2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Steering Decision Transformers via Temporal Difference Learning

Hao-Lun Hsu, Alper Kamil Bozkurt*, Juncheng Dong*, Qitong Gao, Vahid Tarokh and Miroslav Pajic

***Abstract*— Decision Transformers (DTs) have been highly effective for offline reinforcement learning (RL) tasks, successfully modeling the sequences of actions in a given set of demonstrations. However, DTs may perform poorly in stochastic environments, which are prevalent in robotics scenarios. In this paper, we identify that the root cause of this performance degradation is the growing variance of returns-to-go, the signal used by DTs to predict actions, accumulated over the horizon. Building upon this insight, we propose an extension to DTs that allows them to be steered toward high-reward regions, where the expected returns are estimated using temporal difference learning. This way, we not only mitigate the growing variance problem but also eliminate the need for DTs to have access to returns-to-go during evaluation and deployment phases. We show that our method outperforms state-of-the-art offline RL methods in both simulated and real-world robotic arm environments.**

I. Introduction

The true potential of reinforcement learning (RL) is hindered in numerous real-world robotics application domains by the costly trial-and-error approach of online learning [1], [2], where learning from scratch can be both expensive and hazardous [3], [4]. Offline RL [5] can unlock the full potential of RL by eliminating the need for active interactions with an online environment by enabling the use of offline datasets that are often available across a wide range of diverse domains and settings [6], [7]. Consequently, offline RL has attracted significant attention, aiming to learn effective and generalizable policies from offline datasets (e.g., [6], [8], [9]).

Decision Transformers (DTs) [10], recently introduced as part of a more general framework called RL via supervised learning (RvS) [11]–[13], have shown promising performance in offline RL tasks, outperforming conventional RL methods such as Temporal-Difference (TD) learning [14]. DTs recast RL problems into sequence prediction problems where the actions are predicted based on the past trajectory and the cumulative reward to be attained in the future, called returns-to-go (RTG). Although the DT approach has led to great empirical success [10], it has been shown that DTs yield poor performance in environments exhibiting highly stochastic behaviors [15], [16] and have poor stitching ability [17], [18]. The poor performance of DTs in stochastic environments is the primary bottleneck [15], significantly restricting their application in many robotics environments (e.g., [17], [19]).

In this work, we first analyze why the performance of DTs in stochastic environments is poor while being outstanding in near-deterministic environments. Our analysis reveals that the growing variance of RTG accumulating over the horizon is the main cause of the performance degradation in stochastic environments. To demonstrate the correctness of our analysis, we empirically show that replacing RTG with an approximate optimal value function yields higher performance than the original DT-based approach.

Motivated by the insights gained from this analysis, we introduce Steering *Decision Transformers via Temporal Difference* learning (D2T2), which integrates DTs with TD learning. D2T2 utilizes the power of TD learning to overcome the variance problem, reminiscent of the highly effective collaboration between policy optimization and TD learning in Actor-Critic (AC) algorithms [20]. Specifically, D2T2 maps the current state into a guiding vector that steers DTs toward high-reward regions where the expected returns are approximated by TD learning. Through TD learning, D2T2 addresses the variance problem of the RTG and demonstrates significantly improved performance in stochastic tasks compared to DTs. To further tackle the issue of growing variance of RTG in stochastic environments, our approach transforms long-horizon tasks into shorter ones. With the transformed learning problem, D2T2 further improves the performance of DTs. Additionally, our approach eliminates the need to manually craft RTG, a prominent challenge DTs face during evaluation and deployment.

We benchmarked our approach on two illustrative stochastic tasks (FrozenLake and Tailgate driving [16]), two stochastic CARLA benchmarks [21], and three suites (18 tasks) from D4RL (Gym-MuJoCo, AntMaze, and FrankaKitchen) [6] as well as a real-world robotic arm manipulation environment (Fig. 1). These environments with tasks of different levels of difficulty enabled us to comprehensively investigate the capabilities of our approach. We showed that our approach outperforms the state-of-the-art baselines, including both return-conditioned and goal-conditioned methods, in almost all stochastic environments including real-world robotic manipulation environments.

II. Decision Transformers and Stochasticity

A. Problem Setup

Sequential decision-making problems can be formulated as Markov Decision Processes (MDPs), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, T)$, where $\mathcal{S}$ and $\mathcal{A}$ are the set of states and actions, respectively; $\mathcal{R}$ is the reward function where $r_t = \mathcal{R}(s_t, a_t)$ is the received reward at time step t for taking action a_t in state s_t; $P(s'|s, a)$ is the transition probability

*equal contribution with alphabetical order.

This work is sponsored in part by the AFOSR award FA9550-19-1-0169, National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant CNS-2112562, and the NSF NAIAD 2332744 award.

Hao-Lun Hsu and Alper Kamil Bozkurt are with the Department of Computer Science and the other authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA; Emails: {hao-lun.hsu, alper.bozkurt, juncheng.dong, qitong.gao, vahid.tarokh, miroslav.pajic}@duke.edu.

