A Trajectory is Worth Three Sentences: Multimodal Transformer for Offline Reinforcement Learning

Abstract

Transformers hold tremendous promise in solving offline reinforcement learning (RL) by formulating it as a sequence modeling problem derived from language modeling (LM). Prior works using transformers usually model the sample trajectory of RL as one sequence analogous to one sentence in LM, despite the fact that each trajectory includes tokens from three diverse modalities: state, action, and reward while a sentence contains words only. Rather than taking a modality-agnostic approach which uniformly models the tokens from different modalities into one sequence, we propose a multimodal sequence modeling approach in which a trajectory (one sentence) of three modalities (state, action, reward) is disentangled into three uni-modal ones (three sentences). We investigate the correlation of different modalities during sequential decision-making and use the insights to design our multimodal transformer, named Decision Transducer (DTd). DTd outperforms prior art in offline RL on the D4RL benchmarks and enjoys better sample efficiency and algorithm flexibility.

1 INTRODUCTION

Reinforcement learning (RL) has been formulated as a sequential decision-making problem with wide applicability in various areas such as electric sport Ye et al. [2020], Vinyals et al. [2019], robotics [Zeng et al., 2021, Agarwal et al., 2022], and self-driving [Bojarski et al., 2016, Chen et al., 2020]. In numerous cases, collecting experience or data through online interactions can be infeasible or expensive. To address the data acquisition challenge, offline RL, or called batch RL, seeks to learn an optimal policy by leveraging only a pre-collected dataset, without additional exploration Fujimoto et al. [2019], Kumar et al. [2019], Wu et al. [2019]. While a wide literature exists, it remains unclear what architecture could most efficiently learn from the pre-collected history dataset.

Recently, the transformer architecture [Vaswani et al., 2017] has achieved remarkable success in language modeling (LM) by scaling up the model size, data size, etc [Radford et al., 2018, Kaplan et al., 2020]. As opposed to other architectures (e.g., CNN, RNN, MLPs), transforms have distinct advantages in dealing with sequential data (i.e., sentences) in LM. In light of this, as well as the fact that the offline RL data are also sequential, many works formulate offline RL as a sequence modeling problem to enable the use of advanced transformers [Radford et al., 2019, Chen et al., 2021, Janner et al., 2021]. In this way, offline RL methods using the sequence modeling achieved competitive results compared to classical methods (e.g., Q-learning, actor-critic) [Konan et al., 2022, Yamagata et al., 2022, Sudhakaran and Risi, Wang et al., 2022, Xu et al., 2022].

In spite of the similarities between offline RL and LM, their sequential data have inherent differences in terms of modality, which have been overlooked in prior works [Janner et al., 2021, Chen et al., 2021]. Specifically, the transformers in LM implicitly regard a sentence as a sequence consisting of unimodal inputs (i.e., words) with one consistent modality. In contrast, offline RL involves a multimodal sequence that includes three distinct modalities: states, actions, and rewards (returns). We evaluate the interactions/correlations between the modalities quantified by the normalized attention maps, shown in Fig. 1 (the details of the figure are left to Sec. 4.2). Fig. 1 shows that the three modalities have different behaviors and effects that interact with each other, which supports the notion that each modality requires a tailored architecture.

Nevertheless, prior offline RL methods relying on transformers directly follow the unimodal assumption of LM, regardless of the fact that offline RL samples are multimodal, let alone designing diverse architectures for each modality. In addition, in robotics, multimodality has shown potential

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

in transformers [Brohan et al., 2022, Lynch et al., 2022], where the sequential inputs involving observations, actions, and goals are regarded as multimodal sequences (see more related works in Sec. 2). Motivated by the aforementioned concerns — **1) multimodality matches the nature of the offline RL input data; 2) multimodality already brings benefits in the transformers for other areas**, we promote the use of transformers with multimodality in offline RL and test the following hypothesis:

If Offline RL is formulated as a sequence modeling task, then it is beneficial to solve it with a multimodal approach.

Offline RL via Sequence Modeling, Multimodal?

In this work, we explicitly exploit multimodality in transformer architecture design for offline RL. Towards disciplined architecture design, we first quantify the interactions and correlations between modalities by attention maps. In particular, we resort to Decision Transformer (DT) [Chen et al., 2021], a popular offline RL transformer, to discover the essential interactions between modalities during sequential decision-making. Armed with the findings, we refer to the critical levels of these modality interactions discovered by DT to design our multimodal transformer for better performance. Our main contributions are as follows:

1. To achieve a suitable multimodal architecture design, we first quantify the cross-modal and intra-modal interactions in decision-making sequences, by analyzing the attention scores of the offline RL transformer DT (Section 4.2).
2. Taking advantage of the multimodal quantification, we propose Decision Transducer (DTd) [1], a multimodal model-free transformer for offline RL, which outperforms prior transformer baselines and enjoys sample efficiency on the D4RL [Fu et al., 2020] benchmarks.
3. We show that the proposed DTd is more efficient in leveraging various types of goals including target return, 2D goal location, or a targeted state value compared to prior work (e.g., DT), resulting from that it explicitly models each input modality including the goal modality.

