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

Yiqi Wang[1] **Mengdi Xu**[2] **Laixi Shi**[1] **Yuejie Chi**[1]

[1]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
[2]Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

Abstract

Transformers hold tremendous promise in solving offline reinforcement learning (RL) by formulating it as a sequence modeling problem inspired by language modeling (LM). Prior works using transformers model a sample (trajectory) of RL as one sequence analogous to a sequence of words (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 as 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 unimodal ones (three "sentences"). We investigate the correlation of different modalities during sequential decision-making and use the insights to design a multimodal transformer, named Decision Transducer (DTd). DTd outperforms prior art in offline RL on the conducted D4RL benchmarks and enjoys better sample efficiency and algorithm flexibility. Our code is made publicly here.

1 INTRODUCTION

Reinforcement learning (RL) has been formulated as a sequential decision-making problem with wide applicability in areas such as strategy games 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]. Often, collecting data from online interactions can be costly or risky. To address the data acquisition challenge, offline RL seeks to learn an optimal policy by leveraging a pre-collected dataset, without online interactions Fujimoto et al. [2019], Kumar et al. [2019], Wu et al. [2019]. Conventionally, offline RL is approached similarly to online RL via temporal difference (TD) learning [Sutton and Barto, 2018]. While a wide literature Fujimoto et al. [2019], Kumar et al. [2019], Wu et al. [2019] has investigated how to leverage TD learning to solve offline RL, many of these methodologies differ in their model architecture and objectives.

Recently, transformers [Vaswani et al., 2017] have achieved remarkable success in language modeling (LM) [Radford et al., 2018, Kaplan et al., 2020] by using a temporal transformer [Radford et al., 2019] to learn the distributions of concepts given sequential inputs (i.e. sentences). Since samples in offline RL (i.e., trajectories) are also sequential, it is natural to formulate offline RL as a sequence modeling problem [Chen et al., 2021, Janner et al., 2021] and leverage a temporal transformer to model the distributions of the behaviors within the pre-collected dataset, built on the architectural advances in LM. In contrast to TD methods [Fujimoto et al., 2019, Kostrikov et al., 2021] that tackle offline RL with multiple components and objectives, a single temporal transformer with a behavior cloning objective achieves surprisingly competitive and promising results [Konan et al., 2022, Yamagata et al., 2022, Sudhakaran and Risi, Wang et al., 2022, Xu et al., 2022].

However, prior works with transformers [Janner et al., 2021, Chen et al., 2021] have not examined one major difference between offline RL and LM: their sequential inputs are inherently different in terms of modality. Specifically, the transformers in LM regard a sentence as a unimodal sequence where every token belongs to one consistent modality. In contrast, offline RL includes a multimodal sequence includes three distinct modalities: state, action, and reward (return). Inspired by recent developments in robotics where a sequential input involving observations, actions, and goals are regarded as a multimodal sequence (see more detailed discussions in the related work in Section 2), we hypothesize that:

When treating offline RL as a sequence modeling task, it is beneficial to take a multimodal approach.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

Offline RL via Multimodal Sequence Modeling? Before starting to design a multimodal transformer for offline RL, we first investigate the importance of the multimodal interactions that took place within the prior transformers (e.g. the Decision Transformer (DT) [Chen et al., 2021]).

Our findings suggest that some interactions between modalities are more weighted than others (shown in Figure 1), according to the attention map of DT. Therefore, we capitalize on the important interactions of these modalities to design our multimodal transformer for better performance (a detailed discussion will be provided in Section 4.2). The main contributions are as follows:

1. To obtain heuristics for multimodal architecture design, we first quantify the cross-modal and intra-modal interactions in the sequential decision-making process (Section 4.2), by aggregating and analyzing the last-layer attention scores of DT.
2. Using the heuristics from multimodal quantification in Section 4.2, we propose a multimodal transformer called Decision Transducer (DTd)[1], which outperforms prior art and enjoys better sample efficiency on the D4RL [Fu et al., 2020] offline RL benchmarks.
3. Due to the multimodal design, DTd is more efficient and flexible in leveraging diverse types of task goals, such as long-term return, targeted physical locations, and the value function over states, with comparisons to prior arts (e.g., DT).

