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]. This problem is particularly pronounced in the field of general robotics [3]–[5], autonomous vehicles [6], [7] and drones [8], [9], where learning from scratch can be both expensive and hazardous [10], [11]. Offline RL [12] holds great promise to unlock the full potential of RL as it eliminates the necessity of 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 [13], [14]. Consequently, offline RL has attracted significant attention, aimed to learn effective and generalizable policies from offline datasets (e.g., [13], [15]–[17]).

Decision Transformers (DTs) [18], recently introduced as a part of a more general framework called RL via supervised learning (RvS) [19]–[21], have shown promising performance in offline RL tasks, outperforming conventional RL methods such as Temporal-Difference (TD) learning [22]. 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 [18], it has been shown that DT yields poor performance for environments exhibiting highly stochastic behaviors [23], [24] and has poor stitching ability [25], [26]. The poor DT performance in stochastic environments is the main performance bottleneck [23] that substantially limits its use in many robotics environments (e.g., [25], [27], [28]).

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 being accumulated over the horizon is the main cause of the performance degradation in the stochastic environments. To demonstrate the correctness of our analysis, we empirically show that replacing RTG with an approximated optimal value function yields higher performance than the original DT-based approach.

Motivated by the insights from this analysis, we introduce Steering *Decision Transformers via Temporal Difference* learning (D2T2), which integrates DTs with approximated 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 [29]. 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 environment, our approach transforms long horizon tasks into shorter ones. With the transformed learning problem, D2T2 further improves the performance of DTs. Also, 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 [24]), two stochastic CARLA benchmarks [30], and three suites (18 tasks) from D4RL (Gym-MuJoCo, AntMaze, and FrankaKitchen) [13] as well as a real-world robotic arm manipulation environment (see Fig. 1). This set of environments with tasks of different levels of difficulty enabled us to comprehensively investigate the capability 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

*equal contribution with alphabetical order

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

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.

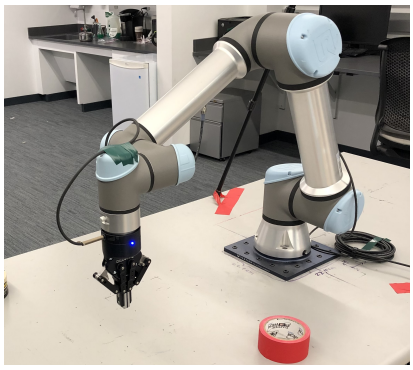


Fig. 1. A UR5e trying to push a red tape to a target position.

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 for $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$; and T is the horizon. We consider finite-horizon MDPs, without discounting, where the ultimate goal is to learn an optimal policy π^* that maximizes the expected total return, *i.e.*,

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

here, ρ_π is the state-action distribution under policy π and r_t is a random variable that represents the reward received by the agent at time t. 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'). We note that $\mathcal{D}$ is usually collected over some behavior policy π_b which can be either a single policy or a mixture of policies and they 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 the 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.

Proposition 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 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 of Theorem 1. Please see in Appendix. □

Theorem 1 concludes that the DT outputs the optimal trajectory almost surely in deterministic environments. The key takeaway is that *the success of DT relies on the successful estimation of R_t^**. In deterministic environments, R_t^* can be estimated almost surely 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 to build an RTG signal that only uses s_t at time step t. We note that this solution shares the same motivation and theoretical support as the use of TD learning to approximate the cumulative future rewards in AC [29], 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, using only 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

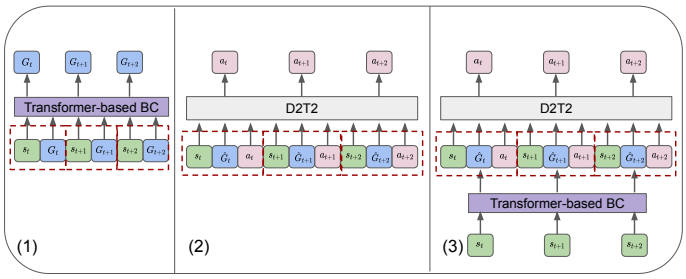


Fig. 2. Overview of the D2T2 Framework. (1) D2T2 first extrapolates a steering guidance function with transformer-based behavior cloning where the labels for cloning are inferred from the offline dataset through TD learning. (2) During training, D2T2 aligns the generated steering-guided action with the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

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

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.