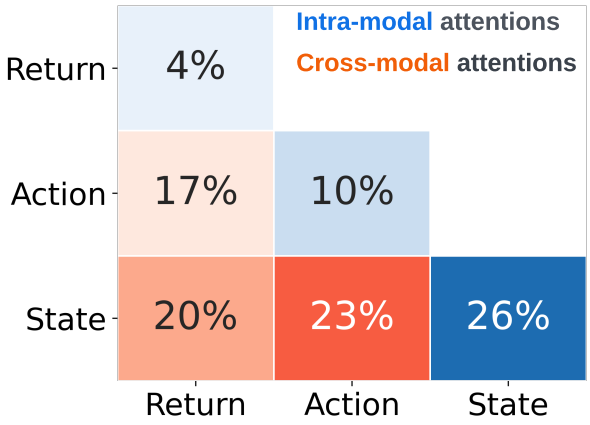

Figure 1: **Importance of Modality Interactions**. On the hopper domain with medium-expert dataset [Fu et al., 2020], scores with respect to different types of modality interactions from Decision Transformer (DT) [Chen et al., 2021] are aggregated across one episode. After normalizing scores into percentages, we found that DT generally pays more attention to cross-modal (orange) interactions (60%) compared to intra-modal (blue) interactions (40%). Therefore, we design a multimodal transformer to exploit the modality importance discovered by DT (concise findings and how to apply them in Section 4.2) to enable more effective multimodal sequence modeling.

2 RELATED WORK

Transformers for Offline RL. Popular model-free transformer DT [Chen et al., 2021] approaches the offline RL via return-conditioned sequence modeling and achieves promising results on D4RL benchmark [Fu et al., 2020]. Following DT's formulation, ConDT [Konan et al., 2022] introduces a contrastive objective for DT's embedding layers to enable more discriminative representation. Other follow-up works of DT have also explored designs that do not rely on return built from reward signals but leverage future state distribution for hindsight information matching [Sudhakaran and Risi] or value function [Yamagata et al., 2022] to generalize DT to sparse reward scenarios. Popular model-based transformer Trajectory Transformer (TT) [Janner et al., 2021] fully leverages the power of autoregressive sequence modeling to model the entire trajectory by learning policy and also a dynamic model. Such an ability could not only facilitate look-ahead planning during test time but also be useful when one wants to bootstrap simulated data from TT to improve data coverage [Wang et al., 2022].

Multimodal Transformers for Robotics. Robotics transformers usually adopt multimodal designs because the input involves complicated observation from visual sensors and sophisticated action representation of the robot. Besides, the agent may be given abstract goals (e.g. language instruction) [Shridhar et al., 2022, Brohan et al., 2022, Pashevich et al., 2021, Guhur et al., 2022, Lynch et al., 2022] instead of a scalar one. As a result, an agent has to leverage multimodal architecture to process the observation, actions, and language instructions that belong to different modalities.

[1] DTd doesn't have a "Transducer" objective, which is a sequence alignment objective proposed by Graves [2012]. The "Transducer" here is to give credit to Transformer Transducer [Zhang et al., 2020] and its neural biasing variant [Chang et al., 2021], which motivates DTd's architecture design.

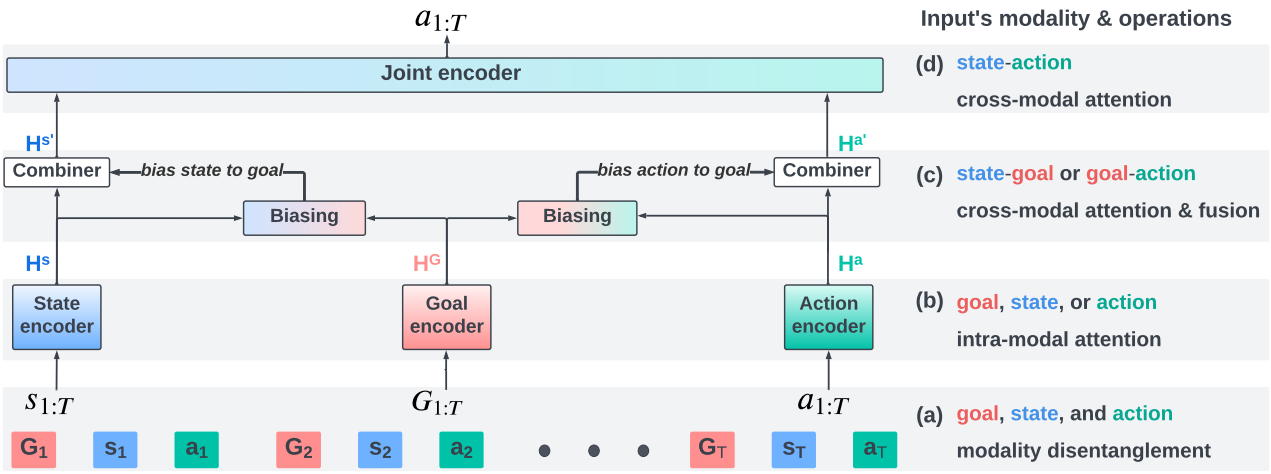

Figure 2: **Decision Transducer (DTd) Overview**. We leverage the modality importance discovered by DT's attention mechanism to develop DTd's architecture. We arrange more important modality interactions recognized by DT in higher layers of DTd and less important ones in lower layers. We list the modality involved in each layer and the operation to incorporate different modalities into the multimodal decision-making at the right side of the plot, grouped by (a), (b), (c), and (d). More connections between modality importance and DTd's choice of input modality are discussed in Section 4.2.

