Biologically Inspired Neuron Structures for Versatile Foundation Models: Augmenting Memory Capacity, Temporal Dynamics, and Information Segregation via Native Dendritic Mechanisms

Kexin Wang[1,2†], Runlin Shi[1,†], Di Shang[1,2], Jianyu Xu[4], Yu Cheng[5], Bo Xu[1,3*], and Guoqi Li[1,2*]

[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Beijing, China
[3]Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Beijing, China
[4]Carnegie Mellon University, Pittsburgh, PA, USA
[5]The Chinese University of Hong Kong, Hong Kong, China
[†]These authors contributed equally to this work.
[*]Corresponding authors (xubo@ia.ac.cn; guoqi.li@ia.ac.cn)

ABSTRACT

Existing large language models (LLMs) built on the Transformer architecture have hit a performance plateau, constrained by diminishing gains from the scaling law and efficiency bottlenecks due to quadratic computational complexity. This bottleneck originates in the architecture's intrinsic limitations. In stark contrast to the human brain's morphologically diverse and dynamically intricate neurons, the Transformer architecture fundamentally originates from the simplistic abstraction of point neurons, evolved primarily through engineering innovations. A fundamental question emerges: under the constraint of a fixed model scale, is it possible that we enhance the model's capabilities by enriching the structure of neurons? To address this, we establish the equivalence between the dynamic characteristics of dendritic neurons and current linear foundation models. According, we propose DendAttn, a dendritic-inspired linear attention architecture integrating sparse activated multi-branch state expansion and multi-compartment block-sparse gating. Our analysis reveals that this dendritic-inspired design yields three pivotal enhancements: augmented memory capacity, complex temporal dynamics, and information segregation. Theoretically, we prove that the information segregation mechanism of dendritic neurons enables multi-task learning to achieve lower loss. Experimentally, our model achieves a $33.7\times$ inference speedup compared to the Transformer. Furthermore, it demonstrates significant advantages in extrapolation ability, language modeling and retrieval task. For multi-task learning, the model exhibited 14.27% improvement in overall task performance. Overall, This work holds significant implications for understanding the role of complex, biologically inspired neuron structures in enhancing the capabilities of AI foundation models.

Introduction

While early AI development drew clear inspiration from the human brain, the evolution of large language models (LLMs) is now defined by a singular focus: scaling model parameters and expanding data size[1–5]. This paradigm, while driving significant progress, has sidelined a critical source of inspiration—the operational mechanisms of the biological brain—with many researchers dismissing meaningful connections between cerebral function and AI advancement[6]. Today, the limitations of this approach are undeniable: as LLMs grow exponentially, performance gains have plateaued, and computational inefficiency has become a bottleneck. In stark contrast, the human brain far surpasses any existing artificial intelligence system in scale, with hundreds of billions of neurons and countless complex neural connections, all while consuming remarkably low energy[7]. These observations offer a profound insight: the future development of large models may need to draw inspiration from the design principles and operational mechanisms of the brain. The brain's neuronal structure benefit from rich morphological and topological complexity, characteristics that current AI models do not exhibit. Thus, we raise a critical question: **could such a complex neuronal structure play a crucial role in enhancing the capabilities of current AI foundation models?**

To answer this question, we first aim to identify a common conceptual ground between neuroscientific mechanisms and foundation models. Although the inspiration from biological neurons has been acknowledged[8,9], how such neuronal computation principles can be systematically related to modern foundation models remains an open challenge. In this context,

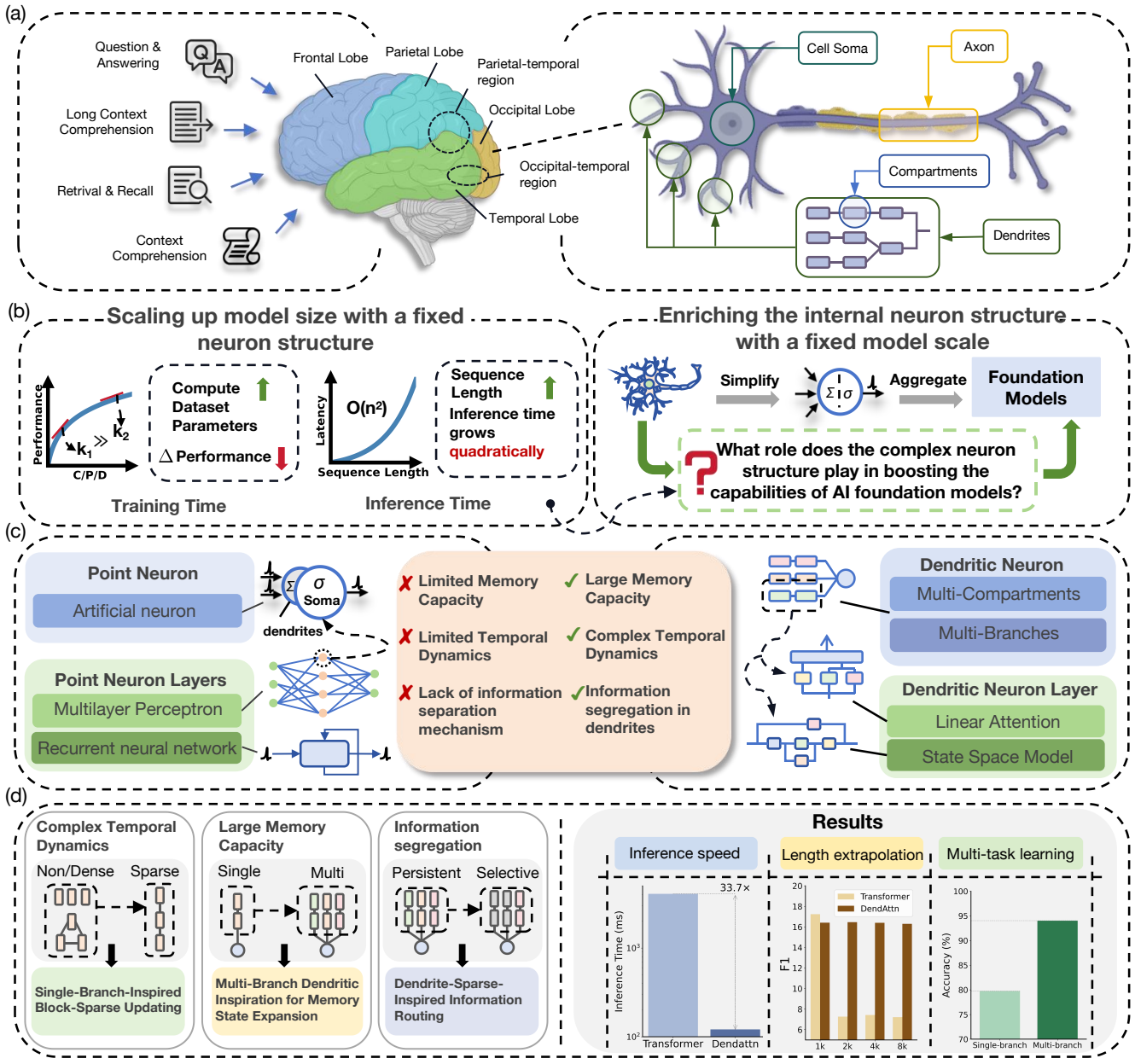

Figure 1. Dendritic-inspired neural architectures. *(a)* Humans are capable of handling complex language tasks, and these functions primarily rely on the brain, whose cortical surface can be divided into distinct functional lobes. At a more fundamental level, these cortical regions are composed of neurons, among which dendritic neurons are the most widely distributed type. Dendritic neurons exhibit highly specialized morphological structures, including a soma, an axon, and multi-branch, multi-compartment dendrites. *(b)* **Left:** Scaling up by increasing model size, training data, or inference computation has driven major progress for years, yet recent scaling laws indicate diminishing returns and motivate new directions. **Right:** Modern foundation models still rely on simplified point neurons. Inspired by the increasing complexity of human neurons, modeling more complex neuronal computations may offer a new route to unlocking future scaling. *(c)* Comparison between artificial point neurons and dendritic neurons. Point neurons and their layered forms (MLPs and RNNs) suffer from limited memory capacity, weak temporal dynamics, and lack of separation mechanisms. In contrast, dendritic neurons leverage multi-compartment and multi-branch structures, supporting large memory capacity, complex temporal dynamics, and information segregation. Their layered organization corresponds to linear attention and state space models. *(d)* **Left:** Dendritic single-branch connection Pattern inspire the design of Block-Sparse Gating for state updating. Memory State Expansion of DendAttn is inspired by multi-branch dendrites. Sparse dendritic activations inspire the design of context-aware routing. **Right:** We present representative experimental results, including a comparison of inference speed with transformers, the performance of DendAttn on long-sequence tasks, and its effectiveness across multi-task evaluations.

we have discovered that biological neurons and linear models exhibit similarities in their dynamic characteristics. Specifically, biological neurons receive currents and use membrane potential to retain dynamic information, while linear models similarly receive tokens and utilize state matrix for memory. Meanwhile, both biological neurons and linear models follow first-order dynamical equations in their state update processes. Building on this insight, we establish a conceptual and mathematical bridge between neuroscience and foundation models, linking neuronal dynamics to the temporal processing module of modern language models. This framework enables a biological reinterpretation of existing foundation models, revealing both their strengths and inherent limitations.

Under this framework, the simplified neuronal assumptions in conventional models, particularly the treatment of neurons as dimensionless point units[10], can be systematically examined. While such point-neuron formulations have enabled scalable implementations in multilayer perceptrons (MLP) and recurrent networks (RNN)[11,12], they fundamentally compress the complex neural computation into oversimplified operations (Fig.1(c))[9,13,14]. As a result, these architectures exhibit: (1) limited memory capacity, (2) weak temporal dynamics, and (3) the absence of mechanisms for information segregation[15,16]. Consequently, they struggle with tasks that demand reasoning over long contexts and precise retrieval[17], as tasks in Fig. 1 (a). Therefore, although these models are efficient in computation, their representational power and generalization ability remain constrained. In contrast, humans can perform complex cognitive tasks, and maintain coherence over long timescales (Fig.1(a))[18]. This remarkable capability stems from the brain's specialized regions and the vast networks of neurons. Among them, dendritic neurons is the most abundant type, serving as the basic computational units, characterized by their complex branching and compartmentalized morphology (Fig.1(a), right)[19,20].

Compared with point-neuron, dendritic neurons exhibit remarkable capability in processing temporal sequences[21,22]. This advantage arises from their characteristic tree-like morphology, which consists of multiple branching dendrites, each branch further divided into smaller compartments[19]. Each dendritic branch operates as a semi-independent computational subunit, capable of performing localized nonlinear integration of synaptic inputs before transmitting the aggregated signal to the soma[20,23]. This hierarchical structure provides significant advantages over point-neuron architectures, offering enhanced memory capacity, richer temporal dynamics, and mechanisms for selective information segregation[14,23,24]. The mathematical formulation of dendritic neurons is computationally equivalent to the basic computational units of state-space models (SSM)[25–28] and linear attention mechanisms[29] (Fig. 1 (c)). Compared with point-neuron-based models such as RNNs or MLPs, these formulations offer superior memory capacity and temporal modeling capability. However, under our established equivalence framework, most existing models capture only the behavior of single-branch dendritic neurons[29,30], thus providing an incomplete abstraction of dendritic computation. As a result, their representational richness and memory size remain limited compared to multi-branch dendritic neurons.

Building on these insights, we construct a large-scale, high-performance foundation architecture inspired by dendritic neurons. Motivated by the limitations of both point-neuron abstractions and partial dendritic formulations, we introduce a dendritic-inspired linear attention mechanism, termed DendAttn. As illustrated in Fig. 1(d,left), DendAttn leverages the full dendritic structure, integrating multi-compartment block-sparse gating, multi-branch state expansion, and context-sensitive routing mechanisms. To validate that our proposed DendAttn embodies the multi-branched, multi-compartmental characteristics of dendritic neurons outlined above, we evaluate DendAttn across foundation language modeling, long-sequence extrapolation, precise retrieval, and multi-task learning, thereby establishing its effectiveness in diverse and challenging scenarios. Moreover, we provide mathematical proof that dendritic neurons endow models with information segregation capability: this mechanism not only reduces multi-task learning loss but also elucidates how dendritic functions boost the generalization of neural networks. Experimentally, as illustrated in Fig. 1(d,right), DendAttn achieves a 33.7× inference speedup at 512k sequence lengths—effectively mitigating the Transformer's core complexity bottleneck, a limitation exacerbated by long inputs, alongside a 14% improvement in multi-task performance. It also robustly extrapolates to sequences several-fold longer than training data, while delivering superior results on language modeling and retrieval benchmarks. This work highlights the transformative potential of biologically inspired, complex neuronal architectures, paving a new path for advancing AI foundation models beyond the constraints of scaling-dependent growth.

Results

Establishing the Equivalence Between the Dendritic Model and the Linear Model

In this subsection, we establish the theoretical foundation through a three-stage analysis. We begin by formalizing the biophysical dynamics of multi-compartment dendritic neurons to capture their spatiotemporal complexity. Next, we investigate the linear attention mechanism. Finally, by comparing dendritic dynamics with linear attention, we establish their mathematical equivalence, showing that biological dendrites can inspire the optimization of linear attention architectures.

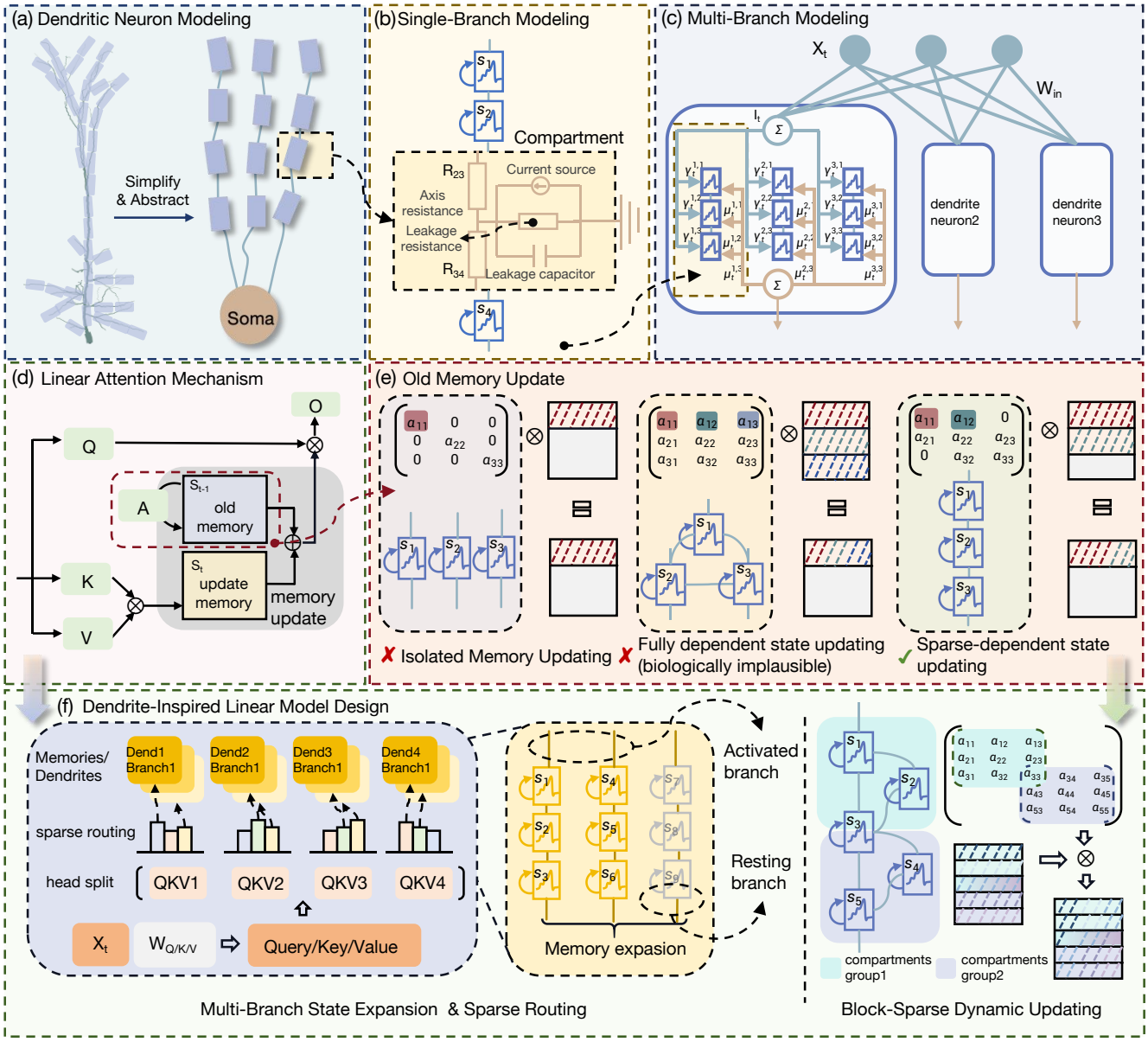

Figure 2. From Dendritic Neuron Modeling to Dendrite-Inspired Linear Architectures. *(a)* Simplification of real dendrites into modelable structures based on cable theory. *(b)* Modeling of single-branch dendrites, with detailed compartment circuits, including leakage resistance, current sources, membrane capacitance, and axial resistance. *(c)* Multi-branch, multi-compartment dendritic neurons arranged in a neural network layer. *(d)* The classical computation flow of linear attention mechanisms, involving query-key-value interactions and memory updates. *(e)* Mapping between single-branch dendritic connectivity and linear model memory updates, highlighting the contrast between biologically implausible isolated or fully dependent updates and the more realistic sparse-dependent state updating strategy. *(f)* Key components of DendAttn. **Left:** Sparse state-space expansion inspired by multi-branch dendrites, where a selective subset of branches is activated, providing expanded memory capacity and information segregation mechanism. **Right:** Block-Sparse state-update scheme inspired by the local-global and dense-sparse connectivity patterns found in single-branch dendrites, providing augmented temporal dynamics.

Single-Branch Multi-Compartment Dendritic Neuron Modeling. Achieving a balance between biologically faithful modeling and computational efficiency remains a fundamental challenge in neuroscience-inspired modeling[10–12,31]. As a result, contemporary neural architectures rely primarily on simple intrinsic neuron dynamics and achieve complexity through large-scale network connectivity. However, this asymmetry between simple internal dynamics and complex external connections neglects the inherent biophysical and morphological complexity of biological neurons[32]. In contrast, dendritic neurons serve as dynamic neuron models whose temporal and structural complexity far surpasses that of point neurons, thereby compensating for their lack of intrinsic complexity[14,23]. Nevertheless, how to model such biologically grounded structures in a manner that enables scalable and efficient computation remains an open and important challenge[28,33].

