ACT: Empowering Decision Transformer with Dynamic Programming via Advantage Conditioning

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Rui Kong, Zongzhang Zhang*, Yang Yu

National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
{gaocx, wucy, caomj, kongr}@lamda.nju.edu.cn, {zzzhang, yuy}@nju.edu.cn

Abstract

Decision Transformer (DT), which employs expressive sequence modeling techniques to perform action generation, has emerged as a promising approach to offline policy optimization. However, DT generates actions conditioned on a desired future return, which is known to bear some weaknesses such as the susceptibility to environmental stochasticity. To overcome DT's weaknesses, we propose to empower DT with dynamic programming. Our method comprises three steps. First, we employ in-sample value iteration to obtain approximated value functions, which involves dynamic programming over the MDP structure. Second, we evaluate action quality in context with estimated advantages. We introduce two types of advantage estimators, IAE and GAE, which are suitable for different tasks. Third, we train an Advantage-Conditioned Transformer (ACT) to generate actions conditioned on the estimated advantages. Finally, during testing, ACT generates actions conditioned on a desired advantage. Our evaluation results validate that, by leveraging the power of dynamic programming, ACT demonstrates effective trajectory stitching and robust action generation in spite of the environmental stochasticity, outperforming baseline methods across various benchmarks. Additionally, we conduct an in-depth analysis of ACT's various design choices through ablation studies. Our code is available at https://github.com/LAMDA-RL/ACT.

1 Introduction

Reinforcement Learning (RL) optimizes policies by interacting with the environment often for millions of steps (Haarnoja et al. 2018; Liu et al. 2018). Such enormous sample complexity prohibits RL from real-world applications (Wu and Zhang 2023) such as robotics and healthcare. As an alternative, offline RL optimizes policies with a pre-collected dataset (Levine et al. 2020) and has gained increasing attention in recent years for its potential in real-life scenarios (Kumar et al. 2021b; Shiranthika et al. 2022; Zhou, Zhang, and Yu 2023; Zhou et al. 2024).

Building upon online RL, a lot of algorithms address offline policy optimization following the spirit of dynamic programming (Levine et al. 2020), i.e. leveraging the structure of Markov Decision Process (MDP) and employing Bellman update to derive value estimates for subsequent policy optimization (Fujimoto, Meger, and Precup 2019; Kostrikov, Nair, and Levine 2022; Brandfonbrener et al. 2021).

In the meantime, the past few years have witnessed huge success in applying sequence modeling to natural language processing (Vaswani et al. 2017; Brown et al. 2020). In light of the similarity between language sequences and RL trajectories, a lot of works have explored the idea of modeling RL trajectories using sequence modeling approaches (Wen et al. 2023). For example, Decision Transformer (DT) (Chen et al. 2021) models offline trajectories extended with the sum of the future rewards along the trajectory, namely the return-to-go (RTG). RTG characterizes the quality of the subsequent trajectory, and DT learns to predict future actions given RTG. During testing, we deliberately provide DT with a high RTG to generate an above-average action. This approach, known as *conditional sequence generation*, has achieved remarkable success in offline policy optimization.

However, we note that RTG-conditioned generation is defective. Firstly, RTG is a hindsight indicator of a complete trajectory. It cannot be computed if we only have incomplete trajectory segments. Besides, we cannot ascertain the most suitable target RTG during testing, which poses a risk of choosing an inappropriate one and degenerating the performance. Secondly, conditioning on RTG fails to leverage the inherent structure of MDPs and cannot stitch trajectory as classical offline RL methods do. Therefore, DT cannot achieve much better performance than RL methods. Lastly, when the environment is stochastic, DT tends to exploit the occasional high return in the offline dataset mistakenly, which negatively impacts its performance.

In this paper, we present Advantage-Conditioned Transformer (ACT), a method that empowers the traditional DT with dynamic programming to effectively address the above-mentioned challenges. To tackle the limitations of conditioning on RTG, we introduce advantages as replacements. To obtain the advantage, we first approximate value functions with separate neural networks. Afterward, we introduce two types of advantage estimations, namely IAE and GAE, which exhibit different characteristics and are proven suited for different types of tasks empirically. We evaluate

*Zongzhang Zhang is the corresponding author.

ACT in various benchmarks including those with stochastic dynamics and delayed rewards. The results show that, with suitable advantage estimation, ACT significantly outperforms existing sequence modeling techniques and achieves performance on par with state-of-the-art offline RL methods. We also carried out extensive ablation studies to highlight the effectiveness of the design choices of ACT. Overall, our empirical results affirm the efficacy of the proposed method in addressing the limitations of DT and showcase its potential in applications.

2 Preliminaries

2.1 Offline Reinforcement Learning

A Markov Decision Process (MDP) can be denoted by a six-tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathcal{R}, \rho_0, \gamma \rangle$, where $\mathcal{S}$ stands for the state space, $\mathcal{A}$ is the action space, T is the transition function, $\mathcal{R}$ is the reward function, ρ_0 is the initial distribution of states, and $\gamma \in [0, 1)$ denotes the discount factor of future rewards. For states $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, $T(s'|s, a)$ specifies the transition probability of arriving at state s' after taking action a at state s, and $\mathcal{R}(s, a)$ gives the immediate reward of taking action a at state s. A policy π is a function mapping states to action distributions, and $\pi(a|s)$ is the probability of taking action a at state s. The policy's quality is measured by the expected discounted return $J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $r_t = \mathcal{R}(s_t, a_t)$ is the reward obtained at time t, s_t, a_t are the state and action at time t, respectively, and the expectation is taken w.r.t. the stochastic interaction of the policy π and the MDP environment. In offline RL, an agent is expected to learn a policy maximizing the expected discounted return with a static dataset $\mathcal{D}$. Typically, $\mathcal{D} = \{\tau_k\}_{k=1}^{K}$ is composed of K trajectories τ_k collected by a behavior policy β, where $\tau_k = \{s_t^k, a_t^k, r_t^k\}_{t=0}^{N_k-1}$, s_t^k, a_t^k, and r_t^k denote state, action, and reward at timestep t of the k-th trajectory, respectively, and N_k is the trajectory length of the k-th trajectory.

