Efficient Decision Sequence Modeling via Feature-Level Masking

Xinzhi Zhang
sezhangxinzhi@mail.scut.edu.cn
South China University of Technology
Guangzhou, China

Yifan Gao
evanngao@tencent.com
Tencent
Shenzhen, China

Yaqing Hou
houyq@dlut.edu.cn
Dalian University of Technology
Dalian, China

Xuechuan Liu
202420145206@scut.edu.cn
South China University of Technology
Guangzhou, China

Zengyang Wang
202420145210@scut.edu.cn
South China University of Technology
Guangzhou, China

Yi Cai
ycai@scut.edu.cn
South China University of Technology
Guangzhou, China

Bo An
boan@ntu.edu.sg
Nanyang Technological University
Singapore

Mengchen Zhao*
zzmc@scut.edu.cn
South China University of Technology
Guangzhou, China

Abstract

Decision models based on sequence modeling have become prevalent in the field of offline reinforcement learning. However, existing approaches such as Decision Transformer and Trajectory Transformer suffer from poor data efficiency. One important reason is that they fail to extract useful information from potentially high-dimensional and noisy states. To resolve this issue, we propose a data-efficient decision sequence modeling method called Data Efficient Decision Sequence Model (DEDS), which dynamically identifies and filters out task-irrelevant state features for more compact state representations. Specifically, DEDS employs a state mask model guided by the bisimulation metric to ensure that only the most task-relevant state features are preserved for decision-making. Extensive experiments on various environments demonstrate that DEDS outperforms existing offline RL methods and achieves a significantly high data efficiency, especially in tasks with high-dimensional and complex state spaces.

CCS Concepts

• **Computing methodologies** → **Sequential decision making**.

Keywords

Sequential decision making, Feature selection

ACM Reference Format:
Xinzhi Zhang, Yifan Gao, Yaqing Hou, Xuechuan Liu, Zengyang Wang, Yi Cai, Bo An, and Mengchen Zhao. 2025. Efficient Decision Sequence Modeling via Feature-Level Masking. In *Proceedings of Distributed AI (DAI).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Mengchen Zhao is the corresponding author.

DAI, London

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

In reinforcement learning (RL), state representation is crucial because it governs how an agent perceives information from the environment. By reducing high-dimensional observations to a smaller set of important features, a good state representation can streamline policy learning and speed up convergence. In online RL, existing works focus on creating compact state representations to improve data efficiency when training in complex environments. Such state representations allow agents to achieve strong performance with significantly fewer samples, which is essential to real-world scenarios where data collection can be expensive or difficult [19, 28].

However, in offline RL, developing effective state representations becomes considerably more challenging due to the lack of access to the environment. First, without the ability to interact with the environment, agents cannot leverage real-time feedback to refine their state representations iteratively. Second, the set of features that are relevant to decision-making may vary at different time steps, making it hard to accurately capture the dynamic task-relevant information. Recent works cast the sequential decision-making problem as a unified sequence modeling problem. For example, Decision Transformer [4] and Trajectory Transformer [11] learn decision policies from offline datasets, which successfully capture the dependencies between states, actions and rewards. While these methods have shown impressive empirical results, they typically lack explicit feature selection or effective representation learning. As a result, they usually exhibit poor data efficiency, particularly in large or noisy state spaces with a large number of task-irrelevant features. Therefore, designing more effective state representations remains critical for offline sequence modeling methods.

In this paper, we propose a Data Efficient Decision Sequence Model (DEDS) to accelerate the learning process by integrating a bisimulation-based feature-level state mask model. Specifically, the implementation of DEDS consists of three steps. First, under

the offline setting, we learn a transition model and a reward model from the offline dataset. Second, guided by the bisimulation metric, we learn a state mask model that ensures any states leading to a similar reward are closely represented in the latent space, so as to reduce the effect of task-irrelevant features. Third, since the set of task-relevant features may change over time, DEDS incorporates an adaptive masking strategy using the learned state mask model to decide which state features (i.e., tokens) to reserve.

By integrating the state mask model in sequence modeling, DEDS substantially enhances data efficiency in offline RL with high-dimensional and noisy state space. Moreover, the state mask model dynamically filters out task-irrelevant state features to reduce the difficulty of policy learning and boost the policy performance. We conduct extensive experiments on the D4RL [8] and Adroit [18] benchmarks using various offline RL and sequence modeling baselines. Experimental results show that DEDS significantly outperforms existing approaches in terms of both data efficiency and policy performance. Our main contributions are summarized as follows:

- We introduce an adaptive state mask model, which is learned using offline datasets under the guidance of bisimulation. The state mask model can effectively filter out task-irrelevant features for any given state.
- We introduce the framework of DEDS, which integrates the state mask model with sequence modeling based decision policies. The state mask model is employed in both training and inference procedures for learning a compact state representation at every time step.
- We demonstrate through extensive experiments on MuJoCo tasks that DEDS significantly outperforms existing offline RL and sequence modeling approaches in terms of both policy performance and data efficiency.