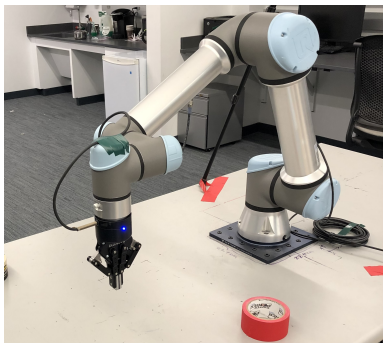

Fig. 1. A UR5e trying to push a red tape to a target position.

for $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$; and T is the horizon. We consider finite-horizon MDPs, without discounting, with the of learning an optimal policy π^* maximizing the expected total return, *i.e.*,

$$\pi^* \in \underset{\pi}{\operatorname{argmax}} \left\{ \mathbb{E}_{\rho_\pi} \left[\sum\nolimits_{t=0}^{T} r_t \right] \right\}, \tag{1}$$

where ρ_π is the state-action distribution under policy π. In the offline setting, the optimal policy π^* is learned from a fixed set of trajectories $\mathcal{D}$ storing the transitions (s, a, r, s'). Here, $\mathcal{D}$ is usually collected over some behavior policy π_b, which can be either a single policy or a mixture of policies that are considered unknown.

B. Decision Transformers in Deterministic Environments

At each time step t, a DT predicts an action a_t after taking as input the observed trajectory and the RTG signal $R_t = \sum_{k=t}^{T} r_k$, i.e., the cumulative return obtained from time step t to the end of the trajectory. We note that, at time step t, R_t is not available during evaluation and deployment phases; however, for an offline trajectory from $\mathcal{D}$, R_t can be computed from all the subsequent rewards received after time step t. This way, DTs aim to solve RL problems via supervised learning, with the following objective.

Problem 1. *Train a DT that predicts the action a_t that is most likely to generate a given RTG R_t based on a given trajectory observed until time step t. In other words, let τ_t denote the observed trajectory until time step t and let $A_{t+1:T} \coloneqq \{a_{t+1}, \ldots, a_T\}$ denote the set of future actions after time step t; then, the objective is to accurately approximate the following action prediction function:*

$$f^*(\tau_t, t, R_t) \coloneqq \underset{a}{\operatorname{argmax}} \left\{ \max_{A_{t+1:T}} \mathbb{P}\left(\sum_{k=t}^{T} r_k = R_t \,\middle|\, \tau_t, a_t, A_{t+1:T} \right) \right\}. \tag{2}$$

Once trained, a DT can be used to predict actions to obtain a desired return R^*, by setting the RTG signals as $R_t = R^* - \sum_{i=0}^{t-1} r_i$ where $\sum_{i=0}^{-1} r_i \coloneqq 0$. The idea here, starting with the desired RTG $R_0 = R^*$, is to decrease the desired RTG by the observed reward r_t after taking the action a_t predicted by the DT at each time step t. If the maximum possible return that can be obtained can be specified as the desired reward R^*, then the DT should predict a sequence of actions that maximizes the probability of obtaining the maximum return.

We next show that a DT, perfectly approximating f^* from (2), is indeed guaranteed to return the optimal sequence of actions that result in the maximum cumulative reward in deterministic environments. This facilitates understanding why the performance of DTs degrades in stochastic environments.

Theorem 1 ('Standard' RTGs lead to optimal trajectories in deterministic environments)**.** *For a given deterministic environment, a finite horizon T and an initial state s_0, let $\{s_0^*(=s_0), a_0^*, r_0^*, \ldots, s_T^*, a_T^*, r_T^*\}$ denote an optimal trajectory, assumed to be unique for simplicity; and let $R_t^* \coloneqq \sum_{t'=t}^{T} r_{t'}^*$ denote the optimal return, the maximum future cumulative reward achievable from time step t onward. Then, for any $s_0 \in S$, it holds that:*

1) *At $t = 0$, a DT, perfectly approximating f^* from (2), will predict a_0^* if the input $R_t = R_0^*$. If the optimal trajectory is not unique, then the predicted a_0^* must be the first action of some optimal trajectory.*
2) *At $t > 0$, if the input to the DT is R_t^*, then the output action must be a_t^*.*
3) *At $t > 0$, the recursively computed signal $R_t = R_{t-1} - r_t$ must equal to R_t^*.*

Proof. Due to the space constraint, the proof is provided in [22]. □

Theorem 1 shows that DTs output the optimal trajectories in deterministic environments and *the success of DTs relies on the successful estimation of R_t^**. In deterministic environments, R_t^* can be estimated if the initial signal is correct. However, in stochastic environments, the variance of R_t becomes larger as the number of steps increases. This results in a signal with growing variance over the horizon, as we now discuss.

C. Improving Returns-to-Go Signals for Decision Transformers in Stochastic Environments

An intuitive solution to reduce the RTG variance is to use the Markov property of the environment and build an RTG signal that only uses s_t at time step t. This solution shares the same motivation and theoretical support as the use of TD learning to approximate the cumulative future rewards in AC [20], where the variance of the Monte-Carlo estimates of cumulative rewards causes unstable gradient estimation. Hence, an ideal RTG signal should provide the maximum amount of future cumulative reward that any agent can collect, based solely on the information in the current state s_t. This signal can be captured by the optimal value function, *i.e.,* the value function of an optimal policy.

Consequently, in this work, we introduce a modified version of DTs where the RTG is replaced with the output of the value function from a pre-trained Q-learning model; we show that such architecture successfully outperforms the original DT in stochastic tasks (see Section IV for details). This demonstrates that TD learning can address the problem of growing variance of the input signal to DTs in stochastic environments.