3 PRELIMINARIES

Offilne Reinforcement Learning (RL). RL is often formulated as sequential decision-making known as Markov Decision Process (MDP) which could be described by a tuple: $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. At each timestep t, an agent will experience state $s_t \in \mathcal{S}$ and execute an action $a_t \in \mathcal{A}$ where $\mathcal{S}$ is the state space and $\mathcal{A}$ denotes the action space. $\mathcal{T}$ is a transition function such that an agent will be exposed to a new state at timestep $t+1$ with a probability of $0 \leq \mathcal{T}(s_{t+1}|s_t, a_t) \leq 1$. A reward will be given by the reward function $r_t = \mathcal{R}(s_t, a_t)$ for each timestep. Given a discount factor $\gamma \in [0, 1]$ and a horizon length of T, a discounted return R at timestep t is defined by: $R_t = \sum_{t'=t}^{T} \gamma^{t'-1} r_{t'}$. In online RL, a policy $\pi(a_t|s_t)$ is maximized with the objective $J = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)}[\sum_t^T \gamma^{t-1} r_t]$ where the data at each timestep is in a tuple form of (s_t, a_t, r_t, s_{t+1}).

While online RL allows one to collect data (s_t, a_t, r_t, s_{t+1}) through online interactions, offline RL does not allow a policy to interact with the environment and generally learns from a fixed dataset $\mathcal{D}$ pre-collected by an inaccessible behavioral policy π_b. In this work, we train a policy π on a dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ with N number of pre-collected trajectories. Each trajectory is the result of the interaction between π_b and the environment in the form of $\tau_i = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$.

4 DECISION TRANSDUCER

We first highlight the difference between our multimodal formulation and our predecessor DT, followed by summarizing the modality importance discovered through analyzing DT'a attention, and finally explain how does DTd leverage DT's findings to enable more effective decision-making.

4.1 MULTIMODAL SEQUENCE MODELING

According to the sequence modeling formulation proposed by DT [Chen et al. [2021]], each trajectory τ of length T within offline dataset $\mathcal{D}$ will be transformed into a sequence of $\tau = (R_1, s_1, a_1, ..., R_T, s_T, a_T)$ where $R_t = \sum_{t'=t}^{T} r_{t'}$ is a un-discounted return (known as returns-to-go in [Chen et al. [2021]]). Given the similarity between sequence modeling and language modeling (LM), the objective of a model-free decision-making transformer such as DT is autoregressive: $logP_\theta(a_t|\tau_{<t})$ where $\tau_{<t} = (R_1, s_1, a_1, ..., R_t, s_t)$.

However, a trajectory is inherently multimodal including states, actions, and returns. To better aligns with the multimodality within the input space, our formulation is different from DT in a way that a trajectory is viewed as three sequences including state sequence $s_{1:T}$, action sequence $a_{1:T}$, and return sequence $R_{1:T}$. Each uni-modal sequence will be processed separately and fused selectively to facilitate multimodal decision-making. Our objective is still autoregressive: $logP_\theta(a_t|R_{\leq t}, s_{\leq t}, a_{<t})$.

4.2 ARCHITECTURE DESIGNS

To design the multimodality architecture, we first analyze the correlations between diverse modalities with the aid of DT'a attention along sequential decision-making. Specifically, we trained a DT with medium-expert data [Fu et al., 2020] until it achieved expert performance and logged its attention maps at each prediction. Since DT takes a tri-modal trajectory including returns (accumulated rewards

from present to future), states, and actions, there are 9 types of interactions within the attention map among modalities, shown in Fig. 1. After aggregating symmetric cross-modal attention scores (e.g. return-state, state-return) together, we narrow down the scope of analysis from 9 to 6 and notice that DT pays more attention to cross-modal interactions (orange colorbar in Figure 1) than intra-modal interactions (blue colorbar in Figure 1).

We list the findings as follows:

1. State-state interaction is more important (26%) than other intra-modal interactions during decision-making.
2. Cross-modal interactions is generally more important (60%) than intra-modal interactions (40%).
3. Different cross-modal interactions are weighted differently in transformer's decision-making.
   - 3.1 State-action interaction is the most salient cross-modal interaction (23%).
   - 3.2 Return-state (20%) and return-action (17%) also play important roles in decision-making.

