Mamba as Decision Maker: Exploring Multi-scale Sequence Modeling in Offline Reinforcement Learning

Anonymous Authors[1]

Abstract

Sequential modeling has demonstrated remarkable capabilities in offline robotics reinforcement learning (RL), with Decision Transformer (DT) being one of the most notable representatives, achieving significant success. However, RL trajectories possess unique properties to be distinguished from the conventional sequence (*e.g.*, text or audio): (1) local correlation, where the next states in RL are theoretically determined solely by current states and actions based on the Markov Decision Process (MDP), and (2) global correlation, where each step's features are related to long-term historical information due to the time-continuous nature of trajectories. In this paper, we propose a novel action sequence predictor, named Mamba Decision Maker (MambaDM), where Mamba is expected to be a promising alternative for sequence modeling paradigms, owing to its efficient modeling of multi-scale dependencies. In particular, we introduce a novel mixer module that proficiently extracts and integrates both global and local features of the input sequence, effectively capturing interrelationships in RL datasets. Extensive experiments demonstrate that MambaDM achieves superior performance in D4RL and Robomimic datasets, including locomotion, manipulation, maze, and game tasks. Furthermore, we empirically investigate the scaling laws of MambaDM, finding that increasing model size does not bring performance improvement, but scaling the dataset amount by $2\times$ for MambaDM can obtain up to 33.7% score improvement on the Atari dataset. This paper delves into the sequence modeling capabilities of MambaDM in the robotics RL domain, paving the way for future advancements in robust and efficient decision-making systems.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Decision-making in Reinforcement Learning (RL) typically involves learning a mapping from state observations to actions that maximize cumulative discounted rewards. Recently, (Chen et al., 2021) introduced the Decision Transformer (DT), which leverages the simplicity and scalability of transformer architectures to establish a new paradigm for learning in offline RL. DT reinterprets the decision-making process as a sequence modeling problem, learning a return-conditioned state-action mapping. This approach predicts the necessary action to achieve a desired return based on a sequence of past returns, states, and actions. Due to its promising performance, this sequence modeling paradigm has been widely applied to various robotics tasks, including manipulation (Guhur et al., 2023; Shridhar et al., 2023), navigation (Shah et al., 2022), and behavior generation (Lifshitz et al., 2024).

However, RL trajectories differ from conventional sequences like text or audio, and thus modeling them cannot be treated as a simple sequence modeling task. RL problems are typically defined by a Markov Decision Process (MDP), where state transition probabilities adhere to the Markov property—meaning the probability of transitioning to the next state depends solely on the current state and action, not on past states. Consequently, local correlations between steps in the trajectory sequence are significant and cannot be ignored. Additionally, since time steps are sequential, each step's features are related to long-term history information, indicating that RL trajectories also exhibit inner global correlations. Therefore, designing a multi-scale model to capture both global and local features of RL trajectories deserves further exploration.

In this paper, we introduce the Mamba Decision Maker (MambaDM), which provides an in-depth study of the effectiveness of the Mamba module within the framework of reward conditional sequence modeling. We innovatively

design the Global-local Fusion Mamba (GLoMa) module to capture both local and global features to better understand the inner correlations within RL trajectories, aiming to enhance model performance. Furthermore, we explore the scaling laws of MambaDM in RL tasks. Unlike in NLP, where larger models typically yield better performance, our findings indicate that, in OpenAI Gym environments, increasing model size does not necessarily enhance results. But providing a larger datasize for MambaDM can bring performance improvement. We also analyze the Mamba module's ability to capture dependency information by visualizing changes in the eigenvalues of Mamba's core transition matrix. Extensive experiments demonstrate that MambaDM achieves state-of-the-art performance across multiple tasks, significantly outperforming the Decision Transformer (DT (Chen et al., 2021)) in locomotion and manipulation tasks (*e.g.*, Adorit (Rajeswaran et al., 2017) and Robomimic (Mandlekar et al., 2021)), and exceeding the performance of state space model-based (Gu et al., 2021; Gu & Dao, 2023) DMamba (Ota, 2024) by up to 75.3%.

Our contributions can be summarized as follows:

- We propose the Mamba Decision Maker, an effective decision-making method that incorporates a novel global-local fusion mamba module to model RL trajectories from both global and local perspectives.
- We investigate the scaling laws of MambaDM in RL tasks. Our experimental results indicate that the scaling laws of MambaDM do not completely align with those observed in NLP, suggesting that gathering larger and more diverse RL datasets could be a more effective strategy for enhancing performance compared to merely increasing the model size.
- Extensive experiments validate that MambaDM achieves superior results in locomotion, manipulation, and maze tasks, significantly outperforming state space model-based DS4 and DMamba. Additionally, our visualization analysis demonstrates MambaDM's ability to capture both short-term and long-term dependencies, supporting the reliability of the proposed module.

2. Related Work

2.1. Offline Reinforcement Learning

Reinforcement Learning (RL) is a framework of algorithms that learn through interactions with an environment. The problem is formulated as a Markov Decision Process (Sutton & Barto, 1999; 1998). RL aims to maximize rewards through interactions with an environment, which is often a time-consuming and expensive process. Drawing inspiration from supervised learning, researchers have explored offline reinforcement learning as a means to circumvent the traditional paradigm of online interactions (Levine et al., 2020; Fujimoto et al., 2019). Compared to online RL, which involves interacting with the environment, offline RL more closely resembles a data-driven paradigm as it relies on the offline dataset. Researchers have developed various approaches to offline RL, including methods based on value functions (Fujimoto et al., 2019; Kumar et al., 2020), uncertainty quantification (Agarwal et al., 2020; Wu et al., 2021; Yu et al., 2020), dynamics estimation (Kidambi et al., 2020; Argenson & Dulac-Arnold, 2020; Depeweg et al., 2016; Swazinna et al., 2021) and conditional behavior cloning that avoid value functions (Ding et al., 2019; Ghosh et al., 2019; Lynch et al., 2020; Schmidhuber, 2019; Srivastava et al., 2019; Emmons et al., 2021). These investigations into offline RL have extensively explored the utilization of offline data, thereby stimulating further scholarly contemplation on leveraging such data for reinforcement learning applications.

