Rethinking Transformers in Solving POMDPs

Chenhao Lu [1] **Ruizhe Shi** [1 *] **Yuyao Liu** [1 *] **Kaizhe Hu** [1] **Simon S. Du** [2] **Huazhe Xu** [1 3 4]

Abstract

Sequential decision-making algorithms such as reinforcement learning (RL) in real-world scenarios inevitably face environments with partial observability. This paper scrutinizes the effectiveness of a popular architecture, namely Transformers, in Partially Observable Markov Decision Processes (POMDPs) and reveals its theoretical and empirical limitations. We establish that regular languages, which Transformers struggle to model, are reducible to POMDPs. This poses a significant challenge for Transformers in learning POMDP-specific inductive biases, due to their lack of inherent recurrence found in other models like RNNs. This paper casts doubt on the prevalent belief in Transformers as sequence models for RL and proposes to introduce a point-wise recurrent structure. The Deep Linear Recurrent Unit (LRU) emerges as a well-suited alternative for Partially Observable RL, with empirical results highlighting the sub-optimal performance of Transformer and considerable strength of LRU. Our code is open-sourced[1].

1. Introduction

Reinforcement Learning (RL) in the real world confronts the challenge of incomplete information (Dulac-Arnold et al., 2019) due to partial observability, necessitating decision-making based on historical data. The design of RL algorithms under partial observability, denoted as **Partially Observable RL** (Kaelbling et al., 1998; Littman & Sutton, 2001; Li et al., 2015), typically employs a hierarchical structure combining $(\texttt{SEQ}, \texttt{RL})$. This structure involves firstly feeding the history into a sequence model `SEQ`, such as Recurrent Neural Network (RNN) (Elman, 1990) or Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), yielding a hidden state containing past information, then processing it using existing `RL` algorithms.

Regarding the sequence model, Transformer (Vaswani et al., 2017), renowned for its achievements in the natural language processing (NLP) domain (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023), stands out as a prominent candidate. Transformers have shown a strong ability to handle contexts in a non-recurrent manner. Compared with their recurrent counterpart like RNNs, the advantages of Transformers as a sequence model shine in several aspects: 1) long-term memory capacity, as opposed to RNNs with rapid memory decay (Ni et al., 2023; Parisotto et al., 2019); 2) effective representation learning from context for specific tasks (Micheli et al., 2022; Laskin et al., 2022; Lee et al., 2022; Robine et al., 2023), benefiting meta-RL or certain environments (Bellemare et al., 2013); 3) stronger learning ability on large-scale datasets (Baker et al., 2022).

However, deploying Transformers in Partially Observable RL introduces challenges, commonly manifesting as sample inefficiency (Parisotto et al., 2019; Ni et al., 2023). This issue is similarly observed in computer vision (CV) and is attributed to the data amount that Transformers require to learn the problem-specific inductive biases (Dosovitskiy et al., 2021). While it is validated in CV, it remains unknown whether data amount is the key ingredient in decision making. Hence, a natural question arises: *Can Transformers effectively solve decision-making problems in POMDPs with sufficient data?*

In this work, we investigate this critical question and challenge the conventional wisdom. We argue that Transformers cannot solve POMDP even with massive data. This stance is inspired by a key observation: While most RNNs are complete for regular languages, Transformers falter to model them (Delétang et al., 2023; Hahn, 2020b) . A notable example is their struggle with tasks like `PARITY`, which is to determine the parity of the occurrence of "1" in a binary string. We hypothesize that, in POMDPs, this limitation becomes pronounced due to the close relationship between regular languages and POMDPs.

To elaborate further, regular languages exhibit a direct correspondence with Hidden Markov Models (HMMs) (Carrasco & Oncina, 1994), and POMDPs can be regarded as HMMs augmented with an incorporated decision-making process.

[*]Equal contribution [1]IIIS, Tsinghua University [2]University of Washington [3]Shanghai Qi Zhi Institute [4]Shanghai AI Lab. Correspondence to: Chenhao Lu <luch21@mails.tsinghua.edu.cn>, Huazhe Xu <huazhe_xu@mail.tsinghua.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]https://github.com/CTP314/TFPORL

We further establish that regular languages can be effectively reduced to POMDPs. From the computational complexity perspective, the parallel structure of the Transformer makes it equivalent to a constant-depth computation circuit. Some regular languages fall outside of this complexity class, making the POMDP problems derived from them harder, and Transformer would struggle to solve them. This is demonstrated both theoretically and empirically in this study.

To alleviate the limitations of Transformers caused by the parallel structure, we propose to introduce a pointwise recurrent structure. Upon reviewing current variants of sequence models with such a structure, we find that they can be broadly generalized as linear RNNs. Based on extensive experiments over a range of sequence models over POMDP tasks with diverse requirements, we highlight LRU (Orvieto et al., 2023) as a linear rnn model well-suited for Partially Observable RL. Our contributions are three-fold:

- We demonstrate the theoretical limitations of Transformers as sequence model backbones for solving POMDPs, through rigorous analysis.
- To better utilize the inductive bias of the sequence model, we study the advantages of the Transformer and the RNN, and advocate the linear RNN as a better-suited choice for solving POMDPs, taking advantage of both models.
- Through extensive experiments across various tasks, We compare the capabilities exhibited by various sequence models across multiple dimensions. Specifically, we show that Transformers exhibit suboptimal performance as the sequence model in certain POMDPs, while highlighting the strength of linear RNNs when assessed comprehensively.

2. Related Work

Theoretical limitations of Transformers. There is a substantial body of work investigating the theoretical limitations of Transformers from the perspective of computational complexity and formal language. For example, Delétang et al. (2023); Huang et al. (2022) experimentally verifies that RNNs can recognize regular languages, but Transformers are unable to achieve this. Additionally, Hahn (2020a) demonstrates that Transformers are not robust in handling sequence length extrapolation. Moreover, Merrill & Sabharwal (2023); Merrill et al. (2022) point out that, under limited precision, TC^0 serves as an upper bound for the computational power of Transformers. Applying this result, Feng et al. (2023) illustrates the challenges Transformers face in solving practical problems such as arithmetic operations and linear systems of equations. Currently, works such as Ni et al. (2023); Morad et al. (2023); Deng et al. (2023) discuss the pros and cons of transformers in RL algorithms, with a focus on analyzing the advantages or providing simple evaluations. In contrast, integrating relevant theories from formal languages, we offer a new theoretical perspective on analyzing the limitations of transformers in RL.

Variants of sequence models for handling long contexts. Multiple variants of mainstream sequence models designed to handle long contexts have provided significant inspiration for this work. In the case of RNN-like models, addressing the issue of rapid memory decay has led to the emergence of linear RNNs (Gu et al., 2021; Orvieto et al., 2023), which remove activation functions in the recurrence part. These models have demonstrated excellent performances in benchmarks for long-range modeling (Tay et al., 2020). For Transformers, to tackle limited training length and inefficient inference, current studies emphasize the introduction of recurrence. Recurrence in Transformers can be categorized into two types: 1) *chunkwise recurrence*, which processes parts exceeding the context length using a recurrent block with minimal alterations to the original parallel structure (Dai et al., 2019; Hutchins et al., 2022); 2) *pointwise recurrence*, which derives recurrence representations by linearizing attention (Sun et al., 2023; Peng et al., 2023; Schlag et al., 2021; Katharopoulos et al., 2020). We argue that linear RNNs and pointwise recurrence Transformers ultimately converge to a similar solution, incorporating the strength of both approaches and are suitable for solving Partially Observable RL problems.

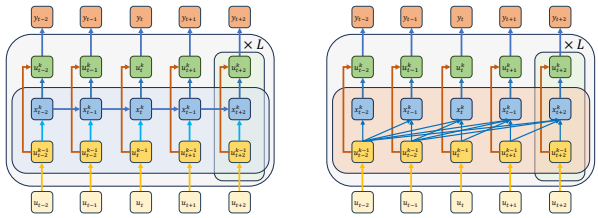
Applications of sequence models in RL. In recent years, there have been many applications of Transformers in RL, such as Decision Transformer (DT) (Chen et al., 2021), its variants (Yamagata et al., 2023; Wu et al., 2023) in offline RL; GTrXL (Parisotto et al., 2019), Online DT (Zheng et al., 2022) in online RL, and the Transformer State Space Model (Chen et al., 2022) as world models. There are works (Reid et al., 2022; Shi et al., 2023) showing that the inductive bias of pre-trained Transformers could help RL, where the states are fully observable. Hu et al. (2023) uses Transformer to solve a specific partially observable setting, frame drops, whereas demanding additional assumptions on the prior distribution. On the other hand, recent works comparing Transformer-based and RNN-based approaches in Partially Observable RL empirically support our idea that Transformers have weaknesses in partially observable environments (Morad et al., 2023; Deng et al., 2023). In many cases, simpler architectures like RNNs and LSTMs prove to be more effective. For instance, DreamerV3 (Hafner et al., 2023), which adopts GRU (Dey & Salem, 2017) as the backbone of the Recurrent State Space Model, has outperformed previous Transformer-based approaches like VPT (Baker et al., 2022) and IRIS (Micheli et al., 2022). Additionally, there has been a recent line of research on the application of linear RNNs in RL (Irie et al., 2021; Lu et al., 2024; Samsami et al., 2024; **?**), which has shown promising results.

This paper will incorporate insights into the limitations of Transformers to analyze why this would be a natural choice.

3. Preliminaries

Sequential neural network. Sequential Neural Networks are a type of deep learning model for sequence modeling. Given a input sequence $\{u_i\}_{i=1}^n$, the model learns a hidden state sequence $\{x_i\}_{i=1}^n$ and yields the output sequence $\{y_i\}_{i=1}^n$. There are currently two mainstream methods for computing the hidden state x_i:

- Recurrent-like: $x_t = \sigma(Ax_{t-1} + Bu_{t-1} + c)$ where σ is the activation function;
- Attention-like: $x_t = \mathrm{attn}\left(W^Q\tilde{U}, W^K\tilde{U}, W^V\tilde{U}\right)_t$. Here $\tilde{U}_i = u_i + p_i$, p_i is a position embedding, and $\mathrm{attn}(Q, K, V)$ is defined as $\mathrm{softmax}(QK^\top + M)V$, where M is an attention mask.



(a) (Pointwise) Recurrent (b) (Parallel) Attention

Figure 1. The Left figure indicates the general structure of recurrent-like sequential neural networks and the right figure represents the attention-like ones. Both of them can be deepened by pointwise transformations and skip connections.

Current sequence models use multiple layers. The overall structure is illustrated in Figure 1. For recurrent models like RNNs, GRUs, etc., the output of the previous layer is directly used as the input for the next layer., while for Transformers, the pointwise transformations typically take the form of MLPs with skip connections.

POMDP. A Partially Observable Markov Decision Process (POMDP) $\mathcal{M}$ can be defined as $(S, A, T, R, \Omega, O, \gamma)$ (Åström, 1965). At time i, the agent is in state $s_i \in S$, observes $o_i \in \Omega \sim O(\cdot|s_i)$, takes action $a_i \in A$, receives reward $R(s_i, a_i)$ and would transit to $s_{i+1} \sim T(\cdot|s_i, a_i)$. The agent's policy based on the observation history $h_t = \{(o_i, a_i, r_i)\}_{i=1}^t$ is denoted as $\pi(\cdot|h_{t-1}, o_t)$. We say an algorithm $\mathcal{A}$ that can *solve* POMDP as being able to find the optimal policy $\pi^\star$ for given POMDP $\mathcal{M}$ where:

$$\pi^\star = \operatorname*{argmin}_{\pi} \mathop{\mathbb{E}}_{\substack{a_t\sim\pi(\cdot|h_{t-1},o_t)\\ s_t\sim T(\cdot|s_{t-1},a_t)\\ o_t\sim O(\cdot|s_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right].$$

DFA & regular language. Deterministic Finite Automata (DFA) can be defined as $A = (S, \Sigma, T, s_0, F)$, where S is a finite set of states, Σ is a finite set of symbols, $T : S \times \Sigma \to S$ is the transition function, s_0 is the start state, and $F \subseteq Q$ is the set of accepting states. A string w whose i-th symbol is w_i is *accepted* by A if $\exists(s_0, s_1, \ldots, s_n)$, s.t. $s_i \in S$ for $1 \le i \le n$, $s_i = T(s_{i-1}, w_i)$ and $s_n \in F$. L is a regular language if it is *recognized* by DFA A, that is, $L = \{w : A \text{ accepts } w\}$.

4. Limitations of Transformer in Partially Observable RL

RL algorithms typically take as inputs the current state with the assumption of the Markov property. In partially observable environments that lack the Markov property, the hidden state extracted by `SEQ` from the observation history is thus anticipated to contain the information of the *real state* to benefit subsequent `RL`.