With all the aforementioned analysis and motivations, in this work, we introduce D2T2 (Fig. 2), which has an input signal that can modify the original prediction problem of the DTs into the less challenging one above. Notably, the input signal employs TD learning to address the growing variance problem and eliminates the need for the RTG upon 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 employ a novel SG that involves a distinct goal with a shorter horizon and utilize 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, approximated 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 the 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 considered 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 [31] 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 nextstate 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.*,

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

here, $V(\cdot)$ is an approximated 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) [32] 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 Section I, a drawback of the original DTs is that the RTGs are not available during evaluation/deployment 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 from 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 causal transformer [18] (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, *i.e.*,

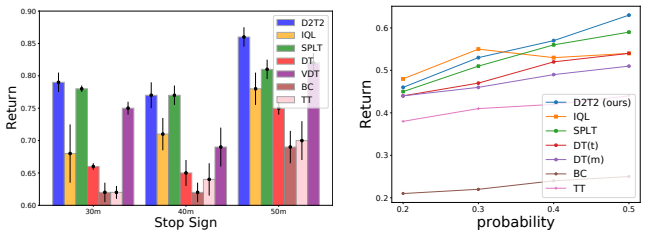
$$\min_{\zeta} \frac{1}{N} \sum_{\tau_i \in \mathcal{D}} \frac{1}{T} \sum_{s_t \in \tau_i} ||g(s_t) - \bar{g}_\zeta(s_{t-k:t})||^2, \quad (5)$$

where ζ is 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. Consequently, 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 whole long-horizon sequence. Here, we specifically choose the transformer architecture as they have shown to be effective in processing long input sequences [33].

Moreover, given that the approximated 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 [12], 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 both be propagated into the DT, in addition to the supervised learning error from the training itself. To resolve this, we leverage variational inference [31] 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 approximated 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 dynamic 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 Appendix VIII for additional details.

In our experiments in Section IV, we explicitly report which SG is used by D2T2.



(a) Tailgate Performance (b) Lake Performance

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

B. D2T2 Training & Evaluation

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

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

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 introduced above. Consequently, 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 [18]. 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 Appendix.

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 [13]; 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) [18], Trajectory Transformer (TT) [34], SeParated Latent Trajectory Transformer (SPLT) [24], and transformer-based behavior cloning (BC) [18]. 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) [35] and IQL. In the latter category, we consider both (1) fully-connected architectures: RvS-R and RvS-G (learning with either conditioned on goals or rewards [22] and (2) transformer-based models: DT, TT, SPLT, and transformer-based behavior cloning (BC). In addition, we also include mildly conservative Q-learning (MCQ) [36] to Gym-MuJoCo suite and contrastive RL (CL) [37] to AntMaze-v2 suite as strong baselines. We 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 Appendix IX for detailed experimental setup, ablation studies, and additional experimental results.

Tailgate Driving. We conducted experiments on the Tailgate driving task to verify our theoretical insights that motivate the integration of TD with DT. 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 [24] 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 resume going forward; or speeds up to the maximum speed and continue 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 d as the stop sign and its corresponding crash rate as the data quality, which is shown in Table V in in Appendix. 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(a). Our empirical results show that VDT improves DT substantially, confirming our findings in Section II-C. Note that DT here has already been hand-tuned. The DT conditioned on the maximum return in the dataset performs worst, which is 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. Frozenlake environment with stochastic transitions is visualized in Fig. 4 in Appendix. We conducted experiments on the datasets with different probabilities p of moving toward the intended direction and the probability $1 - p$ of moving to either between two sides of the intended directions to evaluate with different levels of stochasticity. The lower is p, the more stochastic is the environment. We varied p value for data collection and evaluation. The offline trajectories have the length of 100 and are acquired via training a DQN [38] policy in gym environment. In Fig. 3(b), we show that our D2T2 earns 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) is DT conditioned on the maximum return in the dataset. DT(t) is DT with a hand-tuned conditional return. The standard error is small enough for all methods to be ignored. We observe that IQL is a strong comparison for really low p while our performance is still competitive. Also, when p increases to 0.6, which is still not a large probability, D2T2 outperforms IQL significantly.

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) is DT conditioned on the maximum return in the dataset. DT(t) is 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) is a DT conditioned on the maximum return in the dataset. DT(t) is a 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 NoCrash Benchmark. We evaluate our method on the CARLA NoCrash [30], [39] benchmark whose goal is to navigate in a suburban town to a desired goal waypoint from a predetermined start waypoint without crash, considering safety as a priority [6], [40] in tasks such as autonomous driving. The benchmark consists of 2 towns *Town01* and *Town02*, each with 25 different routes. We trained our method D2T2 as well as all the baselines on the Town01 dataset and then evaluated them in the unseen Town02 routes. In Table I, we present 2 metrics for comparison, observing that DT(t) with tuning target return RTG does improve

success rate and speed against DT(m) with maximum return in the dataset. 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.

