± 0.4	42.5 ± 0.9	**48.7 ± 0.2**
hopper-medium-v2	59.3 ± 4.2	58.5 ± 2.1	66.2 ± 5.7	63.5 ± 5.2	**66.5 ± 6.3**	63.5 ± 5.8	60.6 ± 2.8
walker2d-medium-v2	83.7 ± 2.1	72.5 ± 0.8	78.3 ± 8.7	69.2 ± 4.9	67.1 ± 3.2	72.8 ± 6.2	**80.9 ± 3.5**
halfcheetah-medium-replay-v2	44.6 ± 0.5	45.5 ± 0.5	44.2 ± 1.2	35.4 ± 1.6	35.6 ± 0.5	37.8 ± 1.5	**42.8 ± 0.2**
hopper-medium-replay-v2	60.9 ± 18.8	95.0 ± 6.4	94.7 ± 8.6	43.3 ± 23.9	52.1 ± 20.3	**89.0 ± 8.3**	83.5 ± 9.5
walker2d-medium-replay-v2	81.8 ± 5.5	77.2 ± 5.5	73.8 ± 7.1	58.9 ± 7.1	58.2 ± 5.1	74.8 ± 4.9	**86.3 ± 1.4**
halfcheetah-medium-expert-v2	90.7 ± 4.3	91.6 ± 2.8	86.7 ± 5.3	84.9 ± 1.6	79.0 ± 7.2	-	**91.7 ± 1.5**
hopper-medium-expert-v2	98.0 ± 9.4	105.4 ± 6.8	91.5 ± 14.3	100.6 ± 8.3	94.2 ± 8.2	-	**101.6 ± 5.4**
walker2d-medium-expert-v2	110.1 ± 0.5	108.8 ± 0.7	109.6 ± 1.0	89.6 ± 38.4	101.7 ± 3.4	-	**112.1 ± 0.4**
gym-avg	75.3 ± 4.9	77.6 ± 3.4	76.9 ± 5.8	65.3 ± 10.1	66.3 ± 6.1	-	**78.7 ± 2.8**
antmaze-umaze-v2	78.6	74.0	87.5 ± 2.6	53.6 ± 7.3	67.2 ± 2.3	-	**88.2 ± 2.5**
antmaze-umaze-diverse-v2	71.4	84.0	62.2 ± 13.8	42.2 ± 5.4	**62.1 ± 1.6**	-	58.6 ± 4.3
antmaze-medium-play-v2	10.6	61.2	71.2 ± 7.3	0.0 ± 0.0	0.0 ± 0.0	-	**62.2 ± 2.5**
antmaze-medium-diverse-v2	3.0	53.7	70.0 ± 10.9	0.0 ± 0.0	0.0 ± 0.0	-	**52.6 ± 1.4**
antmaze-large-play-v2	0.2	15.8	39.6 ± 5.8	0.0 ± 0.0	0.0 ± 0.0	-	**16.6 ± 2.9**
antmaze-large-diverse-v2	0.0	14.9	47.5 ± 9.5	0.0 ± 0.0	0.0 ± 0.0	-	**36.4 ± 3.6**
antmaze-avg	27.3	50.6	63.0 ± 8.3	16.0 ± 2.1	21.6 ± 0.7	-	**52.4 ± 2.9**
kitchen-complete-v0	25.0 ± 8.8	43.8	62.5	46.5 ± 3.0	38.8 ± 15.8	-	**55.1 ± 1.4**
kitchen-partial-v0	38.3 ± 3.1	49.8	46.3	31.4 ± 19.5	36.9 ± 10.7	-	**46.0 ± 1.6**
kitchen-mixed-v0	45.1 ± 9.5	51.0	51.0	25.8 ± 5.0	17.7 ± 9.5	-	**46.8 ± 6.3**
kitchen-avg	36.1 ± 7.1	48.2	53.3	34.6 ± 9.2	30.5 ± 12.0	-	**49.3 ± 3.1**
average	52.7	63.7	68.3	43.8 ± 7.3	45.4 ± 5.3	-	**65.0 ± 2.9**

Table 2. Performance of G-ADT across all datasets. The methods on the right of the vertical line are transformer-based methods, the top scores among which are highlighted in **bold**.