Notably, Delétang et al. (2023) points out that Transformers (referred to as `TF`s) fail to recognize regular languages. Inspired by the significant correspondence between regular languages and POMDPs (details in Definition 4.2), we naturally conjecture that `TF` is not capable of retrieving the information of *real state* from partial observations accurately, which would lead to a decline in the performance of the pipeline (`SEQ`, `RL`).

Building upon this view, this section first shows that solving a POMDP problem is harder than solving a regular language problem. Afterward, we will introduce two theoretical results to elucidate the limitations of Transformers, supported by simple examples for illustrations. Consequently, it is inferred that (`TF`, `RL`) cannot address POMDPs generally.

4.1. Reduction from Regular Language to POMDP

Proposition 4.1. *If an algorithm $\mathcal{A} = (\texttt{SEQ}, \texttt{RL})$ can solve POMDPs, then given a regular language L, $\mathcal{A}$ can recognize L by solving a POMDP problem $\mathcal{M}$.*

Proof idea. We construct a POMDP $\mathcal{M}$, such that each state represents a transition in L, and the observation at timestep t is the corresponding character w_t. The agent could output *accept* or *reject*, and the reward is assigned if and only if the final output aligns with the acceptance of the string $w = w_0 \ldots w_{t-1}$ in L. In this way, the optimal policy π on $\mathcal{M}$ is to *accept* all $w \in L$ and *reject* all $w \notin L$, so if an algorithm $\mathcal{A}$ can solve POMDP $\mathcal{M}$, then $\mathcal{A}$ can recognize L. Proof details are deferred to Appendix B.1.

Definition 4.2 (POMDP derived from regular language L)**.** Given a regular language L, the POMDP derived as Proposition 4.1 is denoted as $\mathcal{M}^L$. For an integer n, $\mathcal{M}^L(n)$ represents a special case of $\mathcal{M}^L$ whose horizon is no longer than n.

Remark 4.3. When implementing RL algorithms, especially

in online settings, it is common to set a truncated time n for training and evaluation purposes. For $\mathcal{M}^L(n)$ with maximum horizon n, since observation during training and evaluation come from the same distribution, we can analogize it to *fitting* in supervised learning. Furthermore, in partially observable cases, historical information needs to be considered. If there is no time limit set, there is a need for *length extrapolation*, which we can analogize to *generalization* in supervised learning, as is captured by $\mathcal{M}^L$.

In Figure 2, we illustrate how to construct $\mathcal{M}^L$. We also provide experiments to verify the reduction (cf. Section 6.1).

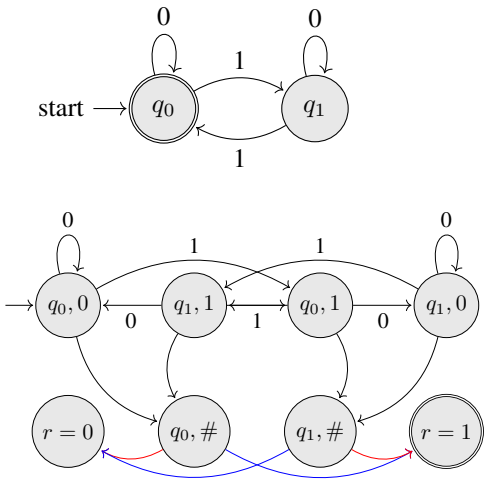


Figure 2. Above: Illustration for DFA of PARITY. There are two states q_0 and q_1, where q_0 is both the initial state and the accepting state. The transitions are plotted in gray arrows. Below: Illustration for $\mathcal{M}^{\text{PARITY}}$. The states are (q_i, w) where $i \in \{0, 1\}$ and $w \in \{0, 1, \#\}$, and the agent could observe w. The initial state are randomly sampled from (q_0, w). The stochastic transitions are plotted in gray arrows. At final state $(q_i, \#)$, blue arrows stand for choosing *accept*, and red arrows stand for choosing *reject*.

4.2. Limitations in Fitting: Solving $\mathcal{M}^L(n)$

While prior research has claimed universality for Transformers, specifically proving their Turing completeness and ability to approximate any seq-to-seq function on compact support (Bhattamishra et al., 2020; Pérez et al., 2021; Luo et al., 2022; Yun et al., 2019), it is crucial to note certain impractical assumptions underlying these assertions as they often rely on assumptions of infinite precision and finite length (Jiang et al., 2023).

In this subsection, we assume that $(\texttt{TF}, \texttt{RL})$ ($\texttt{TF}$ denotes a Transformer) is a **log-precision model** that all values in the model have $O(\log n)$ precision, where n is the input length. This assumption aligns with reality since computer floating-point precision is typically 16, 32, or 64 bits, smaller than the sequence lengths commonly handled by sequence models. Based on this assumption, there exists a class of POMDPs, for which achieving solutions with $(\texttt{TF}, \texttt{RL})$ would demand an excessively large quantity of parameters. This type of problem can be directly mapped to a type of circuit complexity, with its definition provided in the appendix A.1.

Theorem 4.4. *Assume $\mathsf{TC}^0 \neq \mathsf{NC}^1$. Given an NC^1 complete regular language L, for any depth D and a any polynomial $\text{poly}(n)$, there exists a length n such that no log-precision $(\texttt{TF}, \texttt{RL})$ with depth D and hidden dimension $d \leq \text{poly}(n)$ can solve $\mathcal{M}^L(n)$.*

Proof idea. At the heart of the proof is a contradiction achieved through circuit complexity theory (Arora & Barak, 2009). Merrill & Sabharwal (2023) has shown that TC^0 circuits can simulate a log-precision Transformer with constant depth and polynomial hidden dimensions. Consequently, if $(\texttt{TF}, \texttt{RL})$ can solve NC^1 complete problems, it would cause both TC^0 and NC^1 complexities to collapse, a scenario generally deemed impossible (Yao, 1989). Proof details of Theorem 4.4 are deferred to Appendix B.2.

Following syntactic monoid theory (Straubing, 2012) and Barrington's theorem (Barrington, 1986), a significant number of regular languages are NC^1 complete, such as the regular language $((0+1)^3(01^*0+1))^*$ (cf. Appendix A.2.3).

On the other hand, these two works inform us of another fact: for a regular language L, there are only two possibilities—either $L \in \mathsf{NC}^1$ complete or $L \in \mathsf{TC}^0$ (more specifically, $L \in \mathsf{AC}^0$). As of now, the question of whether problems solvable by log-precision Transformers belong to TC^0 remains an open problem (Merrill & Sabharwal, 2023). However, numerous experimental results (Delétang et al., 2023; Huang et al., 2022) suggest that Transformers do not perform well in handling certain regular languages within TC^0, such as PARITY. In the next section, we demonstrate, from a generalization perspective, that Transformers cannot solve $\mathcal{M}^L$ for a broader range of regular languages L.

4.3. Limitations in Generalization: Solving $\mathcal{M}^L$

When deploying the Partially Observable RL algorithm, we anticipate it to demonstrate the ability of length generalization. In this subsection, we examine scenarios corresponding to $\mathcal{M}^L$ and no longer assume that the Transformer model operates with logarithmic precision.

Recent works (Press et al., 2021; Delétang et al., 2023; Ruoss et al., 2023) have empirically demonstrated that length extrapolation is a weakness of Transformers. Lemma 4.5 theoretically indicates that for any Transformer with the dot-product softmax attention mechanism, robust generalization is not achievable as the input length increases.

Lemma 4.5 (Lemma 5 in Hahn (2020a))**.** *Given a Transformer with softmax attention, let n be the input length. If we change one input u_i $(i < n)$ to u_i', then the change in the resulting hidden x_n at the output layer is bounded by*

$O(D/n), D = \|u_i - u_i'\|$ with constants depending on the parameter matrices.

Theorem 4.6. *Given an regular language L, let $c(n, a) = \#\{xa \in L : |x| = n\}$. If there exists $a \in \Sigma$ such that $\{n : 0 < c(n, a) < |\Sigma|^n\}$ are infinite, and $\texttt{RL}$ is a Lipschitz function, then $(\texttt{TF}, \texttt{RL})$ cannot solve $\mathcal{M}^L$.*

Proof idea. Since Σ is a finite set, D is deterministic. Then we will prove for L satisfying the conditions, there exists infinite u, u' such that u and u' differ by only 1 positions but $u \in L, u' \notin L$. According to Lemma 4.5, the hidden states x and x' output by the Transformer differ by $O(1/n)$. Since $\texttt{RL}$ is a Lipschitz function, the results of $\texttt{RL}(x)$ and $\texttt{RL}(x')$ also differ by $O(1/n)$. As n increases, information from non-current time steps will only have a negligible impact on the output of $\texttt{RL}$. Proof details of Theorem 4.6 are deferred to Appendix B.3.

Observing that PARITY satisfies the conditions outlined in Theorem 4.6, we can derive Corrollary 4.7.

Corollary 4.7. *If $L = \texttt{PARITY}$ and $\texttt{RL}$ is a Lipschitz function, then $(\texttt{TF}, \texttt{RL})$ can not solve $\mathcal{M}^L$.*

The Lipschitz property is commonly observed in widely used learning-based RL algorithms, such as employing MLPs to predict Q-values, V-values, or the probability distribution of the next action. For cases that do not satisfy the Lipschitz property, such as those relying on the maximum value rather than logits, Chiang & Cholak (2022) provides a constructive method for a Transformer that can recognize PARITY. This scenario corresponds to the greedy policy based on Q-values. However, this theorem indicates that Transformers do not model sequences in a way that accurately reconstructs the *real states*, which makes it hard for $(\texttt{TF}, \texttt{RL})$ to perform length extrapolation.

4.4. From POMDPs to Regular Languages

Through illustrating the limitations of $(\texttt{TF}, \texttt{RL})$ in handling POMDPs derived from regular languages, we demonstrate that there exist POMDP problems for which Transformers cannot effectively learn the corresponding inductive biases.

This class of POMDP problems corresponding to regular languages can be divided into three levels based on circuit complexity: $< \mathsf{TC}^0, [\mathsf{TC}^0, \mathsf{NC}^1), \mathsf{NC}^1$. The difficulty for Transformers to handle these problems increases progressively. This difficulty classification can be extended to existing POMDP problems. Please refer to Appendix C for detailed discussion.

- $< \mathsf{TC}^0$: Most tasks that solely assess pure memory capabilities are weaker than TC^0. These tasks only involve extracting a finite number of tokens from the past and performing simple logical operations with current observation information. Most memory tasks mentioned in Ni et al. (2023) fall into this category. $(\texttt{TF}, \texttt{RL})$ excel at solving such problems.
- $[\mathsf{TC}^0, \mathsf{NC}^1)$: This category already represents the vast majority of regular languages. The corresponding typical POMDPs are environments such as Passive Visual Match (Hung et al., 2019) or Memory Maze (Pasukonis et al., 2022), where there is a need to infer the current position based on historical information. This is typically manifested in the requirement to reconstruct a relatively simple state from complex historical data.
- NC^1: Currently, no existing discrete-state POMDP problem has been found to correspond to this class of regular languages. According to Theorem 4.4, it is difficult for $(\texttt{TF}, \texttt{RL})$ to learn the optimal policy.

Establishing a direct connection with regular languages is not particularly straightforward in continuous scenarios. However, some standard POMDP scenarios, such as Pybullet Occlusion Task (Ni et al., 2022), are at least not in the first level. These tasks require inferring the current actual state based on contextual information.

Furthermore, the preceding discussion implies that for $(\texttt{TF}, \texttt{RL})$, the hidden state fed to $\texttt{RL}$ is often not the underlying *real state*. In contrast, in subsequent experiments (cf. Section 6), we observe that $(\texttt{RNN}, \texttt{RL})$ behaves differently and can implicitly reconstruct the *real state*. The capability of recovering underlying *real states* with the Markov property is believed to be a prerequisite for solving Partially Observable RL. Therefore, for POMDPs in general cases, $(\texttt{TF}, \texttt{RL})$ may encounter issues.

5. Combining Transformer and RNN

From the analysis in Section 4, it becomes evident that RNN-like models (LSTM, GRU, RNN) emerge as promising sequence model choices for Partially Observable RL. There has been considerable theoretical work demonstrating their completeness on regular languages (Merrill, 2019; Korsky & Berwick, 2019). For cases of log precision, based on definitions, we can directly map the recurrent units of RNNs to transition functions in DFAs. Therefore, RNNs do not suffer from the theoretical constraints encountered by Transformers.

