A Survey on Latent Reasoning

Rui-Jie Zhu[⋆,†], **Tianhao Peng**[⋆], **Tianhao Cheng**[⋆], **Xingwei Qu**[⋆],
Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu[†], **Ge Zhang**[†], **Wenhao Huang, Jason Eshraghian**[†]

UCSC, FDU, NJU, PKU, RUC, UoM, UW-Madison, PolyU, M-A-P

Abstract

Large Language Models (LLMs) have demonstrated impressive reasoning capabilities, especially when guided by explicit chain-of-thought (CoT) reasoning that verbalizes intermediate steps. While CoT improves both interpretability and accuracy, its dependence on natural language reasoning limits the model's expressive bandwidth. Latent reasoning tackles this bottleneck by performing multi-step inference entirely in the model's continuous hidden state, eliminating token-level supervision. To advance latent reasoning research, this survey provides a comprehensive overview of the emerging field of latent reasoning. We begin by examining the foundational role of neural network layers as the computational substrate for reasoning, highlighting how hierarchical representations support complex transformations. Next, we explore diverse latent reasoning methodologies, including activation-based recurrence, hidden state propagation, and fine-tuning strategies that compress or internalize explicit reasoning traces. Finally, we discuss advanced paradigms such as infinite-depth latent reasoning via masked diffusion models, which enable globally consistent and reversible reasoning processes. By unifying these perspectives, we aim to clarify the conceptual landscape of latent reasoning and chart future directions for research at the frontier of LLM cognition. An associated GitHub repository collecting the latest papers and repos is available at: LatentCoT-Horizon.

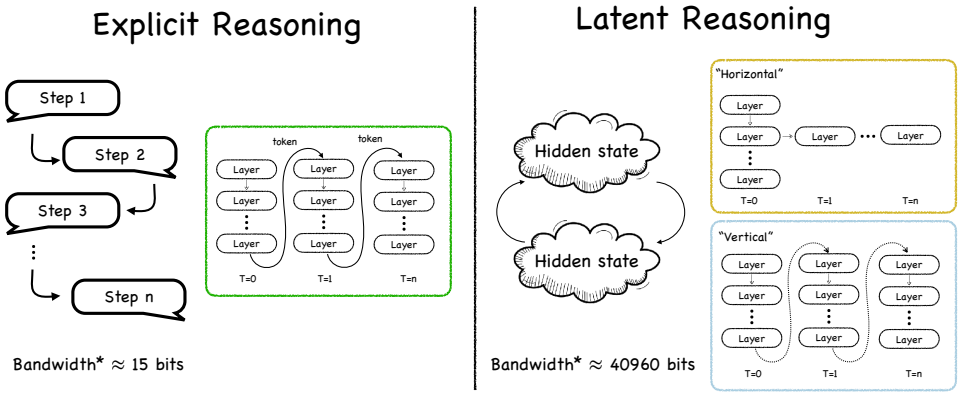

Figure 1. Explicit reasoning transmits discrete tokens (≈ 15 bits each), whereas latent reasoning exchanges full 2560-dimensional FP16 hidden states (≈ 40,960 bits each), revealing a $\sim 2.7 \times 10^3$-fold bandwidth gap between the two approaches.

⋆ Equal Contribution. † Corresponding Authors.

Contents

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in performing reasoning tasks, in some cases even exceeding human-level performance [47, 64, 82, 133]. LLMs often reason more effectively when they produce a Chain-of-Thought (CoT) [115], spelling out each intermediate step in natural language before arriving at a final answer.

Initially viewed as a logical extension to prompt engineering, CoT gained traction once supervised instruction tuning exposed models to many annotated reasoning traces. It then became the norm when RL rewarded answer correctness [49], which encouraged models to generate their own effective chains of thought. As a result, LLMs that "think in language before answering" have attained remarkable performance improvements. This principle now anchors leading reasoning models, including the Qwen3 series [118], DeepSeek-R1 [41], and Gemini 2.5 series [32].

However, just as humans do not always rely on language for their cognitive processes, LLMs spend most of their processing budget in the latent space. Enforcing a CoT to operate with natural language can constrain a model's expressive range and can also impose redundant computation. Latent Chain-of-Thought (Latent CoT) has the potential to overcome these limits [23, 44]. Unlike its explicit counterpart that relies on discrete tokens, latent CoT carries reasoning in continuous internal representations, often via recurrent mechanisms within the model. This offers richer expressivity and access to non-linguistic reasoning paths, potentially unlocking new frontiers in model reasoning.

This survey examines the emerging landscape of Latent CoT and its potential to surpass language-based reasoning constraints. While explicit CoT forces thoughts into a string of tokens, Latent CoT shifts the entire reasoning process into the model's continuous representational space. The aim is to expand expressiveness and raise the performance ceiling: freed from a finite vocabulary, a model can explore reasoning trajectories with no direct linguistic equivalent. We categorize and analyze the technical approaches that leverage these continuous representations to achieve more advanced reasoning.

The structure of this survey is designed to provide a comprehensive understanding of Latent CoT and its various implementations. Our taxonomy breaks this down in Figure 2. We begin by establishing a general formulation that captures most Latent CoT implementations, before classing techniques into more specific categories. These categories can be broadly divided into two types: 1) **vertical recurrence** for expanding computational depth, and 2) **horizontal recurrence** for increasing sequential capacity. Vertical recurrence applies feedback loops to activation values, and can be thought of 'activation-based' reasoning [22, 71]. Alternatively, horizontal recurrence uses hidden states to propagate context across long sequences of reasoning trajectories [87, 100]. We then explore fine-tuning strategies designed to compress or internalize explicit reasoning traces, which concludes the review of Latent CoT implementations.

This sets the stage for understanding the mechanistic interpretability of latent reasoning to understand how these processes are realized within neural networks. This section examines the foundational role of network layers as the primary computational substrate for reasoning [92, 137]. We explore the theory of Layer Specialization, which posits that different layers develop distinct, hierarchical functions—from feature extraction in shallow layers to complex logical operations in intermediate layers and final integration in deep layers—collectively forming an implicit computational pipeline analogous to an explicit CoT. Explicit CoT comes with the benefit of intermediate tokens which offers a degree of post-hoc interpretability, and we similarly aim to uncover the mechanisms that enable latent reasoning.

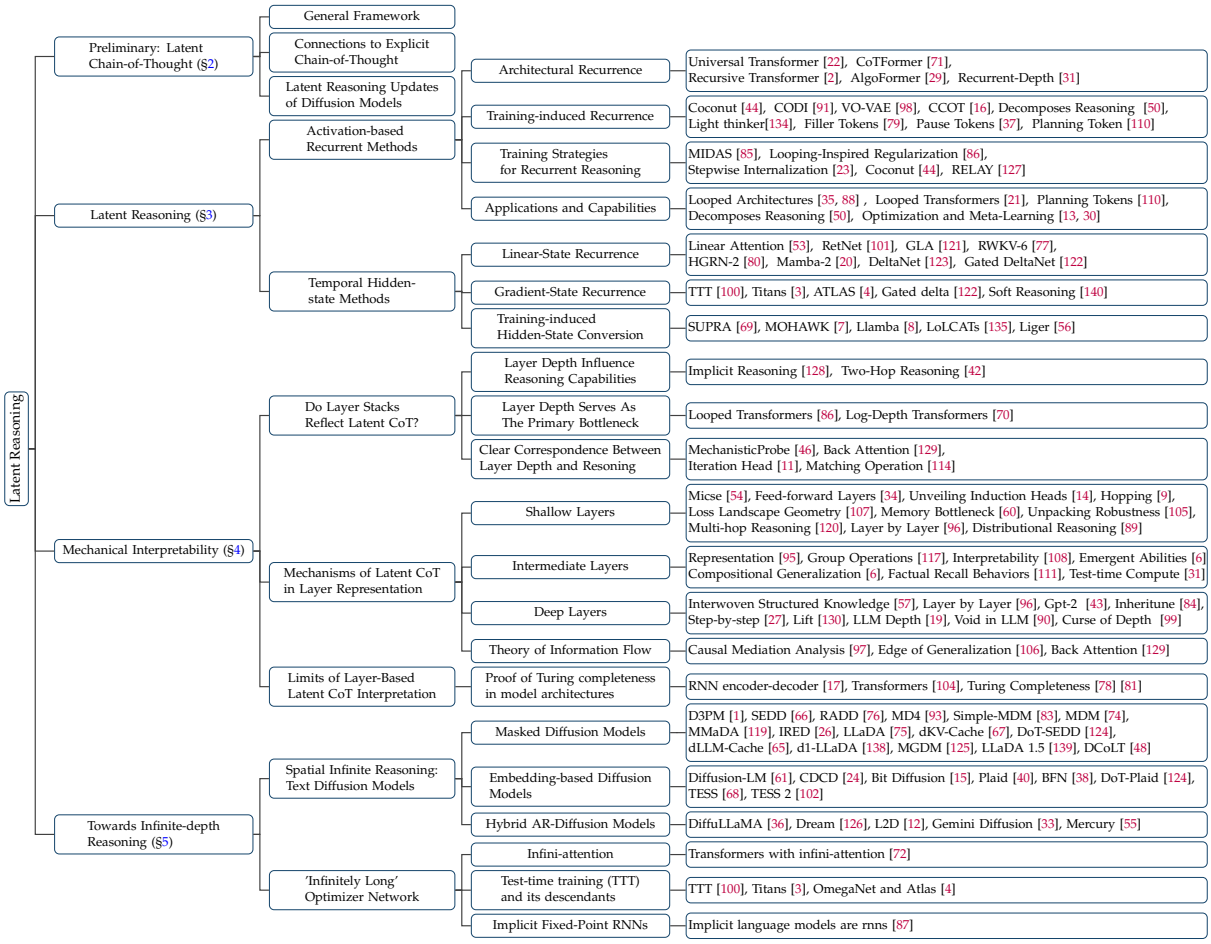

Figure 2. Taxonomy of Latent Reasoning.

Finally, we explore advanced paradigms at the frontier of LLM cognition, focusing on the pursuit of infinite-depth reasoning. This concept refers to a model's ability to devote unbounded computational steps to refine a solution, moving beyond fixed-depth architectures. Our discussion centers on spatial infinite reasoning as realized by text diffusion models [74, 124]. Unlike traditional autoregressive generation, these models operate on the entire output sequence in parallel, enabling global planning and iterative self- correction through bidirectional context. This approach facilitates globally consistent and reversible reasoning processes, offering a promising path toward more powerful and flexible AI systems.

2. Preliminary: Latent Chain-of-Thought

In this section, we present a unified mathematical framework for understanding various Latent CoT approaches. Unlike traditional CoT reasoning that generates explicit textual intermediate steps, latent CoT methods perform reasoning through continuous representations and hidden states within the model's computational graph. We categorize these approaches based on how they propagate information across layers (spatial dimension) and time steps (temporal dimension).

2.1. General Framework

We begin by establishing a general formulation for transformer-based reasoning systems. Consider a transformer model processing information at time step t and layer l. Let $x_t^l \in \mathbb{R}^d$ denote the activation at layer l and time t. We introduce $\mathbf{S}_t^l$ to represent the hidden state that captures historical information. The structure and dimensionality of $\mathbf{S}_t^l$ are architecture-dependent and define how context is maintained. This state can manifest in several forms, including:

- **KV Cache:** In standard Transformers, $\mathbf{S}_t^l$ is the Key-Value (KV) cache, comprising a pair of matrices $(\mathbf{K}_t^l, \mathbf{V}_t^l)$, where $\mathbf{K}_l, \mathbf{V}_l \in \mathbb{R}^{n \times d}$ and n is the sequence length of the context. Note that as t increases, so does n.
- **Linear Attention State:** In models with linear attention, the hidden state can be compressed into a fixed-size state matrix, $\mathbf{S}_t^l \in \mathbb{R}^{d \times d}$, which allows for efficient, recurrent-style updates.
- **Recurrent State:** For RNN-like mechanisms, $\mathbf{S}_t^l$ is a single state vector, $\mathbf{S}_t^l \in \mathbb{R}^d$, that summarizes all past information into a fixed-size representation.

With this generalized view, the fundamental operations in latent reasoning can be decomposed into spatial and temporal transformations.

The spatial transformation propagates information vertically through layers at a fixed time step:

$$\boldsymbol{x}_{t+1}^{l+1} = f(\boldsymbol{x}_{t+1}^l, g(\mathbf{S}_t^l, \boldsymbol{x}_t^l)) \tag{1}$$

where f represents the layer-wise transformation function (e.g., a transformer block), which uses the historical context in $\mathbf{S}_t^l$ to compute the next layer's activation; g captures how historical information is maintained and updated. The implementation of g depends on the form of $\mathbf{S}_t^l$ (e.g., appending to the KV cache, or performing a matrix/vector update).