2 RELATED WORKS

Transformers for Offline RL. The popular model-free DT achieves promising results on the D4RL benchmark by offering return-conditioned sequence modeling for offline RL [Fu et al., 2020]. Based on DT's formulation, ConDT [Konan et al., 2022] further improves DT's performance by introducing contrastive objectives to learn more discriminative representations. Instead of return-conditioning, other works of DT replace return by state distributions [Sudhakaran and Risi] or value function [Yamagata et al., 2022] to tackle the sparse reward scenarios. Alternatively, popular model-based Trajectory Transformer (TT) [Janner et al., 2021] fully exploits the power of autoregressive sequence modeling by learning both the policy and the dynamic model simultaneously. Not only does such an ability facilitate look-ahead planning, but it can also be used to bootstrap TT simulated data to improve coverage [Wang et al., 2022].

[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.

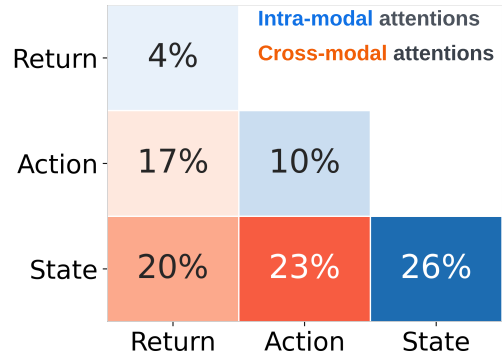

Figure 1: **Importance of Modality Interactions**. On the hopper domain with medium-expert dataset [Fu et al., 2020], attention scores with respect to different types of modality interactions from the last layer of a Decision Transformer (DT) [Chen et al., 2021] are aggregated across one episode. After normalizing scores into percentages, it is observed 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 (see-Section 4.2 for details) to enable more effective multimodal sequence modeling.

Multimodal Transformers for Robotics. Transformers for robotics usually adopt multimodal designs because the input involves complex visual observations and sophisticated robot actions. 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 return. The multimodal architecture enables agents to process observations, actions, and language instructions from different modalities.

3 PRELIMINARIES

Offilne Reinforcement Learning (RL). RL is often formulated as sequential decision-making via the framework of Markov Decision Processes (MDPs), 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: $R_t = \sum_{t'=t}^{T} \gamma^{t'-1} r_{t'}$. The goal is to find a policy $\pi(a_t|s_t)$ that maximizes the objective $J = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)}[\sum_t^T \gamma^{t-1} r_t]$.

While online RL allows one to collect data (s_t, a_t, r_t, s_{t+1})

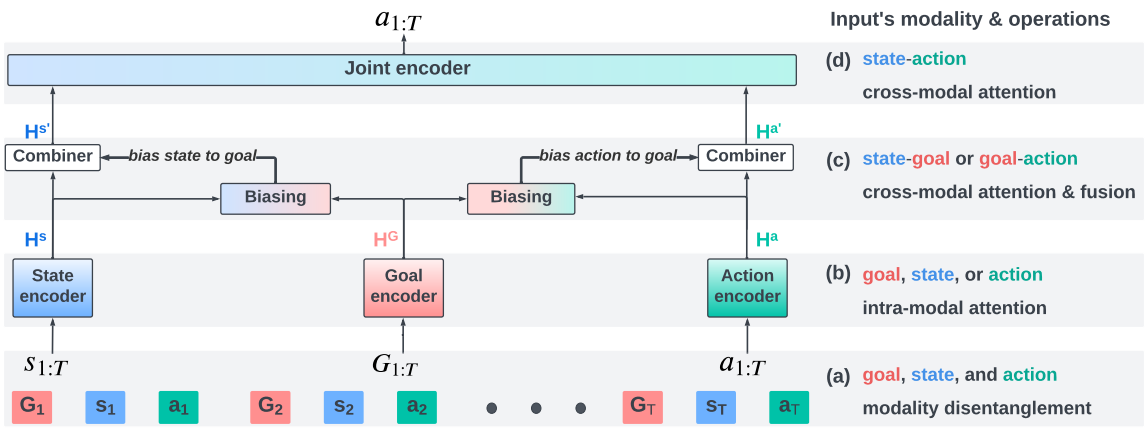

Figure 2: **Decision Transducer (DTd) Overview**. We leverage the modality importance discovered by DT's attention mechanism to develop DTd's architecture. We arrange more important modality interactions recognized by DT in higher layers of DTd and less important ones in lower layers. We list the modality 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.