However, RNN-like models face the challenge of rapid memory decay (Ni et al., 2023; Parisotto et al., 2019), leading to an inferior performance on POMDP problems that demand long-term memory when compared to Transformers (Parisotto et al., 2019; Ni et al., 2023).

Another insight from the previous section is that the attention mechanism of Transformers is primarily to blame for

Table 1. **The recurrent representation for Transformer variants with pointwise recurrence.** We compare different Transformer variants: FART, FWP, RWKV and RetNet. y_i, u_i are as defined in Section 3, s_i, z_i are hidden states, and the other variables are parameters.

Architectures	Recurrent Representation for a Single Head
FART (Katharopoulos et al., 2020)	$y_i = \mathrm{FFN}\left(\frac{\phi(u_iW_Q)^\top}{\phi(u_iW_Q)^\top z_i} + u_i\right), \begin{array}{l} s_i = s_{i-1} + \phi(u_iW_k)(u_iW_V)^\top \\ z_i = z_{i-1} + \phi(u_iW_k) \end{array}$
FWP (Schlag et al., 2021)	$y_i = \frac{1}{z_i\phi(W_qu_i)} W_i\phi(W_qu_i), \begin{array}{l} W_i = W_{i-1} + (W_vu_i) \otimes \phi(W_ku_i) \\ z_i = z_{i-1} + \phi(W_ku_i) \end{array}$
RWKV (Peng et al., 2023)	$y_i = \mathrm{Gate}\left(\frac{s_{i-1}+\mathrm{e}^{v+W_ku_i}\odot u_t}{z_{i-1}+\mathrm{e}^{v+W_ku_i}}\right), \begin{array}{l} s_i = \mathrm{e}^{-w} \odot s_{i-1} + \mathrm{e}^{W_ku_i} \odot u_i \\ z_i = \mathrm{e}^{-w} \odot z_{i-1} + \mathrm{e}^{W_ku_i} \end{array}$
RetNet (Sun et al., 2023)	$y_i = (\tau(XW_G) \odot \mathrm{GN}(z_i)), z_i = \gamma z_{i-1} + (Ku_i)^\top(Vu_i)$

their limitations (see Figure 1b). As articulated in Merrill & Sabharwal (2023), there exists a trade-off between the highly parallel structure of Transformers and their computational capacity.

To alleviate these limitations of Transformers, a natural idea is to endow Transformers with the ability of pointwise recurrence (see Figure 1a). This line of development has been the focus of numerous efforts, resulting in several Transformer variants that incorporate this mechanism, as detailed in Table 1. The shared feature of these methods in their recurrence representation for a single head can be found in the simple linear operations they employ, such as $x_t = \lambda x_{t-1} + u_t$. While the non-linear components can be amortized across the layers through the FFN between layers. If the number of heads is set equal to the dimension of the hidden state h, then

$$\mathbf{x}_t = \mathbf{\Lambda}\mathbf{x}_{t-1} + \mathbf{u}_t \ , \tag{1}$$

where $\mathbf{\Lambda} = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_h)$. In RetNet, operations involving x_t, λ_i, and u_i are performed over $\mathbb{C}$, while the remaining operations are carried out over $\mathbb{R}$. As for RWKV, λ_i is a learnable parameter, while in the rest variants are hyperparameters.

From the perspective of RNN, if we linearize and diagonalize the RNN's recurrent unit $\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t$, we obtain the following form:

$$\tilde{\mathbf{x}}_t = \mathbf{\Lambda}\tilde{\mathbf{x}}_{t-1} + \tilde{\mathbf{u}}_t \ , \tag{2}$$

where $\mathbf{A} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1}, \tilde{\mathbf{x}}_t = \mathbf{P}^{-1}\mathbf{x}_t, \tilde{\mathbf{u}}_t = \mathbf{P}^{-1}\mathbf{B}\mathbf{u}_t$. Since almost all matrices can be diagonalized over $\mathbb{C}$, the operations mentioned above are defined in $\mathbb{C}$ (Horn & Johnson, 2012).

Comparing (1) and (2), the pointwise-recurrence Transformer can be viewed as a linear RNN with certain constraints, and the linear RNN serves as a balance point between Transformers and RNNs. To summarize, we expect linear RNN to be more suitable as a sequence model in Partially Observable RL for the following reasons.

Regular language. Many studies suggest that the recurrence with non-linear activation functions plays a crucial role in the completeness of RNNs in regular languages (Chung & Siegelmann, 2021), while linear RNNs may lose this completeness. However, some researches indicate that linear RNNs can effectively approximate RNNs (Huang et al., 2022; Lim et al., 2023) and perform well on formal language tasks similar in form to NLP (Huang et al., 2022; Irie et al., 2023). Compared to Transformers, their inductive biases are closer to HMMs. In subsequent experiments (cf. Section 6), we validate that $(\texttt{LRNN}, \texttt{RL})$ can implicitly learn the states in POMDPs.

State space model. Linear RNNs have been proven to efficiently fit partially observable *linear dynamic systems* (Wang et al., 2022). While the transformer's fitting capability has theoretical proofs only under certain specific conditions (Balim et al., 2023; Li et al., 2023), with no similar conclusion for more general situations. The linear dynamic system can be considered as a first-order approximation of a state space model, indicating the potential of linear RNNs in addressing a broader range of POMDPs.

Long term memory. The primary reason for the long-term dependency issues in RNNs is the challenge of gradient explosion or vanishing when input length increases during training (Pascanu et al., 2013). Transformers, due to their parallel structure, are less susceptible to this issue. To mitigate this problem, gate mechanisms are introduced to RNNs (Dey & Salem, 2017; Hochreiter & Schmidhuber, 1997). However, Kanai et al. (2017) indicates that the non-linear recurrence is the primary cause of gradient explosions. For linear RNNs, effectively managing the range of parameters λ_i between $[0, 1]$ during initialization successfully addresses both gradient explosion and vanishing issues. This has been validated in certain supervised learning tasks with long-term dependencies (Orvieto et al., 2023; Gu et al.,

2021).

6. Experiments

In this section, we compare the effectiveness of three different sequence models — Transformer, RNN, and linear RNN — in addressing partially observable decision-making problems within the realm of Partially Observable RL (`SEQ`, `RL`). We choose GPT (Radford et al., 2019), LSTM (Hochreiter & Schmidhuber, 1997), and LRU (Orvieto et al., 2023) as the representative architectures for these three types of models. To substantiate our hypotheses, we conduct experiments in three distinct POMDP scenarios, detailed in Sections 6.1 to 6.3. These experiments are designed to assess the models from various perspectives: 1) POMDPs derived from certain regular languages, including `EVEN PAIRS`, `PARITY`, and `SYM(5)`; 2) tasks from Pybullet Partially Observable environments (Ni et al., 2022) that require the ability of state space modeling; 3) tasks that require pure long-term memory capabilities, such as Passive T-Maze and Passive Visual Match (Hung et al., 2018). Comprehensive implementation details, task descriptions, and supplementary results are presented in Appendix D. We also conduct a comparison with some published Transformer in RL there.

6.1. POMDPs Derived from Regular Languages

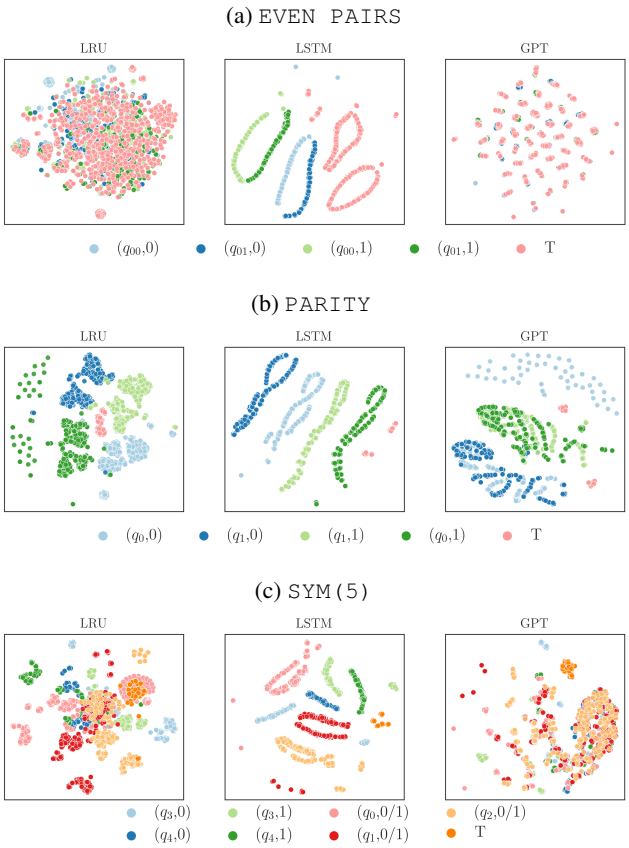
We construct this type of POMDP problem following the approach of Proposition 4.1, and use DQN (Van Hasselt et al., 2016) as the RL component in (`SEQ`, `RL`). Three regular language tasks correspond to the difficulty classification in Section 4.4. The learning curves are shown in Appendix D.5, Figure 16. We provide the experimental results on length extrapolation and model scale for this task in the appendix, where Theorem 4.4 and Theorem 4.6 are validated.

To look into how they model the regular languages, we visualize the hidden states in Figure 3. Generally, all three sequence models can fit scenarios with short lengths. However, as the input length increases, LSTM exhibits the best fitting capability, followed by LRU, and GPT performs the least effectively. We observe that in POMDP tasks derived from the three regular languages, the distinct nature of these languages yields varied results:

`EVEN PAIRS` is a specific regular language that could be directly solved by memorizing the first character and comparing it with the last character, which aligns with the inductive bias of the attention mechanism. As a result, GPT solves $\mathcal{M}^{\texttt{EVEN PAIRS}}$ reasonably well.

`PARITY` is a regular language with simple DFA in TC^0. As shown in Figure 3b, LSTM and LRU are capable of accurately modeling $\mathcal{M}^{\texttt{PARITY}}$. Through colors, it can be observed that the hidden state of the transformer is almost

Figure 3. **Hidden state for regular language tasks**. We visualize the hidden states of each sequence model during evaluation at length 25 using t-SNE (van der Maaten & Hinton, 2008) and annotate them according to their real states. Our classification corresponds to the state the observation history maps to in the reduced POMDP, namely (q, w), while 'T' stands for the terminal state. The states with similar colors in the diagram generally produce the same type of observation.



solely distinguished based on the current observation. It relies on processing the entire history through attention after encountering a terminal symbol. This is more like memorizing all the different strings, resulting in lower final returns.

`SYM(5)` is a NC^1 complete regular language as mentioned in Section 4.2, and we have shown the inability of GPT to solve $\mathcal{M}^{\texttt{SYM(5)}}(n)$ in Theorem 4.4. Experimental results align with our claim, proving that GPT performs worst in this task and fails to recover the true state.

6.2. PyBullet Partially Observable Environments

We conduct experiments on 8 partially observable environments, which are all PyBullet locomotion control tasks with parts of the observations occluded (Ni et al., 2022), and denote them as PyBullet Occlusion. These experiments

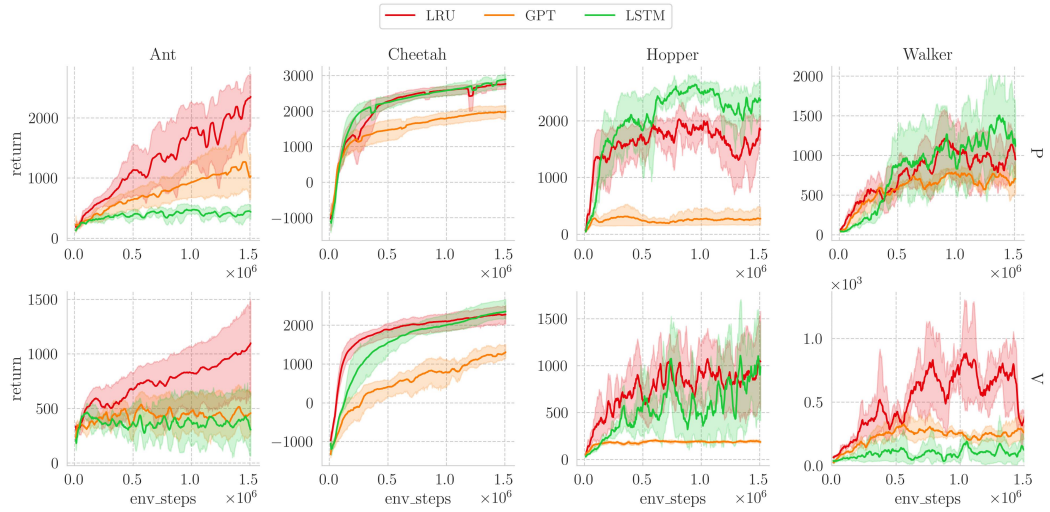


Figure 4. **Learning curves for PyBullet occlusion tasks**. Mean of 5 seeds. The shaded area indicates 95% confidence intervals.

