STATE PREDICTION FOR OFFLINE REINFORCEMENT LEARNING VIA SEQUENCE-TO-SEQUENCE MODELING

Anonymous

Anonymous

ABSTRACT

Recent offline reinforcement learning methods often frame the problem as a sequence modeling task, employing a decoder-only architecture to process states, actions, and a single scalar value representing the sum of future rewards (i.e., returns). However, the distinct characteristics of these modalities, such as the non-smoothness of action sequences and the scalar nature of returns, may hinder effective modeling and optimization when using a shared architecture. In this work, we propose a divide-and-conquer strategy, the Reward-Guided Decision Translator (RGDT), that leverages an encoder-decoder architecture by casting offline reinforcement learning as a sequence-to-sequence modeling problem. Our approach foregoes action prediction in favor of next state prediction, mitigating the challenges posed by the non-smoothness of action sequences. Furthermore, our formulation enables direct conditioning of state generation on sequences of future returns, providing a more informative signal for the model. By disentangling the processing of different modalities, our approach addresses the limitations of shared decoder-only architectures. Empirical results demonstrate that our method significantly outperforms existing generative sequence modeling techniques and matches or surpasses state-of-the-art methods across a range of continuous control tasks from the D4RL benchmark.

***Index Terms*—** Offline Reinforcement Learning, Sequence Modeling, Transformer Architecture

1. INTRODUCTION

Offline reinforcement learning (RL) has emerged as a promising approach for deriving effective policies from static datasets, circumventing the necessity for online interactions with the environment. This paradigm is particularly advantageous in domains where real-world exploration is expensive, hazardous, or impractical, such as robotics, healthcare, and finance [1].

Offline RL datasets typically comprise sequences of three distinct modalities: encountered states, actions executed by a single or a mixture of policies, which may not necessarily be optimal, and scalar reward information, which can be in the form of Q-values or the sum of future rewards per timestep. This sequential nature of offline RL data makes it an ideal candidate for the application of sequence modeling techniques [2]. Recent work in the signal processing community has embraced this perspective, with [3] proposing an offline RL method based on next state supervision, [4] introducing uncertainty estimation with generative adversarial networks, and Wu et al. introducing pre-trained policy guidance in offline learning. These methods harness the capabilities of transformer models, particularly the decoder component, to directly learn policies from trajectories [5].

Transformers, initially designed for natural language processing to handle discrete input tokens, have been adapted for offline RL either by discretization-based tokenization of

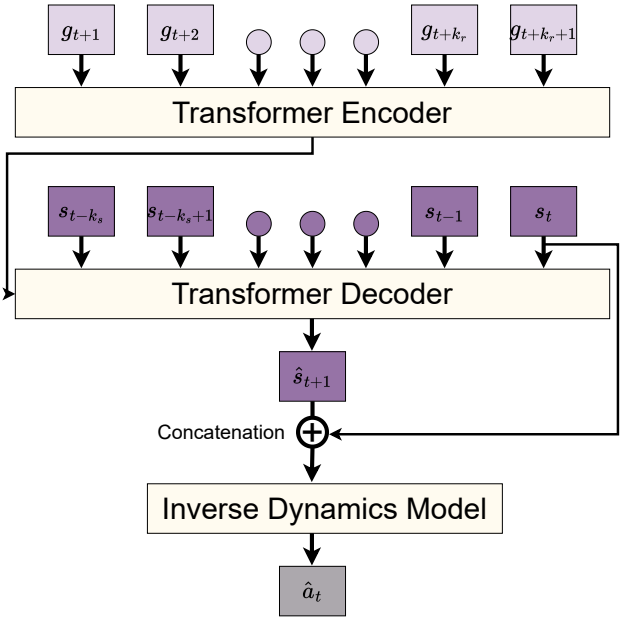


Fig. 1: RGDT framework overview. Our sequence-to-sequence architecture maps future returns to past states via an encoder-decoder structure, with actions inferred through an inverse dynamics model. This design disentangles modality processing, separately handling vector states/actions and scalar returns.

Anonymous.