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 behavior policy π_b. In this work, we train a policy π on a dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ with a number of N pre-collected trajectories. Here, each trajectory is the result of the interaction with the environment via the behavior policy π_b, in the form of $\tau_i = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$.

4 DECISION TRANSDUCER

In this section, we first compare our multimodal formulation to DT's predecessor (Section 4.1), quantify the importance of different modalities derived from DT's attention map (Section 4.2), and finally explain the connections between the results of our multimodal quantification to the design of DTd's architecture (Section 4.3).

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 an offline dataset $\mathcal{D}$ is transformed into a sequence $\tau = (R_1, s_1, a_1, ..., R_T, s_T, a_T)$ where $R_t = \sum_{t'=t}^{T} r_{t'}$ is an un-discounted return (known as return-to-go in Chen et al. [2021]). Given the similarity between sequence modeling and language modeling, a model-free decision-making transformer such as DT only needs an autoregressive objective $\log P_\theta(a_t|\tau_{<t})$ derived from LM, where $\tau_{<t} = (R_1, s_1, a_1, ..., R_t, s_t)$.

However, a trajectory is inherently multimodal including states, actions, and returns. Our formulation aligns better with the multimodality of the input space by considering a trajectory as three sequences, including state sequence $s_{1:T}$, action sequence $a_{1:T}$, and return sequence $R_{1:T}$. To facilitate multimodal decision-making, each unimodal sequence will be processed separately and fused selectively $\log P_\theta(a_t|R_{\leq t}, s_{\leq t}, a_{<t})$.

4.2 MULTIMODAL QUANTIFICATION

To guide our multimodal design, a multimodal quantification on DT was conducted on medium-expert data of hopper environment [Fu et al., 2020], where the DT has its last-layer attention maps logged and analyzed.

Within the attention map, there are 9 types of interactions between modalities since DT takes a tri-modal input including returns, states, and actions. After aggregating symmetric cross-modal attention scores (e.g. return-state, state-return) together, the scope of analysis is narrowed down from 9 to 6 types of interactions. It is noticed that DT pays more attention to cross-modal interactions (orange colorbar in Figure 1) than intra-modal interactions (blue colorbar in Figure 1). Following are our findings:

1. State-state interaction is more important (26%) than other intra-modal interactions during decision-making.
2. Cross-modal interactions are more important (60%) than intra-modal interactions (40%).
3. Different cross-modal interactions are weighted differently in DT's decision-making.
   3.1 State-action interaction is the most salient among cross-modal interactions (23%).

3.2 Return-state (20%) and return-action (17%) interactions also play important roles in decision-making.

Our high-level design heuristics is that less important interactions between modalities should be processed before important ones during representation learning, ensuring the representation learning with the most important interactions involves minimum distraction.

With our findings and heuristics above, we start by placing all intra-modal interactions before cross-modal interactions (cf. (b) before (c,d) in Figure 2) hierarchically. As the state-state interaction is identified as the most important intra-modal interaction within the multimodal input, we disentangle it from other modalities and let an ad-hoc state encoder handle it (cf. the state encoder shown in Figure 2, row (b)). The expectation is to refine state representation by intra-modal attention without distraction from other modalities. After intra-modal interactions are applied, we arrange less important cross-modal interaction with respect to our findings below more important cross-modal interaction, as illustrated in (c,d) in Figure 2. As a result, all interactions between modalities within the model are ranked with respect to their importance in multimodal decision-making.

4.3 ARCHITECTURE DESIGN OF DTD

Putting all the design heuristics together, DTd predicts actions following the procedure below:

1. **Modality encoders** refine unimodal 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 (findings 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 (findings 2 and 3.2).
4. **Joint encoder** fuses the most fundamental modalities for the final decision-making (finding 3.1).