Table 2. **Normalized scores for PyBullet occlusion tasks**. We compare different sequence models LRU, GPT and LSTM. 'V' refers to 'only velocities observable', and 'P' refers to 'only positions observable'. We present normalized scores defined in Appendix D.5, Equation 3. Blue highlight indicates the highest score, and orange highlight indicates the second-highest score.

Task	Type	LRU	GPT	LSTM
Ant	V	29.8 ± 20.4	9.4 ± 7.1	7.4 ± 8.8
	P	81.2 ± 28.7	38.2 ± 26.0	5.7 ± 3.3
Cheetah	V	96.8 ± 8.1	69.3 ± 6.9	98.1 ± 8.3
	P	109.9 ± 4.2	88.8 ± 6.3	112.5 ± 5.4
Hopper	V	94.1 ± 23.4	13.5 ± 0.5	82.5 ± 37.1
	P	147.9 ± 12.3	23.8 ± 18.0	184.1 ± 13.4
Walker	V	61.7 ± 14.6	22.3 ± 6.0	12.2 ± 7.0
	P	79.3 ± 23.1	49.5 ± 4.8	94.6 ± 36.6
Average		88.2	39.3	74.6

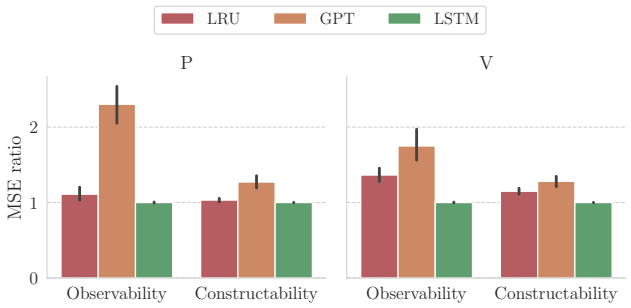


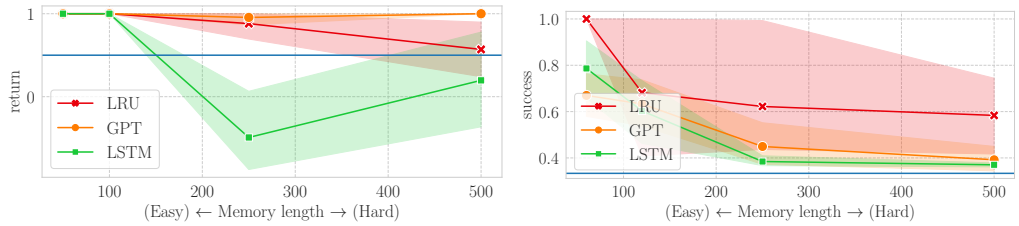
Figure 5. **Mean Squared Error (MSE) ratios for state space modeling tasks**. 'V' refers to 'only velocities observable', and 'P' refers to 'only positions observable'. To enable a comparative analysis of the performance among the three models, we present the MSE ratio, as defined in Appendix D.5, Equation 4.

encompass four distinct tasks: Ant, Cheetah, Hopper, and Walker, and we evaluate the models based on two types of observations: Velocities Only (V) and Positions Only (P). The normalized scores are demonstrated in Table 2, and we also provide learning curves in Figure 4. From the results, it is evident that LRU and LSTM outperforms GPT in all eight tasks, matching our claim that the Transformer architecture struggles at modeling partially observable sequences. The results showing that LSTM outperforms GPT are also verified in Ni et al. (2023).

Moreover, the general performances of LRU and LSTM are notably comparable, and LRU significantly outperforms LSTM in certain tasks, namely Ant (P, V), and Walker (V). Such results demonstrate that after linearization, recurrent-based models can still effectively retain their capacity to model the sequence, and can serve as a well-rounded balance integrating the strengths of both Transformer and RNN architectures.

We conduct ablation experiments with full observability in Appendix D.5, Table 7, and the overall performances of the three models are close, affirming that GPT's inferior performance in POMDP scenarios stems from partial observability rather than other factors.

To enhance our understanding of the capability to extract state information from observation sequences, we meticulously crafted two tasks. These tasks are aimed at determining the initial state, termed "Observability", and forecasting the current state, referred to as "Constructability", using historical observation sequences, and we adopt Mean Square Error (MSE) as our training target. Our experiments were conducted on the D4RL medium-expert dataset (Fu et al.,



(a) Passive T-Maze (b) Passive Visual Match

Figure 6. **Results in pure long-term memory environments with varying memory lengths**. Blue line indicates the return of optimal markov policy, which only has access to the observation.

2020) of the aforementioned tasks, and the results (illustrated in Figure 5) are presented as the average MSE ratios across these tasks. The findings reveal that, in both scenarios, GPT is notably less competent compared to the other two models. In contrast, the LRU model demonstrates capability on par with the LSTM model. This observation lends further support to our hypothesis that GPT's ability to reconstruct states from partially observable sequences is worse than that of the recurrent-based models.

6.3. Pure Long-term Memory Environments

Results for pure long-term memory environments, namely Passive T-Maze and Passive Visual Match, are provided in Figure 6, and learning curves are shown in Appendix D.5, Figure 17. In these experiments, we follow the work of Ni et al. (2023), which tests the long-term memory ability of Transformer-based agent and LSTM-based agent on two memory-demanding tasks. We observe that LRU performs comparably to GPT, while significantly outperforming LSTM. Furthermore, LRU beats GPT on Passive Visual Match, the harder task of the two which involves a complex reward function (Hung et al., 2018), showcasing its powerful long-term memory capability.

7. Conclusion

In this work, we challenge the suitability of Transformers as sequence models in Partially Observable RL. Through theoretical analysis and empirical evidence, we reveal Transformer's limitations in solving POMDPs, particularly their struggle with modeling regular languages, a key aspect of POMDPs. As a remedy to these issues, We propose LRU as a more effective alternative, combining the strengths of recurrence and attention. Supported by extensive experiments, our findings challenge the prevailing use of Transformers in sequential decision-making tasks, and open new avenues for exploring recurrent structures in complex, partially observable environments.

It is also important to acknowledge the limitations of our work. After introducing recurrence, LRU serves as a choice to combine the advantages of Transformer and RNN, while still lacking theoretical guarantees for modeling regular languages. Although LRU demonstrates satisfactory performance in experiments, there remains a need for further exploration in this direction. Additionally, the theoretical analysis in this paper focuses more on the exploitation aspect of RL, while lacking discussion on exploration. Complex POMDP tasks not only require suitable sequence models but also need to be paired with appropriate RL algorithms.

Impact Statement

Our work revisits the application of Transformers in RL, aiming to advance the development of decision intelligence. If misused in downstream tasks, it has the potential to lead to adverse effects such as privacy breaches and societal harm. Nevertheless, this is not directly related to our research, as our primary focus is on theoretical investigations.

References

Arora, S. and Barak, B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Balim, H., Du, Z., Oymak, S., and Ozay, N. Can transformers learn optimal filtering for unknown systems? *arXiv preprint arXiv:2308.08536*, 2023.

Barrington, D. A. Bounded-width polynomial-size branching programs recognize exactly those languages in nc. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 1–5, 1986.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613%2Fjair.3912.

Bhattamishra, S., Patel, A., and Goyal, N. On the computational power of transformers and its implications in sequence modeling. *arXiv preprint arXiv:2006.09286*, 2020.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Carrasco, R. C. and Oncina, J. Learning stochastic regular grammars by means of a state merging method. In *International Colloquium on Grammatical Inference*, pp. 139–152. Springer, 1994.

Chen, C., Wu, Y., Yoon, J., and Ahn, S. Transdreamer: Reinforcement learning with transformer world models. *CoRR*, abs/2202.09481, 2022. URL https://arxiv.org/abs/2202.09481.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *CoRR*, abs/2106.01345, 2021. URL https://arxiv.org/abs/2106.01345.

Chiang, D. and Cholak, P. Overcoming a theoretical limitation of self-attention. *arXiv preprint arXiv:2202.12172*, 2022.

Chung, S. and Siegelmann, H. Turing completeness of bounded-precision recurrent neural networks. *Advances in Neural Information Processing Systems*, 34:28431–28441, 2021.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Delétang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., and Ortega, P. A. Neural networks and the chomsky hierarchy. In *11th International Conference on Learning Representations*, 2023.

Deng, F., Park, J., and Ahn, S. Facing off world model backbones: Rnns, transformers, and s4. *arXiv preprint arXiv:2307.02064*, 2023.

Dey, R. and Salem, F. M. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597–1600. IEEE, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Dulac-Arnold, G., Mankowitz, D. J., and Hester, T. Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901, 2019. URL http://arxiv.org/abs/1904.12901.

Elman, J. L. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 0364-0213. doi: https://doi.org/10.1016/0364-0213(90)90002-E. URL https://www.sciencedirect.com/science/article/pii/036402139090002E.

Feng, G., Gu, Y., Zhang, B., Ye, H., He, D., and Wang, L. Towards revealing the mystery behind chain of thought: a theoretical perspective. *arXiv preprint arXiv:2305.15408*, 2023.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, December 2020a. ISSN 2307-387X. doi: 10.1162/tacl_a_00306. URL http://dx.doi.org/10.1162/tacl_a_00306.

Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020b.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Horn, R. A. and Johnson, C. R. *Matrix analysis*. Cambridge university press, 2012.

Hu, K., Zheng, R. C., Gao, Y., and Xu, H. Decision transformer under random frame dropping. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=NmZXv4467ai.

Huang, F., Lu, K., Yuxi, C., Qin, Z., Fang, Y., Tian, G., and Li, G. Encoding recurrence into transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

Hung, C., Lillicrap, T. P., Abramson, J., Wu, Y., Mirza, M., Carnevale, F., Ahuja, A., and Wayne, G. Optimizing agent behavior over long time scales by transporting value. *CoRR*, abs/1810.06721, 2018. URL http://arxiv.org/abs/1810.06721.

Hung, C.-C., Lillicrap, T., Abramson, J., Wu, Y., Mirza, M., Carnevale, F., Ahuja, A., and Wayne, G. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):5223, 2019.

Hutchins, D., Schlag, I., Wu, Y., Dyer, E., and Neyshabur, B. Block-recurrent transformers. *Advances in Neural Information Processing Systems*, 35:33248–33261, 2022.

Irie, K., Schlag, I., Csordás, R., and Schmidhuber, J. Going beyond linear transformers with recurrent fast weight programmers. *Advances in neural information processing systems*, 34:7703–7717, 2021.

Irie, K., Csordás, R., and Schmidhuber, J. Practical computational power of linear transformers and their recurrent and self-referential extensions. *arXiv preprint arXiv:2310.16076*, 2023.

Jiang, H., Li, Q., Li, Z., and Wang, S. A brief survey on the approximation theory for sequence modelling. *arXiv preprint arXiv:2302.13752*, 2023.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Kanai, S., Fujiwara, Y., and Iwamura, S. Preventing gradient explosions in gated recurrent units. *Advances in neural information processing systems*, 30, 2017.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.

Korsky, S. A. and Berwick, R. C. On the computational power of rnns. *arXiv preprint arXiv:1906.06349*, 2019.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35: 27921–27936, 2022.

Li, X., Li, L., Gao, J., He, X., Chen, J., Deng, L., and He, J. Recurrent reinforcement learning: A hybrid approach. *CoRR*, abs/1509.03044, 2015. URL http://arxiv.org/abs/1509.03044.

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pp. 19565–19594. PMLR, 2023.

Lim, Y. H., Zhu, Q., Selfridge, J., and Kasim, M. F. Parallelizing non-linear sequential models over the sequence length. *arXiv preprint arXiv:2309.12252*, 2023.

Littman, M. and Sutton, R. S. Predictive representations of state. In Dietterich, T., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/1e4d36177d71bbb3558e43af9577d70e-Paper.pdf.

Lu, C., Schroecker, Y., Gu, A., Parisotto, E., Foerster, J., Singh, S., and Behbahani, F. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Luo, S., Li, S., Zheng, S., Liu, T.-Y., Wang, L., and He, D. Your transformer may not be as powerful as you expect. *Advances in Neural Information Processing Systems*, 35: 4301–4315, 2022.

Merrill, W. Sequential neural networks as automata. *arXiv preprint arXiv:1906.01615*, 2019.

Merrill, W. and Sabharwal, A. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.

