ImReasoner: Improving Memory-based Language Models for Reasoning-in-a-Haystack Tasks

Ching-Yun Ko[1], Payel Das[1], Sihui Dai[1,2], Georgios Kollias[1], Subhajit Chaudhury[1], Aurelie C. Lozano[1], Pin-Yu Chen[1],

[1]IBM Research, [2]Princeton University,

Correspondence: Payel Das daspa@us.ibm.com

Abstract

Reasoning over long contexts remains a major challenge for language models, particularly when solving tasks that require integrating multiple facts in sequence or generalizing to new distributions. We argue that this difficulty stems from a lack of structural inductive bias. Recently, alternative frameworks have been proposed to explicitly encode contexts as ordered memory and perform iterative retrieval to construct reasoning chains. Despite the promising results shown in prior arts, they are still heavily reliant on intermediate chain supervision and fall short in showing emergent reasoning generalization in the presence of hard distractions in reasoning-in-a-haystack tasks. Furthermore, we discover that as the amount of distractions increases, traditional episodic memory reads suffer from ill-conditioning problems, which lead to inaccurate context retrievals. In this work, we formalize the motivation for necessary inductive bias in reasoning-in-a-Haystack tasks, propose inference-time memory update procedures mimicking the "identify and remove unnecessary and unrelated details" in *constructively responsive reading*, introduce staged training inspired by human conceptual understanding, and finally demonstrate the possibilities and limits of such framework in the weakly supervised scenario.

1 Introduction

Despite rapid advances, large language models (LLMs) continue to exhibit brittleness on tasks that require multi-step reasoning over long contexts. Oftentimes, irrespective of the specific type of reasoning involved, LLM hallucinations occur from failure to resolve a dependency chain over the input. This issue of incorrectly modeling long-range dependency and reasoning has been reported for a variety of tasks including logical reasoning (Levy et al., 2024; Kuratov et al., 2024; Wan et al., 2024) and algorithmic reasoning (Liu et al., 2023b,a).

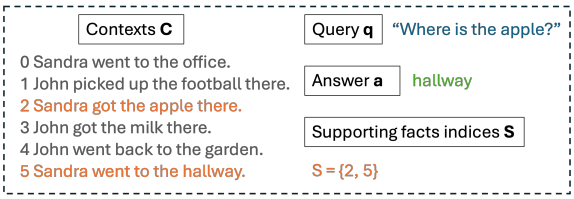


Figure 1: An example of contexts, query, answer, and supporting fact indices.

To better understand and characterize these weaknesses, multiple synthetic benchmarks (Hsieh et al., 2024; Kuratov et al., 2024; Liu et al., 2023a) have been proposed to stress-test these LMs in aspects including temporal awareness, coreference resolution, and fact chaining. These synthetic benchmarks complement real-world benchmarks like Long Range Arena (Tay et al., 2020) and BigBench (Srivastava et al., 2022), and consistently reveal significant reasoning gaps in current LMs (Hosseini et al., 2024). Among them, authors of BABILong (Kuratov et al., 2024) argue that when relevant facts are "hidden" among vast amounts of irrelevant background text ("needle-in-a-haystack"), many models fail to robustly retrieve and reason over the salient facts. They tested models of various sizes and architectures across context lengths and showed that recurrent memory transformers (RMT) (Bulatov et al., 2022) after fine-tuning on BABILong reached the highest performance.

More recently, other alternative LM architectures (Gu and Dao, 2023; Peng et al., 2023; Behrouz et al., 2024; Ko et al., 2024; Schneider et al., 2025; Ren et al., 2025) have been brought out and challenged on the benchmarks. One such example is MemReasoner (Ko et al., 2024) where the authors propose adding explicit temporal ordering to facts and designing iterative read/update operations over memory to enable the model to hop

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 606–622
July 2-7, 2026

across facts in an explicitly stepwise fashion. Take Figure 1 as an example, MemReasoner obtains the embeddings of lines in context independently, and will pass these embeddings sequentially through a GRU to encode the temporal order of these contexts. As a result, it will become clear that "Sandra went to the hallway" happens after "Sandra got the apple there". In this example, these two lines are the supporting facts for answering the question "Where is the apple?" and we denote their indices $\{2, 5\}$ as the supporting fact indices. They argue that RMT and Mamba (Gu and Dao, 2023) achieve superior performance by being exposed to BABILong samples during training as opposed to having fundamental generalization capability (e.g. they show much inferior performance when only fine-tuned on similar questions with no context distractors such as bAbI (Weston et al., 2015)). Without the access to the testing distribution (i.e. BABILong), MemReasoner has shown up as the state-of-the-art model for long-context reasoning-in-a-haystack tasks. Nevertheless, we point out that similar criticism can also be applied to MemReasoner (Ko et al., 2024) since, despite not exposed to the testing distribution, its training requires the indices of corresponding intermediate supporting facts at each hop. This reliance on fine annotations complicates the assessment of whether reasoning capability emerge from the inductive bias (the explicit temporal ordering and memory operations).

In this work, we want to firstly motivate the use of alternative architectures with inductive bias for needle-in-a-haystack tasks. Secondly, we will delve into the details and analyze failure modes in MemReasoner, especially when supporting facts are absent or scarce. We identify major pitfalls during both training time and testing time: (1) ill-conditioning of episodic memory operations under heavy distraction, which leads to unstable and inaccurate retrieval during inference time, and (2) representation collapse during training time when supporting fact information is unavailable or limited. To address these issues, we are motivated by how humans *learn* and *do* better reasoning: 1. learn to have a better knowledge about the problem they are solving, and 2. ignore unimportant parts during problem solving. Specifically, we introduce a staged training procedure inspired by *conceptual understanding and procedural skill* (Byrnes, 1992; Rittle-Johnson et al., 2001) that emphasizes mastering conceptual knowledge before solving tasks. With the trained reasoner, we propose an inference-time memory update method that filters irrelevant information at its first pass, analogous to the "identify and remove unnecessary and unrelated details" strategy in *constructively responsive reading* (Pressley and Afflerbach, 1995) (See Figure 4), which at the same time helps better conditioning the optimization problems in episodic memory operations. Together, these techniques allow us to train and deploy memory-augmented models under weaker supervision.

We highlight our main contributions as follows:

- Training setup: ImReasoner uses staged training that separates representation learning from reasoning-update learning, preventing collapse during multi-step supervision.
- Supervision regimes: We evaluate under both weak supervision (answer-only) and strong supervision (supporting-facts), and we analyze how supervision type interacts with model stability.
- Benchmarks and context lengths: Experiments cover bAbI (0-1k tokens), BABILong (0-128k tokens), and BABILong-soft (0-4k tokens, in-distribution distractors).
- Representative result: On BABILong, inference-time filtering alone improves accuracy dramatically under long contexts (e.g., from 14% to 82% at 16k tokens), and performing staged training further stabilizes and increases the accuracy (e.g., from 82% to 94% at 16k token).

