A Self-Pruning Transformer: Extreme KV-Cache Compression with Universal Attention

Davis Wertheimer[1,*], **Haochen Shen**[2,*], **Ahan Gupta**[2,*], **Derrick Liu**[1],
Yu Chin Fabian Lim[1], **Mudhakar Srivatsa**[1], **Raghu K. Ganti**[1],
Minjia Zhang[2], **Naigang Wang**[1]
[1]IBM Research, USA [2]SSAIL Lab, University of Illinois Urbana-Champaign
davis.wertheimer@ibm.com {hshen14,ag82}@illinois.edu

Abstract

The large KV-cache size of modern LLMs creates a barrier to efficient deployment. Recent work has explored replacing attention layers' RoPE positional embeddings with alternative decay-based mechanisms, which can then be used to prune KV-cache during inference. However, these decay functions have limited expressivity, and in practice devolve into sliding-window-like eviction patterns. In this work, we propose a unifying framework for complementary and novel decay mechanisms, capturing complex key statistics and interactions while preserving expressive RoPE embeddings and Softmax attention. The resulting *Universal Attention* is a highly expressive and end-to-end trainable architecture, whose composite decay mechanism acts as a natural, *adaptive* pruning criterion, removing tokens that contribute least to attention computation. Experimentally, Universal Attention achieves state-of-the-art $10\times$ compression on natural language and synthetic task data, while *improving* downstream performance compared to both state-of-the-art baselines and unpruned oracles. It further demonstrates superior long-context generalization with unprecedented $25\times$ compression at length 16k.

1 Introduction

Large Language Models (LLMs) power many of the applications we use today (OpenAI, 2025; Anthropic, 2026; Google DeepMind, 2026; xAI, 2026). However, they predominantly employ softmax-attention layers (Vaswani et al., 2017), which compute pairwise dot-products between *all* tokens in the input sequence. This entails preserving *every* token, for *each* layer, as the "KV-cache" during inference time, consuming gigabytes (if not terabytes) of device memory, often several times larger than the model itself. This acts as a significant obstacle to deploying LLMs efficiently at scale.

To effectively reduce KV-cache sizes, recent research has proposed the addition of a "forget-gate" to the softmax self-attention formulation, enabling unimportant key/value tokens from the past to be deleted. For example, forgetting transformers (Lin et al., 2025a;b) (FoX) introduces and applies a scalar data-dependent decay term for each new token entering the cache, accumulating over time as an additive bias to the attention matrix. This effectively zeros out certain query-key products after softmax normalization, allowing the corresponding key/value pair to be safely evicted. However, these forget-gates are relatively inexpressive, yielding simple patterns that devolve to soft sliding-windows, which can only evict limited key/value entries before downstream model performance degrades.

In this work, we introduce *Universal Attention* (UA), a unifying framework that combines first- and second-order statistics of stored keys to produce uniquely expressive forget gates. Some components of our framework align with and generalize decay mechanisms from popular Linear Attention models, such as DeltaNet (Yang et al., 2024b; 2025a) & Mamba (Gu

*Equal contribution.

& Dao, 2023; Dao & Gu, 2024). But while these models rely on a constant-sized $\mathsf{d} \times \mathsf{d}$ state that stores at most d linearly independent key/value associations, we combine these and novel mechanisms into a single pruning technique for the full KV-cache of SoftMax attention, while preserving RoPE embeddings (Su et al., 2024). We additionally implement hardware-aware custom UA kernels for high-throughput training. The resulting UA architecture is efficient and expressive, achieving over $10\times$ KV-cache compression on natural language and synthetic task data, for a 1B transformer model trained on 50B tokens. This memory overhead is lower than state-of-the-art pruning methods and even $O(1)$ linear attention architectures, despite *improved* downstream performance, even compared to unpruned oracle baselines. Cache pruning in UA is also adaptive and input-dependent, generalizing to sequence lengths not observed during training, with unprecedented $25\times$ compression in long contexts of 16k.

2 Related Work

Shrinking the KV cache is a topical problem with several orthogonal approaches. These include dedicated architectures with sublinear cache growth at inference time, techniques to reduce KV-cache access without reducing the overall size, post-training pruning techniques for vanilla architectures, and combined architectures that use trained forget gates to decay KV cache entries. UA sits at the intersection of these approaches.