Merrill, W., Sabharwal, A., and Smith, N. A. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

Micheli, V., Alonso, E., and Fleuret, F. Transformers are sample efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

Morad, S., Kortvelesy, R., Bettini, M., Liwicki, S., and Prorok, A. Popgym: Benchmarking partially observable reinforcement learning. *arXiv preprint arXiv:2303.01859*, 2023.

Ni, T., Eysenbach, B., and Salakhutdinov, R. Recurrent model-free RL can be a strong baseline for many pomdps. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16691–16723. PMLR, 2022. URL https://proceedings.mlr.press/v162/ni22a.html.

Ni, T., Ma, M., Eysenbach, B., and Bacon, P.-L. When do transformers shine in rl? decoupling memory from credit assignment. *arXiv preprint arXiv:2307.03864*, 2023.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Parisotto, E., Song, H. F., Rae, J. W., Pascanu, R., Gülçehre, Ç., Jayakumar, S. M., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M. M., Heess, N., and Hadsell, R. Stabilizing transformers for reinforcement learning. *CoRR*, abs/1910.06764, 2019. URL http://arxiv.org/abs/1910.06764.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Pasukonis, J., Lillicrap, T., and Hafner, D. Evaluating long-term memory in 3d mazes. *arXiv preprint arXiv:2210.13383*, 2022.

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K. K., He, X., Hou, H., Lin, J., Kazienko, P., Kocon, J., Kong, J., Koptyra, B., Lau, H., Mantri, K. S. I., Mom, F., Saito, A., Song, G., Tang, X., Wang, B., Wind, J. S., Wozniak, S., Zhang, R., Zhang, Z., Zhao, Q., Zhou, P., Zhou, Q., Zhu, J., and Zhu, R.-J. Rwkv: Reinventing rnns for the transformer era, 2023.

Pérez, J., Barceló, P., and Marinkovic, J. Attention is turing complete. *The Journal of Machine Learning Research*, 22(1):3463–3497, 2021.

Pin, J.-E. Syntactic semigroups. In *Handbook of Formal Languages: Volume 1 Word, Language, Grammar*, pp. 679–746. Springer, 2013.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners, 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Reid, M., Yamada, Y., and Gu, S. S. Can wikipedia help offline reinforcement learning? *CoRR*, abs/2201.12122, 2022. URL https://arxiv.org/abs/2201.12122.

Robine, J., Höftmann, M., Uelwer, T., and Harmeling, S. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023.

Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Ruoss, A., Delétang, G., Genewein, T., Grau-Moya, J., Csordás, R., Bennani, M., Legg, S., and Veness, J. Randomized positional encodings boost length generalization of transformers. *arXiv preprint arXiv:2305.16843*, 2023.

Samsami, M. R., Zholus, A., Rajendran, J., and Chandar, S. Mastering memory tasks with world models. *arXiv preprint arXiv:2403.04253*, 2024.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366. PMLR, 2021.

Shi, R., Liu, Y., Ze, Y., Du, S. S., and Xu, H. Unleashing the power of pre-trained language models for offline reinforcement learning. *CoRR*, abs/2310.20587, 2023. doi: 10.48550/ARXIV.2310.20587. URL https://doi.org/10.48550/arXiv.2310.20587.

Straubing, H. *Finite automata, formal logic, and circuit complexity*. Springer Science & Business Media, 2012.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *CoRR*, abs/2307.08621, 2023. doi: 10.48550/ARXIV.2307.08621. URL https://doi.org/10.48550/arXiv.2307.08621.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

van der Maaten, L. and Hinton, G. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Wang, L., Shen, B., Hu, B., and Cao, X. Can gradient descent provably learn linear dynamic systems? *arXiv preprint arXiv:2211.10582*, 2022.

Wu, Y., Wang, X., and Hamaya, M. Elastic decision transformer. *CoRR*, abs/2307.02484, 2023. doi: 10.48550/ARXIV.2307.02484. URL https://doi.org/10.48550/arXiv.2307.02484.

Yamagata, T., Khalil, A., and Santos-Rodríguez, R. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline RL. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38989–39007. PMLR, 2023. URL https://proceedings.mlr.press/v202/yamagata23a.html.

Yao, A. C. Circuits and local computation. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pp. 186–196, 1989.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27042–27059. PMLR, 2022. URL https://proceedings.mlr.press/v162/zheng22c.html.

Åström, K. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965. ISSN 0022-247X. doi: https://doi.org/10.1016/0022-247X(65)90154-X.

A. Additional Background and Notation

A.1. Circuit complexity

In this subsection, We introduce several basic complexity classes, namely AC^0, TC^0, and NC^1:

- AC^0 contains all languages that are decided by Boolean circuits with constant depth, unbounded fan-in, and polynomial size, consisting of AND gates, OR gates, NOT gates;
- TC^0 is AC^0 with majority gates which outputs true if and only if more than half of the input bits are true;
- NC^1 contains all languages that are decided by Boolean circuits with a logarithmic depth of $\mathcal{O}(\log n)$ where n is the input length, constant fan-in, and polynomial-size, consisting of AND gates, OR gates, and NOT gates.

The relationships between them are $\mathsf{AC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1$, and it is commonly conjectured that $\mathsf{TC}^0 \neq \mathsf{NC}^1$ whereas it remains an open problem in the computation complexity theory. A language $L \in \mathsf{NC}^1$ is NC^1 complete w.r.t. AC^0 reduction if for any $L' \in \mathsf{NC}^1$, $L' \leq_{\text{strong}} L$, *i.e.* L' is reducible to L under AC^0 reduction. More details can be referred to Straubing (2012).

A.2. NC^1 complete regular language

In this subsection, we introduce the approach of connecting regular languages and NC^1 complete problems using the syntactic monoid theory and Barrington's theorem.

A.2.1. Syntactic monoid

The *syntactic monoid* is a concept in the algebraic language theory that establishes a connection between the language recognition and the group theory.

Definition A.1 (Syntactic congruence (Straubing, 2012))**.** Let A be a finite alphabet, and let $L \subseteq A^*$. We define an equivalence relation $\equiv_L$ on A^*: $x \equiv_L y$ iff.

$$\{(u, v) \in A^* \times A^* : uxv \in L\} = \{(u, v) \in A^* \times A^* : uyv \in L\} .$$

Note that $xa \equiv_L ya, ax \equiv_L ay$ if $x \equiv_L y, a \in A$, it follows that $\equiv_L$ is a congruence on A^*, called the *syntactic congruence*.

Definition A.2 (Syntactic monoid (Straubing, 2012))**.** Given a language $L \subseteq A^*$, the quotient of A^* by its congruence $\equiv_L$ is called the *syntactic monoid* of L and is denoted as $M(L)$.

For a regular language L, determining $M(L)$ can be accomplished using a straightforward method (Pin, 2013). The procedure involves initially computing its minimal DFA, with the syntactic semigroup of L being equivalent to the transition semigroup S of the DFA.

A.2.2. Barrington's Theorem

Barrington (1986) demonstrated that the word problem of the group S_5 is NC^1 complete. The word problem of a group G is defined as $\{g_1 \dots g_n = e : g_i \in G\}$. The following theorem offers a comprehensive statement of Barrington's work.

Theorem A.3 (Theorem IX.1.5 in Straubing (2012))**.** *Given a regular language such that $M(K)$ is not solvable. Then for all $L \in \mathsf{NC}^1, L \leq_{strong} K$.*

The methods used in the reduction process are simpler than NC^1; specifically, they involve employing AC^0 or TC^0 for the reduction. The well-known connection between this theorem and the original word problem of S_5 is as follows: for $n \geq 5$, the symmetric group S_n is unsolvable.

A.2.3. Examples of NC^1 complete Regular Language

Proposition A.4. *If $L = ((0 + 1)^3(01^*0 + 1))^*$, then L is NC^1 complete.*

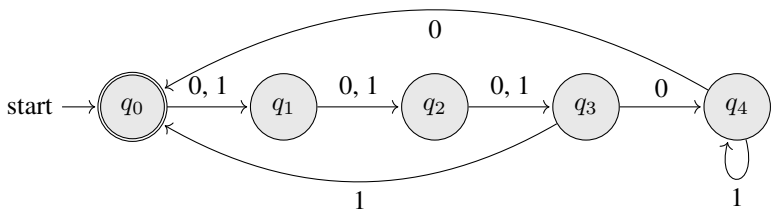


Figure 7. The minimal DFA of $((0+1)^3(01^*0+1))^*$

Proof. Let $f_w : Q \to Q$ represents an element in the transition group L, where $f_w(q)$ denotes reaching the node $f_w(q)$ after inputting the string w at node q. As illustrated in Figure 7, the transition group contains the following elements:

$$f_0 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \end{pmatrix},\ f_1 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 0 & 4 \end{pmatrix}.$$

Then $f_1^{-1} = f_1^3 = f_{111}$ and $f_0^{-1} = f_0^4 = f_{0000}$. Note that $(0\ 1\ 2\ 3\ 4)$ and $(0\ 1\ 2\ 3)$ are the generators of S_5 so $M(L) = S_5$ is not solvable. According to Theorem A.3, L is NC^1 complete. □

B. Theoretical Results

B.1. Proof of Proposition 4.1

Proof of Proposition 4.1. This proof is based on construction. Given a regular language $L \subseteq \Sigma^*$. We insert an end symbol $\# \notin \Sigma$ to obtain a new regular language $L^\# = (Q, \Sigma \cup \{\#\}, \delta, F, q_0)$ s.t. $w \in L$ iff. $w\# \in L^\#$. Construct a POMDP $\mathcal{M} = (S, A, T, R, \Omega, O, \gamma)$. The state space S is $Q \times (\Sigma \cup \{\#\})$. The action space A is $\{\textit{accept}, \textit{reject}\}$ and the observation space Ω is the alphabet $\Sigma \cup \{\#\}$. The initial state is (q_0, w_0), where w_0 is randomly sampled from $\Sigma \cup \{\#\}$. Given a state (q_t, w_t) at timestep t, the agent could observe the character w_t. If $w_t \neq \#$, the next state would be (q_{t+1}, w_{t+1}) where $q_{t+1} = \delta(q_t, w_t)$ and w_{t+1} is randomly sampled from $\Sigma \cup \{\#\}$, and the agent would receive no reward; If $w_t = \#$, the process would terminate, and the agent would receive a reward of 1 if it correctly outputs the acceptance of $w = w_0 \dots w_{t-1}$ in L. Note that the optimal policy π on $\mathcal{M}$ is to *accept* all $w \in L$ and *reject* all $w \notin L$, so if an algorithm $\mathcal{A}$ can solve POMDP problems, then $\mathcal{A}$ can recognize L. □

B.2. Proof of Theorem 4.4

Lemma B.1 (Theorem 2 in (Merrill & Sabharwal, 2023))**.** *Given an integer d and polynomial Q, any log-precision transformer with depth d and hidden size $Q(n)$ operating on inputs in Σ^n can be simulated by a logspace-uniform threshold circuit family of depth $3 + (9 + 2d_\oplus)d$.*

Remark B.2. The scope outlined by Lemma B.1 for Transformers is quite broad, as its description of FNNs allows for any log-precision function. Therefore, in the case of a log-precision $(\texttt{TF}, \texttt{RL})$ algorithm, we can distribute the $\texttt{RL}$ part across the last FNN layer of the original Transformer, treating the entire model as a single Transformer.

Proof of Theorem 4.4. Proof by contradiction. Suppose there exists an integer d and polynomial Q such that for any n, a log-precision $\mathcal{A} = (\texttt{TF}, \texttt{RL})$ with depth d and hidden size $Q(n)$ can solve $\mathcal{M}^L$, where L is a NC^1 complete regular language. Given $w \in \Sigma^*$, The algorithm $\mathcal{A}$ can determine the validity of $w \in L$ by checking whether the action output by $\mathcal{A}(w)$ is "accept". Consequently, $\mathcal{A}$ can solve an NC^1 complete problem.

At the same time, as stated in Remark B.2, we can treat $(\texttt{TF}, \texttt{RL})$ as a single Transformer. Based on Lemma B.1, $\mathcal{A}$ can be interpreted as a logspace-uniform threshold circuit family of constant depth, indicating that $L \in \mathsf{TC}^0$. Since we assume $\mathsf{TC}^0 \neq \mathsf{NC}^1$, the existence of such an algorithm $\mathcal{A}$ is not possible. □

B.3. Proof of Theorem 4.6

Lemma B.3. *Given an integer n, a symbol $a \in \Sigma$, a regular language $L \subseteq \Sigma^*$, let $L[n] = \{x \in L : |x| = n\}$ and*

$$P_{n,a} = \big\{(xa, x'a) \in L[n+1] \times (\bar{L})[n+1] : d(x, x') = 1\big\},$$

where $d(\cdot,\cdot)$ denotes the number of different symbols in x and x'. If $0 < c(n,a) < |\Sigma|^n$, then $|P_{n,a}| > 0$.