979-8-3503-2411-2/25/$31.00 ©2025 IEEE

input features from each modality [6], or by a higher-level tokenization that considers only three tokens: states, actions, and returns [2]. This approach has parallels in signal processing, where Ohta et al. designed a sequential Audio Spectrogram Transformer with a memory token for real-time sound event detection, and Seraphim et al. developed structure-preserving Transformers for sequences of symmetric positive definite matrices in EEG classification.

The primary advantage of discretization is the flexibility it affords the model to learn distinct statistical properties of each feature, as each discretized value is associated with its own parameters [5]. However, this approach faces significant scalability challenges with even relatively small datasets and dimensions [6]. The higher-level modality-based tokenization offers greater scalability and simplicity, but with all transformer layers beyond the embedding layer shared across modalities, it may not adequately capture the unique characteristics of each data type. For instance, in continuous control tasks, action sequences often represent joint torques with non-smooth trajectories [7], while state sequences are governed by the environment's dynamics and exhibit smoother patterns [8].

This disparity explains why discretization, despite being modality-agnostic, often yields superior performance. The advantage of disentangling transformer components per modality has been explored extensively in signal processing applications. [9] introduced RESTAD, a Transformer-based model for time-series anomaly detection that effectively separates the processing of normal and anomalous signal patterns, while [10] employed contrastive representation learning on wireless channel-state sequences for human-orientation detection. [11] proposed "Speed," a scalable preprocessing pipeline for EEG data enabling self-supervised sequence learning, demonstrating how proper signal-domain preprocessing aids downstream representation learning. Specifically, in offline RL, there is work that explores the effect of disentangling transformer components per modality [12]. This approach hardcodes the importance of each modality into the design of a multimodal architecture after attention analysis of the decision transformer, using multiple small transformers. However, instead of training a model to discover the importance and leverage it with a mixture of models, an automated approach within a single model would be more desirable.

In this work, we develop a modality-aware sequence-based approach for offline RL that exhibits both the flexibility of disentangled parameters across different modalities and the scalability of high-level modality-specific tokenization within a single model. We adopt a sequence-to-sequence modeling framework where the model translates from one modality to another—specifically, from sequences of future returns to sequences of past states. This approach aligns with successful encoder-decoder architectures in signal processing: [13] designed CNN-Transformer architectures for audio captioning that map audio features to text descriptions, [14] presented YourMT3+, which transcribes polyphonic audio into multiple instrument tracks, and [15] proposed parameter-efficient transfer learning for Audio Spectrogram Transformers.

Our sequence-to-sequence modeling leverages the vanilla transformer architecture with an encoder-decoder structure [16], ensuring each modality has its own transformer components (see figure 1). This approach is conceptually similar to [17]'s LMCodec, which uses a causal Transformer for neural speech coding with coarse/fine token hierarchies. With our formulation, each modality has disentangled weights while still using the high-level modality tokenization employed by decision transformers. The idea of translating from one modality to another using an encoder-decoder architecture is widely common in the literature and forms the cornerstone for fields like speech-to-text, text-to-speech, and image-to-text translation. Finally, actions are predicted using an inverse dynamics model, a design choice we explain in subsequent sections. Our experimental results demonstrate that this approach substantially outperforms decoder-based counterparts and matches or surpasses state-of-the-art off-policy methods across a range of continuous control tasks.

2. PRELIMINARIES

We adopt the Markov decision process (MDP) framework to model our environment, denoted by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, P, R, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, respectively. The probability distribution over transitions is given by $P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$, while the reward function is defined as $R(\mathbf{s}_t, \mathbf{a}_t)$. The discount factor, γ, is used to weigh the importance of future rewards. The agent starts in an initial state $\mathbf{s}_1$ sampled from the fixed distribution $p(\mathbf{s}_1)$ and chooses an action $\mathbf{a}_t \in \mathcal{A}$ from the state $\mathbf{s}_t \in \mathcal{S}$ at each timestep t. The agent then transitions to a new state $\mathbf{s}_{t+1}$ according to the probability distribution $P(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$. After each action, the agent receives a deterministic reward $r_t = R(\mathbf{s}_t, \mathbf{a}_t)$.

2.1. Setup and Notation

Throughout this paper, we adopt a consistent notation where bold symbols (e.g., $\mathbf{s}$, $\mathbf{a}$) denote vectors, while regular symbols (e.g., r, g) represent scalars. This distinction is particularly important as we deal with various modalities of different dimensionalities in our sequence-to-sequence formulation.