2 Related Work

2.1 Data-Efficient Reinforcement Learning

Many approaches have been proposed to enhance data efficiency in RL. For instance, EfficientZero [26] combines self-supervised learning (SSL) and data augmentation within a model-based framework to reduce data complexity in an end-to-end manner. Methods like CURL [14] and RCRL [16] integrate contrastive learning, aiming to learn robust representations by distinguishing between state–action pairs with different rewards. Although these techniques improve data efficiency, they often rely on auxiliary tasks that can introduce features unrelated to the primary decision-making objective. Another avenue for improving data efficiency lies in bisimulation metrics, which gauge the behavioral similarity of states by comparing their long-term reward sequences under any action sequence [7, 13]. In continuous MDPs, Ferns et al. [6] introduced a Monte Carlo algorithm that leverages the Wasserstein distance between empirically measured transition distributions to approximate the bisimulation metric. More recently, Castro [3] proposed an on-policy method tailored to deterministic settings and policy evaluation, focusing on computing bisimulation metrics without learning explicit representations. In this paper, we incorporate bisimulation metrics into the state mask model that selectively removes task-irrelevant state features. By directly leveraging behavioral similarity, our approach filters out redundant dimensions and fosters a data-efficient representation, leading to more effective and robust policies.

2.2 Sequence Modeling for Offline RL

Transformers have recently been used to reformulate offline RL as a sequence modeling task, enabling the direct prediction of actions from observation and task specification sequences [4, 11]. Two notable examples are the Decision Transformer (DT) [4], which uses reward-to-go together with state and action sequences to predict the next action, and the Trajectory Transformer (TT) [11], which discretizes each dimension of the inputs into tokens for next-action prediction. While these methods have demonstrated promising results, they rely on raw, high-dimensional states without explicit mechanisms to filter out task-irrelevant features, thereby limiting their data efficiency—particularly in environments where noise obscures key information. Although recent works propose variations of the basic transformer-based framework [23–25], the overarching challenge persists: many sequence-modeling methods rely on raw, high-dimensional states without a principled way to filter task-irrelevant features. This shortcoming becomes especially problematic in large or noisy state spaces, where crucial signals can be obscured. Our approach tackles this issue by introducing an adaptive masking strategy, which employs the state mask model to dynamically discard redundant dimensions and thereby improve both data efficiency and policy performance in offline sequence-based Reinforcement Learning.

2.3 Trajectory Masking

Recent transformer-based RL studies have explored masking techniques to enhance representation learning and improve data efficiency. For example, Masked and Inverse Dynamics Modeling [15] masks portions of state-action sequences and trains the model to both reconstruct missing elements and predict inverse dynamics, thereby encouraging robust feature extraction. RePreM [2] adopts a similar pretraining approach by masking input sequences to enhance downstream RL performance, while Masked Trajectory Models [21] focus on sequence reconstruction and prediction to improve control and state representations. Additionally, Mask-based Latent Reconstruction [27] filters out task-irrelevant information by masking latent variables within trajectory data, thereby improving decision-making and task-specific feature extraction. However, these methods typically mask trajectories in a reconstruction-driven manner, which can force the model to learn details unrelated to decision-making simply because they are needed to restore missing parts of the sequence.

3 Preliminaries

3.1 Sequence Modeling for Decision Making

Previous work [4, 11] has explored framing offline RL as a sequential modeling problem. Specifically, Decision Transformer (DT) [4] represents trajectories τ as sequences in the following format:

$$\tau = (\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, ..., \hat{R}_T, s_T, a_T)$$

where s is the state, a is the action and $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ denotes the *return-to-go*, defined as the sum of future rewards starting from timestep t. This trajectory is tokenized and fed into a GPT-based Transformer architecture [17], which serves as an expressive policy function approximator π_θ. The model is trained to predict the next