Environment	RvS-R/G	HIQL	WT	G-ADT
antmaze-umaze-v2	65.4 ± 4.9	83.9 ± 5.3	64.9 ± 6.1	**83.8 ± 2.3**
antmaze-umaze-diverse-v2	60.9 ± 2.5	87.6 ± 4.8	71.5 ± 7.6	**83.0 ± 3.1**
antmaze-medium-play-v2	58.1 ± 12.7	89.9 ± 3.5	62.8 ± 5.8	**82.0 ± 1.7**
antmaze-medium-diverse-v2	67.3 ± 8.0	87.0 ± 8.4	66.7 ± 3.9	**83.4 ± 1.9**
antmaze-large-play-v2	32.4 ± 10.5	87.3 ± 3.7	**72.5 ± 2.8**	71.0 ± 1.3
antmaze-large-diverse-v2	36.9 ± 4.8	81.2 ± 6.6	**72.0 ± 3.4**	65.4 ± 4.9
antmaze-avg	53.5 ± 7.2	86.2 ± 5.4	68.4 ± 4.9	**78.1 ± 2.5**
kitchen-complete-v0	50.2 ± 3.6	43.8 ± 19.5	49.2 ± 4.6	**51.4 ± 1.7**
kitchen-partial-v0	51.4 ± 2.6	65.0 ± 9.2	63.8 ± 3.5	**64.2 ± 5.1**
kitchen-mixed-v0	60.3 ± 9.4	67.7 ± 6.8	**70.9 ± 2.1**	69.2 ± 3.3
kitchen-avg	54.0 ± 5.2	58.8 ± 11.8	61.3 ± 3.4	**61.6 ± 3.4**
average	53.7 ± 6.5	77.1 ± 7.5	66.0 ± 4.4	**72.6 ± 2.8**

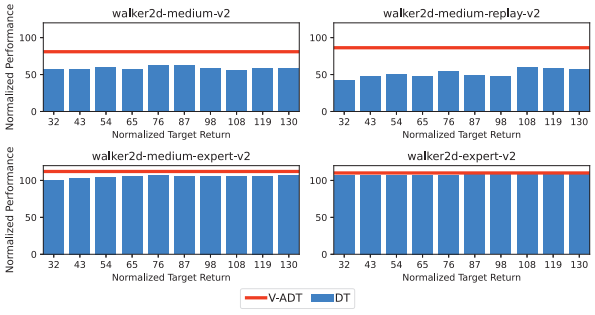

Figure 4. Average normalized results of DT using different prompt. Incorporating manual prompt engineering could not help DT outperform V-ADT.

how value-based conditional prediction can be leveraged to solve sequential decision making problem. However, it is still unclear how much the value prompt contributes to the remarkable empirical performance of V-ADT. This is particularly important to understand as by removing the value prompt, our low-level policy optimization objective (6) becomes exactly the same as advantage-weighted regression (Peng et al., 2019) with a transformer policy. We thus compare the performance of V-ADT with and without using value prompts in Figure 3. Although the value prompt seems to be less useful for the play datasets, it significantly improves the performance of V-ADT for the much harder diverse datasets. This confirms the effectiveness of value prompting for solving complex problems. In addition, compared with vanilla-DT that only imitates the actions at each state, V-ADT can still reach better performance without the prompt. This could be attributed to the using of advantage-weighted regression to learn the low-level policy. In this way, the policy could find the best actions leading to the in-sample optimal return in the dataset, which is referred to the stitching ability.

The main hypothesis behind ADT is that it is essential to learn a policy for adaptive prompt generation in order to make transformer-based sequential decision models able to learn optimal control policies. Since the initial return-to-go prompt of DT is a tunable hyper-parameter, a nature question follows: is it possible to match the performance of ADT through manual prompt tuning? Figure 4 delineates the results of DT using different target returns on four different walker2d datasets. The x-axis of each subfigure represents

the normalized target return input into DT, while the y-axis portrays the corresponding evaluation performance. Empirical results indicate that manual modifications to the target return could not improve the performance of DT, with its performance persistently lagging behind V-ADT. We also note that there is no single prompt that performs universally well across all domains. This highlights that the utility of prompt in DT appears constrained, particularly when working with datasets sourced from unimodal behavior policy.

5.2.2. Effectiveness of Low-Level Policy Optimization Objective

We claim that the sequence prediction loss used by DT does not suit our low-level policy optimization. To verify this claim, we implement a variant of ADT which uses the original DT objective to learn the low-level policy while still keeping learning an adaptive high-level policy. Figure 5 presents a comparison between this baseline and ADT. From the results we observe substantial improvement in performance of both V-ADT and G-ADT when (6) is leveraged. In particular, without using (6) to optimize the low-level policy, the effectiveness of auto-tuned prompting is notably compromised. This also strengthens the need of joint policy optimization of high and low level policies.