CARLA Leaderboard Benchmark. We next evaluated our method on a modified version of the CARLA Leaderboard benchmark following the setting in [24]. 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. Among all the baselines, we would like to emphasize the high performance in DT(t) is similar to the NoCrash benchmark in that manually tuning the target return is arbitrary. Although SPLT is able to disentangle the world dynamics and the agent policy, resulting in a competitive performance in such complexity of 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 has competitive performance in deterministic or near-deterministic tasks, such as the ones in D4RL suites. From Table III, we observe that D2T2 outperforms the original DT with a competitive performance compared to 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 Table XIII, and XII in Appendix respectively. Specifically, D2T2 outperforms DT in AntMaze environments significantly in both Table XIII and Fig. 5 in Appendix. We believe that D2T2 indirectly improves its stitching ability (*i.e.,* learning the optimal policy from sub-optimal trajectories) by integrating with Q-learning [25].

Real-Word Experiment on UR5e Robot Arm. Finally, we mainly compare our D2T2 with vanilla DT. We conducted a stochastic pushing task for UR5e robot arm, which aims to push an object to a target position. We formulate this task as an MDP with both action and state space as 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. Specifically, 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. In practice, we generated offline near-optimal trajectories upon 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 effectively to change the state. If the object does not reach the target, the robot arm resets and tries again with up to 10 steps.

We 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 from all criteria and with a 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 is given.

V. Related Work

A. Offline Reinforcement Learning

Offline RL offers the advantage of learning policies solely from pre-existing data, eliminating the need for real-time interaction, which can be both hazardous and resource-intensive in real-world problems. However, mitigating the distribution shift inherent in offline RL, stemming from disparities between learned policies and behavior policies, is crucial. Previous studies have tackled this challenge through explicit constraints [41], [42] or implicit mechanisms [32]. Additionally, various approaches have been proposed to regularize dynamic programming [35], [43] to alleviate deviations from the behavior policy.

B. Reinforcement Learning as Sequence Modeling

On the other hand, the utilization of transformer architectures has gained prominence in addressing RL challenges. DTs [18] and Trajectory Transformer (TT) [34] stand out for their adeptness in fitting reward-conditioned policies and modeling trajectory distributions, respectively. Moreover, advancements have extended DTs to tackle offline safety settings [44], [45]. As the complexity of multi-task and long-horizon scenarios escalates, researchers have proposed solutions aimed at handling either multiple tasks [46], [47] or sub-tasks within a single overarching task [48].

In addition, several works target the vulnerability of DT with stochasticity issues. For example, ESPER [23] learns trajectory representations disentangled from environmental dynamics via adversarial clustering and SPLT [24] learns environmental stochasticity and agent policy separately with two transformers. Similarly, DoC [49] predicts the trajectories' representations by minimizing the mutual information between the representation and the environment transition.

For robotics applications in practice, the architecture of DT is flexible according to the purpose. For instance, the inputs are only states and actions while pre-training varying terrain parameters with privileged information [50]. Instead of having the RTG as a part of inputs, skill prediction modules [51] and embodiment [52] serve as additional inputs. The work [53] divides the state into images and texts as two separate inputs. None of them keeps the RTG as the guidance.

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 [34] for BC, DT, and TT and the results from the IQL paper [32] for CQL and IQL. For RvS-R [22], SPLT [24], and MCQ [36], we use the results reported from their own papers respectively. We report the mean and standard error for our method over 10 seeds. The top scores are in bold.

C. Integration of Decision Transformers and Q-learning

Alternatively, we incorporate DTs with Q-learning to alleviate the stochasticity problem for DTs. Several works have tried to combine DTs with dynamic programming (*e.g.,* Q-learning). Among them, some work [25], [28] relabel RTG in DTs with precomputed conservative value functions to improve DT's stitching ability. EDT [26] adjusts the context length of DT with interpolation between trajectory stitching and behavior cloning. Q-transformer [54] uses a Transformer to provide a scalable representation for Q-functions trained via offline TD backups. However, none of them tackle stochastic tasks. Another recent work WT [55] aims to utilize behavior cloning to mitigate stitching and instability initialization without considering the learned value functions.