Activation-Based Methods Activation-based methods focus on deepening the computational graph by iteratively refining activations within a single time step. These approaches implement a form of recursive computation where the same transformation is applied multiple times, allowing for progressive refinement of representations.

Formally, activation-based methods compute:

$$\boldsymbol{x}_t^{l+n} = f\left(\ldots f\left(f(\boldsymbol{x}_t^l, g(\mathbf{S}_t^l, \boldsymbol{x}_t^l)), g(\mathbf{S}_t^{l+1}, \boldsymbol{x}_t^{l+1})\right), \ldots, g(\mathbf{S}_t^{l+n-1}, \boldsymbol{x}_t^{l+n-1})\right) \tag{2}$$

This recursive application can be understood as creating a computational loop within the forward pass. At each iteration $i \in \{1, \ldots, n\}$, the model refines its representation by applying the transformation function f, potentially with access to different hidden states $\mathbf{S}_t^{l+i-1}$. Here, l denotes the starting layer index, constrained by $1 \leq l \leq L - n$, where L is the total number of layers in the model. The key insight is that by repeatedly processing the same input with shared parameters, the model can perform iterative refinement analogous to human step-by-step reasoning.

Hidden State-Based Methods Hidden state-based methods take a fundamentally different approach by aggregating information from multiple temporal or spatial contexts simultaneously. Rather than iterative refinement, these methods leverage rich historical representations to inform current computations.

The core computation in hidden state-based methods is:

$$\boldsymbol{x}_t^{l+1} = f\left(\boldsymbol{x}_t^l, g\left(\left(\mathbf{S}_t^l, \mathbf{S}_{t-1}^l, \ldots, \mathbf{S}_{t-n}^l\right), \boldsymbol{x}_t^l\right)\right), \tag{3}$$

This operation allows the model to access a broader context of hidden states, effectively creating a memory bank that spans multiple layers or time steps. The function f must be designed to effectively aggregate and utilize this expanded context, often through specialized attention mechanisms or learnable aggregation functions.

2.2. Connections to Explicit Chain-of-Thought

Understanding how these latent methods relate to explicit Chain-of-Thought reasoning provides important insights. Traditional CoT generates a sequence of tokens $y_1, y_2, \ldots, y_T$ representing intermediate reasoning steps. In the latent framework, these explicit tokens are replaced by continuous representations that evolve according to the dynamics described above.

The correspondence can be formalized by considering the generation process. In explicit CoT:

$$\boldsymbol{y}_{t+1} = \text{Decode}(\text{Transform}(\boldsymbol{x}_t, \mathbf{S}_t)), \tag{4}$$

where the decoding step projects continuous representations back to discrete tokens.

Latent methods eliminate this decoding step, instead maintaining reasoning in the continuous space:

$$\boldsymbol{z}_{t+1} = \text{Transform}(\boldsymbol{z}_t, \mathbf{S}_t), \tag{5}$$

where z_t represents the continuous "thought" at step t.

This fundamental difference enables latent methods to explore reasoning pathways that may not have natural linguistic expressions, potentially discovering more efficient or powerful reasoning strategies unconstrained by the token vocabulary. However, it also introduces challenges in interpretability and training, as the intermediate states no longer correspond to human-readable explanations.

2.3. Latent Reasoning Updates of Diffusion Models

Understanding how latent update methods relate to diffusion models reveals fundamental differences from autoregressive (AR) generation. Traditional diffusion models operate purely through **temporal updates** without explicit spatial transformations, fundamentally differing from the spatial-temporal decomposition in transformer-based reasoning systems.

Temporal-Only Updates Diffusion Models Classical diffusion models perform updates exclusively in the temporal dimension through iterative denoising. The process involves two primary update mechanisms:

Discrete updates (mask-based): Given a sequence of tokens $y_1, \ldots, y_N$, the model selectively updates positions based on masking patterns:

$$\boldsymbol{x}_{t+1}^l(i) = \begin{cases} f(\boldsymbol{x}_t^l(i), \epsilon_t), & \text{if } m_t(i) = 1 \\ \boldsymbol{x}_t^l(i), & \text{otherwise} \end{cases} \tag{6}$$

where $m_t(i)$ represents the mask indicating which tokens to update at step t.

Continuous updates (noise-based): The model applies global noise reduction across all positions:

$$\boldsymbol{x}_{t+1}^{l} = f(\boldsymbol{x}_{t}^{l}, \epsilon_t) \tag{7}$$

where f represents the denoising function that operates uniformly across all token positions.

KV-cache Integrated Diffusion Models Recent advances have begun incorporating bidirectional KV cache mechanisms [67] into diffusion models, introducing spatial-like transformations alongside temporal updates. This hybrid approach bridges the gap between traditional diffusion and transformer-based reasoning.

Confidence-thresholded spatial transformation: All token activations are updated layer-wise at each denoising iteration:

$$\boldsymbol{x}_{t}^{l+1} = f_{\tau}\big(\boldsymbol{x}_{t}^{l}, \mathbf{S}t^{l}, \epsilon_t\big) \tag{8}$$

where f_τ denotes a bidirectional Transformer block that refines every token representation while utilizing cached states.

Selective temporal cache updates: Only tokens whose confidence score $c_t^l(i) = \mathrm{conf}\big(x_t^l(i)\big)$ meets or exceeds threshold τ refresh their KV cache:

$$\mathbf{S}_{t+1}^{l}(i) = \begin{cases} g_\tau\big(x_t^l(i),\, \mathbf{S}_t^l(i)\big), & c_t^l(i) \geq \tau \\ \mathbf{S}_t^l(i), & \text{otherwise} \end{cases} \tag{9}$$

Complete spatio-temporal evolution: The framework combines spatial refinement with selective temporal caching:

$$\boldsymbol{x}_{t+1}^{l+1} = f_{\tau}\big(\boldsymbol{x}_{t+1}^{l},\, \mathbf{S}_{t+1}^{l}\big) \tag{10}$$

This evolution represents a significant departure from traditional diffusion models, incorporating transformer-style spatial processing while maintaining the iterative refinement benefits of temporal diffusion. The confidence-thresholded mechanism enables efficient cache management in bidirectional contexts, addressing the fundamental incompatibility between traditional KV caching and diffusion model architectures.

Consequently, diffusion models scan the entire sequence to identify and update the highest-confidence tokens in parallel—continuously correcting their representations across layers—whereas autoregressive models must commit to a single next token and cannot revisit or refine earlier outputs. As a result, diffusion's spatio–temporal mechanism enables ongoing, bidirectional refinement of multiple reliable latent states, while AR generation proceeds strictly forward, leaving past tokens fixed once generated.

3. Latent Reasoning

The development of latent CoT reasoning follows two fundamental computational paradigms: expanding depth through activation recurrence and expanding temporal capacity through hidden state evolution. As illustrated in Figure 3, activation-based methods create deeper computational graphs by iteratively processing information through the same set of layers, akin to vertical expansion. In contrast, hidden-state-based methods expand the model's memory horizontally, allowing it to access and integrate information over longer sequences.

This distinction raises critical implementation and theoretical questions. For activation-based approaches, **how can a model with a fixed number of layers be architecturally designed or**

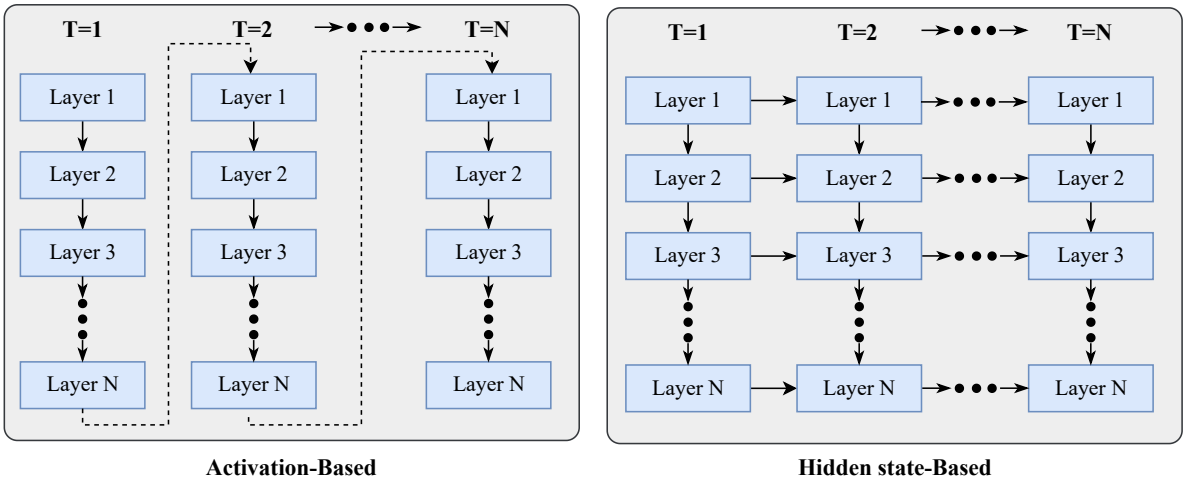

Figure 3. Comparison of Activation-Based and Hidden-state-Based Latent Reasoning. Activation-based methods (left) iteratively refine representations by looping through the same layers over multiple time steps ($T = 1, 2, ..., N$), increasing computational depth. Hidden-state-based methods (right) process information sequentially, evolving a hidden state that carries information across a potentially long temporal sequence ($T = 1, 2, ..., N$).

trained to "think" longer about a problem, effectively creating vertical computational depth on the fly? What are the principles that govern this induced recurrence, and what new capabilities does it unlock? Conversely, for hidden-state methods, as reasoning chains extend, **how can a model maintain a coherent "state of mind" over vast temporal sequences without succumbing to the bottleneck of ever-expanding memory? Can this temporal evolution be reframed as a form of continuous online optimization, conceptually unifying this horizontal expansion with the iterative vertical refinement seen in activation-based methods?**

While both approaches enhance reasoning capabilities, they differ in implementation requirements and deployment flexibility, offering distinct pathways toward more powerful latent reasoning. The following sections of this paper will describe these parts in detail.

3.1. Vertical Recurrent: Activation-based Methods

Activation-based approaches achieve latent reasoning by creating recurrent computational flows, either through architectural design or training-time manipulation. These methods share a common principle: iteratively refining representations without generating explicit reasoning tokens.

3.1.1. Loop/Universal Transformer Recurrence

Loop-based architectures represent the foundational approach to activation-based latent CoT reasoning, implementing continuous activation propagation across Transformer layers through explicit architectural modifications. These models share a core principle: enabling iterative refinement of hidden states within a single forward pass through layer-wise recurrence. Starting from the Universal Transformer (UT) [22], which pioneered dynamic recurrence over layers with its Adaptive Computation Time (ACT) mechanism, this architectural paradigm has established depth-adaptive reasoning as a viable alternative to traditional fixed-depth transformers. The key innovation lies in treating network depth not as a static hyperparameter but as a dynamic

Architecture	Pre/Loop/Coda	Per-iter input x_t	Hidden state S_t	Dynamic stop	Depth-emb d_t
Universal Transformer[22]	No	$x_t^{l-1} + d_t$	standard unroll	ACT, $\sum_t p_t > \tau$	sinusoidal d_t
CoTFormer[71]	No	S_t^{l-1}, x_t^{l-1}	standard unroll	MoR router g_i	learnable d_t
Recursive Transformer[2]	Optional	x_t^{l-1}	share/refill $\hat{h}$	early-exit, $\max_t \Delta h < \varepsilon$	none
AlgoFormer[29]	Yes	x_t^{l-1}	standard roll	fixed	none
Recurrent-Depth[31]	Yes	x_t^1, x_t^{l-1}	modulo (t mod r) reuse	fixed-point iteration	tried, dropped

Table 1. Comparison of activation-based latent CoT architectures and their key design characteristics, showing the evolution from early monolithic designs to structured Pre/Loop/Coda frameworks with simplified dynamic stopping mechanisms.