2.2. Sequence Modeling for Offline Reinforcement Learning

As supervised learning continues to evolve, sequence models have achieved significant success in many research areas. Given the structural similarities between RL's Markov processes and sequence modeling, many researchers have begun to consider integrating sequence models with RL. Decision Transformer (DT) (Chen et al., 2021) employs a Transformer to convert an RL problem into a sequence modeling task and treats trajectories as sequences. The DT was the first to combine the transformer structure with offline RL. Max et al. (Siebenborn et al., 2022) discuss the role of the attention mechanisms in DT, offering various opinions and motivating us to employ advanced sequence modeling techniques. Decision Convformer (Kim et al., 2023) incorporates discussions from the MetaFormer (Yu et al., 2022) and proposes considerations for the use of local features. However, the structure in DC focuses too much on local information and structures, which can impact the utilization of global information. Decision S4 (David et al., 2022) uses S4 layers for sequence modeling and Decision Mamba (Ota, 2024) integrates the latest Mamba sequence model. However, both methods merely apply these new sequence models directly without further exploring and utilizing the potential of these sequence models.

In this paper, we introduce MambaDM, an innovative sequence modeling framework designed for offline RL. MambaDM integrates the unique features of state space models to effectively combine local and global features, enhancing learning capabilities and efficiently handling larger-scale training. Extensive experiments in Section 5 across manipulation, locomotion, maze, and RL games demonstrate the effectiveness of our method.

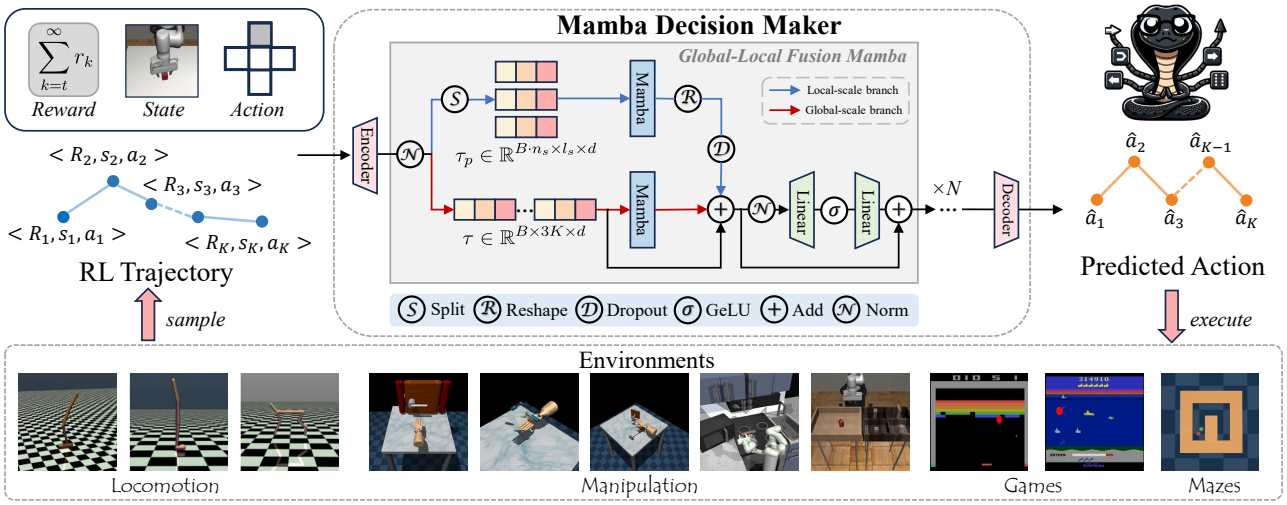


Figure 1. **Overview of our proposed MambaDM.** The input RL trajectory is first processed by the embedding layers, and these embeddings are then passed through both local and global branches to extract multi-scale features. Subsequently, the combined information from these branches is fed into a feed-forward network (FFN). After passing through N layers, the final action sequence is obtained by the action predictor.

3. Preliminaries

3.1. RL Problem Setup

An RL problem can be modeled as a Markov decision process (MDP) $\mathcal{M} = \langle \rho_0, \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$, where ρ_0 represents the initial state distribution, $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. At timestep t, a specific state and action are denoted by s_t and a_t, respectively. The transition probability $P(s_{t+1}|s_t, a_t)$ adheres to the Markov property. The reward function $\mathcal{R}$ provides the reward $r_t = \mathcal{R}(s_t, a_t)$, and $\gamma \in (0, 1)$ is the discount factor. We utilize the return-to-go (RTG, *i.e.*, $\hat{R}_t = \sum_{k=t}^{\infty} r_k$) as the reward input, following the approach of the Decision Transformer. Given an offline RL trajectory dataset, our objective is to determine an optimal policy π^* that predicts the best actions to maximize the expected return by interacting with the environment.

