Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Recursive Inference Machines

Mieszko Komisarczyk

mieszko.komisarczyk@tu-darmstadt.de

Technische Universität Darmstadt

Saurabh Mathur

Technische Universität Darmstadt

Maurice Kraus

Technische Universität Darmstadt

Sriraam Natarajan

The University of Texas at Dallas

Kristian Kersting

Technische Universität Darmstadt

Article

Keywords:

Posted Date: September 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10021738/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Recursive Inference Machines

Mieszko Komisarczyk[1*†], Saurabh Mathur[1*†], Maurice Kraus[1], Sriraam Natarajan[2], Kristian Kersting[1, 3, 4]

[1]Department of Computer Science, TU Darmstadt, Darmstadt, Germany.
[2]Department of Computer Science, University of Texas at Dallas, Richardson, Texas, USA.
[3]Hessian Center for Artificial Intelligence (hessian.ai), Darmstadt, Germany.
[4]German Research Center for AI (DFKI), Germany.

*Corresponding author(s). E-mail(s):
mieszko.komisarczyk@tu-darmstadt.de;
saurabh.mathur@tu-darmstadt.de;
†These authors contributed equally to this work.

Abstract

Neural reasoners such as Tiny Recursive Models (TRMs) solve complex problems by combining neural backbones with specialized inference schemes. Such inference schemes have been a central component of stochastic reasoning systems, where inference rules are applied to a stochastic model to derive answers to complex queries. In this work, we bridge these two paradigms by introducing Recursive Inference Machines (RIMs), a neural reasoning framework that explicitly incorporates recursive inference mechanisms inspired by classical inference engines. We show that TRMs can be expressed as an instance of RIMs, allowing us to extend them through a reweighting component, yielding better performance on challenging reasoning benchmarks, including ARC-AGI-1, ARC-AGI-2, and Sudoku Extreme. Furthermore, we demonstrate the versatility of the RIM framework across two tabular settings: we show that RIMs can improve reasoning under observational noise, outperforming TabPFN on noisy medical datasets, and that a RIM-based hybrid architecture reduces inference time by 35% while matching predictive performance.

1 Introduction

Neural reasoners have garnered considerable interest owing to their impressive performance on challenging reasoning benchmarks. These models solve reasoning tasks by exploiting patterns in their training data, allowing them to generate answers to problems for which exact symbolic reasoning is intractable [1]. However, this also limits them to problems of complexity similar to that of their training data; they struggle to generalize to problems requiring longer horizons. Moreover, learning models to solve more complex problems requires deeper architectures. These limitations are particularly acute in scientific and medical domains, where models must perform complex reasoning over noisy observational data while meeting strict efficiency requirements at deployment time.

To mitigate these limitations, recent work has explored test-time scaling approaches that dynamically allocate additional computation during inference. Techniques such as chain-of-thought prompting [2], self-verification [3], and latent reasoning [4, 5] allow a fixed neural backbone to perform multi-step reasoning, leading to improved performance on long-horizon tasks. Despite their empirical success, these methods are largely introduced as heuristic procedures, lacking a unifying formal framework that explains why they work or how they can be systematically composed and extended. Answering complex queries by reasoning about correlations between patterns has long been studied in probabilistic inference. Approximate inference methods such as belief propagation [6] and Gibbs sampling [7] decompose a long-horizon reasoning task into a sequence of smaller computational blocks; each block is learned or adapted separately.

We argue that modern neural reasoners can be understood through a similar lens. Specifically, we hypothesize that many such procedures can be naturally expressed as modular programs, where neural components implement local inference operators and control flow specifies their recursive interaction. This perspective provides a unified semantics for existing methods, while enabling modularity, compositionality, and reuse across reasoning systems.

To this end, we introduce Recursive Inference Machines (RIMs), a framework for unified neural reasoning. RIMs make test-time reasoning explicit, allowing complex reasoning behavior to emerge from repeated applications of simple, reusable modules. By treating the **Solver**, **Generator**, and **Reweighter** as interchangeable components, RIMs define a modular design space that can be navigated to address different reasoning challenges.

Specifically, we make the following key contributions:

- We present RIMs as a general framework for neural reasoning architectures.
- We empirically demonstrate that RIMs outperform Tiny Recursive Models (TRMs, Jolicoeur-Martineau [5]) on challenging symbolic reasoning benchmarks such as Sudoku Extreme and ARC.
- We show that RIMs can improve reasoning under observational noise in tabular data, outperforming TabPFN [8] on medical diagnosis benchmarks.
- We cast TabPFN itself as an incomplete RIM and complete it with a Hydra-based **Solver**, yielding a 35% reduction in inference time while matching predictive performance.

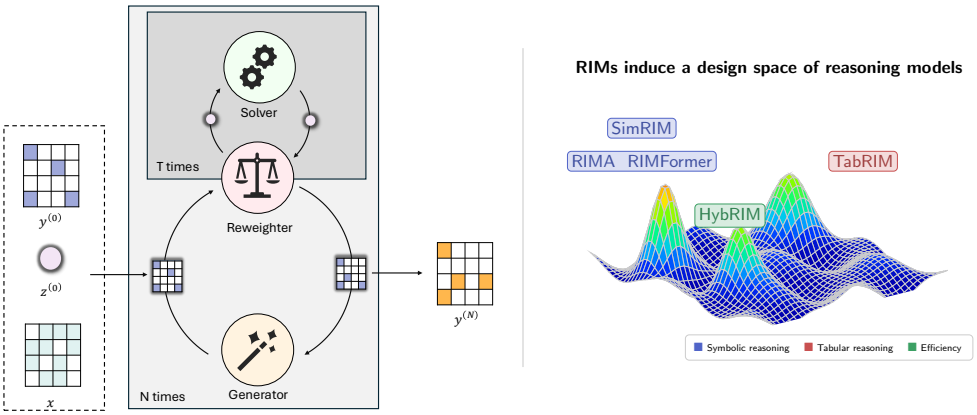


Figure 1: A **Recursive Inference Machine (RIM)** (left) solves problems described by x, given an initial solution $y^{(0)}$, and an initial state $(z^{(0)})$ combining a **Solver**, a **Reweighter**, and a **Generator**. The components solve the problem by alternating between the **Solver** updating the state recursively for T steps, and the **Generator** using these state-updates to generate an updated solution. After repeating this N times, the RIM produces the solution $y^{(N)}$. The symbolic specification via **Solver**, a **Reweighter**, and a **Generator** induces a design space of neural (reasoning) models (right). Labeled instantiations correspond to spots in different reasoning regimes explored in the present paper.

To this end, we proceed as follows. We start off with discussing related work and background. Then, we introduce RIMs and illustrate the induced design space on reasoning models, tabular data and efficient inference. Befor concluding we will present our experimental results.