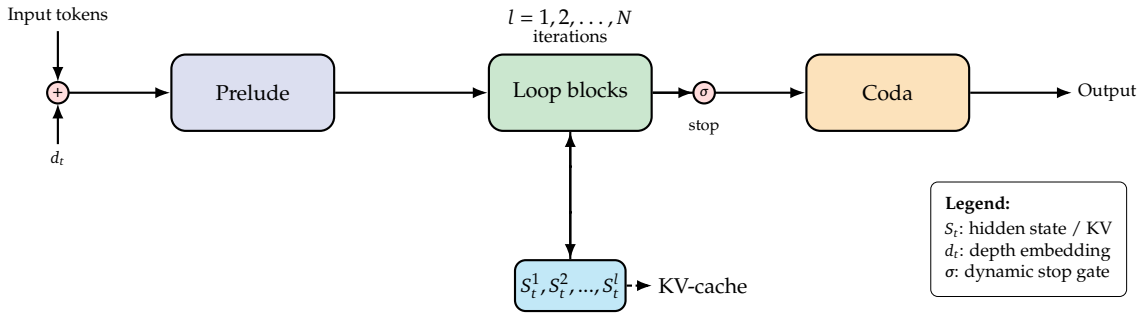

Figure 4. Conceptual diagram of a Pre/Loop/Coda architecture with per-iteration input x_t, hidden state S_t (KV-cache), depth embedding d_t, and a dynamic-stop gate.

computational resource that can be allocated based on task complexity. Extending activation-reuse beyond Universal/Looped Transformers, Zeng et al. [132] introduce a Pondering LM that performs k iterative 'ponder' cycles inside every token prediction. Each cycle converts the model's softmax into a continuous pondering embedding: a weighted sum of all vocabulary vectors, which is fed back via a residual path to refine the hidden state.

Since this seminal work, the field has undergone systematic evolution along several key dimensions, revealing important design principles for latent reasoning architectures (Table 1 and Figure 4).

The Rise of Pre/Loop/Coda Structure Early models like Universal Transformer and CoTFormer [71] adopted monolithic recurrent designs without explicit stage separation. However, recent architectures like Recursive Transformer [2], AlgoFormer [29], and Recurrent-Depth [31] have converged on a three-stage Pre/Loop/Coda structure. This design explicitly separates input encoding (Prelude), iterative reasoning (Loop blocks), and output decoding (Coda), enabling more modular and interpretable computation flows. The modularization of the architecture improves interpretability and facilitates the injection of task-specific priors, such as fixed-point iteration constraints or algorithmic templates, into the reasoning process.

Per-iteration Input and Hidden State Management Input handling strategies vary across models, reflecting different hypotheses about information flow during recurrence. Universal Transformer combines previous layer output x_t^{l-1} with depth embedding d_t. CoTFormer uses both hidden state $\mathbf{S}_t^{l-1}$ and x_t^{l-1}, while Recursive Transformer and AlgoFormer simplify to just x_t^{l-1}. Recurrent-Depth adopts a hybrid approach with both x_t^1 and x_t^{l-1}.

For hidden state management, most models use standard unrolling of KV caches. Notable

exceptions include Recursive Transformer's share/refill mechanism and Recurrent-Depth's modulo-based reuse ($t \bmod r$), which improve memory efficiency through periodic cache recycling, as shown in Table 1. These innovations strike a balance between preserving temporal coherence and managing computational resources.

The Decline of Depth Embeddings Depth embeddings show a clear deprecation trend. Universal Transformer introduced sinusoidal d_t, and CoTFormer experimented with learnable embeddings. However, subsequent models like Recursive Transformer and AlgoFormer completely dropped them. Recurrent-Depth tried but ultimately abandoned depth embeddings, suggesting their limited utility in recurrent architectures despite initial enthusiasm. This trend indicates that explicit positional encoding of depth may be redundant when the architecture inherently encodes iteration count through state evolution.

Simplification of Dynamic Stopping Mechanisms Dynamic stopping mechanisms exhibit a clear trend toward simplicity. Universal Transformer's sophisticated ACT mechanism (with cumulative probability $\sum_t p_t > \tau$) gave way to CoTFormer's MoR router g_i. Recent models adopt even simpler strategies: Recursive Transformer uses early-exit based on change magnitude ($\max_t \Delta h < \varepsilon$), AlgoFormer opts for fixed iterations, and Recurrent-Depth explores fixed-point criteria. This evolution suggests that complex adaptive mechanisms may not justify their computational overhead in practice.

These architectural trends reflect the field's maturing understanding: moving from complex adaptive mechanisms toward stable, modular designs while preserving the core benefit of enhanced reasoning through layer-wise recurrence. The convergence on simpler, more interpretable designs suggests that the key to latent reasoning may lie not in sophisticated control mechanisms but in providing sufficient computational depth with efficient resource management.

3.1.2. Activation with Explicit Hidden-State Feedback

While loop-based architectures refine token representations by rerunning the same set of layers, a distinct family of models *feeds hidden states back into the input stream* between iterations. In these systems the hidden activations themselves become new sequence elements, so each recurrent step simultaneously extends the effective depth **and** exposes internal computation to subsequent attention.

Coconut Proposed by Hao et al. [44], **Coconut** inserts a *continuous thought* vector—the last-layer hidden state of the previous decoding step—as an extra position before the current token. Pondering therefore occurs in latent space without emitting textual reasoning, enabling breadth-first exploration while reusing the same Transformer parameters.

CoTFormer In **CoTFormer** [71], every forward pass first computes preliminary token embeddings; these activations are then *interleaved* back into the sequence and the shared block stack is executed again. Early-exited tokens thus attend to deeper refinements of their own representations, realizing adaptive depth with minimal parameters.

Both models share three properties that distinguish them from "pure" activation-based recurrence:

Key characteristics. Explicit state tokens re-inject hidden vectors as sequence elements, bridging vertical recurrence and horizontal memory; no architectural expansion—the model reuses the same layers so parameter count stays constant while depth grows dynamically; and latent reasoning remains internal, thereby avoiding the latency of producing explicit CoT tokens.

These designs demonstrate that passing hidden states across recurrent hops can unlock stronger reasoning while preserving the efficiency of shared-weight loops, and they foreshadow later hybrids that blend activation and hidden-state paradigms.

3.1.3. Training-induced Recurrence

While architectural recurrence requires explicit structural modifications, an alternative pathway achieves similar computational benefits through specialized training on standard transformer architectures. These methods fundamentally create recurrent activation flows without changing the model's underlying structure, demonstrating that the key insight of iterative refinement can be induced through training alone. This approach is particularly valuable as it enables existing pretrained models to develop latent reasoning capabilities without architectural constraints.

The core principle unifying these methods is the creation of implicit loops in the computation graph: whether by feeding activations back into the model (continuous recurrence), compressing multi-step reasoning into iteratively-processed representations (compressed states), or extending the effective computation depth through strategic token insertion (expanded iterations). All these approaches share the goal of enabling deeper reasoning without explicit architectural loops.

Continuous Activation Recurrence The most direct form of training-induced recurrence involves creating explicit loops of continuous activations. Ref. [44] pioneers this approach with Coconut, which loops the LLM's last hidden state (the "continuous thought") directly back into the model as input for the next step. This mechanism creates a recurrence pattern strikingly similar to architectural approaches like Universal Transformer, but implemented entirely through training. The continuous thought can encode multiple reasoning paths simultaneously, enabling breadth-first search-like exploration in latent space.

Building on this foundation, subsequent work has refined the training methodology while maintaining the core recurrence principle. Shen et al. [91] propose CODI, which frames the problem as learning to align recurrent hidden states through self-distillation. By aligning the hidden activation before the final answer between teacher (with full CoT) and student (with compressed reasoning) paths, CODI effectively learns a fixed-point iteration in activation space. This single-step alignment proves more stable than Coconut's curriculum learning, achieving parity with explicit CoT on GSM8K for the first time among latent methods.

Cheng and Van Durme [16] take a different approach with CCOT, training the model to generate variable-length sequences of continuous embeddings that approximate full reasoning traces. These embeddings function as compressed representations of recurrent computation steps, maintaining the iterative nature while reducing sequence length. The optional decoding back to text preserves interpretability while confirming that meaningful computation occurs in these latent iterations. PCCOT [116] uses Jacobi-iteration allowing parallel continuous thoughts. Building on pause- and filler-token methods that prolong hidden-state computation, System-1.5 Reasoning [109] introduces Depth and Step Shortcuts that dynamically allocate vertical layer depth and horizontal reasoning steps, delivering over 20× faster inference on GSM8K while preserving chain-of-thought accuracy—all without modifying the Transformer backbone.

Compressed State Recurrence Rather than continuous loops, another strategy compresses reasoning steps into discrete or semi-discrete representations that the model processes recurrently. Su et al. [98] replace early CoT segments with discrete latent tokens learned via VQ-VAE, creating "assorted" reasoning that mixes compressed abstract steps with detailed reasoning. This approach effectively creates a hierarchical recurrence where abstract tokens trigger expanded computation in subsequent layers.

Zhang et al. [134] employ "gist tokens" as compression anchors in hidden space. Though these tokens themselves are semantically meaningless, they serve as recurrence checkpoints where the model aggregates and redistributes computational state. The attention mask manipulation enforces that subsequent reasoning depends on these compressed states, creating an implicit recurrence structure through the sequence.

The key insight across these compression methods is that they transform horizontal (sequence-level) reasoning into vertical (depth-level) computation, effectively increasing the recurrence depth available for each logical step.

Iteration Expansion through Strategic Tokens A third category of training-induced recurrence works by expanding the number of implicit iterations through token insertion. This approach recognizes that additional tokens, even without explicit semantic content, can provide more recurrence steps for internal computation.

Pfau et al. [79] demonstrate that even meaningless filler tokens (e.g., ".....") can improve reasoning by simply providing more attention steps, effectively increasing the number of recurrent iterations the model can perform. Goyal et al. [37] refine this with learnable '<pause>' tokens that explicitly signal computation steps, creating trainable recurrence points that the model learns to utilize effectively.

More sophisticated approaches inject structured tokens that organize the recurrence pattern. Wang et al. [110] introduce planning tokens that create a hierarchical recurrence structure, where each planning token initiates a new reasoning loop with specific computational goals. Jin et al. [50] further decompose reasoning into '<memory>' and '<reason>' tokens, creating specialized recurrence patterns for different types of cognitive operations. These structured approaches demonstrate that training can induce not just recurrence, but organized, interpretable recurrence patterns.

Implications and Connections These training-induced methods reveal a fundamental insight: recurrence for reasoning is not solely an architectural property but can emerge from appropriate training objectives. The success of these approaches suggests that standard transformers possess latent capacity for iterative computation that training can unlock. Moreover, the convergence of continuous, compressed, and token-based methods toward similar performance outcomes indicates that the specific implementation of recurrence matters less than ensuring sufficient computational depth for reasoning tasks.

The relationship between these training-induced methods and architectural recurrence is complementary rather than competitive. Future work might explore hybrid approaches that combine architectural loops with training-induced recurrence patterns, potentially achieving the benefits of both explicit structure and learned optimization.

3.1.4. Training Strategies for Recurrent Reasoning

Effectively training models with recurrent activation flows presents unique challenges, as these architectures must learn to leverage iterative computation rather than relying solely on feedforward depth. Researchers have developed specialized training strategies that address both architectural and induced recurrence.

For architectural recurrence, MIDAS [85] proposes a progressive stacking framework to address training stability in loop-based models. It defines a replication operator $\mathcal{M}(f, b)$ that duplicates the middle layers of a base model f by a factor b, enabling gradual depth expansion. Training proceeds through k stages where model depth increases progressively, with each deeper model initialized from the previous stage. This curriculum approach helps models develop stable iterative reasoning patterns. Complementing this architectural focus, Saunshi et al. [86] introduce a looping-inspired regularization that enables even standard Transformers to benefit from recurrence-like properties through a cosine-similarity term $\mathcal{R}_G(k)$ in the loss function. This approach reveals that recurrent behavior can emerge from appropriate training objectives alone.

For training-induced recurrence, Stepwise Internalization [23] pioneered curriculum-based compression of reasoning traces. This technique gradually removes CoT tokens during fine-tuning, allowing models to internalize reasoning patterns into their parameters. This curriculum principle has been widely adopted, notably by Coconut [44] which progressively replaces CoT tokens with continuous thoughts, achieving fully latent inference loops. RELAY [127] takes a more direct approach by explicitly aligning recurrence steps with reasoning steps through a two-stage process: first training looped Transformers with CoT-aligned supervision using loss $\mathcal{L} = \mathcal{L}_{ans} + \lambda \mathcal{L}_{iter}$, then fine-tuning autoregressive models on the generated reasoning chains.

These diverse training strategies converge on key principles: gradual complexity increase, alignment between recurrence depth and reasoning steps, and careful balance between architectural constraints and learned behaviors. The success of both architectural and training-induced approaches suggests that effective recurrent reasoning emerges from the interplay of structure and optimization.

3.1.5. Applications and Capabilities

The true test of recurrent reasoning methods lies in their ability to tackle complex tasks requiring structured, multi-step computation. Both architectural and training-induced recurrence have demonstrated remarkable capabilities across diverse domains.