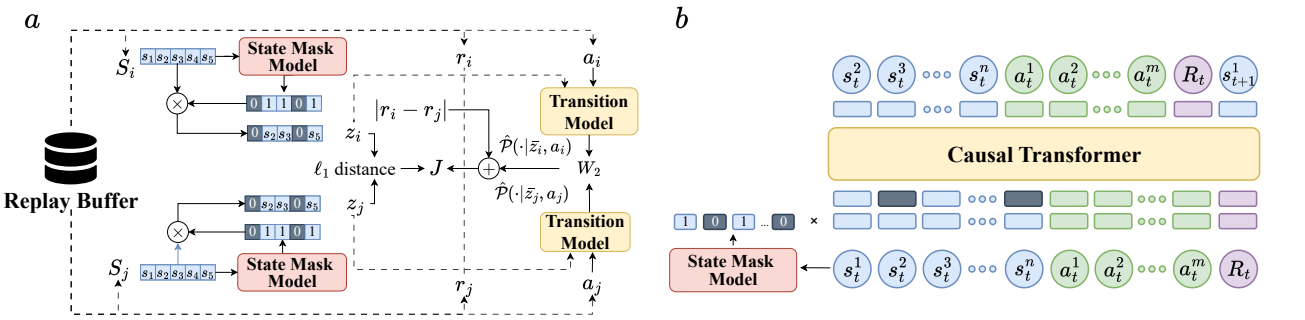


Figure 1: (a) The framework for training the state mask model. Specifically, the state mask model is designed to generate masks so that the ℓ_1 distance of any two masked states z_i and z_j aligns with the bisimulation metric. During training, a pair of trajectory data (s_i, a_i, r_i) and (s_j, a_j, r_j) are sampled. Then, the transition probabilities corresponding to (z_i, a_i) and (z_j, a_j) are estimated by the transition model. The objective function J is defined as the mean squared error between the distance in latent space $||z_i, z_j||_1$ and the calculated bisimulation metric between the two states. (b) Sequence modeling with the state mask model. The input is the sequences of states, actions and returns-to-go. The state mask model takes the states as input and generates a mask vector that indicates which tokens should be masked. Masked tokens are fed into a GPT architecture which predicts the next token autoregressively.

action by maximizing the likelihood of observed actions in the offline dataset $\mathcal{D} = \{\tau^{(m)}\}_{m=1}^{M}$ using the following objective:

$$\mathcal{L}_{DT} = \mathbb{E}_{\tau\sim\mathcal{D}}[\sum_{t=1}^{T} -log\pi_\theta(a_t|\tau_{1:t-1}, s_t, \hat{R}_t)]$$

3.2 Bisimulation Metric

Bisimulation describes the behavioral equivalence of the state space. If s_i and s_j cause the same probability of the accumulated reward for any actions, then s_i and s_j are bisimilar states. A recursive version of this concept is that bisimilar states get the same reward at this timestep and the same transition distribution to the next state.

DEFINITION 1. ***Bisimulation Relations.****[9] Given an MDP $\mathcal{M}$ with state space $\mathcal{S}$, for $s_i, s_j \in \mathcal{M}$, define s_i and s_j is equivalent under Bisimulation Relation B if:*

$$\mathcal{R}(s_i, a) = \mathcal{R}(s_j, a) \qquad \forall a \in \mathcal{A} \qquad (1)$$

$$\mathcal{P}(G|s_i, a) = \mathcal{P}(G|s_j, a) \qquad \forall a \in \mathcal{A}, \forall G \in \mathcal{S}_B \qquad (2)$$

where $\mathcal{S}_B$ is a group of states, $\mathcal{S}_B$ is the partition of state under B, and $\mathcal{P}(G|s, a) = \sum_{s'\in G} \mathcal{P}(s'|s, a)$.

DEFINITION 2. ***Bisimulation Metric****. From Theorem 2.6 in [6]:*

$$d(s_i, s_j) = \max_{a\in\mathcal{A}}(1 - c) \cdot |\mathcal{R}^a_{s_i} - \mathcal{R}^a_{s_j}| + c \cdot W_1(\mathcal{P}^a_{s_i}, \mathcal{P}^a_{s_j}; d). \qquad (3)$$

where c is a parameter that ranges from 0 to 1, W_1 represents the Wasserstein distance.

4 Method

Motivation. There are two main challenges to develop effective state representations under the offline setting: (1) The inability to interact with the environment prevents iterative refinement of state representations; (2) The task-relevant features may change over time, thus it is hard to accurately identify them in the offline setting. Fortunately, we note that the bisimulation metric can more effectively evaluate the distance between states from the decision making perspective. Existing works have introduced the bisimulation metric to online RL [29]. However, whether the metric is effective in offline RL, particularly in the context of sequence modeling, remains largely unexplored. This motivates us to learn an adaptive mask model using the bisimulation metric. By doing so, we aim to endow the agent with the ability to selectively attend to relevant parts of the state, mimicking the way humans filter out distractions and focus on salient information when making decisions.

Overview. Overall, DEDS consists of two key components as illustrated in Figure 1. The first component is the state mask model that is learned from offline data under the guidance of a bisimulation metric. The second component is a feature-level sequence model that integrates the state mask model. Unlike Decision Transformers [4] which treat each state as a token, DEDS treats each feature of state as a token, thus allowing feature-level masking. This is similar to the tokenization procedure in Trajectory Transformers [11]. Note that our state mask model can be combined with any variants of sequence modeling methods as long as the state spaces are consist of multiple features.

4.1 State Mask Model

To identify task-irrelevant features in offline sequence modeling, we propose a state mask model to generate practical state masks. A state mask, denoted as $I \in \mathcal{I}$, is a binary vector of the same size as the state space $\mathcal{S}$. The number of ones in the mask I is denoted by $|I|$, while the number of zeros is represented by $|1 - I|$. $f_\phi(s_t)$ denotes the state mask model parameterized by ϕ, where $I_t \sim f_\phi(s_t)$. We employ $\pi_\theta(a_t|s_t)$ to represent the original policy and $\tilde{\pi}_{\theta,\phi}(a_t|\tilde{s}_t)$ to represent the composite policy with mask. Here, $\tilde{s}_t \in \tilde{S}_t$, where $\tilde{S}_t$ signifies the masked state space at timestep t.