2 Background

Recursive Inference Machines are related to different lines of research.

Neural Reasoning. Neural reasoning addresses reasoning tasks by training or adapting neural networks to perform multi-step inference directly from data. Unlike classical symbolic reasoners that rely on rule-based search, neural reasoners exploit learned latent representations to approximate reasoning behavior [9]. These approaches have achieved strong empirical performance on large-scale benchmarks where symbolic pipelines are often brittle, as seen in models like SATFormer, which learns effective heuristics for satisfiability problems [1].

Despite these successes, standard feedforward and Transformer architectures struggle with long-horizon reasoning due to their fixed computational depth. This constraint effectively limits the number of sequential steps in a single forward pass, preventing them from accurately solving tasks that require intricate algorithmic logic.

To overcome these depth constraints, recent work has focused on test-time scaling, allocating additional computation during inference via specialized schemes. These include chain-of-thought prompting [2] and iterative self-verification or refinement [10].

While such mechanisms allow for arbitrarily complex computations in theory [11], they often lack a systematic interpretation of their underlying "thinking" strategies, making the process difficult to improve.

Hierarchical Reasoning Model and Tiny Recursive Model. A more structured approach to iterative computation is found in recurrent architectures like Hierarchical Reasoning Models (HRMs,Wang et al. [4]) and Tiny Recursive Models (TRMs, Jolicoeur-Martineau [5]) that update latent states over multiple cycles.

HRMs employ two interdependent recurrent modules operating at different temporal scales. A low-level module executes rapid, local refinement until reaching a local equilibrium, at which point a high-level module provides a new abstract context to restart the process. While HRMs achieve competitive performance on benchmarks like ARC-AGI and Sudoku Extreme, parts of the method are motivated by arguments invoking the Implicit Function Theorem [12], whose underlying assumptions have been questioned in practical settings [5].

TRMs simplify the HRM architecture into a single recurrent unit using two nested loops. The latent states are interpreted more naturally: a short-term "scratchpad" z_L captures intermediate states, while z_H represents the evolving global solution [13]. By unrolling recursive steps and employing deep supervision, TRMs outperform significantly larger models with only 7M parameters [14].

Whether through dual-speed modules (HRM) or nested loops (TRM), these models implement a form of sequential refinement. However, they lack a systematic interpretation of their latent state updates, making their reasoning strategies difficult to interpret or improve.

State Space Models. While TRMs and HRMs achieve sequential refinement through recurrent architectures, alternative sequence modeling primitives offer different efficiency and inductive bias tradeoffs. State Space Models (SSMs) are derived from continuous-time dynamical systems, and have emerged as an effective approach to overcoming the limitations of both RNNs and Transformers. Compared to RNNs, SSMs mitigate the vanishing gradient problem through their structured state transition. Compared to Transformers, they achieve near-linear complexity in sequence length [15], avoiding the quadratic cost of attention. Mamba [16] is a prominent example, introducing a selective state space mechanism that adaptively filters information across the sequence. Hydra [17] extends Mamba to non-causal settings via quasiseparable matrix mixers, enabling bidirectional context aggregation while retaining Mamba's efficiency, a property important for tabular data where there is no natural sequence order. While Hybrid architectures interleaving SSM and Transformer layers have shown promise [18], navigating the tradeoff between the efficient state propagation of SSMs and the global reasoning capacity of Transformers remains an open question.

Probabilistic Inference. Reasoning with uncertain models has long been studied in the field of probabilistic inference. Probabilistic inference algorithms aim to answer queries, such as marginal statistics over unobserved variables given observed data. While exact inference is tractable for specialized structures like trees, it remains intractable for general probabilistic models, motivating several approximate inference methods [19, 20].

Approximate inference methods scale reasoning to high-dimensional spaces by employing iterative local computations to approach the true posterior distribution. In

this work, we focus on two such methods: Sequential Monte Carlo (SMC) and Gibbs Sampling. SMC represents the posterior as a set of weighted samples (particles) that evolve through a sequence of proposal and correction steps. A proposal distribution generates new candidate states, which are then adjusted by importance weights. This weighting mechanism is essential for correcting the bias introduced by the proposal. In Section 3, we show that recursive models like TRMs effectively implement the proposal but omit the weighting term, leading to sub-optimal reasoning trajectories. Gibbs sampling approximates complex posteriors by iteratively updating each variable conditioned on all others variables. This iterative process allows the sampler to explore the joint distribution by making local, conditional moves. We use this perspective to adapt tabular transformers to datasets with systematic noise.

Using these stochastic inference strategies, we aim to provide a principled foundation for neural reasoning by interpreting the neural reasoners' thinking steps as samples approximating a posterior. However, rather than using hand-crafted distributions, we follow the Inference Machine paradigm to learn these transitions directly from data.

Inference Machines. While stochastic inference provides a rigorous framework for reasoning under uncertainty, learning explicit generative models remains challenging in complex domains. The Inference Machine (IM) paradigm addresses this by forgoing the strict separation between models and inference methods [21]. Instead of learning a general-purpose model, this paradigm treats the neural architecture as a parameterization of reasoning, recasting it as a sequence of optimized predictions.

By merging the model and the inference algorithm into a single machine, we can learn to answer specific queries directly. For example, Message-Passing Inference Machines (MPIMs, [21]) replace the disconnected components of a graphical model and the belief propagation algorithm with a set of regression modules. These modules learn to perform inference by predicting messages based on local neighborhood context, thereby amortizing the cost of reasoning. Consequently, at test time, the model replaces expensive global optimizations with a fixed sequence of local operations compiled to navigate the specific manifolds of the target task.

We generalize this paradigm toward what we call Recursive Inference Machines (RIMs). We argue that architectures like TRMs are effectively approximate Inference Machines for SMC, parameterized with highly expressive neural backbones. Generalizing TRMs this way allows one to identify missing components, such as the importance **Reweighter**, and refine the architecture accordingly. Moreover, as we will demonstrate, the Inference Machine paradigm extends naturally to other algorithms and in-context learning. This enables one to adapt, for instance, a pretrained tabular transformer (TabPFN) within a RIM to realize a Gibbs sampler that learns to adapt to noisy domains.

3 Recursive Inference Machine

We now introduce Recursive Inference Machines (RIMs), a unified framework for neural reasoning. RIMs cast neural reasoners' latent state updates as a sequence of transitions defining a learned inference machine.