In algorithmic generalization, recurrent models exhibit unprecedented extrapolation abilities. Schwarzschild et al. [88] and Giannou et al. [35] demonstrate that looped architectures can generalize from small problem instances to significantly harder ones by extending recurrence steps at test time—a property unavailable to static-depth Transformers. This recurrence-controlled scaling mimics human-like progressive problem-solving and has been formalized through theoretical frameworks of looped computation graphs. Similarly, training-induced methods like Coconut show that continuous thought loops can solve logical reasoning tasks (ProsQA, PrOntoQA) through latent breadth-first search, while compressed-state methods achieve parity with explicit CoT on mathematical reasoning (GSM8K).

In symbolic reasoning and graph algorithms, recurrent models bridge neural and algorithmic computation. De Luca and Fountoulakis [21] show that looped Transformers with graph-specific attention heads can simulate classical algorithms (BFS, DFS, shortest-path) within bounded memory. This capability extends to training-induced recurrence: models with planning

tokens [110] demonstrate improved performance on multi-hop reasoning by creating hierarchical computation structures. The decomposition of reasoning into specialized tokens (<memory>, <reason>) [50] further enhances performance on tasks requiring both retrieval and logical inference.

In optimization and meta-learning, works like [13, 30] prove that looped models implicitly implement multi-step gradient descent, revealing deep connections between recurrence and optimization. This theoretical insight explains why both architectural loops and training-induced continuous thoughts converge on similar computational patterns: they are fundamentally performing iterative refinement analogous to optimization algorithms.

These applications demonstrate that recurrent reasoning—whether achieved through architecture or training—provides a general framework for complex computation. The convergence of different approaches on similar capabilities suggests that the key insight is not the specific implementation but ensuring sufficient iterative depth for the task at hand.

3.2. Horizontal Recurrent: Hidden state-based Methods

As previously mentioned, activation-based approaches focus on expanding layer depth in networks. However, deeper networks inevitably encounter challenges such as gradient explosion or vanishing. In contrast, the temporal dimension can be readily expanded to millions of tokens. From a theoretical perspective, the temporal dimension can also be conceptualized as a form of depth, which raises an important research question: **How can we effectively expand the latent reasoning process along the temporal dimension?**

The standard Transformer provides a baseline for this horizontal expansion. It handles temporal information by storing all previous token inputs as key-value pairs in what is known as the KV cache. This cache effectively serves as the model's hidden state, preserving a rich history of the sequence. However, this approach has a critical bottleneck: the KV cache grows linearly with the sequence length, leading to unbounded memory consumption that makes processing very long sequences impractical.

To address this challenge, we can compress previous information into a fixed-size vector or matrix, similar to RNNs. When working with hidden states, there are two primary approaches to enhance their expressiveness: (1) the Linear-State recurrence approach, which applies update and decay rules to the hidden states, and (2) Gradient-State recurrence approach, treating hidden states as online-learning parameters and optimizing them using online learning methods. **Notably, although these methods have not yet produced evidence demonstrating enhanced reasoning capabilities, their theoretical properties suggest they may play a significant role in the future, as they represent a form of iterative processing that is conceptually similar to layer stacking.**

3.2.1. Linear-State Recurrence

For the first approach, models such as Mamba-2 [20], GLA [121], RWKV-6 [77], and HGRN2 [80] represent early attempts in this direction. A matrix-valued hidden state S is transmitted and updated along the temporal dimension. At each time step, the hidden state undergoes global decay, followed by updates incorporating information from the current time step.

Remarkably, these diverse linear attention models can be unified under a general framework of associative recurrent neural networks with matrix-valued hidden states [122, 123]. Given a matrix-valued hidden state $\mathbf{S}_t \in \mathbb{R}^{d \times n}$ and current input $\mathbf{x}_t \in \mathbb{R}^d$, these models follow the general

form:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top, \quad \text{(recurrence)} \tag{11}$$

$$\mathbf{o}_t = \mathbf{S}_t \boldsymbol{q}_t, \quad \text{(memory read-out)} \tag{12}$$

where $\bullet$ represents an associative operator (e.g., Hadamard product, matrix multiplication), and $\mathbf{M}_t$, $\boldsymbol{v}_t$, $\boldsymbol{k}_t$, $\boldsymbol{q}_t$ are functions of the current input $\boldsymbol{x}_t$. The use of associative operators enables parallel scan calculations of $\mathbf{S}_1, \ldots, \mathbf{S}_L$, facilitating efficient training. Table 2 illustrates how various models instantiate this framework.

However, a more profound perspective emerges when interpreting this state evolution through the lens of online optimization gradient. A key insight comes from DeltaNet [123], which perfectly exemplifies this duality. While its state update rule has a closed-form algebraic expression (see Table 2 linear recurrent attention part), it is mathematically equivalent to applying a single gradient descent step to an online regression objective $\mathcal{L}(\boldsymbol{S}) = \frac{1}{2}\|\boldsymbol{S}\boldsymbol{k}_t - \boldsymbol{v}_t\|_2^2$.

This gradient-state recurrence view is conceptually transformative. It reframes the temporal evolution of the hidden state $\boldsymbol{S}_t$ as a form of iterative refinement, akin to training a neural network layer. In this sense, the state matrix $\boldsymbol{S}$ is effectively treated as a dynamic, "fast weight" layer that is updated at each step based on a local objective. This perspective conceptually unifies the "temporal" recurrence of hidden-state models with the "depth" recurrence of activation-based models, suggesting a shared underlying principle of iterative processing for latent reasoning.

3.2.2. Gradient-State Recurrence

While linear-state models rely on predetermined decay–add rules, *gradient-state* methods treat the hidden matrix as a set of fast-adapting parameters updated by a learnable optimizer. Each token triggers a lightweight descent step that steers the state toward the current key–value target, allowing the model to internalize task-specific dynamics on the fly. This view shifts the design space from choosing fixed linear kernels to selecting optimization algorithms (SGD, Adam-like, second-order, etc.), opening a rich continuum of memory behaviors governed by learning-rate schedules, momentum terms and higher-order corrections.

This insight paved the way for a second research trajectory that abandons closed-form descriptions entirely, in favor of direct online learning formulations [3–5, 52, 100]. This line of work, progressing from TTT (implementing SGD-like dynamics) [100] to Titans (incorporating Adam-like behaviors) [3] and ATLAS (utilizing Muon optimization principles) [4], formulates the state update explicitly as a gradient-based optimization step. Extending this optimization perspective, Ref. [58] introduce LATENTSEEK, a framework that performs test-time instance-level adaptation by directly optimizing latent representations using policy gradient.Despite their different origins, these approaches converge conceptually and can be understood through the general update rule:

$$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} - \eta_t \nabla_{\boldsymbol{S}} \ell(\boldsymbol{S}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t) \tag{13}$$

While powerful, this approach introduces significant challenges for parallelization. Unlike linear recurrent models that can be parallelized efficiently with a single scan operation, the gradient $\nabla\ell$ at step t depends on the previous state $\boldsymbol{S}_{t-1}$. This inherent sequential dependency prevents parallel computation across the entire sequence length. Furthermore, these recurrent updates are embedded within complex architectural blocks that include standard components like LayerNorm and residual connections, making it difficult to fuse the computation into a single, hardware-efficient kernel.

Method	Unified Memory-update Rule
Linear-State Recurrence	
Linear Attn [53]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
RetNet/Lightning [101]	$\boldsymbol{S}_t = \gamma \boldsymbol{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
GLA [121]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\mathrm{Diag}(\boldsymbol{a}_t) + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
Mamba-2 [20]	$\boldsymbol{S}_t = \alpha_t\, \boldsymbol{S}_{t-1} + b_t\, \boldsymbol{k}_t \boldsymbol{v}_t^\top$
HGRN-2 [80]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\mathrm{Diag}(\boldsymbol{a}_t) + (\boldsymbol{1} - \boldsymbol{a}_t)\boldsymbol{v}_t^\top$
Linear/Gradient-State duality	
DeltaNet [123]	*State-update:* $\boldsymbol{S}_t = \boldsymbol{S}_{t-1}(I - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$ *Optimization:* $\boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \beta_t \nabla_{\boldsymbol{S}} \frac{1}{2} \|\boldsymbol{S}\boldsymbol{k}_t - \boldsymbol{v}_t\|_2^2$
G-DeltaNet [122]	*State-update:* $\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1}(I - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$ *Optimization:* $\boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \beta_t \nabla_{\boldsymbol{S}} \frac{1}{2} \|\boldsymbol{S}\boldsymbol{k}_t - \boldsymbol{v}_t\|_2^2 + \lambda \|\boldsymbol{S} - \alpha_t \boldsymbol{S}_{t-1}\|_F^2$
Gradient-State Recurrence	
TTT [100]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \eta_t\, \nabla_{\boldsymbol{S}} \ell(\boldsymbol{S}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)$
Titans [3]*	$\boldsymbol{S}_t = \alpha_t\, \boldsymbol{S}_{t-1} - \eta_t\, \nabla_{\boldsymbol{S}} \ell(\boldsymbol{S}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)$
Lattice (orth.) [52]	$\boldsymbol{S}_{i,t} = \boldsymbol{S}_{i,t-1} + \alpha_{i,t}\Big(I - \frac{\boldsymbol{S}_{i,t-1}\boldsymbol{S}_{i,t-1}^\top}{\|\boldsymbol{S}_{i,t-1}\|^2}\Big)\boldsymbol{h}_t$
Moneta [5]	$\boldsymbol{S}_t = \mathrm{Norm}_q\big(\alpha_t \boldsymbol{S}_{t-1} - \eta_t \nabla_{\boldsymbol{S}} \ell_p(\boldsymbol{S}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)\big)$
Yaad (Huber) [5]	$\boldsymbol{S}_t = a_t \boldsymbol{S}_{t-1} - \eta_t \begin{cases} \nabla_{\boldsymbol{S}} \ell_2, & \|S(\boldsymbol{k}_t) - \boldsymbol{v}_t\| \le \delta_t \\ \delta_t\, \nabla_{\boldsymbol{S}} \ell_1, & \text{otherwise} \end{cases}$
Memora [5]	$\boldsymbol{S}_t = \mathrm{Softmax}\big(\alpha_t \log \boldsymbol{S}_{t-1} - \eta_t \nabla_{\boldsymbol{S}} \ell_2(\boldsymbol{S}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)\big)$
OmegaNet [4]	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} - \sum_{i=t-c+1}^{t} \gamma_i \nabla_{\boldsymbol{S}} \big\|\boldsymbol{S}_{t-1}\phi(\boldsymbol{k}_i) - \boldsymbol{v}_i\big\|_2^2$
Atlas [4]	$\boldsymbol{S}_t^{\text{aux}} = \theta_t \boldsymbol{S}_{t-1}^{\text{aux}} - \sum_{i=t-c+1}^{t} \eta_i \nabla_{\boldsymbol{S}} \big\|\boldsymbol{S}_{t-1}\phi(\boldsymbol{k}_i) - \boldsymbol{v}_i\big\|_2^2$ $\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \mathrm{NS5}(\boldsymbol{S}_t^{\text{aux}})$

* Titans omits momentum and norm-adaptation terms for brevity.

Table 2. **Unified hidden-state and optimization-based memory updates.** Each model is a recurrence on matrix memory $\boldsymbol{S}_t$: apply decay, projection or an optimization step to $\boldsymbol{S}_{t-1}$, then add an outer-product or gradient correction (read-out: $\boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t$). *Symbols:* For uniformity, α_t generally denotes the gate controlling the retention of the previous state, while η_t denotes the learning rate. An important exception is **DeltaNet** and **Gated-DeltaNet**, whose learning rate or writing strength is denoted by β_t. Additionally, γ_i is the weight for each token's gradient in the **OmegaNet** context window, and θ_t is the momentum decay term in **Atlas**. All these parameters are data-/channel-dependent scalars, typically in $(0, 1)$. δ_t is the Huber threshold; $\nabla \ell_p, \nabla \ell_1, \nabla \ell_2$ are gradients w.r.t. $(\boldsymbol{k}_t, \boldsymbol{v}_t)$; $\mathrm{Norm}_q(\cdot)$ is q-norm normalization (Moneta); $\phi(\cdot)$ denotes a polynomial/high-order feature map; NS5$(\cdot)$ is the Muon/Newton–Schulz 2$^{\text{nd}}$-order update (Atlas); $I - \frac{\boldsymbol{s}\boldsymbol{s}^\top}{\|\boldsymbol{s}\|^2}$ is the orthogonal projector in Lattice.

To overcome these limitations, a practical solution known as **chunk-wise parallelization** has been widely adopted [3, 100, 122]. This strategy balances expressiveness and efficiency:

- **Intra-chunk Parallelism**: Within a small, fixed-size block (chunk) of the sequence, the gradients for all tokens are computed in parallel with respect to the *same initial state* (the final state of the previous chunk). This breaks the sequential dependency within the chunk, allowing for efficient, batched computation.
- **Inter-chunk Recurrence**: The overall sequential nature of the model is maintained between chunks. The final state of one chunk is passed recurrently to become the initial state for the next, forming a chain at the chunk level.