As shown in Fig. 2(a), the dendritic modeling process begins with morphological abstraction, reducing an anatomically detailed dendritic arbor into a multi-branch representation in which each branch is further divided into multiple compartments. In this way, the anatomical complexity of dendrites is simplified while their essential morphological features are preserved. At the most basic level, dendritic single-branch modeling, as shown in Fig. 2(b), focuses on one branch, each compartment can be represented by an equivalent electrical circuit consisting of a Leakage resistance $R_{j,k}$ of the k-th compartment within the j-th branch, a Leakage capacitor C_L, and a current source corresponding to external input I_j. Adjacent compartments are connected in series through axial resistances $R_{j,(k-1,k)}$ and $R_{j,(k,k+1)}$, coupling the k-th compartment to its neighboring segments. The k-th compartment maintains its membrane potential, denoted as U_k, through the continuous redistribution of ionic, axial, and externally injected currents. For each compartment, the voltage variations arise from the balance among capacitive charging, passive leakage, axial coupling, and external inputs, which can be expressed mathematically as follows:

$$\frac{du_{j,k}}{dt} = \frac{1}{C_L R_{j,(k,k-1)}} u_{j,k-1} - \left(\frac{1}{R_{j,(k,k-1)}} + \frac{1}{R_{j,k}} + \frac{1}{R_{j,(k,k+1)}} \right) \frac{1}{C_L} u_{j,k} + \frac{1}{C_L R_{j,(k,k+1)}} u_{j,k+1} + \gamma_k i_j \tag{1}$$

Rearranging the above expressions yields:

$$\frac{du_{j,k}}{dt} = \frac{u_{j,k-1}}{\tau^f_{j,k}} - \frac{u_{j,k}}{\tau_{j,k}} + \frac{u_{j,k+1}}{\tau^p_{j,k}} + \gamma_k i_j, \tag{2}$$

here the membrane potential of k-th compartment within j-th dendritic branch, denoted by $u_{j,k}$, where $j \in 1,\ldots,J$ and $k \in 1,\ldots,K$, with J and K representing the total number of branches and compartments, respectively. The coupling parameters $\tau^f_{j,k}$ and $\tau^p_{j,k}$ quantify the forward and backward interactions between adjacent compartments, whereas $\tau_{j,k}$ governs the intrinsic temporal evolution of each compartment as determined by its structural properties. The decay coefficient $\gamma_k = \frac{r_k}{C_L}$ specifies how the contribution of the external input current I_j diminishes with increasing distance from the input site. By connecting the compartments along a dendritic branch and applying discretization, the dynamics of the multi-compartment system can be represented in matrix form for a single branch. The matrices $\mathbf{A}$, determined by morphological parameters, encodes the interactions between adjacent compartments. When the membrane potentials of all compartments are consolidated into a vector $\mathbf{U}$, the system is compactly expressed as:

$$\mathbf{U}^j_t = \mathbf{A}^j_t \mathbf{U}^j_{t-1} + \Gamma^{j^T}_t \mathbf{I}_t, \tag{3}$$

where Γ, influenced by distance attenuation factors, is a matrix that scales the external input current vector $\mathbf{I}_t$ across compartments. This formulation generalizes the compartmental continuity equation into a linear dynamical system, providing a compact representation of the single-branch dendritic model. Building on the single-branch dynamics, it is easy to extend the framework to the multi-branch regime (Fig. 2(c)):

$$\begin{bmatrix} \mathbf{U}^1_t \\ \mathbf{U}^2_t \\ \mathbf{U}^3_t \end{bmatrix} = \begin{bmatrix} \mathbf{A}^1_t & 0 & 0 \\ 0 & \mathbf{A}^2_t & 0 \\ 0 & 0 & \mathbf{A}^3_t \end{bmatrix} \begin{bmatrix} \mathbf{U}^1_{t-1} \\ \mathbf{U}^2_{t-1} \\ \mathbf{U}^3_{t-1} \end{bmatrix} + \begin{bmatrix} \Gamma^{1^T}_t \\ \Gamma^{2^T}_t \\ \Gamma^{3^T}_t \end{bmatrix} \mathbf{I}_t \tag{4}$$

Each dendritic branch operates as a linear dynamical system, filtering inputs within its compartmental structure. These parallel branches converge at the soma, where their contributions are integrated into a unified response.

Linear attention as an efficient alternative to softmax attention. Softmax attention[34], as a core component of the Transformer architecture, has been widely recognized for its importance since the early success of the model. This mechanism not only effectively captures long-range dependencies, enabling powerful contextual reasoning, but also theoretically offers infinite memory capacity. However, softmax attention faces significant bottlenecks in computational and storage efficiency. In

terms of computational complexity, its requirements grow quadratically with the input sequence length n. In terms of storage, to maintain global memory capability, softmax attention must cache all key-value pairs (KV Cache) during inference, further limiting its practical application.

To overcome these bottlenecks, researchers have developed a new family of architectures collectively referred to as linear models. Compared to softmax attention, linear models reduce the computational complexity of sequence processing to $O(n)$ and maintain constant memory usage during inference. These classic linear models include SSM[26], linear attention[35], and the RNN without nonlinear constraints between state transitions (Linear RNN) [36]. Although their specific implementations vary, they all follow a unified principle: compressing historical information into a fixed-dimensional latent state, which is updated incrementally as new tokens are processed. This mechanism ensures that computational time and memory requirements grow linearly with the sequence length n. However, in terms of long-sequence memory, linear models still lag behind Transformers.

The typical workflow of the linear attention mechanism is shown in Fig. 2(d). At each time step, the model maintains a hidden state $\mathbf{S}_{t-1}$, which is a compressed representation of all past historical information. This state is updated through a transition operator Λ_t, which controls the decay and propagation of old memories. It is then combined with the current input generated based on $\mathbf{K}_t^T\mathbf{V}_t$ for further updating and compression fusion. The update process is formally expressed as:

$$\mathbf{S}_t = \Lambda_t \mathbf{S}_{t-1} + \mathbf{K}_t^T \mathbf{V}_t, \tag{5}$$

The query $\mathbf{Q}_t$ then interacts with the updated memory $\mathbf{S}_t$ to produce the output $\mathbf{O}_t$. This formulation highlights two essential elements of linear models: (1) the recurrent memory state, and (2) the structured state transition governed by Λ_t.

Equivalence between Linear Models and Dendritic Dynamic. By comparing Eq. 3 and 5, it is evident that the update rule of the linear model is mathematically equivalent to the single-branch dendritic integration mechanism in biological neurons. Not only in form, but also functionally, the two models establish equivalence. Specifically, the dendritic membrane potential state $\mathbf{U}_t$ is equivalent to the memory state $\mathbf{S}_t$ in the linear model, and both state updates are controlled by $\mathbf{A}_t$ and Λ_t, respectively. Additionally, the value vector $\mathbf{V}_t$ in the linear model is equivalent to the input $\mathbf{I}_t$ in the dendritic single-branch, and $\mathbf{K}_t$ is analogous to the control matrix Γ_t that governs the input in the dendrite.

Depending on the specific forms of the state transition matrices $\mathbf{A}_t$ and Λ_t, the equivalence between dendritic computation and the linear model can be further categorized into three distinct types. As illustrated in Fig. 2(e), these three types correspond to different modes of equivalence. When A is diagonal, each state dimension evolves independently, corresponding to isolated dendritic compartments with no lateral communication. This *isolated memory updating* is computationally simple but biologically implausible. At the other extreme, a fully dense A would imply that every compartment interacts with every other at each step, leading to *fully dependent state updating*. While mathematically possible, such dense connectivity is biologically unrealistic. The most plausible regime lies in between: a structured *sparse-dependent updating*, where only neighboring local components influence each other. This aligns with the known anatomy of dendrites, in which localized branches engage in selective communication.

Based on the above analysis, it can be concluded that the state update mechanism of linear models is closely related to the structural organization of dendrites. By comparing the differences in information integration and state update mechanisms between biological dendrites and linear models, the design of linear models can draw inspiration from the structural and functional principles of dendrites, thereby providing new biologically-inspired directions for model architecture optimization.

Experiments

We conducted three sets of experiments to demonstrate the advantages of our dendrite-inspired model, DendAttn. Fig.3 shows how its improved dynamics enhances language modeling and long-range modeling. Fig.4 evaluates the memory gains from its dendritic multi-branch state expansion. Fig.5 examines how its multi-branch information segregation mechanism strengthens multi-task performance.

Dendritic Morphology Augments Temporal Dynamics in Linear Models. Fig. 3 presents a comprehensive comparison measuring the capabilities of language modeling with DendAttn, in comparison with Transformer and representative linear models corresponding to different types of neurons. Models are trained from scratch at 1.3B parameter scale on Slimpajama[37] with sequences up to 2k tokens. Fig. 3(a) illustrates the core design of DendAttn, where dendrite-inspired state expansion and complex temporal dynamics enable improved language modeling and enhanced long-range modeling, while its fixed state space and linear-time complexity provide substantially higher computational efficiency. The subsequent results collectively validate these advantages: (i) commonsense reasoning benchmarks, (ii) efficiency in terms of inference runtime, FLOPs, and GPU memory, (iii) long-context generalization on subtasks of SCROLLS[38] benchmark, and (iv) interpretability via attention-map visualizations contrasting single-branch and multi-branch routing. In particular, we compare DendAttn against

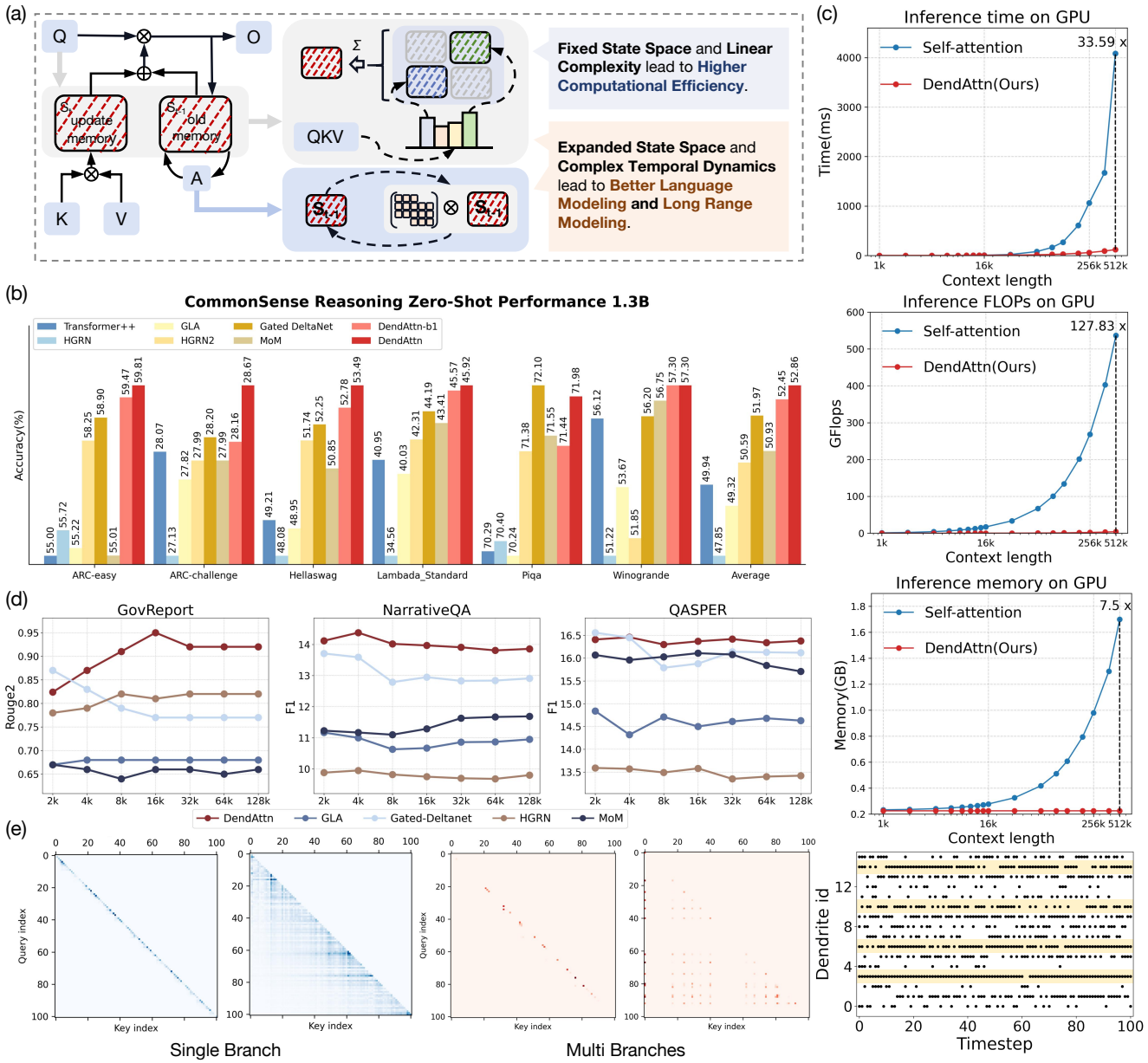

Figure 3. DendAttn: accuracy and efficiency. *(a)* DendAttn (i) maintains a fixed state space and achieves linear-time complexity, resulting in substantially higher computational efficiency, and (ii) employs dendrite-inspired state expansion and complex temporal dynamics that enable improved language modeling and enhanced long-range modeling. *(b)* Zero-shot commonsense reasoning across ARC-easy/ARC-challenge, HellaSwag, LAMBADA, PiQA, and Winogrande at 1.3B scale. DendAttn-b1 denotes the single-branch variant of DendAttn. It serves as an ablation of the block-sparse gating mechanism when compared to Gated-DeltaNet, and as an ablation of the multi-branch structure when compared to DendAttn. *(c)* Efficiency evaluation of DendAttn and self-attention across different context lengths. The three panels respectively depict inference time, floating-point operations (FLOPs), and GPU memory usage as functions of context length, ranging from 1k to 512k tokens. Blue curves correspond to the standard self-attention mechanism, while red curves represent the DendAttn architecture. Each subplot employs logarithmic scaling on the x-axis to visualize the computational trend as sequence length increases. *(d)* Length extrapolation experiments on long-context benchmarks. The figure illustrates the model performance on three datasets of SCROLLS—GovReport, NarrativeQA, and QASPER—as the input context length increases from 2k to 16k tokens. Evaluation metrics include ROUGE-2 for GovReport and F1 scores for NarrativeQA and QASPER. Each line represents a different model architecture. *(e)* Left: attention map visualizations for single-branch and multi-branch dendrites, with Gated DeltaNet representing the single-branch case and DendAttn showing one of the branchs. Right: visualization of the branch selection of each token, where every four rows correspond to the sparsely activated branches within one attention head.

linear models corresponding to multiple types of dendrite-equivalent architectures. Specifically, RetNet[39] and HGRN[40] are treated as *point-neuron* models without explicit dendritic structure; GLA[29], HGRN2[16], and GSA[41] represent *mutually isolated multi-compartments*; Gated DeltaNet[42] and DeltaNet[30] correspond to *single-branch multi-compartment dendrites* with fully connected intra-branch dynamics; DendAttn and MOM[43] exemplifies a *multi-branch multi-compartment dendritic architecture*, combining both structural branching and compartmentalized updates.

Performance. As shown in Fig. 3(b), models with dendritic structures such as GLA clearly outperform point-neuron architectures like HGRN. Among dendritic designs, models with inter-compartment interactions such as DeltaNet and Gated DeltaNet achieve markedly higher accuracy than those with independently updated compartments, such as GLA. Furthermore, multi-branch models demonstrate consistent advantages over single-branch counterparts. DendAttn benefits from a block-sparse gating and finer-grained routing mechanism than MoM, enabling more precise compartmentalization and consequently delivering superior performance. Moreover, compared with Gated DeltaNet, DendAttn-b1 achieves consistently higher accuracy across multiple commonsense reasoning benchmarks, which can demonstrate the advantage of the proposed block-sparse gating. When comparing DendAttn with DendAttn-b1, the performance gap further highlights the contribution of the multi-branch structure, which enhances contextual integration and memory capacity through state expansion and selective state activation. Another essential criterion for evaluating sequential models is their ability to extrapolate beyond the sequence lengths observed during training. We tested DendAttn and competing baselines on three long-document understanding tasks: GovReport, NarrativeQA, and QASPER (Fig. 3(d)). These tasks require integrating information across sequences of up to 128k tokens, extending far beyond the training distribution. DendAttn consistently outperforms other models on the GovReport, NarrativeQA, and QASPER tasks. Its Rouge-2 and F1 scores remain nearly stable, or even slightly improve—across varying input lengths, demonstrating exceptional context extension and memory generalization capabilities. Other linear models (such as Gated-DeltaNet, GLA, and MoM) exhibit lower overall performance during extrapolation, yet maintain relatively smooth curves, indicating a certain degree of length robustness. However, these models still show limited ability in long-range dependency integration and information discrimination, falling short of DendAttn's performance. It is worth noting that the standard Transformer experiences a sharp performance decline as input length increases, making it incomparable in this setting.

Efficiency. A key motivation for linear models lies in their promise of computational and memory efficiency. We therefore systematically compared the runtime, FLOPs, and GPU memory usage of DendAttn against standard softmax attention across varying sequence lengths (Fig. 3(c)). The results demonstrate a striking efficiency advantage for our dendritic formulation. Inference time on GPU scales nearly linearly with context length for DendAttn, whereas self-attention rapidly becomes prohibitive, exhibiting quadratic growth. At 512k tokens, DendAttn is 33.6× faster than softmax attention. FLOPs grow nearly linearly with length, being 127.8× more efficient at long sequences. This demonstrates that DendAttn successfully reduces computational complexity from the traditional $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, effectively mitigating the exponential explosion problem of attention mechanisms in long-sequence inference. Memory footprint on GPU also highlights the scalability advantage: while self-attention memory usage grows linearly with sequence length, DendAttn remains nearly constant, achieving a 7.5× reduction at large contexts.

Visualization. We investigated how dendritic routing impacts the internal structure of attention maps and memory access patterns. We compared single-branch and multi-branch dendritic configurations to highlight the effect of branching on information organization. The left of Fig. 3(e) provides visualizations of attention maps for single-branch (blue) and multi-branch (red) dendritic models. The single-branch model produces densely but dilutedly distributed attention, whereas the multi-branch configuration exhibits sparse and selective patterns, a direct consequence of the routing mechanism that activates only a subset of branches for each input. The visualizations further highlight how multi-branch dendrites extend memory capacity and enable access to earlier context. In single-branch models, memory access is dominated by recent tokens, limiting effective receptive fields. By contrast, the multi-branch configuration distributes memory across branches, allowing retrieval of more distant tokens. This leads to earlier contextual information being retained and reused, thereby supporting stronger extrapolation and improved reasoning. Quantitative analysis confirms that dendritic routing expands both the depth and diversity of retrievable memory traces. Moreover, the rightmost panel of Fig. 3(e) further illustrates dendritic routing behavior: the horizontal axis denotes token IDs, and the vertical axis denotes branch IDs, grouped by four within each head. The visualization reveals the branch selection pattern of tokens across heads, showing that while dendritic choices are broadly dispersed, a clear differentiation emerges between high-frequency and low-frequency branch utilization. This separation suggests that certain dendritic branches specialize in frequently accessed pathways, whereas others act as sparsely activated reservoirs, together contributing to efficient and diversified memory routing.