3.2. State Space Model

The state space models (SSMs) are a recent class of sequence models inspired by a particular dynamic system. SSMs map a 1-dimensional (1D) input $x(t) \in \mathbb{R}$ to the output signal $y(t) \in \mathbb{R}$ through an implicit latent $h(t) \in \mathbb{R}^N$. Concretely, we provide a detailed definition of structure state space models (S4) to illustrate the modeling process:

$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t), \tag{1}$$
$$y(t) = \boldsymbol{C}h(t) + \boldsymbol{D}x(t), \tag{2}$$

where $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ controls the whole continuous system, and $\boldsymbol{A}$ acts as an important state matrix which strongly decides the ability to capture long-range dependencies by different HIPPO initialization (Gu et al., 2020). To operate discrete sequences, S4 needs to be transformed into a discretized form:

$$h'(t) = \overline{\boldsymbol{A}}h(t) + \overline{\boldsymbol{B}}x(t), \tag{3}$$
$$y(t) = \boldsymbol{C}h(t), \tag{4}$$

where S4 uses the zero-order hold (ZOH) discretization rule: $\overline{\boldsymbol{A}} = \exp(\Delta \boldsymbol{A})$ and $\overline{\boldsymbol{B}} = (\Delta \boldsymbol{A})^{-1}(\exp(\Delta \boldsymbol{A}) - \boldsymbol{I})(\Delta \boldsymbol{B})$. Δ represents the step size used to discretize the continuous-time state-space model when applied to a discrete input sequence. $\boldsymbol{D}$ can be considered as parameter-based skip connections, thus we simplify $\boldsymbol{D}$ as 0. After transforming from $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}) \mapsto (\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})$, the model can be computed in two ways: (a) linear recurrence view which only needs $O(1)$ complexity during inference, and (b) global convolution view that achieves fast parallelizable speed during training. This deformable nature gives SSM a remarkable lightweight advantage over classical sequence models such as transformer (Vaswani et al., 2017) and RWKV (Peng et al., 2023) in NLP tasks.

3.3. Mamba

To further enhance selectivity and context-aware capabilities, Mamba (Gu & Dao, 2023) is proposed to handle complex sequence tasks by expanding the embedding space while maintaining efficient implementation. Building on the S4 structure, Mamba transitions the main parameters from time-invariant to time-varying. Additionally, the architecture of the Mamba block integrates elements from both H3 (Fu et al., 2022) and gated MLP (Touvron et al., 2023; Chowdhery et al., 2023), making it versatile for in-

corporation into neural networks. Recent extensive research demonstrates that Mamba significantly contributes to various tasks, including but not limited to vision (Zhu et al., 2024; Liu et al., 2024b; Teng et al., 2024), language (Pióro et al., 2024; Anthony et al., 2024), and robotics (Liu et al., 2024a; Cao et al., 2024) domains.

4. Methodology

4.1. Overview

The overview of our method is presented in Figure 1. We propose a global-local fusion mamba framework where the input trajectory is processed jointly by multi-scale branches. This design ensures that both global and local features are effectively leveraged for optimal performance.

4.2. Motivation and Insights

As mentioned before, RL trajectories differ from conventional sequences with two unique properties: (1) local correlations and (2) global correlations. In this paper, we adopt the Mamba model (Gu & Dao, 2023), known for its efficient sequence modeling and strong ability to capture multi-scale dependencies (Gu et al., 2021), to explore its capability to model the inner relationship of RL trajectories. We develop the global-local fusion mamba module to effectively integrate local features and global features. Inspired by transformer-in-transformer (TNT (Han et al., 2021)), which aims to extract finer-grained features, GLoMA focuses on fusing global and local features within trajectories, differing from TNT's focus on combining with more granular image texture features.

4.3. Mamba Decision Maker

Global-local fusion Mamba (GLoMa). Given a sequence of trajectories τ_{raw} sampled from dataset D with raw RTG, state, and action, we first obtain the trajectory representation τ through the token embedding layer:

$$\tau = \text{Emb}(\tau_{\text{raw}}) = \big(\langle \hat{R}_1, s_1, a_1 \rangle, ..., \langle \hat{R}_K, s_K, a_K \rangle\big) \in \mathbb{R}^{3K \times d}, \tag{5}$$

where K is the context length, d is the embedding dimension. Then, the trajectory sequence τ is processed by both the global and local branches. For the local branch, the input sequence is evenly divided into sub-sequences, each of length l_s. Since the total sequence length is fixed with $3K$, it might lead to uneven sub-sequence lengths. To address this, we pad zeros at the end of the sequence to ensure uniform length and generate the sub-sequences through:

$$l_p = \big(l_s - (3K \bmod l_s)\big) \bmod l_s, \tag{6}$$

$$\tau_{\text{p}} = g_s(\text{Pad}(\tau, l_p), l_s) \in \mathbb{R}^{n_s \times l_s \times d}, \tag{7}$$

where l_p denotes the padding length, $n_s = \frac{3K + l_p}{l_s}$ denotes the number of sub-sequences. $\bmod$ and Pad means the modulo and padding operator, respectively. $g_s(\cdot)$ denotes the split function, which splits the input into l_s-length sequences. The n_s dimension will be merged with the batch dimension during training. We can then obtain the local-scale feature of sub-sequences through the Mamba module by Equation 8. For the global branch, we model the entire sequence directly using the Mamba block with Equation 9. Overall, this multi-scale process can be formulated by:

$$f_{\text{local}} = \text{Mamba}(\text{LN}(\tau_{\text{p}})), \tag{8}$$

$$f_{\text{global}} = \text{Mamba}(\text{LN}(\tau)), \tag{9}$$

where LN denotes the layer normalization. The multi-scale feature f_{c} is combined with a Dropout layer (Srivastava et al., 2014) and then fed into an FFN layer to obtain the output features:

$$f_{\text{c}} = f_{\text{global}} + \text{Dropout}(f_{\text{local}}), \tag{10}$$

$$f = \text{FFN}(\text{LN}(f_{\text{c}})) + f_{\text{c}}. \tag{11}$$

After passing through L layers of the network, the final features f_L are input into the prediction head to predict the action sequence: $\hat{a}_t = \text{Head}(f_L)$ with $t = 1, 2, ..., K$.

By implementing these techniques, our Global-local fusion mamba framework aims to effectively capture and integrate both local and global features to better understand the inner correlations within RL trajectories, enhancing the model's ability to predict action sequences accurately.