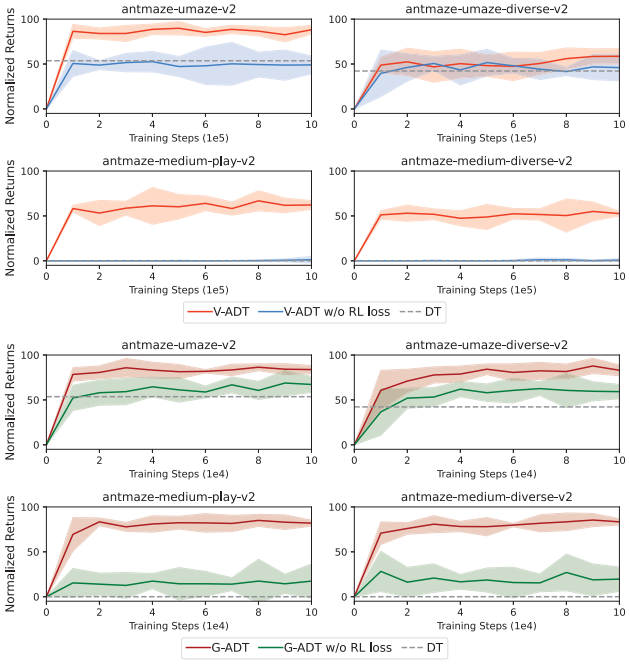

Figure 5. Learning curves of V-ADT and G-ADT with and without using (6). The results demonstrate that (6) is essential in empowering DT with stitching ability to achieve superior performance.

5.2.3. Effectiveness of Tokenization Strategies

In ADT, we diverge from the methodology presented in (Chen et al., 2021) where individual tokens are produced for each input component: return-to-go prompt, state, and action. Instead, we opt for a concatenated representation of prompts and states. Figure 6 presents a comparative analysis between these two tokenization strategies. We observe that our tokenization method contributes to superior performance both for V-ADT and G-ADT.

We postulate that this is attributed to the design of high-policy, which ensures a high degree of correlation between states and the corresponding ideal prompts. Thus we assert that the states and the corresponding prompts should be treated with equal significance when computing attention within the transformer's internal architecture.

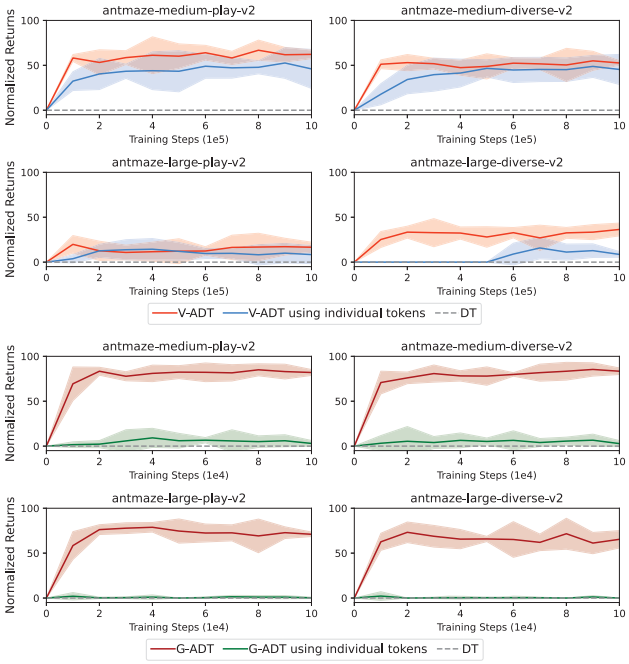

Figure 6. Learning curves of ADT with different tokenization strategies. Our design contributes to superiority by equally treating the states and related prompts when computing attention.