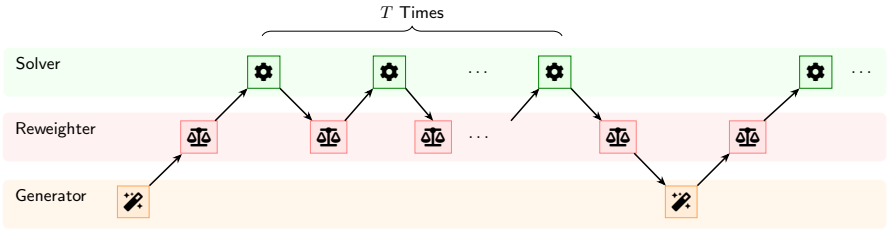


Figure 2: **Unrolling a Recursive Inference Machine (RIM).** The inner loop (indexed by $j = 1, \ldots, T$) acts as the **Solver**, which extends the short-term reasoning "scratchpad" z^j by proposing new states $\hat{z}^{j+1}$ based on the problem description x and current solution y^i. The outer loop (indexed by $i = 1, \ldots, N$) implements the **Generator**, which updates the solution y^i after each phase of local refinement. The **Reweighter** intercepts the updates proposed by the **Solver** and **Generator** and weighs them against their previous values, stabilizing the trajectory.

Definition 3.1 (Recursive Inference Machine) Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ denote the spaces of problem descriptions, solutions, and states, respectively, and let primed variables denote values at the previous step. We define a Recursive Inference Machine as the tuple $\langle S, G, \mathcal{R} \rangle$, where

- S is the *Solver*, a mapping from $(x, y, z') \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ to a candidate state update in $\mathcal{Z}$;
- G is the *Generator*, a mapping from $(x, y', z_{1:T}) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}^T$ to a candidate solution update in $\mathcal{Y}$;
- $\mathcal{R}$ is the *Reweighter*, a mapping from the histories $(y_{0:i}, z_{0:i})$ to the accepted state and solution in $\mathcal{Z} \times \mathcal{Y}$.

Definition 3.2 (RIM Execution) Given a RIM $\langle S, G, \mathcal{R} \rangle$, a problem description x, initial solution $y^{(0)}$, and initial state $z^{(0)}$, the execution of the RIM proceeds by alternating between recursive state updates and solution updates. The **Solver**, along with the **Reweighter**, recursively generates a sequence of T state updates. Each state update is a combination of the **Solver** proposing an update $\tilde{z}^{(i)}$ and the **Reweighter** weighing it against $z^{(0)}, \ldots, z^{(i-1)}$ to produce $z^{(i)}$. This results in state updates $z^{(1)}, \ldots, z^{(T)}$. The **Generator** uses this sequence to produce a candidate solution $\tilde{y}^{(1)}$, which the **Reweighter** processes to obtain the updated solution $y^{(1)}$. This entire process is repeated N times, yielding the final solution $y^{(N)}$.

Fig. 1 visualizes the general structure of a RIM. Fig. 2 visualizes its operation by unrolling it. Starting from $y^{(0)}$ and $z^{(0)}$, the **Solver** proposes a candidate updated state $\tilde{z}^{(1)}$, which is fed to the **Reweighter** to obtain $z^{(1)}$; this is repeated to generate a sequence of T states $z^{(1)}, \ldots, z^{(T)}$. The **Generator** uses this sequence to produce a candidate updated solution $\tilde{y}^{(1)}$, which the **Reweighter** processes to obtain $y^{(1)}$. Repeating this process N times yields $y^{(1)}, \ldots, y^{(N)}$; the last solution is returned as the output.

This unrolling process is a formal generalization of Sequential Monte Carlo (SMC) in reasoning space. A RIM generates a trajectory by starting with $y^{(0)}, z^{(0)}$, extending the state through T steps of the **Solver**, and then resampling a new solution $y^{(1)}$ via the **Generator** and **Reweighter**. Concretely, consider a reasoning task with problem

description x. We define a joint target distribution over a sequence of paired reasoning states and solution estimates:

$$p\big((z^1, y^1), \ldots, (z^M, y^M) \mid x\big)\,, \tag{1}$$

where each (z^i, y^i) represents the latent reasoning state and solution estimate at step i, and $M = N \cdot T$ is the total number of steps across the outer and inner loops. An SMC algorithm targeting (1) proceeds by maintaining a population of K weighted particles, each representing a candidate reasoning trajectory. At each step i, it draws proposals from

$$z^i \sim q\big(z^i \mid z^{i-1}, y^{i-1}, x\big)\,, \tag{2}$$
$$y^i \sim q\big(y^i \mid y^{i-1}, z^i\big)\,, \tag{3}$$

and corrects for proposal bias via the incremental importance weight

$$w^{(i)} \propto \frac{p\big((z^1, y^1), \ldots, (z^i, y^i) \mid x\big)}{\prod_{j=1}^{i} q(z^j \mid z^{j-1}, y^{j-1}, x)\, q(y^j \mid y^{j-1}, z^j)}. \tag{4}$$

This SMC algorithm is an instance of a RIM $\langle S, G, \mathcal{R}\rangle$ under the following conditions: the **Solver** S implements the state proposal $q(z^i \mid z^{i-1}, y^{i-1}, x)$; the **Generator** G implements the solution proposal $q(y^i \mid y^{i-1}, z^i)$; and the **Reweighter** $\mathcal{R}$ computes the importance weight $w^{(i)}$ in (4), correcting for the bias introduced by the proposals. While it is possible in principle to instantiate this SMC algorithm as a RIM for solving reasoning tasks, we focus here on neural instantiations inspired by recent recursive reasoners such as TRMs and HRMs, which we show in Section 4 correspond to RIMs with an identity **Reweighter**. The structural roles of S, G, and $\mathcal{R}$ within this correspondence suggest viewing the latent states of a RIM as belief summaries over the reasoning state, and the **Solver** and **Generator** as transition operators that refine these summaries across steps. As a result, their reasoning trajectories might drift away from the target p as steps accumulate, motivating the use of more expressive **Reweighters**.

In the subsequent sections, we consider various instantiations of RIMs.

4 Family of Recursive Inference Machines

The RIM framework serves as a generative template for reasoning architectures, where specific functional choices for the **Solver**, **Generator**, and **Reweighter** define the resulting model's inductive biases. In this section, we explore this family of machines by formalizing existing neural models within the RIM framework and introducing new instantiations designed for effective reasoning. Table 1 summarizes these instantiations.