Training and Inference. In the training stage, considering the true-optimal action as a_t^*, the training object is to minimize the loss between the predicted action $\hat{a}_t$ and true action a_t^*:

$$\mathcal{L} = \begin{cases} \mathbb{E}_{\tau_{\text{raw}} \sim D}\big[\|a_t^* - \hat{a}_t\|^2\big], & \text{if action is continuous;} \\ \mathbb{E}_{\tau_{\text{raw}} \sim D}\big[-a_t^* \log(\hat{a}_t)\big], & \text{if action is discrete.} \end{cases} \tag{12}$$

In this paper, the actions in the Atari benchmark (Mnih et al., 2013) are discrete, whereas those in D4RL (Fu et al., 2020) are continuous. During inference, we concatenate trajectories up to a fixed length $3K$ (the same as used in training) as the model input. If the available sequence is shorter than $3K$, we apply zero-padding. We set K as 20 and 30 in D4RL tasks and Atari games, respectively. l_s is set as 5. Additionally, the true RTG is unavailable. Therefore, following (Chen et al., 2021), we set the target RTG to be the initial RTG. We find that the initial RTG significantly affects the final results, and we discuss this phenomenon in Section 5.9.

Scaling Factor	Context Length	Layer Number	Embedding Dimension	Dataset Size
Range	(10, 30, 50, 80, 100)	(3, 6, 8, 12)	(128, 256, 512)	(0.25M, 0.5M, 1M)

Table 1. **Experimental settings for scaling factors.**

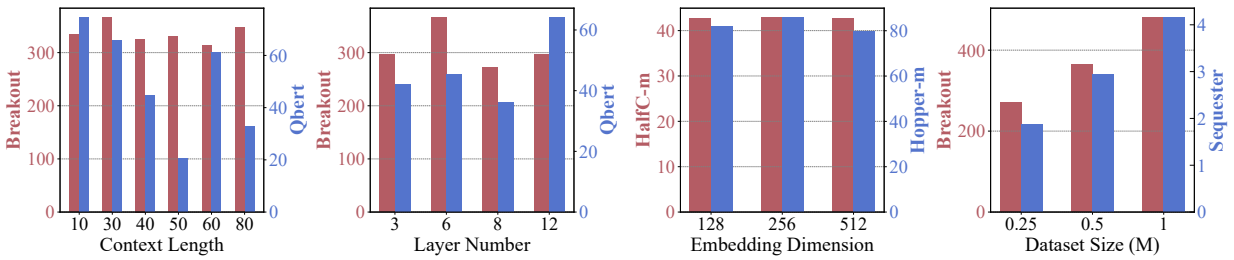


Figure 2. **Visualization of the scaling factors impact on MambaDM's performances in reinforcement learning tasks**, where color red and blue denote different RL domains. We find that MambaDM does not demonstrate a definitive scaling law when scaling the model size, but increasing the dataset size can significantly improve the model's performance.

5. Experiments

5.1. Exploring the Scaling Laws of MambaDM

In this section, we investigate the scaling laws of MambaDM within the context of reinforcement learning (RL) tasks. Scaling laws (Kaplan et al., 2020), as observed in the natural language processing (NLP) domain, suggest that increasing computational resources generally leads to improved performance. Our objective is to explore if similar scaling laws apply to MambaDM when used for RL tasks.

Experiment Setting. To systematically study this, we conduct a series of experiments varying key parameters of MambaDM. Specifically, we manipulate the context length, layer number, embedding dimension, and dataset size. The settings for these parameters are chosen to cover a broad range of values and to provide a comprehensive understanding of their impact on performance. The details of these settings are illustrated in Table 1.

Discussion. In our investigation of the scaling law in MambaDM for RL tasks, we explore how variations in scaling factors affect performance, as depicted in Figure 2. Our findings indicate that MambaDM does not exhibit NLP-like scaling behaviors in Atari and OpenAI Gym. As we increase the model size, performance fluctuations are observed instead of a clear upward trend. Consequently, **MambaDM does not demonstrate a definitive scaling law when scaling the model size**. However, our experiments show that **increasing the dataset size can significantly improve the model's performance**. This suggests that focusing on gathering larger and more diverse RL datasets could be a more effective strategy for enhancing performance compared to merely increasing the model size in Atari and OpenAI Gym.

We delve into the reasons why MambaDM does not exhibit an NLP-like scaling law with several points: (1) The model size is strongly correlated with its capability, but the optimal model size varies across different RL tasks. Our experiments are limited to two standard RL datasets, which may not provide a comprehensive understanding of scaling behavior across diverse RL tasks. (2) RL trajectories possess different attributes compared to text data, despite Mamba showing scaling law behaviors in NLP tasks (Gu & Dao, 2023). (3) Increasing the dataset size enhances model performance. This empirical conclusion aligns with findings from existing studies (Gao et al., 2023; Bhargava et al., 2024). Overall, exploring scaling law in RL remains a worthwhile direction.

5.2. Evaluations on Locomotion Tasks

We adopt the same model hyperparameter settings as those used in DT and DC for fair comparisons. We conduct experiments in three different MuJoCo-locomotion domains (Fu et al., 2020): halfcheetah, hopper, and walker, where each environment has three quality levels, including medium (M), medium-replay (M-R) and medium-expert (M-E). Table 2 presents a performance comparison of MambaDM against several baselines in the D4RL benchmark. MambaDM consistently demonstrates superior performance, often securing top-1 and top-2 rankings in various environments. For instance, in the hopper (M) dataset, MambaDM achieves an average return of 85.7, significantly outperforming DT (68.4) and DS4 (54.2). These results underscore MambaDM's ability to consistently outperform or closely match the top-performing baselines across diverse datasets and tasks, demonstrating its reliability and robustness in continuous RL environments.