Our goal is to model the offline RL problem as a sequence modeling problem, with the agent having access to a fixed-size static training dataset $\mathcal{T}$. We use τ to represent a trajectory and $|\tau|$ to denote its length. The return-to-go, which we refer to as 'return' for short throughout the paper, of a trajectory τ at timestep t is defined as the sum of future rewards starting from that timestep, i.e., $g_t = \sum_{t'=t}^{|\tau|} r_{t'}$. We use G_t to represent the discounted returns which are computed as $G_t = \sum_{t'=t}^{|\tau|-1} \gamma^{t'-t} r_{t'}$.

The sequence of states, actions, rewards, and returns of trajectory τ are represented by $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_{|\tau|})$, $\mathbf{a} =$

$(\mathbf{a}_1, \ldots, \mathbf{a}_{|\tau|})$, $r = (r_1, \ldots, r_{|\tau|})$, and $g = (g_1, \ldots, g_{|\tau|})$, respectively. We use m and n to denote the dimensionality of the states and actions respectively.

To represent segments of trajectories, we define $\tau_{t_1}^{t_2}$ as a segment of the trajectory τ from timestep t_1 to t_2. The sequence of states, rewards, and returns in the segment $\tau_{t_1}^{t_2}$ are denoted by $\mathbf{s}_{t_1}^{t_2} = (\mathbf{s}_{t_1}, \ldots, \mathbf{s}_{t_2})$, and $g_{t_1}^{t_2} = (g_{t_1}, \ldots, g_{t_2})$, respectively.

2.2. Sequence Modeling for Offline RL

In general, sequence modeling approaches in offline RL perceive trajectory data as an unstructured sequence suitable for modeling via the GPT architecture [5]. Typically, this data is articulated in one of the two following forms:

$$\tau_{DT} := \left(g_1, \mathbf{s}_1, \mathbf{a}_1, g_2, \mathbf{s}_2, \mathbf{a}_2, \ldots, g_{|\tau|}, \mathbf{s}_{|\tau|}, \mathbf{a}_{|\tau|}\right), \quad (1)$$

$$\tau_{TT} := \left\{\mathbf{s}_t^1, \mathbf{s}_t^2, \ldots, \mathbf{s}_t^m, \mathbf{a}_t^1, \mathbf{a}_t^2, \ldots, \mathbf{a}_t^n, r_t, G_t\right\}_{t=1}^{|\tau|} \quad (2)$$

In this context, the subscripts on all tokens denote the timestep, while the superscripts on states and actions signify dimensions. The core objective is to develop a model capable of predicting action distributions. This can either be conditioned on desired returns and preceding states, a strategy employed in Decision Transformer (DT) [2], which typically aligns with the trajectory representation in (3); or it can entail crafting a distribution of actions based on previous states, accompanied by the utilization of a discounted return guided beam search for selecting actions, a methodology followed by Trajectory Transformer (TT) [6] and commonly corresponding to the trajectory configuration outlined in (2). Despite the distinct technical variations between these approaches (e.g., the implementation of quantization in [6] as opposed to [2]), both avenues employ a decoder-only architecture and fundamentally aim to employ the transformer architecture as a tool for policy modeling.

3. OFFLINE RL AS A SEQUENCE-TO-SEQUENCE MODELING PROBLEM

In this work, our goal is to disentangle the processing of different modalities in transformer-based approaches for offline RL by formulating the problem as a sequence-to-sequence modeling task. Given a dataset $\mathcal{T}$ of trajectories, where each trajectory τ is composed of sequences of states, actions, and returns-to-go tuples $\mathbf{s}, \mathbf{a}, g$, we model the relationship:

$$\text{Model: } g_{t+1}^{t+K_r+1} \mapsto \mathbf{s}_{t-K_s}^{t}, \quad \forall t > 0 \quad (3)$$

This formulation captures our sequence-to-sequence approach, where we learn to map sequences of future returns $g_{t+1}^{t+K_r+1}$ to sequences of historical states $\mathbf{s}_{t-K_s}^{t}$. Here, $\mathbf{s}_{t-K_s}^{t}$ denotes states from timestep $t-K_s$ to t, and $g_{t+1}^{t+K_r+1}$ denotes returns from timestep $t+1$ to $t+K_r+1$. For simplicity, we refer to these sequences as $\mathbf{s}_{-K_s}$ and g_{+K_r} throughout the paper, with K_r and K_s being the encoder and decoder context lengths, respectively.