2.2 Dynamic Programming in RL

A commonly used categorization of deep RL algorithms is by their optimization paradigm, either *policy gradient* or *approximate dynamic programming* (Levine et al. 2020). The former derives the gradient of RL objectives w.r.t. the policy via the policy gradient theorem (Sutton et al. 1999) and improves the policy via gradient ascent. The latter derives the optimal policy exploiting the value function. The value functions of a policy π are defined recursively:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s,a)\right], \\ Q^\pi(s,a) &= \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim T(\cdot|s,a)}\left[V^\pi(s')\right]. \end{aligned}$$

Solving the value function with dynamic programming, we can derive an improved policy by increasing the probability of selecting actions with positive advantages, where the advantage of an action a at state s is $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. We can get the optimal policy by repeating this process, which is called policy iteration. Alternatively, we can use the value iteration and directly solve the optimal value functions V^* and Q^* with dynamic programming, where

$$\begin{aligned} V^*(s) &= \max_a Q^*(s,a), \\ Q^*(s,a) &= \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim T(\cdot|s,a)}\left[V^*(s')\right]. \end{aligned}$$

The optimal policy π^* is then obtained by greedily selecting the action a with the optimal value $Q^*(s, a)$ at all $s \in \mathcal{S}$.

Compared with policy gradient methods, dynamic programming leverages the structure of MDP, gaining advantages in efficiency and performance empirically.

2.3 Decision Transformer

One of the pioneering works of solving control tasks with high-capacity sequence modeling networks is the Decision Transformer (DT) (Chen et al. 2021). For each trajectory $\tau \in \mathcal{D}$, DT first computes the RTG $\hat{R}_t$ for each timestep t by summing up the rewards along the future trajectory, $\hat{R}_t = \sum_{t'=t}^{N-1} r_{t'}$, where N is the trajectory length. Later, DT fits a GPT-2 model (Radford et al. 2019) on the offline trajectories augmented with RTGs, $\tau^{\mathrm{RTG}} = \{\hat{R}_t, s_t, a_t\}_{t=0}^{N-1}$.

During testing, DT first specifies the desired target return as $\hat{R}_0$, and executes the predicted action $\hat{a}_0$ given the history $(\hat{R}_0, s_0)$. After observing the new state s_1 and reward r_0, DT sets $\hat{R}_1 = \hat{R}_0 - r_0$ and continues to predict the next action given the updated history $(\hat{R}_0, s_0, \hat{R}_1, s_1)$. This process continues until the end of the episode.

3 Defects with RTG Conditioning

Existing DT algorithms are commonly implemented by RTG conditioning. Although RTG conditioning is relatively simple to implement, this naive choice has several defects.

Dependency on future trajectory. The calculation of RTG involves summing up the rewards in the future trajectory. However, in real-world scenarios, the sampling process is often susceptible to unexpected events, making it challenging to gather complete and continuous trajectories. Under such circumstances, the collected offline dataset may consist of broken snippets of trajectories or even independent transition tuples (s, a, r, s'), making it difficult to calculate RTG since the future is missing. As illustrated in Figure 1, if we divide the trajectories into chunks, DT's performance will degenerate severely.

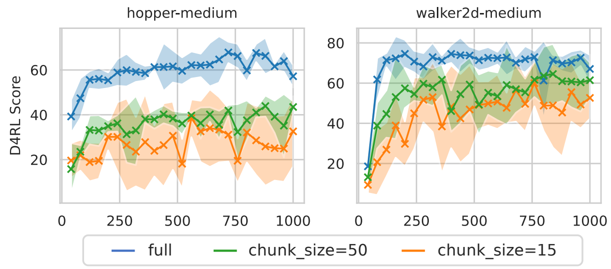

Figure 1: The performance of DT when the trajectory is divided into chunks. *Chunk_size* denotes the granularity of division, and *full* means the complete trajectory is preserved.

Inability to perform stitching. A well-established consensus on why offline RL methods outperform imitation learning is that they can perform *trajectory stitching* (Kumar et al.

2021a), i.e. concatenating multiple snippets of trajectories together to give better performance. However, as pointed out in recent works (Yamagata, Khalil, and Santos-Rodriguez 2023; Wu, Wang, and Hamaya 2023), DT lacks stitching ability, because the RTG is only associated with the current trajectory. Although conditioning on high RTGs might skew the generated action towards better ones, the performance of DT still falls behind traditional offline RL methods.

Failure in stochastic environments. The last failure mode of DT relates to its propensity to make optimistic decisions in stochastic settings, which arises from the intrinsic feature of RTG conditioning to imitate previous achievements indiscriminately while disregarding the randomness.

This phenomenon is also perceived as the conflation of the effects of the policy and the world model (Yang et al. 2023), meaning that RTG fails to distinguish the controllable parts (the policy) from the uncontrollable parts (the dynamics) and, as a result, biases the action generation with blind optimism about the dynamics.

4 Advantage-Conditioned Transformer

The previous discussion motivates us to devise a better alternative in place of RTG. Observing that dynamic programming handles these challenges effectively, we decide to exploit it for conditional generation.