Sub-quadratic architectures: Linear attention models collapse the sequence length into a constant, $\mathsf{d}_{state} \times \mathsf{d}_{state}$-sized, recurrent hidden state Katharopoulos et al. (2020). However, when input sequence lengths exceed d_{state}, retrieval from the hidden state becomes noisy. To mitigate this issue, models in this family impose a recency bias by decaying past key/value associations (Gu & Dao, 2023; Dao & Gu, 2024; Yang et al., 2025a). While still highly expressive, these models experience considerable drops in quality for complex long-context tasks compared to Softmax self-attention (Jelassi et al., 2024; Ben-Kish et al., 2025).

In addition, despite their O(1) memory in the limit, such models can fail to reduce overhead in practice: for typical model configurations, linear attention consumes *greater* memory for sequences shorter than 1k tokens - a sizable portion of the usual training length (see Appendix B). Fixes aimed at improving model capacity and performance only exacerbate the issue: hybrid models interleaving full attention with linear attention still have the same break-even point, but become O(L) in the limit, reducing best-case memory savings to 8-10$\times$ for typical settings where 1/8 to 1/10 of layers are switched to Softmax (NVIDIA et al., 2025; Lieber et al., 2024; Team et al., 2024). Log-Linear Attention (Guo et al., 2026) maintains multiple state buffers, pushing out the break-even point even further. UA instead applies an expressive decay mechanism to Softmax self-attention directly, avoiding the memory lower-bound imposed by a constant-sized recurrent state.

Sparse attention: Zaheer et al. (2020); Child et al. (2019); Yuan et al. (2025) compute only a subset of the attention matrix. These techniques range from computing static patterns (such as sliding-windows around the diagonal) to dynamic patterns based on custom data-dependent heuristics. Simple patterns like sliding-window-attention reduce KV-cache sizes but degrade long-context capability unless interleaved with more expressive layers (Jiang et al., 2023; Beltagy et al., 2020; Team et al., 2025), while content-dependent dynamic sparsity provides a more expressive alternative (Zhang et al., 2025a; Shi et al., 2025). These dynamic sparse attention patterns are either implemented by modifying full-attention masks to zero out unnecessary query-key dot-products, or by selectively reading queries and keys from memory to compute a partial attention matrix. Neither implementation reduces KV-cache sizes.

KV-cache eviction: Methods such as H_2O and SnapKV (Zhang et al., 2023; Li et al., 2024) use custom heuristics during the prefill & decode stages of LLM inference to determine which key/value tokens to evict from the KV-cache. More recent methods refine the eviction criterion under the same fixed-budget setting: CAOTE (Goel et al., 2025) combines attention scores with value vectors to minimize the attention-output error induced by eviction, while KeyDiff (Park et al., 2026) avoids attention scores and instead retains geometrically distinctive keys based on key-space similarity. These strategies compress KV-caches to

a preset target (e.g., evicting 90% of tokens), while UA's adaptive pruning dynamically determines which key/value tokens are dispensable. As a result, UA compresses much further at long sequence length, in addition to improving performance more generally.

Data-dependent forget gates: A range of gate formulations have been added to a variety of models, such as LSTMs (Hochreiter & Schmidhuber, 1997) and transformers (Vaswani et al., 2017), for a variety of purposes. But from a cache compression perspective, many approaches devolve into soft-sliding-window formulations (Lin et al., 2025a;b; Hochreiter & Schmidhuber, 1997). More expressive mechanisms such as the delta rule (Yang et al., 2024b; 2025a;b) preclude the use of well-established elements of attention, such as SoftMax or RoPE embeddings (Su et al., 2024). UA combines multiple forget gates into an additive bias to the standard attention matrix, allowing us to leverage rich positional encoding via RoPE, targeted retrieval via SoftMax, and more sophisticated decay patterns than sliding-window (see Section 5.3), all at the same time.

3 Background

This section introduces the notation and necessary background relevant to UA's design.

Softmax multi-head self-attention: Standard softmax multi-head self-attention takes an input sequence of tokens $(\mathbf{q}_i)_{i=1}^L, (\mathbf{k}_i)_{i=1}^L, (\mathbf{v}_i)_{i=1}^L \in \mathbb{R}^{n \times d}$, where L, n & d are the sequence length, number of heads, and head dimension, respectively and computes:

$$\text{Softmax}(\mathbf{Q}\mathbf{K}^\top + \mathbf{M})\mathbf{V}, \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{L \times L}$ is the causal attention mask broadcasted across heads, $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times L \times d}$ are tensors formed by stacking $\mathbf{q}_i$, $\mathbf{k}_i$ & $\mathbf{v}_i$, respectively. During inference, new tokens need to attend to all previous tokens in $\mathbf{K}$ & $\mathbf{V}$, resulting in per-layer KV-caches that consume considerable memory.

