Decision Mamba: Reinforcement Learning via Hybrid Selective Sequence Modeling

Sili Huang[1] **Jifeng Hu**[2] **Zhejian Yang**[3] **Liwei Yang**[4] **Tao Luo**[5]
Hechang Chen[6*] **Lichao Sun**[7] **Bo Yang**[8*]
[1,8]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education
[1,2,3,6]School of Artificial Intelligence, Jilin University, China
[4,5] Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore
[7]Lehigh University, Bethlehem, Pennsylvania, USA

Abstract

Recent works have shown the remarkable superiority of transformer models in reinforcement learning (RL), where the decision-making problem is formulated as sequential generation. Transformer-based agents could emerge with self-improvement in online environments by providing task contexts, such as multiple trajectories, called in-context RL. However, due to the quadratic computation complexity of attention in transformers, current in-context RL methods suffer from huge computational costs as the task horizon increases. In contrast, the Mamba model is renowned for its efficient ability to process long-term dependencies, which provides an opportunity for in-context RL to solve tasks that require long-term memory. To this end, we first implement Decision Mamba (DM) by replacing the backbone of Decision Transformer (DT). Then, we propose a Decision Mamba-Hybrid (DM-H) with the merits of transformers and Mamba in high-quality prediction and long-term memory. Specifically, DM-H first generates high-value sub-goals from long-term memory through the Mamba model. Then, we use sub-goals to prompt the transformer, establishing high-quality predictions. Experimental results demonstrate that DM-H achieves state-of-the-art in long and short-term tasks, such as D4RL, Grid World, and Tmaze benchmarks. Regarding efficiency, the online testing of DM-H in the long-term task is 28$\times$ times faster than the transformer-based baselines.

1 Introduction

Large transformer models [43] have achieved notable successes across a variety of domains, including text [4], image [9], and audio [1]. In the field of reinforcement learning (RL), large transformer models can treat RL tasks as a type of sequential prediction problem and have shown impressive results with offline training [31, 39]. However, these methods lack self-improvement when used in online environments, where online environments could differ from offline training. To overcome this, in-context RL has been proposed, which enables continued policy improvement [29].

Recent works demonstrated that in-context RL methods can automatically improve online performance by providing prompt conditions called across-episodic contexts [30]. The construction of the across-episodic context is flexible and easy to implement, such as multiple historical trajectories arranged in ascending order of returns [20]. Although no gradient updates are required for self-improvement, current in-context RL still suffers from high computational costs on long-term tasks [29]. This arises from (1) the quadratic complexity of the self-attention mechanism and (2) the multiplicative growth of the long-term sequence caused by across-episodic contexts.

*Corresponding Author. Bo Yang and Hechang Chen.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-2314

Facing the challenge of handling long-term sequences, a novel foundation model called Mamba has attracted widespread attention for its ability to capture long-term dependencies with linear computational costs [17, 6]. Mamba is a state space model-based framework with good potential in natural language processing and vision tasks [47]. Motivated by the success of Mamba in language and vision modeling, it is appealing that we can also transfer this success to RL tasks. Therefore, a natural question arises:

"Can Mamba boost in-context RL with both effectiveness and efficiency on long-term tasks?"

Taking the famous in-context RL method Decision Transformer (DT) [29] as an example, we replace its transformer backbone with a Mamba backbone. Regarding efficiency, there is no doubt that Mamba is superior to transformers as the task length increases because Mamba uses a data-dependent selection mechanism that computes independently on each input of sequences [17]. Compared with the attention mechanism that computes input pairs, the data-dependent selection mechanism makes it easier for Mamba to handle long sequences but also introduces additional independence assumptions. In RL tasks, since there is a sequential relationship between states rather than independence, the attention mechanism may be intuitively more suitable for capturing the information between states, thereby outperforming Mamba in effectiveness.

To this end, instead of implement a Decision Mamba (DM) that simply replaces the backbone of DT, we propose the Decision Mamba-Hybrid (DM-H) with the merits of transformers and Mamba in high-quality prediction and long-term memory. Specifically, the Mamba model first generates sub-goals, represented by a vector, based on long-term contexts. Then, we combine the sub-goal and short-term contexts as a prompt condition to the transformer. The setting of sub-goals enables DM-H to leverage Mamba's ability to efficiently recall long-term contexts while using the transformer to predict high-quality actions. However, it remains to be seen whether the transformer will benefit from sub-goals or ignore them and predict actions focused on short-term contexts. Therefore, we select high-value states from offline trajectories to form extra sub-goals and serve them as input conditions for the transformer. Then, the predicted actions will associate with these selected sub-goals and, in turn, encourage Mamba to generate high-value sub-goals. Our contributions are as follows:

- We investigate Mamba model compared to the transformer model in traditional RL tasks, D4RL, and find that Mamba model is more efficient but slightly inferior to the transformer model in terms of effectiveness.
- We propose DM-H, an in-context RL method that connects Mamba and the transformer with high-value sub-goals. DM-H inherits the merits of Mamba and transformers, achieving both high effectiveness and efficiency in long-term tasks.
- Our extensive experiments across Grid World, D4RL, and Tmaze reveal the superiority of DM-H over other baselines. In the online testing of long-term tasks, DM-H can be 28$\times$ times faster than baselines and more than double the effectiveness.

2 Related Work

Mamba for Long Sequence Modeling. The structured State-Space Sequence (S4) model is a novel alternative to CNNs or transformers to model the long-term dependency [18]. The promising property of linearly scaling in sequence length attracts further exploration. Based on the S4, Smith et al. [40] propose a new S5 layer by introducing MIMO SSM and an efficient parallel scan into the S4 layer. Fu et al. [12] design a new SSM layer, H3, that nearly fills the performance gap between SSMs and transformers in language modeling. Mehta et al. [32] build the Gated State Space layer on S4 by introducing more gating units to improve the expressivity. Recently, Gu and Dao [17] propose a data-dependent SSM layer and build a generic language model backbone, Mamba, which outperforms transformers at various sizes on large-scale real data and enjoys linear scaling in sequence length. Later, Vision Mamba adds position coding and bidirectional scanning to extend it to visual tasks [47]. In this work, we explore transferring Mamba's success to RL, i.e., achieving high effectiveness and efficiency for long-term memory.