According to finding 1, intra-modal state-state interaction is heavily considered by DT's decision-making. Thus, we disentangle the tri-modal input into 3 uni-modal sequences so that the state-state intra-modal interaction is explicitly encouraged by separating it from other modalities (shown as (a,b) in Figure 2). Then, we reflect finding 2 by arranging more important cross-modal interactions after the intra-modal interactions (shown as (c,d) VS. (b) in Figure 2) so that the cross-modal interactions could benefit from the refined representation provided by intra-modal interactions at early stages. Specifically, DTd introduces cross-modal interactions at different layers by placing the most fundamental cross-modal interaction closest to the output (finding 3.1) and placing others in the middle (finding 3.2) so that the most fundamental interaction will benefit from the low-level cross-modal interactions (shown as (c,d) in Figure 2).

Putting all the high-level intuitions together, DTd predicts actions following the procedure below:

1. **Modality encoders** refine uni-modal representations via intra-modal interactions after applying modality disentanglement to the input trajectory (finding 1).
2. **Biasing layer** learns bias-to-goal for each non-goal modality by applying cross-modal attention to the goal and non-goal modalities (finding 2 and 3.2).
3. **Combiner** reminds the transformer about its goal by combining the bias-to-goal into the decision-making via additive fusion (finding 2 and 3.2).
4. **Joint Encoder** will fuse the most fundamental modalities for the final decision-making (finding 3.1).

The overview of the Decision Transducer is shown in Figure 2. Data pre-processing and each component of the model will be explained in the following subsections.

Data Pre-Processing. We follow pre-processing and formulation proposed by Chen et al. [2021]. However, we denote a tri-modal trajectory in a more general way $\tau = (G_1, s_1, a_1, ..., G_T, s_T, a_T)$ which we replace R_t by G_t, denoting the desired outcome or goal of the task. In the reward-dense setting, $G_t = R_t$ but G_t could also become a goal position or learned state-value function in a sparse-reward setting. Specifically, we first disentangle the tri-modal trajectory of length T into three uni-modal sequences.

$$G_{1:T}, s_{1:T}, a_{1:T} = \tau \tag{1}$$

Then, we let modality-specific embedding layers transform tokens from different modalities into the same dimension and add time embedding E^τ with respect to the time horizon of the trajectory and modality embedding E^G, E^s, or E^a to encourage DTd to be aware of the multimodal input.

$$\begin{aligned} h^G &= f^G_{emb}(G_{1:T}) + E^\tau + E^G \\ h^s &= f^s_{emb}(s_{1:T}) + E^\tau + E^s \\ h^a &= f^a_{emb}(a_{1:T}) + E^\tau + E^a \end{aligned} \tag{2}$$

Modality Encoder. After tokens from different modalities are embedded to the same dimension, we apply modality-specific encoders f_{enc} to refine representation.

$$H^G, H^s, H^a = f^G_{enc}(h^G), f^s_{enc}(h^s), f^a_{enc}(h^a) \tag{3}$$

Each f_{enc} is a 3-layer transformer encoder proposed by Vaswani et al. [2017] but with layer-norm applied before the self-attention. Due to the sequence modeling formulation similar to Chen et al. [2021], DTd is autoregressive such that f_{enc} takes a causal attention mask (i.e. temporal transformer) to avoid leaking future information to the model.

Biasing. After refining representation and encouraging intra-modal interactions with f_{enc}, DTd learns bias-to-goal to bias high-level decision-making towards the task's goal. Specifically, A cross-attention layer Vaswani et al. [2017] is applied on 2 different modalities where keys and values come from the goal modality (denotes as H^G) in Figure 2) and queries come from the modalities need to be biased such as states and actions (H^s, H^a in Figure 2).

As shown in the Equation (4), keys and values are obtained from the goal modality (H^G) by applying trainable matrices f_k, f_v and queries comes from a non-goal modality such as states H^s and actions H^a by applying trainable matrix f_q.

$$\begin{aligned} Q^1, K^1, V^1 &= f_{q_1}(H^s), f_{k_1}(H^G), f_{v_1}(H^G) \\ Q^2, K^2, V^2 &= f_{q_2}(H^a), f_{k_2}(H^G), f_{v_2}(H^G) \end{aligned} \tag{4}$$

Finally, the bias-to-goal considering goal and non-goal modalities (state or action) will be learned by applying the attention layer on the learned queries, keys, and values as

shown in Equation (5). The modalities we choose to apply cross-modal attention is a reflection of finding 3.2 in 4.2.

$$H_s^G = \text{Attention}(Q^1, K^1, V^1)$$
$$H_a^G = \text{Attention}(Q^2, K^2, V^2) \quad (5)$$

Combiner. The next step is to fuse the bias-to-goal into the non-goal modality so that the action prediction is biased towards the goal of the task. DTd leverages additive fusion within Combiner to fulfill this requirement.