Dendritic Multi-Branch Augments Memory Capacity in Linear Models. In this section, we explore the memory capacity of multi-branch dendritic models, comparing them to point neurons and single-branch neurons. The models evaluated in this study include point neurons (HGRN), multi-compartment dendrites with no interaction (GLA), multi-compartment dendrites with interaction (Gated Deltanet), and multi-branch multi-compartment dendrites as the baseline (MoM). To assess the memory

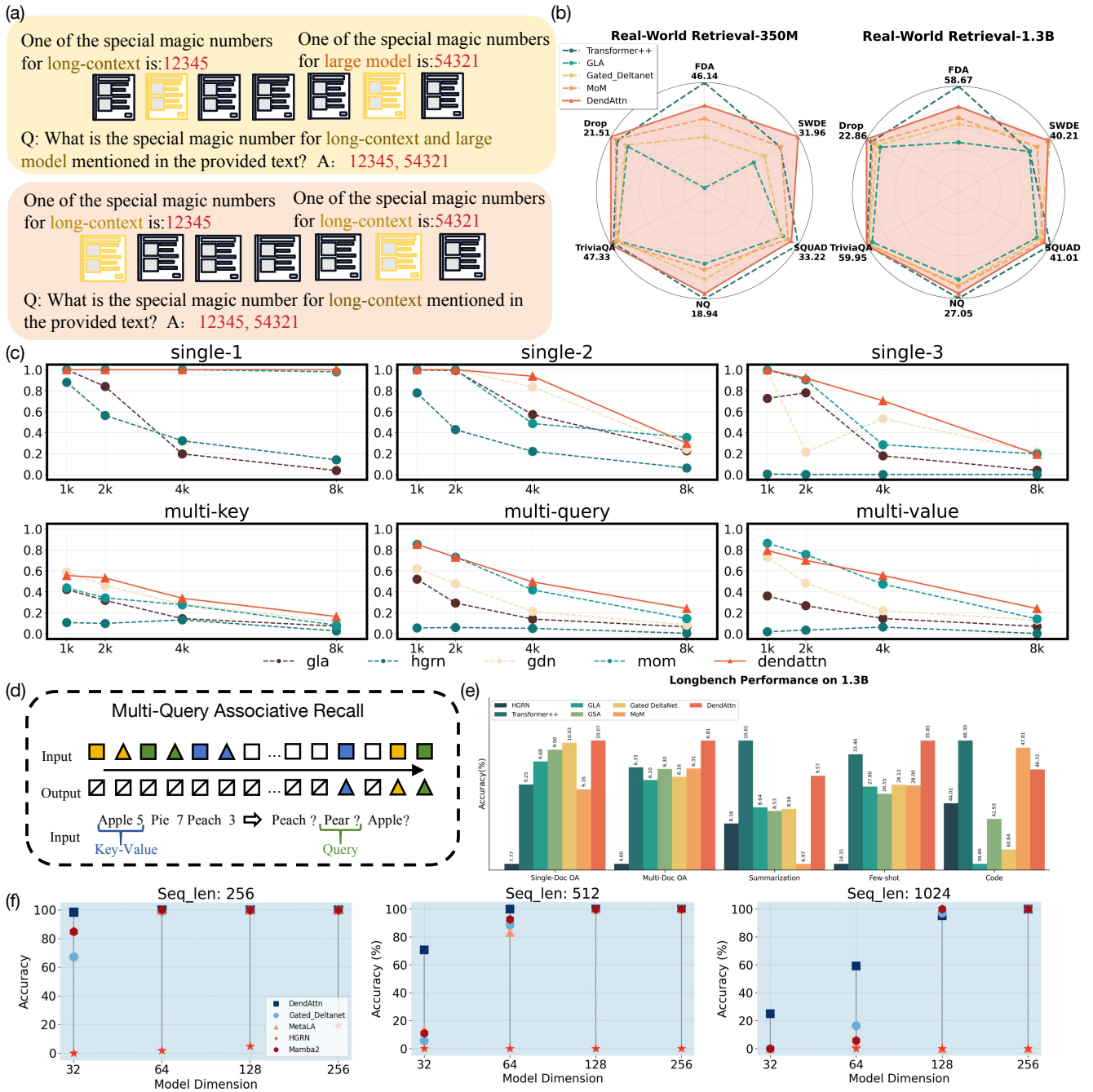

Figure 4. Evaluation of retrieval and associative memory in DendAttn. *(a)* Illustration of NIAH tasks, where models are required to retrieve key-value pairs under single or multiple query conditions. *(b)* Real-world retrieval performance of both 350M and 1.3B scales on FDA, SWDE, SQuAD, NQ, TriviaQA, and Drop *(c)* Performance on the synthetic retrieval benchmark NIAH, covering single-key, multi-key, multi-query, and multi-value settings. Models are trained on sequences of length 2k and extrapolated to 8k. *(d)* Schematic of the Multi-Query Associative Recall (MQAR) task, designed to test the ability to retrieve multiple key–value associations simultaneously. *(e)* LongBench performance comparison on 1.3B models. The figure reports accuracy across five representative long-context tasks, including Single-Document QA, Multi-Document QA, Summarization, Few-shot Reasoning, and Code Completion. *(f)* Experimental results of MQAR across different sequence lengths and model scales. As the sequence length increases and the model size decreases, the task becomes more challenging. The results are shown for sequence lengths of 256, 512, and 1024, where the x-axis denotes model size and the y-axis represents accuracy.

capacity, we employ a variety of synthetic retrieval tasks, including the artificial multi-query associative recall (MQAR) task[44] and the needle-in-a-haystack (NIAH) task[45], which are specifically designed to test memory retrieval efficiency. Additionally, we report results from real-world recall retrieval tasks, as shown in Fig. 4(b). The experiments are further complemented by the evaluation on long-sequence benchmarks, particularly the Longbench task[46,47]. The results confirm that multi-branch dendritic architectures exhibit superior memory retention and retrieval efficiency compared to simpler neuron models, highlighting their potential for advanced memory-intensive tasks.

Synthetic Retrieval Tasks. The Needle-in-a-Haystack (NIAH) task, shown in Fig. 4(a), tests the model's ability to retrieve a value corresponding to a query embedded within a given sentence. The sentence contains a word that serves as the query, and the goal is to identify the associated value based on the key-query relationship. As shown in Fig. 4(c), the horizontal axis represents the test sequence length and the vertical axis denotes accuracy. In the NIAH experiment, as the input length increases from 1k to 8k, all models show a downward trend in performance. However, DendAttn exhibits a relatively smaller decline and consistently maintains a leading position across nearly all tasks. In the single-1 task, DendAttn is able to sustain stable accuracy even at 8k sequence length, whereas most models degrade rapidly beyond 4k, indicating that its dendritic structure provides stronger long-term memory retention in single-target retrieval scenarios. In more complex multi-key, multi-query, and multi-value tasks, conventional linear models and gated structures suffer from severe performance degradation, while DendAttn still retains a clear advantage. This suggests that its dynamic routing between branches and multi-compartment representations effectively mitigate memory interference in long-sequence settings.

The Multi-Query Associative Recall (MQAR) task, depicted in Fig. 4(d), further tests the memory retrieval capacity of the models by providing multiple key-value pairs (e.g., Apple–5, Peach–3) and querying for the associated values. This task requires the model to efficiently retrieve the value corresponding to each query from a set of key-value pairs. The experimental results, shown in Fig. 4(f), demonstrate that DendAttn consistently achieves the highest or near-perfect accuracy across all lengths and dimensions, exhibiting strong associative memory and scalability. In contrast, HGRN and Mamba2 degrade severely even on short sequences, reaching near-zero accuracy. Gated DeltaNet and MetaLA perform reasonably at shorter lengths (256) but deteriorate rapidly as the sequence extends, almost losing retrieval ability at 1024. Notably, when model dimensionality is small (e.g., 32 or 64), DendAttn still maintains high accuracy, whereas other models suffer from capacity bottlenecks.

Real-world Retrieval Tasks. Fig. 4(b) compares the performance of DendAttn with several representative models on real-world question-answering and retrieval tasks. Evaluations are conducted at two model scales, 350M and 1.3B parameters, with the average score used as the overall metric. The results show that DendAttn achieves the best overall performance at both scales. At the 350M scale, DendAttn attains an average score of 31.01, close to the 31.70 of the enhanced Transformer++, while demonstrating more balanced performance across sub-tasks—particularly showing significant advantages on SWDE, FDA, and TriviaQA. As the model scale increases to 1.3B, DendAttn's performance further improves, reaching an average score of 38.94, surpassing all baseline models and maintaining superiority on long-document reasoning and multi-step computation tasks such as TriviaQA and DROP.

Long-sequence Tasks. Long sequences serve as a crucial test for assessing long-term memory capabilities in neural models. As shown in Fig. 4(e), DendAttn consistently and significantly outperforms traditional point-neuron architectures across all tasks, exhibiting stronger stability and generalization in long-context scenarios. In particular, for Single-Document QA and Multi-Document QA, DendAttn achieves scores of 10.07 and 6.81, respectively, surpassing all baseline models by a large margin. This demonstrates its superior ability to integrate cross-paragraph information and maintain coherent reasoning over extended contexts. Moreover, in Summarization and Few-shot Reasoning tasks, DendAttn continues to deliver high-level performance, indicating that its dendritic interaction structure provides distinct advantages in long-range semantic aggregation and compositional generalization.

Dendritic Information Segregation Mechanism Enhance Multi-Task Performance. In this section, we investigate the multi-task performance advantages introduced by the information segregation mechanism of multi-branch dendritic model. We also analyze the effect of state expansion, which are inherently induced by the multi-branch structure. The architectures evaluated include Gated-DeltaNet and variants of DendAttn with different multi-branch configurations, including 330, 812, 815, and 817, where the three digits respectively denote the total number of branches, shared branches, and top-k routing branches. For example, 812 denotes 8 total branches, with 1 shared branches and 2 top-k branches selected among 7 expert branches. Among them, Gated-DeltaNet serves as the baseline model, operating without state expansion or information segregation. Configuration 330 introduces state expansion, while Configuration 812 further incorporates information segregation on top of 330, maintaining the same state-space size. Configurations 815 and 817 then further expand the state space.

Theory. The theoretical rationale for the performance improvement brought by information segregation is illustrated in Fig. 5(b). For the case of absence of information segregation, when trained on mixed multi-task batches, the model converges to a single point in the parameter search space. The resulting parameters are not optimal in each individual task. In the figure, a conceptual two-dimensional loss landscape is used for illustration, where the horizontal axis represents the parameter

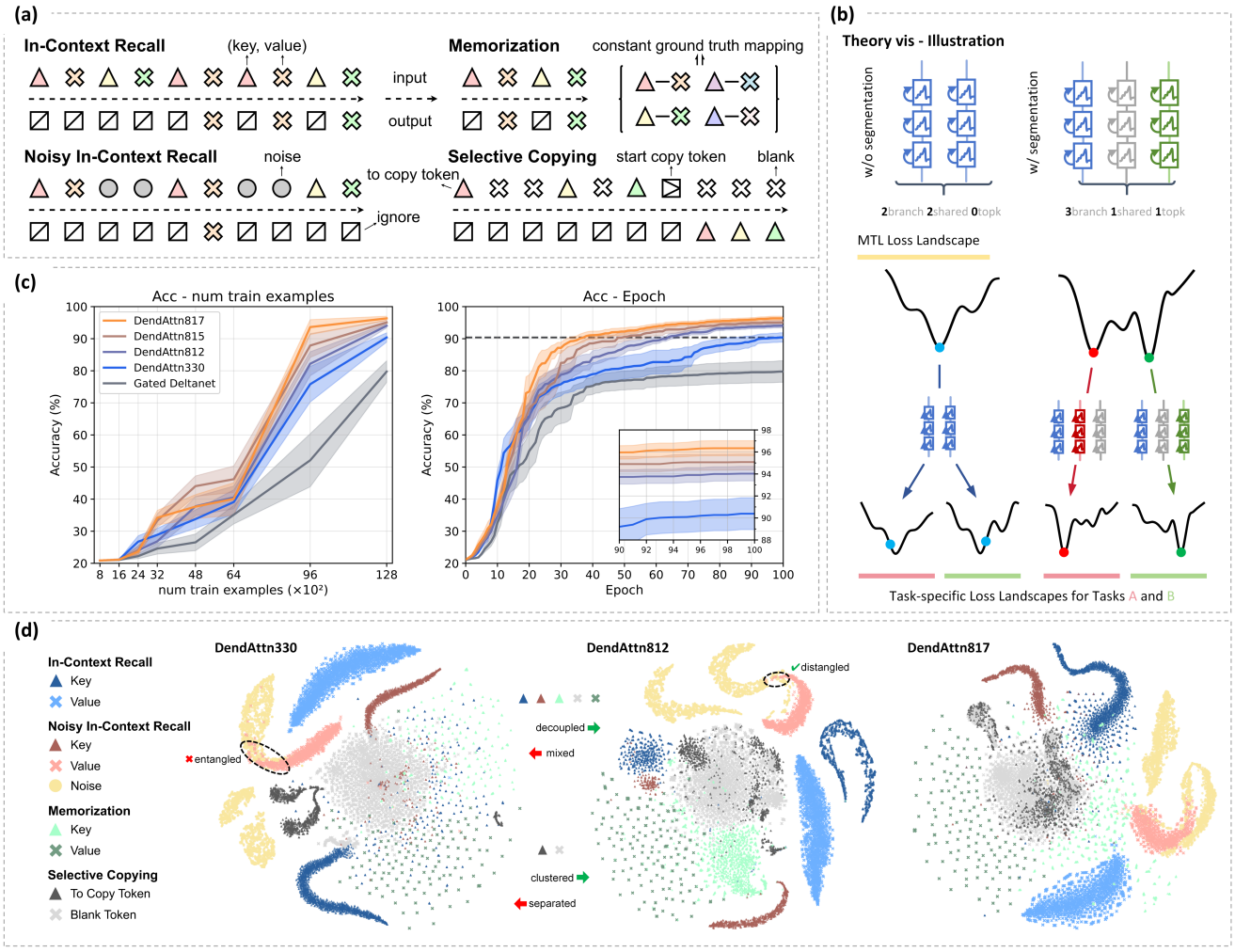

Figure 5. Evaluation of DendAttn's Multi-Task Performance. *(a)* Illustration of the four selected MAD[48] tasks: In-context Recall, Noisy In-context Recall, Selective Copying, and Memorization. In-context Recall: Given an input sequence of key–value pairs, the model must retrieve all values associated with keys that appear in the sequence. Noisy In-context Recall: The same as above, but with noise tokens inserted. Selective Copying: The model must copy specific tokens from one position to a later position while ignoring randomly inserted blank tokens; tokens must be copied in order of appearance. Memorization: The model must learn a fixed key–value mapping that is constant across samples, requiring no in-context reasoning. *(b)* Conceptual illustration of the information segregation mechanism in multi-branch dendritic neurons, which facilitates multi-task learning. The left panel shows a neuron without segregation, where optimization converges to a single point in the joint multi-task loss landscape, failing to reach the optima of individual tasks. The right panel shows a neuron with segregation, where different branches handle different tasks, and the model converges to multiple optima, achieving better per-task performance. *(c)* Comparison of multi-task performance across branch configurations. Gated-DeltaNet serves as a strong linear-model baseline. In DendAttn-xyz, x, y, and z denote the total, shared, and top-k routing branch counts. The improvement from 330 over Gated-DeltaNet and 812 over 330 demonstrates the contribution of information segregation, while comparisons between 812 and other DendAttn multi-branch configurations highlight the benefit of state expansion. *(d)* t-SNE visualization of token features from DendAttn's three representative multi-branch configurations. 812, equipped with information segregation, exhibits the clearest token clustering and separation; 817 follows, and 330 performs worst. Major findings are annotated in the figure, with detailed analysis provided in the main text.

search space and the vertical axis represents the loss value. For the case that the information segregation mechanism is employed, dendritic neurons with multiple branches can autonomously learn task-specific representations during training, routing representations of different tasks to distinct branches. Consequently, the model learns different tasks using different subsets of parameters. If parameter segregation is regarded as a form of partial parameter activation where the remaining parameters are deactivated, the converged model tends to associate each task with a distinct minimum in the multi-task loss landscape. Each task-specific parameter subset is more likely to lie near its corresponding optimum, thereby enhancing overall multi-task performance. A detailed mathematical proof is provided in the Appendix.

Tasks. As shown in Fig. 5(a), we select four MAD tasks, namely In-context Recall, Noisy In-context Recall, Memorization, and Selective Copying, to form mixed multi-task training sets and individual test sets. The In-context Recall task evaluates a model's ability to understand and learn from new information presented within the prompt. Noisy In-context Recall introduces noise tokens to further test the model's ability to ignore irrelevant information. The Selective Copying task examines whether the model can selectively retain relevant tokens while ignoring irrelevant blanks, and Memorization measures the model's ability to memorize static knowledge mappings. In all tasks, the model receives an input sequence and is expected to decode the correct output tokens at the corresponding positions. Detailed task definitions are provided in the figure caption, and complete task configurations are described in the Tasks and Experimental Setup section.

Performance. A comparison of multi-task performance across different multi-branch configurations is shown in Fig. 5(c). The left panel presents the Acc–Epoch curves, illustrating the accuracy improvement trajectories of different models during multi-task training, while the right panel shows the Acc–Number of training examples curves, comparing model performance under varying amounts of training data. The overall trend indicates that Gated-DeltaNet achieves the lowest performance, and its comparison with DendAttn-330 highlights the significant contribution of state expansion. DendAttn-330 performs notably worse than DendAttn-812. In the Accuracy-Epoch plot, 812 achieves the final performance level of 330 by only 60–70 epochs, even though both share the same state-space size, showing strong evidence of the effectiveness of the information segregation mechanism in multi-branch dendritic models. Moreover, 815 and 817 progressively expand the state space, their overall multi-task performance further increases, confirming the benefit of state expansion.

Visualization. In Fig. 5(d), we present t-SNE visualizations of token-level feature vectors extracted from the final SwiGLU layer of representative DendAttn configurations, to examine how different multi-branch DendAttn architectures affect multi-task performance. Comparing 812 with 330, the latter exhibits stronger feature confusion: value and noise tokens from Noisy In-context Recall are poorly separated, and key tokens from In-context Recall and Noisy In-context Recall overlap with those from Memorization and Selective Copying. Moreover, 330 fails to cluster the blank and to-copy tokens of Selective Copying together, indicating incorrect decoding of blanks into corresponding copy targets. In contrast, 812 displays much clearer intra-class cohesion and inter-class separation across all token types. Its blank and to-copy tokens cluster coherently, validating that the information segregation mechanism enables more distinct task representations and thereby improves multi-task learning. For 817, blank and to-copy tokens are also correctly clustered, yet certain confusions similar to those in 330 remain. Thus, 817 benefits mainly from state-space expansion, but such expansion alone cannot match the representational advantages of efficient parameter segregation.

Discussion

Over the past decades, artificial intelligence and neuroscience have advanced along largely independent trajectories[49]. While modern foundation models have achieved remarkable scalability and generalization, they remain constrained by their reliance on simplified point-neuron abstractions that compress neural computation into low-dimensional representations. Conversely, neuroscience has shown that the brain's computational power is significantly influenced by the complex dynamics and compartmentalized morphology of dendritic neurons[50]—yet translating such mechanisms into scalable computational models has remained elusive. This disconnection raises a question: **could such a complex neuronal structure play a crucial role in enhancing the capabilities of current AI foundation models?**

In an effort to answer this question, several researchers have endeavored to integrate neural mechanisms with intricate dynamics into AI foundation models. Spiking neural networks (SNNs)[7,51,52] have been proposed to emulate neuronal firing dynamics and energy-efficient event-driven computation, while models such as Liquid State Machines (LSMs)[53] and the Neural Engineering Framework (NEF)[54] attempt to capture temporal dynamics and state-dependent processing inspired by cortical circuits. Other approaches explicitly incorporate dendritic or compartmental structures, such as the DH-LIF model[55] that employs heterogeneous dendritic time constants, or compartmental recurrent units that mimic localized integration within neurons[56]. In parallel, biophysically grounded simulations based on the Hodgkin–Huxley formalism[57] or multi-compartmental cable equations[58,59] have provided fine-grained insights into the functional role of dendritic nonlinearities. However, these attempts either entail adding external plugins to foundation models or are limited to small-scale simulations, without addressing the core question of how complex neurons can enhance the performance of current foundation models.