5.3. Evaluations on Manipulation Tasks

We conduct experiments in three Adroit manipulation tasks (Rajeswaran et al., 2017): Pen, Hammer, and Door; three Franka Kitchen (Rajeswaran et al., 2017) with different levels; and two Robomimic (Mandlekar et al., 2021)

Dataset	Env.	TD3+BC	IQL	CQL	RvS	DT	DS4	DMamba	**MambaDM**
M	halfcheetah	48.3	47.4	44.0	41.6	42.6	42.5	**42.8**	**42.8** ±0.1
	hopper	59.3	63.8	58.5	60.2	68.4	54.2	83.5	**85.7** ±7.8
	walker2d	83.7	79.9	72.5	71.7	75.5	78.0	**78.2**	**78.2**±0.6
M-R	halfcheetah	44.6	44.1	45.5	38.0	37.0	15.2	**39.6**	39.1 ±0.1
	hopper	60.9	92.1	95.0	73.5	85.6	49.6	82.6	**86.1** ±2.5
	walker2d	81.8	73.7	77.2	60.6	71.2	69.0	70.9	**73.4** ±2.6
M-E	halfcheetah	90.7	86.7	91.6	92.2	88.8	**92.7**	91.9	86.5 ±1.2
	hopper	98.0	91.5	105.4	101.7	109.6	110.8	**111.1**	110.5 ±0.3
	walker2d	110.1	109.6	108.8	106.0	**109.3**	105.7	108.3	108.8 ±0.1

Table 2. **Comparison results of MambaDM and baselines in D4RL benchmark with Medium (M), Medium-Reply (MR) and Medium-Expert (ME) datasets.** We report the expert-normalized returns, following (Fu et al., 2020), averaged across three random seeds. The top-1 and top-2 results among the DT variants are highlighted in deep pink and light pink, respectively.

Environment	BC	SAC-off	BCQ	PLAS	DT	DMamba	**MambaDM**
Pen	34.4	6.3	68.9	67.3	77.8 ±2.3	74.3 ±19.4	**79.5** ±9.6
Hammer	1.5	0.5	0.5	4.6	3.7 ±0.4	2.2 ±0.8	**5.8** ±3.9
Door	0.5	3.9	0.0	4.4	7.5 ±4.7	12.0 ±6.3	**13.9** ±4.3
Kitchen-Complete	33.8	15.0	8.1	34.8	46.5 ±3.0	44.4 ±3.5	**48.7** ±5.2
Kitchen-Partial	33.8	0.0	18.9	43.9	31.4 ±19.5	55.7 ±4.6	**61.4** ±12.1
Kitchen-Mixed	47.5	2.5	8.1	40.8	25.8 ±5.0	42.2 ±13.3	**50.5** ±9.2

Table 3. **Comparison results of MambaDM and baselines in the Adroit and Franka Kitchen datasets.**

Environment	BC	BC-RNN	BCQ	HBC	IRIS	DT	DMamba	**MambaDM**
Can	64.7	68.7	75.3	40.7	48.0	85.0 ±0.0	90.0 ±2.0	**91.0** ±2.0
Lift	65.3	70.7	91.3	47.3	96.0	95.0 ±2.0	96.0 ±0.0	**97.0** ±1.0

Table 4. **Comparison results in the Robomimic datasets.**

tasks: Can and Lift with the machine-generated data.

Our experimental results in Table 3 & 4 demonstrate that MambaDM achieves state-of-the-art performance across multiple robotic manipulation benchmarks. In the Adroit environments, MambaDM attains superior scores of 79.5 (Pen) and 5.8 (Hammer), outperforming strong baselines such as DT (77.8/3.7) and DMamba (74.3/2.2). The advantage persists in complex multi-task settings: on Kitchen-Complete and Kitchen-Partial, MambaDM reaches 48.7 and 61.4 respectively, significantly exceeding DT (46.5) and DMamba (55.7). For real-world sim tasks in Robomimic, MambaDM maintains its lead with 91.0 (Can) and 97.0 (Lift), showcasing consistent improvements over existing methods. These results highlight MambaDM's robustness in handling diverse challenges, from high-precision manipulation to sparse-reward and partial-observability scenarios, establishing it as a powerful approach for offline RL in robotics.

5.4. Evaluations on Maze Tasks

We evaluate different approaches on the D4RL Maze environments, including umaze-v2 and umaze-diverse-v2. As shown in Table 5, MambaDM achieves competitive or superior performance compared to baseline methods. For instance, in umaze-diverse-v2, MambaDM reaches 86.0%, substantially outperforming DT (51.2) and DMamba (70.7). In umaze-v2, although CMD attains the highest return, MambaDM still demonstrates robustness with a strong score of 78.7, surpassing the baselines focusing on network design, *e.g.*, DT and DMamba. These results highlight the effectiveness of MambaDM in handling both standard and diverse maze settings.

Env.	TD InfoNCE	CMD	QRL	DT	DMamba	**MambaDM**
umaze-v2	84.9 ±1.2	90.3 ±4.2	76.8 ±2.3	65.6	75.3 ±10.0	**78.7** ±9.5
umaze-diverse-v2	91.7 ±1.3	90.3 ±4.6	80.1 ±1.3	51.2	70.7 ±12.0	**86.0** ±3.0

Table 5. **Comparison results in the umaze environment.**

Game	CQL	BC	DT	DC	$\text{DC}^{\text{hybrid}}$	DMamba	**MambaDM**
Breakout	211.1	142.7	242.4 ±31.8	352.7 ±44.7	**416.0** ±105.4	239.2 ±26.4	365.4 ±20.0
Qbert	104.2	20.3	28.8 ±10.3	67.0 ±14.7	62.6 ±9.4	42.3 ±8.5	**74.4** ±8.4
Assault	73.2	62.1	52.1±36.2	73.8 ±20.3	79.0±13.1	67.2±6.9	**81.4**±3.1
Gopher	2.8	33.8	34.8 ±10.0	52.5±9.3	51.6±10.7	27.0±3.9	**54.4**±11.1

Table 6. **Comparison results of MambaDM and baselines in the Atari games.** The gamer-normalized returns are reported, following (Ye et al., 2021), and averaged across three random seeds.

