
Is Mamba Compatible with Trajectory Optimization in Offline Reinforcement Learning?

Yang Dai¹ Oubo Ma² Longfei Zhang¹ Xingxing Liang^{1*}
Shengchao Hu³ Mengzhu Wang⁴ Shouling Ji² Jincai Huang¹ Li Shen^{5*}

¹Laboratory for Big Data and Decision, National University of Defense Technology

²Zhejiang University ³Shanghai Jiao Tong University

⁴Hebei University of Technology ⁵Shenzhen Campus of Sun Yat-sen University

{daiyang2000,zhanglongfei,liangxingxing,huangjincai}@nudt.edu.cn

{mob, sji}@zju.edu.cn; charles-hu@sjtu.edu.cn; {dreamkily,mathshenli}@gmail.com

Abstract

Transformer-based trajectory optimization methods have demonstrated exceptional performance in offline Reinforcement Learning (offline RL). Yet, it poses challenges due to substantial parameter size and limited scalability, which is particularly critical in sequential decision-making scenarios where resources are constrained such as in robots and drones with limited computational power. Mamba, a promising new linear-time sequence model, offers performance on par with transformers while delivering substantially fewer parameters on long sequences. As it remains unclear whether Mamba is compatible with trajectory optimization, this work aims to conduct comprehensive experiments to explore the potential of Decision Mamba (dubbed DeMa) in offline RL from the aspect of data structures and essential components with the following insights: (1) Long sequences impose a significant computational burden without contributing to performance improvements since DeMa's focus on sequences diminishes approximately exponentially. Consequently, we introduce a Transformer-like DeMa as opposed to an RNN-like DeMa. (2) For the components of DeMa, we identify the hidden attention mechanism as a critical factor in its success, which can also work well with other residual structures and does not require position embedding. Extensive evaluations demonstrate that our specially designed DeMa is compatible with trajectory optimization and surpasses previous methods, outperforming Decision Transformer (DT) with higher performance while using 30% fewer parameters in Atari, and exceeding DT with only a quarter of the parameters in MuJoCo.

1 Introduction

Offline Reinforcement Learning (Offline RL) [1] has gained significant attention due to its ability to learn strategies without interacting with the environment, which is particularly beneficial in situations where real-time interaction is expensive or risky [2–4]. With a static dataset, offline RL can be implemented through three distinct learning methods [5]: (1) model-based algorithm [6–8], (2) model-free algorithm [9–11], (3) trajectory optimization [12–16]. The first two methods require long-term credit assignment through the Bellman equation, leading to the "deadly triad" problem known to destabilize RL [17]. In contrast, trajectory optimization methods treat RL problems as sequence modeling problems to get better performance and generalization [12]. Most trajectory optimization methods rely on transformers, which perform credit assignment directly through the

*Corresponding authors: Li Shen and Xingxing Liang.

attention mechanism. By leveraging the powerful modeling capabilities of transformers, these methods outperform other offline RL algorithms [18–20].

The transformer attention mechanism [21], which allows the model to focus on the important part of the input sequence [22], has several downsides. The computational demands of the attention mechanism escalate quadratically with the input length, posing a significant constraint on its scalability [23–25]. Moreover, some studies [26, 27] suggest that the attention mechanism may not be the primary factor contributing to the effectiveness of transformers. This notion is also supported in offline RL, where [13] discovers that the attention mechanism of Decision Transformer (DT) does not capture local associations effectively, rendering it unsuitable for RL. Given these limitations, we are led to ponder if a more efficient mechanism with fewer parameters and greater scalability exists for offline RL. Recently, a series of state space models (SSMs) [28], particularly Mamba [29], have been proposed as potential solutions with the ability to scale linearly concerning the sequence length. In particular, Mamba introduces a selective hidden attention mechanism [30] for content-based reasoning and employs parallel scan to enhance computational efficiency, resulting in two approaches to employing Mamba in offline RL. The first is the Transformer-like Mamba, a direct substitution of the transformer [31–33] while the other is the RNN-like Mamba [34], achieving an inference speed with constant time complexity.

Few studies have explored the application of SSMs in offline RL, though they perform well in model-based algorithms [35, 36] and in-context RL learning [37]. Mamba is tailored for memory-required long-sequence tasks, whereas trajectory optimization methods typically utilize short segments during training and inference, as most RL tasks are modeled as Markov Decision Processes (MDPs), i.e. past information may not influence current decisions. Furthermore, due to the lack of a comprehensive investigation of the key component of Mamba, a question has arisen:

Whether Mamba is compatible with trajectory optimization?

In this work, we aim to undertake a thorough investigation and in-depth analysis to explore this question. Specifically, we focus on the data structures and the essential components in trajectory optimization. The extensive experiments provide strong support for the following key findings. (1) We explore the data structures with an analysis of sequence length and concatenating type. The former reveals that long input sequences present computational challenges without enhancing performance due to the hidden attention scores of DeMa evincing an exponential decay pattern. As a result, we opt for the Transformer-like DeMa as opposed to the RNN-like DeMa for efficiency and effectiveness. The latter finds concatenating in the temporal dimension is better for the Transformer-like DeMa. (2) The hidden attention mechanism plays a pivotal role in DeMa’s effectiveness and is compatible with the transformer’s post up-projection residual structure [38], enabling it to replace the attention layer directly and eliminating the need for position embedding. Extensive evaluations show that with a higher average score and nearly 30% fewer parameters, DeMa significantly outperforms DT in eight Atari games. Furthermore, in nine MuJoCo tasks, DeMa’s performance not only exceeds that of DT but does so with only one-fourth of the parameters, highlighting remarkable improvements in both performance efficiency and model compactness.

In the end, our main contributions can be summarized as follows:

1. We find the Transformer-like DeMa surpasses the RNN-like DeMa in both efficiency and effectiveness for trajectory optimization. Extensive experiments on sequence length and concatenating type show the impact of the input data, which guides the design of DeMa.
2. Through various ablation experiments, we discover that the hidden attention mechanism is the core component in DeMa and does not require position embedding. This finding enhances the effectiveness and efficiency of our Transformer-like DeMa.
3. With state-of-the-art performance on both MuJoCo and Atari, our Transformer-like DeMa significantly addresses the challenges posed by transformer-based trajectory optimization methods, particularly the issues of large parameter sizes and limited scalability.

2 Related Work

Offline RL. Offline RL is a data-driven RL paradigm in which the agent learns solely from a pre-collected dataset rather than through interaction with the environment [16]. Distribution shifts [11]

can severely impact performance when RL algorithms are deployed directly in offline environments, leading to significant degradation. To mitigate this problem, several methods have been introduced, which the study [5] categorizes into three primary approaches: (1) learning a dynamics model to generate additional training data (model-based algorithm) [6, 39], (2) learning a policy through a model-free approach by constraining unseen actions or incorporating pessimism into the value function (model-free algorithm) [10, 11, 40], and (3) trajectory optimization [12, 15]. The method of trajectory optimization is usually based on a causal transformer model and converts an RL problem to a sequence modeling problem [13]. It performs credit assignment directly through the attention mechanism in contrast to Bellman backups, thus modeling a wide distribution of behaviors, enabling better generalization and transfer [12].

Sequence Modeling in Offline RL. Following DT [12] and Trajectory Transformer (TT) [15], there has been an increasing trend in employing advanced sequence-to-sequence model to solve RL tasks [14, 41–46].¹ Unfortunately, these improvements are usually transformer-based and hence suffer from the common dilemma of the attention mechanism, i.e. over-parameterization and inability to scale to long sequence tasks. What’s more, Emmons et al. [48] find that simply maximizing likelihood with a two-layer feedforward MLP is close to the results of substantially more complex methods based on sequence modeling with Transformers. Similarly, Lawson et al. [49] find that replacing the attention parameters with those learned in other environments has a minimal impact on the performance. Besides, Decision ConvFormer (DC) [13] indicates that substituting the attention layers with learnable parameters can lead to improved outcomes. These observations suggest significant redundancy in the Transformer architecture, highlighting the potential to explore lighter and more scalable networks for implementation in offline RL. Building on this, the Structure SSM (S4) [50] has emerged as a promising alternative. Studies [35] and [36] use S4 in model-based RL, outperforming traditional Transformer and RNN approaches. The capabilities of S4 and Mamba are further demonstrated by [37, 51], which points to their speed and effectiveness in in-context RL tasks.

The most related work to ours is Decision S4 (DS4) [52] and Decision Mamba (DMamba) [53], where the former uses an RNN-like S4 for inference, and the latter replaces the attention mechanism with Mamba directly. In contrast, our work finds that Transformer-like DeMa outperforms RNN-like DeMa as the long sequences impose a significant computational burden on Mamba without contributing to performance improvements. What’s more, DMamba simply substitutes Mamba for the attention block rather than the transformer block while our investigation shows the key component is the hidden attention mechanism, which eliminates the need for position embedding and hence achieves better performance with fewer parameters.

3 Preliminaries

In this section, we present several necessary preliminaries and terminologies of offline RL, trajectory optimization, state space model, and hidden attention in Mamba.