Specifically, we approximate the value function via approximate dynamic programming (Section 4.1) and estimate advantages with the learned value functions. After labeling the dataset with advantages (Section 4.2), we fit an encoder-decoder transformer on the advantage-augmented dataset via the action reconstruction loss (Section 4.3). We additionally train a predictor c_ϕ to estimate the maximal advantage in samples and condition the action generation on the estimated maximum advantage during testing. The complete algorithm is presented in Algorithm 1.

4.1 Value Function Approximation

We use two parameterized functions Q_θ and V_ψ to approximate the state-action value function and the state value function, respectively. With the offline dataset, we iteratively update the parameters via stochastic gradient descent,

$$\begin{aligned} \theta &\leftarrow \theta - \frac{\eta}{M} \sum_{(s,a,r,s')\in\mathcal{B}} \nabla_\theta \left(r + \gamma V_{\bar{\psi}}(s') - Q_\theta(s,a)\right)^2, \\ \psi &\leftarrow \psi - \frac{\eta}{M} \sum_{(s,a,r,s')\in\mathcal{B}} \nabla_\psi \mathcal{L}_{\sigma_1}\left(r + \gamma V_{\bar{\psi}}(s') - V_\psi(s)\right), \end{aligned} \tag{1}$$

where $\mathcal{B} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^M$ denotes a mini-batch of transition tuples sampled uniformly from the dataset $\mathcal{D}$, M is the batch size, η is the learning rate, $V_{\bar{\psi}}$ is the target network, σ_1 is a hyper-parameter, and $\mathcal{L}_{\sigma_1}(u) = |\sigma_1 - \mathbb{I}(u < 0)|u^2$ is the σ_1-expectile regression loss.

Equation (1) performs in-sample value iteration as outlined by Kostrikov, Nair, and Levine (2022) which does not query out-of-distribution actions as classical value iteration does. When σ_1 equals 0.5, it is conducting on-policy value evaluation, producing an approximation of Q^β and V^β. With infinite samples and $\sigma_1 \to 1$, it produces an approximation of the in-sample optimal value functions:

$$\begin{aligned} V^*_\beta(s) &= \max\nolimits_{\beta(a|s)>0} Q^*_\beta(s,a), \\ Q^*_\beta(s,a) &= \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim T(\cdot|s,a)}\left[V^*_\beta(s')\right]. \end{aligned}$$

Thus, this learning procedure smoothly interpolates between approximating on-policy value functions and the in-sample optimal value functions by varying the value of σ_1.

Upon the convergence of the training, we freeze θ and ψ and do not make further updates to them.

4.2 Advantage Labeling

In the next step, we utilize the pre-trained value function approximation to tag the offline data with their advantage. The most straightforward advantage estimator $\hat{A}_{\text{IAE}}$ is simply the difference between Q_θ and V_ψ:

$$\hat{A}^{\text{IAE}}_t = Q_\theta(s_t, a_t) - V_\psi(s_t).$$

We term this *Immediate Advantage Estimation* (IAE). Since Q_θ averages all subsequent random variations, IAE produces an estimator robust to transitional stochasticity.

However, IAE issues a demand on estimating the state-action value function, which, compared to the state value function, has higher dimensional input and is more difficult to estimate. Besides, solely depending on Temporal Difference (TD) error to optimize the value functions is known to fail in sparse-reward or goal-oriented tasks (Yamagata, Khalil, and Santos-Rodriguez 2023; Hejna, Gao, and Sadigh 2023). Thus, we propose to use the *Generalized Advantage Estimation* (GAE) (Schulman et al. 2016) as an alternative:

$$\hat{A}^{\text{GAE}(\lambda)}_t = (1-\lambda)\sum_{l=1}^{\infty} \lambda^{l-1} \hat{A}^{(l)}_t,$$

where $\hat{A}^{(l)}_t = -V_\psi(s_t) + \sum_{i=0}^{l-1} \gamma^i r_{t+i} + \gamma^l V_\psi(s_{t+i+1})$ denotes the l-step advantage along the given trajectory. Note that GAE can be extended to the finite horizon case, though we present the infinite horizon formulation here for simplicity. The hyper-parameter $\lambda \in [0,1]$ interpolates between TD estimation and Monte-Carlo estimation, controlling the bias-variance trade-off. With λ set to 1, we have $\hat{A}^{\text{GAE}(1)}_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i} - V_\psi(s_t)$, which is functionally equivalent to the discounted RTG, since the constant $-V_\psi(s_t)$ does not count when comparing actions.

After deciding on the form of advantage estimator, we label the offline dataset with their advantages, and construct the advantage-augmented dataset $\mathcal{D}^{\text{A}} = \{\tau^{\text{A}}_k\}_{k=1}^M$, where $\tau^{\text{A}}_k = \{\hat{A}^k_t, s^k_t, a^k_t\}_{t=0}^{N_k-1}$, and $\hat{A}^k_t$ is the estimated advantage at the t-th timestep of the k-th trajectory by either IAE or GAE.