To further explore this question, our work bridges this gap by establishing a mechanistic correspondence between dendritic computation and linear foundation models, showing that the state-update rules of single-branch dendrites are mathematically equivalent to those of modern linear recurrent or attention-based architectures. Building upon this equivalence, we introduce DendAttn, a linear foundation architecture that incorporates multi-compartment block-sparse gating, multi-branch state expansion and selective state activation inspired by the advantages observed in dendritic neurons. In the context of foundation models, architectures conceptually similar with DendAttn include MoE[60], MoM[43], and other related studies[61,62]. While these approaches all seem to incorporate state expansion mechanisms, they differ fundamentally when viewed through the lens of neuro-computational perspective.

From a neuro-computational perspective, DendAttn demonstrates a rigorous alignment with principles of dendritic computation, distinguishing it from existing methods through varying aspects of biological plausibility. Specifically, regarding MoE, the contrast reflects a fundamental difference in scale: MoE realizes sparsity at the macro-scale, akin to selecting among distinct neural assemblies (where an FFN represents a cluster of neurons), whereas DendAttn operates at the meso-scale, modeling each attention head as a dendritic unit that implements sparsity by routing upstream signals across internal sub-branches. This biological fidelity is further highlighted when contrasted with MoM. From a microscopic information processing perspective, DendAttn treats the Value component as a consistent upstream input current—which is reused—while employing expanded Query and Key components to encode branch-specific properties. Conversely, MoM's approach of reusing the Query while branching Keys and Values lacks a clear biological plausibility. Furthermore, from a meso-scale morphological perspective, since each attention head functions as a distinct dendritic unit, branching projections and routing should conceptually occur within the head dimension. DendAttn adheres to this principle. In contrast, MoM violates it by enforcing global branching and routing at the token level, which effectively introduces interference between upstream signals and imposes a biologically implausible uniform selection across distinct dendritic pathways. This biological misalignment manifests as engineering inefficiency, causing MoM to incur significant parameter redundancy and reduced routing flexibility, which ultimately limits its empirical performance. Collectively, these comparisons underscore how biological inspiration can guide the discovery of novel and highly optimized architectures, achieving engineering solutions that extend beyond purely heuristic approaches.

This study provides profound insights for understanding the role of complex neuronal structures in enhancing the capabilities of foundational AI models. By reformulating linear models as simplified dendritic systems, we reveal that many key mechanisms function as algorithmic analogues of synaptic and dendritic integration. This reinterpretation fosters a reciprocal interdisciplinary loop: neuroscience can inform the design of scalable, interpretable AI architectures, while artificial models serve as computational substrates for testing theories of neuronal dynamics. Building on this bio-algorithmic alignment, future work integrating dendritic nonlinearity, synaptic plasticity, and adaptive gating may enable continual learning and lifelong adaptation without catastrophic forgetting. Furthermore, exploring the interplay between dendritic dynamics, meta-learning, and memory consolidation could empower foundation models to not only process but also reorganize and refine knowledge over time. Ultimately, by reconnecting the algorithmic foundations of AI with the physiological principles of the brain, neuromorphic approaches like DendAttn offer a promising path toward unifying scalability, adaptability, and biological realism in next-generation intelligent systems.

Methods

Softmax Multi-head attention and linear attention mechanisms

Softmax Multi-head attention. Softmax attention is the central mechanism underpinning the transformer architecture, enabling models to dynamically focus on relevant parts of an input sequence. The input sequence $\mathbf{X} \in \mathbb{R}^{n\times d}$ is projected into three matrices, queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$), using distinct sets of trainable parameters:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V, \tag{6}$$

where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d\times d_k}$, $\mathbf{W}_V \in \mathbb{R}^{d\times d_v}$, and n denotes the sequence length. Then pairwise interactions are computed through the scaled dot product between queries and keys, and the resulting scores are transformed into a probability distribution via the softmax function. This normalized distribution assigns relative weights to all tokens in the sequence. The output of the attention mechanism is obtained by taking the weighted average of the values:

$$\mathbf{O} = \text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{7}$$

where $\mathbf{O} \in \mathbb{R}^{n\times d_v}$ represents the aggregated contextual representation. Although this formulation has become the foundation of large-scale language models, its computational complexity scales quadratically with the sequence length n, which severely

limits its applicability for long-context modeling. Moreover, during inference, the KV cache must be incrementally stored, introducing an $O(n)$ space complexity that further constrains scalability.

Linear attention mechanisms. To overcome the bottlenecks of both computational complexity and memory usage, linear attention avoids the explicit softmax computation by employing kernel-based feature mappings and cumulative state updates, reducing the complexity from $O(n^2)$ to $O(n)$:

$$\text{softmax}(\mathbf{Q}\mathbf{K}^\top) \approx \varphi(\mathbf{Q})\varphi(\mathbf{K}^\top). \tag{8}$$

During inference, this allows replacing left matrix multiplication $\mathbf{Q}\mathbf{K}^\top$ with right multiplication $\mathbf{K}^\top\mathbf{V}$ to achieve $O(n)$ complexity:

$$\mathbf{o}_t = \frac{\varphi(\mathbf{q}_t)\mathbf{S}_t}{\varphi(\mathbf{q}_t)\mathbf{H}_t}, \quad \mathbf{S}_t = \sum_{i=1}^{t} \varphi(\mathbf{k}_i)^\top \mathbf{v}_i, \quad \mathbf{H}_t = \sum_{i=1}^{t} \varphi(\mathbf{k}_i)^\top, \tag{9}$$

While early work explored various kernel functions as softmax alternatives, recent studies have demonstrated that linear models can achieve competitive performance without explicit kernel functions. Crucially, data-dependent gating mechanisms have been shown to be essential for linear attention performance. Additionally,[63] identified that the denominator term can lead to numerical instability issues. The modern formulation of linear models can be expressed as:

$$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \mathbf{k}_t^\top \mathbf{v}_t, \tag{10}$$

where $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ represents the memory state at time step t, and α_t is a learnable decay or gating factor. Unlike the transformer's KV cache, $\mathbf{S}_t$ remains constant in size regardless of sequence length, which confers efficiency but constrains memory capacity and long-range recall.

Block-sparse Gated Multi-State Extended DendAttn

From single-branch multi-compartment sparsity to block-sparse gate

Conventional linear gated models typically adopt either diagonal or fully dense structured transition matrices. Representative architectures include Gated DeltaNet and GLA, which correspond to the two extremes of dendritic abstraction. In the diagonal case, each compartment updates its state solely based on its own past activity, resembling isolated memory traces without lateral communication. Conversely, dense matrices assume full connectivity within a single branch, enforcing global mixing across all compartments. Neither of these formulations aligns with the connectivity patterns of biological dendrites. As shown in Fig. 2(e), dendritic compartments are characterized by local but not global interactions: neighboring compartments exchange signals, whereas long-range connections remain sparse.

Motivated by this structural insight, we propose a block-sparse gated formulation, where compartments within a block are densely connected, while inter-block communication is restricted to sparse, selective links. This design offers a closer correspondence to dendritic organization. By avoiding unnecessary dense coupling, block-sparse transitions enable richer local dynamics and improved stability, while still preserving efficiency for long-sequence modeling. In the following, we detail two core components built upon this principle: (i) establishing a mathematical model of single-branch multi-compartment dendritic neurons, and (ii) extending linear attention by introducing block-sparse gated structures inspired by dendritic neurons.

Modeling multi-compartment dendritic neurons with sparse connectivity. Each Dendritic branch is composed of multiple interconnected compartments, each of which can be approximated as an RC circuit governed by Kirchhoff's law[64] (Fig. 2(b)). By writing current conservation equations for each compartment, the dynamics can be expressed in matrix form, where the transition matrix A reflects the pattern of information exchange across compartments. Depending on how connectivity is defined, the structure of A gives rise to different updating schemes. Fig. 2(e) illustrates three representative cases: diagonal, fully connected, and locally connected.

Diagonal form: independent compartments. We begin with the simplest case, where each compartment is updated solely based on its own state and external input, without any other connections (Fig. 2(e), left). The governing dynamics for three compartments are:

$$C_{Lk}\frac{du_k}{dt} + \frac{u_k}{R_k^L} = r_k i_t, \quad k \in \{1,2,3\} \tag{11}$$

By defining the effective membrane time constant for each compartment as $\tau_k = C_{mk} R_k^L$, the set of differential equations can be compactly reformulated into a compact form. In this representation, the diagonal entries capture the intrinsic decay dynamics of each compartment, reflecting updates that depend only on its own past state. Specifically, the system can be written as:

$$\begin{bmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \\ \frac{du_3}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_1} & 0 & 0 \\ 0 & -\frac{1}{\tau_2} & 0 \\ 0 & 0 & -\frac{1}{\tau_3} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} i_t \tag{12}$$

It is evident that the resulting transition matrix has nonzero entries only along its main diagonal. Consequently, each compartment updates its state independently of the others, with no lateral exchange of information. This isolated form of computation lacks inter-compartmental interaction and thus severely limits representational capacity. In the context of artificial neural architectures, similar formulations are adopted in linear models such as *GLA*, *HGRN2*, and *Mamba2*, which rely on diagonal gating to maintain efficiency but sacrifice cross-state communication.

Fully connected form: dense compartmental interactions. Next, we turn to the configuration shown in the middle of Fig. 2(e), representing a fully connected single-branch multi-compartment dendrite. In this setting, the update of each compartment is no longer independent, but instead influenced by the states of all other compartments within the same branch. Such global coupling leads to a dense interaction pattern, where information is shared across the entire dendritic structure. The corresponding governing equations can be written as:

$$\begin{cases} C_{L1}\frac{du_1}{dt} + \frac{u_1 - u_2}{R_{12}} + \frac{u_1 - u_3}{R_{13}} + \frac{u_1}{R_1^L} = r_1 i_t \\ C_{L2}\frac{du_2}{dt} + \frac{u_2 - u_1}{R_{12}} + \frac{u_2 - u_3}{R_{23}} + \frac{u_2}{R_2^L} = r_2 i_t \\ C_{L3}\frac{du_3}{dt} + \frac{u_3 - u_1}{R_{13}} + \frac{u_3 - u_2}{R_{23}} + \frac{u_3}{R_3^L} = r_3 i_t \end{cases} \tag{13}$$

Similarly, by defining the effective coupling constants as $\tau_{kl} = C_{mk} R_{kl}$, the system of equations can be reorganized into a compact matrix representation:

$$\begin{bmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \\ \frac{du_3}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_1} & \frac{1}{\tau_{12}} & \frac{1}{\tau_{13}} \\ \frac{1}{\tau_{21}} & -\frac{1}{\tau_2} & \frac{1}{\tau_{23}} \\ \frac{1}{\tau_{31}} & \frac{1}{\tau_{32}} & -\frac{1}{\tau_3} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} i_t \tag{14}$$

In contrast to the single-compartment case, the transition matrix of a fully connected single-branch dendrite is completely dense, implying that each compartment is coupled to every other within the branch and enabling global information exchange. This configuration parallels linear models such as DeltaNet and Gated DeltaNet, where fully connected gating matrices substantially enhance representational capacity. Compared with purely diagonal updates, such dense coupling captures richer temporal dependencies, underscoring the importance of inter-compartmental interactions for improved performance. Nevertheless, this design is biologically implausible, as real dendrites typically exhibit locally dense yet globally sparse connectivity rather than universal all-to-all coupling.

Local connectivity: dendrite-inspired block structure. Between the extremes of diagonal independence and full connectivity lies a more biologically realistic scenario, where interactions are restricted to local neighborhoods (Fig. 2(e), right). In this case, each compartment communicates only with its adjacent neighbors, while distant compartments remain disconnected. The governing dynamics for this setting can be expressed as:

$$\begin{cases} C_{L1}\frac{du_1}{dt} + \frac{u_1 - u_2}{R_{12}} + \frac{u_1}{R_1^L} = r_1 i_t \\ C_{L2}\frac{du_2}{dt} + \frac{u_2 - u_1}{R_{12}} + \frac{u_2 - u_3}{R_{23}} + \frac{u_2}{R_2^L} = r_2 i_t \\ C_{L3}\frac{du_3}{dt} + \frac{u_3 - u_2}{R_{23}} + \frac{u_3}{R_3^L} = r_3 i_t \end{cases} \tag{15}$$

By consolidating the above equations, the system can be reformulated in a compact matrix representation. Specifically, the system can be expressed as follows:

$$\begin{bmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \\ \frac{du_3}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_1} & \frac{1}{\tau_{12}} & 0 \\ \frac{1}{\tau_{21}} & -\frac{1}{\tau_2} & \frac{1}{\tau_{23}} \\ 0 & \frac{1}{\tau_{32}} & -\frac{1}{\tau_3} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} i_t \tag{16}$$

It is evident that the transition matrix derived from this biologically grounded configuration exhibits a block-structured pattern: interactions are dense within local neighborhoods of compartments, yet sparse across distant regions. Notably, such a block-sparse structure stands in sharp contrast to the transition matrices employed in most mainstream linear models, which typically adopt either purely diagonal or fully dense forms.

Block-sparse Gated Linear Attention. As illustrated in the right panel of Fig. 2(f), dendritic compartments can be organized into groups, with full connectivity within each group and have shared compartments across neighboring groups to propagate information from local to global scales. Translating this principle into linear models yields the block-sparse gating scheme, where gating is decomposed into multiple interacting blocks. Below, we describe in detail the sequential computation process of the model adopting the delta rule updating scheme:

For an input $\mathbf{x}_t \in \mathbb{R}^d$ at time step t, we first compute the standard linear projections $\mathbf{k}_t = \mathbf{x}_t \mathbf{W}_k, \quad \mathbf{q}_t = \mathbf{x}_t \mathbf{W}_q, \quad \mathbf{v}_t = \mathbf{x}_t \mathbf{W}_v$ where $\mathbf{W}_k, \mathbf{W}_q, \mathbf{W}_v$ denote learnable parameter matrices. To perform block-sparse gating, the key vector $\mathbf{k}_t \in \mathbb{R}^{d_k}$ is partitioned along the hidden dimension into $M \geq 1$ contiguous blocks. Each block has a window size d_k^b, subject to $d_k \geq d_k^b \geq \frac{d_k}{M}$, and the partitioning proceeds with a stride P. The gating coefficient of the m-th block at time t, denoted $\boldsymbol{\alpha}_t^m$, is then computed as:

$$\mathbf{k}_t^m = \mathbf{k}_t \left[m * P : m * P + d_k^b \right] \tag{17}$$

$$\alpha_t^m = \mathbf{I} - \mathbf{k}_t^{mT} \mathbf{k}_t^m \tag{18}$$

For the state matrix at time step t, $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$, we apply the same partitioning strategy along the d_k dimension. Each block is then updated independently, and the computation for the m-th block is given by:

$$\mathbf{S}_t^m = \alpha_t^m \mathbf{S}_{t-1}^m + \mathbf{k}_t^{mT} \mathbf{v}_t \quad \in R^{d_k^b \times d} \tag{19}$$

$$\mathbf{o}_t^m = \mathbf{q}_t^m \mathbf{S}_t^m \quad \in R^d \tag{20}$$

The final output is obtained by aggregating the contributions of all blocks, expressed as $o_t = \sum_m \mathbf{o}_t^m$.

From multi-branch inspiration to sparsely-activated state expansion

Compared with softmax attention, which requires an ever-growing KV cache, linear models maintain hidden states of constant size, thereby achieving superior computational and memory efficiency. However, this extreme restriction on memory capacity also makes linear models prone to forgetting long-range dependencies, which in turn leads to significantly poorer performance than softmax attention on recall tasks and long-sequence modeling. As shown in the previous section, such linear models can be interpreted as the functional analogue of single-branch dendrites. Whereas, biologically realistic dendrites are organized in multi-branch structures rather than single branches.

Inspired by this morphological property, we extend the hidden state space of linear models into multiple mutually independent sub-states, mimicking the compartmentalization of multi-branch dendrites. Furthermore, drawing on the principle of sparse branch activation observed in biological dendrites, we enforce a sparsely activated scheme for the expanded states. This design not only enlarges the effective memory capacity but also mitigates interference and aliasing across states, thereby enhancing the model's advantage in long-term memory and retrieval. In what follows, we first introduce the modeling of multi-branch dendritic structures, and then present the corresponding state expansion method for linear models inspired by this biological principle.

Modeling multi-branch multi-compartment dendritic neurons. As illustrated in Fig. 2(c), each dendritic neuron in the dendritic layer receives the input from the previous layer, X_t, which is weighted by synaptic connections W_{in} to form the input current $I_t = X_t W_{in}$. This input is then distributed across compartments through weights Γ, such that each branch evolve independently without lateral interactions prior to the soma. Finally, the states of each branch are aggregated through weighted summation with coefficients $\mathscr{M}$, producing the dendritic neuron's output. Taking a single dendritic neuron as an example, the

detailed computation can be expressed as follows:

$$\begin{bmatrix} \dot{u}_{1,1} \\ \dot{u}_{1,2} \\ \dot{u}_{1,3} \\ \dot{u}_{2,1} \\ \dot{u}_{2,2} \\ \dot{u}_{2,3} \\ \dot{u}_{3,1} \\ \dot{u}_{3,2} \\ \dot{u}_{3,3} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \alpha_{11}^1 & \alpha_{12}^1 & 0 \\ \alpha_{21}^1 & \alpha_{22}^1 & \alpha_{23}^1 \\ 0 & \alpha_{32}^1 & \alpha_{33}^1 \end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} \alpha_{11}^2 & \alpha_{12}^2 & 0 \\ \alpha_{21}^2 & \alpha_{22}^2 & \alpha_{23}^2 \\ 0 & \alpha_{32}^2 & \alpha_{33}^2 \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \begin{bmatrix} \alpha_{11}^3 & \alpha_{12}^3 & 0 \\ \alpha_{21}^3 & \alpha_{22}^3 & \alpha_{23}^3 \\ 0 & \alpha_{32}^3 & \alpha_{33}^3 \end{bmatrix} \end{bmatrix} \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ u_{1,3} \\ u_{2,1} \\ u_{2,2} \\ u_{2,3} \\ u_{3,1} \\ u_{3,2} \\ u_{3,3} \end{bmatrix} + \begin{bmatrix} \gamma_{1,1} \\ \gamma_{1,2} \\ \gamma_{1,3} \\ \gamma_{2,1} \\ \gamma_{2,2} \\ \gamma_{2,3} \\ \gamma_{3,1} \\ \gamma_{3,2} \\ \gamma_{3,3} \end{bmatrix} \times i_t \tag{21}$$

From the above formulation, it follows that the state update of each branch can be performed independently. Consequently, the equation can be decomposed into separate recursive updates for each branch, which can then be expressed in a compact matrix form:

$$\begin{cases} \dot{\mathbf{U}}_1 = \mathbf{A}_1 \mathbf{U}_1 + \Gamma_1 I_t \\ \dot{\mathbf{U}}_2 = \mathbf{A}_2 \mathbf{U}_2 + \Gamma_2 I_t \\ \dot{\mathbf{U}}_3 = \mathbf{A}_3 \mathbf{U}_3 + \Gamma_3 I_t \end{cases} \tag{22}$$

At the same time, dendritic branches exhibit the property of sparse activation, such that the final output only integrates the states of the activated branches. For instance, if branch one and branch three are activated, the output is given by:

$$O = \mathscr{M}_1 \dot{\mathbf{U}}_1 + \mathscr{M}_3 \dot{\mathbf{U}}_3 \tag{23}$$

Multi-State Sparse Extended Linear Attention. In this section, we introduce a sparse expansion approach for hidden states, inspired by the multi-branch dendrites. Linear models face limitations in hidden states capacity and aliasing memories, for which sparse expansion serves as an effective solution. As shown in Figure S3, for an input X and E dendritic branch experts, we first compute the components of standard linear attention: $\mathbf{Q}/\mathbf{K}/\mathbf{V} = \mathbf{X}\mathbf{W}_{Q/K/V}$, $\mathbf{G}/\beta = \mathbf{X}\mathbf{W}_{G/\beta}$, where $\mathbf{W}_V \in \mathbb{R}^{D \times D_v}$, $\mathbf{W}_{Q/K} \in \mathbb{R}^{D \times D_k}$ and $\mathbf{W}_{G/\beta} \in \mathbb{R}^{D \times E*H}$. Here, D denotes the model dimension, and D_k, D_v are the dimensions of the Key and Value, respectively. $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are then split into H heads:

$$\mathbf{Q}_i = (\mathbf{Q}_{i,1}, \ldots, \mathbf{Q}_{i,h}, \ldots, \mathbf{Q}_{i,H}) \in \mathbb{R}^{H \times d_k} \tag{24}$$

where $d_k = \frac{D_k}{H}$, $\mathbf{Q}_i$ denotes the query vector corresponding to the i-th token, $i \in 1, 2, \ldots, L$, and L is the sequence length. Then, each head of memory state will be expanded into E branch experts. These E branch experts include E_s shared branch experts and E_r routing branch experts.