This sequence-to-sequence formulation offers several advantages. Firstly, it enables mapping segments from different timesteps, allowing direct conditioning on a sequence of upcoming returns rather than a single scalar sum. Our experiments indicate that this direct conditioning of state generation enhances performance. We attribute this improvement to the richer, more detailed representation of future behavior provided by the sequence, which facilitates more effective credit assignment.

Additionally, this approach permits the use of different context lengths for returns and states. Considering a larger context length for returns, K_r, helps avoid short-sightedness in policy decisions, similar to the role of the discount factor in value-based RL algorithms. This extended context allows the model to capture long-term dependencies more effectively. In contrast, the context length for states, K_s, can be smaller due to the Markov property of the environment, which implies that fewer past states are needed for next state prediction.

3.1. Model Architecture

An encoder-decoder architecture[16] provides a robust framework for addressing sequence-to-sequence modeling problems. In our setting, the source segments of returns, g_{+K_r}, are processed by an encoder model, which consists of a stack of identical layers. Each layer has two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The target sequence, $\mathbf{s}_{-K_s}$, is processed by a decoder model, which also comprises a stack of layers. Notably, the number of layers in the decoder does not necessarily have to match the number of layers in the encoder. Each decoder layer includes its own masked multi-head self-attention mechanism and position-wise fully connected feed-forward network. Additionally, the decoder introduces a third sub-layer that performs multi-head attention over the encoder's output. This architecture allows for separate processing of each token modality with distinct sets of weights and the use of networks with varying depths. In contrast, the GPT architecture [5], used in prior sequence modeling approaches to offline RL [2, 6], which only employs a variant of the decoder part of the transformer model, lacks this level of flexibility.

3.2. Training the Sequence-to-Sequence Decision Translator

In a sequence-to-sequence framework, the mapping from the source space to the target space is learned by performing next token prediction in the target space. In our case, the mapping from g_{+K_r} to $\mathbf{s}_{-K_s}$ is learned by performing next state prediction. To this end, we introduce a probabilistic function ρ_θ which models the distribution of states. This function is parameterized by the transformer model with both its components—the encoder and decoder—which we denote with weights θ, and it assigns probability density at the t-th time step, conditioned on the future K_r instances of returns and

K_s historical state instances.

More precisely, the state distribution function ρ_θ is modeled using a multivariate Gaussian distribution parameterized by the transformer model which outputs its mean μ_θ and a diagonal covariance matrix Σ_θ, i.e.,

$$\rho_\theta(\mathbf{s}_{t+1} \mid \mathbf{s}_{-K_s}, g_{+K_r}) = \mathcal{N}\left(\mu_\theta(\mathbf{s}_{-K_s}, g_{+K_r}), \Sigma_\theta(\mathbf{s}_{-K_s}, g_{+K_r})\right), \quad \forall t > 1 \tag{4}$$

Subsequently, ρ_θ can be learned by minimizing the negative log-likelihood (NLL) loss, which can be expressed as follows:

$$J(\theta) = -\mathbb{E}_{\mathbf{s}_{t-K_s}^{t+1}, g_{+K_r} \sim \mathcal{T}}\Big[\log \rho_\theta(\mathbf{s}_{t+1} \mid \mathbf{s}_{-K_s}, g_{+K_r}) + \sum_{i=t-K_s}^{t-1} \mathbf{1}_{\{i+1 \leq K_s\}} \log \rho_\theta(\mathbf{s}_{i+1} \mid \mathbf{s}_{i-K_s}^{i}, g_{+K_r})\Big] \tag{5}$$

By setting $\mathbf{1}_{\{i+1 \leq K_s\}}$ to 1, we recover the training regime commonly employed in sequence-to-sequence approaches, where the model is trained to predict other tokens in the context, effectively varying the context length from 1 to K_s. This strategy of training with varying context lengths, even when the full context is available, is widely used in sequence prediction tasks to expose the model to diverse scenarios, potentially promoting robustness, adaptability, and generalization to unseen sequences.