2 Methodology

2.1 Background: MemReasoner

Let $\mathcal{X}, \mathcal{Z}, \mathcal{Y}$ be the input, latent, and output space. e be the encoder that maps an input to an embedding in $\mathcal{Z}$, $\mathcal{M}$ be a memory module, $\mathcal{P}$ be a temporal encoding module, and d be the decoder that maps an embedding to an output in $\mathcal{Y}$. MemReasoner (Ko et al., 2024) builds on top of Larimar (Das et al., 2024) that first encodes inputs to their latents, updates the memory $\mathcal{M}$, and performs decoding conditioned on the memory readout. For example, given a list of contexts $C = \{c_1, ..., c_E\}$ composed of E sentences, the target task is to answer a question q conditioned on the given context C. To approach the task within MemReasoner/Larimar frameworks, the input, both con-

text C and query q, are encoded to their latents ($z_1, \ldots, z_E$ and z_q) via the encoder e. Next, let M_0 be the initial memory, write the context to the memory via a $write$ operation (Wu et al., 2018), i.e. $\hat{M} = write(M_0, z) := (Z_\xi M_0^\dagger)^\dagger Z_\xi$, where $Z_\xi = [z_1 + \xi_1, z_2 + \xi_2, \ldots, z_E + \xi_E]$, $\xi_i \sim \mathcal{N}(0, \sigma_\xi^2 I)$, and $\dagger$ demotes matrix pseudo inverses. Then, the $read$ operation translates the query embedding through the lens of the context-aware memory to a query readout z_r, i.e. $z_r = read(\hat{M}, z_q) := (z_q \hat{M}^\dagger + \eta)\hat{M}$, where $\eta \sim \mathcal{N}(0, \sigma_\eta^2 I)$. Lastly, the decoder d decodes the query q conditioned on the readout by using a broadcasting matrix that casts z_r to each decoder layer and obtains h_k^m that serves as the past key values for $k = 1, \ldots, L$, where L is the number of layers in the decoder.

On top of the standard episodic memory network pipeline, MemReasoner introduces two additional features: (1) explicit temporal ordering of facts in the context, and (2) explicit iterative reading from the memory and updating of the query.

Memory with Temporal Order. MemReasoner introduces a temporal encoding module $\mathcal{P}$ that transforms unordered fact latents $\{z_1, ..., z_E\}$ to their ordered counterparts $\{\tilde{z}_1, ..., \tilde{z}_E\}$, explicitly "adding a timestamp" to events to show the order of their appearances. Specifically, this is done by letting $\mathcal{P}$ be a learnable bidirectional GRU, treating $[z_1, ..., z_E]$ as the input sequence to $\mathcal{P}$, and marking the sequential outputs of $\mathcal{P}$ as $[\tilde{z}_1, ..., \tilde{z}_E]$, i.e. $[\tilde{z}_1, ..., \tilde{z}_E] \leftarrow \text{GRU}([z_1, ..., z_E])$. MemReasoner then writes these ordered context embeddings $\{\tilde{z}_1, ..., \tilde{z}_E\}$ to the memory instead of the original unordered ones.

Iterative query update and memory read. On multi-hop reasoning tasks, MemReasoner explicitly performs iterative query update and memory read. Specifically, it updates the query latent by $z_q \leftarrow z_q + \alpha \cdot z_r$ after each memory read, where $\alpha \in \mathbb{R}$ is a hyperparameter to balance the load from the previous readout. The updated query is then fed into the memory module for another $read$ operation to obtain a new $\tilde{z}_r$ until convergence or a user-specified maximum number of iterations.

2.2 Helpful Inductive Bias in Reasoning-in-a-Haystack task

As a single self-attention layer processes query, keys, and values through $\text{Attn}(Q_i, K, V) = \sum_{j=1}^{E} \alpha_{ij} V_j$ for each position i in the query embedding, where V_j is the value representation of the j-th input embedding and $\alpha_{ij} = \frac{\exp(Q_i^\top K_j/\sqrt{d})}{\sum_{j'} \exp(Q_i^\top K_{j'}/\sqrt{d})}$. Thus the output is a convex mixture of all input token representations and multi-head attention concatenates multiple mixtures. While the mixture nature might serves tasks such as summarization or single-hop question answering well, it is intuitively not suitable for tasks that require ordered composition of facts. For example, given "fact 1: Mary asks Daniel to review paper *P*, fact 2: Daniel passes the paper to John to review, query: who needs to review *P* now?" To solve the problem, the model needs to first infer fact 1 to understand the association between Daniel and *P*, and then trace to John from Daniel through fact 2. Modern attention schemes such as RoPE and ALiBi do introduce positional structure within every attention layer by influencing the query–key interactions in a position-dependent manner; however, these schemes do not change the fundamental fact that each layer still produces a weighted sum of value vectors. Consequently, although a multi-layer LM may implicitly capture the reasoning structure across layers, models that lack explicit inductive biases often fail to generalize robustly (Lake and Baroni, 2018; Bahdanau et al., 2019; Hupkes et al., 2020).

We argue that ordered memory is a helpful inductive bias for (some) needle-in-a-haystack tasks. For example, MemReasoner introduces structure by segmenting the contexts into discrete memory slots (e.g. line by line). The ordering via the temporal encoding module $\mathcal{P}$ then reduces the combinatorial ambiguity. That said, for a k-hop problem with E contexts, the effective hypothesis space of the reasoning chain reduces by a factor of $k!$ (from $\binom{E}{k} \cdot k!$ to $\binom{E}{k}$).

We note that solving the reasoning problem in one step remains difficult even with the ordering, and iterative retrieval breaks the query to sub-queries that are easier to address individually. Specifically, let $S = \{i_1, \ldots, i_k\}$ denote the supporting fact indices (see Figure 1 for an example), identifying them all in one step requires a single query readout z_r to lie close to each of the k supporting fact latents $\tilde{z}_{i_1}, \ldots, \tilde{z}_{i_k}$ simultaneously, i.e. $|z_r - \tilde{z}_{i_t}| < |z_r - \tilde{z}_j|$ for $\forall j \notin S$ and $\forall t$. This implicitly assumes that the relevant contexts form a tight cluster in the representation space, a condition often violated due to their semantic differences. Iterative query update and memory read relax this

constraint by allowing the query to evolve after each step. As a result, an accurate fact retrieval requires only that each relevant fact be locally nearest to the current query readout at its respective step, i.e. $|z_r^{(t)} - \tilde{z}_{i_t}| < |z_r^{(t)} - \tilde{z}_j|$ for $j \neq i_t$, a substantially weaker geometric condition.

2.3 ImReasoner

2.3.1 Constructively responsive reading - inference-time memory update

To generalize on long-context tasks with a memory-augmented LLM, a potential hurdle can come from the two fundamental operations of the memory, write and read, as described in Section 2.1. In both operations, numerical solves of the linear systems involve computing matrix pseudo inverses, which can be unstable when the matrix has many more columns than rows or the other way around. Secondly, encoding the temporal order in very long facts with a GRU can further incur vanishing or exploding gradient.