The primary objective of the state mask is to output a binary vector that determines which feature tokens should be masked, so that the agent can focus on task-relevant information. Traditional binary vector generation methods, such as hand-crafted thresholding and classification model based thresholding, can hardly be trained in an end-to-end manner due to the discontinuous gradient flow.

To overcome this limitation, we employ the Gumbel-Softmax trick [10], which approximates discrete binary outputs while preserving differentiability. By adding Gumbel noise to each component of the probability vector and applying a softmax function, Gumbel-Softmax provides a smooth approximation of binary values. This allows the gradient to propagate through the state mask model, facilitating the learning of state masks and enhancing the data efficiency in high-dimensional state spaces. Formally, the mask is generated:

$$I_t = \text{Gumbel}[f_\phi(s_t^1, ..., s_t^N)] \tag{4}$$

where $f(\cdot)$ is the state mask model that transforms an N-dimensional state to an N-dimensional vector of real numbers.

4.2 Learning the State Mask Model

We train the state mask model using a bisimulation metric to ensure its ability to identify and mask task-irrelevant state dimensions. When the model effectively filters out task-irrelevant dimensions, the resulting masked states should closely align with decision-making requirements. This alignment ensures that the distance between any two masked states satisfies the bisimulation metric defined in Equation 3. To guide the model towards this desired property, we define the distance between masked states as $d(s_i, s_j) := ||z_i, z_j||$, where $z = I \cdot s$, represents masked states. During training, we sample batches of state pairs and minimize the mean squared error between the bisimulation metric and the ℓ_1 distance of the masked states. The objective is defined as:

$$\begin{aligned} J(\phi) = (&\beta||z_i, z_j||_1 - |r_i - r_j| \\ &- \gamma W_2(\hat{\mathcal{P}}(\cdot|\bar{z_i}, a_i), \hat{\mathcal{P}}(\cdot|\bar{z_j}, a_j)))^2 \\ &+ \alpha \frac{|1 - I|}{dim(I)} \end{aligned} \tag{5}$$

where r are rewards, and $\bar{z}$ denotes z with stop gradients. $\frac{|1-I|}{dim(I)}$ acts as a regularization term to prevent excessive sparsity in the mask I by penalizing excessive zeros. This ensures that the mask captures sufficient information to support effective decision-making while maintaining computational efficiency. Here, Equation 5 incorporates a probabilistic dynamics model $\hat{\mathcal{P}}$ which outputs a Gaussian distribution. Consequently, we employ the 2-Wasserstein metric W_2 in Equation 5, following [29], rather than the 1-Wasserstein distance defined in Equation 3. This distance is computed as $W_2(\mathcal{N}(\mu_i, \Sigma_i), \mathcal{N}(\mu_j, \Sigma_j))^2 = ||\mu_i - \mu_j||_2^2 + ||\Sigma_i^{1/2} - \Sigma_j^{1/2}||_{\mathcal{F}}^2$ where $||\cdot||_{\mathcal{F}}$ is the Frobenius norm. For all other distances we continue using the ℓ_1 norm. The training of the state mask model is illustrated in Algorithm 1.

4.3 Integrating the State Mask Model with Sequence Modeling

Following prior work [4, 11], we reformulate the offline reinforcement learning problem as a sequence modeling task, where the goal is to predict the probability of the next token x_i conditioned on all previously observed tokens, $P_\theta(x_i|x_{<i})$. We use a Transformer decoder architecture based on GPT [17] as our backbone, taking advantage of its capacity to capture long-range dependencies in sequential data. To obtain more granular insights, we decompose each state and action into its individual dimensions. The resulting input sequences have the form:

$$\tau = (..., s_1^t, s_2^t, ..., s_n^t, a_1^t, a_2^t, ...a_m^t, R^t, ...)$$

where t represents a timestep, n and m are the dimensions of the state and action spaces, $R^t = \sum_{t'=t}^{T} r_{t'}$ denotes the agent's target return for the remaining trajectory.