4.1 Learning to Reason

We conceptualize reasoning as the iterative refinement of a latent state through a structured **Solver-Generator-Reweighter** loop. To demonstrate the flexibility of

Name	Solver (⚙)	Generator (🪄)	Reweighter (⚖)
SimRIM (D)	$f_L(\Theta_L)$	$f_H(\Theta_H)$	$\mathbb{1}$ (Identity)
SimRIM (S)	$f(\Theta)$	$f(\Theta)$	$\mathbb{1}$ (Identity)
RIMA	$f(\Theta)$	$f(\Theta)$	EMA
RIMFormer	$f(\Theta)$	$f(\Theta)$	Transformer
TabRIM	$P_{\text{TabPFN}}(X_i \mid x_{-i}, y, D_{\text{train}})$	$P_{\text{TabPFN}}(Y \mid x, D_{\text{train}})$	$P(e \mid \hat{x})$
HybRIM	Hydra	TabPFN	$\mathbb{1}$ (Identity)

Table 1: **Family of RIMs.** We categorize the architectures based on their components: **Solver**, **Generator**, and **Reweighter**. The two variants of Simple RIM (SimRIM, aka TRM), Decoupled (D) and Shared (S), have a simple identity function **Reweighter**. They are identical to HRM and TRM, respectively. RIMA and RIMFormer improve upon SimRIM (S) by employing a more expressive **Reweighter**. TabRIM implements **Solver** and **Generator** via TabPFN forward passes, while the **Reweighter** encodes prior knowledge about noise. HybRIM extends the pure TabPFN-based model by introducing **Solver** in the form of the Hydra model.

our framework, we now describe how specific choices of these components recover and extend existing neural reasoning architectures.

SimRIM: RIM encoding Tiny and Hierarchical Recursive Models

We first instantiate our framework for recursive reasoning architectures by introducing the *Simple RIM (SimRIM)*. This instantiation provides a unifying formalism that encompasses models such as Hierarchical Reasoning Models (HRM) [4] and Tiny Recursive Models (TRM) [14].

Assuming x is the problem description embedding, $z_H^{(0)}$ and $z_L^{(0)}$ are the initial solution and initial state, we call a RIM $\langle G, S, \mathcal{R} \rangle$ the SimRIM if

- The **Solver** S is implemented by f_L, performing T state updates before every reasoning state update $z_L^i = f_L(z_L^{i-1}, z_H^j, x)$;
- The **Generator** G is implemented by f_H, performing N solution updates $z_H^j = f_H(z_H^{j-1}, z_L^{T-1})$;
- The **Reweighter** $\mathcal{R}$ is the identity function.

Note that SimRIM is identical to the HRM. Moreover, when the **Solver** and **Generator** networks, f_L and f_H, are implemented using the same neural backbone (say, f), the above reduces to a TRM. In both cases, the **Reweighter** is the identity function. So, without a non-trivial identity function, the model starts with the initial solution and state $\langle z_H^{(0)}, z_L^{(0)} \rangle$ and extends it by appending $z_L^{(i)}$ for T steps and directly generates a solution update $z_H^{(1)}$; this process is repeated N times.

RIMA: RIM using Exponential Averages.

The observation that existing models employ an identity reweighting function motivates a more expressive parameterization. In SMC, the **Reweighter** is critical for correcting

proposal bias and preventing reasoning drift. On the other hand, moving averages are ubiquitous and can be found in GRU [22], LRU [23], MAMBA [24], and RWKV [25]. For RIMs, exponential moving averages offer an elegant way to balance the past and the present. Rather than treating all history equally or discarding it abruptly, as the SimRIMs currently do, we can naturally down-weight older information while keeping it in view using exponential moving averages. That is, the more distant an intermediate result ("thought"), the less it influences the current "thought". This makes RIM use a principled and computationally lightweight mechanism for tracking changing thoughts: recent observations and thoughts shape the current thought most strongly, yet the full history leaves a trace. The decay parameters give intuitive control over this trade-off, from long-horizon memory to rapid adaptation, making EMA a versatile building block of RIM $\langle G, S, \mathcal{R} \rangle$.

This leads us to the first proposed model, called RIMA, which has the following dynamics.

- Both the **Solver** S and **Generator** G are defined similar to the SimRIM with $f_H = f_L$, producing updates for $\tilde{z}_L^{(i)}$ and $\tilde{z}_H^{(j)}$;
- The **Reweighter** $\mathcal{R} = (\mathcal{R}_L, \mathcal{R}_H)$ contains two neural backbones. The $\mathcal{R}_L$ is defined as

$$\mathcal{R}_L\left(\tilde{z}_L^{(i)}, z_L^{(i)}\right) = \alpha_L^{(i)} \tilde{z}_L^{(i)} + \left(1 - \alpha_L^{(i)}\right) z_L^{(i)},$$

  where $\alpha_L^{(i)}$ controls the update inertia. This coefficient can be a fixed scalar, learnable scalar, or vector, computed via input-dependent gating:

$$\alpha_L^{(i)} = \sigma\left(\text{LinLayer}\left(\tilde{z}_L^{(i)}\right)\right).$$

  The $\mathcal{R}_H$ is defined analogously.

From now on, we call a **Reweighter** *dynamic* if it contains learnable parameters, and *static* otherwise.

RIMformer: RIM using a Transformer as Reweigther

For reasoning tasks involving long horizons and frequent backtracking, maintaining a deeper historical context is essential. To this end, we introduce the k-lookback **Reweighter** $\mathcal{R}$, a function that conditions the update on the current candidate and k previous values:

$$\mathcal{R} : \underbrace{\mathbb{R}^d \times \mathbb{R}^d \times \cdots \times \mathbb{R}^d}_{k+1 \text{ times}} \to \mathbb{R}^d$$

where d denotes the dimension of the hidden space. Formally, given a candidate output $\tilde{z}_X^{(n)}$ from the backbone, the update is defined as

$$z_X^{(n+1)} = \mathcal{R}_X\left(\tilde{z}_X^{(n)}, z_X^{(n)}, \ldots, z_X^{(n-k)}\right),$$

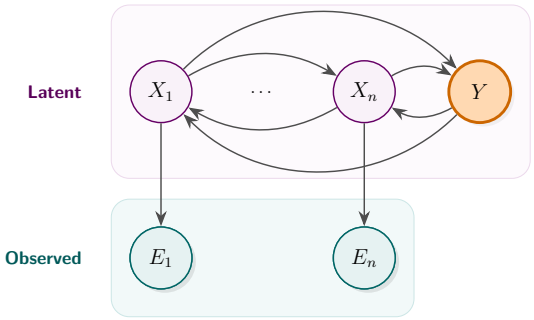


Figure 3: **Tabular RIM (TabRIM) for Reasoning under Uncertainty** To reason about the distribution over the target Y given noisy observations $E = e$, the TabRIM decomposes the task into a **Solver-Reweighter-Generator** loop. The **Solver** iteratively denoises the observation by iteratively resampling latent variables from their full conditionals. Each such conditional is implemented as a single forward pass over TabPFN. The **Reweighter** weighs these samples against the observed evidence to ensure consistency with it. These consistency-weighted samples are used by the **Generator** to infer the final target distribution by computing the empirical expectation over them.

where $X \in \{L, H\}$ and $n \in \{1, \ldots, Y_X\}$, where $Y_L = T$ and $Y_H = N$.