To cope with these, we introduce an inference-time update method inspired by the theory of constructively responsive reading (Pressley and Afflerbach, 1995), which characterizes skilled human comprehension as a two-stage, goal-driven process: readers first skim and filter a text to identify potentially relevant evidence, and only then engage in deeper analysis of the selected content. Analogously, we aim to perform an initial relevance estimation to discard unrelated context and subsequently reason over the remaining materials to answer the question. Specifically, with the first pass of memory write and query read, we identify a subset of contexts that are most relevant to the query by their proximity in the ordered latent space, i.e. $J = \{j \in [E], s.t. ||z_r - \tilde{z}_j||_2 \leq Q_p\}$, where Q_p is the quantile function defined as $\inf\{q|P(||z_r - \tilde{z}_j||_2 \leq q) \geq p$. In practice, we filter out most of the contexts after the first pass and leave less than 1k lines, or $p = 1/m$ for m-k-length needle-in-a-haystack task. Then, we re-encode the temporal order on this remaining contexts (much shorter) to get their ordered counterparts, i.e., $[\tilde{z}'_1, ..., \tilde{z}'_{|J|}] \leftarrow \text{GRU}([z_1, ..., z_{|J|}])$. A subsequent memory write and a query read are followed to get the final readout.

We also note the potential of applying inference-time update mechanism on vanilla transformer architectures. Specifically, during inference, one can aggregate the attention keys over individual sentences, where low-scoring sentences (based on the filtering score) can be down-weighted (or masked) before the next layer's attention computation. From this perspective, it is also relevant to sparse attention pruning in generic transformer architectures.

2.3.2 Conceptual understanding - staged training

Another issue we identify in MemReasoner's training dynamics in the absence of intermediate supporting fact supervision is representation collapse (Goodfellow et al., 2016), meaning the latent space loses its ability to capture meaningful distinctions in the input contexts and the training loss find its way down by creating shortcuts or memorization. For example, across randomly sampled instances, the average variance of the latents (i.e. assume latent $Z \in \mathbb{R}^{E \times d}$, calculate $\text{mean}(\text{Variance}(Z, \text{dim} = 0))$) remains extremely low, around 0.044. The average pairwise cosine similarity between them also appears high, around 0.59, indicating the model maps semantically different context lines into a highly compressed region.

To remedy this, we introduce a staged training strategy designed to first force the model to internalize representations of context before exposing it to answer prediction tasks, avoiding shortcut representations (Ranzato and Szummer, 2008). This idea can trace back to principles in cognitive science which explains learners often master conceptual understanding before procedural competence and that procedural errors can stem from weak conceptual bases (Byrnes, 1992; Rittle-Johnson et al., 2001). Concretely, during the initial phase, spanning the first 50 epochs, the model is trained solely on reconstruction losses akin to an autoencoder objective, ensuring that the latent space does not collapse into overly similar embeddings for different context lines. This step helps to preserve the diversity of the representations. Subsequently, after the 50th epoch, we reintroduce the answer reconstruction loss alongside the reconstruction objective. This staged approach allows the model to first learn robust and varied representations before fine-tuning to the specific task, thereby enhancing overall performance and stability. We note that, with staged training, a significant increase in the latent variance can be seen from 0.044 to 0.097, while the average pairwise cosine similarity decreases from 0.59 to 0.39. Such improved distinguishability provides a

stronger foundation for downstream reasoning and leads to more stable training.

As the above two schemas aim at solving two failures in existing memory-augmented LLM reasoner, we hereby dub our method **ImReasoner**: **i**mproved **m**emory-augmented **reasoner**.

2.3.3 Training objective

Let $\mathcal{D}_{\text{pretrain}}$ denote the pretraining data distribution and $\mathcal{D}_{\text{finetune}}$ denote the data distribution corresponding to the reasoning task. Each sample from $\mathcal{D}_{\text{finetune}}$ is of the form (q, C, a, S) where q is the query, $C = \{c_1, ..., c_E\}$ are the facts in the context, a is the answer, and S is the set of supporting fact indices (see Figure 1 for an example). Depending on the availability of supporting fact information, S can be empty. Meanwhile, the pretraining distribution corresponds to a generic corpus, e.g. Wikipedia. Recall that e and d denotes the encoder and decoder, $z_r^{(t)}$ denote the t-th readout from iterative reading, z_j and $\tilde{z}_j$ denote the un-ordered and ordered context latents, and P_a denotes the prompts for generating the answer. To train the model, we utilize the following loss function.

$$
\begin{aligned}
L = \rho \underbrace{\mathbb{E}_{x\sim\mathcal{D}_{\text{pretrain}}} \ln p(d(e(x)))}_{\text{autoencoding of pretraining dataset}} \\
+ \mathbb{E}_{(q,C,S,a)\sim\mathcal{D}_{\text{finetune}}} \Big[\beta \underbrace{\ln p(d(e(C)))}_{\text{autoencoding of contexts}} \\
+ \gamma \underbrace{\ln p(a|P_a, z_r^{(1)}, \ldots, z_r^{(|S|)})}_{\text{reconstruction of answer}} \\
+ \delta \underbrace{\sum_{t=1}^{|S|} \ell_{\text{order}}(z_r^{(t)}, S_t)}_{\text{ordering loss}} \Big]
\end{aligned}
$$

where $M = \text{write}(M_0, \{\tilde{z}_j\}_{j=1}^E)$, and $\forall\ t, z_r^{(t)} = \text{read}(M, z_q^{(t)}), z_q^{(t)} = z_q^{(t-1)} + \alpha z_r^{(t-1)}$. Following (Ko et al., 2024), $\ell_{\text{order}}(z_r, s)$ is defined by $-\ln v(z_r)_s$ where $v(z_r) = \text{softmax}([-\|z_r - \tilde{z}_1\|_2, ..., -\|z_r - \tilde{z}_E\|_2])$. ρ and β are the hyperparameters controlling regularization strength. During staged training, we let $\gamma = 0$ for the first 50 epochs. When supporting fact supervision is not available, we put $\delta = 0$.

In the overall loss L, the first and second terms correspond to the auto-encoding loss on the pretraining dataset and the contexts. The third term is the reconstruction loss of the answer with respect to the corresponding prompt for obtaining the answer P_a and final readout. The fourth term is an optional ordering loss when the supporting fact is available. As we will show in the experiments, by adopting proper staged training and performing inference-time update, ImReasoner trained under weak supervision (only answer supervision) can yield competitive performance compared to MemReasoner trained under full supervisions (answer + supporting fact supervision). Moreover, we show that depending on the availability of supporting fact supervision, we can further improve the performance and yield a significant margin over MemReasoner with the same level of supervision.

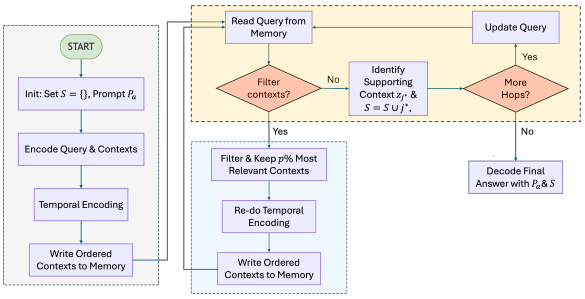


Figure 2: ImReasoner inference algorithm flowchart.