5.2.4. Why ADT falls short of IQL and HIQL?

It can be observed that the overall performance of ADT falls short of IQL and HIQL from Table 1 and 2. Our main conjecture is that **Transformer as a function approximator is harder to optimize compared to MLP for standard RL algorithms**. To verify this, we first implement an oracle algorithm, which distills the IQL policy using a transformer with supervised learning objective (Oracle in Table 3). The oracle algorithm matches the performance of IQL, suggest-

ing that the transformer architecture is not the bottleneck. We then implement another baseline, named IQL-Trans (i.e., V-ADT w/o prompt in Figure 3), by replacing the MLP policy with a transformer policy for IQL. As shown in Table 3, the performance of IQL-Trans cannot match the original IQL, further supporting our conjecture. Previous study also aligns with our findings that MLP is competitive with and sometimes more effective than Transformer in single task offline RL benchmarks (see Table 1 of RvS (Emmons et al., 2021)). The advantage of ADT over IQL-Trans is mainly contributed to the joint optimization of hierarchical policies (the high-level policy optimizes the prompt and the the low-level policy is optimized based on the prompt), since this is the key difference between these two algorithms. Finally, we note that there is still a performance gap between ADT and the oracle algorithm. This motivates the investigation of other techniques to improve transformer-based decision models, which we leave as our future work.

Table 3. Investigations on using different policy bases

Environment	V-ADT	IQL-Trans	Oracle	IQL
antmaze-medium-play-v2	62.2 ± 2.5	50.6 ± 6.6	69.0 ± 1.8	71.2 ± 7.3
antmaze-medium-diverse-v2	52.6 ± 1.4	38.6 ± 5.4	64.8 ± 6.5	70.0 ± 10.9
antmaze-large-play-v2	16.6 ± 2.9	19.4 ± 3.6	50.0 ± 1.7	39.6 ± 5.8
antmaze-large-diverse-v2	36.4 ± 3.6	5.0 ± 5.2	33.4 ± 5.3	47.5 ± 9.5

6. Comparison with HDT

We provide comparison between Hierarchical Decision Transformer (HDT) and ADT in Table 4. As HDT reports the best score, to ensure fair comparison, we report best normalized scores for both HDT and V-ADT. The results of HDT are directly taken and transferred to the normalized score using the function provided in D4RL. As shown in Table 4, except for hopper-medium, ADT outperforms HDT on all datasets. The overall performance of ADT is also significantly better than that of HDT.

Table 4. Comparison with HDT

	HDT	V-ADT
halfcheetah-medium	44.2	**49.9**
hopper-medium	95.0	81.3
walker2d-medium	84.5	**89.5**
kitchen-complete	65.0	**66.0**
maze2d-medium	66.2	**120.2**
average	71.0	**81.3**

7. Conclusion

We propose to rethink transformer-based decision models through a hierarchical decision-making framework. Armed with this, we introduce Autotuned Decision Transformer (ADT), which jointly optimizes the hierarchical policies for better performance when learning from sub-optimal data. ADT designed from Hierarchical RL is to provide general framework that high-level policy considers a latent action space and the low-level policy considers the control action to achieve the guidance given by the high-level latent action. We provide two widely used latent action space as two practical implementations of ADT. On standard offline RL benchmarks, we show ADT significantly outperforms previous transformer-based decision making algorithms.

Our primary focus for future work is to investigate the following problems. *First*, besides employing values and subgoals as latent actions generated by the high-level policy, other options for latent actions in hierarchical RL encompass skills (Ajay et al., 2020) and options (Sutton et al., 1999). We would like to investigate the potential extensions of ADT by incorporating skills and options. *Second*, according to the reward hypothesis, goals can be conceptualized as the maximization of expected value through the cumulative sum of a reward signal (Silver et al., 2021; Bowling et al., 2023). Can we establish a unified framework that bridges value-prompted ADT and goal-prompted ADT? *Finally*, according to our experiments, the advantages of substituting conventional architectures with transformer models in RL remain uncertain. Previous studies have indicated that the incorporation of transformers in RL is most advantageous when dealing with extensive and diverse datasets (Chebotar et al., 2023). With this in mind, we intend to apply ADT to create foundational decision-making models for learning multi-modal and multi-task policies in realistic scenarios (Wang et al., 2020; Ma et al., 2021; Zheng et al., 2019; Zhou et al., 2020; Wang et al., 2022b).

Impact Statement

By rethinking the Decision Transformer model, this work addresses previous limitations and enhances the ability to stitch sub-optimal trajectories. This work sets a precedent for future research in the integration of transformers with reinforcement learning. It opens up possibilities for further exploration into how hierarchical structures can enhance learning models, potentially leading to more groundbreaking discoveries in the field. This improvement is crucial for complex decision-making scenarios, potentially impacting domains like robotics, autonomous systems, and complex game environments.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant Nos. 92370132, 62106172), the Science and Technology on Information Systems Engineering Laboratory (Grant Nos. WDZC20235250409, 6142101220304), and the Xiaomi Young Talents Program of Xiaomi Foundation.