An overview of DTd is shown in Figure 2. Data preprocessing 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 unimodal sequences as

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

Then, we let modality-specific embedding layers transform tokens from different modalities into the same dimension and add 2 types of embeddings:

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

Specifically, E^τ is the time embedding with respect to the time horizon of the trajectory, and modality embeddings — including E^G, E^s, and E^a — are used to encourage DTd to be aware of the multimodal input.

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

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

to refine representation. 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, DTd is autoregressive such that f_{enc} takes a causal attention mask (i.e., temporal transformer) to avoid leaking future information to the model similar to Chen et al. [2021].

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 to 2 different modalities where keys and values come from the goal modality (denotes as H^G in Figure 2) by applying trainable matrices f_k, f_v, and similarly, queries come from non-goal modalities need to be biased such as states and actions (H^s, H^a in Figure 2) by applying trainable matrix f_q, represented by:

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

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 below:

$$\begin{aligned} H^G_s &= \text{Attention}(Q^1, K^1, V^1), \\ H^G_a &= \text{Attention}(Q^2, K^2, V^2). \end{aligned} \tag{5}$$

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

The additive fusion for each non-goal 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, as shown below

$$\begin{aligned} H^{s'} &= \text{GELU}(\text{W}_1^s H^s + \text{W}_2^s H_s^G)\text{W}_3^s, \\ H^{a'} &= \text{GELU}(\text{W}_1^a H^a + \text{W}_2^a H_a^G)\text{W}_3^a. \end{aligned} \tag{6}$$

We expand the dimensions of H^s, H_s^G, H_a^G, and H^a 2 times with linear layers W_1, W_2, 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.

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). We first interleave 2 sequences of length T from 2 combiners into 1 sequence as

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

Secondly, a 1-layer temporal transformer named Joint net f_{joint_enc} similar to f_{enc} will be applied on the joint representation H^{joint}, leading to

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

The Joint net will encourage the most important cross-modal interaction state and action to interact with each other.

Finally, the action prediction is made on the representation belonging to states $H_t^{s''}$. In particular, $H_t^{s''}$ is disentangled from H^{joint_enc} and a prediction head f_{pred} is applied on top to predict action, summarized as follows

$$\begin{aligned} \{(H_t^{s''}, H_t^{a''})\}_{t=1}^{T} &= H^{joint_enc}, \\ a_{1:T} &= f_{pred}(H_{1:T}^{s''}). \end{aligned} \tag{9}$$

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 sequence learned from 3 unimodal sequences. 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 experimental performance of DTd over the offline RL D4RL benchmark [Fu et al., 2020] and analyze from the following perspectives:

1. Effectiveness of the proposed DTd compared to DT when leveraging different types of task goals as the input, such as long-term return, value function over states, and targeted physical positions.
2. The ablation study of the architecture components and input modalities.
3. Pros and cons brought by the explicitly modeled goal after modality disentanglement.

To facilitate the future work and reproducibility, our code is made publicly via github [2].

5.1 EXPERIMENTAL PERFORMANCE

Datasets. We conduct experiments over MuJoCo locomotion tasks including hopper, halfcheetah, and walker2d for evaluation. For each task, we use three different levels of history datasets (i.e., medium-expert, medium, and medium-replay) in the D4RL benchmark [Fu et al., 2020] that are different in data collections and sizes.

Baselines. We compare with several kinds of baselines that are widely used in offline RL. Specifically, the baselines including 1) behavior cloning (BC, scores taken from Chen et al. [2021]); 2) multiple state-of-the-art Temporal Difference (TD) methods such as CQL[Kumar et al., 2020] and IQL [Kostrikov et al., 2021] (scores taken from [Kostrikov et al., 2021]); DT Chen et al. [2021], as the model-free transformer baseline, which has a similar formulation as ours but not from a multimodal perspective, we select DT; and finally two planning-based methods including Trajectory Transformer (TT) [Janner et al., 2021] and Diffuser [Janner et al., 2022], used to evaluate the competitiveness of DTd as a multimodal model-free counterpart.