2.3.4 Inference algorithms and complexity analysis

Consider an input context $C = \{c_1, ..., c_E\}$, a question q, an encoder e, a temporal encoding module $\mathcal{P}$, an initial memory module $\mathcal{M}$, and a decoder d. As shown in the flowchart 2, we first encode the context C and query q to their latents, $z_1, \ldots, z_E$ and z_q, via encoder e. Then, we follow Section 2.1 to perform temporal encoding and transform $z_1, \ldots, z_E$ to $\tilde{z}_1, ..., \tilde{z}_E$. Next, we write the ordered context $\tilde{z}_1, ..., \tilde{z}_E$ to the memory and obtain $\hat{M}$. Subsequently, we read the query latent z_q from the memory $\hat{M}$ and obtain z_r. Next, we follow Section 2.3.1 and filter out the most irrelevant contexts by their distance to z_r. With the remaining subset of contexts $C' \subset C$, we re-transform them to ordered contexts for final memory write and query read. After the second query read, we will proceed to query update described in Section 2.1 and continue to the second hop. Lastly, the decoder d decodes the prompt P_a given for answer generation conditioned on all retrieved supporting facts. We provide the full pseudocode in Algorithm 1 in the Appendix B.

We follow (Ko et al., 2024) notation to estimate the complexity as follows: Let H_1 and d_1 be the

number of transformer layers and hidden state dimension in the encoder, H_2 and d_2 be the number of transformer layers and hidden state dimension in the decoder, d be the latent space dimension, m be the memory size, E be the number of context lines in a sample, L be the maximum context length, and L_1 be the maximum query length. The inference-time computational complexity for ImReasoner can be estimated by the encoder complexity $\mathcal{O}(H_1((EL^2 + L_1^2)d_1 + (EL + L_1)d_1^2))$, pre-filter temporal encoding complexity $\mathcal{O}(Ed^2)$, pre-filter memory operation complexity $\mathcal{O}(Edm^2)$, post-filter temporal encoding complexity $\mathcal{O}(pEd^2)$, post-filter memory operation complexity $\mathcal{O}(pEdm^2)$, decoding complexity $\mathcal{O}(H_2(|P_a|^2 d_2 + |P_a| d_2^2))$, and broadcasting complexity $\mathcal{O}(d_1 dE)$ and $\mathcal{O}(d_2 dH_2)$. Overall, by assuming $P_a \sim L_1$ and $p < 1$, we estimate the total inference-time complexity as $\mathcal{O}(H_1((EL^2 + L_1^2)d_1 + (EL + L_1)d_1^2) + E(d^2 + dm^2) + H_2(L_1^2 d_2 + L_1 d_2^2) + d_1 dE + d_2 dH_2)$.

3 Experiment

3.1 Experimental Setup

Tasks. Following literature (Weston et al., 2015; Kuratov et al., 2024), we are dedicated to stress-test LMs in (long-context) temporal-aware fact chaining. Specifically, to distinguish from (Ko et al., 2024), we will particularly distinguish the level of supervision each model received to analyze if there exists emergent reasoning generalization or pattern matching.

We first resort to the *bAbi* benchmark (Weston et al., 2015) (CC BY 3.0), which were prepared by synthesizing relations among characters and objects across various locations. This benchmark serves as a unit test for LM's temporal-aware fact chaining ability. Each context associated with an example contains multiple line, with each line representing a fact such as "Mary traveled to the garden". Task 1 requires performing a single hop to find the answer, whereas task 2 requires gathering two supporting facts in the right order. For preprocessing bAbi data, we follow literature and treat each training sample comprised of multiple facts as a single context episode, and individual sentence within that context as an instance within that episode. Each fact within an episode contains up to 64 tokens.

Two other variants of the bAbI benchmark were proposed in (Ko et al., 2024) where (1) the location information is changed to another set of locations in the test set, and (2) the test set is swapped across tasks, for example swapping 2-hop tasks' testing data to 1-hop, challenging models' generalization ability from harder tasks to easier tasks. We leave results on these two variants in the Appendix D

Then, we extend the test to long-context scenarios via *BABILong* (Kuratov et al., 2024) (Apache 2.0), which was constructed following bAbI but was artificially prolonged by fusion with irrelevant texts. Specifically, BABILong offers a wide variety of long-context problems of different lengths ranging from 0K (similar to bAbI) to 10M by adding background texts from PG-19 (Rae et al., 2020) (Apache 2.0). Following literature, if sentences are longer than 64 tokens in BABILong, we split the sentences at multiples of 64 tokens.

Finally, we level up and create another version of BABILong with in-distribution distractions as opposed to out-of-distribution distractions (e.g. PG-19), we dub it by *BABILong-soft*. Specifically, BABILong-soft contains distractions of the form: {entity} {verb} to the {location} where the entity is one of "John", "Mary", "Sandra", and "Daniel", verb is one of "moved", "went", "journeyed", "traveled", "went back", and location is one of "bedroom", "bathroom", "kitchen", "garden", "office", and "hallway". Notably, sentences of this form naturally occur in the original bAbI dataset. Therefore, this constitutes a much harder needle-in-a-haystack task, as the haystack is now much similar to the needle. Using this padding, we increase the context length up to 4k tokens. We provide more details about how this data is generated and examples in Appendix C.

Baselines. In literature (Kuratov et al., 2024; Ko et al., 2024), authors have reported performance of more than 30 off-the-shelf baselines with few-shot and chain-of-thought prompting, including popular ones such as Llama, Phi, Qwen, and GPT4 etc. Prior work found that these models effectively only use 10–20% of the context and performance degrades sharply with increased reasoning complexity. Even with Retrieval-Augmented Generation (RAG) methods, only around 60% accuracy can be achieved on single-fact question answering, and they also do not scale well for more complex multi-hop tasks across long contexts. Comparatively, they fine-tuned memory-augmented architectures (e.g. recurrent memory transformers, Mamba,

Table 1: The performance of ImReasoner and baselines on the bAbI test set, measured by accuracy (%).

Model type	Task 1	Task 2
RMT-.77B (bAbI)	99	100
RMT-.77B (bAbI w. 100% SF)	100	100
Mamba-1.4B (bAbI)	100	91
Mamba-1.4B (bAbI w. 100% SF)	100	93
MemReasoner-1.4B (bAbI)	100	39.5
MemReasoner-1.4B (bAbI w. 100% SF)	100	100
ImReasoner-1.4B (bAbI)	100	90
ImReasoner-1.4B (bAbI w. 100% SF)	100	100

MemReasoners) on bAbI/BABILong and showed that these fine-tuned models, albeit being small, achieve much stronger performance and can process longer sequences

Thus, in this paper, we primarily benchmark these finalists from earlier work and scrutinize their performance under different levels of supervision, in comparison with our proposed augmented version ImReasoner. Specially, we report RMT and Mamba models fine-tuned on (1) BABILong (supporting fact indices not available), (2) bAbI without supporting fact, (3) bAbI with supporting fact. For MemReasoner, we test two scenarios, (1) bAbI without supporting fact, (2) bAbI with supporting fact. For ImReasoner, we carefully go over 3 different levels of supervision, (1) bAbI without supporting fact, (2) bAbI with partial supporting fact (1%), and (3) bAbI with supporting fact. It is important to note that, due to the architecture of ImReasoner and MemReasoner, it is straightforward to enforce retrieval accuracy by imposing a categorical loss. On RMT and Mamba models, we could only naively use the next token prediction loss to encourage supporting fact reconstruction. More work on how to introduce more natural supervision loss is beyond the scope of this paper and left as a future work. In all of the above experiments, we quote the original reported numbers from the respective paper whenever available.