Transformer for Decision-Making. In general, reinforcement learning was proposed as a fundamental online paradigm [41]. The nature of online learning comes with some limitations when meeting the applications for which it is impossible to gather online data and learn simultaneously, such as autonomous driving. To this end, offline RL proposed that the agent can learn from a fixed offline dataset without gathering new data during learning [15, 28, 45, 27, 21, 23, 19, 24]. In the

context of offline RL, recent works explored using transformer-based policy by treating RL tasks as a type of sequential prediction problem [22]. Among them, a decision transformer [5] was proposed to model trajectories as sequences and autoregressively predict action conditioning on desired return-to-go, past states, and actions. Trajectory transformer [26] demonstrated that transformer could learn single-task policies from offline data. Subsequently, the multi-game decision transformer [31] and Gato [39] further showed that transformer-based policies could address multi-tasks in the same domain and cross-domain tasks. However, these works focused on distilling expert policies from offline data and failed to enable self-improvement like DM-H. When the offline data are sub-optimal, or the agent is required to adapt to new tasks, the multi-game decision transformers need to finetune the model parameters, while Gato is required to get prompted with expert demonstrations.

Meta RL. DM-H falls into the category of methods of learning to learn, which is also known as meta-learning. More precisely, recent in-context RL methods can be categorized as in-context meta-RL methods. The general idea of learning self-improvement has a long history in RL but is limited to hyper-parameters in the early stages [25]. In-context meta-RL methods [44, 10] are commonly trained in the online setting by maximizing multi-episodic value functions with memory-based architectures through environment interactions. Another online meta-RL attempts to find good network parameter initializations and then quickly adapt through additional gradient updates [11, 36]. More recently, meta-RL has seen substantial breakthroughs, from performance gains on popular benchmarks to offline settings, such as Bayesian RL [8] and optimization-based meta-RL [33]. Considering the difficulty of a completely offline setting, recent work has explored hybrid offline-online settings [46, 37]. DM-H is similar to the hybrid offline-online setting but saves more computing resources because the online phase does not involve gradient updates.

In-Context RL. In-context RL is the one that addresses tasks by providing prompts or demonstrations [5, 26]. By training agents at a large scale, transformer-based policies usually have the ability to learn in context [31, 39]. The learning process is performed entirely in context and does not involve parameter updates of neural networks. In this work, we consider incremental in-context RL, which involves learning from one's own behaviors in a trial-and-error manner. Laskin et al. [29] proposed Algorithm Distillation (AD), which automatically improved its performance by providing multiple historical trajectories. Subsequently, Lee et al. [30] proposed a Decision-Pretrained Transformer, which trained the agent to find optimal behaviors faster by only predicting the optimal trajectory. More recently, Hao Liu [20] further demonstrated that across-episodic contexts encourage large transformer models' emerging self-improvement behaviors. However, these methods suffer from huge computational costs as across-episodic contexts induce too-long sequences. In contrast, DM-H leverages Mamba's ability to efficiently process long-term dependencies while using the transformer to establish high-quality predictions.

3 Preliminaries

Partially Observable Markov Decision Process. We consider learning problems in the context of Partially Observable Markov Decision Processes (POMDP) represented by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, P, \mathcal{R})$. The POMDP tuple consists of states $s \in \mathcal{S}$, observations $o \in \mathcal{O}$, actions $a \in \mathcal{A}$, rewards $r \in \mathcal{R}$, and a transition probability function $P(s_{t+1}|s_t, a_t)$, where t is an integer denoting the timestep. At each timestep t, the agent receives the observation o_t, selects an action $a_t \sim \pi(\cdot|o_t)$ based on its policy, and then receives the next observation o_{t+1}. For convenience, we uniformly use s to denote the observations or states received from the environment. A trajectory is a sequence that consists of observations, actions, and rewards and is denoted by $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$. In addition, a completion token d_t, a binary identifier, is used to indicate whether a trajectory ends at time t.

Transformers. The Transformer [43] architecture consists of multiple layers of self-attention operation and MLP. The self-attention begins by projecting input data X with three separate matrices onto D-dimensional vectors called queries Q, keys K, and values V. These vectors are then passed through the attention function:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{D})V. \tag{1}$$

The QK^T term computes an inner product between two projections of the input data X. The inner product is then normalized and projected back to a D-dimensional vector with the scaling term V. Transformers utilize self-attention as a core part of the architecture to process sequential data [3, 7].

 https://doi.org/10.52202/079017-2314

Table 1: Mamba vs. Transformer on D4RL datasets.

Environment		HalfCheetah			Hopper			Walker2d		
Dataset		Med-Expert	Medium	Med-Replay	Med-Expert	Medium	Med-Replay	Med-Expert	Medium	Med-Replay
Effectiveness	Transformer	**94.21**± **0.46**	**42.28**± **1.18**	**41.28**± **0.21**	108.32± 0.95	72.58± 0.54	**91.32**± **0.66**	**111.36**± **0.46**	**85.96** ± **0.46**	**89.21**± **1.42**
	Mamba	92.21± 0.60	41.92± 0.11	39.68± 0.13	**110.82**± **0.56**	**73.65**± **1.23**	82.65± 1.15	108.31± 0.52	78.26± 0.55	70.92± 1.21
Efficiency (hour)	Transformer	37.10± 0.22			22.23± 0.15			33.77± 0.26		
	Mamba	**28.56**± **0.18**			**18.15**± **0.16**			**26.25**± **0.21**		

In this work, we use GPT [38] architecture that modifies the transformer with a causal self-attention mask to focus on the previous tokens in the sequence ($j \in [1, i]$), enabling us to do autoregressive generation at test time.

S4 and Mamba. S4 [18] and Mamba [17] are inspired by the continuous system, which maps a $1-$D function or a sequence $x(t) \in \mathbb{R} \rightarrow y(t) \in \mathbb{R}$ through a hidden state $h(t) \in \mathbb{R}^N$. The mapping process can be represented as the following linear ordinary differential equation:

$$\begin{aligned} h'(t) =& \mathbf{A}h(t) + \mathbf{B}x(t), \\ y(t) =& \mathbf{C}h(t), \end{aligned} \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ denotes the evolution parameter, $\mathbf{B} \in \mathbb{R}^{N\times 1}$ and $\mathbf{C} \in \mathbb{R}^{1\times N}$ denote the projection parameters. For application to a discrete input sequence instead of a continuous function, S4 uses the zero-order hold to transform the continuous parameters $\mathbf{A}, \mathbf{B}$ to discrete parameters $\overline{\mathbf{A}}, \overline{\mathbf{B}}$. Then, the Equation (2) can be rewritten as:

$$\begin{aligned} h_t =& \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \\ yt =& \mathbf{C}h_t, \end{aligned} \tag{3}$$

where $\overline{\mathbf{A}} = \exp(\Delta)\mathbf{A}$, $\overline{\mathbf{B}} = (\delta\mathbf{A})^{-1}(\exp(\Delta)\mathbf{A} - \mathbf{I})(\Delta\mathbf{B})$, and Δ is a timescale parameter. Based on the S4 framework, Mamba introduces a data-dependent selection mechanism while leveraging a hardware-aware parallel algorithm in recurrent mode. Compared with the Transformer, the combined architecture of Mamba empowers it to capture contexts effectively and maintains computational efficiency, particularly for long sequences.