3.1 Offline RL with Trajectory Optimization

Given a static dataset of transitions $\tau = \{(s_t, a_t, s_{t+1}, r_t)_i\}$, where i presents the timestep of a transition in the dataset. The states and actions are generated by the behavior policy $(s_t, a_t) \sim d^{\pi_\beta}(\cdot)$, while the next states and rewards are determined by the unknown transition dynamics $p(s', r|s, a)$. The goal of offline RL is to find an approximate policy $\pi(a|\cdot)$ that maximizes expected return $\mathbb{E}[\sum_{t=0}^T r_t]$, where T represents the time step at which the episode terminates. Due to the lack of interaction with the environment, trajectory optimization methods transform the goal into minimizing reconstruction loss, i.e. minimizing loss $\mathbb{E}_{(\hat{R}, s, a) \sim \tau} [\frac{1}{T} \sum_{t=1}^T \mathcal{L}_{\text{MSE/CE}}(\hat{a}_t; a_t)]$, where $\hat{a}_t = \pi(\cdot|s_{t-K+1:t}, \hat{R}_{t-K+1:t}, a_{t-K:t-1})$, and $\hat{R}_t = \sum_{t'=t}^T r_{t'}$ is the return-to-go (RTG). At test time, a target RTG R_0 is manually set to represent the desired performance. We input the trajectories from the last K timesteps into policy π , which then generates an action for the current timestep. Subsequently, the next state and reward are received from the environment. These elements are concatenated and also input into the model. The policy is approximated through the sequential model [12, 54]. However, these models typically possess a large number of parameters

¹For detailed insights, one may refer to the relevant comprehensive reviews [16, 47].

and struggle with handling long sequences effectively. Fortunately, this issue can be addressed by using SSMs [28, 50, 29].

3.2 State Space Model and Mamba

There are two approaches to utilizing Mamba in RL, which are both closely related to the modeling methods of SSM. SSM is defined by the following first-order differential equation, which maps a 1-D input signal $u(t)$ to an N -D latent state $h(t)$ before projecting to a 1-D output signal $y(t)$ [55],

$$h'(t) = Ah(t) + Bu(t), \quad y(t) = Ch(t) + Du(t), \quad (1)$$

where $A \in \mathbb{R}^{N \times N}$, $B \in \mathbb{R}^{N \times 1}$, $C \in \mathbb{R}^{1 \times N}$ and $D \in \mathbb{R}$ are trainable matrices. As $u(t)$ is typically discretized as $\{u_i\}_{i=1,2,\dots}$, SSM can be discretized by a step size Δ . Moreover, recurrent SSM can be written as a discrete convolution. Let $h_0 = 0$ and $D = 0$, we have

$$y_i = C\bar{A}^i\bar{B}u_1 + C\bar{A}^{i-1}\bar{B}u_2 + \dots + C\bar{A}\bar{B}u_{i-1} + C\bar{B}u_i, \quad y = u * \bar{K}, \quad (2)$$

where \bar{A}, \bar{B} is the approximation discrete of A, B , and \bar{K} is called the SSM convolution kernel and can be represented by filter

$$\bar{K} = (C\bar{B}, C\bar{A}\bar{B}, \dots, C\bar{A}^i\bar{B}, \dots). \quad (3)$$

S4 and other time-invariant models cannot select the previous tokens to invoke from their history records. To solve this problem, Mamba merges the sequence length and batch size of the inputs, allowing the matrices B, C and the step size Δ to depend on the inputs. Therefore, it is a time-varying system and cannot use the convolution view. To ensure efficient training and inference with Mamba, techniques such as parallel scanning, kernel fusion, and recomputation are employed, resulting in two types of Mamba. One type is the SSM using the recursive view, referred to as RNN-like Mamba, and the other is the SSM utilizing parallel scanning, known as Transformer-like Mamba. RNN-like Mamba is akin to DS4 [52], wherein the complete trajectory is taken as a sample and fully inputted into the model for training. Utilizing this approach, which capitalizes on the ability to capture long-term dependencies, the inference speed can be significantly increased. During the inference process, it is sufficient to input only the current tuple (r_{t-1}, a_{t-1}, s_t) in conjunction with the hidden state h_t . Transformer-like Mamba is a direct replacement for the transformer, where we consistently truncate the input sequences to a fixed length of K before their introduction into the model throughout the training and inference phases [53, 34, 56, 57].

3.3 Hidden Attention in Mamba

Although the role of the self-attention mechanism in offline RL remains uncertain, it is known that this mechanism allows the model to dynamically focus on different parts of the input sequences, following the Equation (4).

$$\text{Self-Attention}(x) = \alpha V(x), \quad \alpha = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right), \quad (4)$$

where Q, K, V represent queries, keys, and values respectively, i.e. input sequences after three linear transformations. d_k is the dimension of the keys. Similarly, current research suggests that the S6 layer in Mamba can be viewed as the hidden attention mechanism with a unique data-control linear operator [30]. Assuming the initial condition $h_0 = 0$, we can obtain a formula similar to Equation (2)

$$y_i = C_i \sum_{j=1}^i (\Pi_{k=j+1}^i \bar{A}_k) \bar{B}_j x_j, \quad h_i = \sum_{j=1}^i (\Pi_{k=j+1}^i \bar{A}_k) \bar{B}_j x_j, \quad (5)$$

where $\bar{A}_i = \exp(\Delta_i(A))$, $\bar{B}_i = \Delta_i(B_i)$, and $\Delta_i = \text{softplus}(S_\Delta(x_i))$. $B_i = S_B(x_i)$, $C_i = S_C(x_i)$, with S_B , S_C and S_Δ are linear projection layers. Softplus is an elementwise function that is a smooth approximation of ReLU.

Since \bar{A}_t is a diagonal matrix, [30] simplifies the hidden matrices and gets the attention mechanism of Mamba:

$$\begin{aligned} \text{Hidden-Attention}(x) &= \tilde{\alpha}x, \quad \tilde{\alpha}_{i,j} \approx \tilde{Q}_i \tilde{H}_{i,j} \tilde{K}_j \\ \tilde{Q}_i &:= S_C(x_i), \quad \tilde{K}_j := \text{ReLU}(S_\Delta(x_j) S_B(x_j)), \quad \tilde{H}_{i,j} := \exp \left(\sum_{\substack{k=j+1 \\ S_\Delta(x_k) > 0}}^i S_\Delta(x_k) \right) A. \end{aligned} \quad (6)$$

Therefore, we can visualize the hidden attention matrices in DeMa, thus gaining a deeper understanding of the behavior inside the model in the setting of offline RL.

4 The Analysis of DeMa

Considering most trajectory optimization methods use short segments during both training and inference, *the compatibility of Mamba with these methods remains an open question*. As shown in Figure 1, this section presents an analysis from the perspectives of data structures and essential components. Section 4.1 discusses the impact of data structure on trajectory optimization. Our study reveals that the RNN-like DeMa does not offer substantial benefits in terms of effectiveness or efficiency. Therefore, we investigate three critical factors: sequence length, the hidden attention mechanism, and the input concatenation types. We find that the balance between performance and efficiency highly depends on the appropriate sequence length selection. Moreover, the input concatenation method significantly influences the results, with temporal concatenation (i.e., B3LD) demonstrating its effectiveness. Section 4.2 conducts ablation studies to identify the hidden attention mechanism as a key component of DeMa, facilitating better utilization and component replacement. Detailed experiments and additional results are in the **Appendix**. Our code is available at <https://github.com/AndssY/DeMa>.

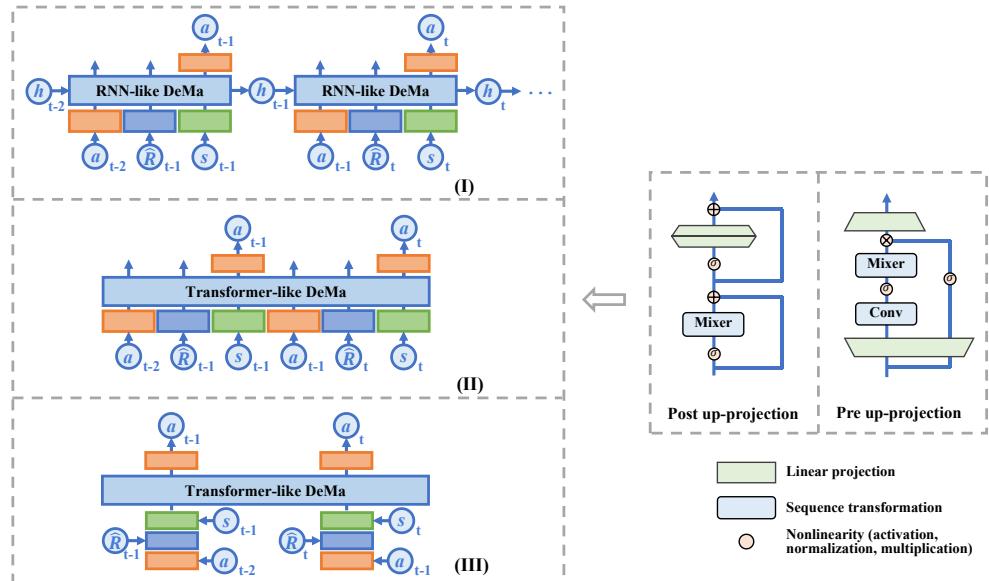


Figure 1: Variant design of the DeMa in trajectory optimization. In the left portion, (I) represents the RNN-like DeMa (B3LD), which requires hidden state inputs at each decision step; (II) indicates the transformer-like DeMa (B3LD); and (III) refers to the transformer-like DeMa (BL3D). The right portion illustrates that both types of these DeMa can incorporate two distinct residual structures, i.e. the post up-projection residual block and the pre up-projection residual block.