Due to the page limit, we refer readers to the Appendix E for more experimental details and Appendix F for the ablation study. We implemented our algorithm and conducted experiments based on Larimar (Das et al., 2024) code base[1].

3.2 Results

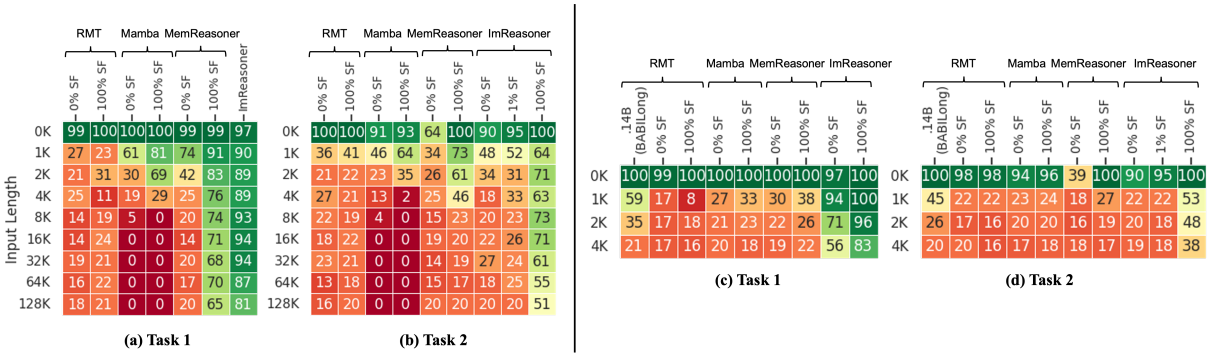
3.2.1 bAbI

Table 1 reports the performance of all baseline methods on bAbI task 1 and task 2 test sets. We note that task 1 and task 2 represent single-hop and two-hop question answering problems, respectively. From the table, we can see that all methods, when finetuned with supporting facts, can reach almost perfect accuracy (93%-100%). When such supervision is not available, RMT's still retains remarkable performance (100%), while Mamba and ImReasoner drops to 90%-91%. It is noteworthy that ImReasoner and MemReasoner, despite sharing the same architecture, have distinct performance with MemReasoner lagging behind by a large margin (39.5%), highlighting the significance of our staged training and inference time improvement.

3.2.2 BABILong

Figure 3 (left) reports the accuracy of all baselines on BABILong of varying lengths from 0K to 128K. We stop at 128K due to computational constraint. From Fig. 3(a-b) , we see that bAbI-finetuned RMT and Mamba models show a significant accuracy drop on BABILong samples beyond 0k input length, though both show near-perfect accuracy at 0k. Interestingly, additional supervision on 100% supporting facts during RMT or Mamba training (in the form of supporting fact reconstruction loss) did provide none–to–little performance boost. In contrast, MemReasoner benefits a lot from supporting fact supervision with its task 1 performance moves from "red zone" to "green zone". Similar improvement can be seen on task 2 as well, despite overall much inferior performance compared to ImReasoner with 100% supporting fact. Taking a closer look at ImReasoner's performance, we see that ImReasoner can achieve fair to excellent performance on task 1 (81%-97%) even without supporting fact supervision. On task 2, as the learning dynamics become much more uncertain without supporting fact, we see a big drop in performance, while still wins over MemReasoner with the same level of supervision by a large margin, e.g. without supporting fact supervision we see 90% vs 64% at 0K tasks; with supporting fact supervision we see 73% vs 23% at 8K tasks. Besides quantitative results, we also include a qualitative error analysis in the Appendix G.

[1] https://github.com/IBM/larimar



(a) Task 1 (b) Task 2 (c) Task 1 (d) Task 2

Figure 3: The performance of ImReasoner and all baselines on varying lengths of BABILong (left) and BABILong-soft (right) data, measured by accuracy (%). Plot (a) only gives ImReasoner with 0% supporting fact supervision as it already win over other baselines by a noticeable margin.

3.2.3 BABILong-soft

Although not plotted in Fig. 3, it is not hard to find that RMT and Mamba, when fine-tuned on BABILong, can achieve extraordinary performance on BABILong. For instance, (Kuratov et al., 2024) reported that RMT after fine-tuning can retain 59% average accuracy over five tasks at 128k length in BABILong while Mamba achieve astonishing 93%. In this experiment, we argue that this almost perfect performance is not due to reasoning generalization since the same performance cannot be reached when we present in-distribution distractions (at least for RMT; Mamba BABILong fine-tuned checkpoint is not publicly-available). Specifically, we show the performance of all baselines fine-tuned on bAbI with supporting facts in Figure 3 (right). By looking at Fig. 3(c-d), we observe a distinct difference between single-hop and multi-hop performance. As shown in Fig. 3(c), ImReasoner obtains near-perfect accuracy on task 1 across context lengths, while RMT fine-tuned on BABILong degrades from 59% at 1k tokens to 21% at 4k. In contrast, as shown in Fig. 3(d), ImReasoner's accuracy on task 2 drops substantially once the context length exceeds 1k, although it still maintains a margin over other baselines. This suggests that our inference-time update does bring benefit for the task, but only partially addresses the challenges of multi-hop reasoning in the presence of in-distribution distractors.

We hypothesize that this gap reflects a fundamental difference between learning single-hop and multi-hop tasks from answer-only supervision. In the multi-hop setting, the model must select an entire sequence of supporting facts, yet the loss only constrains the final answer. As a result, the training dynamics admit many spurious reasoning paths, and the model is incentivized to exploit dataset-specific shortcuts rather than to identify the true supporting chain, in line with the representation collapse and shortcut behavior we analyze in Section 2.3.2. Although staged training improves the diversity of context latents and stabilizes learning, it does not fully resolve the ambiguity of path selection. In BABILong-soft, where distractors share the same template as true facts, this ambiguity is amplified and leads to compounding errors across hops.

4 Related Literature

4.1 Long-range Dependency

Modeling long-range dependencies remains a fundamental challenge for transformer-based language models. Many tasks, including multi-hop logical reasoning and compositional mathematical derivations, require processing information that is scattered across extended contexts and executing reasoning steps in the correct order. Standard transformers often struggle in such settings due to several factors. First, they tend to exploit dataset-specific shortcuts rather than learning general reasoning strategies (Ju et al., 2024; Ruder, 2021; Mitchell, 2023; Wu et al., 2024; Levy et al., 2024). Second, they exhibit fragile internal mechanisms like attention glitches that may misroute information (Liu et al., 2023a; Xiao et al., 2024; Liu et al., 2023c). Such limitations have been connected to the "System 1" cognitive mode (Kahneman, 2011), characterized by fast but shallow inference, contrasting with the deliberative "System 2" reasoning required for multi-step tasks.