The lookback mechanism serves as the primary motivation for our proposed architecture, the RIMformer. In this model, the **Reweighter** $\mathcal{R}$ is implemented as a Transformer block, which explicitly captures the dependencies across the entire reasoning history via its self-attention mechanism. Formally, we define the RIMformer as the tuple $\langle G, S, \mathcal{R} \rangle$, where

- The **Solver** S and **Generator** G are defined as for the RIMA;
- The **Reweighter** $\mathcal{R} = (\mathcal{T}_L, \mathcal{T}_H)$ consists of two transformer models. The $\mathcal{T}_L$ is defined as follows

$$\mathcal{T}_L\left(h_L\right) = \mathrm{MLP}\left(\mathrm{Norm}\left(A'\right)\right) + A',$$

where $h_L = \left(\tilde{z}_L^{(i)}, z_L^{(i)}, \ldots, z_L^{(0)}\right)$ and A' denotes the attention output representations corresponding to the candidate tokens after applying self-attention over the concatenated history and candidate sequence. The definition of $\mathcal{T}_H$ is analogous.

4.2 In-context reasoning with TabPFN

To highlight the applicability of the RIM framework to pretrained models, we consider tabular reasoning with TabPFN [8]. TabPFN is a prior-fitted transformer, which has been trained on a large number of synthetic datasets; it answers predictive queries without an explicit learning step by treating the training data as in-context examples. While this allows TabPFN to exploit prior knowledge embedded in its weights, the lack

of an explicit model makes it challenging to deal with noise at deployment. Deployed models must often process data that is significantly noisier than their training data, which is typically collected from controlled environments.

In traditional machine learning, models are made robust to deployment-time noise by integrating domain-specific knowledge during the training phase, such as by augmenting the loss function with a knowledge-based regularization term [26, 27]. However, exploiting such domain knowledge is uniquely challenging for TabPFNs because they lack an explicit gradient-based learning step. To enable TabPFNs to reason about noisy input features E to infer the distribution over the target Y, we construct the following RIM and call it Tabular RIM (TabRIM):

Problem description consists of a training context $D_{\text{Train}} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ and a set of noisy test features e.

Latent state consists of a sequence of samples $\hat{x}$ drawn from a Markov Chain approximating the distribution over denoised features, $P(X \mid e, y, D_{\text{Train}})$. Each sample represents a potential denoised version of the input.

The Solver is responsible for state refinement. It uses TabPFN's in-context learning to generate denoised samples of the latent features. By iteratively sampling from the full conditionals over each feature, the **Solver** produces a set of clean feature hypotheses $\hat{x}$ consistent with the training data's distribution.

The Generator infers the answer by producing the predictive distribution over the target Y. It marginalizes over the denoised samples provided by the **Solver**:

$$P(Y \mid e) = \frac{1}{Z|H|} \sum\nolimits_{\hat{x} \in H} w(\hat{x}, e) P(Y \mid X = \hat{x}, D_{\text{Train}}),$$

where $Z = \sum_{\hat{x} \in H} w(\hat{x}, e)$.

The Reweighter assigns importance weights $w(\hat{x}, e)$ to the **Solver**'s denoised samples to account for their likelihood given the observed evidence. Given a feature-wise deviation rate ϵ, the weight of a sample can be defined using a function $w : \text{Domain}(X)^2 \mapsto \mathbb{R}^+$. For example, a simple **Reweighter** could be defined as:

$$w(\hat{x}, e) = \prod\nolimits_{i=1}^{n} \epsilon \cdot \mathbb{1}[e_i \neq \hat{x}_i] + (1 - \epsilon) \cdot \mathbb{1}[e_i = \hat{x}_i]$$

where $\epsilon > 0$ is a known feature-wise noise parameter. This reweighting ensures that the **Generator**'s predictive distribution is dominated by the predictions corresponding to the most plausible denoised features, allowing it filter out the effect of noise.

The efficacy of the Tabular RIM stems from its formal equivalence to Gibbs sampling from a probabilistic generative model over the observed features E, their clean version X, and the target Y.

$$P(E, X, Y) \approx P(E \mid X) P(Y \mid X) \prod\nolimits_i P(X_i \mid X_{-i}, Y)$$

The structure of this model is presented in Fig. 3 The operation of TabRIM is identical to inferring the following expectation using Gibbs sampling.

$$P(Y \mid E = e) = \mathbb{E}_{x \sim P(X|e)}[P(Y \mid X = x)]$$

The **Solver** iteratively samples from TabPFN-based full conditionals to generate samples $\{\hat{x}^{(k)}\} \sim P(X)$, which are reweighed using the emission probability of evidence e as $w(\hat{x}) = P(e \mid \hat{x})$. The reweighed samples are used to empirically approximate the expectation as

$$\mathbb{E}_{x \sim P(X|e)}[P(Y \mid X = x)] \approx \frac{1}{|H|} \sum\nolimits_{\hat{x} \in H} P(Y \mid \hat{x}).$$

This model is guaranteed to infer the correct distribution, assuming

(1) the emission model accurately encodes domain knowledge about noise, a property that is outside our models,
(2) the number of iterations is sufficiently large to ensure the Gibbs sampler converges (after discarding burn-in samples), which is a standard assumption for Gibbs sampling, and
(3) the full conditionals are accurately modeled by TabPFN, which is reasonable as the original TabPFN framework is trained on a large number of simulated datasets and exhibits performance that equals or exceeds the best tabular learning methods.

4.3 Hybrid Tabular Model using RIM

While TabRIM uses multiple forward passes over a Tabular foundation model like TabPFN to perform reasoning, the RIM framework can equally be applied within the foundation model to improve its efficiency. We first cast TabPFN as an incomplete RIM, and then refine it to yield TabHybRIM.

The RIM framework admits two limiting cases with respect to parameter sharing across recursive steps. When S and G share parameters, the RIM recovers the recursive, parameter-efficient limit of TRMs. When S and G carry separate parameters per layer, the RIM unrolls into a deep reasoner where each layer specialises to its position in the chain. TabPFN is a Transformer that alternates attention over rows and features, processing the full context of training and test examples in a single forward pass. Within the RIM framework, the Transformer layers play the role of the **Generator** G, proposing candidate solution updates via attention over row and feature embeddings. However, TabPFN has no **Solver** and no **Reweighter**: no component maintains and updates a separate reasoning state, and no correction is applied to candidate updates. TabPFN is therefore a degenerate RIM $\langle \mathbb{1}, G, \mathbb{1} \rangle$, lacking both the state propagation of the **Solver** and the correction of the **Reweighter**.