While TD learning can address the challenge of growing variance of the RTG signal, the fundamental prediction problem faced by DTs remains – to predict the action that

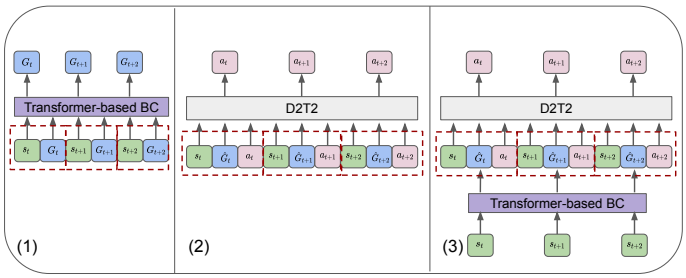

Fig. 2. D2T2 Framework Overview: (1) D2T2 first extrapolates a steering guidance function with transformer-based behavior cloning where the labels for cloning are inferred from the offline dataset via TD learning; (2) During training, D2T2 aligns the generated steering-guided action with the observed action via supervised learning; (3) In evaluation, D2T2 uses the learned steering guidance function to generate actions leading to optimal returns.

most likely leads to the desired value of cumulative reward. This is a challenging long-horizon prediction problem as it requires consideration of *all* the future rewards that need to sum up to the desired return. Thus, we propose a new and less challenging prediction problem for DTs: *Which action is most likely to lead to a desired state as early as possible?* A solution to this problem can be used to obtain the sequence of actions that leads to desired states with high expected returns. Formally, we define the new prediction problem.

Problem 2. *For a given desired state s^* and a sufficiently large discount factor $\gamma < 1$, predict an action a_t that maximizes the discounted probability of reaching s^*; i.e.,*

$$a_t \in \underset{a}{\operatorname{argmax}} \left\{ \max_{A_{t+1:T}} \sum_{t'=t+1}^{T} \gamma^{t'-t} \mathbb{P}(s_{t'}=s^*, s^* \notin \tau_{t'-1} | \tau_t, a, A_{t+1:T}) \right\} \quad (3)$$

where exponentially decaying γ encourages reaching s^* faster.

Based on our preceding analysis and motivations, in this work, we introduce D2T2 (Fig. 2), which uses an input signal to transform the original prediction problem of DTs into the less challenging one described above. Notably, the input signal employs TD learning to address the growing variance problem and eliminates the need for the RTG during evaluation, another prominent challenge faced by the original DTs.

III. D2T2: Decision Tranformers Steered via Temporal Difference Learning

D2T2 converts the tasks of decision-making to sequence modeling problems via supervised learning on an offline dataset $\mathcal{D}$ in order to extrapolate a policy $\pi_\theta(a_t | s_{t-k:t}, a_{t-k:t-1}, \widehat{G}_{t-k:t})$, *i.e.*, the action a_t at time step t is determined not only based on the sequence of prior states $s_{t-k:t}$ and actions $a_{t-k:t-1}$ but also a sequence of steering guidance signals (SGs), $\widehat{G}_{t-k:t}$. At each time step t, $\widehat{G}_t$ provides a signal that can lead π_θ toward achieving a predetermined goal. In the original DT model, $\hat{G}_t$ at time step t is R_t (i.e., the RTG at time step t). On the other hand, D2T2 employs a novel SG that involves a distinct goal with a shorter horizon and utilizes TD learning to address the growing variance problem of the RTG. This leads to superior performance of D2T2 over the original DTs, especially in stochastic tasks (as shown in Section IV). To simplify notation, in the following sections, we use $\widehat{G}_t$ to denote the signal input into D2T2 that is generated by a learned parametric function $\bar{g}_\zeta$, and G_t to denote a signal that is computed from the information in the offline dataset $\mathcal{D}$ and will be used in the training of $\bar{g}_\zeta$. We next introduce the SG used by D2T2.

Algorithm 1 SG Learning for D2T2

Input: Training dataset $\mathcal{D} = \{\tau_1, \tau_2, ..., \tau_n\}$ of training trajectories, approximate optimal value function $V(s)$ from TD learning
Output: Learned function for SG generation $\bar{g}_\zeta(s_{t-k:t})$
Step *I* in Section III-A
for each $\tau_i = \{s_0, a_0, s_1, a_1, ...\}$ in $\mathcal{D}$ **do**
  for $t = 0$ **to** $T-1$ **do**
    $G_t = g(s_t) = \operatorname{argmax}_{s_j} \{\gamma^{j-t} V(s_j) | j > t, s_j \in \tau_i\}$
    Add G_t to τ_i
  end for
  $G_t = s_T$
  Add G_t to τ_i
end for
Step *II* in Section III-A
Behavior cloning with causal transformer by optimizing

$$\min_{\zeta} \frac{1}{N} \sum_{\tau_i \in \mathcal{D}} \frac{1}{T} \sum_{s_t \in \tau_i} ||g(s_t) - \bar{g}_\zeta(s_{t-k:t})||^2$$

with optional VAE.

A. Learnable Steering Guidance over Latent Representations

Value functions $V(\cdot)$ can guide DTs toward achieving optimal cumulative rewards, as discussed in Section II-C. However, it requires $V(\cdot)$ to be near-optimal in order to provide effective guidance, which is challenging in offline RL due to the limited coverage of the state-action space provided by the offline dataset $\mathcal{D}$. To this end, we leverage variational inference [23] to encode guidance provided by the sub-optimal value functions into a compact and expressive latent space; this distills the knowledge acquired from state values as well as environmental transitions and rewards, to formulate the final SGs. The detailed steps for generating SGs are introduced below and summarized in Algorithm 1.