Extending the optimization perspective beyond internal state updates, Zhu et al. [140] introduce Soft Reasoning, which treats the first-token embedding as a controllable latent variable. By injecting Gaussian noise and maximizing an Expected-Improvement objective via Bayesian optimization, the method dynamically searches the hidden space for a reasoning trajectory.

Although current research has not yet produced evidence demonstrating enhanced reasoning capabilities in these models, their theoretical properties suggest significant potential, particularly for enabling self-iteration in the absence of input tokens.

3.2.3. Training-induced Hidden-State Conversion

Building on the success of *training-induced recurrence* for activation-based models, a parallel line of work shows that **fixed-architecture Transformers can be *converted*, rather than redesigned into hidden-state (RNN/SSM) models through targeted fine-tuning or distillation**. These methods preserve most of the teacher's parameters while replacing quadratic self-attention with sub-quadratic mixers that maintain a *single* recurrent state, thereby inheriting constant-memory inference.

Cross-architecture distillation. Earlier "Transformer-to-RNN" (T2R) conversions replaced softmax with trainable linear kernels but required heavy retraining. SUPRA [69] refines this idea: starting from strong Llama-2/Mistral checkpoints, it swaps attention for GroupNorm-stabilized linear kernels and fine-tunes on ~20 B tokens, reaching competitive accuracy with only 5% of the cost of pretraining a recurrent model from scratch. MOHAWK [7] introduces a three-phase procedure (matrix–orientation hidden-state alignment knowledge distillation) that transfers a pretrained Transformer into a Mamba-2 state-space model using only 3B tokens, yielding "Phi-Mamba" which outperforms all prior open recurrent LMs of similar size. The same recipe scales to 1–8 B models in Llamba [8], demonstrating that recurrent students can match Llama-3 teachers with 0.1% of original training compute while enabling larger batch sizes and higher.

Low-rank linearization. LoLCATs [135] shows that high-fidelity conversion does not need full-model updates. It first *matches* every attention head with a sliding-window linear mixer (attention transfer), then restores any residual loss with LoRA adapters touching just 0.2% of weights. This two-step "low-rank linearization" narrows the MMLU gap to ≤1% for 8 B models and scales to 70–405 B parameters within a single day of training.

Gated conversions. Liger [56] repurposes the *pretrained* key matrix to build per-channel forget gates, yielding a gated recurrent student that recovers 93% of teacher performance with only 0.02% of the original token budget and no extra parameters beyond LoRA.

4. Mechanistic Interpretability

This section demonstrates the feasibility of Latent CoT and justifies the use of layers as indicators to facilitate the implementation of Latent CoT. As previously discussed, the majority of latent reasoning behaviors in large language models emerge through operations across layers, both in temporal and spatial dimensions. This raises a fundamental question: **Are layers the basic computational units of reasoning?**

Mechanistic Interpretability providing tools like Probing and Circuit Analysis, enables us to shift from observing model behavior in reasoning to understanding its mechanism. This is crucial to unveil the role of Transformer's layers in reasoning. In this section, we first summarize existing work from an interpretability perspective to address **whether layer stacking represents a form of Latent CoT**. Next, we analyze **how layers function as a latent CoT** by examining aspects such as layer specialization and inter-layer information flow. Finally, we illustrate **the limitations of expressing CoT using layer representations**.

4.1. Do Layer Stacks Reflect Latent CoT?

The concept of Chain of Thought (CoT) reasoning allows models to generate sequential thought tokens, giving them more time and computational resources before arriving at an answer. This idea has been influential in shaping new paradigms for scaling inference in "thinking" models, such as OpenAI o1 [49] and DeepSeek's R1 [41]. In parallel, there's growing evidence suggesting that the stacking of layers in neural networks similarly impacts reasoning capabilities, indicating a "layer-based hidden CoT." This relationship between layer depth and latent reasoning is critical for understanding the model's potential reasoning ability.

At a macro level, a series of studies have found **a close correlation between layer depth and the reasoning capabilities of the model**. Yu [128] found that the model's Implicit CoT capabilities are strictly limited by the number of network layers. For a 5-step reasoning task, although intermediate results emerge within some layers, the final reasoning outcome fails to emerge due to an insufficient number of layers. Guo et al. [42] discovered that at least 2-3 layers are required to form a complete two-step reasoning chain within the model. Insufficient layers or inadequate depth in subsequent layers will hinder the ability to perform multi-hop reasoning. In addition, some studies have explored the structural advantages brought by layer depth from the perspective of representational capacity. Saunshi et al. [86] formally establish that any K-layer transformer performing m-step CoT reasoning can be simulated by an (L+O(1)) layer transformer through m iterative forward passes. Merrill and Sabharwal [70] demonstrate that increasing Transformer depth significantly enhances reasoning abilities, enabling complex tasks like language recognition and graph connectivity that fixed depths cannot achieve. This theorem fundamentally establishes that **layer depth serves as the primary bottleneck for latent reasoning capacity**, where the achievable CoT step length scales linearly with layer count.

At a micro level, studies commonly reveal **a clear correspondence between specific layers and tasks within CoT reasoning**. Just like the various steps in CoT, different layers play distinct roles in the reasoning process, while the overall reasoning depth (layer count) influences the final reasoning performance. A series of interpretability studies have revealed significant functional differentiation across layers of varying depths in reasoning tasks [11, 46]. Layer depth affects the completeness of reasoning chains [42], which expand in parallel and grow exponentially [114], with intermediate information being integrated and transmitted across depths [129]. These observations at the micro-level strongly suggest a structured functional differentiation across layers, each performing distinct computational roles analogous to steps in an explicit CoT. To

better understand how this latent chain emerges from layer stacks, it is necessary to delve deeper into the specific mechanisms of layer specialization and inter-layer information flow.

4.2. Mechanisms of Latent CoT in Layer Representation

Following the evidence from the micro-level analysis, we formalize the theory of **Layer Specialization** as a foundational framework for interpreting Latent CoT. This perspective posits that individual layers within Transformer models systematically specialize to support distinct reasoning operations, collectively forming an implicit computational pipeline analogous to an explicit CoT. Next, we articulate the role each layer group (shallow, intermediate, and deep) plays in supporting this latent reasoning structure, followed by a discussion of how information is propagated across these specialized layers.

Theory of Layer Specilization The Transformer model consists of alternating self-attention and feed-forward network (FFN) modules. A natural assumption is that different layers play distinct roles in reasoning tasks [39, 92, 137]. A series of interpretability studies are focusing on uncovering how these layers work together to build and convey the underlying CoT processes. From shallow to deep layers, the model exhibits a clear "division of labor." The reasoning process transitions from specific, local, and syntactic information in the shallow layers to rich semantic integration and the merging of reasoning paths in the intermediate and deep layers. This differentiated structure leads us to consider each layer as the smallest functional unit in the reasoning process.

Shallow Layers: Basic representational processor of Latent CoT. Transformer's shallow layers perform initial text processing, laying the groundwork for higher-level semantic analysis and reasoning. Functionally, the shallow layers primarily process local information, syntactic structures [54], and surface patterns [34], perform initial data transformations [14], and form early circuit primitives [60, 105, 107]. Additionally, studies indicate that shallow layers are responsible for storing and recalling factual knowledge [96, 120] and bridging entity parsing in multi-hop reasoning tasks [9, 89, 120]. In summary, shallow layers are crucial for processing fundamental information and factual knowledge, with their ability to establish bridging variables directly influencing the model's reasoning performance.

Intermediate Layers: Core of Latent CoT. Intermediate layers play a pivotal role in complex, multi-step reasoning tasks for the following reasons: (1) Intermediate layers form specialized sub-circuits dedicated to reasoning functions, (2) Intermediate layers exhibit superior representational capabilities, and (3) Activations in intermediate layers have a decisive impact on reasoning outcomes.

Intermediate layers contain specific, identifiable computational sub-circuits specialized for distinct reasoning sub-tasks. These circuits typically involve coordinated interactions between attention heads and MLP modules. Wang et al. [108] reverse-engineer the internal algorithm by which GPT-2 identifies indirect objects in sentences. They identify a mid-layer attention sub-circuit responsible for entity tracking and pronoun resolution, showing that intermediate layers carry out essential structured reasoning. Similarly, a series of studies have identified potential reasoning circuits within the intermediate layers [43, 95, 108, 117]. The formation of these circuits is emergent, representing efficient computational patterns spontaneously learned by the model from large-scale data [6, 103].

Intermediate layers exhibit unique characteristics in representation, not only demonstrating powerful expressive capabilities but also playing a crucial role in knowledge storage and

encoding. The performance of intermediate layer embeddings can exceed that of final layer embeddings by up to 16% in text embedding tasks, and show consistency across different model architectures and scales [96]. Some researchers believe that this powerful representation capability stems from the objective function used during pretraining. The autoregressive paradigm induces an information bottleneck at intermediate depths of the model, forcing it to distill the most essential and salient information [57, 108].

Intermediate layers have a causal influence on final reasoning outcomes. Correct activation of these layers is necessary for the model to produce valid inferences. A series of studies identify specialized neurons in intermediate layers and perform causal interventions. They find that enhancing activations significantly improves reasoning performance, while suppressing activations leads to a decline in reasoning ability [31, 111]. Intermediate layer representations, acting as bridging entities, also play a causally critical role in multi-step reasoning outcomes [120]. The functional specialization of intermediate layers makes their correct activation critically decisive for the final reasoning outcomes. For example, Ref. [62] traced failures in multi-hop reasoning to specific Attention modules in the intermediate layers that improperly handled implicit reasoning steps. By successfully "patching" these modules to correct the reasoning, they provided strong causal evidence for the functional specialization of these intermediate layer circuits.

Deep Layers: Output Refinement and Decision-making of Latent CoT. The deep layers of Transformer models lie at the end of the information processing flow, play a pivotal role in **output optimization and decision-making**. Deep layers receive rich representational information from intermediate layers and perform semantic transformation tailored to specific downstream tasks [57, 96], performing more complex logical integration and determine the final answer [27, 43].

However, several layer pruning studies indicate that deeper layers exhibit characteristics such as **poor training performance, limited functionality, and reduced representation learning capabilities** [19, 90, 130]. Existing research attributes this degradation to variance issues in Pre-Layer Normalization and the frequent degeneration of attention matrices. Sun et al. [99] suggest that the exponential growth of output variance in Pre-LN and derivatives approaching the identity matrix in deeper layers are the main causes of layer degradation. Sanyal et al. [84] found that attention matrices in deeper layers frequently degenerate, often collapsing into nearly rank-one single-column patterns. We believe that maintaining the “effectiveness” of each layer during pre-training is crucial. Enhancing the functionality of layers, especially deep layers, is a future direction to improve the model’s reasoning abilities.

Theory of Information Flow Given the layer specialization, the flow of information across these layers is crucial for reasoning process. Stolfo et al. [97] quantify the indirect contributions of MLP and attention modules to clarify internal information flow pathways in LLM during arithmetic tasks. The results highlight the crucial role of the attention mechanism in inter-layer information flow during reasoning, which transmits computational information from early processing layers to the final token. Wang et al. [106] discover a “generalizing circuit” emerging during the grokking process. This circuit enables cross-layer information flow, with lower layers extracting bridge entities and higher layers conducting reasoning. Yu et al. [129] present a neuron-level investigation into the logits flow of LLMs during multi-hop knowledge prediction. With "back attention" mechanism, hidden information can be effectively transmitted from higher layers to lower layers, enhancing model’s reasoning ability. Further research substantiates this by analyzing the "embedding trajectory" across all model layers. One study [113], which terms this the "Chain-of-Embedding," shows that the trajectory’s geometric shape can distinguish

correct from incorrect answers, enabling output-free self-evaluation. Another study [112] uses trajectory "volatility" to detect out-of-distribution mathematical problems, finding that models show an "Early Stabilization" in their reasoning path for familiar tasks but not for unfamiliar ones. Both studies confirm that the vertical, layer-by-layer processing of LLMs contains a rich, interpretable information flow analogous to a latent chain of thought.

4.3. Turing Completeness of Layer-Based Latent CoT

Turing completeness is a fundamental concept in theoretical computer science. It describes the ability of a system to perform any computation that can be performed by a universal Turing machine. A computational system is considered Turing complete if it can simulate the computational process of any Turing machine. In this section, we first attempt to answer **whether the Vanilla Transformer is Turing complete**. Next, we summarize **what modifications are needed to make the Transformer achieve Turing completeness**.