We complete the TabPFN RIM by introducing a **Solver** in the form of Hydra layers [17], a selective state space model that propagates and compresses the current state efficiently across the input sequence. TabHybRIM alternates Hydra layers as the **Solver** S and TabPFN Transformer layers as the **Generator** G, with an identity

		ARC-AGI 1		ARC-AGI 2		Sudoku-Extreme	Maze-Hard
		pass@1 (↑)	pass@2 (↑)	pass@1 (↑)	pass@2 (↑)	Accuracy (↑)	Accuracy (↑)
Symbolic	TRM/SimRIM	40.5	44.38	4.6	7.4	87.16	85.3
	RIMformer	43.25 (+2.75)	47.13 (+2.75)	5.83 (+1.23)	7.1 (-0.30)	80.21 (-6.95)	87.7 (+2.4)
	RIMA	42.5 (+2.00)	47.5 (+3.12)	9.9 (+5.30)	11.3 (+3.9)	89.39 (+2.23)	87.0 (+1.7)

		Cleveland	Ljubljana	Thyroid	Hepatitis
		AUC-ROC (↑)	AUC-ROC (↑)	AUC-ROC (↑)	AUC-ROC (↑)
Tabular	TabPFN	0.85	0.63	0.76	0.96
	TabRIM	0.87 (+0.02)	0.74 (+0.11)	0.80 (+0.04)	0.97 (+0.01)

		Params in Millions(↓)	Inference time @32k (s) (↓)	Accuracy (↑)
Efficiency	12×TabPFN	25	7.22	88.13
	24×Hydra	160	7.27 (+0.05)	87.81 (-0.32)
	HybRIM (8 layers)	16	4.71 (-2.51)	88.04 (-0.09)

Table 2: **RIMs improve over baselines across symbolic reasoning, tabular reasoning under noise, and architectural efficiency.** Symbolic results are on ARC-AGI 1&2, Sudoku-Extreme, and Maze-Hard; tabular results are on four medical diagnosis datasets with 25% feature corruption; efficiency results are on OpenML CC-18. Differences are w.r.t. TRM/SimRIM (symbolic), TabPFN (tabular), and 12×TabPFN (efficiency). Performance scores are color-coded, the more blue the better. (Best viewed in color.)

Reweighter $\mathcal{R} = \mathbb{1}$, and carries separate parameters per layer. The Hydra layers serve a complementary role to the Transformer layers: where the **Generator** attends globally over rows and features to propose solution updates, the **Solver** propagates state across the sequence in linear rather than quadratic time.

5 Experimental Evaluation

We aim to empirically investigate the performance and generalization of different RIMs across a range of tasks. To this end, we conduct several experiments to address the following questions[1]:

(Q1) How important is reweighting for neural reasoning tasks?
(Q2) Which approach is more effective: dynamic or static reweighting?
(Q3) Does a greater lookback lead to more accurate results?
(Q4) Does the RIM framework generalize to tabular domains and provide robustness against observational noise?
(Q5) Can the RIM framework improve the efficiency of a tabular foundation model without sacrificing predictive performance?

Domains. To this end, we focused on two domains to evaluate instantiations of the RIMs: neural reasoning and tabular data. Within the neural reasoning domain, we evaluated on four benchmarks: ARC-AGI-1, ARC-AGI-2, Sudoku Extreme, and Maze-Hard. *ARC-AGI-1* consists of geometric puzzles designed to be solvable by

[1] Code is available at https://anonymous.4open.science/r/Recurent-Inference-Machines-5FDC

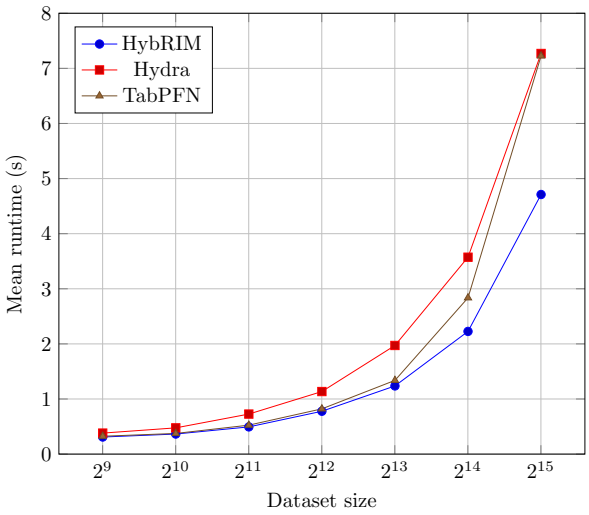


Figure 4: **Mean inference time of HybRIM, Hydra, and TabPFN as a function of dataset size.** HybRIM consistently achieves lower inference time than both baselines across all dataset sizes, with the gap widening at larger scales

humans while challenging current machine learning models [28]. Each task provides 2–3 demonstration examples along with a test input and requires modeling global dependencies over 30×30 grids. ARC-AGI-2 expands on ARC-AGI-1 with a larger and more diverse set of tasks while preserving the few-shot setup [29]. Both benchmarks require long-horizon reasoning across the grid, motivating the use of an attention block in the model backbone. *Sudoku Extreme* contains complex 9×9 puzzles [30]. The training set consists of 1,000 examples, whereas the test set contains 423,000 samples. Prior work has shown that a pure MLP-based backbone achieves the best performance on this benchmark [5], which we adopt for our experiments. *Maze-Hard* includes 30×30 mazes [31], with 1,000 training and 1,000 test examples. Because these tasks involve long-range dependencies across the entire grid, neural modules for this task include an attention block. Together, these benchmarks span a spectrum of neural reasoning challenges, from small, fixed-context tasks to problems with large grids and long-horizon dependencies. Within the tabular domain, we evaluate two distinct instantiations of the RIM framework. To evaluate TabRIM, we use four medical datasets from the UCI Machine Learning Repository: *Cleveland Heart Disease*, *Ljubljana Breast Cancer*, *Thyroid*, and *Hepatitis*. For each test set, we simulate severe data corruption by replacing 25% of all feature values with random values, reflecting realistic deployment conditions where training data is clean but test-time observations are noisy. To evaluate HybRIM, we follow the evaluation procedure of Koch et al. [32] assessing performance on OpenML CC-18 benchmark datasets.

Models. In our experiments for the symbolic benchmarks, we evaluated two RIMs: RIMA and RIMformer, comparing them with SimRIM. Depending on the backbone used for the **Solver** and **Generator**, our models have between 5.5M and 13.6M parameters.