Step *I*. As discussed in Section II-C, compared to the value of the desired future cumulative rewards, a desired next state can potentially decrease the horizon of the decision problem, thus improving the learning efficiency. Accordingly, the first step toward constructing the SGs is to generate a mapping function $g : \mathcal{S} \to \mathcal{S}$ such that each state $s_t \in \tau_i$ is mapped to a corresponding desired next state $G_t = g(s_t) \in \tau_i$ that has the maximum value, *i.e.*,

$$G_t = g(s_t) = \underset{s_j}{\operatorname{argmax}} \{\gamma^{j-t} V(s_j) | j > t, s_j \in \tau_i\}; \quad (4)$$

here, $V(\cdot)$ is an approximate optimal value function that estimates the maximum expected cumulative return from state s, *i.e.*, $V(s) \approx \mathbb{E}_{\rho^{\pi^*}}[\sum_{k=0}^{T} \gamma^k r_{t+k+1} | s_t = s]$ where π^* is the optimal policy. In practice, although one can use any temporal difference (TD) algorithms for optimal value

function approximation, the implicit Q-learning (IQL) [24] remains a compelling algorithm for learning $V(\cdot)$; thus, we alternatively use Q-learning for descriptions of our methods and experiments. We note the purpose of the discount factor γ in (4) is to motivate early achievement of the desired next state. In other words, if two future states have the same value, then the one that appears earlier in the trajectory is more desired as this can help improve the future cumulative reward.

Step *II*. As discussed in Introduction, a drawback of DTs is that the RTGs are not available during deployment (i.e., after the training is completed), as they depend on future information. Hence, they become hyper-parameters to be tuned which may lead to unstable performance. The guidance provided by the value function, following (4), faces the same limitation as it depends on the value of future states. To address this, we employ behavior cloning with a causal transformer [10] (Fig. 2) to learn a function $\bar{g}_\zeta : \mathcal{S}^k \to \mathcal{S}$ that extrapolates the function $g(\cdot)$ by minimizing the squared loss

$$\min_{\zeta} \frac{1}{N} \sum_{\tau_i \in \mathcal{D}} \frac{1}{T} \sum_{s_t \in \tau_i} ||g(s_t) - \bar{g}_\zeta(s_{t-k:t})||^2; \qquad (5)$$

here, ζ represents the parameters of the causal transformer, N is the number of trajectories in the offline dataset $\mathcal{D}$, and T is the horizon of the environment. At each time step t, given the sequence of prior states $s_{t-k:t}$, the SG $G_t = g_t(s_t) \approx \bar{g}_\zeta(s_{t-k:t})$ can be obtained without leveraging any information from the future. We choose a causal transformer over other architectures (*e.g.*, MLP, RNN, LSTM) to ensure that the model has access to the entire long-horizon sequence. We specifically choose the transformer architecture as it has shown to be effective in processing long input sequences [25].

Moreover, given that the approximate optimal value function $V(\cdot)$ is highly likely to be sub-optimal if the offline dataset does not comprehensively cover the state and action space [5], its approximation errors can be propagated into $\bar{g}_\zeta(\cdot)$ whose training requires $V(\cdot)$ as shown by (5) and (4). Thus, directly using $\bar{g}_\zeta(\cdot)$ as the SG could be problematic, as the errors from the previous two steps can be propagated into the DT, in addition to the supervised learning error from the training itself. To resolve this, we leverage variational inference [23] to concentrate the learned knowledge into a compact and expressive latent space via a variational auto-encoder (VAE), leading to $\widehat{G}_t \sim q_\psi(\cdot|\bar{g}_\zeta)$, where $q_\psi(\cdot|\bar{g}_\zeta)$ is the approximate posterior that encodes $\bar{g}_\zeta$ to the latent space.

In practice, we observe that variational inference is not always necessary. For example, for complex tasks with complicated environment dynamics or high-dimensional states, a VAE can improve the performance. On the other hand, for simple tasks, the negative impact induced by the learning error of the VAE may outweigh the benefits brought by the latent space. In this case, variational inference should be employed with caution. Hence, the SG inputs of D2T2 in Fig. 2 can be either $\widehat{G}_t = \bar{g}_\zeta(s_{t-k:t})$ or $\widehat{G}_t \sim q_\psi(\cdot|\bar{g}_\zeta)$, depending on whether or not a latent representation is employed. Please refer to our technical report [22] for additional details. In our experiments in Section IV, we explicitly report which SG is used by D2T2.

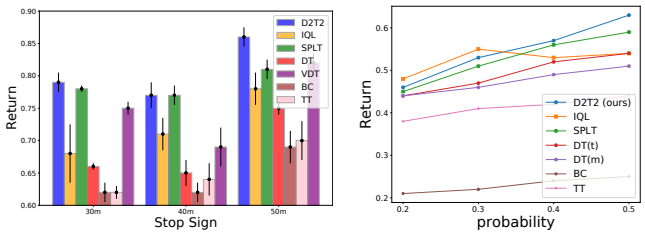

Fig. 3. (left) Tailgate task performance with different stop signs (data optimality): the smaller the stop sign is, the higher the crash rate of the data is. (right) FrozenLake task performance with different probability (stochasticity): the smaller the x-value is, the larger the stochasticity is.