As shown in Equation (6), the additive fusion for each modality (H^s, H^a) is implemented by projecting the representation and its bias-to-goal (H_s^G, H_a^G) into a new hidden space by expanding their original dimension 2 times with linear layers $\mathrm{W}_1, \mathrm{W}_2$ and followed by an activation function GELU [Hendrycks and Gimpel, 2016]. Lastly, we project the representation back to the original dimension of the model hidden state with another linear layer W_3.

$$H^{s'} = \text{GELU}(\mathrm{W}_1^s H^s + \mathrm{W}_2^s H_s^G)\mathrm{W}_3^s$$
$$H^{a'} = \text{GELU}(\mathrm{W}_1^a H^a + \mathrm{W}_2^a H_a^G)\mathrm{W}_3^a \quad (6)$$

The reason to apply additive fusion instead of a more complicated multiplicative fusion [Jayakumar et al., 2020] is that adding information has become the widely-adopted practice to remind transformers about something important. For example, a standard practice to help the transformer to be aware of positional information within a sequential input is by adding positional embedding [Vaswani et al., 2017]. Similarly, to help a transformer to distinguish different modalities within a sequential input, modality-type embedding is usually added to the input [Bao et al., 2021, Kim et al., 2021]. In our case, to remind high-level DTd layers about the goal of the task, we choose to fuse the bias-to-goal into the representation with an additive operation.

Joint Encoder. We decide to let biased states $H^{s'}$ and biased actions $H^{a'}$ interact before DTd makes a decision (finding 3.1 in findings 4.2). Specifically, we first interleave 2 sequences of length T from 2 combiners into 1 sequence:

$$H^{joint} = \{(H_t^{s'}, H_t^{a'})\}_{t=1}^{T} \quad (7)$$

Secondly, a 1-layer temporal transformer encoder f_{joint_enc} similar to f_{enc} will be applied on the joint representation H^{joint} to encourage state-action cross-modal interaction.

$$H^{joint_enc} = f_{joint_enc}(H^{joint}) \quad (8)$$

Finally, the action prediction is made on the representation belonging to states with a prediction head f_{pred}.

$$\{(H_t^{s''}, H_t^{a''})\}_{t=1}^{T} = H^{joint_enc}$$
$$a_{1:T} = f_{pred}(H_{1:T}^{s''}) \quad (9)$$

We provide an architecture comparison table in supplement materials to highlight the difference between DTd and DT in terms of architecture design. In short, DT's decision backbone is the joint encoder of DTd but with a tri-modal sequential input instead of a bi-modal one. Besides, DTd attaches several components before the joint encoder to reflect the multimodal nature of a trajectory. In Section 5.3, we found these components are crucial for good performance.

5 EVALUATIONS AND DISCUSSIONS

In this section, we evaluate the overall performance of DTd on the offline RL benchmark D4RL [Fu et al., 2020] and analyze from the following perspectives:

1. Effectiveness of DTd compared to DT when leveraging different goals such as state value and goal position.
2. The necessity of different cross-modal interactions within the DTd to achieve good performance.
3. Pros and cons brought by the explicitly modeled goal after the modality disentanglement.

5.1 OVERALL PERFORMANCE

Datasets. We choose MuJoCo locomotion tasks including hopper, halfcheetah, and walker2d for evaluation. For each task, we experiment with three datasets (medium-expert, medium, medium-replay) in the D4RL benchmark [Fu et al., 2020] that are different in data collections and sizes.

Baselines. We compare with baselines including behavior cloning (BC, scores taken from Chen et al. [2021]), and state-of-the-art Temporal Difference (TD) method CQL[Kumar et al., 2020], IQL [Kostrikov et al., 2021] (scores taken from [Kostrikov et al., 2021]). Since our formulation is similar to DT but from a multimodal perspective, we select DT Chen et al. [2021] as the model-free transformer baseline. We also include planning-based method (highlighted with * in Table 1) including Trajectory Transformer (TT) [Janner et al., 2021] and Diffuser [Janner et al., 2022] to demonstrate the competitiveness of DTd as a multimodal model-free transformer.

DTd is more sophisticated than DT due to the multimodal design and has about 2.5M parameters whereas the original DT has about 0.7M parameters. Therefore, in order to provide a fair comparison between DT and DTd, we present DT-large, a variant of DT with larger amount of parameters. While DT in Table 1 refers to the DT with the original hyper-parameters evaluated with our protocol, DT-large is a DT with about 2.4M parameters. Hyper-parameters of DTd and DT-large are provided in the supplementary material.

All transformer-based approaches, including DT, DT-large, DTd, and TT, are evaluated by reporting average and standard deviation (std) across four runs, where each run has a different training seed and is evaluated with three different seeds. Scores are normalized by the performance of expert

Table 1: **D4RL Locomotion Performance**. We evaluate DTd and other offline RL transformers (DT, TT) by reporting the mean and standard deviation of normalized scores across 12 seeds (4 independent models and 3 evaluations for each). DTd achieves competitive results across environments and datasets on average. Note that TT and Diffuser require to forward the model multiple times to plan ahead for competitive performance while DTd requires 1 forward pass only without planning.