4.2 Benchmarking Long-range Dependency

Evaluating reasoning capabilities in long-context settings has become an essential part of understanding and advancing language model architectures. Synthetic benchmarks in particular have emerged as powerful diagnostic tools because they provide controlled conditions that isolate specific reasoning challenges without the confounds of real-world data. Early benchmarks often focused on "needle-in-a-haystack" retrieval tasks, where a single relevant fact must be located amid large amounts of distractor text (Li et al., 2024; Hsieh et al., 2024). Recent efforts have shifted toward evaluating multi-step compositional reasoning, where multiple scattered pieces of evidence must be integrated to reach a correct conclusion (Kuratov et al., 2024; Hsieh et al., 2024). These benchmarks reveal that even state-of-the-art transformer models degrade sharply as context length increases, highlighting persistent weaknesses in long-term memory, reasoning depth, and multi-hop compositionality. As a result, they have become standard testing grounds for both transformer-based LLMs and emerging alternatives such as state-space models (Gu and Dao, 2023).

4.3 Alternative Frameworks/Models

Recent works have attempted to include few-shot demonstrations in the context (Brown et al., 2020; Min et al., 2022) and chain of thought (CoT) prompting (Wei et al., 2022), providing access to external tools/reward models/verifiers (Schick et al., 2023; Khalifa et al., 2023), etc. Augmenting language models with memory modules has been proposed, e.g, in (Nye et al., 2021) the model is asked to output immediate reasoning steps to a "scratchpad" which is then recurrently processed by the model. Another promising research direction is to train the transformer model with an external latent memory module (Das et al., 2024) or with additional learnable memory tokens (Burtsev et al., 2021). Later, architectures that include segment-level recurrent processing over internal memory tokens have been proposed, e.g., Transformer-XL (Dai et al., 2019) and Recurrent Memory Transformer (RMT) (Bulatov et al., 2022). This line of work has shown the ability to process very long input and has emerged as a promising path for modeling long-term dependencies and exploiting memory processing ability for tasks like algorithmic and reasoning. Structured state space models such as Mamba (Gu and Dao, 2023) have emerged as a promising alternative to self-attention layers and transformers for sequence modeling due to its selection mechanism. Mamba offers faster inference due to its fixed-memory recurrent architecture, which allows for efficient processing of long sequences. However, this constant-memory also can make the in-context recall ability brittle, compared to transformers (Jelassi et al., 2024; Waleffe et al., 2024; Park et al., 2024). Recently, MemReasoner (Ko et al., 2024) is also proposed that does segment-level processing in the latent memory module, which is further augmented with a recency awareness and iterative read mechanism. Different from recurrent passing of the global memory tokens from the previous segment to the next segment within the transformer layers themselves, MemReasoner performs multiple hops over the "ordered" segment encodings stored in memory, updates the query, and provides only the final readout(s) to the decoder. MemReasoner does not maintain a memory of explicit (generated) tokens, rather operates over the latent encodings of context stored in memory. However, as we discussed in this paper, its performance largely comes from the supporting fact supervision, without which it in fact finds it hard to generalize.

5 Conclusion

This work explored how structured inductive biases can enhance language models' ability to perform multi-step reasoning in long and noisy contexts. We introduced ImReasoner, a memory-augmented framework that combines an inference-time memory update mechanism with a staged training strategy to improve retrieval precision and maintain robust latent representations under weak supervision. Across diverse benchmarks, ImReasoner consistently outperforms prior memory-based approaches, particularly when intermediate supervision is scarce, demonstrating the effectiveness of selective context filtering and representation-centric training. These findings highlight the importance of integrating adaptive memory operations and principled training dynamics, paving the way for more generalizable reasoning systems in complex, real-world settings.

Limitations

While ImReasoner demonstrates strong performance gains in reasoning-in-a-haystack tasks and shows resilience under weak supervision, several

limitations remain.

First, while we demonstrate competitive results under weak supervision, ImReasoner's performance still benefits significantly from supporting fact annotations when available. This reliance suggests that the model has not yet achieved full reasoning generalization and that supervision signals remain important for guiding multi-hop inference.

Secondly, as our inference-time update procedure was designed specifically for memory-augmented architectures, the underlying principle of selective context filtering can, in theory, be applied to more generic language models as well. However, how to effectively integrate such mechanisms into transformer models remain an open question.

Lastly, as is also true for literature on the topic (Kuratov et al., 2024; Ko et al., 2024), our current experiments are primarily conducted on synthetic benchmarks such as bAbI and BABILong, and our new benchmark to further stress test the models, BABILong-soft, is also synthetic. Although these benchmarks provide controlled and interpretable settings to evaluate reasoning behavior, they may not fully capture the complexity, ambiguity, and domain diversity of real-world tasks. In a way, we argue that being able to solve this set of synthetic tasks is necessary for all language models, but it might not be sufficient. We hope our work can inspire more research in alternative frameworks other than transformers.

Ethics Statement

Given the rapid progress in large language model research, it is important to review alternative frameworks that use fewer computational resources and understand their capability. We do not anticipate any negative social impact from this work.

References

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2019. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2024. Titans: Learning to memorize at test time. *Preprint*, arXiv:2501.00663.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.

Mikhail S. Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V. Sapunov. 2021. Memory transformer. *Preprint*, arXiv:2006.11527.

James P Byrnes. 1992. The conceptual basis of procedural learning. *Cognitive Development*, 7(2):235–257.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *Preprint*, arXiv:1901.02860.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarath Swaminathan, Sihui Dai, Aurélie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiří, Navrátil, Soham Dan, and Pin-Yu Chen. 2024. Larimar: Large language models with episodic memory control. *Preprint*, arXiv:2403.11901.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*, volume 1. MIT press Cambridge.

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

Arian Hosseini, Alessandro Sordoni, Daniel Toyama, Aaron Courville, and Rishabh Agarwal. 2024. Not all llm reasoners are created equal. *Preprint*, arXiv:2410.01748.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *Preprint*, arXiv:2404.06654.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. 2024. Repeat after me: Transformers are better than state space models at copying. *Preprint*, arXiv:2402.01032.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. 2024. Investigating multi-hop factual shortcuts in knowledge editing of large language models. *Preprint*, arXiv:2402.11900.

Daniel Kahneman. 2011. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Grace: Discriminator-guided chain-of-thought reasoning. *Preprint*, arXiv:2305.14934.

Ching-Yun Ko, Sihui Dai, Payel Das, Georgios Kollias, Subhajit Chaudhury, and Aurelie Lozano. 2024. Memreasoner: A memory-augmented llm architecture for multi-hop reasoning. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *arXiv preprint arXiv:2406.10149*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *Preprint*, arXiv:2402.14848.

Mo Li, Songyang Zhang, Yunxin Liu, and Kai Chen. 2024. Needlebench: Can llms do retrieval and reasoning in 1 million context window? *Preprint*, arXiv:2407.11963.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023a. Exposing attention glitches with flip-flop language modeling. *Preprint*, arXiv:2306.00946.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023b. Transformers learn shortcuts to automata. *Preprint*, arXiv:2210.10749.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023c. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *Preprint*, arXiv:2202.12837.