Proof of Turing completeness in model architectures Before the emergence of Transformers [104], Recurrent Neural Networks (RNN) [28, 51] were the dominant architecture for processing sequential data. Owing to their inherent recursive nature, RNNs were theoretically proven to be Turing complete as early as 1996, setting a precedent for neural networks to achieve universal computational capabilities [94]. Subsequently, LSTM [45] and GRU [17] were proposed to address the vanishing gradient problem in RNNs, enabling more stable memory states over long sequences.

A series of research efforts have attempted to prove the Turing completeness of Transformers from an architectural perspective under certain assumed constraints. Pérez et al. [78] formally proved for the first time that **the Transformer architecture is Turing complete**, possessing the universal capability to execute any computable function. However, the validity of this proof relies on three crucial theoretical assumptions: Arbitrary Precision, Positional Encodings, and Hard-Max Attention. Following this idealized and groundbreaking proof, more researchers began to consider the conditions under which a Transformer can achieve Turing completeness. Further, Li and Wang [59] proved for the first time that Turing completeness can be achieved under constant numerical precision. This study directly addresses the controversial assumption of infinite precision from earlier proofs, bringing the theoretical model closer to the computational constraints of the real world.

Proof of Turing completeness with Chain-of-Thought Additionally, another research path focuses on achieving more universal computational capabilities through CoT reasoning. Functionally, CoT transforms the Transformer from a limited context window into a dynamic computational tape. The model employs an autoregressive approach, writing each step's calculation result on a notepad and reusing the intermediate results in subsequent calculations. Qiu et al. [81] proposed that **"prompting is Turing complete".** They demonstrate that a single, finite-sized Transformer, as long as it is given a suitably constructed prompt, can compute any computable function.This is the first time the Turing completeness of Transformers has been revealed from the perspective of prompts. Li et al. [63] discovered that a Transformer with constant depth can simulate a Boolean circuit of size T, provided it is allowed to perform T-step CoT reasoning. These studies on the Turing completeness of CoT indicate a shift in the definition of general computation. Generality does not necessarily need to be embedded within the model architecture; it can also be achieved through interaction paradigms using fixed-depth models.

Enhancing Transformer for Turing Completeness Beyond theoretical proofs, a series of studies have enhanced the expressive power of Transformers through architectural modifications, aiming to approach their theoretical limit of Turing completeness. A series of studies have introduced recurrent mechanisms to break through the fixed depth constraints of Transformers, as discussed in Section 3. Additionally, some studies have incorporated external memory into Transformers [10].

A Unifying View of Implicit and Explicit Reasoning The reasoning process of Transformers can be viewed as "thought unfolding" across two dimensions. The well-known CoT unfolds along the "horizontal" sequence dimension, creating visible reasoning steps. Meanwhile, the network's layer-by-layer computation can be seen as implicit unfolding and refinement of each token along the "vertical" depth dimension. As discussed above, CoT acts as the scratchpad between questions and answers, allowing the model to perform reasoning in an auto-regression mode, theoretically possessing Turing completeness. Meanwhile, each layer of the Transformer represents an implicit reasoning step, progressively optimizing the prediction of the next token. **Thus, both methods represent a form of computational expansion, differing fundamentally in whether they unfold across the sequence or through the network's depth.**

	Operation	**Storage**	**Resource Constraint**	**Optimization Objective**
Standard CoT	Full Model Forward Pass	Explicit Tokens In the Sequence	Context Window	End-to-end Task
Layer-based Latent CoT	Single Layer Forward Pass	Hidden States	Layer Nums	Next Token Prediction

Table 3. A comparison of Standard Chain-of-Thought (Horizontal Expansion) and Layer-based Latent CoT (Vertical Expansion) across key computational dimensions.

Moreover, a series of studies have sought to break the boundary between implicit CoT and explicit CoT. Chowdhury and Caragea [18] propose Universal Transformers (UTs), which approach Turing completeness by implementing adaptive computation depth. The core idea of UTs is to repeatedly apply the same Transformer block across multiple "layers" or computational steps, thereby introducing a form of recurrence into the architecture. Zelikman et al. [131] integrate the CoT between layers and CoT between tokens, allowing for the output of intermediate thought processes among tokens as well. Furthermore, they proposed Fast Quiet-Star, which retains the token-level thinking trace while reducing computational cost. Dong et al. [25] reframed next-token prediction as a reasoning task trained using Reinforcement learning, where the model receives verifiable rewards for correctly predicting the next token for a given context.

5. Towards Infinite-depth Reasoning

Infinite-depth reasoning refers to an AI's ability to devote unbounded "thinking time" to refine and perfect a solution irrespective of output length. In this section we first introduce spatial infinite-depth reasoning and then temporal infinite-depth reasoning. Spatial infinite-depth reasoning is realized by diffusion models that begin with a fully masked or noisy draft of the entire output and iteratively denoise it in parallel: each pass has bidirectional access to the full context, enabling global planning, logical consistency across distant segments, and iterative self-correction, with the number of refinement steps adjustable at inference time to trade speed

Method	Unified Latent-Update Formula
Masked Diffusion Models (Temporal-only)	
• D3PM [1], SEDD [66], RADD [76] • MD4 [93], Simple-MDM [83], MDM [74] • MMaDA [119], IRED [26]	$\boldsymbol{x}_{t+1}^{l}(i) = f(\boldsymbol{x}_{t}^{l}(i))$
Masked Diffusion Models (With Cache)	
• LLaDA [75], dKV-Cache [67], DoT-SEDD [124] • dLLM-Cache [65], d1-LLaDA [138], DCoLT [48] • LLaDA 1.5 [139], MGDM [125]	$\boldsymbol{x}_{t}^{l+1} = f_{\tau}(\boldsymbol{x}_{t}^{l}, \mathbf{S}_{t}^{l})$ $\mathbf{S}_{t+1}^{l}(i) \approx g_{\tau}(\boldsymbol{x}_{t}^{l}(i), \mathbf{S}_{t}^{l}(i))$
Embedding-based Diffusion Models	
• Diffusion-LM [61], CDCD [24] • Plaid [40], DoT-Plaid [124] • TESS [68], TESS 2 [102], Bit Diffusion [15]	$\boldsymbol{x}_{t+1}^{l} = f(\boldsymbol{x}_{t}^{l}, \epsilon_{t})$
Hybrid AR-Diffusion Models	
• DiffuLLaMA [36], Dream [126] • L2D [12], Gemini Diffusion [33] • Mercury [55]	$\boldsymbol{x}_{t}^{l+1} = f_{\tau}(\boldsymbol{x}_{t}^{l}, \mathbf{S}_{t}^{l})$ $\mathbf{S}_{t+1}^{l}(i) = g_{\tau}(\boldsymbol{x}_{t}^{l}(i), \mathbf{S}_{t}^{l}(i))$ + AR prefix caching

Table 4. Text diffusion models organized by cache integration capabilities, showing the evolution from temporal-only updates to spatial-temporal frameworks with KV cache mechanisms.

for depth of reasoning. Temporal infinite-depth reasoning, by contrast, relies on autoregressive extensions that generate tokens one at a time in a left-to-right stream and can in principle produce arbitrarily long sequences—but their irreversible early decisions can accumulate errors and limit true global coherence.

5.1. Spatial Infinite Reasoning: Text Diffusion Models

Text diffusion models represent a paradigm shift for complex reasoning tasks, offering an alternative to traditional AR generation. Unlike sequential models that generate text token-by-token, diffusion models enable *spatial infinite reasoning* through iterative global refinement. This approach allows models to engage in holistic planning and develop logical connections across the entire reasoning chain simultaneously, overcoming the limitations of sequential generation where early decisions become irreversible constraints. The connection between diffusion models and infinite depth reasoning lies in their iterative refinement capacity. While traditional models are constrained by fixed computational depth, diffusion models can theoretically refine reasoning through unlimited denoising steps. Each step provides additional reasoning depth, allowing progressive elaboration from high-level plans to detailed solutions.

We organize text diffusion models into three architectural families: *Masked Diffusion Models* that enable bidirectional context awareness, *Embedding-based Diffusion Models* that preserve structured reasoning while enabling global refinement, and *Hybrid AR-Diffusion Models* that combine diffusion and AR paradigms.

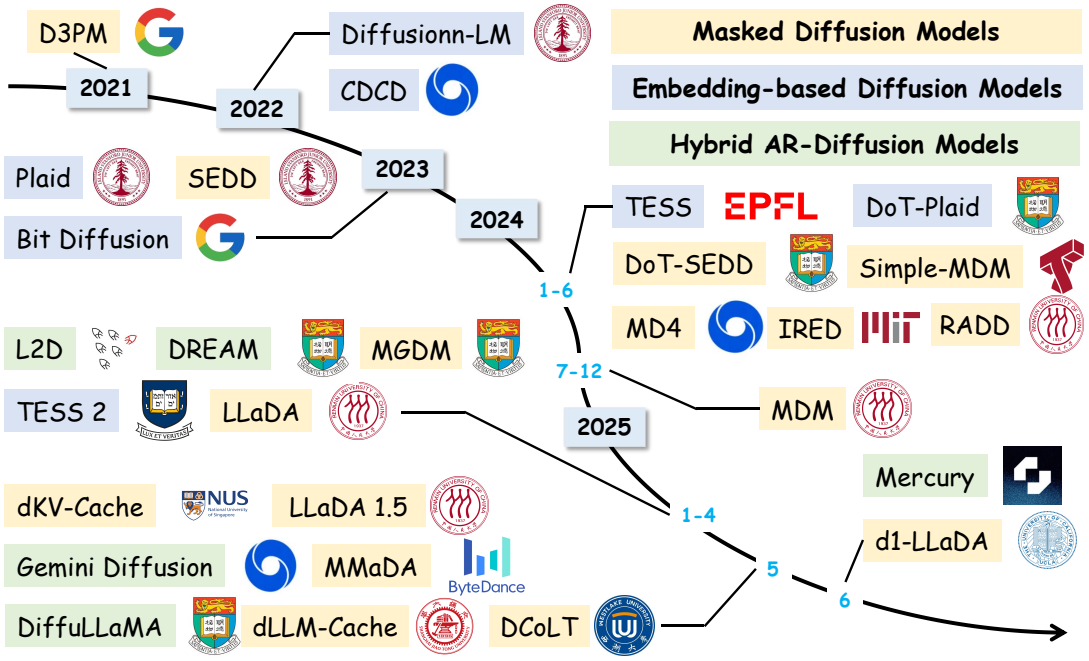

Figure 5. An evolutionary graph of the text diffusion models, including three architectural families: Masked Diffusion Models, Embedding-based Diffusion Models, and Hybrid AR-Diffusion Models.

5.1.1. Masked Diffusion Models

Masked Diffusion Models (MDMs) exemplify spatial reasoning in text generation. These models operate on complete text sequences where tokens are initially masked, requiring simultaneous prediction of all missing tokens based on bidirectional context. This provides full access to the entire information landscape at each denoising step.

MDMs adopt a latent update mechanism driven by an explicit token-level mask M_t at each step t, the corresponding unified latent-update formulas are described in Table 4. For **temporal-only MDMs**, the latent update formula, $\boldsymbol{x}_{t+1}^{l}(i) = f(\boldsymbol{x}_{t}^{l}(i))$, describes how the representation of an individual token, $\boldsymbol{x}_{t}^{l}(i)$, is updated from denoising step t to $t+1$ within a specific layer l. This indicates a direct token-level update, where the model's focus is on iteratively refining the masked parts of the sequence. For **MDMs with cache**, two formulas describe the process. The first formula, $\boldsymbol{x}_{t}^{l+1} = f_{\tau}(\boldsymbol{x}_{t}^{l}, \mathbf{S}_{t}^{l})$, describes the temporal transformation. It shows that at denoising step t, the output $\boldsymbol{x}_{t}^{l+1}$ for layer $l+1$ is generated by a function f_{τ} that takes the current token representations $\boldsymbol{x}_{t}^{l}$ and the current KV-cache $\mathbf{S}_{t}^{l}$ as input. This indicates that the Transformer block's processing for generating token representations directly leverages the KV-cache for spatial context.