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 a larger amount of parameters. DT-large is designed to match DTd not only in terms of total parameters but also in terms of capabilities. DT-large has

[2] https://github.com/berniewang8177/Official-codebase-for-Decision-Transducer/

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). The methods reproduced by ourselves using the protocol above are highlighted with *. 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 thanks to its multimodal designs.

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 can more effectively leverage state value (**-V**) or goal position (**-goal**) in the D4RL AntMaze domain compared to DT when returns based on dense rewards are not available. The scores reported are the average across 40 evaluations (4 independent models with different training seeds and 10 evaluations for each with different evaluation seeds).

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**

4 layers (analogous to DTd 3-layer encoder + 1 Joinet net) and has 3 attention heads (analogous to 3 1-head modality encoders of DTd). We also raise the dimension of representation to increase its total 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. Detailed comparisons of hyper-parameters and training details between DT-large and DTd can be found in the supplement materials.

All transformer-based approaches, including DT, DT-large, DTd, and TT, are evaluated by reporting average and standard deviation (std) across 4 runs, where each run has a different training seed and is evaluated with 3 different seeds. Scores are normalized by the performance of expert 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]. We highlight the methods reproduced by ourselves using the protocol above with * in Table 1.

Performance results. As shown in Table 1, DTd outperforms its modality-agnostic predecessor DT and other methods, including TD and planning-based methods. While TD methods required to design different networks including actor, critic, target network and complicated objectives [Fujimoto et al., 2019, Kostrikov et al., 2021, Kumar et al., 2020], DTd only requires a uniform network architecture (stacking transformer blocks) and a simple behavior cloning objective. While a planning-based method such as TT or Diffuser requires unrolling a full model multiple times during deployment, DTd is capable to achieve model-based method performance without unrolling a dynamic model, which shows its potential application in real-time decision-making problems.

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 goals including state value, and goal position. In AntMaze, an agent receives a reward of 1 only if it reaches the goal position within the maze and 0 in most cases. Such a sparse-reward setting makes an episodic return binary and less useful to prompt the model for action sequence generation. Therefore, leveraging other types of goals effectively 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 agent on every dataset and using its state value to represent the task's goal. This concept is similar to the Q-function guided

Table 3: **Ablation on the architecture**. To justify our architecture design, we compare DTd to its variants by removing the cross-modal interactions introduced by the Biasing-Combiner layer within DTd everywhere (DTd-zero), left (DTd-left), or right (DTd-right). The most important finding is that all cross-modal interactions we selected are necessary for good performance (DTd VS. DTd-left/right). Interestingly, while DT-large has access to all cross-modal interactions, DTd-left still outperform it on average by leveraging a number of limited and but essential cross-modal interactions.

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**

Table 4: **Ablation on the modality order**. In the current DTd, state, goal (e.g. return), and action (S-G-A) are input modality from left to right. In order to meet our design heuristics, it provides states and actions as inputs to the Joint net where goals are placed in the center. Our result in Table 4 shows that any order who fails to provide the state-action interaction required by our heuristics will lead to bad model performance. Our discussion discards one order from a pair of symmetric order (e.g. S-G-A, A-G-S) since DTd is symmetric. More insights on the input order and heuristics are provided in the Section 5.4.

Dataset	Environment	S-G-A (current)	S-A-G	A-S-G
Medium-Expert	Hopper	**112.5±1.2**	5.1±1.4	5.0±1.3
	Walker2d	**109.0±0.4**	0.9±0.9	0.8±0.2
	HalfCheetah	**92.1±0.7**	2.1±0.03	2.1±0.1
Medium-Replay	Hopper	**91.2±6.8**	5.3±1.4	4.1±0.9
	Walker2d	**81.8±3.0**	1.0±0.1	1.0±0.1
	HalfCheetah	**41.4±0.8**	2.1±0.09	2.25±0.1
Average		**88.0**	2.7	2.5

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 maze (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 replacing 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. The advantage of DTd might be due to its architecture, which allows it to learn future-related (3 steps 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, affecting the latent representation less effectively.

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. Credit should be given to the explicit representation of goals, which prompts the model to recall action sequences in an effective manner.