Data dependent forget-gates: To reduce large KV-cache sizes, forget-gates decay past key/value tokens, deleting sufficiently decayed entries from the KV-cache. In particular, FoX (Lin et al., 2025a;b) expresses a forget-gate as an additive bias to the attention matrix. Using terminology from (Lin et al., 2025a), FoX computes a gate: $f_t = \sigma(\mathbf{x}_t \mathbf{w} + b) \in \mathbb{R}$ for each token in the input sequence and computes:

$$\text{Softmax}(\mathbf{Q}\mathbf{K}^\top + \mathbf{D})\mathbf{V}, \tag{2}$$

where $\mathbf{D}_{ij} = \sum_{l=j+1}^{i} \log(f_l)$. These decay-values decrease monotonically from the main-diagonal of the attention matrix, ensuring that cache entries with near-zero gates stay near-zero in perpetuity. Sufficiently decayed keys can thus be safely deleted from the KV-cache. However, because KV-cache entries decay in lockstep, this decay mechanism resembles a sliding-window, an inexpressive KV-cache eviction strategy. UA generalizes and incorporates several such decay mechanisms, to achieve more sophisticated behavior.

4 Universal Attention

We propose Universal Attention (UA), a novel framework and architecture that incorporates context-aware decay into softmax attention, inspired by modern linear attention methods and the structure of the attention matrix itself. The result is a model that learns adaptive and dynamic sparsity patterns balancing efficiency with expressivity.

Figure 1 depicts a UA layer, incorporating decay into softmax attention as a bias term. However, rather than relying on a single decay mechanism, UA combines multiple context-aware patterns via their geometric mean to form a unified decay rate value for each pair of positions. We choose the geometric mean over arithmetic because it preserves the veto property: if any single mechanism deems that a token should not decay, the combined decay is driven to zero.

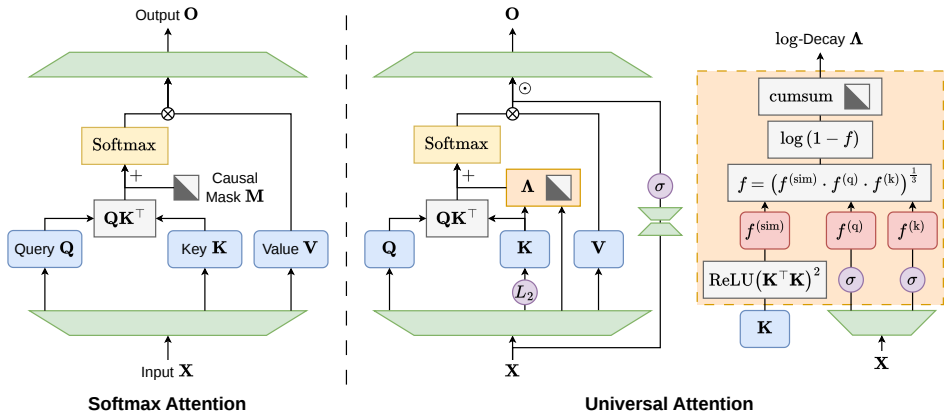

Figure 1: **Overview of Universal Attention.** (**Left**) Standard softmax attention with a fixed causal mask $\mathbf{M}$. (**Middle**) UA augments the attention logits $\mathbf{QK}^\top$ with a learned, content-aware decay bias $\mathbf{\Lambda}$, while applying L_2-normalization to keys, and a low-rank sigmoid gate to the attention output. (**Right**) Construction of $\mathbf{\Lambda}$ from three decay mechanisms: similarity-based f_{sim}, query-aware f_{q}, and key-aware f_{k}. These are combined via a geometric mean, transformed to log-space, and accumulated causally to produce the decay matrix.

Formally, consider the input to a UA layer as $\mathbf{X} \in \mathbb{R}^{L \times d_{model}}$, with queries, keys, and values $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times L \times d}$, where keys are L2-normalized. Universal Attention computes:

$$\text{Softmax}\left(\mathbf{QK}^\top + \mathbf{\Lambda}\right)\mathbf{V}, \quad \mathbf{\Lambda}_{ij} := \begin{cases} \sum_{k=j+1}^{i} \mathbf{A}_{kj} & i \geq j \\ -\infty & i < j, \end{cases} \tag{3}$$

where $\mathbf{\Lambda}_{ij}$ represents the accumulated log-decay from key j to query i: each column j of $\mathbf{A}$ tracks how a single key decays across subsequent positions, and $\mathbf{\Lambda}_{ij}$ sums this decay causally, with $-\infty$ enforcing causal masking through the softmax. We operate in log-space so that multiplicative decay composes additively. The matrix $\mathbf{A}$ aggregates the decay functions in $\mathcal{F}$ via their geometric mean:

$$\mathbf{A} = \log\left(\mathbf{1} - \left(\prod_{f \in \mathcal{F}} f(\mathbf{X})\right)^{\frac{1}{|\mathcal{F}|}}\right), \tag{4}$$

where $\prod$ denotes the elementwise product and each context-aware decay mechanism $f\colon \mathbb{R}^{L \times d_{model}} \to [0,1]^{L \times L}$ produces a pairwise decay matrix. Below, we introduce three representative mechanisms, linked to the first- and second-order statistics of stored keys.