Algorithm 1 State Mask Model Learning

Data:Offline Dataset Data $\mathcal{D}$ and number of iterations T.
Result: State Mask Model

```
for t = 0 to T do
    Sample data (s_i, a_i, r_i), (s_j, a_j, r_j) ~ D
    Compute state mask: I_i = f_φ(s_i), I_j = f_φ(s_j)
    Compute masked state: z_i = I_i · s_i, z_j = I_j · s_j
    Estimate the transition distributions:P̂(·|z_i, a_i)
    Train state mask model with loss J(φ) (Equation 5)
    Train dynamics: J(P̂, φ) = (P̂(·|z_i, a_i) − z')^2
end for
```

For continuous states and actions, we adopt a quantile-based discretization method [11], in which each dimension is split into V bins, each representing an equal probability mass from the empirical data distribution. This ensures that every token corresponds to one of the V quantiles, capturing a balanced range of values observed in the dataset. While this discretization improves the model's ability to handle complex dynamics, it also produces a larger token set, potentially hindering data efficiency in high-dimensional tasks. To mitigate this issue, we employ a state mask mechanism that identifies and discards task-irrelevant tokens, allowing the Transformer to focus on decision-critical information. Concretely, dimensions that contribute minimally to action selection are masked out, thereby reducing both noise and computational overhead. This targeted filtering not only enhances the data efficiency of our approach but also enables more robust learning in scenarios where task-irrelevant or redundant features might otherwise dominate. An overview of DEDS is shown in Figure 1.

During training, we sample mini-batches of sequences with a length K from the offline dataset, resulting in $(m + n + 1) * K$ tokens. Denoting the parameters of the transformer as θ and including conditional probabilities as P_θ, the following objective function maximized during training is:

$$\begin{aligned} \mathcal{L}(\tau) = \sum_{t=1}^{T} (&\sum_{i=1}^{n} I_t^i \cdot logP_\theta(s_i^t | \mathbf{s}_i^{<t}, \tau_{<t}) \\ &+ \sum_{j=1}^{m} W_a \cdot logP_\theta(a_t^j | \mathbf{a}_t^{<j}, \mathbf{s}_t, \tau_{<t}) \\ &+ W_r \cdot logP_\theta(r_t | \mathbf{a}_t, \mathbf{s}_t, \tau^{<t})) \end{aligned} \tag{6}$$

where $\tau^{<t}$ denotes the trajectory from timesteps 0 through $t-1$, $\mathbf{s}_{<i}^t$ denotes the first $i-1$ dimensions of the state at timestep t, and similarly for $\mathbf{a}_{<j}^t$. We assign different weights to different components of the loss function. The state weight is represented by I_t^i, which is the mask corresponding to the state dimension s_t^i. Only when $I_t^i = 1$ does the corresponding state dimension contribute to the loss. Additionally, we set the action weight $W_a = 5$ and the reward weigh

$W_r = 1$. We optimize the parameters θ using the Adam optimizer[5] with a learning rate of 2.5×10^{-4}.

5 Experiments

Our experiments aim to answer the following key questions:

- (RQ1:) How does DEDS perform compared to existing offline RL and sequence modeling methods?
- (RQ2:) How effective is the state mask model in filtering out task-irrelevant features?
- (RQ3:) How does DEDS improve the data efficiency during the training process?

5.1 Baselines

We use six different offline reinforcement learning methods as baselines, each represents a different type of approach. The descriptions are as follows:

- **Behavior Cloning (BC)**: BC is a supervised learning approach where a policy is trained to directly mimic the actions observed in the dataset.
- **Model-Based Offline Planning (MBOP)** [1]: MBOP builds an environment model from offline data, generating future trajectories to select actions by simulating long-term returns, improving policy performance without interaction.
- **Behavior-Regularized Actor-Critic (BRAC)** [22]: BRAC mitigates distributional shift in offline RL by adding a regularization term that keeps the learned policy close to the behavior in the offline dataset.
- **Conservative Q-Learning (CQL)** [12]: CQL prevents overestimation of out-of-distribution actions by penalizing Q-values of actions not present in the dataset.
- **Decision Transformer (DT)** [4]: DT models reinforcement learning as a sequence prediction task by taking states, actions, and rewards-to-go as input sequences, leveraging a Transformer architecture to predict future actions.
- **Trajectory Transformer (TT)** [11]: TT by treating each dimension of states, actions, and rewards as an individual token.

5.2 Benchmark Datasets

In this section, we draw upon a variety of continuous control tasks and datasets that leverage the MuJoCo simulator[20]. The different datasets are described below.

- **D4RL** [8]: D4RL is a popular offline RL benchmark consisting of several environments and datasets. Following a number of prior work, we focus on the locomotion subset: HalfCheetah, Hopper, and Walker2D.
- **Adroit** [18]: Adroit is a collection of dexterous manipulation tasks with a simulated five-fingered robotic hand.

5.3 Performance on Datasets (RQ1)

Our results on the D4RL datasets are shown in Table 1, with scores normalized so that a value of 100 corresponds to an expert policy, following the methodology in [8]. As illustrated, our method achieves strong performance across all tasks, consistently surpassing other offline reinforcement learning baselines, including Sequence Modeling approaches such as Decision Transformer (DT) and Trajectory Transformer (TT). This consistent improvement across different datasets indicates that DEDS is not only competitive in specific cases but also maintains a reliable advantage in a broad range of offline RL scenarios. Although DT and TT also utilize Sequence Modeling to capture long-term dependencies, our method attains the highest average score. This suggests that simply modeling sequential dependencies is insufficient when the input state contains a high proportion of irrelevant or redundant information. This improvement can be attributed to our approach's effective masking of task-irrelevant state features, allowing the model to focus on more useful information and perform better in complex and high-dimensional environments.