Melanie Mitchell. 2023. How do we know how smart ai systems are?

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *Preprint*, arXiv:2112.00114.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. 2024. Can mamba learn how to learn? a comparative study on in-context learning tasks. *Preprint*, arXiv:2402.04248.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, and 13 others. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore. Association for Computational Linguistics.

Michael Pressley and Peter Afflerbach. 1995. *Verbal Protocols of Reading: The Nature of Constructively Responsive Reading*. Routledge.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Marc'Aurelio Ranzato and Martin Szummer. 2008. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799.

Liliang Ren, Yang Liu, Yadong Lu, yelong shen, Chen Liang, and Weizhu Chen. 2025. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *The Thirteenth International Conference on Learning Representations*.

Bethany Rittle-Johnson, Robert S Siegler, and Martha Wagner Alibali. 2001. Developing conceptual understanding and procedural skill in mathematics: An iterative process. *Journal of educational psychology*, 93(2):346.

Sebastian Ruder. 2021. Challenges and opportunities in nlp benchmarking.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Nadav Schneider, Itamar Zimerman, and Eliya Nachmani. 2025. Differential mamba. *arXiv preprint arXiv:2507.06204*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro.

2024. An empirical study of mamba-based language models. *Preprint*, arXiv:2406.07887.

Yuxuan Wan, Wenxuan Wang, Yiliu Yang, Youliang Yuan, Jen tse Huang, Pinjia He, Wenxiang Jiao, and Michael R. Lyu. 2024. Logicasker: Evaluating and improving the logical reasoning ability of large language models. *Preprint*, arXiv:2401.00757.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Jian Wu, Linyi Yang, Zhen Wang, Manabu Okumura, and Yue Zhang. 2024. Cofca: A step-wise counterfactual multi-hop qa benchmark. *Preprint*, arXiv:2402.11924.

Yan Wu, Greg Wayne, Alex Graves, and Timothy Lillicrap. 2018. The kanerva machine: A generative distributed memory. *Preprint*, arXiv:1804.01756.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

A How human reason do reasoning? Remove irrelevant facts

Question: Who will be impacted by the loss of biodiversity in the Amazon?

~~The Amazon rainforest spans over five and a half million square kilometers in South America.~~ It is home to more than 10% of the world's known species of plants and animals. ~~Scientists warn that deforestation in the Amazon has accelerated over the past decade due to agricultural expansion.~~ Many indigenous communities depend on the Amazon for their livelihood and cultural traditions. The loss of biodiversity in the Amazon could disrupt global climate regulation and ecosystem balance. ~~Trees in the Amazon produce roughly 20% of ....~~

Figure 4: An example of human removing apparently unrelated details after the first pass in a question answering / reading comprehension task.

B Algorithm

Algorithm 1: ImReasoner inference algorithm

Function ImReasoner($q, \{c_1, \ldots, c_E\}, \alpha, p$):

    // q: query; $\{c_i\}$: E context lines; α: query-update combine parameter;
    // N: #iterations; p: filter ratio

    $S \leftarrow set\{\}$

    $P_a \leftarrow$ prompt for answer generation given q and task specification

    $z_q \leftarrow \mathsf{encode}(q)$

    for $i \leftarrow 1$ **to** E **do**

        $z_i \leftarrow \mathsf{encode}(c_i)$

    $\tilde{z}_1, \ldots, \tilde{z}_E \leftarrow \mathsf{temporalEncoding}(z_1, \ldots, z_E)$

    $\hat{M} \leftarrow \mathsf{write}(\tilde{z}_1, \ldots, \tilde{z}_E)$

    for $n \leftarrow 1$ **to** N **do**

        // first pass to filter out most irrelevant contexts

        $z_r \leftarrow \mathsf{read}(\hat{M}, z_q)$

        $\mathcal{J} \leftarrow$ indices of the $p\%$ smallest distances $\{\|z_r - \tilde{z}_j\|_2\}_{j=1}^{E}$

        $(\tilde{z}'_j)_{j\in\mathcal{J}} \leftarrow \mathsf{temporalEncoding}\big((z_j)_{j\in\mathcal{J}}\big)$

        $\hat{M}' \leftarrow \mathsf{write}((\tilde{z}'_j)_{j\in\mathcal{J}})$

        // second pass to find supporting facts

        $z_r^{(n)} \leftarrow \mathsf{read}(\hat{M}', z_q)$

        $z_q \leftarrow z_q + \alpha\, z_r^{(n)}$

        $S \leftarrow S \cup \{j^* : j^* = \arg\min_{j\in\mathcal{J}} \|z_r^{(n)} - \tilde{z}'_j\|_2\}$

    return decode $(P_a, \{z_j\}_{j\in S})$

C Generation of BABILong-soft data

To generate BABILong-soft data, we begin with the bAbI test set for tasks 1 and 2. We pad the data with sentences of the form:

```
{entity} {verb} to the {location}.
```

where the entity is one of “John", “Mary", “Sandra", and “Daniel", verb is one of “moved", “went", “journeyed", “travelled", “went back", and location is one of “bedroom", “bathroom", “kitchen", “garden", “office", and “hallway". Sentences of this form naturally occur in the bAbI train and test set for these tasks. The location of the padding is randomly sampled and thus this padding can occur before, between, and after the true sentences in the context of the bAbI dataset. In order to avoid changing the answer to the bAbI question when adding padding, we keep track of the location of the last supporting fact and ensure that padding added after the last supporting fact does not mention the entity that is in the supporting fact. Meanwhile, padding added before the last supporting fact can be have any entity.

To illustrate, consider a bAbI sample with the question “Where is the football?" and context:

```
Mary travelled to the office.
Sandra went to the bedroom.
Sandra moved to the hallway.
Daniel journeyed to the garden.
Sandra discarded the football.
Daniel went back to the kitchen
```

Here, the last supporting fact in the line “Sandra moved to the hallway." and this line tells us the answer of the question is “hallway". To generate a corresponding BABILong-soft data point, we randomly sample a location at which we insert a sentence of padding. If this line is to be added before “Sandra moved to the hallway." then we sample any entity of the bAbI entities for the padding. If it is to be added after “Sandra moved to the hallway.", then we sample an entity that is not Sandra.

We use this approach to generate padding to increase context lengths to 1k, 2k, and 4k tokens.

D Two bAbI variants

We follow (Ko et al., 2024) and test on a variant of bAbI where the tasks remains the same, but the answer changes from training to test set. Specifically, the dataset changes the location information present in the answer set of bAbI training → test as follows: office → library, garden→ garage, kitchen→ cafe, bathroom → attic, bedroom→ basement, hallway → gym.

We report baselines' results in Table 2. Whenever the results are available in (Ko et al., 2024), we directly cite their numbers and note by "*". Thus the numbers might not be utmost comparable since the dataset generation has randomness and we are be using the exact same dataset for measurement. As shown in Table 2, when no SF supervision is used, on task 1 the accuracy order is RMT $<$ Mamba $<$ ImReasoner, while on task 2 the order is RMT $<$ ImReasoner $<$ Mamba. Given that, in this setting the length of the test sequence is similar to that of the training one, Mamba handles it better for task 2.