4.3 Advantage-Conditioned Sequence Modeling

The original DT straightforwardly applies the GPT-2 architecture to RL sequence modeling, offering convenience but

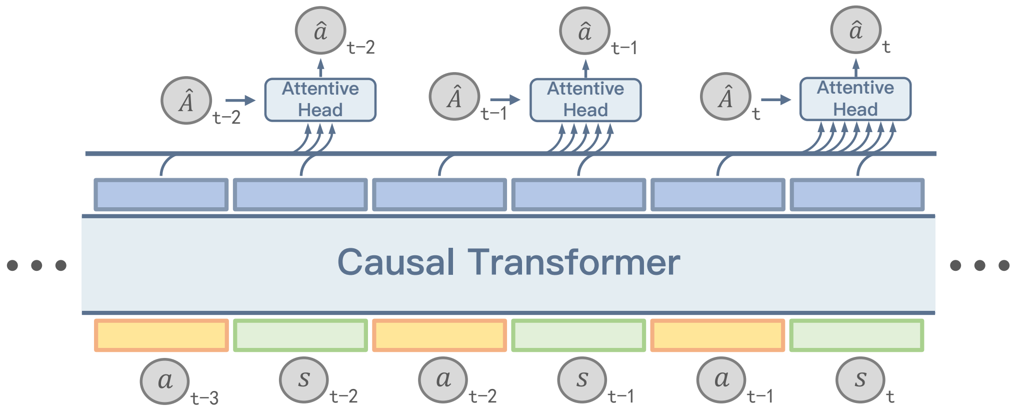

Figure 2: The encoder-decoder architecture of ACT. The encoder encodes the historical state-action sequence into a continuous representation. The attentive head of the decoder queries historical representation with the advantage and predicts an action.

falling short of optimization for decision-making tasks. In our experiments, we observed that modeling using GPT-2 architecture causes overfitting and unstable generation during testing, aligning well with recent research findings (Carroll et al. 2022). In light of this, we propose an encoder-decoder transformer architecture illustrated in Figure 2.

In our architecture, the encoder only accepts the state-action sequence as input and applies a causal attention mechanism (Vaswani et al. 2017) to mask out future inputs. It transforms the input sequence $(s_{<t}, a_{<t}, s_t)$ to a representation sequence, where $s_{<t}$ and $a_{<t}$ denote the states and actions before timestep t, respectively. With cross attention, the decoder queries historical representation with the estimated advantage $\hat{A}_t$ of the current timestep and predicts an action $\hat{a}_t$. The complete function is represented as $\hat{a}_t = \mathtt{ACT}(s_{<t}, a_{<t}, s_t, \hat{A}_t)$. We found that the separation of the state-action sequence and the advantage benefits the overall performance. Moreover, such separation leaves room for self-supervised pre-training. We may pre-train the encoder using a large unlabelled dataset by techniques similar to Masked Language Modeling (MLM), reinitialize the decoder, and fine-tune the transformer to accomplish control tasks (Sun et al. 2023; Carroll et al. 2022; Wu et al. 2023).

Another change we made is using the sinusoidal positional encoding in place of learnable positional embedding because we found the latter one sometimes causes instability, which echoes the finding by Zheng, Zhang, and Grover (2022). Other details are deferred to Appendix B.1[1] due to the limitation of space.

The loss for training ACT is the action reconstruction loss,

$$\mathcal{L}_{\text{ACT}} = \mathbb{E}_{\tau^{\text{A}} \sim \mathcal{D}^{\text{A}}} \left[\sum_t \left(a_t - \mathtt{ACT}(s_{<t}, a_{<t}, s_t, \hat{A}_t) \right)^2 \right], \tag{2}$$

where $\tau^{\text{A}} \sim \mathcal{D}^{\text{A}}$ means sampling τ^{A} uniformly from $\mathcal{D}^{\text{A}}$.

Finally, similar to DT, we need to specify the target advantage for each state during testing. We additionally train a predictor network c_ϕ with the expectile regression loss,

$$\mathcal{L}_\phi = \mathbb{E}_{(\hat{A}_t, s_t) \sim \mathcal{D}^{\text{A}}} \left[\mathcal{L}_{\sigma_2} \left(\hat{A}_t - c_\phi(s_t) \right) \right], \tag{3}$$

where $(\hat{A}_t, s_t) \sim \mathcal{D}^{\text{A}}$ denotes that $(\hat{A}_t, s_t)$ is sampled uniformly from $\mathcal{D}^{\text{A}}$. In the limit of $\sigma_2 \to 1$, c_ϕ learns to predict the maximal in-sample advantage given state s, which fully exploits ACT's potential to generate a good action.

4.4 Why Advantage Conditioning Offers Benefits?

Recall the Performance Difference Lemma (Kakade and Langford 2002), the performance difference between two policies π and π' can be expressed as the expected advantage w.r.t. π' on the state-action distribution induced by π, $\eta(\pi) - \eta(\pi') = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi'}(s_t, a_t) \right]$. Assume that the dataset is labeled using the advantage function of π' and our learners are exempt from sampling and approximation errors, then when conditioned on a desired advantage A, ACT will generate an action a satisfying $A^{\pi'}(s, a) = A$. By auto-regressively generating a trajectory and assigning ACT a positive target advantage at each timestep, ideally, all actions in the sequence possess positive advantages. Taking the expectation, we have $\mathbb{E}_{\tau \sim p_\pi}[\sum \gamma^t A_t^{\pi'}] > 0$ indicating that the induced ACT policy improves over the behavior policy π' by the lemma. As discussed previously, with $\sigma_1 = 0.5$, the learned value function approximates the value function of the behavior policy β, i.e., $\pi' = \beta$. If $\sigma > 0.5$, then the estimated advantage corresponds to a policy π' that is already improved over β.

Since RTGs are Monte-Carlo value estimates for the value of the behavior policy β, DT is restricted to improve on top of β, which limits the potential for improvement compared with ACT. Moreover, RTGs suffer from high variance, often assigning high values to actions due to occasional factors, while advantage offers a more robust action assessment by taking the expectation of the future.