However, in our case, since subsequent states are highly correlated, as can be seen in Figure 2, this approach may lead the model to overfit easily to these correlations by assigning high attention to nearby states, potentially neglecting more distant tokens that could be valuable for the ultimate objective of return maximization. To address this issue, we propose a simple adaptation: setting $\mathbf{1}_{\{\mathbf{i+1} \leq \mathbf{K_s}\}}$ to $\mathbf{0}$. As Figure 2 shows, this final token prediction approach makes the model less biased toward the linear correlations between subsequent states in the dataset.

3.3. Policy Extraction and Inference

Predicting states using the transformer model is insufficient for defining a controller. Nevertheless, a policy can be inferred by estimating the action $\mathbf{a}_t$ that caused the transition from state $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ at any timestep t in τ. Given two consecutive states, we generate an action according to the inverse dynamics model [18, 19], which we model as a conditional multivariate Gaussian distribution, as follows:

$$\pi_\phi\left(\mathbf{a}_t \mid \mathbf{s}_t, \mathbf{s}_{t+1}\right) = \mathcal{N}\left(\mu_\phi\left(\mathbf{s}_t, \mathbf{s}_{t+1}\right), \Sigma_\phi\left(\mathbf{s}_t, \mathbf{s}_{t+1}\right)\right) \tag{6}$$

Note that the same offline data utilized to train the state predictor model ρ_θ can also be employed to learn π_ϕ. Furthermore, we employ the same framework as used for the transformer model, utilizing maximum likelihood estimation to learn the parameters ϕ.

Algorithm 1 Inference with RGDT

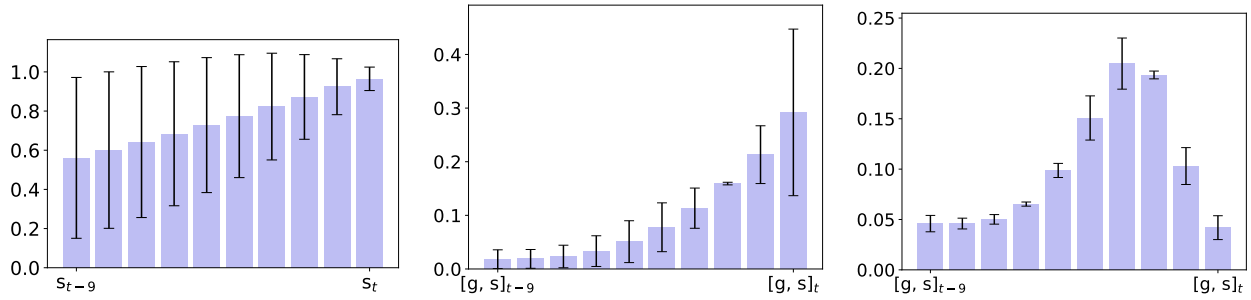
Input: $g_{+K_r}, \mathbf{s}_{-K_s}$
Output: $\hat{\mathbf{a}}_t$
procedure INFERENCE($g_{+K_r}, \mathbf{s}_{-K_s}$)
  Initialize $t \leftarrow 1$
  while not done **do**
    $\hat{\mathbf{s}}_{t+1} \leftarrow \mu_\theta(\mathbf{s}_{-K_s}, g_{+K_r})$
    $\hat{\mathbf{a}}_t \leftarrow \mu_\phi(\mathbf{s}_t, \hat{\mathbf{s}}_{t+1})$
    Execute $\hat{\mathbf{a}}_t$
    Observe $\mathbf{s}_{t+1} \sim P(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$
    $t \leftarrow t + 1$
  end while
end procedure

During inference, a desired sequence of returns $g_2^{K_r+1}$ and an initial state $\mathbf{s}_1$ are specified. In practice, to filter for good behavior during inference, the whole sequence of returns g is constructed by taking the best return per-timestep from the dataset of trajectories $\mathcal{T}$. The RGDT model then generates the first state $\hat{\mathbf{s}}_2 = \mu_\theta(\mathbf{s}_1, g_2^{K_r+2})$. Then, both $\mathbf{s}_1$ and $\hat{\mathbf{s}}_2$ are fed to the inverse dynamics model π_ϕ which generates a deterministic action according to $\mathbf{a}_1 = \mu_\phi(\mathbf{s}_1, \mathbf{s}_2)$. After executing the action $\mathbf{a}_1$, the agent observes the next state $\mathbf{s}_2$ from the transition probability distribution $P\left(\cdot \mid \mathbf{s}_1, \mathbf{a}_1\right)$. Subsequently, the RGDT generates the next goal state $\hat{\mathbf{s}}_3$ based on $g_3^{K_r+3}, \mathbf{s}_1, \mathbf{s}_2$ as inputs. This process continues until the episode terminates.