4 Method

In this section, we first compare Mamba and transformer models in the D4RL dataset, and investigate the potential of Mamba in RL tasks. Then, We present DM-H, which can handle long-term dependencies from contexts with high effectiveness and efficiency, as shown in Figure 1.

4.1 Mamba vs. Transformer in RL tasks

We first consider the Algorithm Distillation (AD) as the baseline, which is a classic in-context RL method using a transformer as the backbone [29]. AD can predict high-quality actions by recalling the historical trajectories from the context, but it also incurs higher computational costs. Under the same settings, we replace the transformer in the AD algorithm with Mamba to compare their effectiveness and efficiency.

As shown in Table 1, the simple backbone replacement did not significantly improve effectiveness. Regarding efficiency, Mamba brings predictable improvements, thus saving training time under the same settings. Compared with the attention mechanism acting on state pairs, Mamba uses a data-dependent selection mechanism acting on each state independently, which brings a more efficient method of recalling long-term memory. However, since states in RL tasks commonly exhibit sequential relationships, the attention mechanism is more suitable for capturing the information between states, thereby outperforming Mamba in terms of effectiveness. To this end, we aim at a new in-context RL approach that leverages Mamba's strengths in processing long-term memory while preserving high-quality predictions from transformers.

4.2 Decision Mamba-Hybrid

In-context RL can automatically improve its performance through trial-and-error when across-episodic contexts serve as prompt conditions. Specifically, an across-episodic context consisting of n

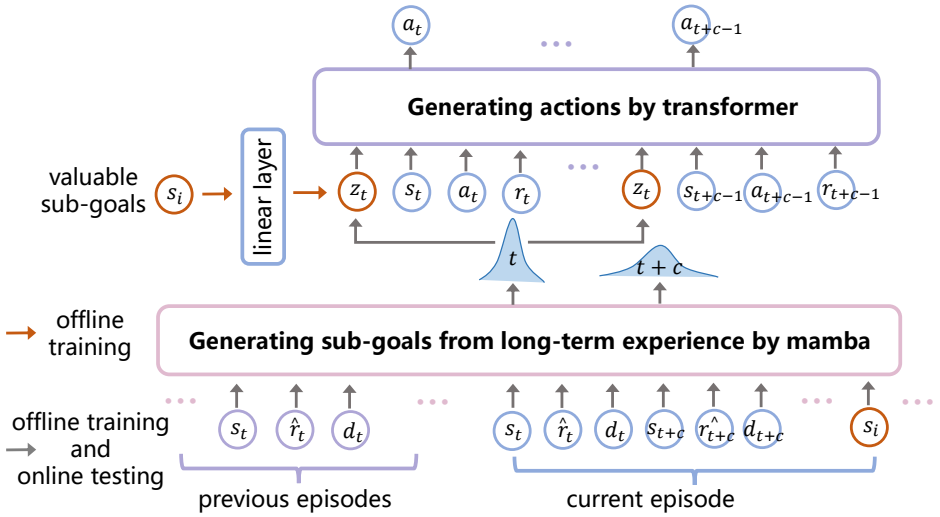

Figure 1: The architecture of DM-H. During offline training, Mamba module generates sub-goals from long-term experience, where the long-term experience consists of multiple historical trajectories arranged in ascending order of the total rewards. Based on the generated sub-goals, the transformer is required to predict better actions by supervising the expert behaviors. Meanwhile, the linear layer feeds the valuable sub-goals into the transformer module and associates them with the generated actions. During online testing, DM-H can automatically improve its performance in a trial-and-error manner without requiring gradient updates.

trajectories is represented as $(\tau^1, \tau^2, \ldots, \tau^n)$, where

$$\tau^i = (s_0^i, a_0^i, r_0^i, d_0^i, \ldots, s_T^i, a_T^i, r_T^i, d_T^i). \tag{4}$$

The trajectories are sorted according to their total rewards, i.e.,$\sum_{t=0}^{T} r_t^1 \leq \sum_{t=0}^{T} r_t^2 \leq \cdots \leq \sum_{t=0}^{T} r_t^n$. With autoregressive training and generation, the transformer can uncover meaningful patterns from multiple trajectories and improve itself conditioned on experience. However, the quadratic complexity of the attention mechanism suffers from huge computational costs along with the growth in task horizon. Inspired by Mamba's success with long sequences, we propose that Mamba handles across-episodic contexts and preserves local short-term sequences for the transformer. For Mamba, we reconstruct the long-term sequences $(\tau_m^1, \tau_m^2, \ldots, \tau_m^n)$ from across-episodic contexts. Each τ_m^i is denoted as

$$\tau_m^i = (s_0^i, \hat{r}_0^i, d_0^i, s_c^i, \hat{r}_c^i, d_c^i, \ldots, s_{kc}^i, \hat{r}_{kc}^i, d_{kc}^i), \tag{5}$$

where c represents the local sequence length for the transformer, $T - c \leq kc \leq T$, and $\hat{r}_c^i = \sum_{t=c}^{2c-1} r_t^i$ is the sum of c steps rewards. Mamba module will generate sub-goals to prompt the transformer, where the sub-goal is represented by a vector $\mathbf{z}$ sampled from a multivariate Gaussian distribution. Then, the local short-term sequence is represented as:

$$\tau_{\mathbf{z}}^{i,j} = (\mathbf{z}_j^i, s_j^i, a_j^i, r_j^i, \mathbf{z}_j^i, s_{j+1}^i, a_{j+1}^i, r_{j+1}^i, \ldots, \mathbf{z}_j^i, s_{j+c-1}^i, a_{j+c-1}^i, r_{j+c-1}^i), \tag{6}$$

where $\tau_{\mathbf{z}}^{i,j}$ starts from the generation step $j \in \{0, c, \ldots, kc\}$ and completes c steps actions based on the generated sub-goal $\mathbf{z}_j^i$.

4.3 Decision Mamba-Hybrid with Valuable Sub-goals

DM-H links Mamba and the transformer through the sub-goal $\mathbf{z}$ to efficiently recall long-term contexts while ensuring high-quality predictions. As sub-goals are commonly hidden in the offline data, Mamba must infer them along with the transformer training. However, it remains to be seen whether the transformer will benefit from $\mathbf{z}$ or ignore it and only imitate expert behaviors based on the local context. Therefore, we select extra high-value states from the offline data and transform them into sub-goals $\mathbf{z}$ to align actions generated by the transformer.

https://doi.org/10.52202/079017-2314

Valuable sub-goals. Intuitively, a sub-goal should be highly valuable for the agent to reach and have a high probability of appearing at subsequent time steps of the current state in the trajectory. When reached sequentially, these states should mark milestones in the trajectory, making it highly probable that the agent successfully performs the task. In this point, the valuable sub-goals guide the agent through the task, meaning they have the same purpose as the returns-to-go in the Decision Transformer [5]. Therefore, we can model this behavior by finding states with high accumulated reward values in the trajectory. Specifically, for state s_i at timestep i, the value of state s_j at timestep j is $\sum_{t=i+1}^{j} r_k$, where $i+1 \leq j \leq T$. However, this may prioritize the last state of trajectories when the environment only provides positive rewards. To encourage selecting states that are close to the current state s_i, we divide the accumulated rewards by the distance between their timesteps $\sum_{t=i+1}^{j} r_k/(j-i)$. With the punishment of the distance, the weighted average of accumulated rewards can identify the short-term and important future states.