VI. CONCLUSION

In this work, we investigated the Decision Transformers to understand their strengths and weaknesses. Specifically, our analysis provided an explanation for the strong performance of DTs in deterministic task, revealing a potential reason for its less competitive performance in stochastic environment, 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. On a variety of environments and tasks, our method outperformed state-of-the-art methods, revealing the promising potential of combining DTs with TD learning.

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.

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] N. Sontakke, H. Chae, S. Lee, T. Huang, D. W. Hong, and S. Ha, "Residual physics learning and system identification for sim-to-real transfer of policies on buoyancy assisted legged robots," in *2023 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

[4] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Learning and adapting agility skills by transferring experience," *Robotics: Science and Systems (RSS)*, 2023.

[5] J. Dong, H.-L. Hsu, Q. Gao, V. Tarokh, and M. Pajic, "Robust reinforcement learning through efficient adversarial herding," *https://arxiv.org/abs/2306.07408*, 2023.

[6] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7, IEEE, 2020.

[7] L. Wang, J. Liu, H. Shao, W. Wang, R. Chen, Y. Liu, and S. L. Waslander, "Efficient reinforcement learning for autonomous driving with parameterized skills and priors," *Robotics: Science and Systems (RSS)*, 2023.

[8] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, 2023.

[9] H. Hsu, H. Meng, S. Luo, J. Dong, V. Tarokh, and M. Pajic, "Reforma: Robust reinforcement learning via adaptive adversary for drones flying under disturbances," in *2024 International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] A. Padalkar, A. Pooley, A. Jain, and et al, "Open x-embodiment: Robotic learning datasets and rt-x models," *https://arxiv.org/abs/2310.08864*, 2023.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

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

[27] A. Correia and L. A. Alexandre, "Hierarchical decision transformer," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1661–1666, 2023.

[28] 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.

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

[30] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "An open urban driving simulator," in *Conference on robot learning*, Proceedings of Machine Learning Research, pp. 1–16, PMLR, 2017.

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

[32] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022.

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

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

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

[36] 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.

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

[38] 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.

[39] 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.

[40] H.-L. Hsu, Q. Huang, and S. Ha, "Improving safety in deep reinforcement learning using unsupervised action planning," in *2022 Int. Conf. on Robotics and Automation (ICRA)*, p. 5567–5573, IEEE, 2022.

[41] A. Kumar, J. Fu, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," in *arXiv preprint arXiv:1906.00949*, 2019.

[42] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*, p. 2052–2062, PMLR, 2019.

[43] S. Fujimoto, D. Meger, and D. Precup, "Offline reinforcement learning with fisher divergence critic regularization," in *International Conference on Machine Learning*, Proceedings of Machine Learning Research, p. 5774–5783, PMLR, 2021.

[44] Q. Zheng, A. Zhang, and A. Grover, "Online decision transformer," in *Int. Conf. on Machine Learning*, pp. 27042–27059, 2022.

[45] Q. Zhang, L. Zhang, H. Xu, L. Shen, B. Wang, Y. Chang, X. Wang, B. Yuan, and D. Tao, "Saformer: A conditional sequence modeling approach to offline safe reinforcement learning," in *International Conference on Learning Representations*, 2023.

[46] K.-H. Lee, O. Nachum, M. S. Yang, L. Lee, D. Freeman, S. Guadarrama, I. Fischer, W. Xua, E. Jang, H. Michalewski, and I. Mordatch, "Multi-game decision transformers," in *Advances in Neural Information Processing Systems*, pp. 27921–27936, 2022.

[47] M. Xu, Y. Lu, Y. Shen, S. Zhang, D. Zhao, and C. Gan, "Hyper-decision transformer for efficient online policy adaptation," in *International Conference on Learning Representations*, 2023.

[48] S. G. Konan, E. Seraj, and M. Gombolay, "Contrastive decision transformers," in *Proceedings of The 6th Conference on Robot Learning*, vol. 205 of *Proceedings of Machine Learning Research*, pp. 2159–2169, PMLR, 14–18 Dec 2023.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

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

[56] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

APPENDIX

VII. PROOF OF THEORETICAL RESULTS

Proof of Theorem 1. Following the proposed conjure that given the input R_t, DT is predicting the most likely action a_t that can lead a future cumulative reward of R_t, we assume that a well-trained DT can accurately predict such actions at each step.