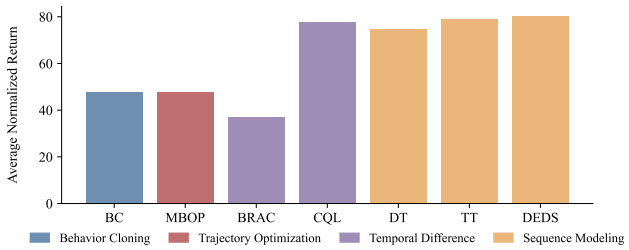


Figure 2: Average return on D4RL datasets. This plot shows the average per-algorithm performance in Table 1, with bars colored according to a crude algorithm categorization.

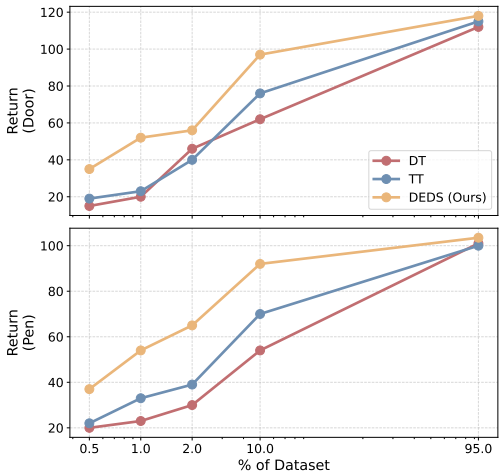


Figure 3: Dataset efficiency on Adroit datasets. We train DEDS, DT and TT on the Adroit Door and Pen tasks across a range of dataset sizes, measured by the percent of the original expert dataset. We observe that DEDS consistently outperforms DT and TT in the low-data regime, demonstrating the advantage of adaptive masking.

Results for Adroit tasks are presented in Table 2. Across both the Expert and Medium-Replay datasets for the Pen and Door tasks,

Dataset	Environment	BC	MBOP	BRAC	CQL	DT	TT	DEDS
Medium-Expert	HalfCheetah	59.9	**105.9**	41.9	91.6	86.8	95.0	83.5
Medium-Expert	Hopper	79.6	55.1	0.9	105.4	107.6	110.0	**113.9**
Medium-Expert	Walker2d	36.6	70.2	81.6	108.8	108.1	101.9	**109.0**
Medium	HalfCheetah	43.1	44.6	46.3	44.0	42.6	**46.9**	46.1
Medium	Hopper	63.9	48.8	31.3	58.5	67.6	61.1	**69.2**
Medium	Walker2d	77.3	41.0	81.1	72.5	74.0	79.0	**82.8**
Medium-Replay	HalfCheetah	4.3	42.3	**47.7**	45.5	36.6	41.9	42.1
Medium-Replay	Hopper	27.6	12.4	0.6	**95.0**	82.7	91.5	93.4
Medium-Replay	Walker2d	36.9	9.7	0.9	77.2	66.6	82.6	**83.4**
Average		47.7	47.8	36.9	77.6	74.7	78.9	**80.4**

Table 1: Performance on D4RL datasets. We report the result for three random seeds.

DEDS consistently outperforms Decision Transformer (DT) and Trajectory Transformer (TT). This performance advantage is particularly evident in the high-dimensional Adroit environments, which require agents to handle complex state and action spaces effectively. Unlike simpler locomotion tasks, Adroit environments involve fine-grained manipulation with a five-fingered robotic hand, leading to greater sensitivity to irrelevant or redundant input signals. This makes representation learning especially difficult, as spurious features can easily degrade performance if not properly filtered. By filtering out task-irrelevant features, DEDS enables more focused and efficient policy learning. The agent is better equipped to learn stable and high-quality policies, even when trained on noisy or partially suboptimal offline trajectories.

Dataset	Environment	DT	TT	DEDS
Expert	Pen	115.0	117.2	**120.3**
Expert	Door	104.7	102.8	**106.0**
Medium-Replay	Pen	89.7	90.4	**93.8**
Medium-Replay	Door	71.6	73.6	**75.1**
Average		95.3	96	**98.8**

Table 2: Performance on Adroit datasets. This table presents a comparison of DT, TT, and DEDS across Expert and Medium-Replay datasets for Pen and Door tasks, with average performance reported over three random seeds.