4.1 Input Data Structures

First, we compare the RNN-like DeMa (B3LD) with the Transformer-like DeMa (B3LD)². The average results are shown in Table 1 (with detailed results in Appendix E), where the performance of the RNN-like DeMa is significantly inferior to that of the Transformer-like DeMa, especially in Atari games. These findings suggest that the recurrent mode may be unnecessary in trajectory optimization methods. Given that the hyper-parameters are identical for both types of DeMa except for the sequence length, we assume that variations in sequence length are likely the primary cause of the observed disparities in results. Therefore, we explore the effect of sequence length on the Transformer-like DeMa in subsequent sections.

²BL3D and B3LD represent different concatenation types. Section 4.1 gives a comprehensive explanation. Unless otherwise specified, all references to DeMa in this context refer to the B3LD type.

Table 1: The average result of DT, RNN-like DeMa and Transformer-like DeMa in Atari [58] and MuJoCo [59]. The results are reported with the normalization following [60, 11]. Detailed results can be seen in Appendix E.

Env	DT	RNN-like DeMa	Transformer-like DeMa
Atari	62.2	67.3	111.8
MuJoCo	63.4	61.1	66.0

How does sequence length affect the computational load? We investigate the impact of sequence length on single-step training time, single-step inference time and GPU memory usage for models including DT, Transformer-like DeMa, and RNN-like DeMa. Figure 2 shows that the Transformer-like DeMa operates faster than the RNN-like DeMa when dealing with short sequence lengths, despite that the inference time of RNN-like DeMa is independent of the sequence length. With conventional sequence lengths (such as 20), Transformer-like DeMa holds an advantage in forward speed, training speed, and GPU memory consumption.



Figure 2: The impact of sequence length on single-step forward computation time, single-step training time, and GPU memory usage. The sequence length of RNN-like DeMa is 1000.

Finding 1: Transformer-like DeMa is not only faster but also more memory-efficient than RNN-like DeMa for short sequence length. The latter only becomes competitive when processing exceptionally long sequences.

How does sequence length affect the performance of DeMa? While the computational cost of Transformer-like DeMa increases linearly with the expansion of the sequence length, it is crucial to recognize that the increased computational cost may not ensure a corresponding enhancement in the model’s performance. Transformer-like DeMa’s Performance may plateau or even decline as the input sequence length exceeds a certain threshold. As illustrated in Figure 3, Transformer-like DeMa’s performance reaches a plateau in MuJoCo [61] when the input sequence surpasses a specific length; while significantly deteriorates with excessively long input sequences in Atari.

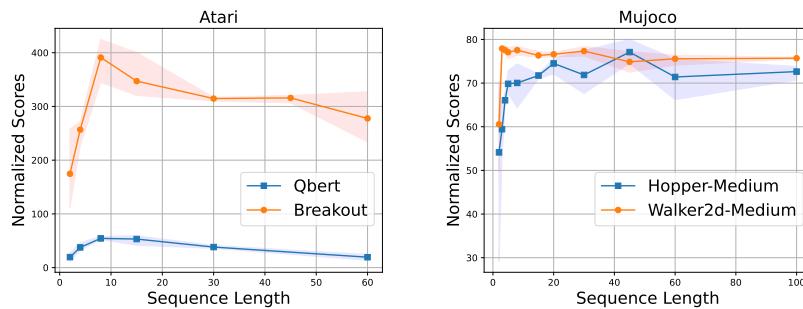


Figure 3: Comparison of Transformer-like DeMa’s Performance on Atari and MuJoCo Tasks. We report mean values averaged over 3 seeds, shaded areas represent deviations.

Finding 2: Transformer-like DeMa performs well with a short sequence length. Extending the sequence length beyond an optimal threshold does not yield further improvements and may adversely affect the model’s performance.

Why does DeMa require merely short input sequences? We calculate the hidden attention scores in DeMa via Eq. (5)-(6), which reflect the importance of historical information to DeMa. Figure 4 shows the hidden attention scores of the last K tokens at each decision-making step (from the 300th to the 600th step). It can be seen that the attention scores exhibit exponential decay as the tokens become

increasingly distant from the current decision-making moment, which aligns with the forgetting property of a Markov chain [13]. What's more, the hidden attention across different decision steps exhibits a periodic pattern towards the current token, suggesting that the model may have learned kinematic features, as agents in these environments engage in periodic movements.³

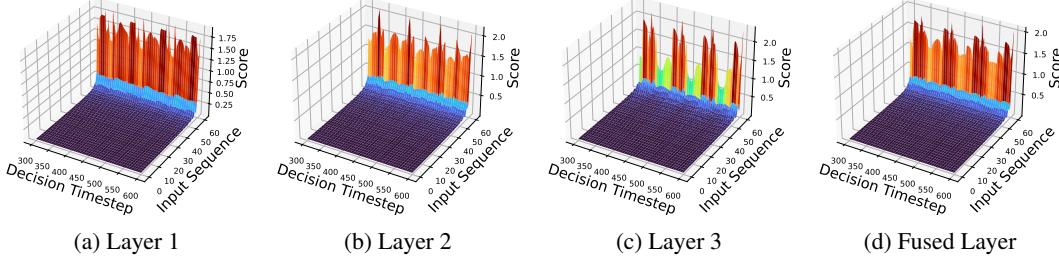


Figure 4: Hidden attention scores of DeMa from the 300th to the 600th timestep in Hopper-medium-replay. The X-axis represents timesteps from 300 to 600, the Y-axis represents the past K tokens, and the Z-axis indicates the attention scores given to the K tokens at the time of the current decision. More can be seen in Appendix J.

Finding 3: The reason that Transformer-like DeMa requires only short input sequences is its hidden attention mechanism primarily focusing on the current token. As a result, Longer sequences can lead to difficulties in training without providing benefits.

Which type of concatenation is suitable for DeMa?

Models like the Transformer and Mamba typically process inputs token by token. However, given an MDP, there are three elements s, a, r to consider. Therefore a significant design consideration is the method of concatenating these three elements into a suitable token format for the model. We experiment to investigate the suitable design for DeMa. By Table 2, concatenating the three elements in the temporal dimension yields better results. This may be due to the significant differences between the three elements of the MDP. As illustrated in [14], states and actions symbolize fundamentally dissimilar notions, concatenating them in the embedding dimension directly may make it more difficult for the model to recognize, leading to poorer results.

Finding 4: Concatenating state, action, and rtg along the embedding dimension has a significant negative impact on the results.

Table 2: Input concatenation types comparison: "BL3D" refers to the concatenation of input tokens across the embedding dimension, while "B3LD" indicates concatenation across the temporal dimension, as depicted in Figure 1. Outcomes are averaged across three random seeds.

Game	BL3D	B3LD
Breakout	72.8±10.6	314.7±10.7
Qbert	32.2±14.1	54.4±6.8
Pong	101.9±6.9	98.2±12.0
Seaquest	1.3±0.0	2.7±0.002
Asterix	3.9±0.3	7.8±0.4
Frostbite	26.3±20.9	31.1±0.01
Assault	127.9±7.1	169.4±33.1
Gopher	190.3±60.1	215.8±29.2
Average	69.6	111.8

4.2 The Essential Components of DeMa

Aside from the perspective of input data, this section delves into DeMa from the standpoint of network components. We primarily investigate the following questions: (1) Considering that some DTs do not heavily rely on attention mechanism [13, 49], is the hidden attention mechanism crucial for DeMa? (2) As the Mamba block is an integration of the hidden attention mechanism with pre up-projection residual blocks [38], what impact will it have on the performance when integrating it with other residual structures (i.e. the post up-projection residual block in the transformer)? (3) With the inherent recurrent nature of SSM [62], does DeMa need position embedding? (Appendix G)

Is the hidden attention mechanism crucial for DeMa? [27] shows that the transformer does not heavily rely on attention, and [13] finds the attention mechanism of DT is not suitable for RL. Given these insights, we aim to investigate whether a similar phenomenon exists in hidden attention.

³It is worth noting that what we want to know is the attention scores to the previous K tokens at each decision-making step, which is a bit different from the attention scores between output y_i and input x_j , which is explained in detail in Appendix J.

In line with [49], we evaluate DeMa by swapping the hidden attention weights trained in different environments, in addition to randomizing and zeroing these weights. As depicted in Figure 5, the performance exhibits a marked decrease regardless of whether the parameters are replaced with those pre-trained in other environments or randomized. Interestingly, when the parameters of hidden attention are set to zero, the model still maintains a certain level of performance. This zeroing of parameters completely removes the hidden attention, ceasing to process historical information and relying solely on residual connections to transmit information. This suggests that the residual connections are functional and the role of hidden attention is crucial for DeMa.