First, it is important to note that the maximum probability that any action can lead to a future cumulative reward of the amount R_t^* is 1. For any initial state s_0, we consider the trajectory with the highest cumulative reward as the optimal trajectory and denote it by $\tau^* \doteq \{s_0(s_0^*), a_0^*, r_0^*, \ldots, s_T^*, a_T^*, r_T^*\}$. If $R_0 = R_0^*$, then a_0^* can lead to a trajectory with a cumulative reward of R_0^* with probability one because the environment is deterministic. If the optimal trajectory is unique, *i.e.,* only τ^* can has a cumulative reward of R_0^*, then because DT is well-trained, it must output a_0^* if the input is R_0^*. At time $t > 0$, because the environment is deterministic DT can receive R_t^* if taking action a_t^* and follow the sequence of actions listed in τ^*. This implies that DT can receive a cumulative reward of R_t^* with probability one by taking a_t^*. Hence, because DT is well-trained, it will output a_t^*.

Next we prove that $R_t = R_t^*$ if $R_0 = R_t^*$. We use induction to prove it. Because we have shown that a well-trained DT will output a_0^* and by assumption that the reward is deterministic, $R_1 = R_0^* - r_0^* = R_1^*$. Hence, the base case is proved. Assume toward induction that at $t = k > 2$, we have $R_{k-1} = R_{k-1}^*$. We have proved that at $t = k$, DT will output a_k^* and due to the fact the reward is deterministic, $r_k = r_k^*$, then by definition of R_t^*, $R_k = R_{k-1}^* - r_k^*$. This concludes the proof for the case where the optimal trajectory is unique. For the case where there are multiple optimal trajectories all with cumulative rewards of amount R^*, note that the sequence of actions generated by DT must lead to the realization of one of such optimal trajectories. Thereby the proof is concluded. □

Algorithm 2 Training and Evaluation of D2T2

Input: Training dataset $\mathcal{D} = \{\tau_1, \tau_2, ..., \tau_n\}$ of training trajectories, $V(s)$ from TD learning
for each $\tau_i = \{s_0, a_0, \hat{G}_0, s_1, a_1, \widehat{G}_1, ...\}$ in $\mathcal{D}$ **do**
  Compute $\pi_\theta{=}(a_t|s_{t-k:t}, a_{t-k:t-1}, \widehat{G}_{t-k:t})$
  Calculate $L_\theta(\tau){=}-\sum_t \log \pi_\theta(a_t|s_{t-k:t}, a_{t-k:t-1}, \widehat{G}_{t-k:t})$
  Backpropagate gradients w.r.t $L_\theta(\tau)$ to update model parameters θ
end for
{Evaluation of D2T2}
Input: Initial state s_0, behavior cloning transformer model π_ζ
for $t = 0$ **to** $T - 1$ **do**
  $\widehat{G}_t = \bar{g}_\zeta(s_{t-k:t})$ (optional: $\widehat{G}_t \sim q_\psi(\cdot|\bar{g}_\zeta)$)
  Acquire action from $\pi_\theta{=}(a_t|s_{t-k:t}, a_{t-k:t-1}, \widehat{G}_{t-k:t})$
  Receive next state from environment
end for

TABLE V
TAILGATE DRIVING ENVIRONMENT: INFORMATION FOR THE AUTONOMOUS DRIVING TASKS, INCLUDING THE NUMBER OF OFFLINE TRAJECTORIES AND CRASH RATE.

d **mark (m)**	**30**	**40**	**50**
# trajectories	2728	2771	2954
crash rate	22.69%	17.65%	10.47%

VIII. METHOD SUPPLEMENTARY

a) (Optional).: As discussed in Section III-A, given that the value function $V(\cdot)$ is considered sub-optimal, its approximation errors can be propagated into the downstream supervised model $\bar{g}_\zeta(\cdot)$ trained following (5). As a result, directly using $\bar{g}_\zeta(\cdot)$ as the guidance signal for DT could be problematic, as the errors from the previous two steps can all be propagated into DT, in addition to the supervised learning error from the training of DT itself. To resolve this, we leverage variational inference [31] to concentrate the learned knowledge into a compact and expressive latent space; as existing works have found that the latent space can facilitate detecting similar objects by devising a mapping over the L_2 distances in the latent space, such as stochastic neighbor embedding [56]. Consequently, we train a variational auto-encoder (VAE) to formulate such a space over $\bar{g}_\zeta(\cdot)$. Specifically, given a set $\bar{\mathcal{G}} = \bigcup_{\tau_i \in \mathcal{D}} \bar{G}(\tau_i)$, where $\bar{G}(\tau_i) = \{\hat{G}_0, \hat{G}_1, \ldots, \hat{G}_T\}$, where $\hat{G}_t = \bar{g}_\zeta(s_{t-k:t})$ with $s_t \in \tau_i, t \in [0, T]$, one can train a VAE to reconstruct $\bar{\mathcal{G}}$ by maximizing the evidence lower bound (ELBO), *i.e.*,