References

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Badrinath, A., Flet-Berliac, Y., Nie, A., and Brunskill, E. Waypoint transformer: Reinforcement learning via supervised learning with intermediate targets. *arXiv preprint arXiv:2306.14069*, 2023.

Bowling, M., Martin, J. D., Abel, D., and Dabney, W. Settling the reward hypothesis. In *International Conference on Machine Learning*, pp. 3003–3020. PMLR, 2023.

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. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chebotar, Y., Vuong, Q., Irpan, A., Hausman, K., Xia, F., Lu, Y., Kumar, A., Yu, T., Herzog, A., Pertsch, K., et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. *arXiv preprint arXiv:2309.10150*, 2023.

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. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Correia, A. and Alexandre, L. A. Hierarchical decision transformer. *arXiv preprint arXiv:2209.10447*, 2022.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? In *International Conference on Learning Representations*, 2021.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

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.

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

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.

Klissarov, M. and Machado, M. C. Deep laplacian-based options for temporally-extended exploration. *arXiv preprint arXiv:2301.11181*, 2023.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liu, H. and Abbeel, P. Emergent agentic transformer from chain of hindsight experience. *arXiv preprint arXiv:2305.16554*, 2023.

Liu, H., Sferrazza, C., and Abbeel, P. Chain of hindsight aligns language models with feedback. *arXiv preprint arXiv:2302.02676*, 3, 2023.

Ma, Y., Hao, X., Hao, J., Lu, J., Liu, X., Xialiang, T., Yuan, M., Li, Z., Tang, J., and Meng, Z. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. *Advances in neural information processing systems*, 34: 23609–23620, 2021.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Park, S., Ghosh, D., Eysenbach, B., and Levine, S. Hiql: Offline goal-conditioned rl with latent states as actions. *arXiv preprint arXiv:2307.11949*, 2023.

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.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*, 2022.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=SyAS49bBcv.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022a.

Wang, Z., Jusup, M., Guo, H., Shi, L., Geček, S., Anand, M., Perc, M., Bauch, C. T., Kurths, J., Boccaletti, S., et al. Communicating sentiment and outlook reverses inaction against collective risks. *Proceedings of the National Academy of Sciences*, 117(30):17650–17655, 2020.

Wang, Z., Mu, C., Hu, S., Chu, C., and Li, X. Modelling the dynamics of regret minimization in large agent populations: a master equation approach. In *IJCAI*, pp. 534–540, 2022b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

Wu, Y.-H., Wang, X., and Hamaya, M. Elastic decision transformer. *arXiv preprint arXiv:2307.02484*, 2023.

Xiao, C., Wang, H., Pan, Y., White, A., and White, M. The in-sample softmax for offline reinforcement learning. *arXiv preprint arXiv:2302.14372*, 2023.

Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., and Gan, C. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pp. 24631–24645. PMLR, 2022.

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

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

Zheng, Y., Xie, X., Su, T., Ma, L., Hao, J., Meng, Z., Liu, Y., Shen, R., Chen, Y., and Fan, C. Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 772–784. IEEE, 2019.

Zhou, M., Luo, J., Villella, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., Alban, M., Fadakar, I., Chen, Z., Huang, A. C., Wen, Y., Hassanzadeh, K., Graves, D., Chen, D., Zhu, Z., Nguyen, N., Elsayed, M., Shao, K., Ahilan, S., Zhang, B., Wu, J., Fu, Z., Rezaee, K., Yadmellat, P., Rohani, M., Nieves, N. P., Ni, Y., Banijamali, S., Rivers, A. C., Tian, Z., Palenicek, D., bou Ammar, H., Zhang, H., Liu, W., Hao, J., and Wang, J. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving, 11 2020. URL https://arxiv.org/abs/2010.09776.

A. Implementation Details

A.1. Environments

MuJoCo For the MuJoCo framework, we incorporate nine version 2 (v2) datasets. These datasets are generated using three distinct behavior policies: '-medium', '-medium-play', and '-medium-expert', and span across three specific tasks: 'halfcheetah', 'hopper', and 'walker2d'.