Proof. Suppose $|P_{n,a}| = 0$. Note that if $xa \in L[n+1]$, then $\{x'a : d(x,x') = 1\} \subseteq L[n+1]$. Repeating this deduction, we can cover all $x \in \Sigma^n$. Hence, $\Sigma^n a \subseteq L[n+1]$, that is, $c(n,a) = |\Sigma|^n$. If $xa \notin L[n+1]$, then $xa \in (\bar{L})[n+1]$. As stated above, $c(n,a) = 0$. By contradiction, $P_{n,a} > 0$. □

Proof of Theorem 4.6. Since Σ is finite, $D = \max_{a,b\in\Sigma} \|u(a) - u(b)\|$ where u denotes the embedding vector of the given symbol. By Lemma 4.5, there exists n such that $\{n : |P_{n,a}| > 0\}$ is infinte.

Therefore, there exists infinite sequences u and u' such that u and u' differ by only 1 position, yet $u \in L$ while $u' \notin L$. According to Lemma 4.5,

$$\|x - x'\| \leq \|u - u'\| = O(D/n) = O(1/n) \ .$$

Since $\texttt{RL}$ is a Lipschitz function, there exists a constant C such that

$$\|\texttt{RL}(x) - \texttt{RL}(x')\| \leq C\|x - x'\| = O(1/n) \ .$$

As n increases, information from non-current time steps will only have a negligible impact on the output of $\texttt{RL}$. □

C. Discussion on Existing POMDP Problems

Using existing POMDP problems as examples, here demonstrates the derivation of POMDPs cast into regular languages.

Passive T-Maze (Ni et al., 2023). The movement strategy towards the corridor's endpoint is akin to recognizing the regular language $0(01)^*$ with a DFA. If this DFA accepts the string formed by all current histories, then the agent moves upwards; otherwise, it moves downwards.

Passive Visual-Match (Hung et al., 2019). The complete state space of this environment is large, including player coordinates, coordinates of all fruits, whether fruits are collected, and other information. For convenience, we decouple this POMDP. Consistent with the analysis of Ni et al. (2023), this environment is divided into immediate greedy policies and long-term memory policies. For long-term memory policies, the environment not only needs to recognize the regular language $0(01)^*$ as in Passive T-Maze; but due to the existence of greedy policies, the player needs a strategy to move from any position in the room to the endpoint. The states required for this strategy only involve player coordinates, and judging the current coordinates based on historical information only requires a simple regular language. Considering directly treating the grid of the current room as the states of the DFA, and treating the actions $\{L, R, U, D\}$ as characters, if the player wants to determine the current position (x, y), it is equivalent to recognizing the regular language that accepted by a DFA whose terminal state is (x, y).

D. Experimental Details

D.1. Task descriptions

D.1.1. PYBULLET TASKS

Ant. This task is to simulate a hexapod robot resembling an ant. The objective is to develop a control policy that enables the ant to leverage the six legs for specific movements.

Walker. This task is to simulate a bipedal humanoid robot. The goal is to design a control strategy that facilitates stable and efficient walking, mimicking human-like locomotion patterns.

HalfCheetah. This task is to simulate a quadruped robot inspired by the cheetah's anatomy. The aim is to devise a control policy that allows the robot to achieve rapid and agile locomotion.

Hopper. This task is to simulate a single-legged robot, and the objective is to develop a control strategy for jumping locomotion to achieve efficient forward movement.

Task (F) stands for the original task with full observation, while Task (V) and Task (P) stand for that only velocities or positions are observable, respectively. In Figure 8, we provide the visualization of each task.

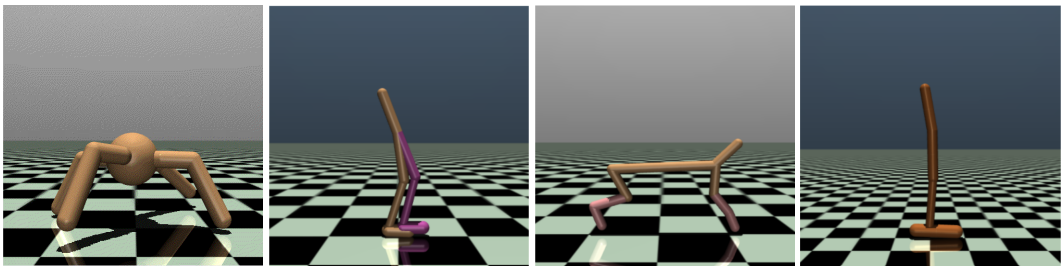


Figure 8. **Visualizations of Tasks in Pybullet**. From left to right are Ant, Walker, HalfCheetah and Hopper.

D.1.2. POMDPs DETERMINED BY REGULAR LANGUAGES

`PARITY`. Given a 01 sequence, compute whether the number of 1 is even. The formal expression is $0^*(10^*1)^*0^*$.

`EVEN PAIRS`. Given a 01 sequence, compute whether its first and last bit are the same. The formal expression is written as $(0[01]^*0)|(1[01]^*1)$.

`SYM(5)`. Given a 01 sequence, compute whether it belongs to a case of NC^1 complete regular languages, namely $\mathtt{S}_5$, with the formal expression $((0+1)^3(01^*0+1))^*$.

D.1.3. PURE LONG-TERM MEMORY TASKS

Passive T-Maze (Ni et al. (2023)). The environment is a long corridor of length L from the initial state O to J and J is connected with two terminal states G_1 and G_2. The horizontal positions of O and J are 0 and L, respectively. No information is observable except at states J, O, G_1, G_2. At J, O, G_1, G_2, the agent could observe its current position x_t, and at O, the agent could observe a signal G, which is uniformly sampled in $\{G_1, G_2\}$. The available actions are to move in 4 directions: left, right, up and down. The transitions are deterministic, and the agent would not move on hitting the wall. And the goal is to maximize the rewards, which is given by $\sum_{t=0}^{L} \frac{1(x_t \geq t)-1}{L} + 1(o_{L+1} = G)$.

Passive Visual Match (Hung et al. (2018)). The environment is a 7×11 grid-world consisting of 3 phases while only a 5×5 grid surrounding the agent is observable. In each episode of the task, a randomly assigned color is shown to the agent in the first phase; in the second phase, the agent could pick up apples with immediate rewards; in the final phase, the agent should pick up the assigned color shown among three randomly colored squares.

We show the visualization of each task taken from Ni et al. (2023) in Figure 9.

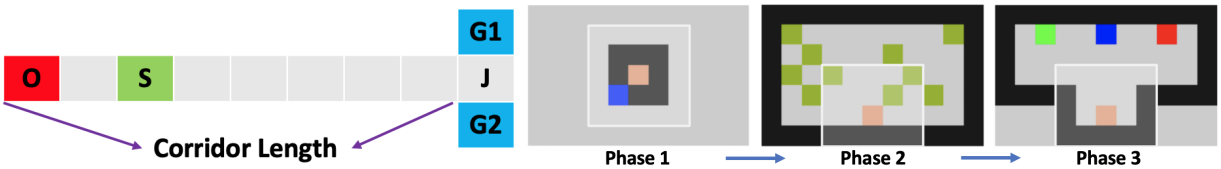


Figure 9. **Visualizations of pure long-term memory Tasks**. The left figure illustrates Passive T-Maze, and the right one indicates Passive Visual Match. Both of them are directly extracted from Ni et al. (2023).

D.2. Codebase

Our code is mainly based on Ni et al. (2022) (`https://github.com/twni2016/pomdp-baselines`) and Ni et al. (2023) (`https://github.com/twni2016/Memory-RL`). The implementation of LRU follows Orvieto et al. (2023) (`https://github.com/yueqirex/LRURec`;`https://github.com/Gothos/LRU-pytorch`). Our official code is released at `https://github.com/CTP314/TFPORL`.

D.3. Network architecture

Our network architecture is borrowed from Ni et al. (2023), and for completeness we show the extracted illustrations in Figure 10.

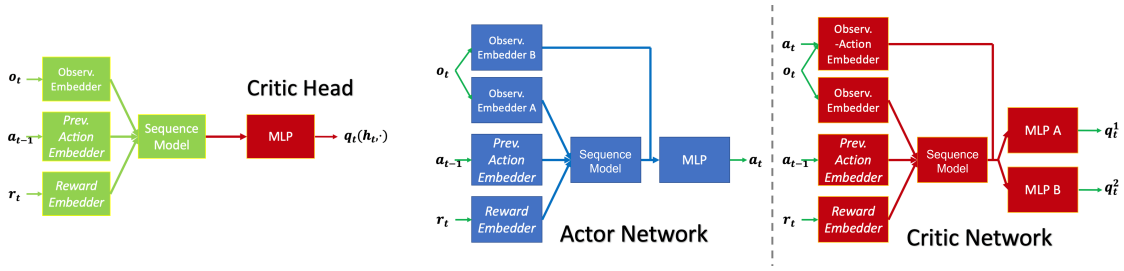


Figure 10. **The network architecture of (SEQ, RL).** The left figure illustrates (SEQ, DQN) and (SEQ, SACD), and the right one indicates (SEQ, TD3). Both of them are directly extracted from Ni et al. (2022; 2023).

D.4. Hyperparameters

We provide the configuration of hyperparameters in Table 3 and Table 4. Notably, to ensure strong alignment with the existing literature, we select RL algorithms identical to those in Ni et al. (2022; 2023), namely TD3 (Fujimoto et al., 2018) for PyBullet occlusion tasks, DQN (Van Hasselt et al., 2016) for Passive T-Maze and SACD (Haarnoja et al., 2018) for Passive Visual Match. For POMDP derived from Regular Language, due to its similarity with Passive T-Maze (equivalent to recognizing 01 sequences starting with 0), we employ DQN as well.

Table 3. **Hyperparameters of different POMDP tasks.**

	Hyperparameter	Regular Language	PyBullet Occlusion	Passive T-Maze	Passive Visual Match
Network	LSTM embedding $[h_o, h_a, h_r]$	32, 0, 0	32, 16, 16	32, 16, 0	2-layer CNN
	LSTM $[n_{\text{layer}}, h]$	128, 1	128, 1	128, 1	16
	LRU embedding $[h_o, h_a, h_r]$	64, 0, 0	32, 16, 16	64, 64, 0	2-layer CNN
	LRU $[h, n_{\text{layer}}]$	64, 2	64, 2	128, 1	128, 1
	GPT embedding $[h_o, h_a, h_r]$	64, 0, 0	32, 16, 16	64, 0, 0	2-layer CNN
	GPT $[h, n_{\text{heads}}, n_{\text{layer}}]$	64, 2, 2	64, 2, 2	128, 1, 1	128, 1, 1
	Length	$l_{\text{trajectory}}$	64	$l_{\text{trajectory}}$	$l_{\text{trajectory}}$
	RL	DQN	TD3	DQN	SACD
	MLP	not used	[256, 256]	[256, 256]	[256, 256]
Train & Eval	Environment steps	0.4M,1M,2M	1.5M	1M, 2M, 4M, 4M	2.4M, 3.75M, 5M
	Gradient steps	40k	1.5M	20k, 20k, 16k, 8k	40k, 30k, 32k
	Batch size	64	64	64	64
	Evaluation interval	100	4k	10	50
	Evaluation episodes	100	10	10	20

Table 4. **Hyperparameters of different RL algorithms.**

RL algorithm	DQN	TD3	SACD
Hidden dimension	(256, 256)	(256, 256)	(256, 256)
Exploration noise	-	0.1	-
Target noise	-	0.2	-
Target noise clip	-	0.5	-
Discount factor	0.99	0.99	0.99
Smoothing coefficient	0.005	0.005	0.005
Learning rate	3e-4	3e-4	3e-4
Replay buffer size	1M	1M	1M

D.5. Supplementary results

In this subsection, we provide additional experimental results for supplementation.

PyBullet occlusion tasks. In Table 6, we show the original return for each PyBullet occlusion task and sequence model, while the normalized score in Table 2 is calculated as

$$\text{Normalize score} = \frac{R - R_{\text{random}}}{R_{\text{expert}} - R_{\text{random}}} \times 100\,, \tag{3}$$

where R denotes the return of a certain instance, R_{expert} denotes the return from the expert policy, and R_{random} denotes the return from a random policy (both provided by Fu et al. (2020)). Scores used for normalization are provided in Table 5. The results elucidate that LRU and LSTM demonstrate superior performance over GPT across all eight tasks and LRU distinctly surpasses LSTM in specific tasks, particularly in Ant (P, V), and Walker (V).