$$\max_{\phi,\psi} \frac{1}{N_{\bar{\mathcal{G}}}} \sum_{\bar{g} \in \bar{\mathcal{G}}} \Big[\log p_\phi(\bar{g}|z) - KL(p(z)||q_\psi(z|\bar{g}_\zeta)) \Big]; \quad (7)$$

here, $p(z)$ is the latent prior following a normal Gaussian distribution, $q_\psi(z|\bar{g}_\zeta)$ is the approximated posterior that encodes $\bar{g}_\zeta$to the latent space, and $p_\phi(\bar{g}|z)$ is the decoder. Following from [31], both ϕ and ψ are neural networks that output the mean and diagonal variance of p_ϕ and q_ψ which both follow Gaussian distributions. Finally, we can leverage q_ψ to map each $\bar{g}_\zeta \in \bar{\mathcal{G}}$ to its latent representation which we use as the SG, $\hat{G}_t \sim q_\psi(\cdot|\bar{g}_\zeta)$. As $\bar{g}_\zeta$ have already captured the state value information from future steps, and considered that such knowledge has been encapsulated into the latent space, we envision SG to be able to capture long-term goals for DT to achieve, as well as provide guidance for DT during evaluation.

The training and testing stage of D2T2 are described in Section III-B and summarized in Algorithm 2.

IX. EXPERIMENTAL DETAILS

We run all our experiments on Nvidia RTX A5000 with 24GB RAM and each experiment setting is averaged over 10 trials with different random seeds.

A. Tailgate Driving

In this stochastic problem, we collect around $100k$ samples and train the RL policies with $300k$ total timesteps. Our

environment setting and baseline comparison are partly from [24]. However, note that [24] only considers one stop sign and does not investigate the data quality with crash rate. We list 3 different datasets in V. To decide a better hyper-parameter settings, we investigate the hyper-parameters and their corresponding tables with stop sign $= 50m$ as follows:

1) The discount factor γ of the mapping function in the range of [0.7, 0.95]. Note that the γ here is for steering guidance function mapping instead of the inputs for DT. In DT, the discount factor is 1.0.
2) The number of layers of the transformer model in the list of [2, 3, 4, 5]
3) The number of self-attention blocks in the list of [4, 8, 16]
4) The value of the context in the list of [3, 4, ..., 10]

TABLE VI
THE DISCOUNT FACTOR γ OF THE MAPPING FUNCTION IN THE RANGE OF $[0.7, 0.95]$

Metric	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**
Return	0.75 $\pm$0.03	0.83 $\pm$0.025	0.82 $\pm$0.01	**0.86** $\pm$0.03	0.83 $\pm$0.045	0.79 $\pm$0.02

TABLE VII
THE NUMBER OF LAYERS OF THE TRANSFORMER MODEL IN THE LIST OF $[2, 3, 4, 5]$

Metric	**2**	**3**	**4**	**5**
Return	0.83$\pm$0.035	0.84$\pm$0.019	**0.86$\pm$0.03**	0.84$\pm$0.017

TABLE VIII
THE NUMBER OF SELF-ATTENTION BLOCKS OF THE TRANSFORMER MODEL IN THE LIST OF $[4, 8, 16]$

Metric	**4**	**8**	**16**
Return	0.81$\pm$0.06	**0.86$\pm$0.03**	0.85$\pm$0.02

We conclude that the discount factor of mapping function and context length are relatively critical hyper-parameters in D2T2 for tailgate driving tasks. We list the additional hyper-parameters of transformer we choose for BC, DT, and D2T2 in Table X, and TT in Table XI.

B. Stochastic Frozenlake

FrozenLake environment is a standard toy text environment in Open AI gym with discrete action and state spaces in dimensions of 4 and 16 respectively. In Fig. 4, we show that the agent starts with **Start** point to achieve **Goal** along with **Frozen** while avoiding **Hole**. To modify it as a stochastic environment, the setup can be referred to in Section IV and [49].