The iterative unmasking process enables sophisticated reasoning capabilities impossible in sequential generation. Some pioneers provide a strong foundation for masked diffusion models and supporting reasoning tasks through better input, intermediate steps, and outputs. D3PM [1] goes beyond corruption processes with uniform transition probabilities, while SEDD [66] introduces the EBLO loss that naturally extends score matching to discrete spaces. Further refinements, like RADD [76], MD4 [93], and Simple-MDM [83], have streamlined training through hybrid masked losses, facilitating conversion of encoder models like BERT into effective generative reasoning systems. Besides, MMaDA [119] adopts a unified diffusion

architecture for multi-modal reasoning and aligns reasoning processes between textual and visual domains. Research has shown that MDMs can be scaled effectively, achieving strong performance and efficiency [74]. IRED [26] framed reasoning as an energy minimization process implemented through diffusion models. This approach enables iterative refinement from vague reasoning paths to precise solutions, particularly effective for multi-constraint problems. Energy diffusion demonstrated significant advantages over traditional methods in complex reasoning tasks. The LLaDA model [75] uses discrete random masking, enabling sophisticated capabilities like reverse-order reasoning. To accelerate masked diffusion language models, dKV-Cache [67] introduces a delayed and conditioned key–value caching strategy that achieving up to 2-10× inference speedup. dLLM-Cache [65] introduces an adaptive caching strategy achieves up to 9.1× speedup over the standard inference method of LLaDA [75]. DoT-SEDD [124] subsequently generalized chain-of-thought (CoT) reasoning to the MDM framework, enhancing coherence and accuracy through natural self-correction, with particular strengths in mathematical reasoning. The framework has been extended through Multi-Granularity Diffusion Modeling (MGDM) [125], which prioritizes difficult subgoals and achieves state-of-the-art results on complex planning tasks. d1-LLaDA [138] introduces diffu-GRPO, a lightweight policy-gradient algorithm tailored to masked diffusion models that surpasses SFT across mathematical and planning benchmarks. LLaDA 1.5 [139] advances this line with VRPO, which combines unbiased Monte-Carlo budget allocation and antithetic sampling to sharply reduce the variance of ELBO-based preference optimization. DCoLT [48] applies outcome-based reinforcement learning by using a probabilistic policy or ranking-based Unmasking Policy Module to jointly optimize the entire reasoning trajectory.

5.1.2. Embedding-based Diffusion Models

Embedding-based diffusion models (EDMs) extend the paradigm of spatial reasoning by first mapping discrete token sequences into continuous token embeddings and then operating on these embeddings, where they are disrupted with Gaussian noise. The models denoise every latent vector using bidirectional context, enjoying complete visibility of the information landscape at each refinement step. Although this high-level objective mirrors that of MDMs, EDMs inhabit a fundamentally different design space due to their continuous embeddings formulation.

EDMs achieve latent update by applying noise to all tokens uniformly and allow denoising dynamics to determine recovery, the corresponding unified latent-update formulas are described in Table 4. The formula describes how the representation of the entire sequence's tokens, $\boldsymbol{x}_t^l$, is updated from denoising step t to $t+1$ for a given layer l, enabling iterative refinement within the continuous latent space. The function f represents the diffusion model's core denoising network (typically a Transformer), taking the current noisy embeddings $\boldsymbol{x}_t^l$ and a noise term ϵ_t to compute the denoised embeddings. Conceptually, this process operates on the entire sequence's embedding representation, rather than specific parts of individual tokens or their hidden states.

Early EDM research emphasized controllable generation [61] and sequence-to-sequence tasks [24, 68], as well as efficient latent encodings of discrete sequences [15, 40, 102]. Plaid [40] systematically characterizes the capacity of this model family by deriving empirical scaling laws, closing the compute-efficiency gap with autoregressive language models to 64×. DoT-Plaid [124] subsequently generalized chain-of-thought (CoT) reasoning to the EDM framework, allowing entire reasoning paths to evolve through iterative latent refinement and enhancing coherence and accuracy through natural self-correction, with particular strengths in mathematical reasoning.

5.1.3. *Hybrid AR-Diffusion Models*

The third family explores direct integration of diffusion and autoregressive paradigms, creating hybrid systems that leverage complementary strengths. These models recognize that while diffusion excels at global planning, autoregressive generation remains effective for certain sequential dependencies.

Hybrid AR-Diffusion models integrate autoregressive generation with diffusion-based latent refinement, combining the strengths of sequential coherence and bidirectional reasoning. The corresponding unified latent-update formulas are described in Table 4. The formula $\boldsymbol{x}_t^{l+1} = f_\tau(\boldsymbol{x}_t^l, \mathbf{S}_t^l)$ details the temporal transformation. Here, a Transformer block f_τ refines token representations $\boldsymbol{x}_t^l$ from layer l to $l+1$ at denoising step t. This refinement explicitly uses the current KV-cache $\mathbf{S}_t^l$. Crucially, this temporal update is enhanced by **AR prefix caching**, which brings in forward-context alignment from the already generated text $x_{<t}$. The second formula governs the spatial update of the KV-cache for individual token i. This update is driven by the function g_τ that takes the token's representation $\boldsymbol{x}_t^l(i)$ and its old cache $\mathbf{S}_t^l(i)$ as input. The explicit inclusion of "AR prefix caching" in this formula indicates that the KV-cache update directly incorporates AR prefix caching mechanisms, enhancing the cache with forward context. This allows the model to dynamically stabilize reliable representations, focusing refinement on uncertain tokens while leveraging the strength of pre-existing sequential information.

DiffuLLaMA [36] introduces a continual pre-training approach that converts existing autoregressive models (like GPT-2 and LLaMA) into diffusion models, which provides a powerful and scalable tool for complex reasoning tasks that demand efficient and flexible text processing. L2D [12] uses a modular design integrating a diffusion pipeline with a pre-trained autoregressive model, creating synergy between global reasoning and sequential fluency. The Dream model [126] leverages autoregressive initialization for training stability and context-adaptive noise scheduling. By leveraging a diffusion method for parallel, coarse-to-fine token generation, commercial frameworks such as Gemini Diffusion [33] and Mercury [55] significantly boost the speed and efficiency of code processing in large language models. This provides a more effective solution for latency-sensitive reasoning tasks like chain-of-thought and agentic workloads. These hybrid approaches represent a promising direction, acknowledging that different reasoning aspects may benefit from different computational paradigms.

5.2. The optimization-Based Perspective: Trading Time for Depth

The optimization-based perspective introduced in Section 3.2.2 suggests that **time itself can be traded for network depth**. When the hidden state $\mathbf{S}_t$ is updated by a gradient-like rule $\mathbf{S}_t = \mathbf{S}_{t-1} - \eta_t \nabla_{\mathbf{S}} \ell(\mathbf{S}_{t-1}; \mathbf{k}_t, \mathbf{v}_t)$, each additional token performs one extra step of a (stochastic) optimizer that refines an implicit layer. Consequently, processing a longer sequence is mathematically equivalent to running the same layer for more optimization iterations, thereby yielding **greater reasoning depth without adding parameters**. This observation converts the long-context challenge into a new question: **how can we instantiate a *network of unbounded depth* that remains trainable and efficient?**

5.2.1. *Towards an 'Infinitely Long' Optimizer Network*

Recent work pursues three complementary strategies:

Infini-attention: Munkhdalai et al. [72] attach a compressive memory to every Transformer block. Each incoming segment updates this memory via a linear–delta rule that asymptotically approaches the fixed-point of an associative array, allowing the model to stream *infinitely* long inputs with $O(1)$ memory. However, a reproduction [73] attempt documented significant practical challenges with this approach. Their key finding was that the model's long-context performance degraded as the number of memory compression steps increased. The authors also reported severe convergence issues, particularly with the gating parameters that balance local and compressed memory, ultimately concluding that the technique was not reliable for extending pretrained models. This empirical evidence suggests that while the idea of memory compression is promising, the specific mechanism in Infini-attention may not be effective in practice, and other methods like rope scaling or Ring Attention are currently more viable options.

Test-time training (TTT) and its descendants: Sun et al. [100] pioneered the idea of performing a few steps of SGD on the hidden state during inference. Follow-up models like Titans [3], OmegaNet and Atlas [4], replace first-order updates with Adam- or Muon-style optimizers and introduce chunk-wise parallelism so that 10^6–token streams can be handled on modern accelerators. Empirically, Titans-S (~250 M) already matches a 1.3 B Transformer on 1-shot recall after only ~1 M optimization steps, demonstrating that "deeper through time" can substitute for "deeper via layers".

In contrast to methods relying on frequent, small-batch updates, recent work [136] argues that this strategy suffers from severe computational inefficiency due to low hardware utilization. The proposed solution, Large Chunk Test-Time Training (LaCT), advocates for the opposite: updating "fast weights" using extremely large chunks of data, ranging from thousands to over a million tokens. This large-chunk paradigm dramatically improves hardware utilization without custom kernels and, more importantly, enables the scaling of nonlinear state sizes to a much larger fraction of the model's parameters (up to 40%). This enhanced state capacity, combined with sophisticated optimizers like Muon, has been validated across diverse tasks, including novel view synthesis, language modeling, and autoregressive video diffusion.

Implicit Fixed-Point RNNs: An orthogonal line of work revisits classical RNNs through the lens of *implicit layers*. Schöne et al. [87] show that iterating a state-space block until convergence yields non-linear, non-diagonal transitions that recover the expressivity of general RNNs while retaining training parallelism. Practically, one runs only a small, adaptive number of self-iterations ($\leqslant 16$ for most natural-language tokens), giving another route to unbounded depth: the model simply halts when additional refinement becomes irrelevant.

5.2.2. A Unifying View

All three families embody the same principle:

Depth emerges from optimization over time.

The hidden state plays the role of a "fast-weight" layer whose parameters are refined either **explicitly** (TTT, Titans, Atlas), **implicitly** (fixed-point RNNs), or through an **associative cache** (Infini-attention). Longer sequences therefore unlock deeper reasoning. Crucially, chunk-wise scans and parallel fixed-point solvers keep the wall-clock cost nearly linear, enabling experiments with **million-token** contexts on a single GPU.

6. Discussion and Conclusion

This survey provides a comprehensive overview of Latent CoT and reasoning, an emerging paradigm in AI reasoning. While large language models have demonstrated impressive reasoning using explicit CoT that verbalizes intermediate steps, this approach is limited by the expressive bandwidth of natural language. Latent CoT addresses this by shifting the entire reasoning process into the model's continuous hidden state, aiming to enhance expressive power and performance. By operating in a continuous space, the model is freed from the constraints of a finite token vocabulary and can explore more efficient and powerful reasoning strategies that may not have direct linguistic equivalents.

Latent reasoning methodologies primarily follow two paradigms: vertical and horizontal recurrence. Vertical recurrence, or activation-based methods, expands computational depth by iteratively refining information within the same set of layers, either through explicit architectural loops or induced through specialized training. In contrast, horizontal recurrence, or hidden-state-based methods, expands the model's temporal capacity by evolving a compressed hidden state over long sequences, allowing for the integration of vast amounts of information. These approaches are complemented by mechanistic interpretability research, which examines how different network layers specialize to form an implicit computational pipeline analogous to an explicit CoT.

Notably, this survey does not offer a direct empirical comparison across these varied models. The field is developing rapidly, with different models being created under disparate training conditions—some are pre-trained from scratch, while others are adapted from existing foundation models via continual pre-training. Furthermore, most studies compare their models to non-reasoning LLM baselines rather than to each other. This lack of consistent training methodologies and standardized benchmarks currently makes a direct, apple-to-apples comparison of empirical results challenging. It is our hope that a unified evaluation framework will emerge in the future to enable a clearer assessment of the relative strengths of these approaches.

The survey culminates by exploring the frontier of infinite-depth reasoning, which aims to give models the ability to use unbounded computational steps to refine a solution. Text diffusion models are a key innovation in this area, as they operate on the entire output sequence in parallel. This allows for global planning, iterative self-correction, and logically consistent reasoning processes that are not constrained by sequential, irreversible decisions. By unifying these perspectives, the survey charts the conceptual landscape of latent reasoning and points toward future directions in advanced AI cognition.

References

1 Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

2 Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. *arXiv preprint arXiv:2410.20672*, 2024.

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

4 Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025.

5 Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025.

6 Leonardo Berti, Flavio Giorgi, and Gjergji Kasneci. Emergent abilities in large language models: A survey. *arXiv preprint arXiv:2503.05788*, 2025.

7 Aviv Bick, Kevin Li, Eric Xing, J Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. *Advances in Neural Information Processing Systems*, 37:31788–31812, 2024.

8 Aviv Bick, Tobias Katsch, Nimit Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. *arXiv preprint*, cs.LG, 2025. URL https://arxiv.org/abs/2502.14458.

9 Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. Hopping too late: Exploring the limitations of large language models on multi-hop queries. *arXiv preprint arXiv:2406.12775*, 2024.

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

11 Vivien Cabannes, Charles Arnal, Wassim Bouaziz, Xingyu Yang, Francois Charton, and Julia Kempe. Iteration head: A mechanistic study of chain-of-thought. *Advances in Neural Information Processing Systems*, 37:109101–109122, 2024.

12 Edoardo Cetin, Tianyu Zhao, and Yujin Tang. Large language models to diffusion finetuning. *arXiv preprint arXiv:2501.15781*, 2025.