AntMaze The AntMaze represents a set of intricate, long-horizon navigation challenges. This domain uses the same umaze, medium, and large mazes from the Maze2D domain, but replaces the agent with an 8-DoF Ant robot from the OpenAI Gym MuJoCo benchmark. For the 'umaze' dataset, trajectories are generated with the Ant robot starting and aiming for fixed locations. To introduce complexity, the "diverse" dataset is generated by selecting random goal locations within the maze, necessitating the Ant to navigate from various initial positions. Meanwhile, the "play" dataset is curated by setting specific, hand-selected initial and target positions, adding a layer of specificity to the task. We employ six version 2 (v2) datasets which include '-umaze', '-umaze-diverse', '-medium-play', '-medium-diverse', '-large-play', and '-large-diverse' in our experiments.

Franka Kitchen In the Franka Kitchen environment, the primary objective is to manipulate a set of distinct objects to achieve a predefined state configuration using a 9-DoF Franka robot. The environment offers multiple interactive entities, such as adjusting the kettle's position, actuating the light switch, and operating the microwave and cabinet doors, inclusive of a sliding mechanism for one of the doors. For the three principal tasks delineated, the ultimate objective comprises the sequential completion of four salient subtasks: (1) opening the microwave, (2) relocating the kettle, (3) toggling the light switch, and (4) initiating the sliding action of the cabinet door. In conjunction, three comprehensive datasets have been provisioned. The '-complete' dataset encompasses demonstrations where all four target subtasks are executed in a sequential manner. The '-partial' dataset features various tasks, but it distinctively includes sub-trajectories wherein the aforementioned four target subtasks are sequentially achieved. The '-mixed' dataset captures an assortment of subtask executions; however, it is noteworthy that the four target subtasks are not completed in an ordered sequence within this dataset. We utilize these datasets in our experiments.

A.2. Hyper-parameters and Implementations

Table 5. ADT Actor (Transformer) Hyper-parameters

	Hyper-parameter	Value
Architecture	Hidden layers	3
	Hidden dim	128
	Heads num	1
	Clip grad	0.25
	Embedding dim	128
	Embedding dropout	0.1
	Attention dropout	0.1
	Residual dropout	0.1
	Activation function	GeLU
	Sequence length	20 (V-ADT), 10 (G-ADT)
	G-ADT Way Step	20 (kitchen-partial, kitchen-mixed), 30 (Others)
Learning	Optimizer	AdamW
	Learning rate	1e-4
	Mini-batch size	256
	Discount factor	0.99
	Target update rate	0.005
	Value prompt scale	0.001 (Mujoco) 1.0 (Others)
	Warmup steps	10000
	Weight decay	0.0001
	Gradient Steps	100k (G-ADT, AntMaze), 1000k (Others)

We provide the lower-level actor's hyper-parameters used in our experiments in Table 5. Most hyper-parameters are set

following the default configurations in DT. For the inverse temperature used in calculating the AWR loss of the lower-level actor in V-ADT, we set it to 1.0, 3.0, 6.0, 6.0, 6.0, 15.0 for antmaze-'umaze', 'umaze-diverse', 'medium-diverse', 'medium-play', 'large-diverse', 'large-play' dataset, respectively; for other datasets, it is set 3.0. As for G-ADT, the inverse temperature is set to 1.0 for all the datasets. For the critic used in V-ADT and G-ADT, we follow the default architecture and learning settings in IQL (Kostrikov et al., 2022) and HIQL (Park et al., 2023), respectively.

B. IQL and HIQL

Implicit Q-learning (IQL) (Kostrikov et al., 2022) offers an approach to avoid out-of-sample action queries. This is achieved by transforming the traditional max operator in the Bellman optimality equation to an expectile regression framework. More formally, IQL constructs an action-value function $Q(s,a)$ and a corresponding state-value function $V(s)$. These are governed by the loss functions:

$$\mathcal{L}_V = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(\bar{Q}(s,a) - V(s)\right)\right], \tag{7}$$

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma V\left(s'\right) - Q(s,a)\right)^2\right], \tag{8}$$

Here, $\mathcal{D}$ represents the offline dataset, $\bar{Q}$ symbolizes the target Q network, and L_2^τ is defined as the expectile loss with a parameter constraint $\tau \in [0.5, 1)$ and is mathematically represented as $L_2^\tau(x) = |\tau - \mathbb{I}(x < 0)|x^2$. Then the policy is extracted with a simple advantage-weighted behavioral cloning procedure resembling supervised learning:

$$J_\pi = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\exp\left(\beta \cdot \tilde{A}\left(s,a\right)\right)\log\pi\left(a \mid s\right)\right], \tag{9}$$

where $\tilde{A}\left(s,a\right) = \bar{Q}(s,a) - V(s)$.