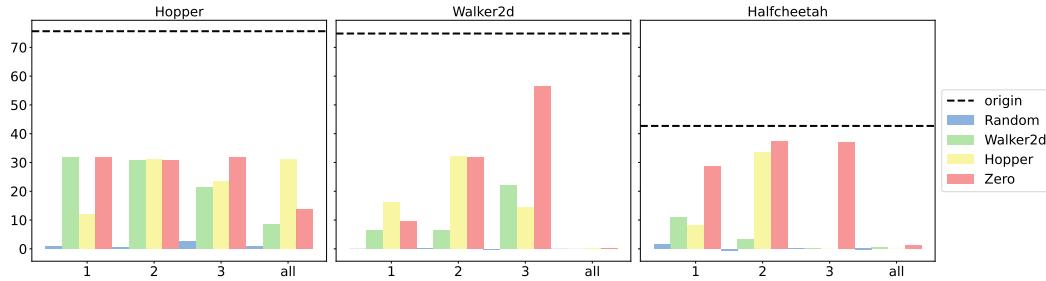


Figure 5: Normalized return after swapping the hidden attention of a single layer from another DeMa at a time. The black dashed line represents the evaluation results of the original model. "1", "2", and "3" represent the index of swap layers respectively, and "all" represents the result after swapping all parameters of the hidden attention. It can be seen that swapping the hidden attention has a significant impact on the results.

Finding 5: Replacing the hidden attention mechanism would lead to a reduction in performance, unlike the attention mechanisms used in transformers. Therefore, the hidden attention mechanism plays a crucial role in DeMa.

Table 3: Performance comparison between DT, hidden attention with post up-projection residual block in the transformer (DeMa with post.) and hidden attention with pre up-projection residual block (DeMa) in Atari.

Game	DT	DeMa with post.	DeMa
Breakout	242.4±31.8	296.2±216.3	314.7±10.7
Qbert	28.8±10.3	56.9±10.4	54.4±6.8
Pong	105.6±2.9	104.6±11.9	98.2±12.0
Seaquest	2.7±0.7	2.6±0.001	2.7±0.002
Asterix	5.2±1.2	6.5±1.8	7.8±0.4
Frostbite	25.6±2.1	31.8±4.8	31.1±0.01
Assault	52.1±36.2	146.4±16.1	169.4±33.1
Gopher	34.8±10.0	228.9±81.5	215.8±29.2
Average	62.2	109.2	111.8

Table 4: Performance comparison between DT, hidden attention with post up-projection residual block in the transformer (DeMa with post.) and hidden attention with pre up-projection residual block (DeMa) in MuJoCo.

Dataset	Env-Gym	DT	DeMa with post.	DeMa
M	HalfCheetah	42.6	42.7±0.02	43±0.01
M	Hopper	68.4	68.4±2.1	74.5±2.9
M	Walker	75.5	77.5±2.2	76.6±0.2
M-R	HalfCheetah	37.0	40.8±0.18	40.7±0.03
M-R	Hopper	85.6	86.1±26.9	90.7±6.1
M-R	Walker	71.2	74.43±2.1	70.5±0.1
M-E	HalfCheetah	88.8	83.8±18	93.2±0.01
M-E	Hopper	109.6	109.8±0.2	111±0.03
M-E	Walker	109.3	109.6±0.3	106±11.7
Average-Gym		76.4	77.0	78.5

What occurs when combining hidden attention with post up-projection residual blocks? Mamba represents the integration of the hidden attention mechanism with pre up-projection residual blocks as discussed in [38]. To determine the contributing factor to the model's enhanced performance,

we explore the combination of hidden attention with post up-projection residual blocks in transformer. According to the results in Table 3 and Table 4, although the overall average results of DeMa are slightly better than those of DeMa with post., it is observable that they each have advantages in different environments. Hence, we believe that the performance differences when integrating with the two types of residual blocks are not statistically significant. It suggests that the structure of the residual blocks exerts minimal influence on the outcome. Given that both configurations yield a measurable performance improvement over the DT, it is reasonable to conclude that the hidden attention mechanism within DeMa plays a pivotal role.

Finding 6: The results obtained using both post up-projection and pre up-projection types of residual block structures are similar while they both perform better than DT. Therefore, the hidden attention mechanism is key to its success.

5 Evaluations on Offline RL Benchmarks

Table 5: Results for 1% DQN-replay datasets. We evaluate the performance of DeMa on eight Atari games.

Game	CQL	BC	DT	DC	DC ^{hybrid}	DeMa(Ours)
Breakout	211.1	142.7	242.4±31.8	352.7±44.7	416.0 ±105.4	314.7±10.7
Qbert	104.2	20.3	28.8±10.3	67.0±14.7	62.6 ±9.4	54.4±6.8
Pong	111.9	76.9	105.6±2.9	106.5±2.0	111.1 ±1.7	98.2±12.0
Seaquest	1.7	2.2	2.7±0.7	2.6±0.3	2.7 ±0.04	2.7±0.002
Asterix	4.6	4.7	5.2±1.2	6.5±1.0	6.3 ±1.8	7.8±0.4
Frostbite	9.4	16.1	25.6±2.1	27.8±3.7	28.0 ±1.8	31.1±0.01
Assault	73.2	62.1	52.1±36.2	73.8±20.3	79.0 ±13.1	169.4±33.1
Gopher	2.8	33.8	34.8±10.0	52.5±9.3	51.6 ±10.7	215.8±29.2
Average	64.9	44.9	62.2	86.2	94.7	111.8

Table 6: Results for MuJoCo. The dataset names are abbreviated as follows: "medium" as "M", "medium-replay" as "M-R" and "medium-expert" as "M-E". The results are reported with the expert-normalized following [11].

Dataset	Environment	CQL	DS4	RvS	DT	GDT	DeMa(Ours)
M	HalfCheetah	44.0	42.5	41.6	42.6	42.9	43±0.01
M	Hopper	58.5	54.2	60.2	68.4	65.8	74.5±2.9
M	Walker	72.5	78.0	71.7	75.5	77.8	76.6±0.2
M-R	HalfCheetah	45.5	15.2	38	37.0	39.9	40.7±0.03
M-R	Hopper	95.0	49.6	73.5	85.6	81.6	90.7±6.1
M-R	Walker	77.2	69.0	60.6	71.2	74.8	70.5±0.1
M-E	HalfCheetah	91.6	92.7	92.2	88.8	92.4	93.2±0.01
M-E	Hopper	105.4	110.8	101.7	109.6	110.9	111±0.03
M-E	Walker	108.8	105.7	106.0	109.3	109.3	106±11.7
Average		77.6	68.6	71.7	76.4	76.8	78.5

Table 7: The resource usage for training DT, DC and DeMa on Atari and MuJoCo.

	Complexity	DT	DC	DeMa(Ours)
Atari	Training time per step(ms)	55	43	50
	GPU memory usage(GiB)	4.2	3.0	4.2
	MACs	12.1G/46.5G	11.1G/40.6G	8.8G/36.3G
	All params #	2.35M	1.94M	1.7M
Gym	Training time per step(ms)	56/58	53.6/53.9	57.6/58.8
	GPU memory usage(GiB)	0.65/0.8	0.55/0.6	1.0/1.0
	MACs	2.5G/9.5G	1.6G/6.1G	0.7G/2.1G
	All params #	726.2K/2.6M	536K/1.9M	175.5K/500.0K

In this section, we delve into a comparative analysis of DeMa’s performance against various DTs. Our investigation primarily centers on the influence of disparate network architectures on the experimental outcomes. Consistent with antecedent studies, we assessed both discrete (Atari [58]) and continuous control tasks (MuJoCo [63]), presenting the normalized scores accordingly. Given that the sequence

length considerably affects the results, we selected the optimal outcomes from sequence lengths $K = 8$ to $K = 20$ for DeMa. The detailed hyper-parameters on DeMa are available in Appendix D. Our main results are shown in Table 5 and Table 6. DeMa achieves a significantly higher average score compared to DT in Atari games, while the number of parameters and the number of MACs in DeMa are each five times fewer than those in DT, as shown in Table 7. Moreover, DeMa has better scalability for input length which can be seen in Figure 2, it maintains a slow linear growth with the input sequence length increases while the computational cost of the Transformer grows quadratically. These results demonstrate that our transformer-like DeMa is well-suited for integration with trajectory optimization methods.

6 Conclusion

To investigate Mamba’s compatibility with trajectory optimization, this work conducts comprehensive experiments from the aspect of data structures and network architectures. Our findings reveal that (1) DeMa benefits from short sequence lengths due to its exponentially decaying focus on sequences. Consequently, we incorporate a Transformer-like DeMa. (2) The hidden attention mechanism plays a crucial role in DeMa. It can combine with other residual structures and does not require position embedding. Based on the insights gained from the investigation, our DeMa surpasses previous methods, achieving higher performance over the DT while using 30% fewer parameters in eight Atari games. In the MuJoCo, our DeMa outperforms DT with only a quarter of the parameters. In conclusion, our DeMa is compatible with trajectory optimization in offline RL.