B. D2T2 Training and Evaluation

In contrast to the original DT where the RTGs are not available during evaluation and deployment, the SG of D2T2 can be determined without the need for future information. Specifically, at step t, the inputs to D2T2 can be formulated as

$$\tau_{input} = (s_0, a_0, \widehat{G}_0 ..., s_{t-1}, a_{t-1}, \widehat{G}_{t-1}, s_t, \widehat{G}_t), \qquad (6)$$

with $\widehat{G}_t \sim q_\psi(\cdot|\bar{g}_\zeta(s_{t-k:t})$ following the step above where $\bar{g}(s_{t-k:t})$ is deterministic, as previously introduced. Hence, a D2T2 can be trained with only the offline trajectories $\tau_i \sim \mathcal{D}$ to minimize the squared loss between the actions provided in the offline trajectory, $a \sim \tau_i$, and the actions predicted from D2T2, $a_t = \pi_\theta(s_{t-k:t}, a_{t-k:t-1}, \widehat{G}_{t-k:t})$, following the regular supervised learning schema as in the original DT [10]. Upon evaluation, a D2T2 takes as the input the sequence (6) (up to the current step t), and subsequently predicts the action, following $a_t = \pi_\theta(s_{t-k:t}, a_{t-k:t-1}, \widehat{G}_{t-k:t})$. The training and testing stages are summarized in Algorithm 2 in [22].

IV. Experiments

In this section, we provide a comprehensive empirical study of our method and compare it with state-of-the-art methods. We consider a broad range of tasks, including 2 illustrative stochastic examples: Tailgate driving and FrozenLake; 2 stochastic CARLA benchmarks; 3 suites from D4RL: Gym-MuJoCo, AntMaze, and FrankaKitchen [6]; and a real-world robot arm manipulation experiment. We first demonstrate the implications of our analysis in Section II-B, which motivates the integration of DT with the value function, which we refer to as VDT. We then focus on demonstrating the competitive performance of our proposed algorithm D2T2.

We mainly compare methods that follow a similar architecture branch: Decision Transformer (DT) [10], Trajectory Transformer (TT) [26], SeParated Latent Trajectory Transformer (SPLT) [16], and transformer-based behavior cloning (BC) [10]. We also include a strong offline Q-learning (IQL) for comparison. For D4RL tasks, we compare methods that are representative of both Q-learning and RvS. In the former category, we compare Conservative Q-learning (CQL) [27] and IQL. In the latter, we consider both (1) fully-connected architectures: RvS-R and RvS-G (learning with either conditioned on goals or rewards [14] and (2) transformer-based models: DT, TT, SPLT, and transformer-based behavior cloning (BC). We also include mildly conservative Q-learning (MCQ) [28] to Gym-MuJoCo suite and contrastive RL (CL)

[29] to AntMaze-v2 suite as strong baselines. Note that D2T2 employs VAE in all the tasks other than the two illustrative tasks (Tailgate and FrozenLake) due to their simplicity. Due to space limitation, please refer to our technical report [22] for detailed experimental setup, ablation studies, and results.

Tailgate Driving. The goal of the tailgate driving problem is to maximize the ego vehicle's distance within one episode following a leading vehicle in the same direction on a one-dimensional path. We collected trajectories following the prior work [16] for an autonomous driving problem whose state space is the absolute position and velocity of both the ego and leading vehicles, $s \in [e_x, e_v, l_x, l_v]$, and the action space is the acceleration of the ego vehicle $a \in [-1, 1]$. The leading vehicle either begins hard-braking at the last possible moment to stop just before the d mark meters and then resumes going forward; or speeds up to the maximum speed and continues for the entire trajectory, which makes this scenario a stochastic event to the ego vehicle.

We collected around 100k samples for each dataset with different values of d as the stop sign and its corresponding crash rate as the data quality. The reward for each time step records the normalized distance $d_n = \frac{d}{d_{max}}$, where d is the actual distance the ego vehicle moving forward and d_{max} is the maximum moving distance between the ego and leading vehicles considering the safe following distance. We report the task performance with standard error in Fig. 3(left). Our empirical results show that VDT improves DT substantially, confirming our findings in Section II-C. Note that DT used here has already been hand-tuned. The DT conditioned on the maximum return in the dataset performs worst, and is therefore omitted. Our empirical results in this stochastic problem show that D2T2 has a significantly higher return compared with the original DT and VDT, which supports our previous argument. Therefore, we focus on D2T2 as our method for the remaining experiments.

Stochastic Frozenlake. We conducted experiments on the datasets with different probabilities p of moving toward the intended direction and a probability $1-p$ of moving to either between two sides of the intended directions to evaluate with different levels of stochasticity. The lower the value of p, the more stochastic the environment. We varied the value of p for data collection and evaluation. The offline trajectories have a length of 100 and are acquired via training a DQN [30] policy in the *gym* environment. In Fig. 3(right), we show that our D2T2 achieves higher returns among all stochasticity levels compared with most of the baselines, including SPLT, which is state-of-the-art in solving stochasticity issues of DT. DT(m) refers to DT conditioned on the maximum return in the dataset while DT(t) refers to DT with a hand-tuned conditional return. The standard error is small enough to be negligible for all methods. We observe that IQL is a strong comparison for very low p while our performance is still competitive. Additionally, when p increases to 0.6, which is still not a large probability, D2T2 outperforms IQL significantly.

CARLA NoCrash Benchmark. We evaluated our method on the CARLA NoCrash [21], [31] benchmark whose goal is to navigate in a suburban town to a desired goal waypoint from a predetermined start waypoint without crashing. The benchmark consists of 2 towns *Town01* and *Town02*, each with 25 different routes. We trained our method D2T2 and all the baselines on the Town01 dataset and then evaluated them on the unseen Town02 routes.Table I presents a comparison of two metrics, showing that DT(t) with tuned target return (RTG) outperforms DT(m) with the maximum return in terms of both success rate and speed. However, it would be difficult to estimate the best target return without online evaluation or prior domain knowledge, especially since the training and testing are in different scenarios, *i.e.,* Town01 and Town02. By contrast, D2T2 leverages the benefit of Q-learning with a properly designed guidance signal, leading to both the highest success rate and speed and outperforming IQL.