For the tabular reasoning experiments, we evaluated three models: TabPFN, TabRIM, and HybRIM. In TabRIM, the input is treated as a noisy realization of a latent clean state, which is refined by iteratively re-sampling each variable's values conditioned on the remaining variables' values. Equivalently, this can be viewed as a Gibbs sampling-based denoiser, where each conditional is defined via forward passes through a common TabPFN model. We generated 10 samples, after discarding 5 burn-in samples, and estimated the final answer as their mean predictive probability. The HybRIM has 16M parameters and alternates the TabPFN's Transformer layers with Hydra layers. It follows the training procedure of Koch et al. [32].

Metrics. To evaluate the performance on ARC-AGI, we report pass@1 and pass@2, i.e., the fraction of tasks where the top-1 or top-2 outputs match the ground truth. For Sudoku Extreme and Maze Hard, we report standard exact-match accuracy. For tabular reasoning, we quantified predictive performance in terms of AUC-ROC and AUC-PR. For the hybrid models we use the standard mean accuracy over all evaluated datasets. The inference time is measured in synchronized elapsed seconds.

Answer (Q1), Reweighting: Table 2 summarizes the performance of SimRIM, RIMformer, and RIMA on the 4 symbolic reasoning benchmarks. We observe that the RIMs with non-trivial **Reweighters** (RIMFormer and RIMA) outperform identity-reweighted SimRIM on most benchmarks, with RIMA achieving the strongest overall results. However, the gains are not uniform across all tasks and variants; we discuss the role of lookback size in Answer (Q3). Overall, these results suggest that reweighting is an important component of neural reasoning, as it enables the model to identify and prioritize high-signal latent trajectories.

Answer (Q2), Dynamic vs. Static Weighting: Table 3 presents an ablation study comparing dynamic and static reweighting strategies on the Sudoku Extreme benchmark. We evaluate the efficacy of our learned reweighting mechanism against static baselines and hybrid variants where reweighting is restricted to either the **Solver** or **Generator** outputs. Static EMA **Reweighters** with low fixed weights, such as $(\alpha_L, \alpha_H) = (0.4, 0.4)$, perform worse than no reweighting at all: overly conservative weights prevent the model from making progress toward a solution by suppressing proposed state updates. In contrast, learning the weights as global scalars or via a neural module as in RIMA allows problem- and context-specific adaptation, yielding consistently superior performance. This suggests that joint adaptation of both latent trajectories is essential for solving high-complexity reasoning tasks.

Answer (Q3), Lookback size: Table 2 compares RIMA, having a lookback of size 1, with RIMformer, having a lookback of size $\max(N, T)$. We observe that increasing the lookback window size does not uniformly improve performance. While it improves performance on Maze-Hard, it reduces performance on Sudoku-Extreme. In contrast, Sudoku-Extreme does not require substantial backtracking; in such cases, a smaller lookback acts as a beneficial inductive bias, regularizing the model against overly complex search trajectories. So, the underperformance of a larger lookback RIMformer model might be attributed to overfitting, as the high-capacity Transformer **Reweighter** may over-parameterize the relatively simple constraints of these smaller instances.

Answer (Q4), Tabular reasoning: Table 2 compares TabRIM with TabPFN on 4 noisy medical diagnosis benchmark datasets in terms of AUC-ROC and AUC-PR.

TRM Baselines	Sudoku Extreme	RIM Instances	Sudoku Extreme
SimRIM (aka TRM)	87.16	RIMformer	80.21 (-6.95)
$(\alpha_L, \alpha_H) = (0.8, 0.8)$	82.76 (-4.40)	RIMA ($\tilde{z}_L$ only)	87.84 (+0.68)
$(\alpha_L, \alpha_H) = (0.6, 0.6)$	85.29 (-1.87)	RIMA ($\tilde{z}_H$ only)	87.57 (+0.41)
$(\alpha_L, \alpha_H) = (0.4, 0.4)$	80.26 (-6.90)	RIMA	89.39 (+2.23)
$(\alpha_L, \alpha_H) = (0.2, 0.2)$	74.52 (-12.64)		
$(\alpha_L, \alpha_H) = (\theta_1, \theta_2)$	84.27 (-1.87)		

Table 3: **Reweighting Ablation: Static vs. Dynamic.** Test set accuracies on the Sudoku Extreme benchmark for RIMA (with full dynamic reweighting), and its variants with partial dynamic reweighting ($\tilde{z}_L$ only and $\tilde{z}_H$ only), learnable scalars (θ_1, θ_2) and fixed values. Clearly, the full reweighting outperforms other configurations. Scores are color-coded (the bluer, the better) and the differences are w.r.t. SimRIM/TRM. (best viewed in color)

TabRIM consistently outperforms direct inference on all datasets and both metrics. The gains are most pronounced on Ljubljana Breast Cancer and Thyroid, where feature corruption is most damaging to a model without a mechanism for reasoning about observational noise. The smaller but consistent gains on Cleveland Heart Disease and Hepatitis suggest that TabRIM's iterative reasoning provides robust improvements across varying levels of dataset difficulty.

Answer (Q5), Tabular hybrid: Table 2 and Figure 4 compare the performance and efficiency of the HybRIM to TabPFN and Hydra baselines on the OpenML CC-18 benchmark. HybRIM matches the predictive performance of TabPFN, while achieving a 35% speedup in inference time, and does so with substantially fewer parameters (16M vs. 25M). This confirms that casting a foundation model as an incomplete RIM and completing it with a Hydra-based **Solver** yields efficiency gains without sacrificing accuracy.

6 Conclusion

We introduced Recursive Inference Machines (RIMs), a unifying design space for neural reasoning architectures expressed symbolically via **Solver**, **Generator**, and **Reweighter**. By formalizing inference dynamics as an explicit, iterative process, RIMs provide a global architectural perspective that exposes previously implicit design choices. This clarity enables principled generalizations—such as our RIMformer and k-lookback variants—that extend the capabilities of existing models. Our empirical evaluation across diverse symbolic and tabular reasoning domains demonstrates that RIMs outperform neural and tabular reasoning architectures.

The choice of the **Reweighter** in the presented instantiations leaves substantial room for further exploration. One promising direction is to employ an Extended Long Short-Term Memory (xLSTM) [33] as the **Reweighter**, enabling the model to capture the influence of previous reasoning steps through an explicit long-term memory mechanism. Exploring alternative neural architectures for other components may further

improve the performance; in particular, in an automatic fashion based on scaling laws, analogous to an AI scientist for reasoning architectures. For instance, replacing the TRM's neural backbone with a Universal Transformer has been shown to yield strong improvements on reasoning tasks [34]. While we do not evaluate this architecture in our experiments, it can be naturally combined with the proposed **Reweighter** within the RIM framework. Further, while RIM goes beyond linear, Chain-of-Thought reasoning by explicitly reasoning about previously generated solutions and states, it could be extended to a Tree-of-Thoughts–style setup [35], where multiple reasoning trajectories are explored concurrently and subsequently evaluated or reweighted. Such branching can be interpreted as parallel inference chains whose latent states are selectively retained or aggregated, offering a principled way to trade off exploration and computation.