[1]https://www.lamda.nju.edu.cn/gaocx/AAAI24-supp.pdf

Algorithm 1: Advantage-Conditioned Transformer

```
Input: Initialized value networks $Q_\theta, V_\psi$, predictor network $c_\phi$, transformer ACT, offline dataset $\mathcal{D}$
 1: // Training Phase
 2: for $k = 1, 2, \dots, K$ do
 3:     Update $Q_\theta$ and $V_\psi$ by Equation (1) on dataset $\mathcal{D}$
 4: end for
 5: Label the dataset $\mathcal{D}$ with estimated advantages to get the advantage-augmented dataset $\mathcal{D}^{\mathrm{A}}$
 6: Learn ACT by minimizing Equation (2) via stochastic gradient descent on dataset $\mathcal{D}^{\mathrm{A}}$
 7: Learn $c_\phi$ by minimizing Equation (3) via stochastic gradient descent on dataset $\mathcal{D}^{\mathrm{A}}$
 8:
 9: // Test Phase
10: Initialize history $h = \emptyset$
11: while episode not ended at timestep $t$ do
12:     Let $A_t = c_\phi(s_t)$
13:     Execute $a_t = \mathtt{ACT}(h, s_t, A_t)$
14:     Update $h = h \cup \{s_t, a_t\}$
15: end while
```

5 Related Work

Dealing with stochastic environments for DT. Numerous related works are dedicated to addressing the vulnerability of DT when confronted with stochastic environments. ESPER (Paster, McIlraith, and Ba 2022) uses adversarial clustering to learn trajectory representations disentangled from environmental stochasticity. DoC (Yang et al. 2023) attributes such failure to the fact that generative models make no distinction between the parts it can control (agent actions) and those it cannot (environmental transition). Thus, DoC proposes to extract predictive representations for trajectories, while minimizing the mutual information between the representation and the environment transition. A similar idea is also explored in Villaflor et al. (2022), where the authors explicitly model the policy and the world model with two separate transformers. Brandfonbrener et al. (2022) provided a theoretical analysis of the return-conditioning regime, where they found the near-determinism of the environment is one of the conditions that return-conditioning can find the optimal policy.

Combining DP and DT. Most of the offline RL methods employ dynamic programming to optimize policies, such as CQL (Kumar et al. 2020), IQL (Kostrikov, Nair, and Levine 2022), and TD3+BC (Fujimoto and Gu 2021). As discussed in this paper as well as in Brandfonbrener et al. (2022), DP and DT bear distinct characteristics, for example, DP has no preference for the determinism of the environment, while DT may be more applicable in long-horizon tasks. Several works have tried to combine DP and DT. Among them, QDT (Yamagata, Khalil, and Santos-Rodriguez 2023) precomputes conservative Q-values and V-values using CQL, and labels the offline data with the maximum of RTG and the conservative V-value. Our method differs from QDT in that, we compute the value estimations via an in-sample value iteration, which prevents instability and inaccuracy caused by bootstrapping from out-of-dataset actions. Moreover, the flexible advantage estimation also enables ACT to combine the best of both worlds. Recently, EDT (Wu, Wang, and Hamaya 2023) was proposed to vary the context length of DT during testing. When provided a shorter context, EDT can recover from bad history and switch to a higher rewarding action; provided a long context, EDT can stably recall the subsequent decisions. In this way, EDT interpolates between trajectory stitching and behavior cloning from a new perspective. This work is complementary to ACT, and they can be seamlessly integrated to achieve better performance.

6 Experimental Evaluations

Our assessments are designed to comprehensively analyze the efficacy of ACT when presented with diverse benchmarks and tasks encompassing a spectrum of characteristics and challenges, including deterministic, stochastic, and delayed reward tasks. We also conduct an ablation study on the use of network c_ϕ and the choice of transformer architecture.

Deterministic Gym MuJoCo tasks. We investigate the performance of ACT in the most widely studied Gym MuJoCo tasks. We focus on three domains, namely *halfcheetah (hc)*, *hopper (hp)*, and *walker2d (wk)*, which are deterministic both in state transition and reward functions. We leverage the *v2* datasets provided by D4RL (Fu et al. 2020), which includes three levels of quality: *medium (med)*, *medium-replay (med-rep)*, and *medium-expert (med-exp)*.

For this benchmark, we choose GAE(0) as the advantage estimator across all tasks to maximize the utility of dynamic programming. For σ_1, we sweep its value through $[0.5, 0.7]$. In order to mitigate the instability of TD learning (Li et al. 2023), we conduct model selection by choosing a value model that has the lowest training error on the offline dataset. See Appendix B.2 for details.

We contrast the performance of ACT against two groups of algorithms. The first group, including CQL (Kumar et al. 2020), IQL (Kostrikov, Nair, and Levine 2022), Onestep-RL (Brandfonbrener et al. 2021), 10%-BC, and RvS (Emmons et al. 2022), optimizes a Markovian policy that depends only on states. The second group, including DT (Chen et al. 2021), QDT (Yamagata, Khalil, and Santos-Rodriguez 2023), and EDT (Wu, Wang, and Hamaya 2023), models the RL trajectories with sequence-modeling methods and learns a non-Markovian policy that depends on history.