TABLE I

SUCCESS RATE (%) AND SPEED (M/S) ON NOCRASH BENCHMARK*

Metric	BC	TT	DT(m)	DT(t)	IQL	SPLT	D2T2
Success (%)	92.2 ±0.5	85.4 ±2.5	89.7 ±6.2	94.2 ±2.9	97.5 ±0.4	95.1 ±2.6	**98.3** ±0.8
Speed (m/s)	2.44 ±0.01	2.62 ±0.30	2.70 ±0.06	2.76 ±0.03	2.79 ±0.06	2.75 ±0.09	**2.81** ±0.11

*We evaluated D2T2 with the baselines for 10 seeds through all 25 routes in the unseen Town02. DT(m) refers to DT conditioned on the maximum return in the dataset. DT(t) refers to DT with a hand-tuned conditional return. For both Success (%) and Speed (m/s), a larger value is better.

TABLE II

MULTIPLE METRICS ON LEADERBOARD BENCHMARK*

Metric	BC	TT	DT(m)	DT(t)	IQL	SPLT	D2T2
Total score	53.4 ±7.1	56.2 ±8.5	62.3 ±11.3	68.6 ±4.5	63.6 ±6.8	65.8 ±4.3	**70.2** ±4.5
Completion (%)	94.2 ±4.5	70.6 ±10.2	95.6 ±4.8	98.3 ±2.7	**100.0** ±0.0	93.5 ±8.6	**100.0** ±0.0
Collision (/km)	4.1 ±1.2	2.8 ±2.2	1.8 ±1.7	**1.7** ±0.5	2.3 ±0.5	2.5 ±0.3	1.8 ±1.3
Infraction (/km)	2.4 ±2.2	**0.0** ±0.0	2.6 ±0.9	2.5 ±0.8	2.2 ±0.7	2.2 ±1.3	2.1 ±0.8

*We evaluated D2T2 with the baselines for 10 seeds and on Leaderboard devtest routes. DT(m) refers to DT conditioned on the maximum return in the dataset. DT(t) refers to DT with a hand-tuned conditional return. For both Total Score and Completion (%) a larger value is better, while for Collision (/km) and Infraction (/km) a smaller value is better. A larger performance value is better.

CARLA Leaderboard Benchmark. We next evaluated our method on a modified version of the CARLA Leaderboard benchmark following the setting in [16]. Compared with NoCrash Benchmark, this task involves more maneuvers like lane-changing in urban and highway situations as well as 8 additional variables. The results are summarized in Table II. Total scores are calculated with route completion rate, collisions per kilometer (/km), and infractions (/km), which is the main indicator of the performance for all methods. Notably, the high performance of DT(t) relies heavily on manually tuning the target return, similar to the approach used in the NoCrash benchmark. Although SPLT is able to disentangle the world dynamics and the agent policy, leading to competitive performance in CARLA Leaderboard scenarios,

our better total score indicates that steering guidance is a more proper condition variable.

D4RL Suites. As discussed in Section II-B, DT performs competitively in deterministic or near-deterministic tasks, such as those in the D4RL suites. From Table III, we observe that D2T2 outperforms the original DT performs competitively with strong offline Q-learning methods (*e.g.,* IQL and CQL) in MuJoCo tasks. We report the other complete results with standard error for AntMaze and FrankaKitchen tasks in [22] showing that D2T2 significantly outperforms DT in AntMaze environments. We posit that D2T2 indirectly improves its stitching ability (*i.e.,* learning an optimal policy from sub-optimal trajectories) through its integration with Q-learning [17].

Experiments on UR5e Robot Arm. Finally, we compare our D2T2 with the vanilla DT. We conducted a stochastic pushing task for a UR5e robot arm, which aims to push an object to a target position. We formulate this task as an MDP where both the action and state spaces are defined by the positions of the current object and the target, as shown in Fig. 1. The reward function is the distance between the object and the target. During the data collection using the UR5e robot arm, we started by detecting the current object, the positions of the target, and the end effector of the robot arm via QR codes. We then divided the whole motion planning into 4 steps with the input states as the concatenation of the positions of the current object and the target: the end effector (1) approaches the object from above, (2) it goes down to the same height as the object, then (3) pushes the object to the target's position, and lastly (4) goes up to the previous height. We generated offline near-optimal trajectories using expert demonstrations with object and target offsets in execution. We then added the action noise given the current state (object and target position) to increase the stochasticity. In addition, with probability $p \leq 0.3$, the action is not executed and the state does not change. If the object does not reach the target, the robot arm resets and tries again with up to 10 steps.

akbtodoI think this sentence can be rephrased in a better wayWe eventually trained our collected offline dataset (*i.e.,* 104 samples) on both DT and D2T2 and then re-deployed both policies on the UR5e robot arm. We evaluated both approaches with 12 different initial states via 3 different criteria: reward, success rate (*i.e.,* whether achieving the target eventually), steps (*i.e.,* number of steps to achieve the target). In Table IV, we show that D2T2 outperforms DT in all criteriawith smaller variance. In addition, DT could not accurately predict how to approach the object in one of the initial states no matter how many steps it was given.