Limitations. We investigate the application of Mamba in trajectory optimization and present findings that provide valuable insights for the community. However, there remain several limitations: (1) Trajectory optimization tasks typically involve shorter input sequences, raising questions about how well the RNN-like DeMa performs in terms of memory capacity in RL compared to models such as RNNs and LSTMs. Furthermore, the potential of both types of DeMa warrants further exploration, particularly in some POMDP environments and long-horizon non-Markovian tasks that require long-term decision-making and memory. (2) We examine the importance of the hidden attention mechanism in Section 4.2, future work could leverage interpretability tools to examine further the causal relationship between memory and current decisions in DeMa, ultimately contributing to the development of interpretable decision models. (3) While we have assessed the properties of DeMa and identified improvements in both performance efficiency and model compactness compared to DT, it remains unclear whether DeMa is suitable for multi-task RL and online RL environments.

Acknowledgments and Disclosure of Funding

This work is supported by STI 2030–Major Projects (No. 2021ZD0201405), National Natural Science Foundation of China (No. 72301289), and the Zhejiang Province Science Foundation under Grants LD24F020002. We thank zigzagcai for his PR: support variable-length sequences for mamba block. We thank Liang Zhang and Yang Ma for their valuable suggestions and collaboration. We sincerely appreciate the time and effort invested by the anonymous reviewers in evaluating our work and are grateful for their valuable and insightful feedback.

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] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.
- [3] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.
- [4] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

[5] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[6] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[7] Cong Lu, Philip J Ball, Jack Parker-Holder, Michael A Osborne, and Stephen J Roberts. Revisiting design choices in offline model-based reinforcement learning. *arXiv preprint arXiv:2110.04135*, 2021.

[8] Haoyang He. A survey on offline model-based reinforcement learning. *arXiv preprint arXiv:2305.03360*, 2023.

[9] Phillip Swazinna, Steffen Udluft, Daniel Hein, and Thomas Runkler. Comparing model-free and model-based algorithms for offline reinforcement learning. *IFAC-PapersOnLine*, 55(15):19–26, 2022.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

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

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

[13] Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Decision conformer: Local filtering in metaformer is sufficient for decision making. *arXiv preprint arXiv:2310.03022*, 2023.

[14] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Graph decision transformer. *arXiv preprint arXiv:2303.03747*, 2023.

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

[16] Shengchao Hu, Li Shen, Ya Zhang, Yixin Chen, and Dacheng Tao. On transforming reinforcement learning with transformers: The development trajectory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

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

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

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

[20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

[22] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5):1–32, 2021.

[23] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021.

[24] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[25] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[26] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pages 10183–10192. PMLR, 2021.

[27] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829, 2022.

[28] James D Hamilton. State-space models. *Handbook of econometrics*, 4:3039–3080, 1994.

[29] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[30] Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of mamba models. *arXiv preprint arXiv:2403.01590*, 2024.

[31] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. *arXiv preprint arXiv:2403.06977*, 2024.

[32] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024.

[33] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

[34] Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*, 2024.

[35] Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. *arXiv preprint arXiv:2403.04253*, 2024.

[36] Fei Deng, Junyeong Park, and Sungjin Ahn. Facing off world model backbones: Rnns, transformers, and s4. *Advances in Neural Information Processing Systems*, 36, 2024.

[37] Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[38] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

[39] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[40] Tenglong Liu, Yang Li, Yixing Lan, Hao Gao, Wei Pan, and Xin Xu. Adaptive advantage-guided policy regularization for offline reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024.

[41] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Prompt-tuning decision transformer with preference ranking. *arXiv preprint arXiv:2305.09648*, 2023.

[42] Shengchao Hu, Ziqing Fan, Chaoqin Huang, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Q-value regularized transformer for offline reinforcement learning. In *International Conference on Machine Learning*, 2024.

[43] Shengchao Hu, Ziqing Fan, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Harmodt: Harmony multi-task decision transformer for offline reinforcement learning. In *International Conference on Machine Learning*, 2024.

[44] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] Jifeng Hu, Yanchao Sun, Sili Huang, SiYuan Guo, Hechang Chen, Li Shen, Lichao Sun, Yi Chang, and Dacheng Tao. Instructed diffuser with temporal condition guidance for offline reinforcement learning. *arXiv preprint arXiv:2306.04875*, 2023.

[46] Sili Huang, Jifeng Hu, Hechang Chen, Lichao Sun, and Bo Yang. In-context decision transformer: Reinforcement learning via hierarchical chain-of-thought. *arXiv preprint arXiv:2405.20692*, 2024.

[47] Wenzhe Li, Hao Luo, Zichuan Lin, Chongjie Zhang, Zongqing Lu, and Deheng Ye. A survey on transformers in reinforcement learning. *arXiv preprint arXiv:2301.03044*, 2023.

[48] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

[49] Daniel Lawson and Ahmed H Qureshi. Merging decision transformers: Weight averaging for forming multi-task policies. *arXiv preprint arXiv:2303.07551*, 2023.

[50] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[51] Sili Huang, Jifeng Hu, Zhejian Yang, Liwei Yang, Tao Luo, Hechang Chen, Lichao Sun, and Bo Yang. Decision mamba: Reinforcement learning via hybrid selective sequence modeling. *arXiv preprint arXiv:2406.00079*, 2024.

[52] Shmuel Bar David, Itamar Zimerman, Eliya Nachmani, and Lior Wolf. Decision s4: Efficient sequence-based rl via state spaces layers. In *The Eleventh International Conference on Learning Representations*, 2022.

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

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

[55] Sidd Karamcheti Sasha Rush. The annotated s4. <https://srush.github.io/annotated-s4/>, 2023.

[56] Yijun Yang, Zhaohu Xing, and Lei Zhu. Vivim: a video vision mamba for medical video object segmentation. *arXiv preprint arXiv:2401.14168*, 2024.

[57] Zeyu Zhang, Akide Liu, Ian Reid, Richard Hartley, Bohan Zhuang, and Hao Tang. Motion mamba: Efficient and long sequence motion generation with hierarchical and bidirectional selective ssm. *arXiv preprint arXiv:2403.07487*, 2024.

[58] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[59] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

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

[61] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[62] Weihao Yu and Xinchao Wang. Mambaout: Do we really need mamba for vision? *arXiv preprint arXiv:2405.07992*, 2024.

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

[64] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[65] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

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

[67] Xiao Zhou, Yujie Zhong, Zhen Cheng, Fan Liang, and Lin Ma. Adaptive sparse pairwise loss for object re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19691–19701, 2023.

Supplementary Material for

Is Mama Compatible with Trajectory Optimization in Offline Reinforcement Learning?

Contents

A Environment and Dataset	14
B Baselines	14
C The Procedure of Training and Inference	14
D Implementation details of DeMa	15
E Detailed results	17
F Tasks Requires Long Horizon Planning Skills	17
G Further Ablation Study	19
H Integrating DeMa with Other Methods	19
I MuJoCo and Atari Tasks Scores	19
J Types of Hidde Attention Scores	19

A Environment and Dataset

MuJoCo. The MuJoCo domain [59] evaluates the performance of RL algorithms in continuous control tasks. In keeping with previous studies, we select three games from the standard locomotion environments [59] in Gym [61], namely HalfCheetah, Hopper, and Walker, and three different dataset settings, namely medium, medium-replay, and medium-expert [63].

Atari. Atari [58] is an ideal platform for evaluating an agent’s ability in long-term credit assignments. We conduct experiments in eight different games: Breakout, Qbert, Pong, Seaquest, Asterix, Frostbite, Assault, and Gopher. We use 1% DQN Replay Dataset [64] as our training dataset, which encompasses a total of 500,000 timesteps worth of samples generated throughout the training process of a DQN agent [65]. It’s worth noting that the version of “atari-py” and “gym” we use is 0.2.5 and 0.19.0 respectively, which is noted by the official code in https://github.com/google-research/batch_rl.

B Baselines

Baselines for MuJoCo. To evaluate DeMa’s performance in the MuJoCo, we compare DeMa with one value-based method: CQL [66] and four trajectory optimization methods with different network architectures: DS4 [52], RvS [4], DT [12], GDT [14] and obtain baseline performance scores for CQL and DS4 from [13], for RvS from [4] and for GDT from [14].

Baselines for Atari. In the Atari domain, we compare DeMa with CQL [66], DT [12], DC and DC^{hybrid} [13]. The results of baselines are directly borrowed from [13].

C The Procedure of Training and Inference

Training resources We use one NVIDIA GeForce RTX 4090 to train each model in MuJoCo and one NVIDIA GeForce RTX 3090 to train each model in Atari. Training each model typically takes

3-8 hours and 5-14 hours in MuJoCo and Atari respectively. However, since each environment needs to be trained three times with different seeds, the total training time is usually multiplied by three.

The procedure of Transformer-like DeMa The Training and evaluation for Transformer-like DeMa are similar to variant DTs. Given a dataset of offline trajectories, we randomly select a starting point and truncate it into a sequence of length K . After forming a batch of data, it is input into the model for training. We minimize the reconstruction loss between the predicted action and the actual action, i.e. the cross-entropy loss for discrete actions and Mean Square Error (MSE) for continuous actions. The input data is also a sequence of length K in the evaluation phase.

The procedure of RNN-like DeMa For RNN-like DeMa, the input during training is a batch of complete trajectories. As different trajectories have different lengths, we pad the trajectories to the same length before inputting them into the model and mask the loss of the padding. However, training with full trajectories rather than truncated sequences may be more inefficient, especially in scenarios where sequence lengths vary widely. In the DQN Replay Dataset in Atari, the lengths of different trajectories varied dramatically. Some trajectories might only be 500 timesteps long, while others could contain a sample with a length of 10,000 timesteps. This causes a lot of computing resources to be wasted on meaningless padding, resulting in inefficiency and ineffectiveness. Some techniques can avoid this issue. One can refer to this PR.