Based on the selected sub-goals, we reconstruct the local short-term sequence (Equation (6)) by replacing $\mathbf{z}$ generated from Mamba. The reconstructed local short-term sequence is represented as:

$$\tau_g^{i,j} = (f(s_g^i), s_j^i, a_j^i, r_j^i, f(s_g^i), s_{j+1}^i, a_{j+1}^i, r_{j+1}^i, \ldots, f(s_g^i), s_{j+c-1}^i, a_{j+c-1}^i, r_{j+c-1}^i), \quad (7)$$

where s_g^i is the most valuable sub-goal for state s_j^i and f is a linear layer that maps s_g^i to the same dimension of $\mathbf{z}_j^i$. The reconstructed local short-term sequence aligns the actions generated by the transformer with subsequent high-valued states. Since Equation (6) is consistent with the reconstructed sequence except for $\mathbf{z}_j^i$, this encourages Mamba module to generate high-value sub-goals from the long-term context to ensure that the transformer module predicts similar actions.

4.4 Implementation of DM-H

Architecture. We feed n trajectories into Mamba module, which results in $3 \times n \times T/c$ tokens, with one token for each of the three modalities: state, reward, and completion. In the transformer module, we feed $4 \times c$ tokens, with one token for each of the four modalities: sub-goal, state, action, and reward. To create the token embeddings, we train a linear layer for each modality, which transforms the raw inputs into the desired embedding dimension, followed by layer normalization [2]. Finally, we freeze a linear layer that maps the high-value sub-goals s_g in Equation (7) to the same dimensions as the sub-goals $\mathbf{z}$ generated by Mamba.

Offline Training and Online Testing. During offline training, we are given a dataset of offline trajectories, where the trajectories can be suboptimal. In each iteration, we sample minibatches of trajectories from the dataset. Then, Mamba module first predicts the sub-goals $\mathbf{z}_t$ every c steps, given the input token s_t and past trajectories. Then, the transformer module autoregressively predicts c steps of actions $\{a_t, \ldots, a_{t+c-1}\}$ given $\mathbf{z}_t$ and $\{s_t, \ldots, s_{t+c-1}\}$. Meanwhile, we use the weighted average of accumulated rewards to select valuable sub-goals from the offline data and feed them to the transformer model to predict the same c steps actions. The predicted actions are evaluated with either cross-entropy loss or mean-squared error, depending on whether the actions are discrete or continuous. The losses from each step are averaged and updated in all modules end-to-end. At online testing, we roll out the DM-H with multiple trajectories and report the return of the last trajectory. Following the configuration from related works [20, 29], we set a context size across $n = 4$ episodes. The pseudocode for DM-H is summarized in Appendix A. Source code and more hyperparameters are described in Appendix B.

5 Experiments

In this section, we will introduce datasets and baselines in Section 5.1. Then, in Section 5.2, Section 5.3, Section 5.4, and 5.5, we report the comparison results, ablation study, and parameters sensitivity analysis. In Appendix C, we report additional results about offline training time, online testing time, and ablation study.

5.1 Environmental Settings

Dataset: Grid World. We first consider the discrete control environments from the Grid World [31], which is a commonly used benchmark for recent in-context RL methods. The environments support many tasks that cannot be solved through zero-shot generalization after pre-training because

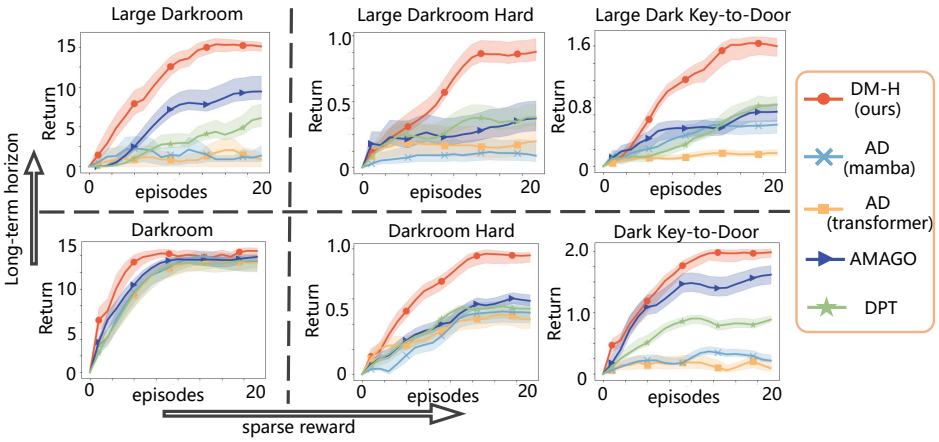

Figure 2: Results for Grid World. An agent is expected to solve a new task by interacting with the environments for 20 episodes without online model updates. Our DM-H significantly outperforms baselines on long-term tasks with sparse rewards because it inherits the merits of transformers and Mamba in high-quality prediction and long-term memory.

these tasks cannot be inferred easily from the observation. Specifically, we test our method on three environments: Darkroom, Darkroom Hard, and Dark Key-to-Door. In addition, we create a long-term variant of Large Darkroom, Large Darkroom Hard, and Large Darkroom Key-to-Door, where the coordinate space of each environment is expanded to 20 times, and the episode length is expanded 10 times. The dataset is collected from learning histories that are generated by training gradient-based RL algorithms, such as Deep Q-Network [34]. For each environment, we randomly create 60 tasks from the coordinate space and collect data for 1 million steps.

Dataset: Tmaze. We also evaluate our method on the Tmaze [35], a benchmark for testing the recall ability of in-context agents. In Tmaze, any policy that achieves the maximum return must be able to recall information from the first step at the final step. Since the task horizon can be set arbitrarily, it is often used to test the limits of the model's processing of long-term memory.

Dataset: D4RL. D4RL [13] is a commonly used offline RL benchmark, including continuous control tasks. The dataset is collected from Mujoco environments, including HalfCheetah, Hopper, and Walker. The episode length in D4RL is 1000, which is far more than that of Grid World. Therefore, current in-context RL methods require huge computational costs in D4RL, even though it is a commonly used benchmark for conventional RL algorithms.