The hyperparameters and code used in our experiments are derived from Ni et al. (2022), ensuring consistency and fairness with the results presented in Table 7, Figure 9, and Figure 16 of Ni et al. (2022). We additionally conducted experiments on the Ant environment that show significant deviations from Ni et al. (2023) .The training curves in Figure 11 demonstrate that LRU still outperforms LSTM.

Table 5. **Scores used for normalization.** Scores of each task are linearly normalized by the corresponding random score and expert score.

Task Name	Random Score	Expert Score	Task Name	Random Score	Expert Score
Ant	373.71	2650.50	Cheetah	-1275.77	2381.67
Walker	16.52	1623.65	Hopper	20.06	1441.81

Table 6. **Returns for PyBullet occlusion tasks**. We compare different sequence models LRU, GPT and LSTM. 'V' refers to 'only velocities observable', and 'P' refers to 'only positions observable'. Blue highlight indicates the highest score, and orange highlight indicates the second-highest score.

Task	Type	LRU	GPT	LSTM
Ant	V	1052.4 ± 465.5	587.0 ± 161.6	542.0 ± 200.0
	P	2222.5 ± 652.8	1242.5 ± 591.8	504.1 ± 74.1
Cheetah	V	2266.2 ± 297.7	1258.9 ± 251.5	2313.6 ± 304.6
	P	2743.1 ± 151.9	1972.1 ± 232.2	2840.0 ± 198.1
Hopper	V	1357.8 ± 332.5	211.4 ± 7.7	1193.3 ± 527.1
	P	2122.4 ± 174.7	357.8 ± 256.4	2637.0 ± 190.7
Walker	V	1007.7 ± 234.3	374.4 ± 95.7	212.6 ± 112.4
	P	1290.3 ± 371.1	812.2 ± 77.4	1536.8 ± 588.2

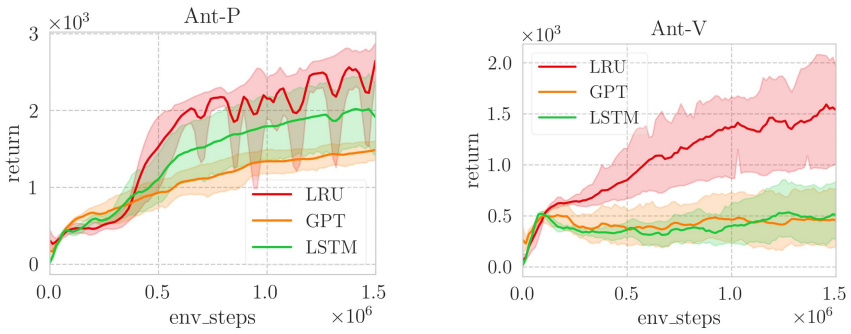


Figure 11. **Learning curves for PyBullet occlusion tasks(Ant) in Ni et al. (2023)**. The shaded area indicates 95% confidence intervals.

PyBullet fully observable tasks. To verify that the performance degradation of GPT compared to the other two models is due to partial observability of the environment, we carry out an ablation study on the partial observability, providing the normalized and original returns for fully observable PyBullet tasks in Table 7, and learning curves in Figure 12.

Table 7. **Returns for PyBullet fully observable tasks**. We compare different sequence models LRU, GPT, and LSTM. 'F' refers to 'fully observable'. The left table stands for the original score, and the right one indicates the normalized score. Blue highlight indicates the highest score, and orange highlight indicates the second-highest score. The overall performance of the three models are close, affirming that GPT's inferior performance in POMDP scenarios stems from partial observability rather than other factors.

Task	Type	LRU	GPT	LSTM
Ant	F	3218.2 ± 42.9	2980.8 ± 46.1	2799.2 ± 122.9
Cheetah	F	3101.9 ± 182.3	2959.6 ± 424.6	3150.4 ± 458.5
Hopper	F	2613.6 ± 73.2	2533.9 ± 2.1	2793.7 ± 18.9
Walker	F	2134.4 ± 154.9	2152.8 ± 239.4	2241.6 ± 109.5

Task	Type	LRU	GPT	LSTM
Ant	F	124.9 ± 1.9	114.5 ± 2.0	106.5 ± 5.4
Cheetah	F	119.7 ± 5.0	115.8 ± 11.6	121.0 ± 12.5
Hopper	F	182.4 ± 5.1	176.8 ± 0.1	195.1 ± 1.3
Walker	F	131.8 ± 9.6	132.9 ± 14.9	138.5 ± 6.8
Average		139.7	135.0	140.3

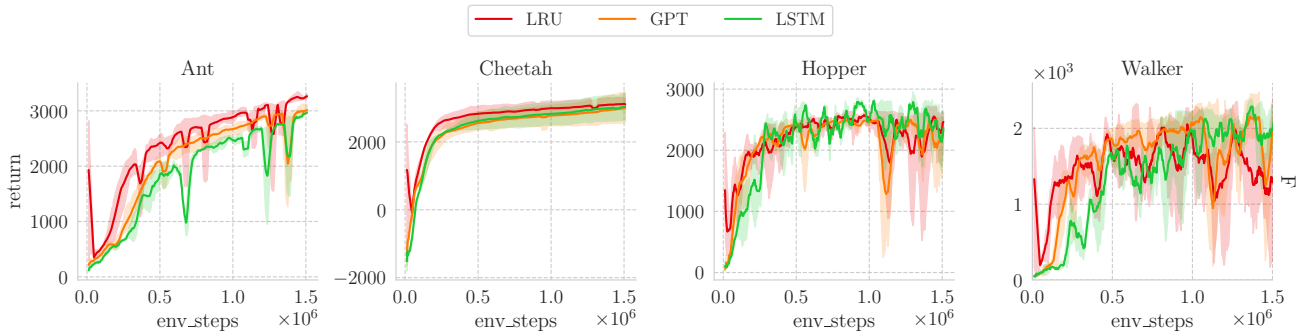


Figure 12. **Learning curves for PyBullet fully observable tasks**. The shaded area indicates 95% confidence intervals.

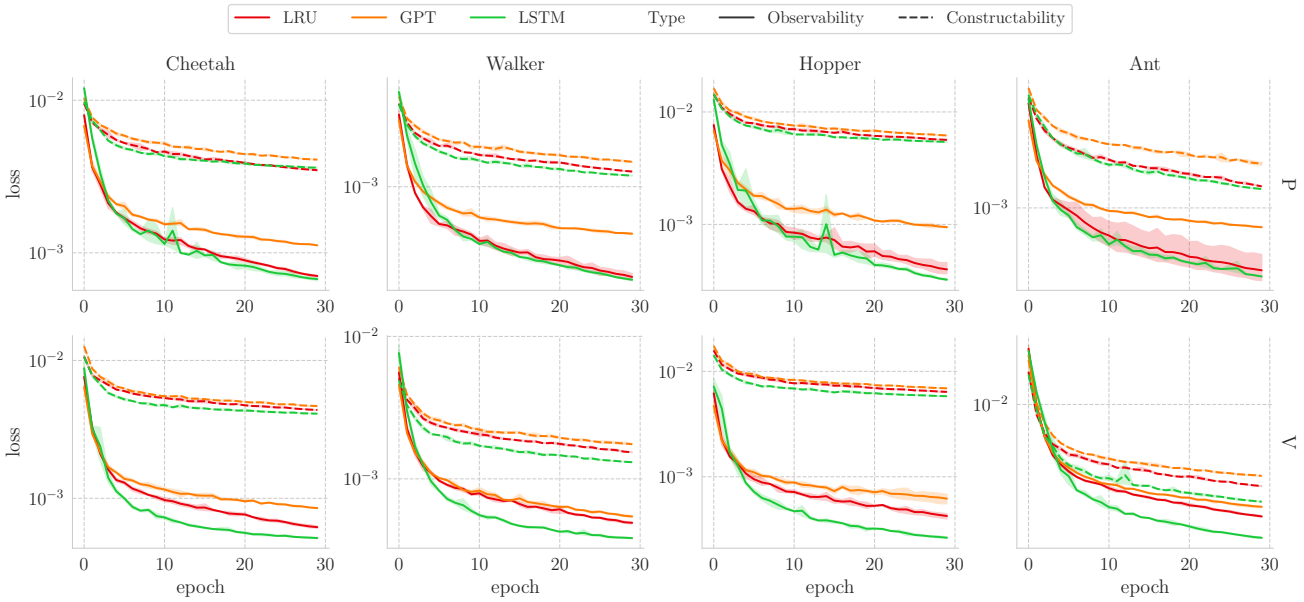


Figure 13. **Learning curves for state space modeling tasks**. We present the curves of average MSE loss over 3 seeds. The shaded area indicates 95% confidence intervals.

PyBullet state space modeling tasks. We provide the MSE loss curve throughout training for PyBullet state space modeling tasks in Figure 13, while in Figure 5, we present the average MSE ratios over the 8 tasks. The MSE ratio is computed as

$$\text{MSE ratio} = l/l_{\min} \ , \tag{4}$$

where l denotes the final MSE loss of the training curve, and $l_{\min}$ denotes the minimal l over the 3 models.

Length Extrapolation in Regular Language Tasks. Figure 14 shows the results for regular language tasks with training lengths less than or equal to $n = 25$, tested at $n/2$ and $n + i$ for $i \in \{1, 2, 3, 4, 8, 16, 32\}$. Due to RNN's completeness over regular languages, RNN exhibits the best generalization ability. On `SYM(5)` and `PARITY`, LRU significantly outperforms GPT, which aligns with the discussion in Sec 5, mitigating the theoretical constraints of GPT. On `EVEN PAIRS`, which naturally conforms to attention's inductive bias, GPT also demonstrates better generalization ability than LRU, but it fails at longer lengths (all three languages fall within the scope of Theorem 4.6). This result is consistent with the visualization results in Figure 3, where LRU shows good recovery effects for hidden states on `SYM(5)` and `PARITY`, while GPT's hidden states exhibit significant discrepancies from the true distribution.

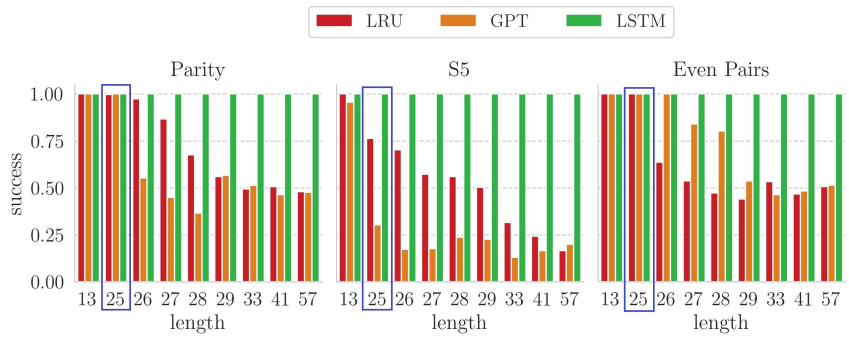


Figure 14. **Length Extrapolation in Regular Language Tasks**. Mean of 5 seeds. Training length with the longest length enclosed in a blue box.

GPT Scale in Regular Language Tasks. Figure 15 shows learning curves of different scales on the `SYM(5)` ($n = 25$) task with Transformer. The results show that scaling up GPT does not yield significant effects, consistent with the statement of Theorem 4.6. Transformer is only equivalent to a computation circuit of width $O(\text{poly}(n))$ and depth $O(1)$. Changing the parameter count merely increases the width or changes the constant width, while not significantly enhancing the Transformer's ability to solve NC^1 complete problem.

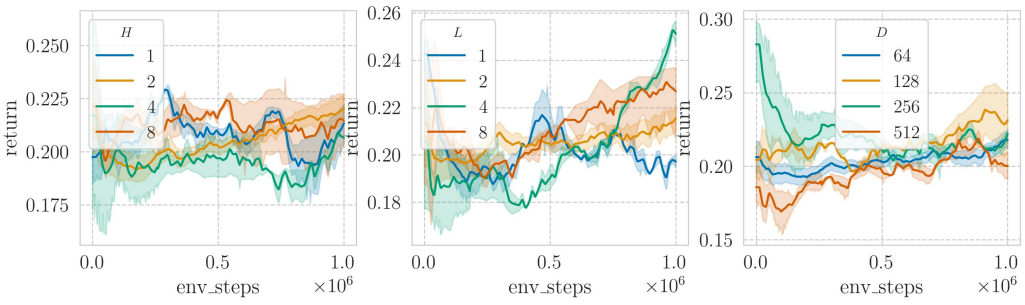


Figure 15. **GPT Scale in `SYM(5)`**. Mean of 5 seeds. H, L, D denote hidden size, the number of layers and the depth, respectively.