4. EXPERIMENTS

We evaluate RGDT against leading offline RL approaches on the D4RL benchmark [20], comparing performance with both sequence modeling methods and off-policy algorithms. Our experiments focus on three aspects: (1) overall performance across continuous control tasks, (2) benefits of final token prediction over full context prediction, and (3) advantages of our sequence-to-sequence formulation.

Implementation Details and Hyperparameters. Our RGDT implementation utilizes a transformer with hidden dimension 64 and 2 attention heads. The encoder and decoder comprise 2 and 3 layers, respectively, reflecting the asymmetric nature of our sequence-to-sequence formulation. We employed context lengths of $K_r = 20$ for returns and $K_s = 5$ for states, enabling the model to consider extended future horizons while maintaining computational efficiency. For optimization, we used the AdamW optimizer with a learning rate of 5×10^{-4}, linear warmup over 10,000 iterations, weight decay of 10^{-3}, and dropout probability of 0.1. Training continued for 150,000 gradient steps with a batch size of 256. The inverse dynamics model consists of a neural network with 2 hidden layers of 512 units each, trained using the Adam optimizer with learning rate 10^{-4}, weight decay of 10^{-4}, and dropout of 0.1. This model was trained for 200,000 gradient steps with a batch size of 1024.



(a) Linear correlation in the dataset (b) Full context prediction (c) Final token prediction

Fig. 2: (a) Pearson correlation between each state and its subsequent states in Walker2d-Medium, decreasing with timestep offset. (b) Attention weights for GPT model with full context token prediction. (c) Attention weights with final token prediction only. Full context prediction (b) biases attention toward linear state correlations observed in (a), while final token prediction (c) shows more balanced attention distribution.

Benchmark and Compared Baselines. We evaluate RGDT on the D4RL benchmark [20] using the Gym environment [21], focusing on six continuous control tasks: halfcheetah-medium (HC-M), hopper-medium (HP-M), walker2d-medium (WK-M) and their medium-expert (HC-ME, HP-ME, WK-ME) variants. We compare against four established baselines: 10%BC [2], which replicates the top decile of behaviors in the dataset; and two leading dynamic programming methods - Conservative Q-Learning (CQL) [22] and Implicit Q-Learning (IQL) [23], as well as Decision Transformer (DT) [2], the primary sequence modeling approach. All baseline results are sourced from their respective papers with CQL results adapted from [23] for the v2 datasets.

Full Context Token Prediction vs. Final Token Prediction. An important aspect of our method is the training regime of final token prediction as implied by equation 5. As demonstrated in Figure 3, this approach leads to significantly better performance compared to full context token prediction across the D4RL medium tasks.

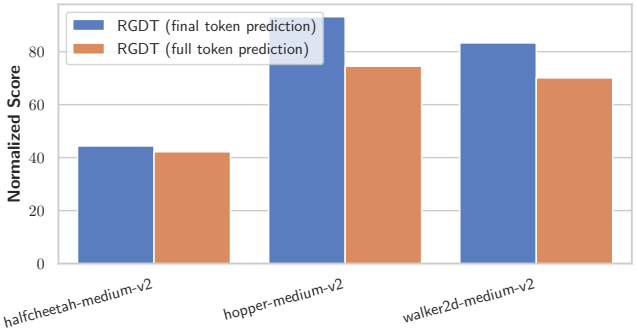


Fig. 3: Comparison of RGDT training strategies on Gym D4RL medium tasks.

5. CONCLUSION

We introduced RGDT, a sequence-to-sequence approach for offline RL that disentangles modality processing through an encoder-decoder architecture. By translating future returns to state sequences and using an inverse dynamics model for action inference, our method effectively addresses the limitations of decoder-only architectures and achieves superior performance on the D4RL benchmark.