The routing experts simultaneously expand the effective memory capacity and mitigate memory aliasing through sparse expert selection. For each token $X_{i,h}$ within head h, its corresponding query vector $\mathbf{Q}_{i,h}$ is used to determine expert participation via an auxiliary router parameterized by $\mathbf{W}_h^{\text{rout}} \in \mathbb{R}^{d_k \times E_r}$. This routing mechanism is formulated as follows:

$$\mathbf{r}_{i,h} = \sigma(\mathbf{W}_h^{rout} \mathbf{Q}_{i,h}) \tag{25}$$

$$\mathbf{r}_{i,h}^{topk}, Ind_{i,h} = TopK(r_{i,h}, topk) \tag{26}$$

where σ denotes the softmax function. $\mathbf{r}_{i,h} \in \mathbb{R}^{E_r}$ represents the probability distribution over the E_r experts for the i-th token in the h-th head, and $\text{Ind}_{i,h}$ is the set of indices of the selected experts.

For each selected expert $e_r \in 1, 2, \ldots, E_r$, the vectors $\mathbf{Q}_{i,h}$ and $\mathbf{K}_{i,h}$ are projected using dedicated routing expert projection matrices $W_{h,e_r}^{\text{Exp}_{Q/K}} \in \mathbf{R}^{d_k \times d_k}$ to obtain the expert-specific representations:

$$\mathbf{Q}/\mathbf{K}_{i,h,e_r} = \mathbf{W}_{h,e_r}^{\text{Exp}_{Q/K}} \mathbf{Q}/\mathbf{K}_{i,h}, \quad \mathbf{G}/\beta_{i,h,e_r} = \mathbf{G}/\beta_{i,(h-1)*E+E_s+e_r} \tag{27}$$

Each expert then computes a linear attention output:

$$\mathbf{O}_{i,h,e_r} = LinearAttn(\mathbf{Q}_{i,h,e_r}, \mathbf{K}_{i,h,e_r}, \mathbf{V}_{i,h}, \mathbf{G}_{i,h,e_r}, \beta_{i,h,e_r}) \tag{28}$$

Then DendAttn aggregates the outputs across all selected experts within each head by weighting them with the routing scores:

$$\mathbf{O}_{i,h,e_r} = \mathbf{r}_{i,h,e_r}\mathbf{O}_{i,h,e_r} \tag{29}$$

$$\mathbf{O}_{i,h} = \sum_{e_r} \mathbf{O}_{i,h,e_r}, \quad \text{if } e_r \in Ind_{i,h} \tag{30}$$

where $\mathbf{r}_{i,h,e_r}$ is the corresponding routing weight from $\mathbf{r}_{i,h}^{\text{topk}}$. Finally, $\mathbf{O}_{i,h}$ is mapped back to the original sequence positions to obtain the final output from the routing experts $\mathbf{O}^{\text{rout}} \in R^{L\times H\times D_v}$.

The shared experts are always-active modules that encode dense, global information. Similar to the operations in routing experts, shared experts perform per-token per-head transformations, but they do not involve a routing projection matrix or expert selection process. Each token's head-specific representations $\mathbf{Q}_{i,h}$, $\mathbf{K}_{i,h}$, and $\mathbf{G}_{i,h}$ are projected through shared expert-specific matrices $\mathbf{W}_{h,e_s}^{\text{Exp}_{Q/K/G}} \in \mathbb{R}^{d_k\times d_k}$ to yield expert-transformed representations $\mathbf{Q}_{i,h,e_s}, \mathbf{K}_{i,h,e_s}, \mathbf{G}_{i,h,e_s}$ for each shared expert $e_s \in 1,2,\ldots,E_s$. Linear attention is then applied within each shared expert:

$$\mathbf{O}_{i,h,e_s} = LinearAttn(\mathbf{Q}_{i,h,e_s}, \mathbf{K}_{i,h,e_s}, \mathbf{V}_{i,h}, \mathbf{G}_{i,h,e_s}, \beta_{i,h,e_s}) \tag{31}$$

Unlike routing experts, shared experts do not involve token-level routing weights. Instead, we assign a uniform weight $\lambda_s = \frac{1}{E_s}$ to each shared expert's output. The aggregated output from shared experts is then computed as:

$$\mathbf{O}_{:,h,e_s} = \lambda_s \mathbf{O}_{:,h,e_s} \tag{32}$$

$$\mathbf{O}_{:,h}^{\text{share}} = \sum_{e_s} \mathbf{O}_{:,h,e_s} \tag{33}$$

Finally, the output of DendAttn for head h is obtained by summing the contributions from both routing and shared experts:

$$\mathbf{O}_{:,h}^{linear} = \mathbf{O}_{:,h}^{\text{share}} + \mathbf{O}_{:,h}^{\text{rout}} \tag{34}$$

DendAttn performs expert selection at the token and head level. Therefore, the sequence lengths for each expert may vary. This can be efficiently adapted to existing linear attention operators using *varlen* techniques.

Tasks and experimental setup

Overview of Fig3 Fig. 3 primarily demonstrates the fundamental language modeling capability of our proposed multi-branch, multi-compartment dendrites inspired linear attention DendAttn. In Fig. 3, we conduct a comparative study with a diverse set of neuron-based linear models and the standard softmax attention mechanism. These baselines include point-neuron representatives such as HGRN and RetNet; independently update multi-compartment dendritic neuron models such as GLA and HGRN2, where state updates are influenced only by their own compartments; fully connected compartmental models such as DeltaNet and Gated DeltaNet; as well as simplified dynamic variants of multi-branch structures like MOM. In the following sections, we detail the task formulations, experimental setups, and datasets across four perspectives: (i) language modeling performance, (ii) efficiency comparisons, and (iii) visualization-based analyses.

Language Modeling (Fig. 3(b) and (d)). We pretrain two model sizes (350M and 1.3B parameters) on the open-source Slimpajama corpus[37] and then measure language-modeling capability on downstream evaluations, including a Commonsense Reasoning suite and SCROLLS[38]. The Slimpajama dataset is a large-scale open-source pretraining corpus introduced by Together in 2023 as a high-quality alternative to The Pile. It contains approximately 627 billion tokens collected from diverse sources, including Common Crawl web pages, Wikipedia, ArXiv papers, GitHub repositories, StackExchange discussions, and books. During pretraining, models follow the standard autoregressive next-token objective—predicting the subsequent token from its left context—optimized via cross-entropy between the model distribution and the empirical data. For the 350M model, we trained on 15B tokens with a batch size of 0.5M tokens per step, while for the 1.3B model, we trained on 100B tokens with a batch size of 2M tokens per step. After pretraining, we report performance on two families of tasks, each highlighting a distinct facet of language modeling: i) *Commonsense Reasoning*: a collection assessing broad linguistic understanding and commonsense inference, spanning multiple-choice QA, semantic coherence, and physical reasoning; results are averaged in a zero-shot setting over standard benchmarks; ii) *SCROLLS*: a long-context evaluation covering varied domains—NLP research articles, books, movie scripts, narratives, and legal text—where we present average zero-shot scores. Concretely, we evaluate on GovReport, NarrativeQA, and QASPER, and additionally study length extrapolation with sequence contexts up to 128k tokens, probing both baseline modeling quality and long-range understanding.

Efficiency comparison between softmax attention and DendAttn architecture (Fig. 3(c)). Both softmax attention and linear models are optimized through custom acceleration algorithms designed to enhance their training efficiency on GPU

clusters. For transformers, the FlashAttention[65–67] is an optimized attention algorithm designed to accelerate Transformer training and inference by reducing the memory and computation overhead of the standard softmax attention mechanism. Instead of materializing the full attention matrix, FlashAttention leverages tiling and on-chip SRAM caching to compute attention in a streaming fashion, which avoids expensive reads and writes to high-bandwidth memory. For linear models, Flash Linear Attention provides a fused GPU kernel for linear-time attention mechanisms, eliminating redundant memory operations and leveraging tiling strategies similar to FlashAttention. While linear attention inherently scales as $O(n)$ by replacing the softmax kernel, naive implementations are often memory-bound and inefficient. Flash Linear Attention bridges this gap by streamlining the computation on modern accelerators, yielding substantial speedups and reduced memory footprint for long-sequence training and inference. The implementation of DendAttn is built upon Flash Linear Attention. To mitigate redundant computation, we adopt the varlen technique to efficiently accelerate sparse multi-branch activations. For inference efficiency experiments, We benchmark inference by comparing the softmax attention implement with FlashAttention kernel against DendAttn implement with flash linear attention and varlen in terms of end-to-end latency, peak memory footprint, and FLOPs under matched conditions. To normalize workload, we fix the total tokens processed per step and vary the sequence length. At each setting, we log end-to-end latency, peak GPU memory, and estimated FLOPs under the same hardware, software, and precision configuration.

Visual experimental setup (Fig. 3(e)). To better understand how different architectures allocate representational capacity, we visualize attention maps from the pretrained models. Specifically, we extract attention weights from selected layers and heads of the 1.3B Gated DeltaNet and DendAttn, and for DendAttn we further visualize the contribution of each individual branch within every head. For comparability, we evaluate models on the same held-out sequences with length 100. All visualizations are generated under identical inference conditions. We reconstruct approximate attention maps for both Gated DeltaNet and DendAttn, ensuring that visualizations remain aligned with their underlying mechanisms. In addition to standard attention heatmaps, we also visualize branch-level selection dynamics within the 1.3B-parameter DendAttn model. The architecture comprises 8 heads, each containing 8 parallel branches, of which only the sparsely activated subset is visualized.

Overview of Fig4 Fig. 4 aims to demonstrate that expanding to a multi-branch architecture enhances memory capacity and yields clear advantages on recall, and long-context tasks. To this end, we adopt baselines including point-neuron, single-branch neuron, and simple dynamic multi-branch dendrites inspired linear models. We conduct comparisons on two synthetic recall benchmarks as well as one real-world recall benchmark. In addition, we evaluate DendAttn against alternative models on LongBench to assess long-range modeling performance. In the following paragraphs, we describe task details, experimental setups, and datasets in three parts: synthetic recall benchmarks, real-world retrieval benchmarks, and LongBench.

Synthetic Recall Benchmarks (Fig. 4(a),(c) and (d),(f)). We evaluate associative memory in linear models using Multi-Query Associative Recall (MQAR), a synthetic setting that mirrors how real text often demands multiple in-context lookups at different locations and over a large vocabulary. Each sequence of length N begins with T key–value (KV) pairs occupying the first $2T$ tokens; subsequent positions $[2T, N-1]$ contain keys sampled from the initial set, and the model must output the corresponding values at the queried spots. This formulation departs from classic single-query AR by stressing concurrent retrieval and variable query positions, thereby exposing scalability and parameter-efficiency limits across architectures. We instantiate MQAR with 100k training and 3k test sequences and consider three sequence lengths—256 (16 KV pairs), 512 (64 KV pairs) and 1024 (128 KV pairs). To probe memory capacity versus model width, all models use two layers while varying hidden size $d \in 32, 64, 128, 256$. We sweep learning rates on a log scale from 10^{-4} to 10^{-2} and train for 64 epochs to ensure convergence. Batch sizes are 256 and 128 for sequence lengths 256 and 512, respectively. This protocol provides a controlled stress test for storing, retrieving, and associating dispersed tokens, and it highlights the gap between synthetic competence and behavior on real language data reported in prior analyses. In addition, we also conduct evaluations on the Needle In A Haystack (NIAH) benchmark. NIAH is a synthetic task designed to probe whether models can recover a single relevant item from a long distractor sequence. Each input sequence of length N is composed of mostly filler tokens interspersed with one or more "needle" tokens that encode the information to be retrieved. We instantiate NIAH across sequence lengths ranging from 1k to 8k tokens, with both single-needle, multi-needle and multi-qury/key/value variants.

Real-World Retrieval Benchmark (Fig. 4(b)). To evaluate retrieval and reasoning in realistic scenarios, we construct the Real-World Retrieval Benchmark by integrating six complementary datasets. TriviaQA contains large-scale open-domain question–answer pairs grounded in evidence from Wikipedia and the web, while DROP emphasizes discrete reasoning such as arithmetic, counting, and symbolic comparison over natural language passages. Natural Questions (NQ) consists of real user queries paired with full Wikipedia articles, testing both retrieval and comprehension at scale. SWDE provides semi-structured webpages across domains such as movies and products, requiring robust information extraction from noisy layouts, and FDA focuses on functional dependency analysis in structured tabular data, probing relational and symbolic reasoning. Finally, SQuAD serves as a canonical reading comprehension task, where answers must be extracted directly from Wikipedia passages. We evaluate 350M and 1.3B models on this benchmark in a zero-shot setting, using only the official test sets and applying a unified evaluation protocol to ensure comparability across tasks.

LongBench.(Fig. 4(e)). We evaluate long-context understanding on LongBench, a bilingual benchmark covering both English and Chinese tasks. The benchmark spans multiple categories, including single-document and multi-document question answering, summarization, few-shot learning, code completion, and synthetic retrieval, with each category further divided into finer-grained sub-benchmarks that target specific skills such as reasoning, factual recall, or cross-document integration. Since our models are primarily trained on English corpora, we report results as the average across all English sub-benchmarks under each category. All evaluations are conducted in a zero-shot setting using the official test splits.

Overview of Fig5 Fig. 5 presents the advantages of the state expansion and information segregation mechanisms embedded in our multi-branch dendritic DendAttn architecture for multi-task performance. To do so, we constructed a multi-task scenario based on the MAD framework and evaluated the Gated-DeltaNet and DendAttn architectures.

MTL Dataset Construction (Fig. 5(a)). We selected four MAD-defined tasks, unified the sequence length to 256, and generated equal numbers of samples per task before mixing them to form the multi-task dataset. *In-context Recall.* Given an input sequence of key–value pairs, the model must retrieve all values associated with keys that appear in the sequence. *Noisy In-context Recall.* The same as above, but with noise tokens inserted. *Selective Copying.* The model must copy specific tokens from one position to a later position while ignoring randomly inserted blank tokens; tokens must be copied in order of appearance. *Memorization.* The model must learn a fixed key–value mapping (analogous to factual knowledge) that is constant across samples, requiring no in-context reasoning. During training, each batch is mixed, containing randomly sampled instances from all tasks. To prevent token-semantic conflicts between subtasks, we explicitly constrained the input-ID spaces of different tasks to be disjoint during data construction.

Model Architecture and Training Hyperparameters. Following MAD's setup, we employ a Decoder-only Causal LM with four main blocks: Attn–SwiGLU–Attn–SwiGLU. The two attention layers are instantiated as either Gated-DeltaNet or DendAttn. The hidden size is 128, the head dimension 64 (i.e., two heads total), and other settings follow the standard configuration of FLA-series linear models (see Supplementary Materials). Training uses the Adam optimizer with initial learning rate = $5e^{-4}$, batch size = 128, weight decay = 0.1, cosine schedule, no warm-up, and 100 epochs.

Figures and Plots. Fig. 5(a) visualizes the four MAD tasks. Fig. 5(b) schematically explains the theoretical advantage of multi-branch dendritic information segregation (formal proof in the Supplementary). Fig. 5(c) reports multi-task results. The Acc-Epoch curve shows accuracy evolution during training (with num train examples = 12 800 per subtask). The Acc-Num Train Examples curve shows accuracy versus training-set size. Both results average 10 random seeds with 67% confidence intervals as error bands, controlling randomness from sample generation through evaluation. Fig. 5(d) displays t-SNE projections of token features from three representative DendAttn configurations. Visualization is performed on models trained with num train examples = 12 800 and identical random seed 67890. For each subtask, 20 test samples are fed forward, and token features are extracted from the final SwiGLU layer outputs.

Data availability

All data used in this paper are publicly available and can be accessed at https://huggingface.co/datasets/cerebras/SlimPajama-627B for Slimpajama dataset, https://github.com/HazyResearch/zoology for MQAR dataset, https://github.com/EleutherAI/lm-evaluation-harness for Commonsense Reasoning, SCROLLS, Real-World Retrieval and RULER datasets, https://github.com/THUDM/LongBench for LongBench dataset. https://github.com/athms/mad-lab for MAD dataset.

Code availability

Accession codes will be available at https://github.com/WKX933 before publication.