Our D2T2 integrates vanilla DT with TD learning. For stochastic FrozenLake task, we follow the hyper-parameters and codebase used in [49] with the well-trained Q-learning model under the hyper-parameters in [32]. We add extra code to replace RTG with our learned value functions after transformer-based behavior cloning on steering guidance.

C. CARLA Benchmark

We evaluate our method on the CARLA NoCrash [30], [39] benchmark whose goal is to navigate in a suburban town to a desired goal waypoint from a predetermined start waypoint without crash, considering safety as a priority [6], [40] in tasks such as autonomous driving. The benchmark consists of 2 towns *Town01* and *Town02*, each with 25 different routes. We train our method D2T2 as well as all the baselines on the Town01 dataset and then evaluate them in the unseen Town02 routes.

NoCrash task is relatively more realistic and difficult compared with tailgate driving task. The state space includes various parameters such as relative heading error, distance from the target lane center, ego vehicle speed, relative distance to the leading vehicle (or max sensing range if none), speed of the leading vehicle (or max speed if none), and distance to the upcoming red light (or max sensing range if none). The reward consists of an increase with higher speed, a slight penalty for land deviations, and a huge penalty for crashes or traffic infractions. The trajectory terminates when the car crashes, incurs an infraction, times out, or reaches its destination.

Next, we evaluate our method on a modified version of the CARLA Leaderboard benchmark following the setting in [24]. Compared with NoCrash Benchmark, the policy is learned to change lanes in urban or highway situations. Since the goal is not just following the leading vehicle, 8 additional variables are included in the state space, providing the distance and speed of surrounding vehicles in each of the 4 diagonal directions.

Our D2T2 integrates vanilla DT with TD learning. We keep the general Transformer hyper-parameters consistent as [24] with 4 layers of self-attention blocks with 8 heads and 128 as an embedding size. We add extra code to replace RTG with our learned value functions after transformer-based behavior cloning on steering guidance.

D. D4RL Suites

Although our D2T2 aims to address stochastic problems, we also provide comprehensive studies on near-deterministic tasks in Gym-MuJoCo suite (Table III), AntMaze-v2 suite (Table XIII), and FrankaKitchen-v0 suite (Table XII. We do conduct experiments for Gym-MuJoCo tasks on D2T2 without learning latent representation. However, we do not list it in Table III because they are relatively less challenging compared with AntMaze and FrankaKitchen tasks so that the performance difference between with and without VAE is not significant.

On the other hand, The performance of D2T2 in AntMaze and FrankaKitchen tasks will be influenced by the dimension of the state/steering guidance space and the difficulty level of the data experience for imitation in high-dimensional

TABLE IX

THE VALUE OF THE CONTEXT k IN THE LIST OF $[3, 4, ..., 10]$

Metric	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Return	0.76±0.06	0.82±0.03	**0.86±0.03**	0.83±0.055	0.85±0.014	0.81±0.04	0.83±0.027	0.83±0.045

TABLE X

THE HYPER-PARAMETER USED FOR BC, DT, AND D2T2 FOR TAILGATE DRIVING

Hyper-parameters	Values
Discount factor	1.0
No of layers	4
No of heads	8
No of embed	16
action weight	5
reward weight	1
value weight	1
Batch Size	256
Learning Rate	0.0001

TABLE XI

THE HYPER-PARAMETER USED FOR TT FOR TAILGATE DRIVING

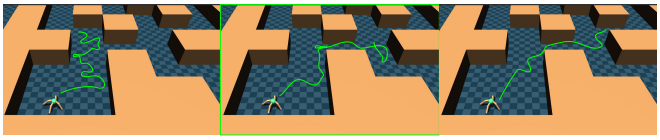
Hyper-parameters	Values
Discount factor	0.99
No of layers	4
No of heads	8
No of embed	16
action weight	5
reward weight	1
value weight	1
Batch Size	256
Learning Rate	0.0006

TABLE XII

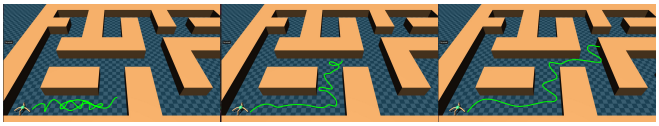
AVERAGED NORMALIZED SCORES ON FRANKAKITCHEN-V0 SUITE. WE USE THE RESULTS REPORTED FROM THE RVS PAPER FOR BC, RVS-G AND RVS-R, AND THE RESULTS FROM THE IQL PAPER FOR CQL AND IQL. WE REPORT THE MEAN AND FOR OUR METHOD OVER 10 SEEDS. NOTE THAT D2T2-n INDICATES LEARNING D2T2 WITHOUT LATENT REPRESENTATION. SINCE DT DOES NOT REPORT ITS OFFICIAL RESULTS, WE DO NOT INCLUDE IT HERE FOR FAIR COMPARISON.