Table 2: Robustness to location changes in bAbI test set. Bold: highest; underlined; second highest.

Model type	Task 1	Task 2
RMT*	45.8	23.7
RMT + 100%SF	31.3	0
Mamba*	67	44
Mamba + 100%SF	52.7	45.2
MemReasoner + 100%SF*	<u>87.2</u>	<u>52.7</u>
ImReasoner	**98.6**	30.4
ImReasoner + 100% SF	-	**63.9**

We also follow the literature and check if the models trained on 2-hop bAbI can solve the simpler 1-hop version, but on the corresponding long context samples. We again cite baselines results whenever available in Table 3. From the table, it can be seen that Mamba can benefit a lot from having supporting fact supervision on this task, while the same improvement can hardly be seen on RMT. On ImReasoner, supporting fact supervision helps to promote ImReasoner to get the second place for 2K and 4K tasks. It is worth noting that, without supporting fact supervision, ImReasoner is a strong candidate performing better than RMT and Mamba when context >1k.

Table 3: Performance on bAbI task 2 → BABILong task 1 generalization.

Model type	0k	1k	2k	4k
RMT*	**100**	13	11	13
RMT (bAbI) + 100%SF	95	21	22	16
Mamba*	81	8	0	0
Mamba + 100%SF	<u>99</u>	<u>55</u>	27	9
MemReasoner + 100%SF*	83	**58**	**50**	**45**
ImReasoner	49	30	28	19
ImReasoner + 100% SF	56	44	<u>43</u>	<u>38</u>

E Training Details

Following (Ko et al., 2024), we use Larimar as our backbone for ImReasoner using publicly available codebase[2]. Specifically, we use the default script to train a 1.3B Larimar model. We generate the answer to the question by passing a prompt P_a to the decoder (i.e. in the case of bAbI Task1-2, the prompt has the from "<BOS> X is in the" where X denotes subject of the query q).

Task 1 and 2 each contains 10K training samples and 1K testing samples. For task 1, We train ImReasoner models for 20 epochs using Adam optimizer with learning rate 5e-6. We set batch size to be 20. For task 2, we decrease the batch size to 5 due to computational constraints. Throughout the experiments, we set hyperparameters $\alpha = \beta = \gamma = 1$. Since bAbI Task 1 is a single hop task, we do not perform query update during either training or inference. When fine-tuning on bAbI Task 2, we perform a fix number of 2 hop (equivalent to 1 query update) during the training. All our reported results are gathered from a single run.

F Ablation study

As the two key innovations of this paper are: 1. inference-time update method, and 2. staged training. In this section, we give the ablation study of these two components individually to show their significance. Specifically, as ImReasoner = MemReasoner + "inference-time update" + "staged training", we use BABILong task 1 as an example to show: (1) MemReasoner, (2) MemReasoner + "inference-time update" = ImReasoner - "staged training", (3) and ImReasoner, with $(1) \rightarrow (2)$ highlights the contribution from our inference-time update stratefy, and $(2) \rightarrow (3)$ highlights the contribution from our staged training strategy.

From Table 4, we see that the introduction of inference-time update strategy largely improves the performance, saving the accuracy from 14% to 82% for 16k-length tasks. With staged training, it furthers increase the accuracy from 82% to 94%, underscoring the contribution from both components.

Table 4: Ablation study on the inference-time update and staged training.

BABILong (k)	0	1	2	4	8	16	32	64	128
MemReasoner	99	74	42	25	20	14	20	17	20
ImReasoner w/o staged training	97	86	82	84	82	82	85	79	79
ImReasoner	97±1.9	90±3.3	89±3.1	89±2.5	93±3.1	94±3.2	94±2.0	87±3.9	81±2.7

We further vary the quantile p in our inference-time update to examine its effect on the downstream performance. Specifically, we give the accuracy under multiple grids, i.e. p =1 (no filtering -> no inference-time update), 0.5, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001. We note that in the most extreme case, when p is small enough to filter out all contexts but one or very few, the performance should be close to no inference-time update as there would only have one or few context left as supporting fact candidate (the same as directly picking one supporting fact in one-shot). From the following table, we can see that our strategy of p chooses p dynamically according to context lengths (i.e. p= $\frac{1}{4m}$), effectively encouraging the number of remaining contexts to be of the same range regardless of the initial lengths. If we are allowed to do finetuning on p, the accuracy can generally be further improved.

Table 5: Ablation stufy on the inference-time update hyperparameter.

p	proposed strategy: $\frac{1}{4m}$	1	0.5	0.2	0.1	0.05	0.02	0.01	0.005	0.002	0.001
1k (m=1)	86	75	82	**91**	83	70	-	-	-	-	-
4k (m=4)	84	21	43	67	75	**89**	69	52	-	-	-
16k (m=16)	82	14	23	24	45	65	82	**85**	71	47	-
64k (m=64)	**79**	12	16	19	21	32	50	64	**79**	78	59

[2]https://github.com/IBM/larimar

G Error analysis

Since BABILong does not provide groundtruth supporting fact information, it is practically hard for us to root cause the errors, e.g. wrong supporting fact retrieval at early steps, wrong supporting fact retrieval at later steps, identify confusing facts with the correct identity, identify confusing facts with wrong identities, identify completely irrelevant PG19 texts, correct supporting fact retrieval but wrong answering, supporting fact missing after inference-time update, etc. Nevertheless, we are able to identify and confirm certain types of errors by manually inspecting incorrect samples:

- **{identify confusing facts with the correct identity; wrong supporting fact retrieval at later steps}** In an example, the question was “Where is the milk¿‘ and the supporting fact for the question should be (inspected by us) “Daniel discarded the milk.“ and “Daniel journeyed to the bedroom.“. In this example, ImReasoner correctly identifies “Daniel discarded the milk.“ at the first hop, but wrongly retrieved “Daniel got the milk there.“ at the second hop, indicating it may not look for Daniel’s location when identifying the second supporting fact but was instead still tracing "milk".
- **{identify completely irrelevant PG19 texts; wrong supporting fact retrieval at early steps}** In an example, the question was “Where is the milk¿‘ and the supporting fact for the question should be (inspected by us) “John dropped the milk.“ and “John went back to the bathroom.“ In this example, ImReasoner was wrong from the first hop and identifies a PG19 line “If you’ll excuse the hint, that old thing...“ Specifically, we double-checked that “John dropped the milk.“ was still among the candidate supporting facts after the inference-time update in this case.
- **{identify completely irrelevant PG19 texts; wrong supporting fact retrieval at early steps}** In an example, the question was “Where is the apple¿‘ and the supporting fact for the question should be (inspected by us) “Sandra took the apple there.“ and “Sandra journeyed to the office.“ In this example, ImReasoner retrieved a PG19 line “The pillar bears the following inscription, which you may think“ at the first hop and identifies the correct (first) supporting fact “Sandra took the apple there.“ at the second hop.

The above errors generally cover all the errors types we saw during our manual inspections, which highlighted the difficulties in reasoning-in-a-haystack tasks and the increasing complexities in multihop scenarios.