V. Related Work

Reinforcement Learning as Sequence Modeling: The use of transformer architectures has gained prominence in addressing RL challenges. DTs [10] and Trajectory Transformers (TTs) [26] stand out for their adeptness in fitting reward-conditioned policies and modeling trajectory distributions, respectively. Several studies target the challenges DTs face due to stochasticity. For example, ESPER [15] learns trajectory representations disentangled from environmental dynamics via adversarial clustering and SPLT [16] learns environmental stochasticity and agent policy separately with two transformers. Similarly, DoC [32] predicts the trajectories' representations by minimizing the mutual information between the representation and the environment transition.

In robotics, the DT architecture can be customized to specific needs. For instance, the inputs can be only states and actions while incorporating pre-trainedterrain parameters with privileged information [33]. Instead of including the RTG as an input, alternative methods integrate skill prediction modules [34] and embodiment [35] as additional inputs. The work [36] divides the state into images and texts as two separate inputs. None of them keeps the RTG as the guidance.

Integration of Decision Transformers and Q-learning: Alternatively, we combine DTs with Q-learning to alleviate the stochasticity problem for DTs. Several works have attempted to combine DTs with dynamic programming (*e.g.,* Q-learning). Among them, [17], [19] relabel the RTG in DTs with precomputed conservative value functions to improve DT's stitching ability. EDT [18] adjusts the context length of DT with interpolation between trajectory stitching and behavior cloning. Q-transformer [37] uses a Transformer to provide a scalable representation for Q-functions trained via offline TD backups. However, none of them tackles stochastic tasks.

VI. Conclusion

In this work, we investigated DTs to understand their strengths and weaknesses. Our analysis provided an explanation for the strong performance of DTs in deterministic tasks, revealing a potential reason for their less competitive performance in stochastic environments, and suggesting two potential improvements. Motivated by these insights, we introduced a new approach, D2T2, with a TD-learned guiding signal that significantly improves the performance of DTs in stochastic tasks. In a variety of environments and tasks, our method outperformed state-of-the-art methods, revealing the promising potential of combining DTs with TD learning.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 2013.

[2] N. Gürtler, S. Blaes, P. Kolev, and et al, "Benchmarking offline reinforcement learning on real-robot hardware," in *International Conference on Learning Representations*, 2023.

[3] A. Kumar, A. Singh, S. Tian, C. Finn, and S. Levine, "A workflow for offline model-free robotic rl," in *Conference on Robot Learning*, 2021.

[4] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine, "Cog: Connecting new skills to past experience with offline reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2020.

[5] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020.

[6] J. Fu, A. Kumar, G. Tucker, and S. Levine, "D4rl:datasets for deep data-driven reinforcement learning," in *arXiv:2004.07219*, 2020.

[7] J. Liu, Z. Zhang, Z. Wei, and et al, "Beyond ood state actions: Supported cross-domain offline reinforcement learning," in *Annual AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

[8] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," in *CoRL 2019: Volume 100 PMLR*, 2019.

[9] A. Brohan, N. Brown, J. Carbajal, and et al, "Rt-1: Robotics transformer for real-world control at scale," *Robotics: Science and Systems*, 2023.

TABLE III

AVERAGED NORMALIZED SCORES ON GYM-MUJOCO SUITE*

Environment	BC	RvS-R	DT	TT	SPLT	CQL	IQL	MCQ	D2T2
halfcheetah-Med-Expert-v2	59.9	92.2	86.8±1.3	**95.0±0.2**	91.8±0.5	91.6	86.7	87.5±1.3	90.9±0.8
walker2d-Med-Expert-v2	36.6	106.0	108.1±0.2	101.9±6.8	108.6±1.1	108.8	109.6	114.2±0.7	**109.9±1.7**
hopper-Med-Expert-v2	79.6	101.7	107.6±1.8	110.0±2.7	104.8±1.1	105.4	91.5	111.2±0.1	**114.8±0.5**
average-Med-Expert-v2	58.7	100.0	100.8	102.3	100.7	95.9	101.9	104.3	**107.1**
halfcheetah-Med-Replay-v2	4.3	38.0	36.6±0.8	41.9±2.5	42.7±0.3	45.5	44.2	56.8±0.6	**62.5±0.2**
walker2d-Med-Replay-v2	36.9	60.5	66.6±3.0	82.6±6.9	57.7±4.7	77.2	73.9	91.3±5.7	**101.8±5.2**
hopper-Med-Replay-v2	27.6	73.5	82.7±7.0	91.5±3.6	75.0±23.8	95.0	94.7	**101.6±0.8**	92.8±4.7
average-Med-Replay-v2	22.9	57.3	62.0	72.0	58.5	72.6	70.9	83.2	**85.7**
halfcheetah-Medium-v2	43.1	41.6	42.6 ±0.1	46.9±0.4	44.3±0.7	44.0	47.4	64.3±0.2	**79.6±0.8**
walker2d-Medium-v2	77.3	71.7	74.0±1.4	79.0±2.8	77.9±0.3	72.5	78.3	**91.0±0.4**	89.2±0.4
hopper-Medium-v2	63.9	60.2	67.6±1.0	61.1±3.6	53.4±6.5	58.5	66.3	**78.4±4.3**	74.8±3.4
average-Medium-v2	61.4	57.8	61.4	62.3	58.5	58.3	64.0	77.9	**81.2**
average-Gym-v2	47.7	71.7	74.7	78.9	72.9	77.6	76.9	88.5	**91.3**

*We use the results from the TT paper [26] for BC, DT, and TT and the results from the IQL paper [24] for CQL and IQL. For RvS-R [14], SPLT [16], and MCQ [28], we use the results reported from their own papers. We report the mean and standard error for our method over 10 seeds. The top scores are in bold.