D Implementation details of DeMa

We implement DeMa based on the official code of DT and the Mamba. We have also adopted the code from HiddenMambaAttn to calculate the attention scores of DeMa on the current input sequence at each decision step. Given that the official Mamba code utilizes Triton, we also employ Mamba-minimal which is fully based on pytorch to compute the MACs of DeMa.

Tables 8-10 provide a comprehensive list of hyper-parameters for our proposed transformer-like DeMa and RNN-like DeMa applied to MuJoCo and Atari environments. To ensure a fair comparison, we adopt similar hyper-parameter settings to DT [12] and DC [13].

D.1 Hyper-parameters in MuJoCo

Table 8: Hyper-parameters of DeMa for MuJoCo.

Hyper-parameter	Value
Layers	3
Embedding dimension	128
Nonlinearity function	\
Batch size	64
Context length K	20
Dropout	0.0
Learning rate	10^{-4}
Grad norm clip	0.25
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 10^5 training steps
d_{model}	64
d_{state}	64
expand	2

For our training on MuJoCo, the majority of the hyper-parameters in Table 8 are adapted from [13]. For the learning rate, we use a learning rate of 10^{-4} for training in hopper-medium, hopper-medium-replay, and walker2d-medium and use 10^{-3} for other environments. For the embedding dimension, we use an embedding dimension of 256 in hopper-medium and hopper-medium-replay, while use 128 in the other environments. What's more, as DeMa does not use multilayer perceptron (MLP), so there is no nonlinearity function for DeMa. As for DeMa with post. in Table 3, we use ReLU as per convention. For DeMa's hyper-parameters, we use a d_{model} of 128 in all expert datasets, while use

64 in the other environments. As for d_{state} and expand , we set 64 and 2 respectively for all env. We keep the experimental parameters consistent for all types of DeMa in MuJoCo.

D.2 Hyper-parameters in Atari

Transformer-like DeMa. For the Atari game we mostly follow those in Table 9 from [12]. The only adjustment made is to the context length K and return-to-go conditioning. As revealed in Figure 3, the sequence length is not always better when it's longer. Thus for Qbert and Frostbite we use $K = 8$. For other games, we keep $K = 30$. As for the return-to-go conditioning, we find the return obtained by DeMa in some games has already exceeded the initial "return to go" set for DT. Therefore, we increase the "return to go" so that DeMa can fully demonstrate its performance.

Table 9: Hyper-parameters of Transformer-like DeMa for Atari.

Hyper-parameter	Value
Layers	6
Embedding dimension	256
Nonlinearity function	ReLU(state encoder)
Batch size	128
Context length K	30
Return-to-go conditioning	90 Breakout, 12000 Qbert 20 Pong, 1750 Seaquest 700 Asterix, 1450 Frostbite 1200 Assault, 6500 Gopher
Dropout	0.1
Learning rate	6×10^{-4}
Grad norm clip	1
Weight decay	0.1
Learning rate decay	Linear warmup and cosine decay (see code for details)
Max epochs	10
Adam betas	(0.9, 0.95)
Warmup tokens	512×20
Final tokens	$6 \times 500000 \times K$
d_{model}	128
d_{conv}	4
d_{state}	64
expand	2

Table 10: Hyper-parameters of RNN-like DeMa for Atari. The other hyper-parameters are kept consistent with those in Table 9.

Hyper-parameter	Value
Context length	all trajectory
Batch size	8
Learning rate	10^{-4}
inner_it	200

RNN-like DeMa. Since the RNN-like DeMa utilizes trajectories for training and the trajectories in Atari are exceptionally lengthy, the available sample size becomes significantly limited when only 1% of the DQN-replay dataset is utilized. If the prior parameter settings were to be used, the training would done after only a few hundred upgrades, thereby resulting in an unsatisfactory performance. Therefore, we consider multiple updates for a single sample, while simultaneously lowering the learning rate as shown in Table 10. What's more, due to the limitation of GPU memory, we can only set a batch size of 8 for Atari. Specifically, For the Frostbite, we set a batch size of 1, an epoch of 50. The other hyper-parameters are kept consistent with those in Table 9.

E Detailed results

Table 11 and Table 12 show detailed results between RNN-like DeMa⁴ and Transformer-like DeMa. It can be observed that the performance of the RNN-like DeMa is not as good as that of the Transformer-like DeMa, and Figure 2 also shows that the RNN-like DeMa requires more computational overhead. Hence, using the RNN model in trajectory optimization seems to be unnecessary, as section 4.1 finds that past historical information does not provide much assistance to current decision-making. However, in tasks that require memory capability or are model-based, the RNN-like DeMa could be a better choice. This could be a direction for deeper future research based on [35–37].

Table 11: The Comparison of DT, RNN-like DeMa, and Transformer-like DeMa in Atari Games.

Env	DT	RNN-like DeMa	Transformer-like DeMa
Breakout	242.4±31.8	166.0	314.7±10.7
Qbert	28.8±10.3	13.6	54.4±6.8
Pong	105.6±2.9	109.6	98.2±12.0
Seaquest	2.7±0.7	1.7	2.7±0.002
Asterix	5.2±1.2	4.7	7.8±0.4
Frostbite	25.6±2.1	8.6	31.1±0.01
Assault	52.1±36.2	117.8	169.4±33.1
Gopher	34.8±10.0	116.7	215.8±29.2
Average	62.2	67.3	111.8

Table 12: The comparison between DT, RNN-like DeMa, and Transformer-like DeMa in MuJoCo.

Dataset	Environment	DT	RNN-like DeMa	Transformer-like DeMa
M	HalfCheetah	42.6	42.6±0	43±0.01
M	Hopper	68.4	61.7±4.9	74.5±2.9
M	Walker	75.5	76.7±0.2	76.6±0.2
M-R	HalfCheetah	37.0	36.9±0.3	40.7±0.03
M-R	Hopper	85.6	80.5±25.8	90.7±6.1
M-R	Walker	71.2	68.1±8.1	70.5±0.1
Average		63.4	61.1	66.0

F Tasks Requires Long Horizon Planning Skills

Three experiments involving delayed rewards(MuJoCo with delayed rewards) and maze navigation(maze2d, antmaze) are conducted to investigate how would DeMa perform on tasks that require long horizon planning skills.

F.1 MuJoCo with Delayed Rewards

To investigate DeMa’s performance on tasks with delayed rewards, we conduct an experiment on a delayed return version of the D4RL benchmarks [12], in which the agent does not receive any rewards along the trajectory but instead receives the cumulative reward of the trajectory in the final timestep. In this environment, we train DeMa using the same hyper-parameters settings, and the results are shown in Table 13. Results show that CQL is the most affected, while DT also experiences a certain degree of influence. In contrast, DeMa is relatively less impacted. The results indicate that DeMa demonstrates effective performance in tasks with delayed rewards.

F.2 Maze Navigation

There are two environments in maze navigation. **Maze2d**: This environment aims at reaching goals with sparse rewards, which is suitable for assessing the model’s capability to efficiently integrate data and execute long-range planning. The objective of this domain is to guide an agent through a maze to reach a designated goal. **Antmaze**: This environment is similar to maze2d, while the agent is an ant

⁴Due to the high experimental costs, we only run the RNN-like DeMa in Atari once.

Table 13: Results for D4RL datasets with delayed (sparse) reward. The "Origin Average" in the table represents the normalized scores of evaluations across six datasets under the original dense reward setting.

Dataset	Env-Gym	CQL	DS4	DT	GDT	DeMa(Ours)
M	HalfCheetah	1.0 ± 1.0	42.7 ± 0	42.2 ± 0.2	43 ± 0	42.9 ± 0.01
M	Hopper	23.3 ± 1.0	58.2 ± 0.7	57.3 ± 2.4	58.2 ± 2.4	69.1 ± 6.5
M	Walker	0.0 ± 0.4	75.7 ± 0.5	69.9 ± 2.0	78.9 ± 0.1	77.6 ± 1.5
M-R	HalfCheetah	7.8 ± 6.9	15.5 ± 0	33.0 ± 4.8	41 ± 0.1	41.1 ± 0.15
M-R	Hopper	7.7 ± 5.9	77.5 ± 0.4	50.8 ± 14.3	79.8 ± 15.9	83.8 ± 6.9
M-R	Walker	3.2 ± 1.7	69.1 ± 3.2	51.6 ± 24.6	70.4 ± 8.7	71.7 ± 5
Average-Gym		7.2	56.45	50.8	61.9	64.4

with 8 degrees of freedom. For our training on Maze, the majority of the hyper-parameters in Table 14 and Table 15. For maze2d-medium, we use $K = 8$ and embedding_dim=256. For maze2d-umaze, we use hyper-parameters in Table 8.

Table 14: Hyper-parameters of DeMa for antmaze.

Hyper-parameter	Value
Layers	3
Embedding dimension	128
Nonlinearity function	\
Batch size	32
Context length K	5
Dropout	0.1
Learning rate	$2e^{-5}$
Grad norm clip	0.25
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 10^5 training steps
d_model	128
d_state	64
num_eval_episodes	50
max_iters	50
num_steps_per_iter	2000

Table 15: Hyper-parameters of DeMa for maze2d.