5.3 ABLATION ON THE ARCHITECTURE

In DTd's architecture, the biasing and combiner layers selectively introduce different types of cross-modal interactions, including state-return interaction and action-return interactions suggested by our design heuristics. We conducted ablation studies on these cross-modal interactions to better understand the role of them in multimodal decision-making

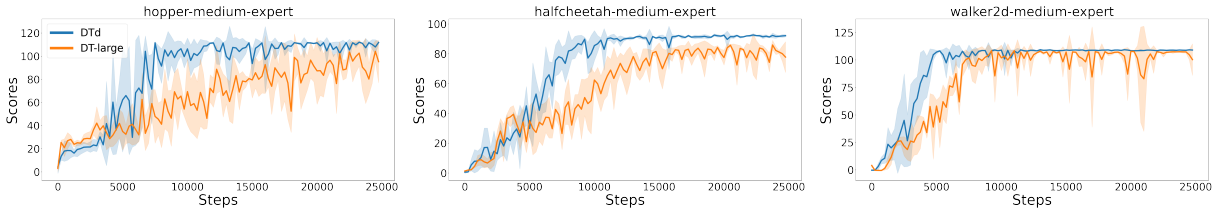

Figure 3: **Sample efficiency**. We plot the evaluation curve of our DTd against DT-large throughout the training across 4 runs. To reach DT-large's performance, DTd only requires 50% or less amount of gradient steps. In the end, DTd achieves not only better performance but also smaller variance across multiple runs and many evaluations. We provide evaluation curves for all environments in the supplement materials.

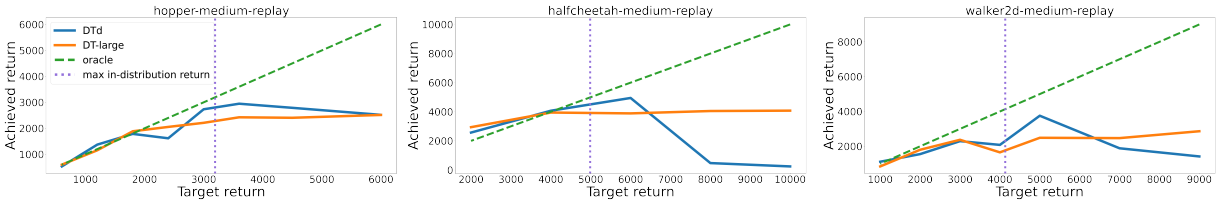

Figure 4: **Robustness**. To study whether DTd is robust to the out-of-distribution (OOD) goal at test time, we varied the target return (goal) at test time from in-distribution return to OOD return (target return higher than the max return logged in the dataset). Notice that if the target return is no more than 1.2x of the max return logged in the dataset, DT and DTd could both achieve extrapolation. However, DTd is more vulnerable to unrealistically target returns 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 the biasing layer (additive fusion).

and show the results in Table 3.

We provide 3 variants of DTd in this ablation study:

1. **DTd-zero** has biasing layers, combiner layers, and goal encoder removed. It is a simple behavior cloning model taking 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. If important designs are removed from the model, we expect to observe a performance reductions. Additionally, we include the medium-replay dataset for ablation. A removal of Biasing-Combiner at either side (DTd-left/right) of DTd should result in a more significant degradation in performance compared with medium-expert. Since DTd heavily relies on bias-to-goal from Biasing-Combiner to recall suitable trajectory to fulfill the task goal, a diverse dataset with varies goals will further verify the role of a Biasing-Combiner layer within the DTd.

Table 3 shows that DTd-zero performs worse than DTd. This observation verifies the importance of cross-modal interaction again, as aforementioned in Section 4.2. Losing cross-modal interaction from both sides of the model results in DTd-zero strictly performs worse than DTd. Since DTd-zero still includes the most important cross-modal attention (state-action) at Joinet net, it makes DTd-zero outperforms BC on average.