4.1 Decay Components

Our representative decay mechanism family $\mathcal{F} := \left\{f^{(\text{q})}, f^{(\text{k})}, f^{(\text{sim})}\right\}$ includes (1) a Mamba-style query-aware forget gate $f^{(\text{q})}$, (2) a novel key-aware forget gate $f^{(\text{k})}$, and (3) a DeltaNet-style similarity decay $f^{(\text{sim})}$. The first two mechanisms capture first-order statistics aligned to the query and key axes of the attention matrix, respectively, while the third captures second-order interaction effects between embeddings at current and past time steps.

Query-aware forget gate $f^{(\text{q})}$: Similar to Mamba2 (Dao & Gu, 2024) and FoX (Lin et al., 2025a), each query position is assigned a scalar forget-gate computed as a sigmoid with a learned bias. Since this decay value depends only on the query position, it is broadcast along the key axis to produce the corresponding pairwise decay matrix $f^{(\text{q})}$:

$$g_i^{(\text{q})} := \sigma\left(\mathbf{x}_i \mathbf{w}^{(\text{q})} + b^{(\text{q})}\right), \quad f^{(\text{q})}(\mathbf{X})_{ij} := g_i^{(\text{q})}, \tag{5}$$

where $\mathbf{w}^{(\mathrm{q})}$ and $b^{(\mathrm{q})}$ are learned parameters. These gates accumulate causally, so that a key's importance degrades with the number of intermediate positions visited. This mechanism can be interpreted as each token advancing the entire cache by a variable-size time step.

Key-aware forget gate $f^{(\mathbf{k})}$: Symmetric to $f^{(\mathrm{q})}$, each *key* is assigned a scalar forget gate via the same sigmoid+bias parameterization, which then broadcasts over the *query* dimension:

$$g_j^{(\mathrm{k})} := \sigma\left(\mathbf{x}_j \mathbf{w}^{(\mathrm{k})} + b^{(\mathrm{k})}\right), \quad f^{(\mathrm{k})}(\mathbf{X})_{ij} := g_j^{(\mathrm{k})}. \tag{6}$$

While $f^{(\mathrm{q})}$ represents how much history to forget, our novel $f^{(\mathrm{k})}$ lets each key control its own intrinsic decay rate, independent of future updates. This allows the model to designate tokens that persist through time, or that self-prune quickly, on a case-by-case basis.

Similarity decay $f^{(\mathbf{sim})}$: To capture higher-order interaction effects between stored keys, we derive a pairwise decay mechanism similar to DeltaNet (Yang et al., 2024b). The DeltaNet update recurrence takes the form:

$$\mathbf{S}_t = \mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right) + \beta_t \mathbf{v}_t \mathbf{k}_t^\top \tag{7}$$

The state $\mathbf{S}_t$ accumulates key/value outer products with step size β_t, and so we can unroll the prior state $\mathbf{S}_{t-1} := \sum_{i=0}^{t-1} \mathbf{v}_i' \mathbf{k}_i'^\top$, where $\mathbf{v}_i', \mathbf{k}_i'$ represent $\mathbf{v}_i, \mathbf{k}_i$ with changes absorbed from subsequent state updates. DeltaNet sets $\mathbf{k}_i' := \mathbf{k}_i$ and absorbs all changes into $\mathbf{v}_i'$, but an equivalent formulation sets $\mathbf{v}_i' := \mathbf{v}_i$ and absorbs all changes into $\mathbf{k}_i'$:

$$\mathbf{S}_{t-1}\left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right) = \sum_{i=0}^{t-1} \mathbf{v}_i \mathbf{k}_i'^\top \left(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top\right) = \sum_{i=0}^{t-1} \mathbf{v}_i \left(\mathbf{k}_i' - \beta_t \mathbf{k}_i' \mathbf{k}_t^\top \mathbf{k}_t\right)^\top \tag{8}$$

For L_2-normalized $\mathbf{k}$, the right-hand term inside the summation represents the projection of $\mathbf{k}_i'$ onto $\mathbf{k}_t$, scaled by $\beta_t \in [0, 1]$. Thus we can interpret the DeltaNet update as a *partial orthogonalization* of all prior keys against the newest key entering the cache. Over successive rounds of orthogonalization, the magnitude of $\mathbf{k}_i'$ decays to zero, slowly when it is not aligned to each new $\mathbf{k}_t$, and quickly when it is.