References

1. Devlin, J. *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL* (2019).
2. Brown, T. *et al.* Language models are few-shot learners. *NeurIPS* (2020).
3. OpenAI. Gpt-4 technical report. *ArXiv* **abs/2303.08774** (2023).
4. Anil, R. *et al.* Palm 2 technical report. *arXiv preprint arXiv:2305.10403* (2023).
5. Liu, A. *et al.* Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

6. Turing, A. M. Computing machinery and intelligence. In *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer*, 23–65 (Springer, 2007).
7. Roy, K., Jaiswal, A. & Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature* **575**, 607–617 (2019).
8. Marblestone, A. H., Wayne, G. & Kording, K. P. Toward an integration of deep learning and neuroscience. *Front. Comput. Neurosci.* (2016).
9. Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y. & et al. A deep learning framework for neuroscience. *Nat. Neurosci.* (2019).
10. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin mathematical biophysics* **5**, 115–133 (1943).
11. Elman, J. L. Finding structure in time. *Cogn. Sci.* (1990).
12. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* (1997).
13. Poirazi, P. & Mel, B. W. Impact of dendritic nonlinearities on the computational properties of pyramidal neurons. *Neuron* **29**, 779–796 (2001).
14. London, M. & Häusser, M. Dendritic computation. *Annu. Rev. Neurosci.* **28**, 503–532 (2005).
15. Arora, S. *et al.* Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations* (2024).
16. Qin, Z. *et al.* HGRN2: Gated linear RNNs with state expansion. In *First Conference on Language Modeling* (2024).
17. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* (1994).
18. Baddeley, A. Working memory: Theories, models, and controversies. *Annu. Rev. Psychol.* (2012).
19. Spruston, N. Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.* **9**, 206–221 (2008).
20. Stuart, G. J. & Spruston, N. Dendritic processing: new insights into neuronal computation. *Nat. Rev. Neurosci.* (2016).
21. Major, G., Larkum, M. E. & Schiller, J. Active properties of neocortical pyramidal neuron dendrites. *Annu. Rev. Neurosci.* **36**, 1–24, DOI: 10.1146/annurev-neuro-062111-150343 (2013).
22. Branco, T., Clark, B. A. & Häusser, M. Dendritic discrimination of temporal input sequences in cortical neurons. *Science* **329**, 1671–1675, DOI: 10.1126/science.1189664 (2010).
23. Poirazi, P., Brannon, T. & Mel, B. W. Pyramidal neuron as two-layer neural network. *Neuron* **37**, 989–999 (2003).
24. Mel, B. W. Information processing in dendritic trees. *Neural Comput.* **4**, 502–517 (1992).
25. Gu, A., Goel, K. & Ré, C. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations* (2022).
26. Gu, A. & Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling* (2024).
27. Dao, T. & Gu, A. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning* (2024).
28. Chen, X. *et al.* Pmsn: A parallel multi-compartment spiking neuron for multi-scale temporal processing. *arXiv preprint arXiv:2408.14917* (2024).
29. Yang, S. *et al.* Gated linear attention transformers with hardware-efficient training. *arXiv preprint* (2024).
30. Yang, S., Wang, B., Zhang, Y., Shen, Y. & Kim, Y. Parallelizing linear transformers with the delta rule over sequence length. *Adv. neural information processing systems* **37**, 115491–115522 (2024).
31. Hebb, D. O. *The organization of behavior: A neuropsychological theory* (John Wiley & Sons, 1949).
32. He, L. *et al.* Network model with internal complexity bridges artificial intelligence and neuroscience. *Nat. Comput. Sci.* **4**, 584–599 (2024).
33. Wang, K. *et al.* Mmdend: Dendrite-inspired multi-branch multi-compartment parallel spiking neuron for sequence modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 27459–27470 (2025).

34. Vaswani, A. *et al.* Attention is all you need. *Adv. Neural Inf. Process. Syst.* (2017).

35. Katharopoulos, A., Vyas, A., Pappas, N. & Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, 5156–5165 (PMLR, 2020).

36. De, S. *et al.* Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427* (2024).

37. Computer, T. Slimpajama (2023).

38. Shaham, U. *et al.* Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533* (2022).

39. Sun, Y. *et al.* Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621* (2023).

40. Qin, Z., Yang, S. & Zhong, Y. Hierarchically gated recurrent neural network for sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS) 36*, vol. 36, 23122–23141 (2023). Spotlight poster.

41. Zhang, Y. *et al.* Gated slot attention for efficient linear-time sequence modeling. *Adv. Neural Inf. Process. Syst.* **37**, 116870–116898 (2024).

42. Yang, S., Kautz, J. & Hatamizadeh, A. Gated delta networks: Improving mamba2 with delta rule. *CoRR* (2024).

43. Du, J., Sun, W., Lan, D., Hu, J. & Cheng, Y. Mom: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685* (2025).

44. Arora, S. *et al.* Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927* (2023).

45. Yu, Y. *et al.* Sequential-niah: A needle-in-a-haystack benchmark for extracting sequential needles from long contexts. *arXiv preprint arXiv:2504.04713* (2025).

46. Bai, Y. *et al.* Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3119–3137, DOI: 10.18653/v1/2024.acl-long.172 (Association for Computational Linguistics, Bangkok, Thailand, 2024).

47. Bai, Y. *et al.* Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204* (2024).

48. Poli, M. *et al.* Mechanistic design and scaling of hybrid architectures. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, 40908–40950 (2024).

49. Hassabis, D., Kumaran, D., Summerfield, C. & Botvinick, M. Neuroscience-inspired artificial intelligence. *Neuron* **95**, 245–258 (2017).

50. Fischer, L. *et al.* Dendritic mechanisms for in vivo neural computations and behavior. *J. Neurosci.* **42**, 8460–8467 (2022).

51. Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural Networks* **10**, 1659–1671 (1997).

52. Zenke, F. & Vogels, T. P. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Comput.* **33**, 899–925 (2021).

53. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).

54. Eliasmith, C. & Anderson, C. H. *Neural engineering: Computation, representation, and dynamics in neurobiological systems* (MIT Press, 2003).

55. Feng, J., Zhang, Z., Zhao, R., Pan, G. & Xu, B. Dh-lif: A dendritic hierarchical leaky integrate-and-fire model for spatio-temporal information processing. *Neural Comput.* **31**, 2502–2521 (2019).

56. Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A. & Naud, R. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nat. Neurosci.* **24**, 1010–1019 (2021).

57. Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The J. Physiol.* **117**, 500–544 (1952).

58. Rall, W. Theoretical significance of dendritic trees for neuronal input-output relations. In Reiss, R. F. (ed.) *Neural Theory and Modeling*, 73–97 (Stanford University Press, 1964).

59. Segev, I. & Rall, W. Compartmental models of complex neurons. *Methods Neuronal Model. From Ions to Networks* 93–136 (1998).

60. Shazeer, N. *et al.* Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

61. Guo, H. *et al.* Log-linear attention. *arXiv preprint arXiv:2506.04761* (2025).

62. Liu, K., Gao, J. & Chen, K. Scaling up the state size of rnn llms for long-context scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11516–11529 (2025).

63. Qin, Z. *et al.* The devil in linear transformer. *ArXiv* **abs/2210.10340** (2022).

64. Kirchhoff, G. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik* **148**, 497–508 (1847).

65. Dao, T., Fu, D., Ermon, S., Rudra, A. & Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Adv. neural information processing systems* **35**, 16344–16359 (2022).

66. Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691* (2023).

67. Shah, J. *et al.* Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Adv. Neural Inf. Process. Syst.* **37**, 68658–68685 (2024).

Acknowledgements

This work was partially supported by CAS Project for Young Scientists in Basic Research (YSBR-116), National Distinguished Young Scholars (62325603), National Natural Science Foundation of China (62236009, U22A20103), Beijing Science and Technology Plan (Z241100004224011).

Author contributions statement

K.W., B.X., and G.L. conceived the work. K.W. proposed the key concepts and oversaw the design and organization of all experiments. K.W. designed and implemented the DendAttn design. K.W. conducted the dendritic computation modeling. K.W. implemented experiments to validate that dendritic morphology augments temporal dynamics and dendritic multi-branch augments memory capacity. R.S. implemented experiments to validate that dendritic state expansion and information segregation mechanisms enhance multi-Task performance. K.W., R.S., and B.X. contributed to the discussion of the experiment result analysis, and G.L. led the discussion. K.W. drafted the manuscript, with additional contributions from G.L. The whole project is supervised by G.L. All authors reviewed the manuscript.

Additional information

Competing interests. The authors declare no competing interests.

Biologically Inspired Neuron Structures for Versatile Foundation Models: Augmenting Memory Capacity, Temporal Dynamics, and Information Segregation via Native Dendritic Mechanisms

Kexin Wang[1,2†], Runlin Shi[1,†], Di Shang[1,2†], Jianyu Xu[4], Yu Cheng[5], Bo Xu[1,3*], and Guoqi Li[1,2*]

[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Beijing, China
[3]Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Beijing, China
[4]Carnegie Mellon University, Pittsburgh, PA, USA [5]The Chinese University of Hong Kong, Hong Kong, China [†]These authors contributed equally to this work.
[*]Corresponding authors (xubo@ia.ac.cn; guoqi.li@ia.ac.cn)

Contents

S1 Supplementary explanation of the equivalence between dendrites and Linear models

In this supplementary section, we provide a detailed diagram and table of equivalences. As shown in Figure S1, the top left illustrates multi-branch dendritic modeling, the bottom left shows its mathematical formulation for a single branch, the top right depicts the workflow of linear attention, and the bottom right presents the mathematical expression of the linear model. We use consistent color highlights and bidirectional arrows to indicate the equivalence between these elements. Specifically, u_t in the linear model corresponds to the input current i_t in the dendritic model, k_t corresponds to the distribution weights of the input current, and S_t and u_t represent the memory state of the linear model and the membrane potential of the dendritic model, respectively. The factor α_t controls the state update in the linear model, which corresponds to the connection pattern within a single branch in dendritic modeling. The term q_t corresponds to how the dendritic model reads out from the membrane potential of dendritic compartments. The detailed correspondences are summarized in Table S1.

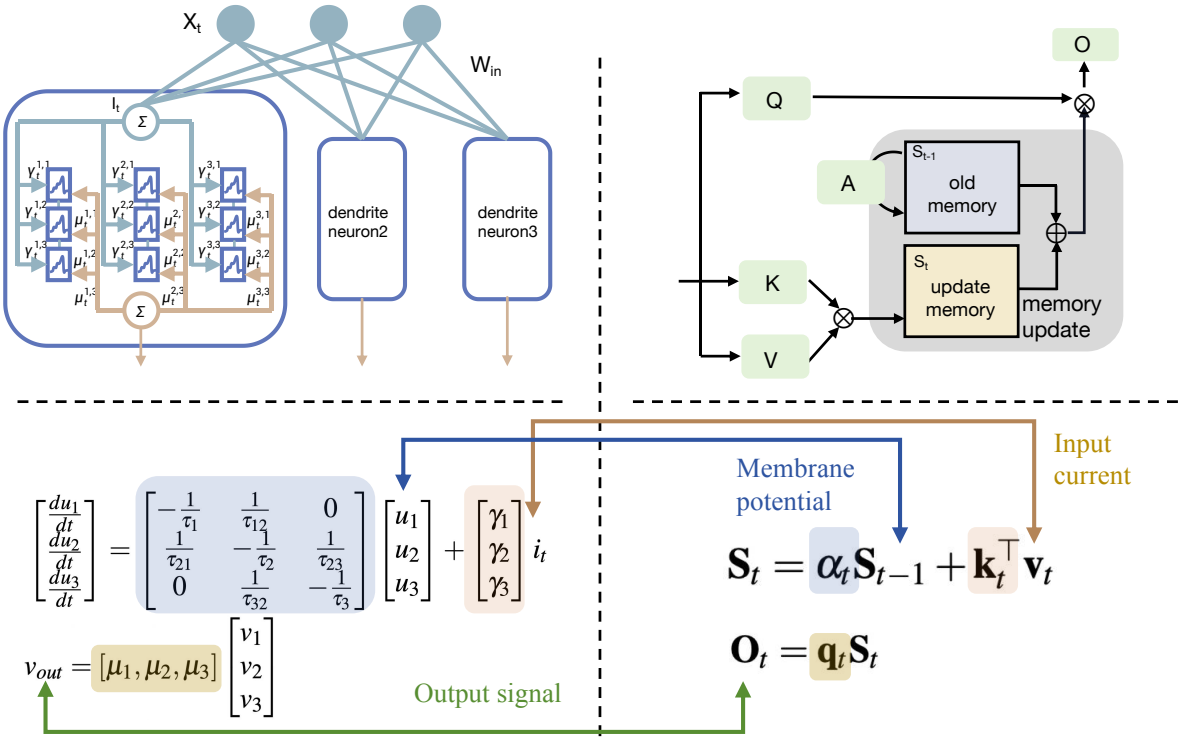

Figure S1. equivalence between dendrites and Linear models. Top left: multi-branch dendritic neuron modeling. Bottom left: mathematical formulation of a single branch with compartmental dynamics. Top right: workflow of linear attention. Bottom right: mathematical formulation of the linear model. Colored highlights and bidirectional arrows indicate equivalence between corresponding components in dendritic modeling and linear attention mechanisms.

	Input	Input Modulation	States	State Transition	State Readout
Dendrite	I	Γ	U	T	M
Linear Model	Value	Key	S	Gate	Query

Table S1. Equivalence mapping between dendritic modeling and linear models. Correspondence between dendritic neuron components and elements of linear models, including input, input modulation, states, state transitions, and readout mechanisms.

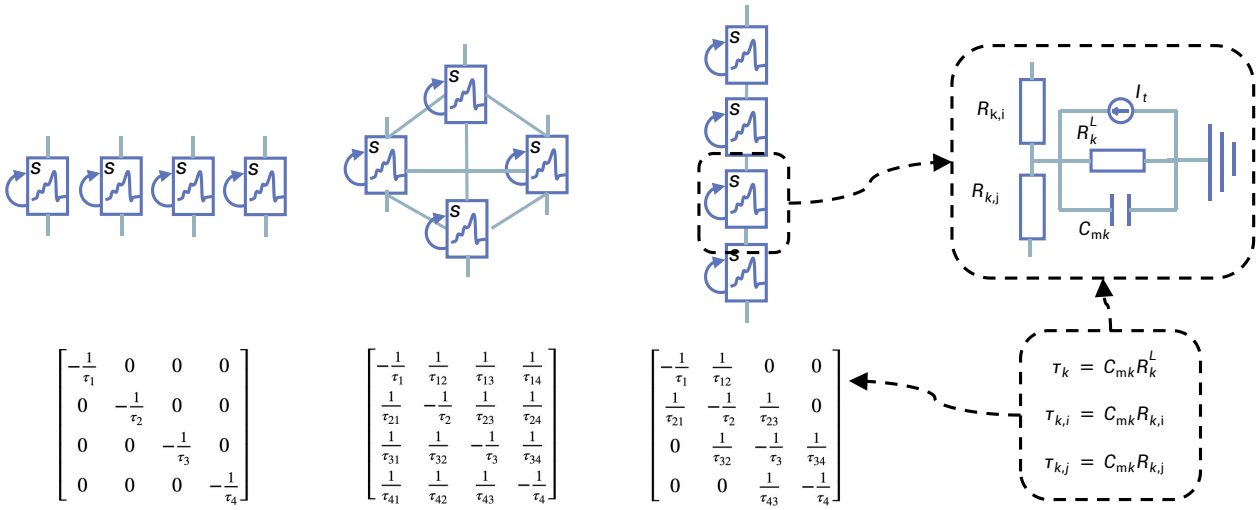

Figure S2. equivalence between single-branch dendrites and the state transfer matrices of linear models. From left to right, the figure illustrates three scenarios of single-branch dendritic circuits: (a) no interaction between compartments, (b) global interaction between compartments, and (c) biologically plausible local connectivity. The rightmost diagram shows a single compartment circuit. The matrices represent the decay of past states: in the non-interactive case (a), the transfer matrix has only diagonal elements; in the fully connected case (b), the matrix is dense; and in the locally connected case (c), the matrix exhibits a block-dense, globally sparse structure. These matrices are derived from the respective configurations of the circuit components, as shown in the rightmost diagrams.

S2 Supplementary explanation of the DendAttn routing mechanism

In this section, we provide a detailed workflow of DendAttn in Fig. S3. Before computing attention, the QKV inputs are split across heads. At each time step, each query q within a head is processed by a router that selects among the branches associated with that head. In addition, DendAttn maintains shared branches (e.g., *linear attn*$_1$), in which all tokens are involved in the computation. Finally, the outputs from all branches are aggregated through weighted summation to form the final output.

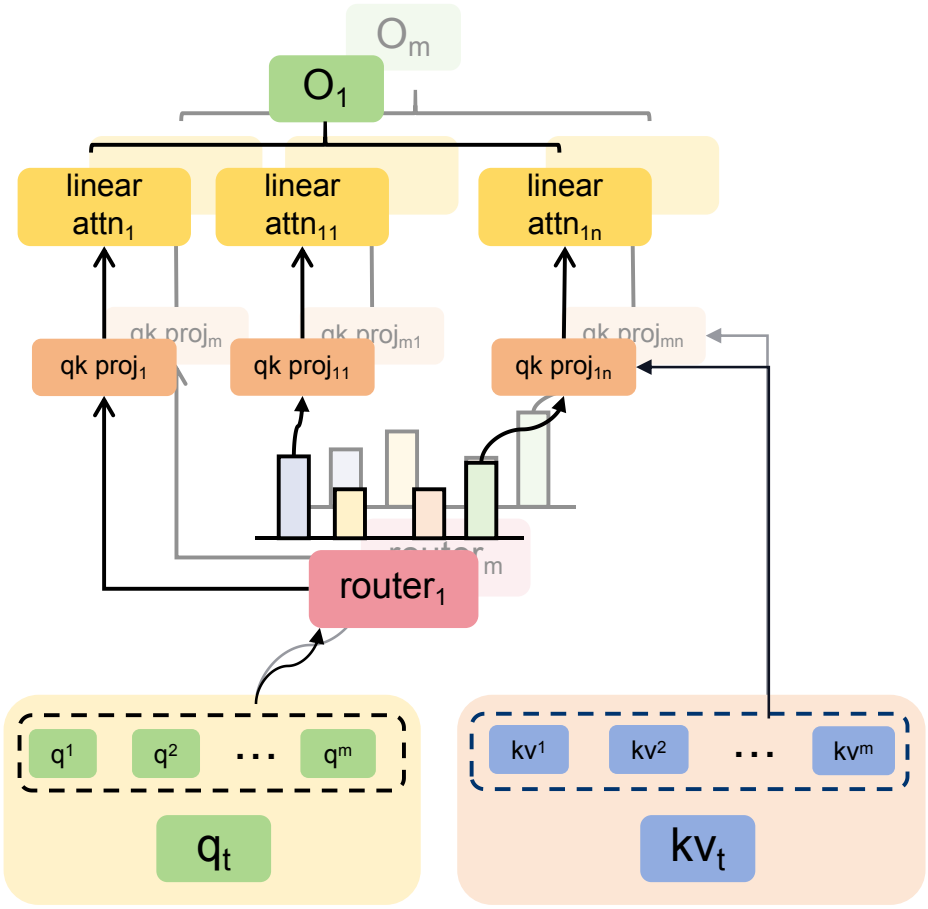

Figure S3. Detailed workflow of DendAttn. This figure illustrates the routing-based multi-branch linear attention mechanism. The dashed box containing q^i indicates the segment of the query vector assigned to and processed by the i-th head at time step t. Similarly, kv^i denotes the corresponding key and value segments handled by the same head. For each head i, the head-wise inputs q^i and k^i are further transformed by a branch-specific projection, producing $\{q^{i,b}, k^{i,b}\}_{b=1}^{B}$, where E denotes the number of branches. These projections map the original head representations into multiple distinct subspaces, enabling each branch to capture heterogeneous attention patterns. The value representation v^i is shared across branches and duplicated to form $\{v^{i,b}\}_{b=1}^{B}$, ensuring that different branches attend over a consistent value space while operating with their own specialized query–key transformations. After the projection and expansion steps, each branch (of each head) (i,b) is treated as an independent linear attention unit. The faded workflow in the background illustrates the same process for the m-th head, which operates on its own head-specific inputs and branch-specialized projections.