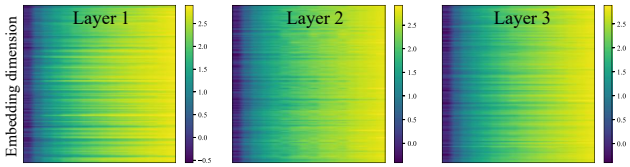
5.5. Evaluations on Atari Games

As illustrated in Table 6, our proposed MambaDM has a significant improvement across different Atari games. For instance, MambaDM's performance on Qbert significantly improved by 45.6% and 32.1% compared with DT and DMamba. When compared with the latest state-of-the-art method DC, MambaDM achieves favorable gains across all games. These results demonstrate the effectiveness of our method in the discrete Atari domain

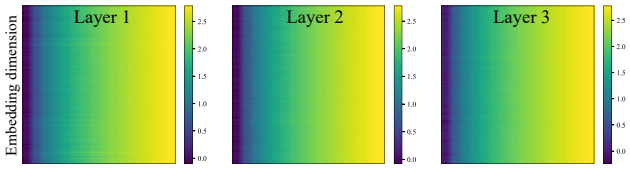
5.6. Analysis of the Learned Dependencies in MambaDM

It is important to note that in state-space models, $\overline{\boldsymbol{A}}$ is defined as a transition matrix parameterized by $\boldsymbol{A}$ and Δ, with $\boldsymbol{A}$ playing a critical role in influencing the stability of the dynamic system (Gu et al., 2021; 2020). The magnitude of the eigenvalues in $\boldsymbol{A}$, $|\lambda_j|$, is pivotal in determining the range of dependencies captured by the model. If $|\lambda_j| \geq 1$, the model can retain historical information across numerous time steps, effectively capturing long-range dependencies. Conversely, when $|\lambda_j|$ is significantly less than 1, the historical information decays rapidly, leading to a focus on short-term dependencies.

We visualize the key transition matrix $\boldsymbol{A}$ in the Mamba block under the Atari domain. For simplified visualization, the eigenvalues of $\boldsymbol{A}$ are presented in log-scale. Our analysis reveals two key findings: (1) The distribution of eigenvalues in the Global Mamba varies significantly across different embedding dimensions. As the layers deepen, there is a noticeable trend of increasing eigenvalues, indicating enhanced preservation of long-range information, indicating that long-range information is progressively preserved and accentuated as the network depth increases. (2) The eigenvalues in the Local Mamba exhibit a stable distribution across different embedding dimensions and layers, suggest-



(a) Matrix **A** in Global Mamba



(b) Matrix **A** in Local Mamba

Figure 3. **Visualization of the eigenvalues in the core matrices $\boldsymbol{A}$ of our proposed GLoMA**, including eigenvalues in (a) global mamba and (b) local mamba matrices.

Method	Breakout	Pen	Kitchen	Can
CMC	330.3 ±78.8	77.3 ±8.0	47.4 ±3.8	86.0 ±4.0
PMC	298.6 ±81.7	72.6 ±2.5	48.5 ±1.4	87.0 ±6.0
GLoMa (ours)	**365.4** ± 20.0	**85.8** ±6.0	**49.3** ±3.2	**91.0** ±2.0

Table 7. **Exploration on Mamba module combinations.** We set up three kinds of mamba methods, including Cascaded Mamba-Conv (CMC), Parallel Mamba-Conv (PMC), and our proposed GLoMa.

ing that the Local Mamba effectively captures both short-range and long-range dependencies, maintaining a balanced weight distribution between them. In summary, Figure 3 illustrates how different scale (*i.e.*, global or local) blocks handle varying ranges of dependencies. This highlights the complementary roles of global and local Mamba blocks in processing and preserving multi-scale information within the model.

5.7. Exploration on Mamba Module Combinations

To design a robust and effective module, we implemented two methods with different embedding configurations (cascaded/parallel) and sub-module choices (mamba/convolution). The visualization of the module design is in Figure 4. Results in Table 7 show that GLoMa significantly outperforms both PMC (Figure 4b) and CMC (Figure 4c) on Breakout, Pen, Kitchen, and Can tasks, confirming the robustness of our GLoMa design. The analysis of module design is discussed as follows.

Our model is designed with a parallel architecture that includes a global branch and a local branch. The global branch extracts global information within the RL trajectory, while the local branch focuses on local information based on MDP. To better validate the effectiveness of our design, we tried some other reasonable modules. As claimed in the DC (Kim et al., 2023), due to the MDP, the current and previous features are critical and convolutional operations can effectively extract such local information. Therefore, we replaced our local branch with the convolutional module from DC to create the Parallel Mamba-Conv. However, as shown in Table 7, PMC did not yield satisfactory results, demonstrating that the convolutional operation is not suited to extract local features when cooperating with Mamba. Additionally, considering that the processing order may affect performance, we designed a coarse-to-fine cascaded Mamba-Conv structure. This approach aimed to first extract global features and then refine local features, but the results for CMC were also suboptimal. Our proposed GLoMa module is specifically tailored for capturing the global and local correlations to better understand the inner relationships in RL trajectories. Therefore, our module design is well-justified, and the results in the main text indicate that our model achieved the best performance, thereby demonstrating its effectiveness.

Method	Training	Inference	Module Complexity	Param. (M)	Expand size	FLOPS (G)	Time (s / batch)
DT	$O(L^2)$	$O(L)$	Attention $O(L^2D + LD^2)$	0.73	-	0.09	0.11
DMamba	$O(L)$	$O(1)$	Selective Scan $O(NLD)$	0.88	2	0.04	0.10
MambaDM	$O(L)$	$O(1)$	Selective Scan $O(NLD)$	0.88	1	0.04	0.12

Table 8. **Comparisons of computation results.** MambaDM exhibits superior efficiency to DT, while achieving the same complexity as DMamba.