Hyper-parameter	Value
Layers	3
Embedding dimension	128
Nonlinearity function	\
Batch size	32
Context length K	20
Dropout	0.1
Learning rate	$2e^{-5}$
Grad norm clip	0.25
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 10^5 training steps
d_model	64
d_state	64
num_eval_episodes	50
max_iters	50
num_steps_per_iter	2000

We compare DeMa with DT [12], GDT [14] and DC [13]. The results of DT and GDT are directly borrowed from [42]. Results in Table 16 show that DeMa performs better compared to DT in the maze

navigation task. The visualization analysis of the hidden attention mechanism in these environments can be found in Figure 10 and Figure 11.

Table 16: Results for maze2d and antmaze.

Dataset	Env-Gym	DT	GDT	DC	DeMa(Ours)
umaze medium large	maze2d	31.0	50.4	36.3 ± 3	54.3 ± 9.4
		8.2	7.8	2.1 ± 1.02	10.3 ± 3.1
		2.3	0.7	0.9 ± 0	2.8 ± 2.2
umaze umaze-diverse	antmaze	59.2	76	85.00	82 ± 0
		53	69	78.5	80.7 ± 6.2

G Further Ablation Study

Table 17: The affection of position embedding.

Dataset	Env	DeMa with pos. embed.	DeMa without pos. embed.
M	HalfCheetah	42.8 ± 0	43 ± 0.01
	Hopper	71.2 ± 14.6	74.5 ± 2.9
	Walker	77.2 ± 0.1	76.6 ± 0.2
M-R	HalfCheetah	40.2 ± 0.1	40.7 ± 0.03
	Hopper	77.2 ± 35	90.7 ± 6.1
	Walker	69.1 ± 10.2	70.5 ± 0.1
Average		63.0	66.0
All params #		431.5K	175.5K

DeMa does not need the position embedding. Position embedding is generally used in transformers to help the model understand the sequential nature of the data. It’s a way of encoding the position of tokens in the sequence, and it can be crucial in tasks where the order of the data matters. Although we can use DeMa similar to using a transformer, which has input and output dimensions of (B, L, D) during training and inference, it differs in that it does not require position embedding to help the model have the ability to remember sequential information. As shown in Table 17, the addition of position embedding not only failed to enhance the performance of the model but also led to a significant decrease in performance on certain tasks. Additionally, the introduction of position embedding significantly increased the model’s parameter count, thereby adding to its computational burden. This finding highlights the advantage of the DeMa in terms of lightweight design, indicating its suitability for tasks with limited resources.

H Integrating DeMa with Other Methods

We conduct additional experiments to show that DeMa can be combined with other trajectory optimization methods to achieve even better performance. By integrating DeMa with QT [42], we develop Q-DeMa. As shown in Table 18, Q-DeMa achieves performance comparable to state-of-the-art models while utilizing less than one-seventh of the parameter size of QT. This finding underscores the significant potential of applying Mamba to RL.

I MuJoCo and Atari Tasks Scores

Table 19 shows the normalized scores used in MuJoCo and Atari tasks, followed by [63] and [60].

J Types of Hidde Attention Scores

In the previous articles [13, 67], the visualization of attention in DT was in the form of a lower triangular matrix. However, this lower-triangular matrix reflects the attention scores of each generated

Table 18: Q-DeMa's Results for D4RL datasets.

Dataset	Env-Gym	DT	DeMa(Ours)	QT	Q-DeMa(Ours)
M	HalfCheetah	42.6	43±0.01	51.4 ± 0.4	51.2±0.04
M	Hopper	68.4	74.5±2.9	96.9 ± 3.1	88.1±9.61
M	Walker	75.5	76.6±0.2	88.8 ± 0.5	89.1±0.2
M-R	HalfCheetah	37.0	40.7±0.03	48.9 ± 0.3	48.6±0.3
M-R	Hopper	85.6	90.7±6.1	102.0 ± 0.2	101.5±0.1
M-R	Walker	71.2	70.5±0.1	98.5 ± 1.1	99.8±1
Average-Gym		63.4	66.0	81.0	79.7
All params #		726.2K/2.6M	175.5K/500.0K	3.7M	500K

Table 19: MuJoCo and Atari baseline scores used for normalization

	Env/Game	Random	Expert/Gamer
Gym	Hopper	-20.3	3234.3
	Halfcheetah	-280.2	12135
	Walker2d	1.6	4592.3
Atari	Breakout	1.7	30.5
	Qbert	163.9	13455
	Pong	-20.7	14.6
	Seaquest	68.4	42054.7
	Asterix	210	8503
	Frostbite	65	4335
	Assault	222	742
	Gopher	258	2412

token to the input sequence during the training phase, and it cannot accurately illustrate the context information that the model focuses on at each decision-making step. As can be seen, the element in Figure 6 at the i -th row and j -th column represents presents the output y_i 's attention score to input x_j . In the training phase, all corresponding predicting actions are used to calculate the reconstructed loss with target actions. However, during the evaluation phase, i.e. when interacting with the environment, we input $x : (1, L, D)$, and the model also outputs $y : (1, L, D)$. At this time, we only use the last one of the model's output, which is $y[:, -1, :]$, corresponding to the last row in the matrix. Therefore, it is not quite appropriate to judge the context information the model focuses on at each decision-making step based on the lower-triangular matrix in Figure 6, as we want to understand the model's decision-making behavior at each step, thus leads to the creation of Figure 4, Figure 7 and Figure 8. It also demonstrates strong forgetting characteristics. This aligns with the properties of Markov chains as described in [13], where the sequence of states precisely forms a Markov chain. We also conduct additional explorations in environments involving delayed rewards(MuJoCo with delayed rewards) and maze navigation(maze2d, antmaze). The performance of the hidden attention mechanism is illustrated in Figures 9-11. Although DeMa's attention to past information increases, the hidden attention mechanism still prioritizes the current information when the Markov property of the environment is relaxed. Furthermore, among historical information, the hidden attention mechanism demonstrates a significantly higher focus on states compared to rewards or actions.

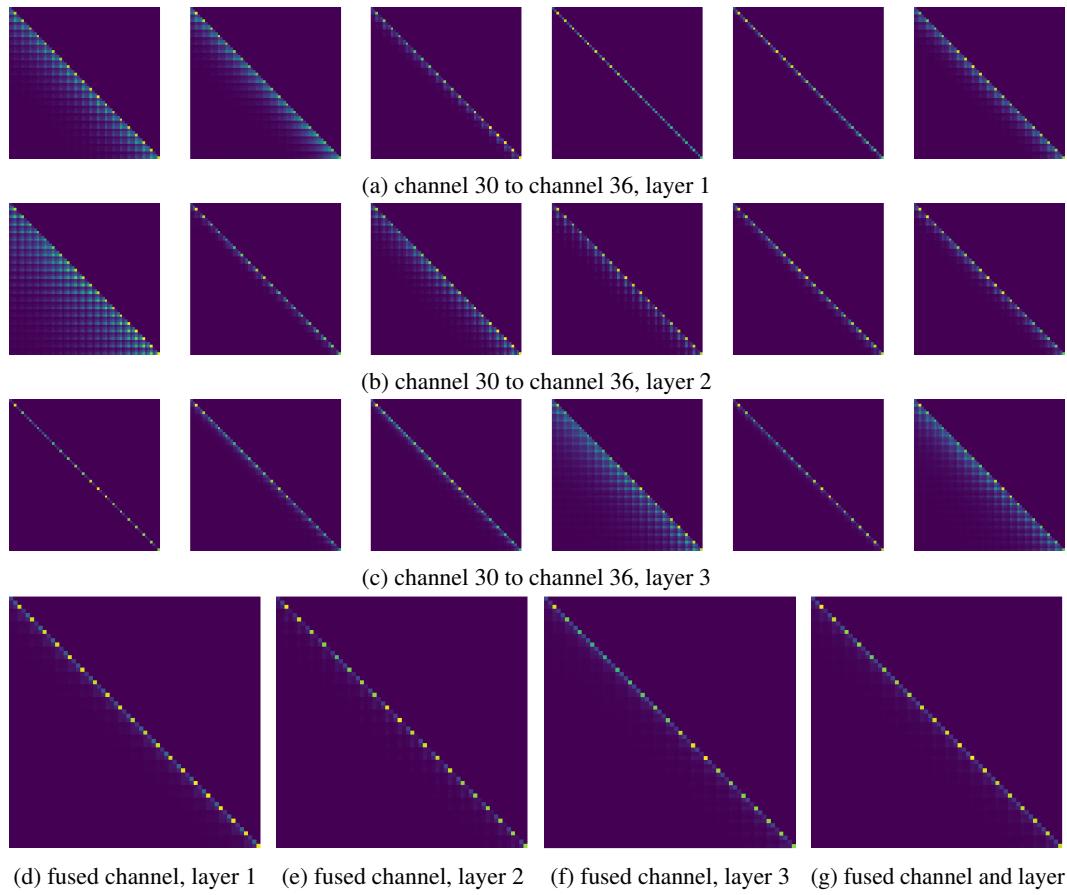


Figure 6: Hidden Attention Score Matrix of each channel and layer of DeMa, trained on the Hopper-medium dataset. The element A_{ij} present the attention score between output y_i and input x_j