Baselines. We investigate the effectiveness and efficiency of DM-H relative to in-context RL, dedicated offline RL, and imitation learning algorithms. Our baselines can be categorized as follows:

- In-context RL: These methods use the transformer to model trajectory sequences and predict actions autoregressively. We compare with recent methods, AMAGO [16], Decision Pretrained Transformer (DPT) [30], and Algorithm Distillation (AD) [29], which achieve impressive results based on the setting of across-episodic contexts.
- Temporal-difference learning: Most temporal-difference (TD) learning methods use an action space constraint or value pessimism and will serve as faithful comparisons to DM-H, representing standard RL methods. Following recent work [20], we consider state-of-the-art TD3+BC [14] that is demonstrated to be effective on D4RL.
- Imitation learning: Imitation learning methods similarly utilize supervised losses for training, such as Behavior Cloning (BC) [42] and Decision Transformer (DT) [5]. We compare with BC-10%, which is shown to be competitive with state-of-the-art on D4RL. DT also uses a transformer to predict actions autoregressively but is limited to a single episode context.

For all comparison methods, we adhere closely to the original hyper-parameter settings. To evaluate DM-H and other in-context RL algorithms, we roll out 10 episodes in D4RL and 20 episodes in Grid World and Tmaze. For each result, we report mean and standard error across 10 random seeds.

https://doi.org/10.52202/079017-2314

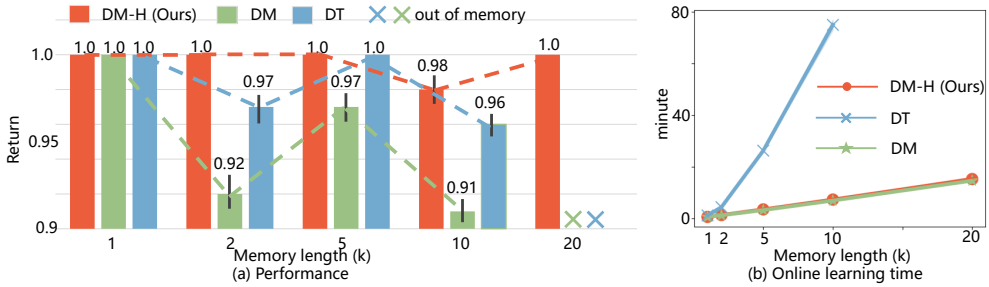

Figure 3: Results for (a) performance and (b) online testing times on Tmaze tasks. We train each method to address Tmaze tasks that have different horizons until we run out of GPU memory at context length to achieve 10k (DT, DM) or 20k (our DM-H). We report the online testing time for 20 episodes of Tmaze tasks.

Table 2: Results for D4RL datasets. DM-H outperforms both in-context RL (DT, AD) and supervised learning (BC) and performs competitively with conventional RL algorithms (TD3+BC and TD3) on almost all tasks.

Dataset	Environment	BC-10%	TD3+BC	TD3	DT	AD (Mamba)	AD (Transformer)	DM-H
Med-Expert	HalfCheetah	94.11	**96.59**	87.60	93.40	92.21± 0.60	94.21 ± 0.46	96.21± 0.28
Med-Expert	Hopper	113.13	113.22	98.41	111.18	110.82± 0.56	108.32 ± 0.95	**117.19± 0.65**
Med-Expert	Walker2d	109.90	112.21	100.52	108.71	108.31± 0.52	111.36 ± 0.46	**118.21± 0.56**
Med	HalfCheetah	43.90	**48.93**	34.60	42.73	41.92± 0.11	42.28 ± 1.18	45.45± 0.35
Med	Hopper	73.84	70.44	56.98	69.42	73.65± 1.23	72.58 ± 0.54	**83.15± 0.63**
Med	Walker2d	82.05	86.91	70.95	74.70	78.26± 0.55	85.96 ± 0.46	**88.29± 0.76**
Med-Replay	HalfCheetah	42.27	45.84	38.81	40.31	39.68± 0.13	41.28 ± 0.21	**45.26± 0.43**
Med-Replay	Hopper	90.57	98.12	78.90	88.74	82.65± 1.15	91.32 ± 0.66	**98.36± 0.51**
Med-Replay	Walker2d	76.09	91.17	65.94	71.22	70.92± 1.21	89.21 ± 1.42	**95.66± 1.16**
Total Average		80.65± 1.34	84.83± 1.10	70.28± 1.20	77.69± 1.45	76.97± 0.67	82.84± 0.70	**87.53± 0.59**

5.2 Grid World Results

To evaluate DM-H's self-improvement capabilities in unseen tasks, we compared recent in-context RL methods in the Grid World environments. The agent is required to solve an unseen task by interacting with the environments for 20 episodes without online model updates. In addition, we added a variant of the representative in-context RL AD, replacing the transformer with Mamba as the backbone.

As shown in Figure 2, DM-H achieves state-of-the-art performance in a wide range of tasks. DM-H achieves 28× times faster than baselines and more than double the effectiveness as the task horizon increases and rewards become sparse. In long-term tasks, DM-H leverages Mamba's ability to obtain long-term memories while retaining high-quality predictions established by the transformer in decision-making. On the contrary, AD (transformer) and AD (Mamba) only retain one aspect of the advantages of DM-H. In terms of sparse rewards, DM-H is similar to the structure of hierarchical RL, in which Mamba performs a decision every c steps and is, therefore, better at handling reward sparse scenarios. Overall, DM-H demonstrated that the hybrid method is not only feasible but combines more advantages synergetically.

5.3 Tmaze Results

In this section, we test the limits of DM-H's recall capabilities in the Tmaze task. Solving this task requires accurate recall of the first step at the last step. Therefore, we set the context length equal to the task horizon and compare it with DT and DM, where DM replaces the DT's transformer backbone with a Mamba backbone. We train each method to recover the optimal policy until we run out of GPU memory at a context size equal to 20k (DM-H) or 10k (DT and DM).

As shown in Figure 3, DM-H achieves the maximum reward with minimum online testing costs at any task horizon, demonstrating that Mamba's recalled memory positively prompts the transformer. Regarding efficiency, DM-H is comparable to DM using only Mamba during online testing and outperforms DM during offline training (Figure 5 in Appendix C). This is because (1) the context of

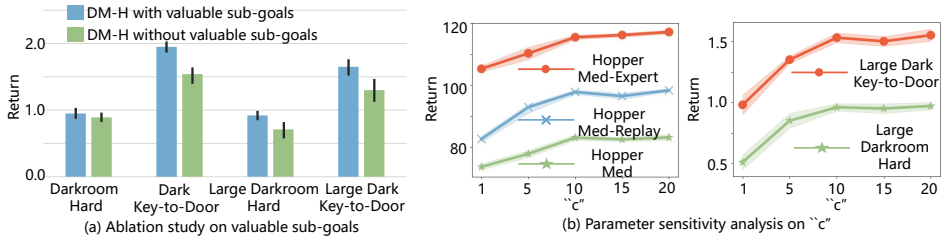

Figure 4: (a) The ablation study on DM-H with or without valuable sub-goals. (b) The parameter sensitivity analysis of "c."

the transformer in DM-H is fixed to the hyperparameter c, which does not change with the task length. (2) Mamba model generates a sub-goal every c steps, thereby shortens the sequence it processes by $c\times$ times. Benefiting from these merits, DM-H handles tasks twice the length and has fewer memory resource requirements than DT and DM.