The results are listed in Table 1. We find that ACT achieves the highest score in 8 out of 9 tasks compared to other methods that learn non-Markovian policies, including QDT and EDT which share a similar motivation with us, i.e., improving DT with the ability of trajectory stitching. The improvement over these methods underscores the efficacy of advantage conditioning. If we extend the comparison to all methods, ACT still manages to outperform in 6 out of 9 tasks. This demonstrates that ACT not only exploits the power of DP but also gains benefits from the sequence modeling tech-

Dataset		Markovian					Non-Markovian			
		CQL	Onestep RL	IQL	10% BC	RvS	DT	QDT	EDT	ACT
hc	med	44.0	48.4	47.4	42.5	41.6	42.6	42.4 ± 0.5	42.5 ± 0.9	**49.1 ± 0.2**
hp		58.5	59.6	66.2	56.9	60.2	62.3	60.7 ± 5.0	63.5 ± 5.8	**67.8 ± 5.5**
wk		72.5	81.8	78.3	75.0	71.7	74.3	63.7 ± 6.4	72.8 ± 6.2	**80.9 ± 0.4**
hc	med-rep	45.5	38.1	44.2	40.6	38.0	36.9	32.8 ± 7.3	37.8 ± 1.5	**43.0 ± 0.4**
hp		60.9	97.5	94.7	75.9	73.5	75.8	38.7 ± 26.7	89.0 ± 8.3	**98.4 ± 2.4**
wk		77.2	49.5	73.8	62.5	60.6	59.4	29.6 ± 15.5	**74.8 ± 4.9**	56.1 ± 10.9
hc	med-exp	91.6	93.4	86.7	92.9	92.2	86.3	–	–	**96.1 ± 1.4**
hp		105.4	103.3	91.5	110.9	101.7	104.2	–	–	**111.5 ± 1.6**
wk		108.8	113.0	109.6	109.0	106.0	107.7	–	–	**113.3 ± 0.4**

Table 1: Normalized score on deterministic Gym MuJoCo tasks, with datasets from D4RL. The performances of QDT and EDT are taken from their original papers, and the numbers for other baselines are taken from (Garg et al. 2023). For ACT and DT, we use our own implementations and report the average performance and the standard deviation of the final checkpoint across 10 evaluation episodes and 5 seeds. We bold the highest score among sequence-modeling methods, and add background shading to the highest score among all methods.

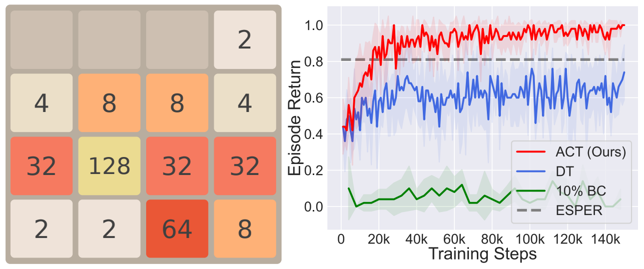

Figure 3: Performance on the 2048 game. We report the average and std of the performance across 5 independent runs and mark ESPER's score as the grey dotted line.

nique, thus combining the best of both worlds.

Stochastic benchmarks. Another important property of ACT is its robustness against the stochasticity of the environments. To validate this, we choose to assess ACT, DT, QDT, ESPER (Paster, McIlraith, and Ba 2022), and 10%-BC on stochastic benchmarks. Among them, ESPER is an algorithm that enhances DT to cope with environmental stochasticity. In the following evaluations, we select IAE as the advantage estimator, and σ_1 is kept to 0.5 by default.

As a sanity check, we first evaluate the algorithms on the 2048 game. This environment is stochastic in that, after each action which moves all the tiles along some direction and merges them if their numbers are equal, a random 2 or 4 will be placed in a random empty grid. The agent is given a reward of 1 once it manages to produce a tile of 128, and the episode will be ended. Thus, the maximum return in this environment is 1. Further introduction about this environment and the dataset can be found in Appendix B.3. Figure 3 depicts the performance curve as the training proceeds. While DT and 10%-BC exploit the occasional success in the dataset and converge to suboptimal policies, ACT and ESPER show robustness to stochasticity. Besides, ACT outperforms ESPER and converges to 100% success rate to produce 128, validating the power of advantage conditioning.

We also created a more sophisticated benchmark, by reusing the Gym MuJoCo tasks and following Yang et al. (2023) to add noise to the agent's action before passing it to the simulator. More details can be found in Appendix B.4. The results are illustrated in Figure 4. As expected, we observe that 10%-BC severely overfits the high-return trajectories in the dataset, albeit with a sharp drop in test scores as the training proceeds. DT and QDT yield a comparatively robust policy, while their performances are still inferior to ACT. This further justifies the validity of ACT in stochastic control tasks.

Dataset	CQL	IQL	DT	ACT
hp-med	$46.1_{\pm 2.9}$	$32.4_{\pm 7.7}$	$\mathbf{65.7_{\pm 1.4}}$	$49.4_{\pm 1.5}$
hp-med-exp	$0.8_{\pm 0.3}$	$97.6_{\pm 14.8}$	$\mathbf{106.6_{\pm 3.7}}$	$95.4_{\pm 13.9}$
wk-med	$-0.3_{\pm 0.1}$	$53.2_{\pm 6.7}$	$72.1_{\pm 4.3}$	$\mathbf{73.8_{\pm 1.9}}$
wk-med-exp	$7.0_{\pm 6.4}$	$67.4_{\pm 22.1}$	$107.4_{\pm 0.3}$	$\mathbf{108.1_{\pm 0.2}}$
Average	13.4	62.7	**88.0**	81.7

Table 2: Normalized score on delayed reward tasks. The average and std are taken across 4 independent runs.