S3 Supplementary experimental data and visualization for Figure 3 (b)-(d)

In this section, we provide the detailed experimental results corresponding to Fig. 3. Table S2 reports the results of the 350M and 1.3B models on the commonsense reasoning benchmarks. Tables S3, S4, and S5 present the detailed results on NarrativeQA, QASPER, and GovReport, respectively. Table S6 summarizes the results of the inference efficiency experiments. Figure S4 presents the performance comparison of zero-shot commonsense reasoning at 350M scale.

Scale	Model	Wiki. ppl↓	Lamb. ppl↓	ARC-e acc↑	ARC-c acc↑	Hella. acc↑	Lamb. acc↑	PIQA acc↑	Wino. acc↑	Avg.
360M Params 15B Tokens L=24, d=1024	Transformer++	-	84.28	46.04	23.55	35.28	26.02	65.4	50.51	41.13
	RetNet	33.89	125.75	44.15	23.63	32.49	20.59	63.22	**53.04**	39.52
	GLA	29.33	86.9	44.49	23.81	34.03	24.06	63.93	51.07	40.23
	GSA	27.93	81.23	45.54	23.72	34.85	24.35	65.34	51.22	40.84
	DeltaNet	27.79	91.45	**46.84**	23.89	34.39	21.70	63.49	50.43	40.12
	Gated DeltaNet	26.46	58.35	46.34	23.29	36.03	27.19	**66.54**	51.7	41.85
	MoM (520M)	25.86	55.41	44.65	**24.74**	36.54	27.93	66.16	51.78	41.97
	DendAttn-mm	26.18	53.48	**46.84**	24.15	36.40	27.85	**66.54**	50.83	42.10
	DendAttn-mm-mb-4	25.62	45.22	46.55	22.78	36.64	29.81	**66.54**	52.09	42.40
	DendAttn-mm-mb-8	**25.24**	**44.94**	46.55	24.06	**37.32**	**30.80**	65.83	50.67	**42.54**
1.3B Params 100B Tokens L=24, d=2048	Transformer++†	17.61	19.29	55.01	28.07	49.21	40.95	70.08	56.27	49.93
	RetNet†	18.18	21.97	57.49	26.88	48.09	37.75	69.37	53.28	48.81
	HGRN2†	17.32	15.65	58.33	28.07	51.93	42.31	71.33	52.01	50.66
	GLA†	17.61	19.66	55.18	27.56	48.89	40.03	69.86	53.91	49.24
	GSA†	16.69	16.02	58.33	28.33	50.98	42.03	72.25	53.43	49.89
	Gated DeltaNet	17.14	18.80	56.82	27.39	49.77	39.94	71.76	51.78	49.58
	MoM†	16.64	14.83	55.35	27.99	50.95	43.43	71.27	56.83	49.97
	DendAttn-mm	16.00	**13.05**	59.47	28.16	52.78	45.57	71.44	**57.30**	52.45
	DendAttn-mm-mb	**15.84**	13.18	**59.81**	**28.67**	**53.49**	**45.92**	**71.98**	**57.30**	**52.86**

Table S2. Performance comparison across various models on commonsense reasoning benchmarks. The table is divided into two sections corresponding to 350M and 1.3B settings. Each model is evaluated on a diverse suite of tasks, including language modeling (WikiText and LAMBADA perplexity) and commonsense reasoning benchmarks (ARC-e, ARC-c, HellaSwag, LAMBADA, PIQA, and Winogrande). *ppl*↓ denotes lower perplexity values indicating better language modeling performance, while *acc*↑ represents accuracy metrics where higher values indicate better task performance. The last column reports the average score across all *acc*↑ evaluation tasks.

Model	Context Length						
	2k	4k	8k	16k	32k	64k	128k
GLA	0.67	0.68	0.68	0.68	0.68	0.68	0.68
HGRN	0.78	0.79	0.82	0.81	0.82	0.82	0.82
Gated-Deltanet	**0.87**	0.83	0.79	0.77	0.77	0.77	0.77
MoM	0.67	0.66	0.64	0.66	0.66	0.65	0.66
DendAttn	0.824	**0.87**	**0.91**	**0.95**	**0.92**	**0.92**	**0.92**

Table S3. F1 Scores on NarrativeQA Task with Different Context Lengths. The table compares five representative architectures (GLA, HGRN, Gated-Deltanet, MoM, and DendAttn) under multiple input context settings ranging from 2k to 128k tokens. Each column corresponds to a specific context length, and each row reports the corresponding evaluation value for a given model.

Model	Context Length						
	2k	4k	8k	16k	32k	64k	128k
GLA	11.17	11.00	10.63	10.67	10.86	10.87	10.95
HGRN	9.88	9.95	9.82	9.75	9.70	9.68	9.80
Gated-Deltanet	13.71	13.59	12.79	12.95	12.83	12.84	12.91
MoM	11.23	11.17	11.10	11.29	11.67	11.69	11.77
DendAttn	**14.12**	**14.38**	**14.02**	**13.97**	**13.91**	**13.81**	**13.86**

Table S4. F1 Scores on QASPER Task with Different Context Lengths. This table displays the F1 scores on the QASPER task at varying context lengths (2k to 128k). Each column corresponds to a specific context length, while each row lists the corresponding metric values for one model. Across all evaluated context lengths, DendAttn consistently obtains the highest F1 scores, outperforming all competing models from 2k to 128k context lengths, highlighting its superior robustness and scalability for long-context question answering.

Model	Metric	Context Length						
		2k	4k	8k	16k	32k	64k	128k
GLA	ROUGE-1	2.85	2.87	2.87	2.87	2.84	2.86	2.88
	ROUGE-2	0.67	0.68	0.68	0.68	0.68	0.68	0.68
	ROUGE-L	2.57	2.60	2.60	2.60	2.58	2.60	2.61
HGRN	ROUGE-1	3.13	3.19	3.26	3.27	3.25	3.26	3.27
	ROUGE-2	0.78	0.79	0.82	0.81	0.82	0.82	0.82
	ROUGE-L	2.79	2.85	2.89	2.90	2.90	2.90	2.90
Gated-Deltanet	ROUGE-1	3.24	3.19	3.06	3.01	3.00	3.01	3.01
	ROUGE-2	**0.87**	0.83	0.79	0.77	0.77	0.77	0.77
	ROUGE-L	**2.91**	2.86	2.77	2.73	2.73	2.73	2.73
MoM	ROUGE-1	2.76	2.66	2.61	2.63	2.63	2.63	2.62
	ROUGE-2	0.67	0.66	0.64	0.66	0.66	0.65	0.66
	ROUGE-L	2.52	2.48	2.45	2.47	2.47	2.47	2.46
DendAttn	ROUGE-1	**3.25**	**3.30**	**3.41**	**3.51**	**3.47**	**3.42**	**3.44**
	ROUGE-2	0.824	**0.87**	**0.91**	**0.95**	**0.92**	**0.92**	**0.92**
	ROUGE-L	2.87	**2.91**	**2.98**	**3.03**	**3.01**	**2.98**	**3.01**

Table S5. Model Performance on GovReport Task with Different Context Lengths. This table reports ROUGE-1, ROUGE-2, and ROUGE-L scores for models evaluated under varying context lengths (2k–128k). Across all evaluated context lengths above 4k, DendAttn consistently delivers the strongest performance, demonstrating both superior accuracy in key content extraction and greater robustness to increasing sequence length. These results indicate that DendAttn preserves both local coherence and global structural fidelity more effectively than competing approaches.

Sequence Length	Inference Time (ms)		Inference Memory (GB)		Inference FLOPs (GFLOPs)	
	DendAttn	Transformer	DendAttn	Transformer	DendAttn	Transformer
1k	1.978	0.081	0.226	0.233	0.01	1.07
2k	1.948	0.132	0.227	0.236	0.02	2.15
4k	2.196	0.255	0.229	0.242	0.04	4.30
6k	2.482	0.816	0.231	0.248	0.05	6.44
8k	2.452	0.817	0.233	0.254	0.07	8.59
10k	2.855	1.100	0.235	0.260	0.09	10.74
12k	3.278	3.954	0.237	0.266	0.10	12.89
14k	3.821	4.398	0.239	0.272	0.12	15.03
16k	4.286	6.990	0.241	0.278	0.14	17.18
32k	7.958	20.960	0.265	0.326	0.27	33.56
64k	15.414	80.571	0.289	0.418	0.53	67.11
96k	22.876	164.636	0.321	0.512	0.79	100.66
128k	30.389	271.118	0.354	0.608	1.05	134.22
192k	45.492	610.732	0.418	0.794	1.58	201.33
256k	60.477	1061.330	0.483	0.980	2.10	268.44
384k	91.023	1671.220	0.613	1.300	3.15	402.65
512k	121.528	4082.428	0.742	1.700	4.20	536.87

Table S6. Inference Efficiency Comparison: DendAttn vs Transformer. This table compares the inference efficiency of the DendAttn and the Transformer across different sequence lengths. The metrics provided include inference time (in milliseconds), memory usage (in gigabytes), and the number of FLOPs (in giga floating-point operations). The bolded entries indicate the sequence lengths at which DendAttn begins to outperform Transformer. DendAttn consistently shows superior inference time and lower memory usage compared to the Transformer model, especially at longer sequence lengths. Notably, when the sequence length reaches 512k, DendAttn achieves a 33.7x speedup in inference time and reduces memory usage to 43.6%. DendAttn demonstrates better efficiency with fewer FLOPs, indicating its advantage in large-scale deployments. It is worth noting that, in short-sequence regimes, Transformers exhibit higher inference speed than DendAttn for two main reasons. First, when the sequence length $n < d$ (the model dimension), the theoretical computational complexity of linear attention becomes higher than that of the Transformer. Second, Transformers benefit from highly optimized inference kernels such as FlashAttention-3, which further accelerate computation for short sequences. As a result, DendAttn does not fully manifest its speed advantages in this setting.

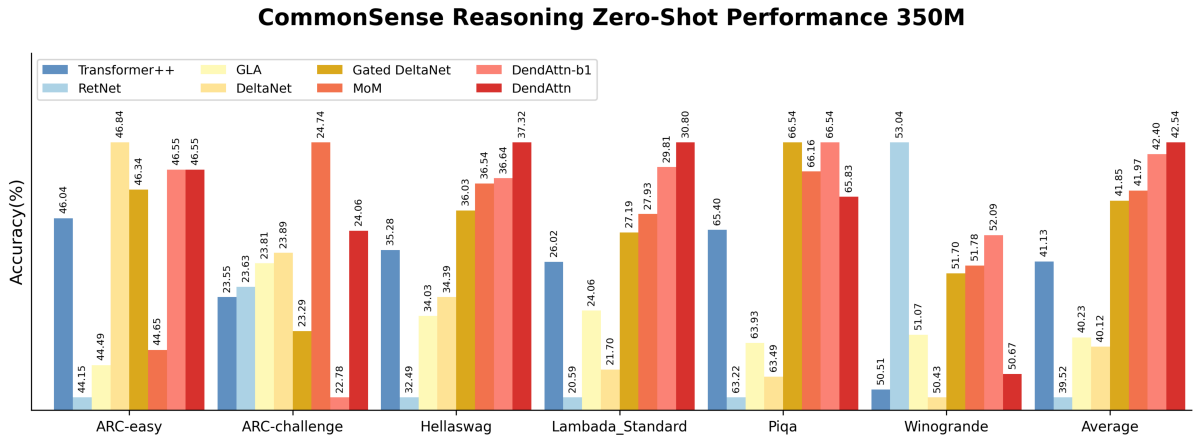

Figure S4. Zero-shot commonsense reasoning at 350M scale. Zero-shot commonsense reasoning across ARC-easy, ARC-challenge, HellaSwag, LAMBADA, PiQA, and Winogrande at 350M scale. DendAttn-b1 denotes the single-branch variant of DendAttn. It serves as an ablation of the block-sparse gating mechanism when compared to Gated-DeltaNet, and as an ablation of the multi-branch structure when compared to DendAttn.

S4 Supplementary visualization data for Figure 3 (d)

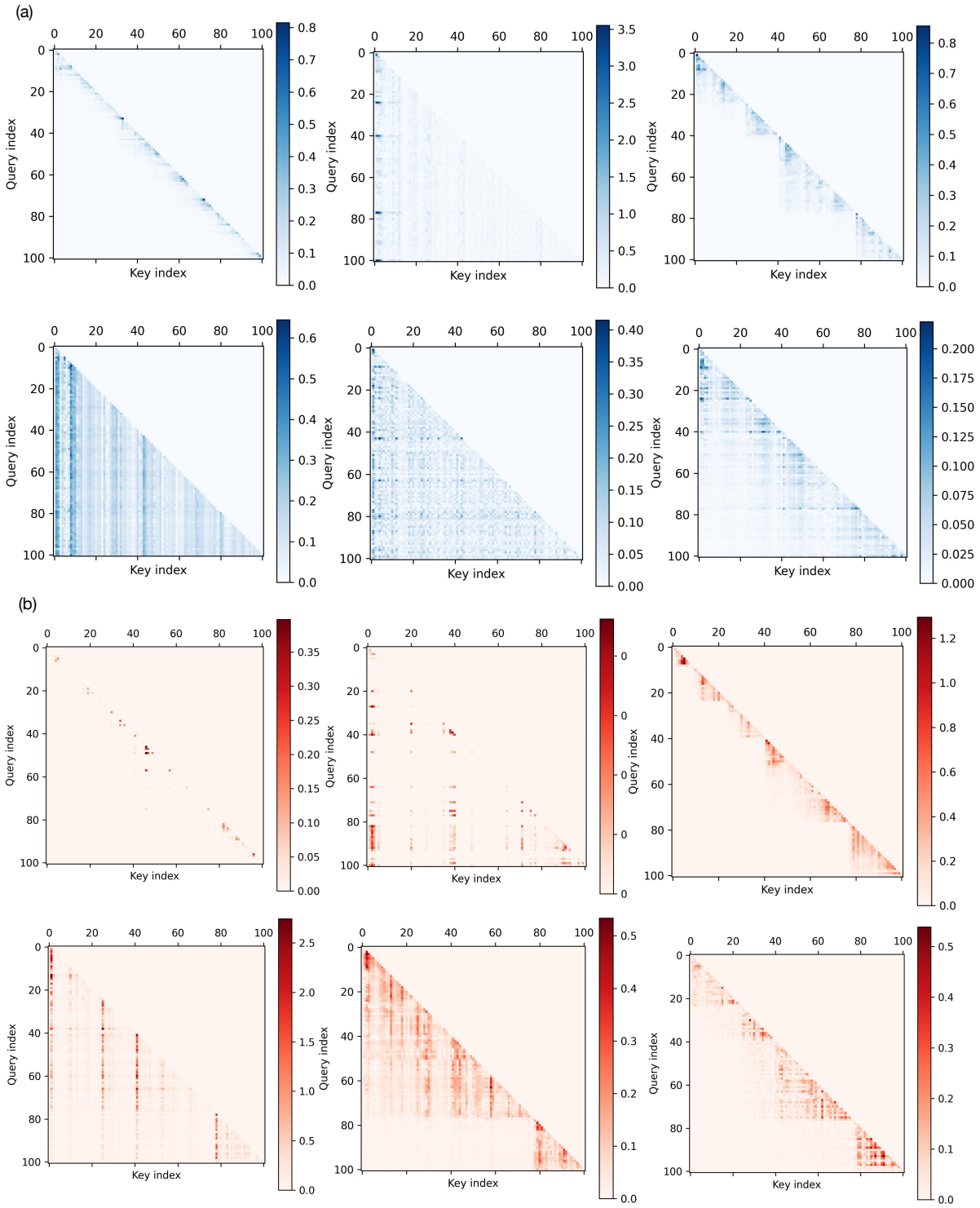

Figure S5. Attention Maps Comparison for Gated Deltanet and DendAttn Models. (a): The attention maps for Gated Deltanet are displayed, showing the results for different heads. The heatmaps represent the attention weights between query and key indices, with varying intensity indicating the strength of attention at different positions. (b): The attention maps for DendAttn are presented, showing the results across different heads and branches. Overall, the attention maps of DendAttn exhibit a more globally sparse yet locally concentrated structure, reflecting superior information processing.

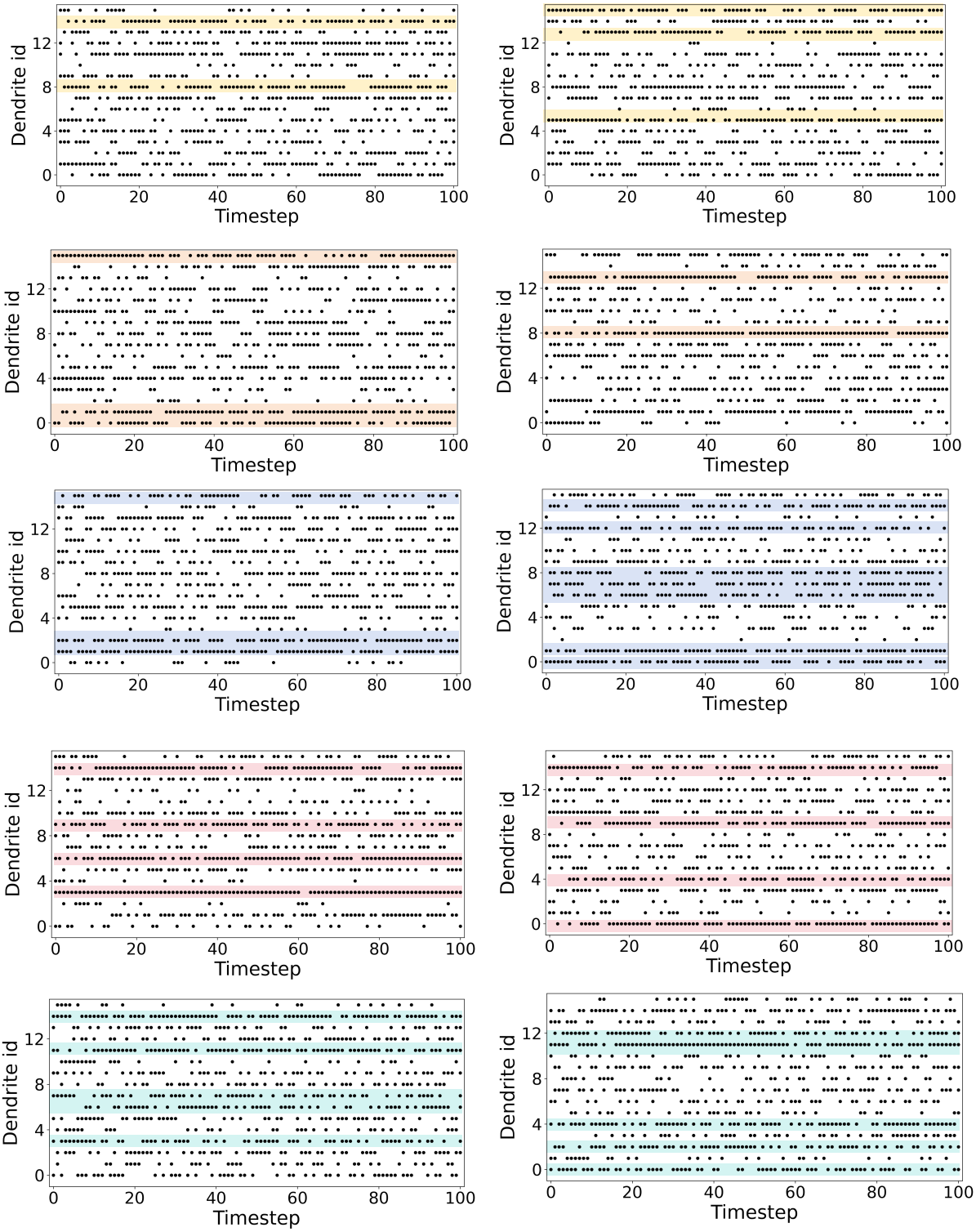

Figure S6. Visualization of the branch selection of tokens in DendAttn. The figure shows the dendritic branch selection over time, with the x-axis representing tokens (timesteps) and the y-axis representing branch IDs. The chart is organized such that every four branches are grouped into the same head. The color-coded horizontal bars represent branches with higher selection frequencies.