Future work could explore: (1) adapting RGDT to visual observations and complex state spaces, (2) investigating transfer learning across tasks with similar dynamics but different return structures. The sequence-to-sequence formulation presented here offers a promising foundation for addressing the multimodal nature of sequential decision-making while maintaining computational efficiency.

Table 1: Normalized scores on Gym D4RL tasks (10 seeds). Our method **RGDT** achieves the best performance.

Dataset	10% BC	CQL	IQL	DT	RGDT
HC-M	42.5	44.0	**47.4**	42.6	44.4
HP-M	56.9	58.5	66.3	67.6	**93.2**
WK-M	75.0	72.5	78.3	74.0	**83.3**
HC-ME	92.9	91.6	86.7	86.8	**93.4**
HP-ME	**110.9**	105.4	91.5	107.6	**112.4**
WK-ME	109.0	108.8	109.6	108.1	**113.3**
Total	487.2	480.8	479.8	486.7	**539.0**

6. REFERENCES

[1] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[2] 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*, vol. 34, pp. 15084–15097, 2021.

[3] Jie Yan, Quan Liu, and Lihua Zhang, "Offline reinforcement learning based on next state supervision," in

ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 5775–5779.

[4] Lan Wu, Quan Liu, Lihua Zhang, and Zhigang Huang, "Offline reinforcement learning with generative adversarial networks and uncertainty estimation," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 5255–5259.

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

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

[7] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal, "Is conditional generative modeling all you need for decision-making?," 2023.

[8] Russ Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation," 2023, Course notes for MIT 6.832.

[9] Ramin Ghorbani, Marcel J.T. Reinders, and David M.J. Tax, "Restad: Reconstruction and similarity based transformer for time series anomaly detection," in *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2024, pp. 1–6.

[10] Hojjat Salehinejad, Navid Hasanzadeh, Radomir Djogo, and Shahrokh Valaee, "Contrastive representation of channel state information for human body orientation recognition in interaction with machines," in *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, 2023, pp. 1–6.

[11] Anders Gjølbye, Lina Skerath, William Lehn-Schiøler, Nicolas Langer, and Lars Kai Hansen, "Speed: Scalable preprocessing of eeg data for self-supervised learning," in *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2024, pp. 1–6.

[12] Yiqi Wang, Mengdi Xu, Laixi Shi, and Yuejie Chi, "A trajectory is worth three sentences: multimodal transformer for offline reinforcement learning," in *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, Robin J. Evans and Ilya Shpitser, Eds. 31 Jul–04 Aug 2023, vol. 216 of *Proceedings of Machine Learning Research*, pp. 2226–2236, PMLR.

[13] Xuenan Xu, Arshdeep Singh, Mengyue Wu, Wenwu Wang, and Mark D Plumbley, "Investigating passive filter pruning for efficient cnn-transformer audio captioning," in *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2024, pp. 1–6.

[14] Sungkyun Chang, Emmanouil Benetos, Holger Kirchhoff, and Simon Dixon, "Yourmt3+: Multi-instrument music transcription with enhanced transformer architectures and cross-dataset stem augmentation," in *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2024, pp. 1–6.

[15] Umberto Cappellazzo, Daniele Falavigna, Alessio Brutti, and Mirco Ravanelli, "Parameter-efficient transfer learning of audio spectrogram transformers," in *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2024, pp. 1–6.

[16] 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*, vol. 30, 2017.

[17] Teerapat Jenrungrot, Michael Chinen, W Bastiaan Kleijn, Jan Skoglund, Zalán Borsos, Neil Zeghidour, and Marco Tagliasacchi, "Lmcodec: A low bitrate speech codec with causal transformer models," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[18] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine, "Learning to poke by poking: Experiential learning of intuitive physics," 2017.

[19] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell, "Zero-shot visual imitation," 2018.

[20] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine, "D4rl: Datasets for deep data-driven reinforcement learning," 2021.

[21] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine, "Conservative q-learning for offline reinforcement learning," 2020.

[23] Ilya Kostrikov, Ashvin Nair, and Sergey Levine, "Offline reinforcement learning with implicit q-learning," 2021.