Delayed reward tasks. As unveiled in previous literature (Chen et al. 2021; Yamagata, Khalil, and Santos-Rodriguez 2023), DT has an advantage over conventional RL methods when the reward is sparse and thus long-term credit assignment is required. In the following part, we aim to investigate whether ACT still retains such ability. We once again revise the D4RL datasets, deferring the rewards for each trajectory until the final timestep. We exclude the *hc* and *med-rep* datasets as they contain time-out trajectories which are not applicable for delayed-reward settings. For the advantage estimator, we choose GAE(1) for its resemblance

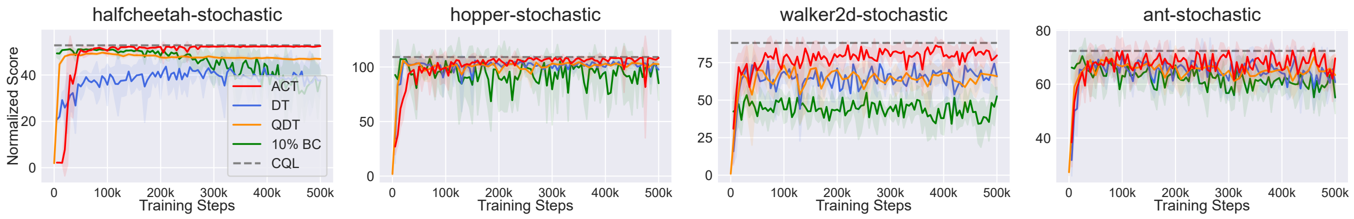

Figure 4: Performance curve on the stochastic Gym MuJoCo tasks as the training proceeds. We report the average and the standard deviation of the performance across 4 independent runs.

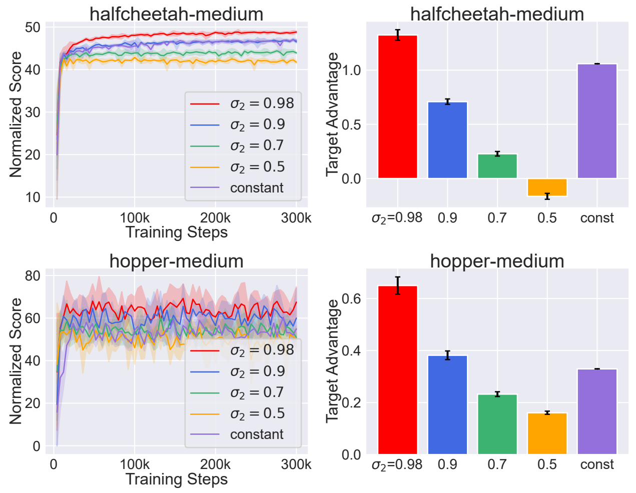

Figure 5: Ablation study on the effect of σ_2, using datasets from D4RL. The left column depicts the performance of each variant as the training proceeds, and the right column depicts the target advantages given by c_ϕ. The results are taken from 4 independent runs.

to the RTGs. The results are listed in Table 2. While DP-based methods like CQL and IQL suffer from degeneration, DT upholds its performance and is minimally affected by reward sparsity. Although ACT is not entirely spared from the influence, it still preserves benefits from the sequence modeling architecture and outperforms CQL and IQL.

Ablation study. Finally, we investigate the effects of the design choices of ACT through ablation studies. Our first ablation focuses on the effect of σ_2, which determines the expectile to approximate by c_ϕ. In the above experiments, we set the σ_2 to 0.98, and in this ablation study, we additionally test with 0.5, 0.7, and 0.9 to analyze the actual effect of this parameter. According to the discussions in Section 4.4, providing ACT with a positive advantage already induces policy improvement. Therefore we also include a variant of ACT which uses a constant positive value as the target advantage for all states. The positive value is set as the average of the positive advantages in the offline dataset. The results illustrated in Figure 5 suggest a close relationship between the value of σ_2 and the final performance. When σ_2 is set to higher values, the predictor network c_ϕ tends to give higher desired advantages $\hat{A}$ during test time, and ACT would retrieve better actions in response. Giving a positive constant value also brings about improvement in certain environments such as *hc-med*, but in *hp-med* it performs substantially worse than using a predictor network. Comparisons on more datasets are deferred to Appendix C.2.

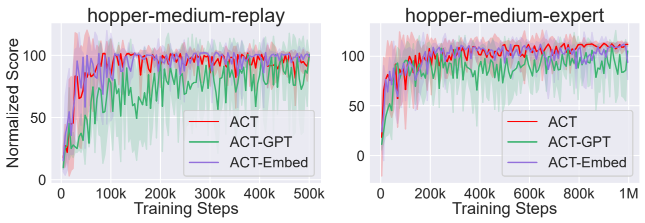

Figure 6: Ablation study on the transformer architecture, using datasets from D4RL with 4 independent runs.

We also carried out an analysis of the choice of transformer architecture. ACT differs from DT in two ways: 1) ACT employs an encoder-decoder structure while DT is a decoder-only transformer, and 2) ACT uses sinusoidal encoding rather than learnable embedding to prevent overfitting. Thus, we additionally implement two variants of ACT, one with GPT-2 architecture (`ACT-GPT`) and another using learnable positional embedding (`ACT-Embed`), and keep other hyper-parameters the same with ACT. In Figure 6, we select two datasets of the hopper task for comparison and observe that `ACT-GPT` has undergone significant oscillation and degradation in its performance, implying that the direct application of GPT-2 structure to our algorithm is not effective. The other variant, `ACT-Embed`, suffers from instabilities in tasks like *hp-med-exp* in the late training, although the overall gap is minor.

7 Conclusion and Future Work

In this paper, we introduce ACT, which enhances popular sequence modeling techniques with dynamic programming to address their limitations in offline policy optimization. We achieve this by conditioning the transformer architecture on the estimated advantages. By selectively choosing the type of advantage estimator, our framework can be applied to a variety of tasks, demonstrating consistent performance improvements and generalizability. Our framework also leaves room for the incorporation of self-supervised learning techniques and has the potential to be extended to multi-task settings and to a larger training scale. We will continue this line of research and investigate these topics in the future.

Acknowledgements

This work is supported by the National Science Foundation of China (62276126, 62250069), the Natural Science Foundation of Jiangsu (BK20221442), and the Fundamental Research Funds for the Central Universities (14380010).