Unfortunately, this decay mechanism is not UA-compatible as-is, because dynamically shifting $\mathbf{k}$ both 1) interferes with RoPE, and 2) requires overwriting the entire key cache at each time step, which is computationally prohibitive. We propose a simple modification that transforms DeltaNet decay into an attention-bias-friendly form:

$$\mathbf{k}_i' - \beta_t \mathbf{k}_i'^\top \mathbf{k}_t \mathbf{k}_t \quad \Rightarrow \quad \mathbf{k}_i' - \beta_t \mathbf{k}_i'^\top \mathbf{k}_t \mathbf{k}_i' \;=\; (1 - \beta_t \mathbf{k}_i'^\top \mathbf{k}_t)\mathbf{k}_i' \tag{9}$$

By replacing $\mathbf{k}_t$ with $\mathbf{k}_i'$ in the rightmost term, the update reduces to a multiplicative decay factor determined by key similarity. The decay factor is thus $\beta_t \mathbf{k}_i'^\top \mathbf{k}_t$, which serves as the basis for $f^{(\mathrm{sim})}$. We let $f^{(\mathrm{query})}$ absorb β_t and define $f^{(\mathrm{sim})}$ based solely on the dot product:

$$f^{(\mathrm{sim})}(\mathbf{X})_{ij} := \mathrm{ReLU}(\mathbf{k}_i^\top \mathbf{k}_j)^2 \tag{10}$$

Since keys are L_2-normalized, $\mathbf{k}_i^\top \mathbf{k}_j \in [-1, 1]$, and squared ReLU ensures a gate value in $[0, 1]$ as desired. The result is a conservative decay mechanism biased heavily toward zero, but still capable of eliminating a key/value pair when a new entry is perfectly aligned.

While we focus on these three representative mechanisms, the framework is extensible: additional decay functions can be incorporated into $\mathcal{F}$ as new advances emerge in linear attention — hence the *universal* nature of our method.

4.2 Layer Design and Algorithm

Plugging $\mathcal{F} = \{f^{(\mathrm{q})}, f^{(\mathrm{k})}, f^{(\mathrm{sim})}\}$ into the general framework 4, the decay matrix becomes:

$$\mathbf{A}_{ij} = \log\left(1 - \left(g_i^{(\mathrm{q})} \cdot g_j^{(\mathrm{k})} \cdot \mathrm{ReLU}(\mathbf{k}_i^\top \mathbf{k}_j)^2\right)^{\frac{1}{3}}\right). \tag{11}$$

The accumulated log-decay $\mathbf{\Lambda}_{ij} = \sum_{k=j+1}^{i} \mathbf{A}_{kj}$ can then be added to the pre-softmax logits of our attention layer, depicted in Fig. 1. We also take architectural inspiration from linear attention models, which often apply additional channel-wise sigmoid gates to the intermediate representation (Dao & Gu, 2024; Yang et al., 2025a; Poli et al., 2023). We adopt analogous gates for attention layers (Qiu et al., 2025), forming unique per-channel gates via low-rank projection (Zhang et al., 2025b). With rank equal to head width, these gates add a single attention head's worth of extra parameters per layer, a negligible quantity with outsize impact on performance. By reducing reliance on sink tokens (Qiu et al., 2025), channel gating allows the decay mechanisms greater freedom to learn expressive patterns. We ablate this component and compare to both gated and vanilla attention baselines in Section 5.1.

Inference pruning: Beyond shaping attention, UA's learned decay also provides a natural signal for inference-time KV-cache pruning. The log-decay matrix Λ is implemented as a bias on attention logits, but it can also be interpreted, through elementwise exponentiation, as a set of token-wise masks applied to the logits post-softmax:

$$\text{Softmax}\left(\mathbf{Q}\mathbf{K}^\top + \mathbf{\Lambda}\right)_{ij} = \frac{\mathrm{e}^{(\mathbf{Q}_i\mathbf{K}_j^\top + \mathbf{\Lambda}_{ij})}}{\sum_{k=0}^{j} \mathrm{e}^{(\mathbf{Q}_i\mathbf{K}_k^\top + \mathbf{\Lambda}_{ik})}} = \frac{\mathrm{e}^{\mathbf{Q}_i\mathbf{K}_j^\top} \exp(\mathbf{\Lambda})_{ij}}{\sum_{k=0}^{j} \mathrm{e}^{\mathbf{Q}_i\mathbf{K}_k^\top} \exp(\mathbf{\Lambda})_{ik}} \tag{12}$$