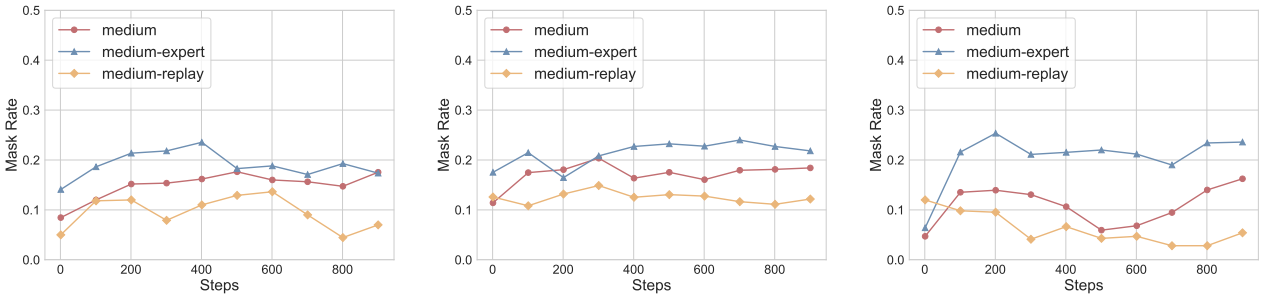
5.4 Mask Rate (RQ2)

To assess the efficiency of the state mask model, we compute the mask rate across complete episodes generated by policies trained on the D4RL datasets. The mask rate is defined as $|1 - I|/dim(I)$, where $dim(I)$ represents the dimensionality of the mask vector, and $1 - I$ denotes the count of masked dimensions in the state space. The data is collected from three independent random seeds to account for variability in the results. The mean mask rate, representing the average proportion of task-irrelevant state features removed by the state mask model, is computed and reported for each environment in Table 3. This metric provides a quantitative measure of the model's ability to filter out non-essential features, offering insight into its capacity to enhance the efficiency of the RL process, particularly in environments with high-dimensional state spaces.

The results indicate that the state mask model significantly improves data efficiency by effectively filtering out task-irrelevant state features, reserving approximately 80% of the state features and achieving a data efficiency improvement of around 20%. Additionally, we observed a correlation between mask rate and dataset quality, with higher mask rates in Medium-Expert datasets and lower rates in Medium and Medium-Replay datasets. This pattern likely arises because high-quality datasets, such as Medium-Expert, provide clear and consistent decision-making signals, whereas other datasets include random or partially trained behaviors, making it more difficult to isolate task-relevant features.

To analyze how the mask rate changes over time, we compute the mean mask rate at 100-step intervals, revealing consistent trends across datasets, the results are shown in Fig 4. A strong correlation emerges between dataset quality and mask rate: Medium-Expert datasets display the highest rates, reflecting the model's ability to filter out task-irrelevant features when decision-making signals are clear, whereas Medium and Medium-Replay datasets exhibit lower rates due to noise from suboptimal or exploratory policies. Notably, the mask rate remains relatively stable in Medium-Expert settings, indicating their consistent structure, while Medium and Medium-Replay show more fluctuation—especially early in each episode.

To visually illustrate the effectiveness of our approach in reducing state features while preserving interpretability, we generated visualizations of the mask rates for each state dimension across different environments, as shown in Figure 5.



(a) Hopper (b) Halfcheetah (c) Walker2d

Figure 4: Mask rate during episode. This figure illustrates how the mask rate changes over the course of an episode, calculated at 100-step intervals for three environments: Hopper, HalfCheetah, and Walker2d.

Dateset	Environment	Mask Rate(%)
Medium-Expert	HalfCheetah	21.4
Medium-Expert	Hopper	19.0
Medium-Expert	Walker2d	20.5
Medium	HalfCheetah	17.2
Medium	Hopper	14.9
Medium	Walker2d	10.8
Medium-Replay	HalfCheetah	12.5
Medium-Replay	Hopper	9.5
Medium-Replay	Walker2d	6.2

Table 3: Mask rate on D4RL datasets. This table shows the mask rate, which indicates the percentage of task-irrelevant state dimensions filtered out by the state mask model, for various environments and datasets within the D4RL benchmark.

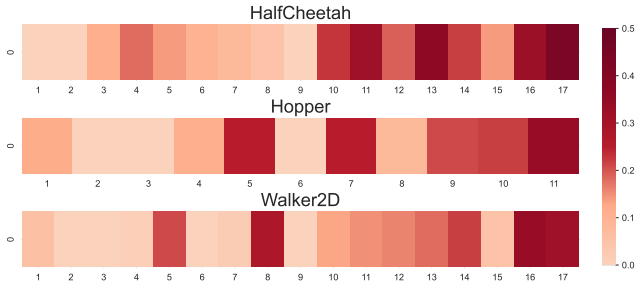


Figure 5: Mask rate for each state dimension. This figure shows the state dimension mask rates for three environments: Hopper (top), HalfCheetah (middle), and Walker2d (bottom). Darker colors represent higher mask rates.

5.5 Data Efficiency (RQ3)

For these experiments, we use the Expert subset of our trajectory data and evaluate DEDS, Decision Transformer (DT), and Trajectory Transformer (TT) on the Door and Pen tasks, with dataset sizes ranging from 0.5% to 95% of the original expert dataset. As shown in Figure 3, all methods improve with more data, but DEDS maintains a clear advantage in the low-data regime. Its masking mechanism identifies and focuses on task-relevant features, enabling robust policy learning even with limited samples. In contrast, DT and TT, relying on raw high-dimensional inputs without filtering, suffer significant performance declines in low-data settings.