When we compare DTd-left and DTd-right to DT-large as shown in Table 3, we found DT-large has no significant advantages in terms of performance (DTd-left even outperforms DTd-large on average). This observation is interesting because both DTd-left and DTd-right are restricted to a limited number of cross-modal interactions yet occasionally outperform DT-large 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) creats ineffective cross-modal interactions via self-attention at every layer while receiving marginal benefits in terms of performance.

As expected, leveraging all necessary cross-modal interactions (DTd) leads to the best performance compared to leveraging representations derived from restricted cross-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 ABLATION ON THE MODALITY ORDER

In addition to justifying the heuristics behind DTd's architecture design in Table 3, we show that the design heuristics of DTd suggest the best order for DTd's modalities. In Section 4.2, state-action cross-modal interactions are prioritized over less important cross-modal interactions. Therefore, the current DTd has an input order of state, goal, and action (S-G-A) so that the Joint net takes the biased state and action with respect to the goal encoder in the center. Different from the current S-G-A setting, our ablation experiment tests DTd variants without placing goal modality in the center including S-A-G and A-S-G. As a result, the Joinet net no longer takes the state and action representation required by our heuristics. Note that our discussion below will ignore one order of a pair of symmetric order since DTd's architecture is symmetric. For example, we ignore A-G-S (i.e., S-G-A), G-A-S (i.e., S-A-G), and G-S-A (i.e., A-S-G). Specifically, we argue that an order which fails to offer a biased state and action to the Joint net will suffer from performance degradation, which is revealed in Table 4. While S-A-G offered a biased state and goal to the Joint net, A-S-G presented biased action and goal to the Joint net. Neither of them are presenting the most important cross-modal interaction to the Joint net as S-G-A does. As a result, they both have bad performance as we expected.

5.5 PROS & CONS OF DISENTANGLEMENT

On medium-expert datasets, we plot 100 evaluations of DTd and DT-large for MuJoCo locomotion tasks. DTd is more sample efficient than DT, as shown in Figure 3. In general, DTd achieves a higher average score with a smaller variance and consuming at least 50% fewer gradient steps. Since DTd includes cross-modal interactions only when necessary, confusion in decision-making is reduced due to de-entanglement at input. We found that DTd's sample efficiency is less evident when trajectories are of variable quality (medium replay) or suboptimal (medium). We provides comparison between DT-large and DTd cross all environments and datasets in the supplementary material for reader's reference.

As a consequence of modality disentanglement, DTd is more sensitive to out-of-distribution goals than DT. In MuJoCo locomotion tasks, we train and evaluate DTd and DT-large on datasets with diverse returns (medium-replay), ranging from in-distribution returns to out-of-distribution returns. The raw scores are averaged across four runs. As shown in Figure 4, when the user specifies an unrealistic target return, DTd experiences a steep performance drop. The disentanglement of modalities is likely to have led to this negative result. A multimodal decision was made by DTd by explicitly learning a representation of return and fusing it into other modalities. 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. 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 (namely, 26% + 23% in Figure 1). Since return is not the major modality involved in DT's decision-making, an OOD return has less effect on its performance.

6 CONCLUSION AND FUTURE WORK

We advocate that solving offline RL via sequence modeling may benefit from a multimodal approach. While prior works such as DT model a tri-modal trajectory as one sequence assuming every token belongs to the same modality analogous to LM, our multimodal DTd models a trajectory by disentangling it into three unimodal 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 transformers, TD learning, 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.

DTd is our first step towards more effective and efficient sequential decision-making leveraging a multimodal approach. We point out some future directions that are worth pursuing.

1. **Automating the modality-driven architecture.** Our work "hard-code" the importance of modality into the design of a multimodal architecture after the attention analysis of another model. Instead of training a model to discover the importance and leverage it with a new model, could such a process be automated and done within one model?
2. **Initializing decision-making models.** It has been shown that transformer-based RL can benefit from a pre-text task that involves text modality [Reid et al., 2022]. How should we initialize a model like DTd which has many small transformer components?

Author Contributions

Y. Wang devised the idea, conducted the experiment, and drafted the paper. M. Xu and L. Shi contributed to the experiment design and paper writing. Y. Chi supervised the entire project.

Acknowledgements

This work is supported in part by NSF via CCF-2106778.

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.

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

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*.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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.