Building on this foundation, Hierarchical Implicit Q-Learning (Park et al., 2023) introduces an action-free variant of IQL that facilitates the learning of an offline goal-conditioned value function $V(s,g)$:

$$\mathcal{L}_V = \mathbb{E}_{(s,s')\sim\mathcal{D},g\sim p(g|\tau)}\left[L_2^\tau\left(r(s,g) + \gamma\bar{V}\left(s',g\right) - V(s,g)\right)\right] \tag{10}$$

where $\bar{V}$ denotes the target Q network. Then a high-level policy $\pi_h^h\left(s_{t+k} \mid s_t, g\right)$, which produces optimal k-steps jump, i.e., k-step subgoals s_{t+k}, is trained via:

$$J_{\pi^h} = \mathbb{E}_{(s_t,s_{t+k},g)}\left[\exp\left(\beta \cdot \tilde{A}^h\left(s_t,s_{t+k},g\right)\right)\log\pi^h\left(s_{t+k} \mid s_t,g\right)\right], \tag{11}$$

where β represents the inverse temperature hyper-parameter, and the value $\tilde{A}^h\left(s_t,s_{t+k},g\right)$ is approximated using $V\left(s_{t+k},g\right) - V\left(s_t,g\right)$. Similarly, a low-level policy is trained to learn to reach the sub-goal s_{t+k}:

$$J_{\pi^l} = \mathbb{E}_{(s_t,a_t,s_{t+1},s_{t+k})}\left[\exp\left(\beta \cdot \tilde{A}^l\left(s_t,a_t,s_{t+k}\right)\right)\log\pi^l\left(a_t \mid s_t,s_{t+k}\right)\right], \tag{12}$$

where the value $\tilde{A}^l\left(s_t,a_t,s_{t+k}\right)$ is approximated using $V\left(s_{t+1},s_{t+k}\right) - V\left(s_t,s_{t+k}\right)$.

For a comprehensive exploration of the methodology, readers are encouraged to consult the original paper.

C. Complete Experimental Results

Here we provide the learning curves of our methods on all selected datasets.