Dataset	Environment	No Planning						Planning*	
		BC	CQL	IQL	DT	DT-large	DTd (Ours)	TT*	Diffuser*
Medium-Replay	Hopper	27.6	95.0	94.7	55.2±18.4	75.9±4.6	91.2±6.8	82.6±6.9	**96.8**
	Walker2d	36.9	77.2	73.9	59.2±12.8	62.4±11.6	**81.8±3.0**	71.5±10.9	61.2
	HalfCheetah	4.3	**45.5**	44.2	33.3±3.1	33.6±4.6	41.4±0.8	44.3±1.3	42.2
Medium	Hopper	63.9	58.5	66.3	**67.8±5.8**	62.9±11.0	57.4±4.2	60.0±5.1	58.5
	Walker2d	77.3	72.5	78.3	77.8±4.4	61.7±13.0	78.8±3.8	70.4±20.2	**79.7**
	HalfCheetah	43.1	44.0	**47.4**	42.9±0.4	42.5±0.4	42.7±0.3	46.2±1.4	44.2
Medium-Expert	Hopper	79.6	105.3	91.5	110.6±1.7	95.2±16.0	**112.5±1.2**	82.2±16.8	107.2
	Walker2d	36.6	108.8	**109.6**	100.6±10.8	100.3±12.0	109.0±0.4	105.2±3.52	108.4
	HalfCheetah	59.9	91.6	86.7	83.2±3.0	77.7±9.0	**92.1±0.7**	90.2±7.2	79.8
Average		47.7	77.6	77.0	70.1	68.0	**78.5**	72.5	75.3

Table 2: **Leveraging Diverse Types of Goals.** DTd could more effectively leverage state value (**-goal**) or goal position (**-V**) in the D4RL AntMaze domain compared to DT when returns based on dense rewards is not available.

Dataset	DT-goal	DTd-goal (Ours)	DT-V	DTd-V (Ours)
Umaze-v0	**67.5 ± 18.0**	55.0± 21.0	**75.0 ± 15.0**	67.5 ± 15.0
Umaze-diverse	**67.5 ± 16.4**	57.5 ± 14.8	60.0 ± 18.7	**62.5 ± 21.7**
Medium-play	0.0 ± 0.0	**22.5 ± 11.0**	10.0 ± 7.1	**40.0 ± 8.2**
Medium-diverse	0.0 ± 0.0	**32.5 ± 13.0**	15.0 ± 15.0	**57.5 ± 8.3**
Average	33.8	**41.9**	40	**56.9**

and random policy according to the instruction from D4RL [Fu et al., 2020]. During the evaluation, DTd uses the same initial R_t as the DT implemented by Chen et al. [2021].

Overall Performance. As shown in Table 1, DTd outperforms its modality-agnostic predecessor DT and other methods, including TD and planning-based methods. Note that scaling DT's parameters up to 2.5M (DT-large) does not help DT to fill the performance gap compared to a multi-modal architecture of DTd, which confirms our hypothesis that solving the decision-making problem via sequence modeling benefits from a multimodal approach.

5.2 EFFECTIVENESS WITH DIVERSE GOALS

To evaluate whether DTd could effectively leverage goals other than R_t, we choose the AntMaze navigation task from D4RL [Fu et al., 2020] and challenge DTd with other types of goal including state value, and goal position. In AntMaze, an agent only receives a reward of 1 if it reaches the goal position within the maze, which makes the return sparse and less useful to prompt the model for action sequence generation. Therefore, leveraging other types of goals in AntMaze becomes critical for good performance.

State Value as G_t. As a first step, we evaluate the ability of DT and DTd to leverage state value by training an IQL on each dataset and using its state value to represent the task's goal. This concept is similar to the Q-function guided planning of TT [Janner et al., 2021] such that the state value specifies the desired outcome (goal) of the model.

Goal Position as G_t. In addition to using a state value that provides a return-like scalar as G_t, we further challenge DTd and DT to condition the goal position of the maze. Specifically, G_t becomes the 2D position of the agent at timestep t concatenated with the goal position. To encourage useful hidden space, we ask DT and DTd to predict the 2D position (waypoint) 3 steps away from the current 2D position as an auxiliary task. For DT, this is implemented by using an extra prediction head to predict waypoints based on G_t's representation from the last layer. For DTd, we apply the prediction head right above the modality encoder for the goal representation (not the joint encoder).

Results. We show the results in Table 2. The mean and std of the success rate across 4 runs are reported where each run is trained with different training seeds and evaluated with 10 different seeds. All methods are evaluated on 2 types of environment (Umaze, medium) with 3 types of datasets (Umaze-v0, play, diverse) from the D4RL. The DT-large and DTd variants with a value function have a suffix of **V**. The variant of DT-large and DTd with auxiliary waypoint prediction and goal position as input has a suffix of **goal**.

When we replace G_t with the concatenation of goal position and 2D position, only DTd-goal can reach the goal position and gain reward in the medium domain across 2 types of datasets, whereas DT-goal fails to solve the medium domain entirely. We believe the advantage of DTd is because of its architecture, which allows it to learn future-related (3 steps

Table 3: **Ablation Studies on DTd**. We test DTd variants (DTd-zero/left/right) by ablating Biasing and Combiner at each side of DTd to reduce types of cross-modal interactions involved in decision-making. We find that cross-modal interactions are complementary to each other (DTd strictly outperforms DTd-left/right). However, blindly encouraging all intra/cross-modal interactions at every layer (DT-large) could be worse than strategically encouraging intra/cross-modal interactions at different layers (Dtd-left) with respect to the interaction importance. Notice that if we remove goal modality entirely from decision-making (DTd-zero), DTd becomes a transformer-based behavior cloning agent with poor performance.