Regular language tasks and long-term memory tasks. The learning curves for regular language tasks are shown in Figure 16: in `EVEN PAIRS`, all the sequence models could solve this task even in the hardest case; in `PARITY`, LSTM shows significant advantages over LRU and GPT; in `SYM(5)`, GPT performs the worst, and is not capable of solving the medium case.

In this table 8, the silhouette scores (Rousseeuw, 1987) for each task are shown for different models to demonstrate the quality of hidden state representations.

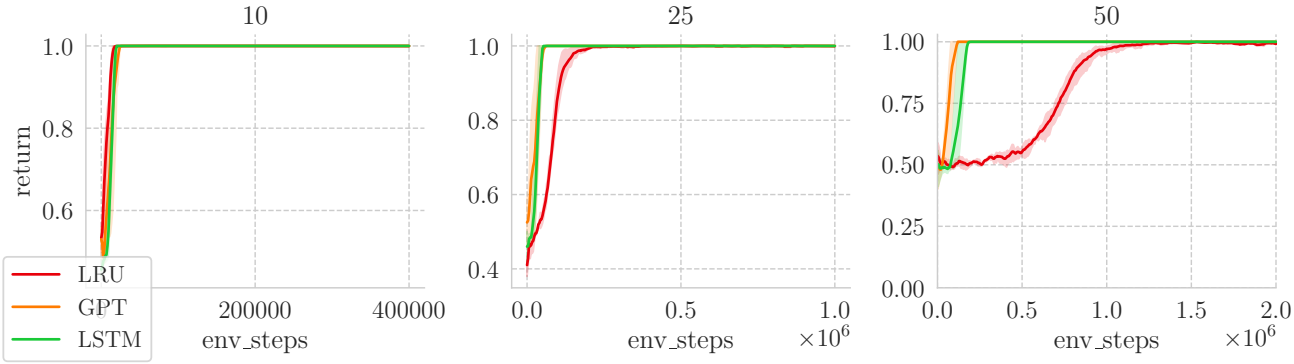
- In PARITY and SYM(5) tasks, both LRU and LSTM significantly outperform GPT, consistent with the performance shown in Figure 3. In PARITY, both LRU and LSTM implicitly learn the true state within the (SEQ, RL) framework, hence their scores are relatively close. However, in the more complex DFA structure of SYM(5), there remains a gap between LRU and LSTM, consistent with the training situation depicted in Figure 3.
- In EVEN PARIS, LSTM performs noticeably better than the other two models. Both LRU and GPT struggle to learn the true state. For LRU, it faces difficulty in capturing information about the first token from history. For GPT, although its state reconstruction in intermediate moves is poor, it manages to retain contextual information throughout, allowing it to directly extract information about the first token at termination states. This aligns with the visualization results where termination states match previous states, and each termination state exhibits significant separation, as depicted.

The learning curves for pure long-term memory tasks are shown in Figure 17: LRU and GPT far outweigh LSTM in these two tasks, demonstrating their long-term memory abilities which LSTM prominently lacks. Although in the Passive T-Maze, GPT could solve the hardest case while LRU and LSTM could not, LRU exhibits strong sample efficiency and memory capacity in the more complicated task, Passive Visual Match, beating GPT and LSTM remarkably.

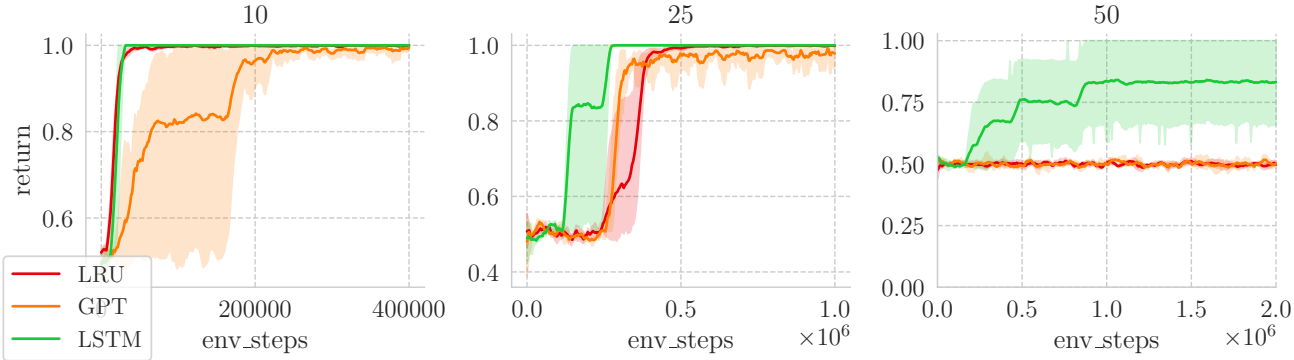
Table 8. **Scores for hidden state representation in regular language tasks** The silhouette score ranges from -1 to 1, where a score closer to 1 indicates that the samples are well-clustered and far from neighboring clusters, signifying a good separation between different states and good cohesion within the same state.

Task	LRU	GPT	LSTM
PARITY	0.12	-0.03	0.1
SYM(5)	0.05	-0.13	0.12
ENV PARIS	0.02	-0.02	0.2

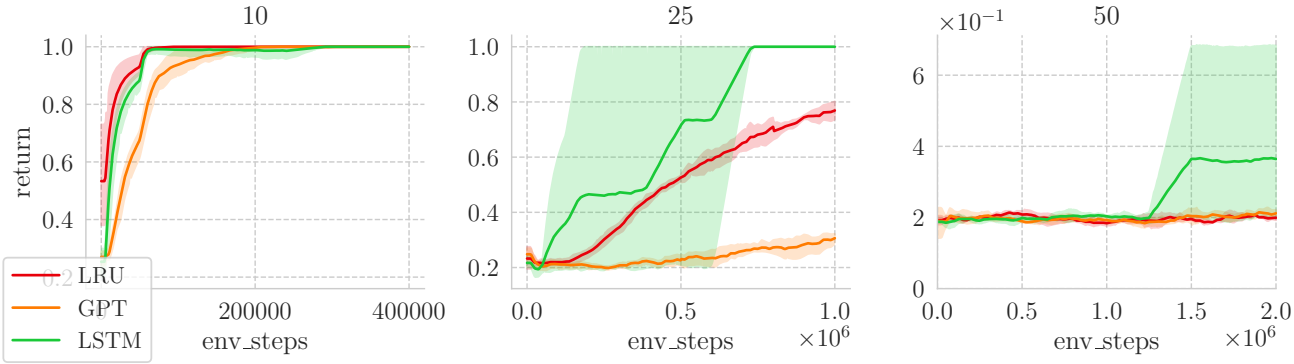
Comparison with some published Transformer in RL. We compared another practical POMDP environment: control tasks with random frame dropping (Hu et al., 2023), and the popular DT (Chen et al., 2021) used in current Offline RL approaches. When applied to the processed dataset, DT failed to learn a control policy in the frame-dropping situation and required additional information to mitigate the partially observable condition. However, when replacing Transformer with RNN or Linear RNN, DRNN or DLRNN exhibited the ability to handle such problems without additional conditions. Figure 18



(a) EVEN PAIRS

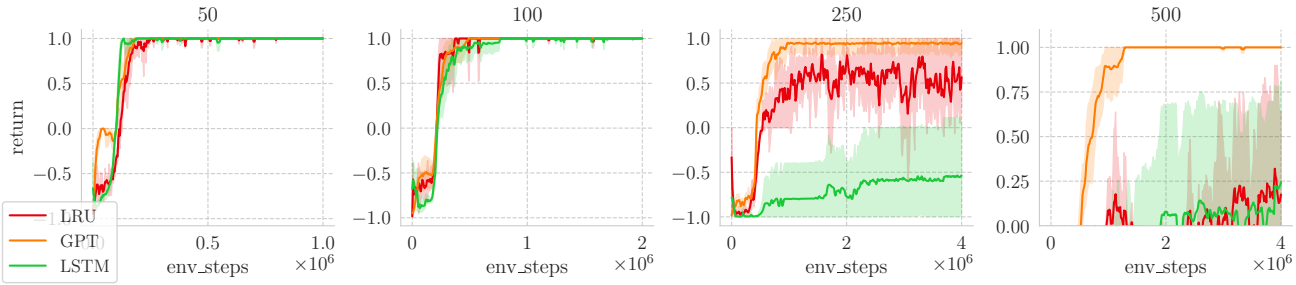


(b) PARITY

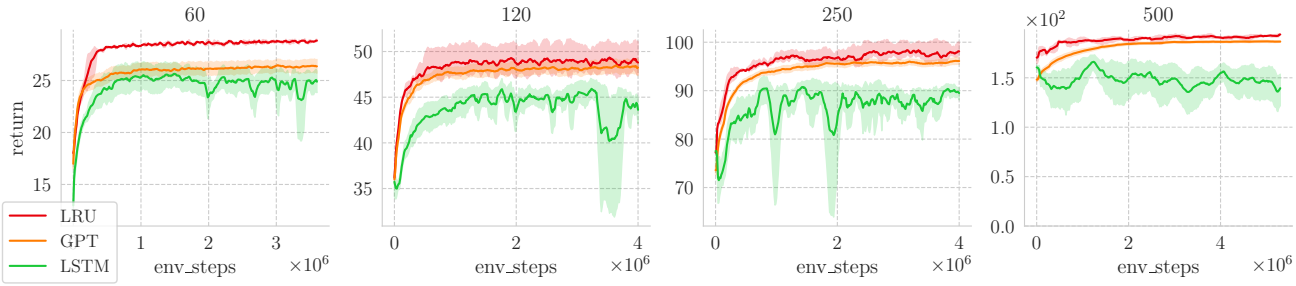


(c) SYM(5)

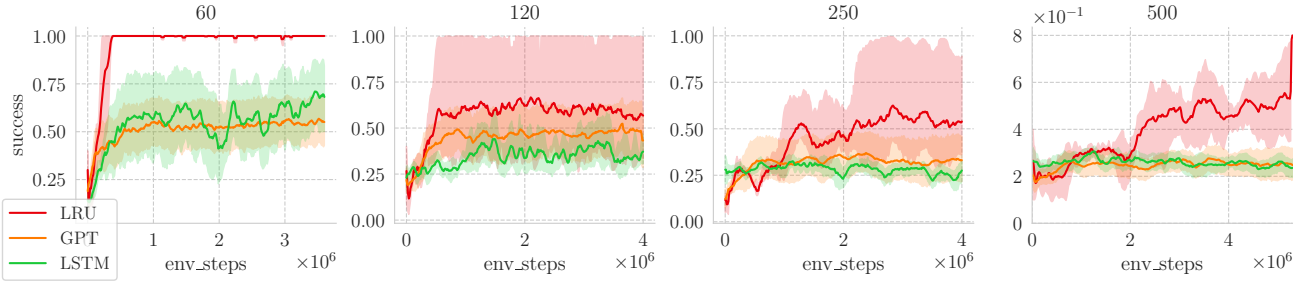
Figure 16. **Learning curves for regular language tasks**. The shaded area indicates 95% confidence intervals. The upper number stands for the memory length.



(a) Passive T-Maze (return)



(b) Passive Visual Match (return)



(c) Passive Visual Match (success)

Figure 17. **Learning curves for pure long-term memory tasks**. Mean of 3 seeds. The shaded area indicates 95% confidence intervals. The upper number stands for memory length.

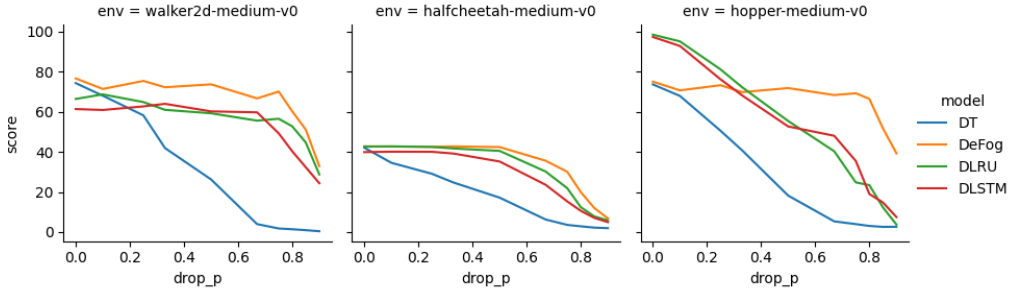


Figure 18. **Performance on continuous control tasks. with random frame dropping.** Mean of 5 seeds. We selected these medium datasets as the training set. The x-axis represents the probability of dropped frames, and the y-axis represents the D4RL normalized score (Fu et al., 2020).