5.4 D4RL Results

In addition to navigation tasks, we also test DM-H on the control tasks from the D4RL dataset, which is commonly used in conventional offline RL methods. Based on previous work [13], the results on D4RL are normalized so that 100 denotes an expert policy. Baseline numbers are reported by the Agentic Transformer [20] and from the D4RL paper. As shown in Table 2, DM-H outperforms baselines in a majority of the tasks and is competitive with the state-of-the-art in the remaining tasks. In the TD learning and imitation learning categories, TD3+BC is generally the most remarkable algorithm. Compared with them, the superior performance demonstrates the self-improvement of DM-H on suboptimal data.

5.5 Ablation Study and Parameter Sensitivity Analysis

Ablation Study on Valuable Sub-goals. To validate the effectiveness of valuable sub-goals, we conduct an ablation study of DM-H in Grid World tasks. Figure 4 (a) presents the ablation results, which report the mean episode return across 10 seeds. We can observe that sub-goals can significantly improve DM-H's performance, proving that the sub-goal strengthens the transformer's dependency on long-term contexts from Mamba. In Appendix C, we also carry out ablation studies in the D4RL dataset. Please refer to Figure 6.

Sensitivity of Key Hyperparameters. In this experiment, we introduce an important hyperparameter c. A large c enables Mamba model to scan multiple episodes with smaller context sizes, significantly improving the computational efficiency. In addition, c also controls the short-term context length in the transformer, which affects the quality of generated actions. As shown in Figure 4 (b), DM-H performs well within the appropriate range of c, i.e., $10 \leq c \leq 20$.

6 Conclusion, Limitations, and Broader Impacts

In this work, we propose an in-context RL method DM-H that achieves both high effectiveness and efficiency in long-term tasks. The core idea of DM-H is to use sub-goal settings to leverage Mamba's ability to efficiently recall long sequence contexts while using the transformer to perform high-quality predictions. Unlike current in-context RL methods limited to short-term tasks, DM-H is also good at standard RL benchmarks, which typically have long-term sequences under the across-episodic context setting. On the Grid World, D4RL, and Tmaze benchmarks, we demonstrate that DM-H can outperform baselines in both efficiency and effectiveness.

Regarding limitations, our method has an important hyperparameter, which is the context length of the transformer "c." As "c" increases, Mamba model can scan multiple episodes with smaller context sizes, significantly improving the computational efficiency. On the contrary, as "c" decreases, Mamba can generate high-quality sub-goals from the context that is closer to the original episode. However, this reduces the short-term context available to the transformer when making predictions. Therefore, a meaningful future direction is for "c" to adapt autonomously to different tasks. In terms of the

https://doi.org/10.52202/079017-2314

potential broader impact, we do not anticipate any negative ethical and societal impacts of our work while using our method in practice.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant No. 2021ZD0112500; the National Natural Science Foundation of China under Grant Nos. U22A2098, U2341229, 62172185, 61976102, 62206105, 62202200 and 62476110; the Key R&D Project of Jilin Province under Grant Nos. 20240212003GX and 20240304200SF; the International Cooperation Project of Jilin Province under Grant No. 20220402009GH.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35: 23716–23736, 2022.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] 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.

[4] 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.

[5] 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.

[6] Yang Dai, Oubo Ma, Longfei Zhang, Xingxing Liang, Shengchao Hu, Mengzhu Wang, Shouling Ji, Jincai Huang, and Li Shen. Is mamba compatible with trajectory optimization in offline reinforcement learning? *arXiv preprint arXiv:2405.12094*, 2024.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Advances in Neural Information Processing Systems*, 34:4607–4618, 2021.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[12] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

[13] 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.

[14] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[15] 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.

[16] Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. 2024.

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

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

[19] Siyuan Guo, Yanchao Sun, Jifeng Hu, Sili Huang, Hechang Chen, Haiyin Piao, Lichao Sun, and Yi Chang. A simple unified uncertainty-guided framework for offline-to-online reinforcement learning. *arXiv preprint arXiv:2306.07541*, 2023.

[20] Pieter Abbeel Hao Liu. Emergent agentic transformer from chain of hindsight experience. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2023)*, 2023.

[21] Jifeng Hu, Li Shen, Sili Huang, Zhejian Yang, Hechang Chen, Lichao Sun, Yi Chang, and Dacheng Tao. Continual diffuser (cod): Mastering continual offline reinforcement learning with experience rehearsal. *arXiv preprint arXiv:2409.02512*, 2024.

[22] Sili Huang, Jifeng Hu, Hechang Chen, Lichao Sun, and Bo Yang. In-context decision transformer: Reinforcement learning via hierarchical chain-of-thought. In *Forty-first International Conference on Machine Learning*.

[23] Sili Huang, Hechang Chen, Haiyin Piao, Zhixiao Sun, Yi Chang, Lichao Sun, and Bo Yang. Boosting weak-to-strong agents in multiagent reinforcement learning via balanced ppo. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[24] Sili Huang, Yanchao Sun, Jifeng Hu, Siyuan Guo, Hechang Chen, Yi Chang, Lichao Sun, and Bo Yang. Learning generalizable agents via saliency-guided features decorrelation. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] Shin Ishii, Wako Yoshida, and Junichiro Yoshimoto. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4-6):665–687, 2002.

[26] 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.

[27] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] 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.

[29] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Stenberg Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. In *The Eleventh International Conference on Learning Representations*, 2023.

[30] Jonathan N Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023.

 https://doi.org/10.52202/079017-2314

[31] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

[32] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022.

[33] Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pages 7780–7791. PMLR, 2021.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[35] Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[37] Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pages 17811–17829. PMLR, 2022.

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

[39] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[40] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

[41] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[42] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[43] 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.

[44] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[45] 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.

[46] Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. *Advances in neural information processing systems*, 33:20913–20924, 2020.

[47] 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.

Appendix of paper "Decision Mamba: Reinforcement Learning via Hybrid Selective Sequence Modeling"

A Pseudocode of Decision Mamba-Hybrid

Algorithm 1: Decision Mamba-Hybrid.