13 Bo Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. Bypassing the exponential dependency: Looped transformers efficiently learn in-context by multi-step gradient descent. *arXiv preprint arXiv:2410.11268*, 2024.

14 Siyu Chen, Heejune Sheen, Tianhao Wang, and Zhuoran Yang. Unveiling induction heads: Provable training dynamics and feature learning in transformers. *arXiv preprint arXiv:2409.10559*, 2024.

15 Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning, 2023. URL https://arxiv.org/abs/2208.04202.

16 Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

17 Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

18 Jishnu Ray Chowdhury and Cornelia Caragea. Investigating recurrent transformers with dynamic halt. *arXiv preprint arXiv:2402.00976*, 2024.

19 Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*, 2025.

20 Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

21 Artur Back De Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers. *arXiv preprint arXiv:2402.01107*, 2024.

22 Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

23 Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

24 Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022. URL https://arxiv.org/abs/2211.15089.

25 Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.

26 Yilun Du, Jiayuan Mao, and Joshua B. Tenenbaum. Learning iterative reasoning through energy diffusion. In *International Conference on Machine Learning (ICML)*, 2024.

27 Subhabrata Dutta, Joykirat Singh, Soumen Chakrabarti, and Tanmoy Chakraborty. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning. *arXiv preprint arXiv:2402.18312*, 2024.

28 Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

29 Yihang Gao, Chuanyang Zheng, Enze Xie, Han Shi, Tianyang Hu, Yu Li, Michael K Ng, Zhenguo Li, and Zhaoqiang Liu. Algoformer: An efficient transformer framework with algorithmic structures. *arXiv preprint arXiv:2402.13572*, 2024.

30 Khashayar Gatmiry, Nikunj Saunshi, Sashank J Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? *arXiv preprint arXiv:2410.08292*, 2024.

31 Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

32 Team Gemini. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. 2025.

33 Team Gemini. Gemini diffusion is our new experimental research model. 2025.

34 Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

35 Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pages 11398–11442. PMLR, 2023.

36 Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=j1tSLYKwg8.

37 Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ph04CRkPdC.

38 Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks, 2025. URL https://arxiv.org/abs/2308.07037.

39 Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024. URL https://arxiv.org/abs/2403.17887.

40 Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-based diffusion language models, 2023. URL https://arxiv.org/abs/2305.18619.

41 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

42 Tianyu Guo, Hanlin Zhu, Ruiqi Zhang, Jiantao Jiao, Song Mei, Michael I Jordan, and Stuart Russell. How do llms perform two-hop reasoning in context? *arXiv preprint arXiv:2502.13913*, 2025.

43 Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.

44 Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

45 Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

46 Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. *arXiv preprint arXiv:2310.14491*, 2023.

47 Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for top-tier code large language models. 2024. URL https://arxiv.org/pdf/2411.04905.

48 Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. Reinforcing the diffusion chain of lateral thought with diffusion language models, 2025. URL https://arxiv.org/abs/2505.10446.

49 Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

50 Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. *arXiv preprint arXiv:2411.13504*, 2024.

51 Michael I. Jordan. An outsider's view of neural nets. *Cognitive Science*, 10(1):17–21, 1986.

52 Mahdi Karami and Vahab Mirrokni. Lattice: Learning to efficiently compress the memory. *arXiv preprint arXiv:2504.05646*, 2025.

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

54 Tassilo Klein and Moin Nabi. micse: Mutual information contrastive learning for low-shot sentence embeddings. *arXiv preprint arXiv:2211.04928*, 2022.

55 Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion, 2025.

56 Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. *arXiv preprint arXiv:2503.01496*, 2025.

57 Ge Lei and Samuel J Cooper. The representation and recall of interwoven structured knowledge in llms: A geometric and layered analysis. *arXiv preprint arXiv:2502.10871*, 2025.

58 Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, et al. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space. *arXiv preprint arXiv:2505.13308*, 2025.

59 Qian Li and Yuyi Wang. Constant bit-size transformers are turing complete. *arXiv preprint arXiv:2506.12027*, 2025.

60 Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

61 Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation, 2022. URL https://arxiv.org/abs/2205.14217.

62 Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. Understanding and patching compositional reasoning in llms. *arXiv preprint arXiv:2402.14328*, 2024.

63 Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1, 2024.

64 Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*, 2025.

65 Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching, 2025. URL https://arxiv.org/abs/2506.06295.

66 Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32819–32848, 2024.

67 Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

68 Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew E. Peters, and Arman Cohan. Tess: Text-to-text self-conditioned simplex diffusion, 2024. URL https://arxiv.org/abs/2305.08379.

69 Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

70 William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*, 2025.

71 Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. Cotformer: A chain-of-thought driven architecture with budget-adaptive computation cost at inference. *arXiv preprint arXiv:2310.10845*, 2023.

72 Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 101, 2024.

73 neuralink, Leandro von Werra, and Thomas Wolf. A failed experiment: Infini-Attention, and why we should keep trying?, August 2024. URL https://huggingface.co/blog/infini-attention. Hugging Face Blog post.

74 Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

75 Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2025.

76 Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

77 Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 3, 2024.

78 Jorge Pérez, Javier Marinković, and Pablo Barceló. On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*, 2019.

79 Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NikbrdtYvG.

80 Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

81 Ruizhong Qiu, Zhe Xu, Wenxuan Bao, and Hanghang Tong. Ask, and it shall be given: On the turing completeness of prompting. *arXiv preprint arXiv:2411.01992*, 2024.

82 Haoran Que, Jiaheng Liu, Ge Zhang, Chenchen Zhang, Xingwei Qu, Yinghao Ma, Feiyu Duan, ZhiqiBai, JiakaiWang, Yuanxing Zhang, Xu Tan, Jie Fu, Jiamang Wang, Lin Qu, Wenbo Su, and Bo Zheng. D-CPT law: Domain-specific continual pre-training scaling law for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=JzKFN5fWOk.

83 Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

84 Sunny Sanyal, Ravid Shwartz-Ziv, Alex Dimakis, and Sujay Sanghavi. Inheritune: Training smaller yet more attentive language models. *arXiv preprint arXiv:2404.08634*, 2024.

85 Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank Jakkam Reddi, and Sanjiv Kumar. On the inductive bias of stacking towards improving reasoning. *Advances in Neural Information Processing Systems*, 37:71437–71464, 2024.

86 Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

87 Mark Schöne, Babak Rahmani, Heiner Kremer, Fabian Falck, Hitesh Ballani, and Jannes Gladrow. Implicit language models are RNNs: Balancing parallelization and expressivity. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=5EbiopWH6e.

88 Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34: 6695–6706, 2021.

89 Yuval Shalev, Amir Feder, and Ariel Goldstein. Distributional reasoning in llms: Parallel reasoning processes in multi-hop reasoning. *arXiv preprint arXiv:2406.13858*, 2024.

90 Mani Shemiranifar. Void in language models. *arXiv preprint arXiv:2505.14467*, 2025.

91 Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

92 Guangyuan Shi, Zexin Lu, Xiaoyu Dong, Wenlong Zhang, Xuanyu Zhang, Yujie Feng, and Xiao-Ming Wu. Understanding layer significance in llm alignment. *arXiv preprint arXiv:2410.17875*, 2024.

93 Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

94 Hava T. Siegelmann and Eduardo D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995.

95 Oscar Skean, Md Rifat Arefin, Yann LeCun, and Ravid Shwartz-Ziv. Does representation matter? exploring intermediate layers in large language models. *arXiv preprint arXiv:2412.09563*, 2024.

96 Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

97 Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. *arXiv preprint arXiv:2305.15054*, 2023.

98 DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

99 Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025.

100 Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

101 Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

102 Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. Tess 2: A large-scale generalist diffusion language model. *arXiv preprint arXiv:2502.13917*, 2025.

103 Cheng Tang, Brenden Lake, and Mehrdad Jazayeri. An explainable transformer circuit for compositional generalization. *arXiv preprint arXiv:2502.15801*, 2025.

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

105 PaweĹ Walkowiak, Marek Klonowski, Marcin Oleksy, and Arkadiusz Janz. Unpacking robustness in inflectional languages: Adversarial evaluation and mechanistic insights. *arXiv preprint arXiv:2505.07856*, 2025.

106 Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *arXiv preprint arXiv:2405.15071*, 2024.

107 George Wang, Matthew Farrugia-Roberts, Jesse Hoogland, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape geometry reveals stagewise development of transformers. In *High-dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, 2024.

108 Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

109 Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. System-1.5 reasoning: Traversal in language and latent spaces with dynamic shortcuts. *arXiv preprint arXiv:2505.18962*, 2025.

110 Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wi9IffRhVM.

111 Yifei Wang, Yuheng Chen, Wanting Wen, Yu Sheng, Linjing Li, and Daniel Dajun Zeng. Unveiling factual recall behaviors of large language models through knowledge neurons. *arXiv preprint arXiv:2408.03247*, 2024.

112 Yiming Wang, Pei Zhang, Baosong Yang, Derek Wong, Zhuosheng Zhang, and Rui Wang. Embedding trajectory for out-of-distribution detection in mathematical reasoning. *Advances in Neural Information Processing Systems*, 37:42965–42999, 2024.

113 Yiming Wang, Pei Zhang, Baosong Yang, Derek F Wong, and Rui Wang. Latent space chain-of-embedding enables output-free llm self-evaluation. *arXiv preprint arXiv:2410.13640*, 2024.

114 Zhiwei Wang, Yunji Wang, Zhongwang Zhang, Zhangchen Zhou, Hui Jin, Tianyang Hu, Jiacheng Sun, Zhenguo Li, Yaoyu Zhang, and Zhi-Qin John Xu. Towards understanding how transformer perform multi-step reasoning with matching operation. *arXiv e-prints*, pages arXiv–2405, 2024.

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

116 Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. *arXiv preprint arXiv:2506.18582*, 2025.

117 Wilson Wu, Louis Jaburi, Jacob Drori, and Jason Gross. Unifying and verifying mechanistic interpretations: A case study with group operations. *arXiv preprint arXiv:2410.07476*, 2024.

118 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

119 Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

120 Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.

121 Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

122 Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

123 Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.

124 Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, and Lingpeng Kong. Diffusion of thoughts: Chain-of-thought reasoning in diffusion language models, 2024. URL https://arxiv.org/abs/2402.07754.

125 Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning, 2025. URL https://arxiv.org/abs/2410.14157.

126 Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025. URL https://hkunlp.github.io/blog/2025/dream.

127 Qifan Yu, Zhenyu He, Sijie Li, Xun Zhou, Jun Zhang, Jingjing Xu, and Di He. Enhancing auto-regressive chain-of-thought through loop-aligned reasoning. *arXiv preprint arXiv:2502.08482*, 2025.

128 Yijiong Yu. Do llms really think step-by-step in implicit reasoning? *arXiv preprint arXiv:2411.15862*, 2024.

129 Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. Back attention: Understanding and enhancing multi-hop reasoning in large language models. *arXiv preprint arXiv:2502.10835*, 2025.

130 Shuzhou Yuan, Ercong Nie, Bolei Ma, and Michael Färber. Why lift so heavy? slimming large language models by cutting off the layers. *arXiv preprint arXiv:2402.11700*, 2024.

131 Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

132 Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhiyu Li, and Zhouhan Lin. Pretraining language models to ponder in continuous space. *arXiv preprint arXiv:2505.20674*, 2025.

133 Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, Raven Yuan, Tuney Zheng, Wei Pang, Xinrun Du, Yiming Liang, Yinghao Ma, Yizhi Li, Ziyang Ma, Bill Lin, Emmanouil Benetos, Huan Yang, Junting Zhou, Kaijing Ma, Minghao Liu, Morry Niu, Noah Wang, Quehry Que, Ruibo Liu, Sine Liu, Shawn Guo, Soren Gao, Wangchunshu Zhou, Xinyue Zhang, Yizhi Zhou, Yubo Wang, Yuelin Bai, Yuhan Zhang, Yuxiang Zhang, Zenith Wang, Zhenzhu Yang, Zijian Zhao, Jiajun Zhang, Wanli Ouyang, Wenhao Huang, and Wenhu Chen. Map-neo: Highly capable and transparent bilingual large language model series. *arXiv preprint arXiv: 2405.19327*, 2024.

134 Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *arXiv preprint arXiv:2502.15589*, 2025.

135 Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. Lolcats: On low-rank linearizing of large language models. *arXiv preprint arXiv:2410.10254*, 2024.

136 Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

137 Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. *arXiv preprint arXiv:2409.14381*, 2024.

138 Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning, 2025. URL https://arxiv.org/abs/2504.12216.

139 Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models, 2025. URL https://arxiv.org/abs/2505.19223.

140 Qinglin Zhu, Runcong Zhao, Hanqi Yan, Yulan He, Yudong Chen, and Lin Gui. Soft reasoning: Navigating solution spaces in large language models through controlled embedding exploration. *arXiv preprint arXiv:2505.24688*, 2025.