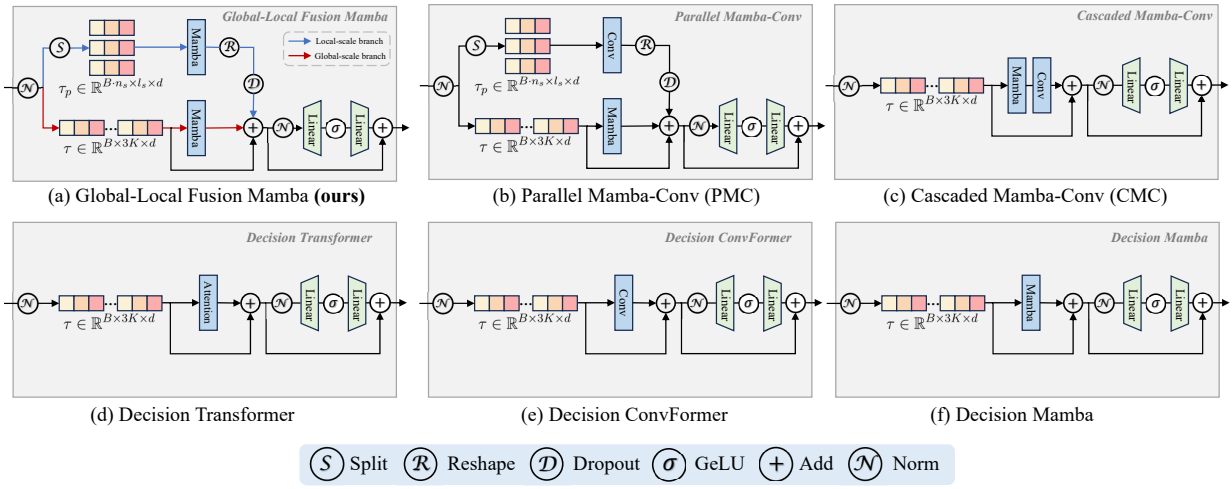
5.8. Comparisons on Model Complexity

We provide a detailed comparison of computational efficiency in Table 8. MambaDM achieves the same linear training and constant-time inference complexity as DMamba, while further reducing the expansion size without increasing parameters or FLOPs. Compared with DT, MambaDM exhibits superior efficiency on complexity and FLOPs. This results in lower overall computational cost, demonstrating efficiency gains. We emphasize, however, that our primary goal lies in leveraging Mamba for multi-scale sequence modeling rather than pursuing minimal complexity.

5.9. Ablation Studies

Ablations on Mamba initialization. As demonstrated by (Gu et al., 2020; 2021), the initialization of the state matrix $\boldsymbol{A}_n$ significantly influences the stability of overall performance, as the eigenvalues of $\boldsymbol{A}_n$ are closely related to the information transformation process. Consequently, we conduct experiments by varying the initial values of $\boldsymbol{A}_n$. The results presented in Table 9 indicate that our MambaDM exhibits stable performance across different initializations.

Evaluating with different context length. Mamba has demonstrated strong stability in handling long sequences within NLP tasks (Gu & Dao, 2023). To verify the stability of our proposed method, we evaluate the effect of context length K compared to DMamba. The results in Table 10 reveal that MambaDM not only maintains stable performance



(a) Global-Local Fusion Mamba **(ours)** (b) Parallel Mamba-Conv (PMC) (c) Cascaded Mamba-Conv (CMC)

(d) Decision Transformer (e) Decision ConvFormer (f) Decision Mamba

Figure 4. **Visualization of different module designs.**

A_n Initialization	Breakout	Pen	Kitchen	Can
$A_n = -1/2$	329.8 ±12.3	73.4 ±0.5	**52.1**±2.0	87.0±1.0
$A_n = -(n+1)$	**365.4**±20.0	**85.8** ±6.0	49.3 ±3.2	**91.0** ±2.0

Table 9. **Ablation results of different initialization of Mamba matrix A_n.**

Context length	10	30	40	60	Mean
DMamba	231.6	239.2	295.9	271.1	259.5
MambaDM	334.4	365.4	325.7	313.2	**334.7 (+75.2)**

Table 10. **Ablation results of different context length K in Atari Breakout.**

		Breakout				Qbert				Pong			
Atari	Init RTG	45	90	900	1800	1000	2500	10000	14000	10	20	100	1000
	Score	194.5	300.3	**365.3**	337.0	24.4	25.4	**45.4**	38.6	87.2	106.2	**110.8**	107.6
		Halfcheetah-M				**Hopper-M**				**Walker2d-M**			
OpenAI Gym	Init RTG	2500	5000	10000	50000	1800	3600	7200	36000	6000	12000	24000	240000
	Score	75.3	77.3	**78.2**	76.9	58.0	81.2	84.9	**85.7**	42.6	**42.8**	42.7	**42.8**

Table 11. **Ablation results of MambaDM with different initialized RTG.**

across various context lengths K, but also achieves significantly improved results, with an average increase of 75.2 points compared to DMamba.

Ablations on different initialized RTG. Table 11 explores the impact of different initial return-to-go (RTG) settings on the final performance across various datasets. The results indicate that, generally, as the initial RTG increases, the overall performance of the model improves. However, an excessively high initial RTG can negatively affect results (*e.g.*, in Breakout). Overall, the performance of the model is highly correlated with the initial RTG, showing a positive relationship within a certain range. This suggests that while increasing RTG can enhance performance, there is a threshold beyond which the benefits diminish or even reverse. Efficiently determining the optimal RTG setting to avoid extensive experimentation and to enable the model to achieve the best performance is a pertinent area for future research.