Input: A dataset of Trajectories, Max Iterations M as training phase, Max episodes m at testing phase, A number of trajectories n in across-episodic contexts used in Mamba model, A number of steps of actions c for one sub-goals

Output: The generated actions

1 **//Training**
2 **for** $i = 1$ ***to*** M **do**
3 Randomly sample n episodes from dataset $s = (\tau^1, \tau^2, \ldots, \tau^n)$
4 Sort n episodes ascending according to their returns $\sum_{t=0}^{T} r_t^1 \leq \sum_{t=0}^{T} r_t^2 \leq \cdots \leq \sum_{t=0}^{T} r_t^n$
5 Concatenate n episodes as a across-episodic context $s_h = (\tau_m^1, \tau_m^2, \ldots, \tau_m^n)$ based on Equation (5)
6 Mamba module predicts the next sub-goals tokens from the across-episodic context
7 Build a local short-term sequence every c steps based on Equation (6)
8 Select the high-value states from the offline data based on the weighted average of accumulated rewards $\sum_{t=i+1}^{j} r_k/(j-i)$ Build a similar local short-term sequence every c steps based on Equation (7)
9 The transformer module predicts the next c steps action tokens for each predicted sub-goals token from Mamba and selected high-value sub-goals
10 Train Mamba and transformer models based on the loss of predicted actions end-to-end
11 **end**
12 **//Testing**
13 **for** $i = 1$ ***to*** m **do**
14 Start a new episode i and reset the timestep $t = 0$
15 **while** $t \leq T$ **do**
16 Mamba model generates next sub-goal token $\mathbf{z}_t^i$ based on the historical trajectories $(\tau_m^1, \tau_m^2, \tau_m^{i-1}, \ldots, s_t^i)$, where τ_m^i is expressed as Equation (5)
17 **for** $k = 0$ ***to*** $c - 1$ **do**
18 The transformer model generates next action a_{t+k}^i based on the local context $(\mathbf{z}_t^i, s_t^i, a_t^i, r_t^i, \ldots, \mathbf{z}_t^i, s_{t+k}^i)$
19 **end**
20 Compute the sum of c steps rewards $\hat{r}_t^i = \sum_{k=t}^{t+c-1} r_k^i$
21 Receive the next observation or state s_{t+c}^i
22 Update the across-episodic context $(\tau_m^1, \tau_m^2, \tau_m^{i-1}, \ldots, s_t^i, \mathbf{z}_t^i, \hat{r}_t^i, d_t^i, s_{t+c}^i)$
23 Update time step $t = t + c$
24 **end**
25 **end**

In Algorithm 1, we introduce the training and testing process of DM-H. At each iteration, we first construct a long-term sequence consisting of multiple trajectories, as described in lines 3-5. Mamba model predicts sub-goals based on the long-term sequence (line 6). Then, each sub-goal will correspond to a short sequence of c steps actions, as described in line 7. To encourage the transformer to rely on Mamba's predictions, we select the high-value states from the offline data and transform them into sub-goals to reconstruct another short sequence of c steps actions (line 8). Based on the short sequence and reconstructed sequences, the transformer model predicts c steps actions (line 9). Finally, the predicted actions are evaluated with either cross-entropy loss or mean-squared error, depending on whether the actions are discrete or continuous. The losses from each time step are averaged and updated in all models end-to-end, as described in line 10.

During online testing, DM-H needs to generate actions autoregressively and interact with the environment in m episodes. At step t of episode i (line 16), Mamba module first generates a sub-goal token $\mathbf{z}_t^i$

https://doi.org/10.52202/079017-2314

Table 3: Hyperparameters of DM-H.

	Hyperparameters	Value
Mamba	Number of layers	2
	Embedding dimension	128
	Expand factor	2
	Convolution size	4
Transformer	Number of layers	3
	Number of attention heads	3
	Embedding dimension	128
	Activation function	ReLU
	c steps controlled by one sub-goal	20 D4RL and Large Grid World 5 Grid World and Tmaze
Training	Batch size	128
	Dropout	0.1
	Learning rate	1e-4
	Learning rate decay	Linear warmup for 1e5 steps
	Grad norm clip	0.25
	Weight decay	1e-4
	Number of trajectories to form across-episodic contexts n	4 (Large) Dark Key-to-Door 10 other tasks in Grid World 4 D4RL
Testing	Target return for Tmaze	1
	Target return for HalfCheetah	12000
	Target return for Hopper	3600
	Target return for Walker	5000
	Target return for Darkroom	20
	Target return for Darkroom Hard	1
	Target return for Darkroom Key-to-Door	2
	Target return for Large Darkroom	15
	Target return for Large Darkroom Hard	1
	Target return for Large Darkroom Key-to-Door	2
	Number of trajectories to form across-episodic contexts n	4 (Large) Dark Key-to-Door 10 other tasks in Grid World 4 D4RL

conditioned on the historical context $(\tau_m^1, \tau_m^2, \tau_m^{i-1}, \ldots, s_t^i)$, where τ_m^i is expressed as Equation (5). Then, the transformer will generate the following c steps actions $(a_t, \ldots, a_{t+c-1})$ autoregressively, as described in lines 17-19. Finally, we update the across-episodic context for generating the next sub-goal $\mathbf{z}_{t+c}^i$, as described in lines 20-23.

B Experimental Details

Dataset: Grid World. The evaluation environments of Grid World provide a 2D discrete POMDP where an agent spawns in a room and must find a goal location. The agent only observes its own (x, y) coordinates but does not know the goal location, which is required to deduce it from the rewards received. The room dimensions are 9×9 with the agent's possible actions, including moving one step either left, right, up, down, or staying idle. In Darkroom, an episode lasts 20 steps, and the agent can obtain a reward ($r = 1$) each time the goal is achieved. The Darkroom Hard is a variant of Darkroom. In the Darkroom Hard, agents only obtain a reward when the goal is achieved first. In the Dark Key-to-Door, the length of an episode is 50, where the agent is required to locate an invisible key to receive a one-time reward first and then identify an invisible door to obtain another one-time reward. In addition, we create a long-term variant of Large Darkroom, Large Darkroom Hard, and Large Darkroom Key-to-Door, where the coordinate space of each environment is expanded to 40×40, and the episode length is expanded 10 times.

Dateset: D4RL. D4RL [13] is a commonly used offline RL benchmark, including continuous control tasks. The different dataset settings are described below.

- Medium: 1 million timesteps generated by a "medium" policy that performs approximately one-third as well as an expert policy.

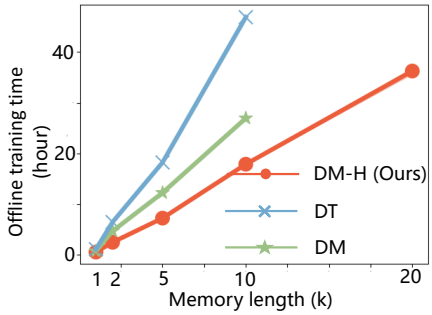

Figure 5: Results for offline training times on Tmaze tasks. We train each method to address Tmaze tasks that have different horizons until we run out of GPU memory at context length to achieve 10k (DT, DM) or 20k (DM-H). We report the training times for 10k gradient updates on Tmaze tasks.