Notably, DEDS also reserves a performance edge at higher data proportions, thereby reflecting the continued benefits of its adaptive masking strategy. Moreover, although the performance gap gradually narrows as more data is provided, DEDS's ability to dynamically focus on decision-critical information still allows it to better utilize the dataset and effectively avoid distractions from task-irrelevant or redundant features. Overall, this adaptability becomes particularly important in high-dimensional state spaces, where noisy inputs can otherwise dominate and significantly reduce learning efficiency.

6 Conclusions

In this work, we propose the Data Efficient Decision Sequence Model (DEDS) for offline RL. By integrating a state mask model guided by the bisimulation metric, DEDS dynamically filters out noisy or task-irrelevant features, enabling effective state representations for decision-making. Through experiments on MuJoCo tasks, we demonstrate that DEDS consistently outperforms existing approaches and achieves superior data efficiency. During our experiments, we found that the efficiency of DEDS decreases when facing high-dimensional action spaces, where an action might be represented by hundreds of tokens. We plan to address this issue more thoroughly in our future work.

Looking ahead, our study suggests that feature-level masking offers a promising path to improve data efficiency in sequence-based decision models. Extending DEDS to more complex benchmarks or real-world applications, such as robotic control, could yield valuable insights into its generalization and practical utility. Furthermore, integrating our approach with recent transformer advances may further boost both efficiency and generalization, positioning DEDS as a flexible framework for diverse sequential decision-making tasks.

Acknowledgements

This research is supported by Guangdong Basic and Applied Basic Research Foundation (2025A1515010247), the Fundamental Research Funds for the Central Universities (2024ZYGXZR069).

References

[1] Arthur Argenson and Gabriel Dulac-Arnold. 2020. Model-based offline planning. *arXiv preprint arXiv:2008.05556* (2020).

[2] Yuanying Cai, Chuheng Zhang, Wei Shen, Xuyun Zhang, Wenjie Ruan, and Longbo Huang. 2023. RePreM: representation pre-training with masked model for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 6879–6887.

[3] Pablo Samuel Castro. 2020. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 10069–10076.

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

[5] P Kingma Diederik. 2014. Adam: A method for stochastic optimization. *(No Title)* (2014).

[6] Norm Ferns, Prakash Panangaden, and Doina Precup. 2011. Bisimulation metrics for continuous Markov decision processes. *SIAM J. Comput.* 40, 6 (2011), 1662–1714.

[7] Norman Ferns and Doina Precup. 2014. Bisimulation Metrics are Optimal Value Functions.. In *UAI*. 210–219.

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

[9] Robert Givan, Thomas Dean, and Matthew Greig. 2003. Equivalence notions and model minimization in Markov decision processes. *Artificial intelligence* 147, 1-2 (2003), 163–223.

[10] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

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

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

[13] Kim G Larsen and Arne Skou. 1989. Bisimulation through probabilistic testing (preliminary report). In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 344–352.

[14] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2020. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*. PMLR, 5639–5650.

[15] Young Jae Lee, Jaehoon Kim, Young Joon Park, Mingu Kwak, and Seoung Bum Kim. 2024. Masked and Inverse Dynamics Modeling for Data-Efficient Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[16] Guoqing Liu, Chuheng Zhang, Li Zhao, Tao Qin, Jinhua Zhu, Jian Li, Nenghai Yu, and Tie-Yan Liu. 2021. Return-based contrastive representation learning for reinforcement learning. *arXiv preprint arXiv:2102.10960* (2021).

[17] Alec Radford. 2018. Improving language understanding by generative pre-training. (2018).

[18] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087* (2017).

[19] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. 2020. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929* (2020).

[20] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[21] Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. 2023. Masked trajectory models for prediction, representation, and control. In *International Conference on Machine Learning*. PMLR, 37607–37623.

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

[23] Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. 2024. Elastic decision transformer. *Advances in Neural Information Processing Systems* 36 (2024).

[24] Zhihui Xie, Zichuan Lin, Deheng Ye, Qiang Fu, Yang Wei, and Shuai Li. 2023. Future-conditioned unsupervised pretraining for decision transformer. In *International Conference on Machine Learning*. PMLR, 38187–38203.

[25] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. 2022. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*. PMLR, 24631–24645.

[26] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. 2021. Mastering atari games with limited data. *Advances in neural information processing systems* 34 (2021), 25476–25488.

[27] Tao Yu, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. 2022. Mask-based latent reconstruction for reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 25117–25131.

[28] Yang Yue, Bingyi Kang, Zhongwen Xu, Gao Huang, and Shuicheng Yan. 2023. Value-consistent representation learning for data-efficient reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11069–11077.

[29] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. 2020. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742* (2020).