Environment	**BC**	**RvS-G**	**RvS-R**	**CQL**	**IQL**	**D2T2-n**	**D2T2**
kitchen-complete-v0	**65.0**	50.2	1.5	43.8.	62.5	61.1±0.9	62.3±1.4
kitchen-partial-v0	38.0	**51.4**	0.5	49.8	46.3	43.1±2.0	47.8±2.5
kitchen-mixed-v0	51.5	**60.3**	1.1	51.0	51.0	51.2±1.3	52.2±1.9
average-Kitchen-v0	51.5	54.0	1.0	48.2	53.3	51.8	**54.1**

Fig. 4. Visualization of the stochastic FrozenLake task.



(a) Antmaze-medium-diverse-v2



(b) Antmaze-large-play-v2

Fig. 5. Evaluation among DT, D2T2-n, and D2T2. The goal of both certain environments is at the top right corner and the path is recorded in green. (a): Antmaze-medium-diverse-v2. (b): Antmaze-large-play-v2

problems. Therefore, we learn steering guidance over latent representation and report different versions of D2T2 for ablation study, where D2T2-n denotes the algorithm learning without latent representation.

FrankaKitchen tasks contain the experience that is easier to imitate [13], so we observe that D2T2-n (without latent steering guidance) is still competitive in Table XII but relatively poorly on the AntMaze tasks in Table XIII. With latent steering guidance as input to D2T2, it performs comparably to the best-performing prior method, CL, on AntMaze tasks while CL is only developed for goal-conditioned offline RL. Specifically, in Fig. 5 we show that D2T2-n is already able to get closer to the goal compared with vanilla DT while D2T2 can even extract information, resulting in better performance.

Our D2T2 integrates vanilla DT with TD learning. For

TABLE XIII

AVERAGED NORMALIZED SCORES ON ANTMAZE-V2 SUITE. WE USE THE RESULTS REPORTED FROM THE RVS PAPER FOR BC AND RVS-G, AND THE RESULTS FROM THE IQL PAPER FOR CQL AND IQL. FOR SPLT AND CL, WE USE THE RESULTS REPORTED FROM THEIR OWN PAPERS RESPECTIVELY. WE REPORT THE MEAN AND FOR OUR METHOD OVER 10 SEEDS. NOTE THAT D2T2-n INDICATES LEARNING D2T2 WITHOUT LATENT REPRESENTATION.

Environment	**BC**	**RvS-G**	**DT**	**CL**	**SPLT**	**CQL**	**IQL**	**D2T2-n**	D2T2
umaze-v2	54.6	65.4	65.6	79.8±1.4	70.5	74.0	87.5	65.2±2.7	**87.9±0.7**
umaze-diverse-v2	45.6	60.9	51.2	77.6±2.8	40.2	**84.0**	62.2	52.7±1.1	69.9±1.9
average-umaze-v2	50.1	63.2	58.4	78.7	55.4	**79.0**	74.9	59.0	78.9
medium-play-v2	0.0	58.1	1.0	72.6±2.9	25.0	61.2	71.2	42.5±2.0	**72.7±2.3**
medium-diverse-v2	0.0	67.3	0.6	71.5±1.3	15.3	53.7	70.0	40.6±0.4	**72.6±1.8**
average-medium-v2	0.0	62.7	0.8	72.1	20.2	57.5	70.6	41.6	**72.7**
large-play-v2	0.0	32.4	0.0	**48.6±4.4**	2.5	15.8	39.6	18.2±1.6	44.1±3.2
large-diverse-v2	0.0	36.9	0.2	**54.1±5.5**	2.5	14.9	47.5	27.0±2.1	50.8±4.6
average-large-v2	0.0	34.7	0.1	**51.4**	2.5	15.4	43.6	22.6	49.7
average-Antmaze-v2	16.7	53.5	19.8	**67.4**	26.0	50.6	63.0	41.0	67.1

D4RL suites, we follow the hyper-parameters and codebase used in [18] with the well-trained Q-learning model under the hyper-parameters in [32]. We add extra code to replace RTG with our learned value functions after transformer-based behavior cloning on steering guidance.