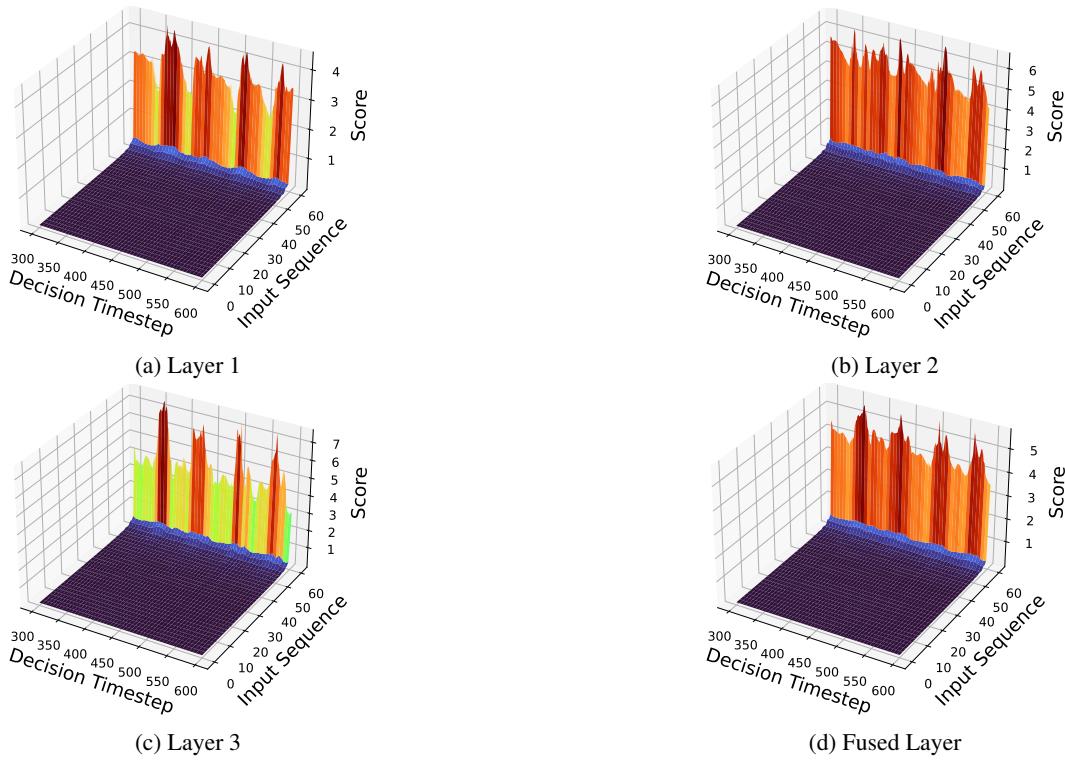


Figure 7: Hidden attention scores of DeMa from the 300th to the 600th timestep, trained on the Walker2d-medium dataset.

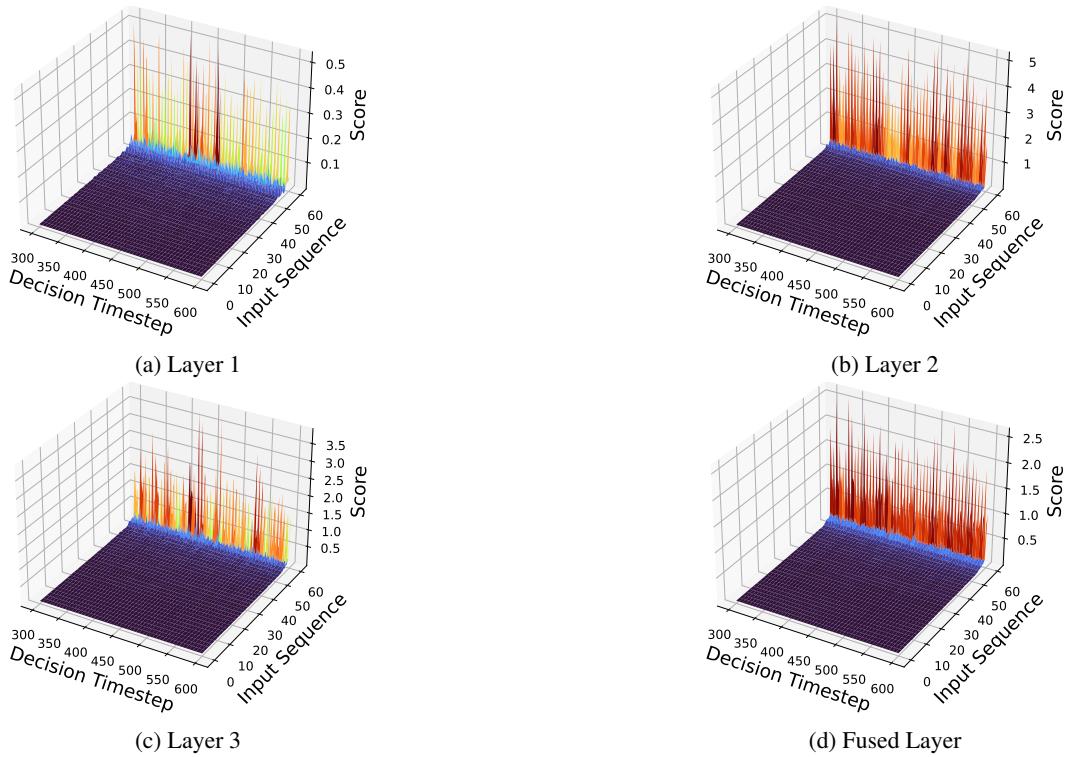
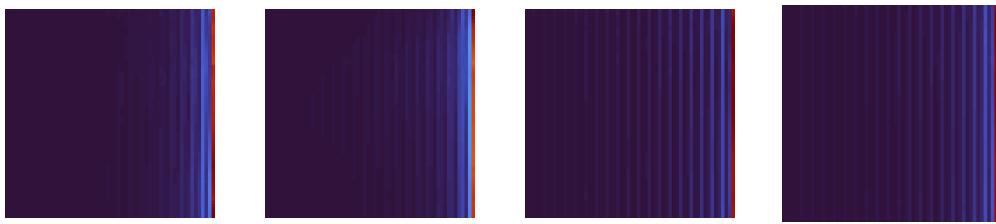


Figure 8: Hidden attention scores of DeMa from the 300th to the 600th timestep, trained on the Halfcheetah-medium-expert dataset.



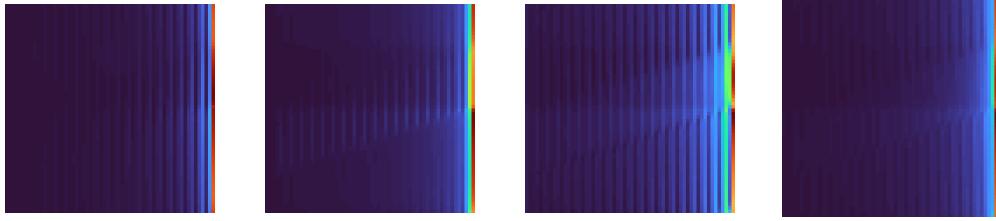
(a) Layer 1

(b) Layer 2

(c) Layer 3

(d) Fused Layer

Figure 9: Hidden attention scores of DeMa from the 540th to the 600th timestep, trained on the Walker2d-medium dataset with delayed rewards. Hidden attention mechanism highlighting more focus on historical observations. **Top:** 2D Representation, **Bottom:** 3D Representation.



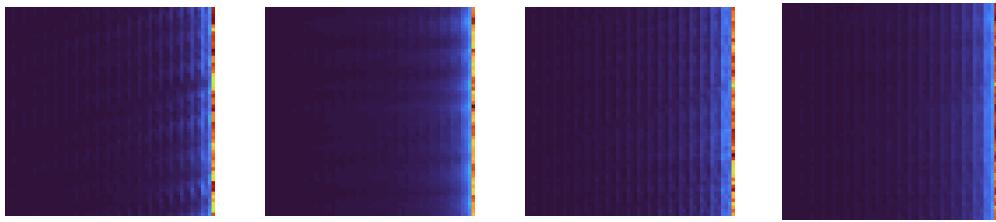
(a) Layer 1

(b) Layer 2

(c) Layer 3

(d) Fused Layer

Figure 10: Hidden attention scores of DeMa from the 80th to the 140th timestep in maze2d-umaze. **Top:** 2D Representation, **Bottom:** 3D Representation.



(a) Layer 1

(b) Layer 2

(c) Layer 3

(d) Fused Layer

Figure 11: Hidden attention scores of DeMa from the 80th to the 140th timestep in antmaze-umaze-diverse. **Top:** 2D Representation, **Bottom:** 3D Representation.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: In the last paragraph in Section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: There is no theoretical result, this study is a empirical research.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Mostly following previous studies, some parameters have been modified according to our investigation as can be seen in Section 4.1 and Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: The results are averaged across three random seeds, with deviation for Fig 3 and variance in others.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer “Yes” if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: In Appendix C

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification:

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[NA\]](#)

Justification:

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and Research with Human Subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.