References

Brandfonbrener, D.; Bietti, A.; Buckman, J.; Laroche, R.; and Bruna, J. 2022. When Does Return-conditioned Supervised Learning Work for Offline Reinforcement Learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 1542–1553.

Brandfonbrener, D.; Whitney, W.; Ranganath, R.; and Bruna, J. 2021. Offline RL Without Off-Policy Evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 4933–4946.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Carroll, M.; Paradise, O.; Lin, J.; Georgescu, R.; Sun, M.; Bignell, D.; Milani, S.; Hofmann, K.; Hausknecht, M. J.; Dragan, A. D.; and Devlin, S. 2022. Uni[MASK]: Unified Inference in Sequential Decision Problems. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 15084–15097.

Emmons, S.; Eysenbach, B.; Kostrikov, I.; and Levine, S. 2022. RvS: What is Essential for Offline RL via Supervised Learning? In *International Conference on Learning Representations (ICLR)*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *arXiv preprint arXiv:2004.07219*.

Fujimoto, S.; and Gu, S. S. 2021. A Minimalist Approach to Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 20132–20145.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *International Conference on Machine Learning (ICML)*, 2052–2062.

Garg, D.; Hejna, J.; Geist, M.; and Ermon, S. 2023. Extreme Q-Learning: MaxEnt RL without Entropy. In *International Conference on Learning Representations (ICLR)*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning (ICML)*, 1856–1865.

Hejna, J.; Gao, J.; and Sadigh, D. 2023. Distance Weighted Supervised Learning for Offline Interaction Data. *arXiv preprint arXiv:2304.13774*.

Kakade, S.; and Langford, J. 2002. Approximately Optimal Approximate Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 267–274.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations (ICLR)*.

Kumar, A.; Hong, J.; Singh, A.; and Levine, S. 2021a. Should I Run Offline Reinforcement Learning or Behavioral Cloning? In *International Conference on Learning Representations (ICLR)*.

Kumar, A.; Singh, A.; Tian, S.; Finn, C.; and Levine, S. 2021b. A Workflow for Offline Model-Free Robotic Reinforcement Learning. In *Conference on Robot Learning (CORL)*, 417–428.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1179–1191.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643*.

Li, Q.; Kumar, A.; Kostrikov, I.; and Levine, S. 2023. Efficient Deep Reinforcement Learning Requires Regulating Overfitting. In *International Conference on Learning Representations (ICLR)*.

Liu, Q.; Zhai, J.; Zhang, Z.; Zhong, S.; Zhou, Q.; Zhang, P.; and Xu, J. 2018. A Survey on Deep Reinforcement Learning. *Chinese Journal of Computers*, 41(1): 1–27.

Paster, K.; McIlraith, S.; and Ba, J. 2022. You Can't Count on Luck: Why Decision Transformers and RvS Fail in Stochastic Environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 38966–38979.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. In *OpenAI Blog*.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations (ICLR)*.

Shiranthika, C.; Chen, K.; Wang, C.; Yang, C.; Sudantha, B. H.; and Li, W. 2022. Supervised Optimal Chemotherapy Regimen Based on Offline Reinforcement Learning. *IEEE Journal of Biomedical and Health Informatics*, 26(9): 4763–4772.

Sun, Y.; Ma, S.; Madaan, R.; Bonatti, R.; Huang, F.; and Kapoor, A. 2023. SMART: Self-supervised Multi-task pretrAining with contRol Transformers. In *International Conference on Learning Representations (ICLR)*.

Sutton, R. S.; McAllester, D. A.; Singh, S.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 1057–1063.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 5998–6008.

Villaflor, A. R.; Huang, Z.; Pande, S.; Dolan, J. M.; and Schneider, J. 2022. Addressing Optimism Bias in Sequence Modeling for Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 22270–22283.

Wen, M.; Lin, R.; Wang, H.; Yang, Y.; Wen, Y.; Mai, L.; Wang, J.; Zhang, H.; and Zhang, W. 2023. Large Sequence Models for Sequential Decision-Making: A Survey. *Frontiers of Computer Science*, 17(6): Article Number 176349.

Wu, C.; and Zhang, Z. 2023. Surfing Information: The Challenge of Intelligent Decision-Making. *Intelligent Computing*, 2: Article 0041.

Wu, P.; Majumdar, A.; Stone, K.; Lin, Y.; Mordatch, I.; Abbeel, P.; and Rajeswaran, A. 2023. Masked Trajectory Models for Prediction, Representation, and Control. In *International Conference on Machine Learning (ICML)*, 37607–37623.

Wu, Y.; Wang, X.; and Hamaya, M. 2023. Elastic Decision Transformer. *arXiv preprint arXiv:2307.02484*.

Yamagata, T.; Khalil, A.; and Santos-Rodriguez, R. 2023. Q-learning Decision Transformer: Leveraging Dynamic Programming for Conditional Sequence Modelling in Offline RL. In *International Conference on Machine Learning (ICML)*, 38989–39007.

Yang, S.; Schuurmans, D.; Abbeel, P.; and Nachum, O. 2023. Dichotomy of Control: Separating What You Can Control from What You Cannot. In *International Conference on Learning Representations (ICLR)*.

Zheng, Q.; Zhang, A.; and Grover, A. 2022. Online Decision Transformer. In *International Conference on Machine Learning (ICML)*, 27042–27059.

Zhou, R.; Gao, C.; Zhang, Z.; and Yu, Y. 2024. Generalizable Task Representation Learning for Offline Meta-Reinforcement Learning with Data Limitations. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Zhou, R.; Zhang, Z.; and Yu, Y. 2023. Model-based Offline Weighted Policy Optimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 16392–16393.