Mask values in $\exp(\mathbf{\Lambda})$ start at 1 and decay to 0 asymptotically, providing a natural pruning criterion: during inference, maintain a scalar bias λ_i for each key/value pair representing the corresponding entry of $\mathbf{\Lambda}$, continuously update λ_i using Eq. 11, and evict any key-value pair whose effective mask e^{λ_i} falls below a chosen threshold $p \in [0, 1]$. Decay bias updates in $\mathbf{A}$ are negative by construction, so λ_i always decreases, thus a token with $\lambda_i < \log p$ will not return to importance later. Although simple, this rule fully exploits the learned decay and lets the model itself decide, on a head-by-head basis, which and how many tokens should be retained. More sophisticated pruning algorithms are left to future work.

Implementation: We codesign UA to be *hardware-aligned* – interoperating with GQA (Ainslie et al., 2023) and FlashAttention (Dao et al., 2022; Dao, 2023; Shah et al., 2024; Zadouri et al., 2026), as current kernels do not support both GQA and float-valued biases. We implement UA as two kernels, separately computing (1) the decay-mask, and (2) the attention-output. Additionally, we apply Activation-Checkpointing (Chen et al., 2016) to mitigate storing the $\mathtt{O}(\mathtt{L}^2)$ sized decay-mask *per-layer* in device memory - which would otherwise be required to compute gradients in the backwards pass. Our custom implementation boosts end-to-end training throughput by 1.13× compared to PyTorch-2. We explain our kernel implementations and end-to-end pipeline in detail in Appendix C.

5 Evaluation

Setup: We evaluate Universal Attention in a 1B-parameter transformer following the Llama3 architecture (Grattafiori et al., 2024), which uses RoPE (Su et al., 2024) and Grouped Query Attention (Ainslie et al., 2023). Models are trained for 50B tokens on Dolma v1.7 data (Soldaini et al., 2024), reweighted per Chu et al. (2024). Model and training details and hyperparameters are provided in Appendix A. For long-context inputs, we apply YaRN (Peng et al., 2024) as a tuning-free adjustment to RoPE. We report results at two pruning thresholds $p = (.05, .15)$, to illustrate performance tradeoffs at conservative/aggressive levels of compression. For UA models, we additionally prune in chunks of 128 tokens during prefill, rather than running the whole input and only pruning afterwards.

Baselines: We compare against both linear- and softmax-attention baselines: Mamba2 (Dao & Gu, 2024), Mamba2-hybrid (NVIDIA et al., 2025), Gated DeltaNet (GDN) and its hybrid (Yang et al., 2025a), as well as standard Llama3 and gated-attention Llama3-G. All baselines are trained identically to UA.

For context extension, we apply YaRN (Peng et al., 2024) only to the Llama3-based models, where RoPE rescaling is a standard extension recipe. For Mamba2-hybrid, we follow Nemotron-H and keep its native RoPE-free design in the attention layers. For GDN and its hybrid, we retain the base RoPE setup in the main comparison to avoid introducing

an additional positional intervention. Appendix I further evaluates GDN-H with YaRN at 8K and 16K, where it improves long-context performance but does not change the overall comparison with UA.

To compare KV cache compression efficiency, we also apply H_2O (Zhang et al., 2023), SnapKV (Li et al., 2024), and their pyramid variants (Yang et al., 2024a) to the vanilla Llama3 model. We also include KeyDiff (Park et al., 2026) as an additional baseline using the publicly available kvpress (Devoto et al., 2025) implementation. To match the compression rate of UA, we aggressively set these methods to keep only 10% of the KV-cache. This strong suite of baselines thus encompasses a range of complementary recent approaches to reducing memory overhead, as well as two full-size oracles. Further details are provided in Appendix A.

Benchmarks: We evaluate the models' performance on two sets of benchmarks: information retrieval, including LongBench-E (Bai et al., 2024) and RULER (Hsieh et al., 2024), and common-sense reasoning tasks, following Yang et al. (2025a). For each task, we track the size of the stored states and KV-caches ("effective cache size") using their actual sizes in bytes.