Together, these directions position RIM as a modular path forward for (potentiall even automatically) designing the next generation of efficient, interpretable neural engines that could bridge the gap between raw pattern matching and high-level symbolic manipulation.

7 Acknowledgement

We gratefully acknowledge support from the BMFTR project XEI (grant number 16IS24079B), the Cluster of Excellence "Reasonable AI" funded by the German Research Foundation (DFG) under Germany's Excellence Strategy (EXC-3057), EU-SAI: Sovereign Open Source Foundations Models (grant number 13IPC040G), the DYNAMIC center funded by the LOEWE program of the Hessian Ministry of Science and Arts (grant number LOEWE1/16/519/03/09.001(0009)/98), and the US Army Research Office (award W911NF2010224).

References

[1] Shi, Z., Li, M., Liu, Y., Khan, S., Huang, J., Zhen, H.-L., Yuan, M., Xu, Q.: Satformer: Transformer-based unsat core learning. In: 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 1–4 (2023). IEEE

[2] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., *et al.*: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)

[3] Wang, X., Wei, J., Schuurmans, D., Le, Q.V., Chi, E.H., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. In: The Eleventh International Conference on Learning Representations

[4] Wang, G., Li, J., Sun, Y., Chen, X., Liu, C., Wu, Y., Lu, M., Song, S., Yadkori, Y.A.: Hierarchical reasoning model. arXiv preprint arXiv:2506.21734 (2025)

[5] Jolicoeur-Martineau, A.: Less is more: Recursive reasoning with tiny networks. arXiv preprint arXiv:2510.04871 (2025)

[6] Pearl, J.: Reverend bayes on inference engines: A distributed hierarchical approach. In: Probabilistic and Causal Inference: the Works of Judea Pearl, pp. 129–138 (2022)

[7] Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on pattern analysis and machine intelligence (6), 721–741 (1984)

[8] Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S.B., Schirrmeister, R.T., Hutter, F.: Accurate predictions on small data with a tabular foundation model. Nature **637**(8045), 319–326 (2025)

[9] Bhuyan, B.P., Ramdane-Cherif, A., Tomar, R., Singh, T.: Neuro-symbolic artificial intelligence: a survey. Neural Computing and Applications **36**(21), 12809–12844 (2024)

[10] Liu, X., Liang, T., He, Z., Xu, J., Wang, W., He, P., Tu, Z., Mi, H., Yu, D.: Trust, but verify: A self-verification approach to reinforcement learning with verifiable rewards. arXiv preprint arXiv:2505.13445 (2025)

[11] Schuurmans, D., Dai, H., Zanini, F.: Autoregressive large language models are computationally universal. arXiv preprint arXiv:2410.03170 (2024)

[12] Krantz, S.G., Parks, H.R.: The Implicit Function Theorem: History, Theory, and Applications vol. 202. Springer, ??? (2002)

[13] Asadulaev, A., Banerjee, R., Karray, F., Takac, M.: Deep improvement supervision. arXiv preprint arXiv:2511.16886 (2025)

[14] Roye-Azar, A., Vargas-Naranjo, S., Ghai, D., Balamurugan, N., Amir, R.: Tiny recursive models on arc-agi-1: Inductive biases, identity conditioning, and test-time compute. arXiv preprint arXiv:2512.11847 (2025)

[15] Somvanshi, S., Islam, M.M., Mimi, M.S., Polock, S.B.B., Chhetri, G., Das, S.: From s4 to mamba: A comprehensive survey on structured state space models. arXiv preprint arXiv:2503.18970, 1–30 (2025)

[16] Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)

[17] Hwang, S., Lahoti, A., Dao, T., Gu, A.: Hydra: Bidirectional state space models through generalized matrix mixers, 2024. URL https://arxiv. org/abs/2407.09941

[18] Merrill, W., Li, Y., Romero, T., Svete, A., Costello, C., Dasigi, P., Groeneveld, D., Heineman, D., Kuehl, B., Lambert, N., et al.: Olmo hybrid: From theory to practice and back. arXiv preprint arXiv:2604.03444 (2026)

[19] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible

Inference. Elsevier, ??? (2014)

[20] Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT press, ??? (2009)

[21] Ross, S., Munoz, D., Hebert, M., Bagnell, J.A.: Learning message-passing inference machines for structured prediction. In: CVPR 2011, pp. 2737–2744 (2011). IEEE

[22] Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111 (2014)

[23] Ahuja, C., Morency, L.-P.: Lattice recurrent unit: Improving convergence and statistical efficiency for sequence modeling. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

[24] Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. In: First Conference on Language Modeling (2024)

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

[26] Karpatne, A., Kannan, R., Kumar, V.: Knowledge Guided Machine Learning: Accelerating Discovery Using Scientific Knowledge and Data. CRC Press, ??? (2022)

[27] Karanam, A., Mathur, S., Sidheekh, S., Natarajan, S.: A unified framework for human-allied learning of probabilistic circuits. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 17779–17787 (2025)

[28] Chollet, F.: On the measure of intelligence. arXiv preprint arXiv:1911.01547 (2019)

[29] Chollet, F., Knoop, M., Kamradt, G., Landers, B., Pinkard, H.: Arc-agi-2: A new challenge for frontier ai reasoning systems. arXiv preprint arXiv:2505.11831 (2025)

[30] Dillon, T.: Tdoku: A Fast Sudoku Solver and Generator. https://t-dillon.github.io/tdoku/. Accessed: 2025-01-31 (2025)

[31] Lehnert, L., Sukhbaatar, S., Su, D., Zheng, Q., Mcvay, P., Rabbat, M., Tian, Y.: Beyond a*: Better planning with transformers via search dynamics bootstrapping. arXiv preprint arXiv:2402.14083 (2024)

[32] Koch, F., Wever, M., Raisch, F., Tischler, B.: State-space models for tabular prior-data fitted networks. arXiv preprint arXiv:2510.14573 (2025)

[33] Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: xlstm: Extended long short-term memory. Advances in Neural Information Processing Systems **37**, 107547–107603 (2024)

[34] Gao, Z., Chen, L., Xiao, Y., Xing, H., Tao, R., Luo, H., Zhou, J., Dai, B.: Universal reasoning model. arXiv preprint arXiv:2512.14693 (2025)

[35] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in neural information processing systems **36**, 11809–11822 (2023)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- RecursiveInferenceMachines21.pdf