S5 Supplementary experimental data and implementation details for Figure 4

Model Size	Model	Dataset Performance						Avg.
		FDA	SWDE	SQUAD	NQ	TriviaQA	Drop	
350M	transformer++	46.14	25.87	33.22	18.94	45.97	20.03	31.695
	gla	1.26	16.78	27.85	12.77	43.90	17.68	20.040
	retnet	4.36	8.43	20.32	6.78	37.20	16.63	15.620
	gsa	7.90	15.00	22.14	12.89	41.53	15.33	19.132
	deltanet	30.34	19.49	26.54	14.29	43.19	15.47	24.887
	gated_deltanet	22.98	20.52	27.24	15.52	43.66	18.21	24.688
	DendAttn	**36.42**	**31.96**	**30.84**	**18.02**	**47.33**	**21.51**	**31.010**
1.3B	transformer++	58.67	31.12	41.01	27.05	59.83	21.66	39.890
	gla	27.61	30.93	35.04	22.27	56.28	19.45	31.930
	hgrn	2.63	11.81	21.80	13.62	47.99	16.77	19.103
	gsa	23.25	32.80	35.61	22.96	56.99	20.60	32.035
	gated_deltanet	37.69	40.21	36.14	23.73	58.71	20.32	36.133
	mom	41.14	34.40	37.72	24.01	58.71	21.27	36.208
	DendAttn	**47.41**	**39.18**	**38.43**	**25.82**	**59.95**	**22.86**	**38.942**

Table S7. Real-World Retrieval Tasks. Performance comparison on real-world retrieval tasks, including datasets such as FDA, SWDE, SQUAD, NQ, TriviaQA, and Drop. Results are shown for different model sizes (350M and 1.3B), with average performance across all datasets provided.

Model	Single-Doc QA	Multi-Doc QA	Summarization	Few-shot	Code	Average
Transformer++	9.25	6.33	10.61	33.46	48.30	21.59
HGRN	7.77	4.60	8.16	14.31	44.01	15.77
GLA	9.68	6.10	8.64	27.80	39.86	18.42
GSA	9.90	6.30	8.53	26.55	42.93	18.84
Gated DeltaNet	10.03	6.16	8.59	28.12	40.84	18.75
MoM	9.16	6.31	6.97	28.00	**47.81**	19.65
DendAttn	**10.07**	**6.81**	**9.57**	**35.85**	46.32	**21.72**

Table S8. LongBench Task Performance. All results are reported for the English tasks average, using the 1.3B model.

Model	Sequence Length and Embedding Dimension											
	L256				L512				L1024			
	d32	d64	d128	d256	d32	d64	d128	d256	d32	d64	d128	d256
DendAttn	**98.48**	**100**	**100**	**100**	**70.68**	**100**	**100**	**100**	**25.02**	**59.17**	95.21	**100**
gated_deltanet	67.37	99.07	**100**	**100**	5.64	88.58	99.23	**100**	0.00	16.36	97.17	**100**
retnet	86.41	**100**	**100**	**100**	13.48	83.31	**100**	**100**	0.00	4.12	0.00	0.00
hgrn	0.00	1.90	4.80	19.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mamba2	84.85	**100**	**100**	**100**	10.94	92.57	**100**	**100**	0.00	0.00	**100**	**100**

Table S9. Model Performance on Multi-Query Associative Recall (MQAR) Experiment. Performance of different models (DendAttn, gated-deltanet, retnet, hgrn, and mamba2) on the MQAR experiment, evaluated across different sequence lengths (L256, L512, and L1024) and embedding dimensions (d32, d64, d128, and d256).

Model	Task and Context Length											
	s-nich-1				s-nich-2				s-nich-3			
	1k	2k	4k	8k	1k	2k	4k	8k	1k	2k	4k	8k
gla	**100**	84.0	19.6	3.8	99.8	99.2	57.2	22.6	72.6	78.0	17.8	4.0
hgrn	88.0	56.4	32.2	14.0	77.8	42.8	22.0	6.2	0.4	0.0	0.0	0.0
gdn	**100**	**100**	**100**	98.0	**100**	**100**	83.6	24.2	99.2	21.4	53.4	**21.4**
mom	**100**	**100**	**100**	98.0	**100**	99.8	48.6	**35.4**	99.6	90.4	28.4	19.6
DendAttn	**100**	**100**	**100**	**100**	**100**	**100**	**93.8**	29.8	**100**	**92.0**	**70.4**	19.2
	multi-key				multi-query				multi-value			
	1k	2k	4k	8k	1k	2k	4k	8k	1k	2k	4k	8k
gla	42.2	31.8	14.4	7.2	52.05	29.3	13.95	6.5	36.0	26.75	14.5	6.95
hgrn	10.6	9.8	13.2	2.6	5.5	5.9	5.1	0.4	1.85	3.4	6.35	0.15
gdn	**58.8**	45.8	28.2	11.4	62.05	47.8	21.3	8.15	72.8	48.3	22.0	12.95
mom	43.8	34.2	27.6	8.2	**85.3**	73.1	41.8	14.35	**86.35**	**75.6**	47.35	14.1
DendAttn	55.6	**53.2**	**33.8**	**16.4**	**85.3**	**73.05**	**49.45**	**24.0**	79.45	70.01	**55.6**	**24.0**

Table S10. NIAH Task Performance. Model performance on the NIAH task, assessing the ability of models to perform associative memory tasks. The performance is reported across different context lengths from 1k to 8k.

S6 Proof for Dendritic Implementation of Multitasking

In this section, we will present the proof of the effect of dendritic information segregation mechanisms on multi-task learning. First, we introduce some symbol definitions. When the dendritic segregation mechanism is not used, and all tasks share the same parameters, the parameters are denoted as θ; when the dendritic segregation mechanism is used $\theta_i = \{\theta_s, \theta_{ri}\}$ represent the parameters for task i, where θ_s is the shared branch, and θ_{ri} is the branch parameters activated by task i, $i \in \{1, 2, \ldots, m\}$. Li is the loss function for task i.

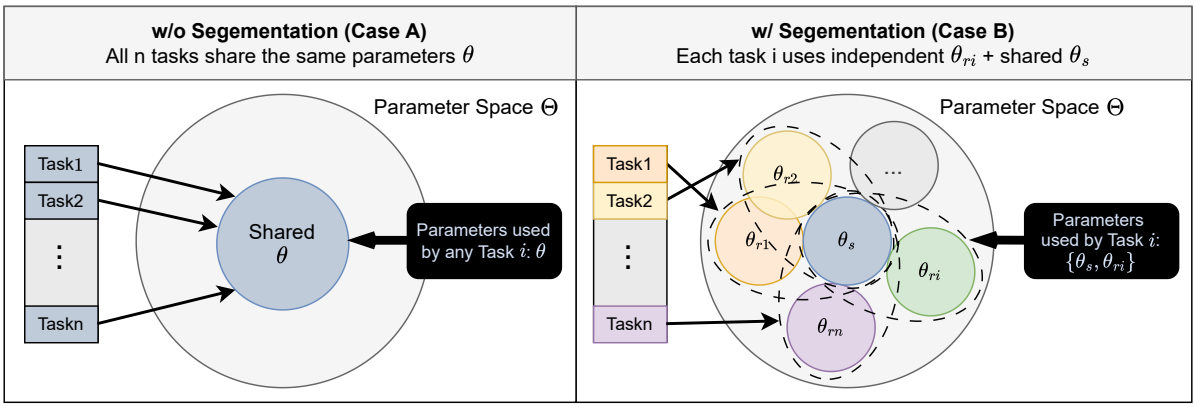

Figure S7. Parameter distribution in multi-task learning with and without information segmentation. This figure illustrates the parameter space distribution under two conditions: (left) without segmentation (Case A), where all tasks share the same parameters θ, and (right) with segmentation (Case B), where each task i uses both independent task-specific parameters θ_{ri} and shared parameters θ_s.

Theorem 1: Multi-branch dendrites with information-segregation mechanisms can converge to better solutions than fully activated ones in multi-task settings. *For different dendritic activation patterns, consider two scenarios: without information segregation mechanism (Case A) and with information segregation mechanism (Case B). The total loss functions are defined as follows:*

$$L_A(\theta) = \sum_{i=1}^{m} L_i(\theta), \quad \text{(\textbf{Case A}: shared parameter space, as shown in the left portion of Fig.S7)}$$

and for $\theta_i = \{\theta_s, \theta_{ri}\}, i = 1, \ldots, m$:

$$L_B(\theta_1, \theta_2, \ldots, \theta_m) = \sum_{i=1}^{m} L_i(\theta_i), \quad \text{(\textbf{Case B}: separate parameter spaces for each task, as shown in the right portion of Fig.S7).}$$

such that the following holds:

$$\min_{\theta} L_A(\theta) \geq \min_{\{\theta_i, i=1,\ldots,m\}} L_B(\theta_1, \theta_2, \ldots, \theta_m).$$

Proof. When the tasks select the same routing branch, i.e., $\theta_{r1} = \theta_{r2} = \ldots = \theta_{rm}$, Case A with loss L_A can be considered a special case of Case B with loss L_B. Let $\mathscr{C}_A = \{(\theta_s, \theta_{r1}, \theta_{r2}, \ldots, \theta_{rm}) \in \Theta_s^{m+1} \mid \theta_{r1} = \theta_{r2} = \ldots = \theta_{rm}\}$ and $\mathscr{C}_B = \{(\theta_s, \theta_{r1}, \theta_{r2}, \ldots, \theta_{rm}) \in \Theta_s^{m+1}\}$ represent the parameter spaces of a dendritic branch without and with a routing mechanism, respectively. Obviously:

$$\mathscr{C}_A \subseteq \mathscr{C}_B$$

Then it can be proved that:

$$\min_{\theta} \left[\sum_{i=1}^{m} L_i(\theta)\right] = \min_{\{\theta_i | \theta_1 = \ldots = \theta_m\}} \left[\sum_{i=1}^{m} L_i(\theta_i)\right] \geq \min_{\{\theta_i, i=1,\ldots,m\}} \left[\sum_{i=1}^{m} L_i(\theta_i)\right] \Rightarrow \min_{\theta} L_A(\theta) \geq \min_{\{\theta_i, i=1,\ldots,m\}} L_B(\theta_1, \theta_2, \ldots, \theta_m)$$

Theorem 2: Strict Inequality Condition. *If L_i, for $i = 1, \ldots, m$, are continuous and have unique optimal points $\theta_1^*, \theta_2^*, \ldots, \theta_m^*$, where θ_i^* is the unique optimal point for task i, for $\forall \theta \in \Theta$, the following holds:*

$$L_A(\theta) > L_B(\theta_1^*, \theta_2^*, \ldots, \theta_m^*),$$

where θ_i^* are the unique minimizers of L_i.

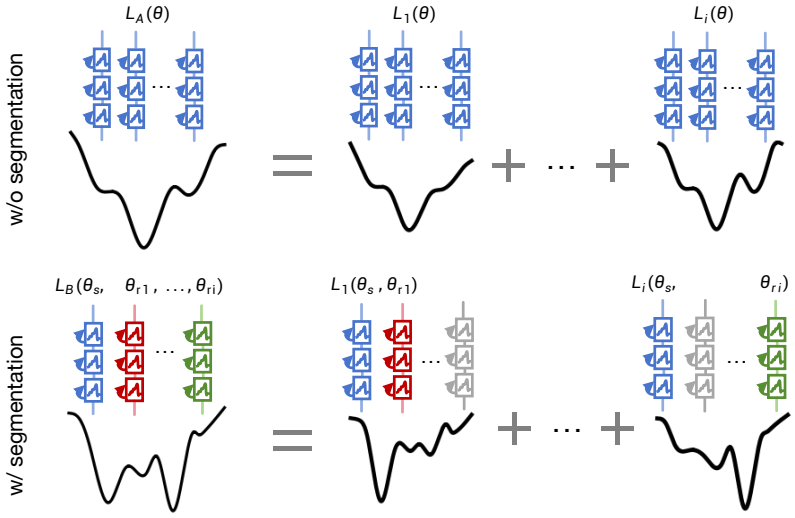

Figure S8. Diagram illustrating the theoretical definition of multi-task information segregation.

Proof. When $\theta \neq \theta_1^*$, we have $L_1(\theta) > L_1(\theta_1^*)$. Similarly, for $\theta \neq \theta_i^*$, we also have $L_i(\theta) > L_i(\theta_i^*)$. Therefore, it is impossible for any $\theta \in \Theta$ to minimize $\{L_i \mid i \in 1, \ldots, m\}$ simultaneously. Hence:

$$\begin{cases} L_1(\theta) = L_1(\theta_1^*) + \varepsilon_1, & \varepsilon_1 \geq 0 \\ L_2(\theta) = L_2(\theta_2^*) + \varepsilon_2, & \varepsilon_2 \geq 0 \\ \quad \ldots \\ L_m(\theta) = L_m(\theta_m^*) + \varepsilon_m, & \varepsilon_m \geq 0 \end{cases}$$
$$\Rightarrow \sum_{i=1}^{m} L_i(\theta) = \sum_{i=1}^{m} L_i(\theta_i^*) + \sum_{i=1}^{m} \varepsilon_i$$

Assume that $\sum_{i=1}^{m} \varepsilon_i = 0$. This implies that $\varepsilon_i = 0$ for all $i = 1, ..., m$, which would mean that for each i, we have:

$$L_i(\theta) = L_i(\theta_i^*) \quad \text{for all} \quad \theta \in \Theta.$$

However, this contradicts the fact that the $L_i(\theta)$ have unique optimal θ_i^* for each i. Therefore, the assumption that $\sum_{i=1}^{m} \varepsilon_i = 0$ must be false. Hence, we conclude that:

$$\sum_{i=1}^{m} \varepsilon_i > 0.$$

Thus, we can proceed with the following:

$$\sum_{i=1}^{m} L_i(\theta) = \sum_{i=1}^{m} L_i(\theta_i^*) + \sum_{i=1}^{m} \varepsilon_i, \sum_{i=1}^{m} \varepsilon_i > 0 \Rightarrow \min_{\theta \in \Theta} L_A(\theta) > \min_{\theta_i \in \Theta} L_B(\theta_1, \theta_2, \ldots, \theta_m)$$

S7 Supplementary experimental data and implementation details for Figure 5

MTL Settings		
Data	Vocabulary size	128
	Sequence length	256
	Fraction of noise	0.6
	Noise vocabulary size	32
	Number of tokens to copy	64
	Number of training examples	see Fig. 5
	Number of test examples	1280
Model	Hidden size	128
	Head dim	64
	Expand_v	1
	Number of layers	4 (attn–swiglu–attn–swiglu)
Training	Optimizer	AdamW
	Optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$
	Dropout	None
	Batch size	128
	Training epochs	100
	Learning rate schedule	cosine decay
	Base learning rate	0.0005
	Minimum learning rate	1×10^{-6}
	Weight decay	0.1

Table S11. MTL Settings. Training, dataset, and model settings for multi-task learning. In the data setting, *Fraction of noise* and *Noise vocabulary size* are used for the noisy in-context recall task, whereas *Number of tokens to copy* applies to the selective copying task. *Number of training examples* follows the multiple values shown in Fig. 5(c).

S8 Supplementary visualization data for multitask

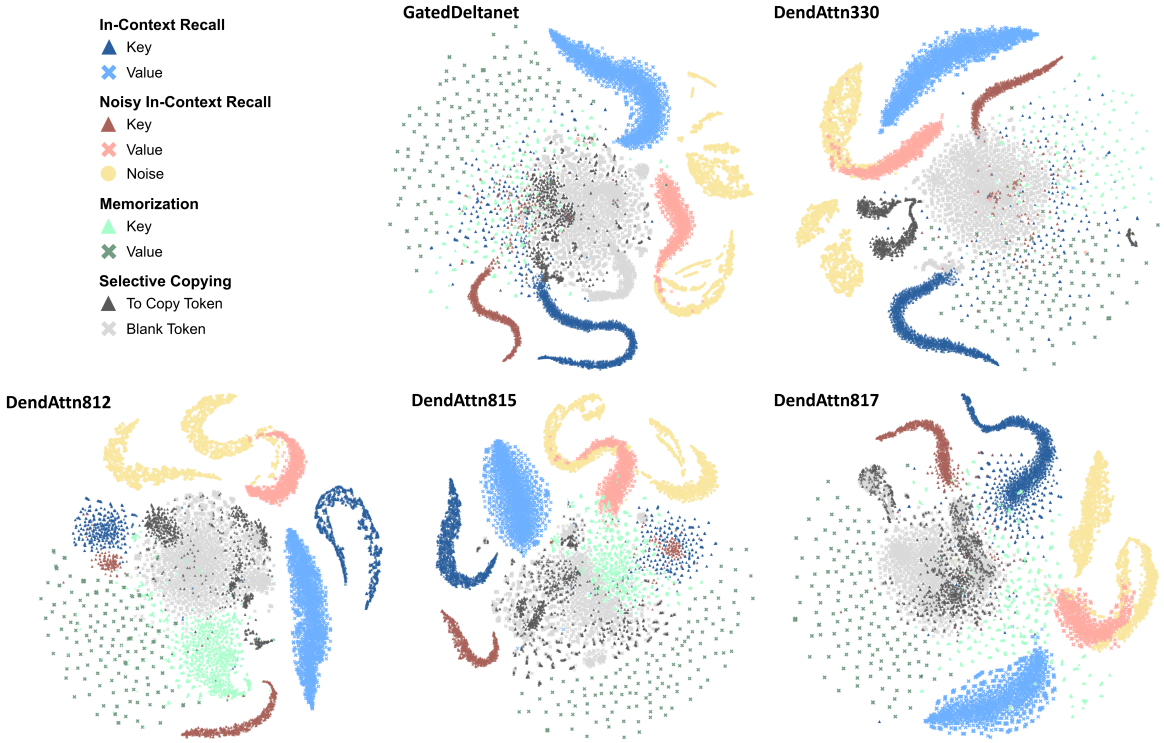

Figure S9. t-SNE token feature visualization. In this figure, GatedDeltanet and DendAttn815 are presented for the first time. Building on the analysis in the main text, GatedDeltanet shows noticeably more disordered clustering compared with DendAttn330, with tokens from nearly all tasks intermixed across a large central region of the plot. In contrast, DendAttn815 exhibits a trend positioned between DendAttn812 and DendAttn817: while most of its advantageous clustering structure is preserved, the entanglement of value and noise tokens from Noisy In-context Recall becomes more pronounced, and the clusters of recall-related key tokens from In-context Recall and Noisy In-context Recall begin to disperse.