6. Conclusion

In this study, we proposed the Mamba Decision Maker (MambaDM), a novel action predictor designed to capture multi-scale features and better understand the correlations within RL trajectories. MambaDM leverages a unique global-local fusion mamba (GLoMa) mixer module that adeptly integrates global and local features of input sequences. Extensive experiments have demonstrated that MambaDM achieves state-of-the-art performance on several robotics benchmarks. Moreover, our investigation into the scaling laws of MambaDM reveals that increasing the dataset size results in significant performance improvements, underscoring the importance of dataset size over model size in enhancing performance under Atari and OpenAI Gym. In summary, MambaDM presents a promising alternative for sequence modeling in RL, offering efficient multi-scale dependency modeling and paving the way for future advancements in the field.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *ICML*, pp. 104–114. PMLR, 2020.

Anthony, Q., Tokpanov, Y., Glorioso, P., and Millidge, B. Blackmamba: Mixture of experts for state-space models. *arXiv preprint arXiv:2402.01771*, 2024.

Argenson, A. and Dulac-Arnold, G. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

Bhargava, P., Chitnis, R., Geramifard, A., Sodhani, S.,

and Zhang, A. When should we prefer decision transformers for offline reinforcement learning? In *ICLR*, 2024. URL https://openreview.net/forum?id=vpV7fOFQy4.

Cao, J., Zhang, Q., Sun, J., Wang, J., Cheng, H., Li, Y., Ma, J., Shao, Y., Zhao, W., Han, G., et al. Mamba policy: Towards efficient 3d diffusion policy with hybrid selective state models. *arXiv preprint arXiv:2409.07163*, 2024.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 34:15084–15097, 2021.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *JMLR*, 24(240):1–113, 2023.

David, S. B., Zimerman, I., Nachmani, E., and Wolf, L. Decision s4: Efficient sequence-based rl via state spaces layers. In *ICLR*, 2022.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.

Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. Goal-conditioned imitation learning. *NeurIPS*, 32, 2019.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? *ICLR*, 2021.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Re, C. Hungry hungry hippos: Towards language modeling with state space models. In *ICLR*, 2022.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *ICML*, pp. 2052–2062. PMLR, 2019.

Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. *NeurIPS*, 33:1474–1487, 2020.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2021.

Guhur, P.-L., Chen, S., Pinel, R. G., Tapaswi, M., Laptev, I., and Schmid, C. Instruction-driven history-aware policies for robotic manipulations. In *CoRL*, pp. 175–187. PMLR, 2023.

Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., and Wang, Y. Transformer in transformer. *NeurIPS*, 34:15908–15919, 2021.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *NeurIPS*, 33:21810–21823, 2020.

Kim, J., Lee, S., Kim, W., and Sung, Y. Decision convformer: Local filtering in metaformer is sufficient for decision making. In *ICLR*, 2023.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *NeurIPS*, 33:1179–1191, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review. *and Perspectives on Open Problems*, 5, 2020.

Lifshitz, S., Paster, K., Chan, H., Ba, J., and McIlraith, S. Steve-1: A generative model for text-to-behavior in minecraft. *NeurIPS*, 36, 2024.

Liu, J., Liu, M., Wang, Z., An, P., Li, X., Zhou, K., Yang, S., Zhang, R., Guo, Y., and Zhang, S. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *NeurIPS*, 37:40085–40110, 2024a.

Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., and Liu, Y. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024b.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *CoRL*, pp. 1113–1132. PMLR, 2020.

Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Martín-Martín, R. What matters in learning from offline human demonstrations for robot manipulation. In *CoRL*, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ota, T. Decision mamba: Reinforcement learning via sequence modeling with selective state spaces. *arXiv preprint arXiv:2403.19925*, 2024.

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M., Derczynski, L., et al. Rwkv: Reinventing rnns for the transformer era. In *EMNLP Findings*, pp. 14048–14077, 2023.

Pióro, M., Ciebiera, K., Król, K., Ludziejewski, J., and Jaszczur, S. Moe-mamba: Efficient selective state space models with mixture of experts. *ICLR Workshops*, 2024.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Schmidhuber, J. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

Shah, D., Bhorkar, A., Leen, H., Kostrikov, I., Rhinehart, N., and Levine, S. Offline reinforcement learning for customizable visual navigation. In *NeurIPS Workshops*, 2022.

Shridhar, M., Manuelli, L., and Fox, D. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, pp. 785–799. PMLR, 2023.

Siebenborn, M., Belousov, B., Huang, J., and Peters, J. How crucial is transformer in decision transformer? *arXiv preprint arXiv:2211.14655*, 2022.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Sutton, R. S. and Barto, A. G. The reinforcement learning problem. *Reinforcement learning: An introduction*, pp. 51–85, 1998.

Sutton, R. S. and Barto, A. G. Reinforcement learning: An introduction. *Robotica*, 17(2):229–235, 1999.

Swazinna, P., Udluft, S., and Runkler, T. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104: 104366, 2021.

Teng, Y., Wu, Y., Shi, H., Ning, X., Dai, G., Wang, Y., Li, Z., and Liu, X. Dim: Diffusion mamba for efficient high-resolution image synthesis. *arXiv preprint arXiv:2405.14224*, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *NeurIPS*, 30, 2017.

Wu, Y., Zhai, S., Srivastava, N., Susskind, J., Zhang, J., Salakhutdinov, R., and Goh, H. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

Ye, W., Liu, S., Kurutach, T., Abbeel, P., and Gao, Y. Mastering atari games with limited data. *NeurIPS*, 34:25476–25488, 2021.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *NeurIPS*, 33:14129–14142, 2020.

Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., and Yan, S. Metaformer is actually what you need for vision. In *CVPR*, pp. 10819–10829, 2022.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.