Dataset	Environment	BC	DT-large	DTd-zero	DTd-left	DTd-right	DTd
Medium-Expert	Hopper	79.6	95.2±16.0	89.8±16.8	108.3±6.0	109.4±4.4	**112.5±1.2**
	Walker2d	36.6	100.3±12.0	107.7±0.4	108.8±0.2	108.1±0.3	**109.0±0.4**
	HalfCheetah	59.9	77.7±9.0	58.8±0.6	91.2±0.8	91.9±0.5	**92.1±0.7**
Medium-Replay	Hopper	27.6	75.9±4.6	16.8±2.8	81.6±5.1	33.3±25.6	**91.2±6.8**
	Walker2d	36.9	62.4±11.6	34.0±16.4	44.8±30.5	20.6±12.0	**81.8±3.0**
	HalfCheetah	4.3	33.6±4.6	31.3±9.8	36.1±9.7	38.4±4.9	**41.4±0.8**
Average		41.3	74.2	56.4	78.4	67.0	**88.0**

away waypoint) latent representation to affect the high-level decision-making while DT-goal is only able to apply such an auxiliary task at the last layer of the model which affects the latent representation ineffectively.

In Umaze tasks, DTd does not seem to offer any obvious advantages over DT. Umaze domains require a trivial U-shape solution in order to reach a goal position from a starting point, whereas DTd models require a waypoint prediction within the model (on top of the goal encoder) to affect latent representations, which may be overkill.

In the setting where the goal is represented by the state value, both DT and DTd are capable of solving the problem, but DTd-V shows better performance on average across environments and datasets. In our opinion, credit should be given to the explicit representation of goals, which prompts the model to recall action sequences in an effective manner.

5.3 NECESSITY OF MODALITY INTERACTIONS

In DTd, the biasing and combiner layers introduce different types of cross-modal interactions, including state-reward interaction and action-reward interactions. We conducted ablation studies to better understand the role of those cross-modal interactions in multimodal decision-making and show the results in Table 3.

We provide 3 variants of DTd in this ablation study:

1. **DTd-zero** has all biasing layers, combiner layers, and goal encoder removed. It has bi-modal inputs, including states and actions.
2. **DTd-left** only applies biasing and combiner layers for state representation. All modality encoders exist.
3. **DTd-right** only applies biasing and combiner layers for action representation. All modality encoders exist.
4. **DTd** is the complete model drawn in the Figure 2.

The score reported in Table 3 follows the same protocol as in Table 1. We selected the medium-expert dataset where DTd yields the best result, since ablation with important designs may result in significant performance reductions, which may help us understand the importance of cross-modal interactions introduced by different components. Additionally, we include medium-replay dataset for ablation. A removal of Biasing-Combiner at either end of DTd should result in a more significant degradation in performance compared with medium-expert, as DTd heavily relies on bias-to-goal from Biasing-Combiner to recall suitable trajectory sequences within a diverse dataset to fullfil its goal, and losing it will have a major impact on its performance.

Table 3 shows that DTd-zero performs worse than DTd when expert data is available (medium-expert), and significantly worse when data of varying quality is available (medium-replay). This observation illustrates one of the key differences between behavior cloning and return-conditioned sequence modeling. The return prompts DTd to recall action sequences that might be effective in achieving the goal. Removing the goal modeling from DTd (i.e. DTd-zero) makes it similar to naive behavior cloning.

When we compare DTd-left and DTd-right to DT-large as shown in Table 3, both variants of DTd without sufficient cross-modal interactions could outperform DT occasionally. This observation is interesting because both DTd-left and DTd-right are restricted to a limited number of cross-modal interactions between the three modalities yet occasionally outperform DT which leverages all possible intra-modal and cross-modal interactions by self-attention at every layer. Therefore, we argue that modality-agnostic design (e.g. DT) introduces computation overhead by creating ineffective cross-modal interactions via self-attention at every layer while receiving marginal benefits in terms of performance.

As we expected, leveraging all necessary cross-modal interactions (DTd) leads to the best performance compared to leveraging representations derived from restricted cross-

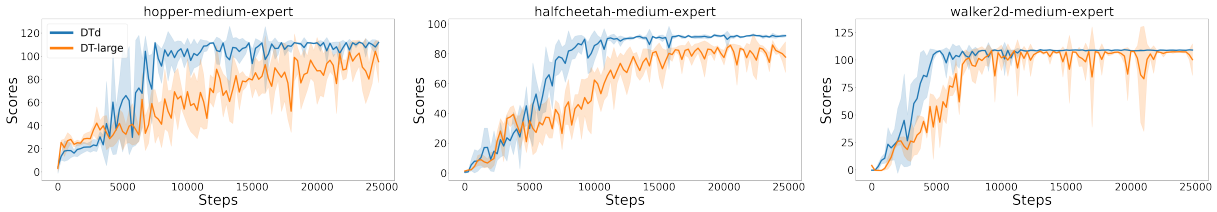

Figure 3: **Sample efficiency**. To reach DT's performance, DTd only requires 50% or less amount of gradient steps. In the end, DTd achieves not only better performance but also smaller variance across multiple runs and many evaluations.