5.1 Downstream Performance

Information retrieval: Table 1 reports results on inputs within the training sequence length, together with effective cache size, for RULER (4K) and LongBench-E (0-4K). On RULER, UA delivers significant improvement, even over unpruned Llama3 oracle models, claiming the highest and second-highest average scores at its two pruning thresholds. This improvement is mostly fueled by gains in tracking and retrieval tasks, though performance on aggregation suffers. Aggregation is a common bottleneck for KV-cache eviction methods, since it requires combining information from broadly distributed context. UA shares this weakness, but unlike SnapKV, H_2O, and KeyDiff, its substantial degradation is concentrated in aggregation. On LongBench-E, the best average is obtained by Mamba2-H, yet UA remains highly competitive under a much tighter cache budget: p=0.15 uses under half the memory of Mamba2-H, and exceeds both Llama3 and Llama3-G scores with only a fraction of their cache size. This favorable comparison to full-cache Llama3 also persists at 3B model scale, with results in Appendix F. Overall, UA offers superior memory-performance tradeoffs.

Table 1: RULER (4k sequence length) and LongBench-E (0-4k sequence length split).

	RULER (4k)						LongBench-E (0-4k)	
Model	**Retrieval**	**Tracking**	**Aggregation**	**QA**	**Avg.**	**Cache (MB)**	**Avg.**	**Cache (MB)**
Sub-quadratic models								
Mamba2	31.0	0.8	18.1	20.0	25.0	20	12.7	20.0
Mamba2-H	57.2	3.2	20.9	29.7	43.2	35.5	**18.9**	31.7
GDN	37.7	2.6	18.1	18.4	29.0	32	14.9	32.0
GDN-H	59.9	4.4	19.6	24.9	44.1	46.0	15.7	42.2
Quadratic models								
Llama3	54.2	8.4	13.6	**30.4**	40.8	143.7	16.4	113.3
+ SnapKV	21.7	13.4	7.5	25.6	19.5	14.4	12.9	11.3
+ Pyramid SnapKV	21.2	20.3	6.2	25.1	19.4	14.4	12.9	11.3
+ H_2O	4.2	0.2	10.7	19.9	7.3	14.4	13.0	11.3
+ Pyramid H_2O	4.3	0.2	9.7	20.0	7.2	14.4	12.6	11.3
+ KeyDiff ($K = 10\%L$)	33.8	24.7	10.2	24.3	28.0	14.4	15.2	11.3
Llama3-G	53.7	16.3	**21.4**	28.3	41.9	143.7	17.9	113.3
Universal Attention								
p=0.05	**68.4**	**36.4**	5.7	29.5	**50.3**	21.4	17.9	20.2
p=0.15	67.3	30.6	6.9	29.1	49.3	12.9	18.0	14.4

Common-sense reasoning: Table 2 presents language modeling and zero-shot reasoning results at a context length of 4k. Due to benchmark setup, H_2O, SnapKV, and KeyDiff only prune their caches *after* producing output logits from full-cache attention, and so are omitted from comparison. UA achieves competitive-to-superior perplexity and reasoning scores, with notable gains in LAMBADA accuracy, PIQA, and ARC-easy, despite aggressive pruning to only 38.7 KV-cache tokens on average. While the absolute zero-shot reasoning gains are modest, these results provide strong evidence that UA preserves model quality at

substantially lower cache cost: UA matches or slightly exceeds Llama3 in average reasoning accuracy (50.74/50.88 vs. 50.28) while reducing cache from 2.6 MB to 1.9/1.5 MB, and uses roughly 10–13× less cache than Mamba2 and 17–21× less than GDN while maintaining comparable or better accuracy. Note that due to short input sequence length, the Mamba2 and GDN linear attention architectures consume more memory than the corresponding full attention models, and UA compression levels are less dramatic in comparison, though easily still the lowest effective cache size overall. This demonstrates that the learned decay preserves short-context quality and improves the memory–quality tradeoff, while potentially acting as an implicit regularizer for perplexity and downstream task performance.

Table 2: Language modeling perplexity and zero-shot common-sense reasoning accuracy.