D. Visualization of decision-making process of G-ADT

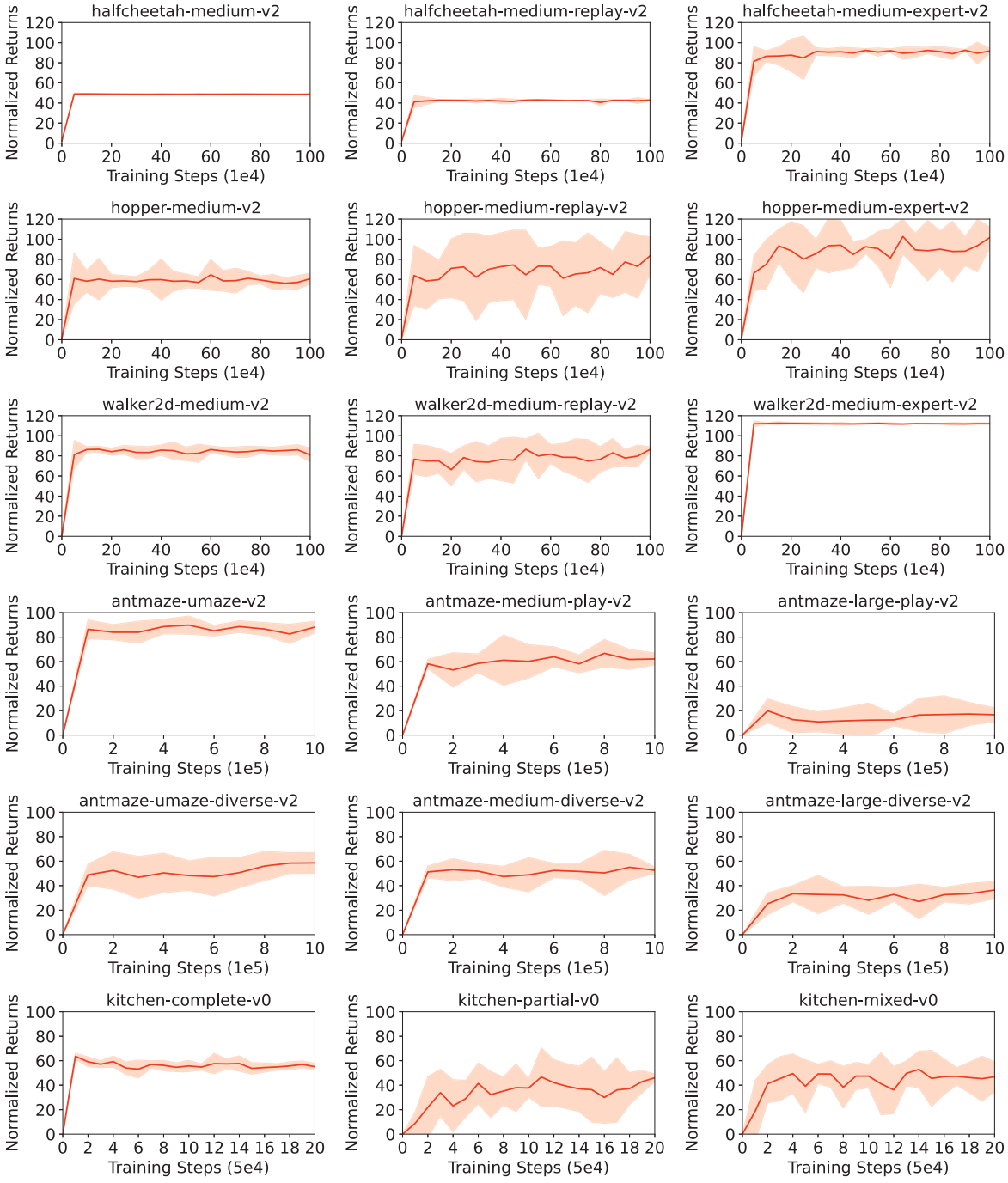

Figure 7. Learning curves of V-ADT.

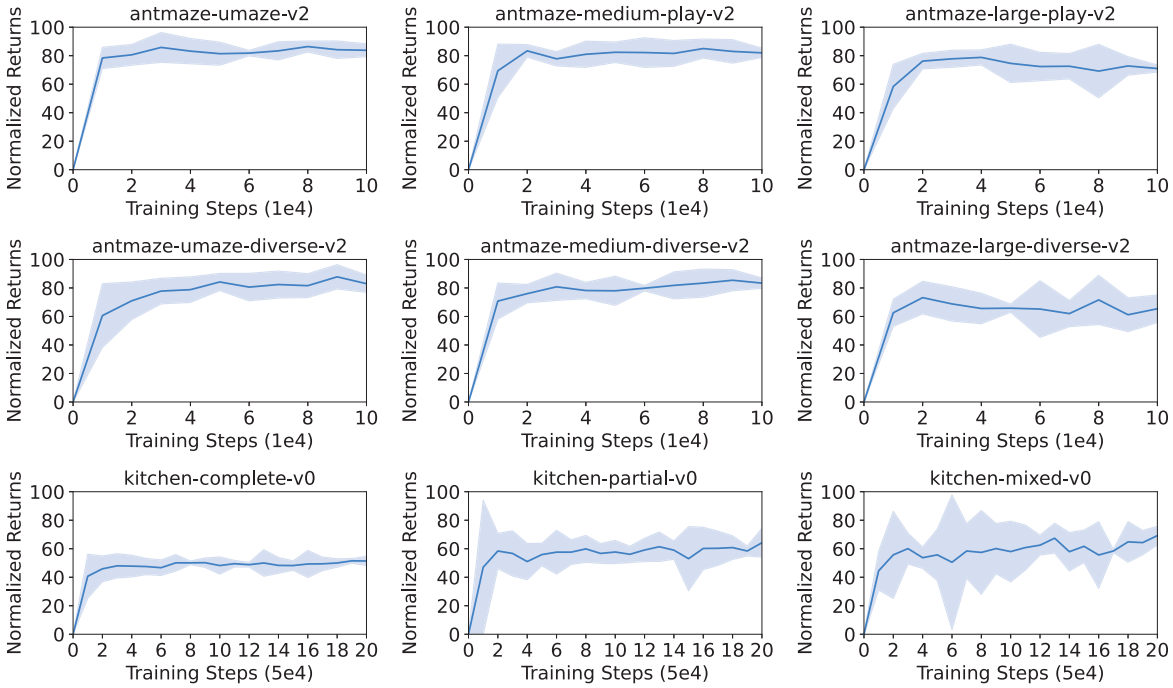

Figure 8. Learning curves of G-ADT.

Figure 9. Example of decision-making process of G-ADT in antmaze-large-play-v2 environments. We present some snapshots within an episode. The red circle represents the sub-goal given by the prompt policy. The pentagram indicates the target position to arrive.