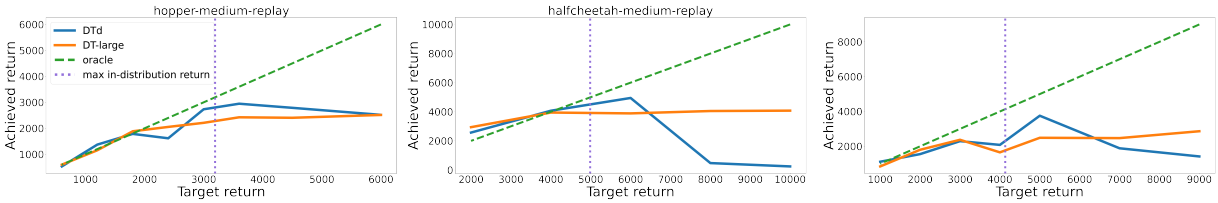

Figure 4: **Robustness**. We test DTd's robustness by probing it with target return (goal) at tests time ranging from in-distribution return to out-of-distribution (OOD) return (taget return higher than max return logged in dataset). Notice that if target return is no more than 1.2X of max return logged in dataset, DT and DTd could both achieve extrapolation. However, DTd is more vulnerable to unrealistically target return that are 1.2X larger than the max trajectory return logged within the dataset since the negative effect created by OOD return will be cascaded into the final decision-making after additive fusion.

modal interactions (DTd-left, DTd-right). Additionally, since DTd is strictly better than DTd-left and DTd-right, we believe that return-state and return-action interactions are complementary in multimodal decision-making.

5.4 PROS & CONS OF DISENTANGLEMENT

On the MuJoCo locomotion task where high equality trajectories are available, DTd is more sample efficient than DT. In Figure 3, we show the normalized score of DTd and DT-large during 100 evaluations with medium-expert datasets following the protocol in Section 5.1. We notice that DTd achieves the best performance by consuming at least 50% less gradient steps and essentially achieving a higher average score with smaller variance across different runs. This is likely due to the disentanglement at the input because it enables DTd to only include necessary cross-modal interaction at the later phase of representation learning so that there's less confusion in the decision-making within the last layer of the model. The sample efficiency of DTd is less evident when the dataset contains trajectories of variable quality (medium replay) or suboptimal (medium), the evaluation curves of which are in the supplementary material.

The disadvantage of modality disentanglement is that it is more sensitive to out-of-distribution goals than DT. We use dataset with the most diverse returns (medium-replay) to train DTd and DT-large on MuJoCo locomotion tasks, and evaluate both models based on results ranging from in-distribution returns to out-of-distribution returns. We report the raw scores averaged across four runs. As shown in Figure 4, DTd experiences a steep performance drop if the target return specified by the user at test time is too unrealistic to reach. Such a negative result is likely to be a side effect of the disentanglement of modalities. DT is robust to OOD return because it mainly makes the multimodal decision based on state-state and state-action interactions, contributing 49% of the attention weights (26% + 23% in Figure 1). In contrast, DTd explicitly learned a representation of return and fused it with other modalities to make an effective multimodal decision. During test time, if the distribution of return is significantly different from that during training, the representation of the other two modalities will also be influenced, and the negative effect created by OOD return will cascade into the prediction.

6 CONCLUSION

We attempt to advocate that offline reinforcement learning (RL) via sequence modeling benefits from a multimodal approach. While prior works such as Decision Transformer (DT) model a tri-modal trajectory as one sequence assuming every token belongs to the same modality, our multimodal transformer Decision Transducer (DTd) models a trajectory by disentangling it into three uni-modal sequences. After investigating the importance of cross-modal interactions within DT, we use the ranking of the importance discovered by DT as our heuristic for selective cross-modal fusion within DTd. Dtd not only outperforms prior transformer, temporal-difference, and diffusion-based approaches on the D4RL benchmark, but also enjoys sample efficiency during training and algorithm flexibility to leverage diverse types of goals including the return, state-value, and 2D goal position.

References

Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. *arXiv preprint arXiv:2211.07638*, 2022.

Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv preprint arXiv:2111.02358*, 2021.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Feng-Ju Chang, Jing Liu, Martin Radfar, Athanasios Mouchtaris, Maurizio Omologo, Ariya Rastrow, and Siegfried Kunzmann. Context-aware transformer transducer for speech recognition. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 503–510. IEEE, 2021.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. *arXiv preprint arXiv:2209.04899*, 2022.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Siddhant M Jayakumar, Wojciech M Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. 2020.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021.

Sachin G Konan, Esmaeil Seraj, and Matthew Gombolay. Contrastive decision transformers. In *6th Annual Conference on Robot Learning*, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*, 2022.

Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15942–15952, 2021.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022.

Shyam Sudhakaran and Sebastian Risi. Skill decision transformer. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, 2, 2019.

Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. *arXiv preprint arXiv:2206.08569*, 2022.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *International Conference on Machine Learning*, pages 24631–24645. PMLR, 2022.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. *arXiv preprint arXiv:2209.03993*, 2022.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6672–6679, 2020.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7829–7833. IEEE, 2020.