TABLE IV

REWARD, NUMBER OF STEPS FOR COMPLETION, AND SUCCESS RATE (%) ON REAL ROBOT MANIPULATION TASK*

Metric	DT	D2T2
Reward	-0.157 ± 0.228	$\mathbf{-0.110} \pm 0.111$
Success (%)	91.7 ± 27.64	$\mathbf{100} \pm 0$
Steps	4.83 ± 7.71	$\mathbf{3.08} \pm 1.93$

*We evaluated D2T2 with a DT on 12 different initial states. For both Reward and Success (%) a larger value is better while we aim to complete the task with a smaller number of steps.

[10] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 15084–15097, 2021.

[11] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel, "Goal-conditioned imitation learning," *Advances in Neural Information Proc. Sys.*, 2019.

[12] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. M. Devin, B. Eysenbach, and S. Levine, "Learning to reach goals via iterated supervised learning," in *International Conference on Learning Representations*, 2021.

[13] F. Codevilla, M. Muller, A. Lopez, V. Koltun, and A. Dosovitski, "End-to-end driving via conditional imitation learning," in *2018 Int. Conf. on Robotics and Automation (ICRA)*, p. 4693–4700, IEEE, 2018.

[14] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, "Rvs: What is essential for offline rl via supervised learning?," in *arXiv preprint arXiv:2112.10751*, 2021.

[15] K. Paster, S. McIlraith, and J. Ba, "You can't count on luck: Why decision transformers fail in stochastic environments," in *arXiv preprint arXiv:2205.15967*, 2022.

[16] A. R. Villaflor, Z. Huang, S. Pande, J. M. Dolan, and J. Schneider, "Addressing optimism bias in sequence modeling for reinforcement learning," in *39th Int. Conf. on Machine Learning*, vol. 162, pp. 22270–22283, PMLR, 2022.

[17] T. Yamagata, A. Khalil, and R. Santos-Rodríguez, "Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline RL," in *International Conference on Machine Learning*, 2023.

[18] Y.-H. Wu, X. Wang, and M. Hamaya, "Elastic decision transformer," in *37th Conference on Neural Information Processing Systems*, 2023.

[19] D. Lawson and A. H. Qureshi, "Control transformer: Robot navigation in unknown environments through prm-guided return-conditioned sequence modeling," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 9324–9331, IEEE, 2023.

[20] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[21] A. Dosovitskiy, F. Codevilla, A. Lopez, and V. Koltun, "An open urban driving simulator," in *Conf. on Robot Learning*, pp. 1–16, 2017.

[22] H.-L. Hsu, A. K. Bozkurt, J. Dong, Q. Gao, V. Tarokh, and M. Pajic, "Steering decision transformers via temporal difference learning," tech. rep., Duke University, 2024. `https://cpsl.pratt.duke.edu/files/docs/d2t2.pdf`.

[23] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[24] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *Int. Conf. on Learning Repr. (ICLR)*, 2022.

[25] V. Melnychuk, D. Frauen, and S. Feuerriegel, "Causal transformer for estimating counterfactual outcomes," in *Int. Conf. on Machine Learning*, pp. 15293–15329, PMLR, 2022.

[26] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Advances in Neural Information Processing Systems*, 2021.

[27] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *arXiv:2006.04779*, 2020.

[28] J. Lyu, X. Ma, X. Li, and Z. Lu, "Mildly conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, 2022.

[29] B. Eysenbach, T. Zhang, R. Salakhutdinov, and S. Levine, "Contrastive learning as a reinforcement learning algorithm," in *arXiv preprint arXiv:2206.07568*, 2022.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *arXiv preprint arXiv:1312.5602*, 2013.

[31] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. of the IEEE/CVF Int. Conf. on Computer Vision*, pp. 9329–9338, 2019.

[32] M. Yang, D. Schuurmans, P. Abbeel, and O. Nachum, "Dichotomy of control: Separating what you can control from what you cannot," in *International Conference on Learning Representations*, 2023.

[33] H. Lai, W. Zhang, X. He, C. Yu, Z. Tian, Y. Yu, and J. Wang, "Sim-to-real transfer for quadrupedal locomotion via terrain transformer," in *2023 Int. Conf. on Robotics and Automation (ICRA)*, IEEE, 2023.

[34] X. Huang, D. Batra, A. Rai, and A. Szot, "Skill transformer: A monolithic policy for mobile manipulation," in *Proc. of the IEEE/CVF International Conference on Computer Vision*, pp. 10852–10862, 2023.

[35] C. Yu, W. Zhang, H. Lai, Z. Tian, L. Kneip, and J. Wang, "Multi-embodiment legged robot control as a sequence modeling problem," in *2023 Int. Conf. on Robotics and Automation (ICRA)*, IEEE, 2023.

[36] A. Hu, L. Russell, H. Yeo, Z. Murez, G. Fedoseev, A. Kendall, J. Shotton, and G. Corrado, "Gaia-1: A generative world model for autonomous driving," in *arXiv preprint arXiv:2309.17080*, 2020.

[37] Y. Chebotar, Q. Vuong, and et al, "Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions," in *Conference on robot learning*, Proc. of Machine Learning Research, PMLR, 2023.