Table 4: Results for offline training and online testing times. We report the offline training time per 10k gradient updates, the online testing time for 20 episodes over Grid World, and 10 episodes over D4RL. As the task length increases, the context length is forced to grow exponentially, resulting in a square increase in computational costs. In contrast, DM-H completes trial-and-error on Mamba model in sizes smaller than other baselines, significantly reducing computational costs.

Context size (step)	Tasks	Offline training (hour)			Online testing (minute)		
		AD (Mamba)	AD (Transformer)	DM-H (Ours)	AD (Mamba)	AD (Transformer)	DM-H (Ours)
200	Darkroom	0.21	0.23	0.18	0.20	0.61	0.21
	Darkroom Hard	0.22	0.28	0.20	0.20	0.56	0.19
	Darkroom Dynamic	0.24	0.31	0.21	0.21	0.62	0.22
	Dark Key-to-Door	0.65	1.01	0.41	0.45	1.50	0.46
2000	Large Darkroom	3.52	4.70	2.38	5.06	45.08 (9×)	5.12
	Large Darkroom Hard	4.26	6.69	2.78	5.61	44.96 (7×)	5.59
	Large Darkroom Dynamic	2.71	5.84	2.63	5.21	42.12 (8×)	5.36
	Large Dark Key-to-Door	6.87	18.23	3.16	5.88	76.79 (12×)	6.08
4000	HalfCheetah	28.56	37.10	20.96	6.16	173.11 (28×)	6.21
	Walker2d	26.25	33.77	19.96	6.01	172.34 (26×)	6.11
	Hopper	18.15	22.23	11.52	6.05	172.92 (26×)	6.06

- Medium-Replay: 1 million timesteps collected from the replay buffer of an agent trained to the performance of a “medium" policy.
- Medium-Expert: It consists of 1 million timesteps generated by the “medium" policy and another 1 million timesteps generated by the expert policy.

Dataset: Tmaze. The Tmaze task provides a T-shaped maze where the agent is rewarded only by reaching the end point at the last step. However, the endpoint’s location is unknown; it is only told once, at an oracle early in the maze. Therefore, any policy that achieves the maximum return must be able to recall information from the first step at the final step.

Compute. Experiments are carried out on NVIDIA GeForce RTX 3090 GPUs and NVIDIA A10 GPUs. Besides, the CPU type is Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz. In particular, our method has lower memory requirements because it naturally shortens the across-episodic contexts.

Hyperparameters. The default length of across-episodic is four trajectories unless mentioned otherwise. In D4RL, Tmaze, and Large Grid World, the transformer model generates $c = 20$ steps actions while Mamba model generates one sub-goal. In conventional Grid World, we set $c = 5$ because the task is too short. In summary, Table 3 shows the hyperparameters used in our DM-H model.

C Additional Experimental Results

Additional Tmaze Results. As described in the experiments section, we use Tmaze tasks to test the limits of DM-H’s recall capabilities. We train each method to recover the optimal policy until

https://doi.org/10.52202/079017-2314

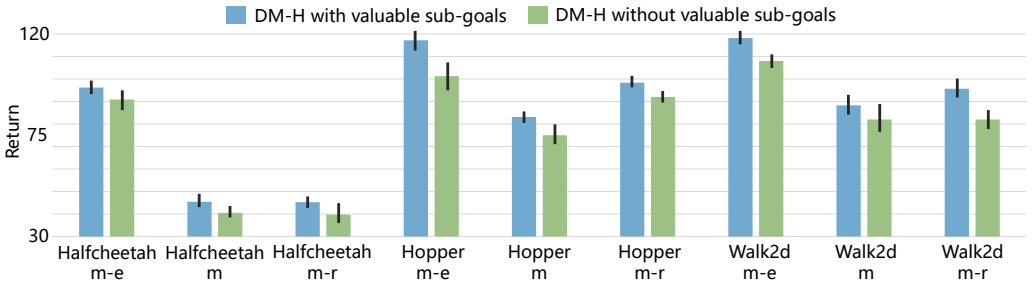

Figure 6: The ablation study on DM-H with or without valuable sub-goals.

we run out of GPU memory at a context size equal to 20k (DM-H) or 10k (DT and DM). As shown in Figure 5, DM-H achieves the maximum reward with minimum offline training costs at any task horizon. Regarding efficiency, DM-H is even faster than DM using only Mamba. This is because (1) the context of the transformer in DM-H is fixed to the hyperparameter c, which does not change with the task length. (2) Mamba model generates a sub-goal every c steps, which shortens the sequence it processes by $c\times$ times.

Additional Evaluation of Computing Costs. An important property of in-context RL is that it can improve itself without expensive gradient updates during online testing. However, the computational costs of forward propagation are hidden in short-horizon tasks. Therefore, we reported the offline training time per 10k gradient updates, the online time for 20 episodes over Grid World, and 10 episodes over D4RL. As shown in Table 4, our DM-H has efficient training and significantly reduces the online testing time compared to the baselines, approximately **28**$\times$ times faster in D4RL and **12**$\times$ times faster in large Grid World. As the task length increases, the online testing time of AD grows quadratically. This is because the across-episodic contexts multiply the sequence length, leading to intolerable computational costs in the self-attention mechanism. In contrast, DM-H leverage Mamba to process long-term sequence, where the computational complexity increases linearly with length.

Additional Ablation Study on Valuable Sub-goals. To validate the effectiveness of valuable sub-goals, we also conduct an ablation study of DM-H in D4RL tasks. Figure 6 presents the ablation results, which report the mean episode return across 10 seeds. We can observe that sub-goals can significantly improve DM-H's performance, proving that the sub-goal strengthens the transformer's dependency on long-term contexts from Mamba.

https://doi.org/10.52202/079017-2314

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: **[TODO]**

   Guidelines:

   - 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: We discuss the limitations 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]

https://doi.org/10.52202/079017-2314

Justification: **[TODO]**

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: The experiments and implementation details are shown in Section 5 and Appendix A.

   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: **[TODO]**

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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to Appendix B for more details.

   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: **[TODO]**

   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.

https://doi.org/10.52202/079017-2314

- 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: We report the computer resources of our experiments in Appendix B.

   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: **[TODO]**

   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: [Yes]

    Justification: In Section 6, we discuss the broader impacts.

    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: **[TODO]**

    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: **[TODO]**

    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?

https://doi.org/10.52202/079017-2314

Answer: [NA]

Justification: **[TODO]**

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: **[TODO]**

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: **[TODO]**

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.