	Perplexity (↓)		Zero-Shot Common-Sense Reasoning (acc ↑)									
Model	**Wiki.**	**LMB.**	**LMB.**	**PIQA**	**Hella.**	**Wino.**	**ARC-e**	**ARC-c**	**SIQA**	**BoolQ**	**Avg.**	**Cache (MB)**
Mamba2	19.36	12.20	47.20	72.20	50.94	54.93	55.47	27.39	**41.25**	54.86	50.53	20
Mamba2-H	**18.54**	10.97	49.62	71.71	**51.49**	**55.41**	56.36	26.54	40.02	55.11	50.78	17.8
GDN	19.29	11.42	48.65	71.49	50.25	54.70	**57.32**	27.30	39.46	57.65	50.85	32
GDN-H	18.90	11.21	49.27	71.87	50.61	53.35	55.81	27.82	39.51	56.94	50.65	28.3
Llama3	19.40	11.95	47.89	71.71	49.65	53.99	54.42	25.68	39.61	**59.30**	50.28	2.6
Llama3-G	19.38	12.28	47.23	71.87	49.58	55.17	54.21	**27.82**	39.82	58.84	50.57	2.6
UA, p=0.05	18.67	**10.52**	50.92	72.14	50.89	53.51	56.82	25.68	39.76	56.18	50.74	1.9
UA, p=0.15	19.03	10.61	**51.12**	**72.36**	50.93	53.12	56.69	25.94	39.87	57.03	**50.88**	1.5

5.2 Context Length Extension

To better understand how the memory-performance trade-off evolves with context length, Figure 2 plots average score against effective cache size for RULER (left) and LongBench-E (right). Each curve traces a model across increasing sequence lengths from 1K to 16K, so both cache size and task difficulty vary along the curve. Consequently, a lower score at a larger cache size does not imply that additional cache is harmful, but simply reflects evaluation at a longer and more challenging context. Better configurations at a given context length lie toward the upper-left, and UA consistently traces the strongest frontier as sequence length increases.

The two UA variants expose a clear trade-off within the method itself: p=0.05 preserves higher scores, while p=0.15 achieves more aggressive compression. Unlike full-attention baselines such as Llama3 and Llama3-G, whose cache grows linearly with context length and quickly exceeds the plotting range, UA keeps cache usage under 25 MB even at the longest contexts shown. Relative to Llama3, UA with p=0.15 achieves 4.5×–45.3× cache reduction on RULER and 7.9×–26.9× on LongBench-E. The gap widens at longer contexts because UA's cache plateaus, while Llama3 continues to grow. By contrast, SnapKV and H_2O reduce cache at a larger accuracy cost, while hybrid baselines grow the cache with diminishing returns. The full breakdown of benchmark scores by task and sequence length is provided in Appendix I.

5.3 Empirical Analysis

We examine UA's actual learned patterns, uncovering several interesting and spontaneous behaviors. Fig. 3 (left) shows an example learned decay pattern, expressed as the set of accumulated masks $\exp(\mathbf{\Lambda})$. Attention logits are not pictured. We can isolate contributions from each of our three decay mechanisms: vertical stripes represent tokens with low key-linked decay $f^{(\mathrm{k})}$, horizontal lines represent selective filtering through similarity-based decay $f^{(\mathrm{sim})}$, and progressive query-linked decay $f^{(\mathrm{q})}$ causes entries to fade with distance from the diagonal. UA's mixture of mechanisms appears to be expressive.

Dynamic pruning is illustrated in Fig. 3 (middle), plotting average cache size across heads as a function of input length. While cache length is small relative to input length, the exact size varies, both over time and between different inputs. The model learns to be naturally parsimonious, accumulating information when it is needed, and pruning aggressively once it is not. This ability to discard unnecessary information could explain the consistent improvement of UA over the oracle Llama3 baselines in Section 5.1.

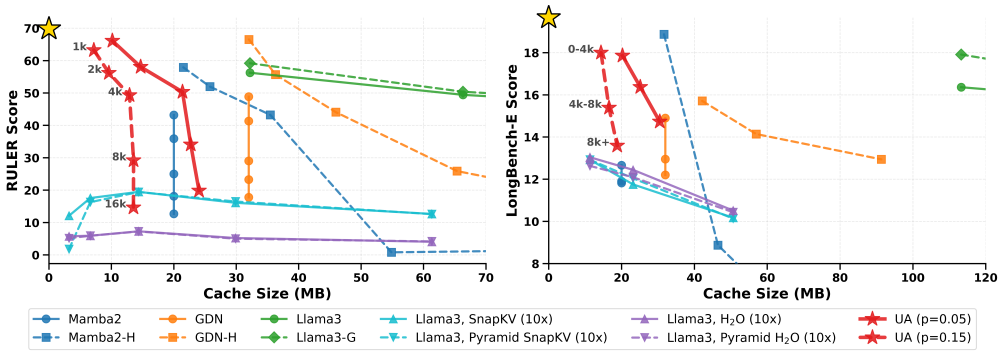

Figure 2: Score versus effective cache size trade-off on RULER (**left**) and LongBench-E (**right**). Each line traces a model across increasing sequence lengths from 1K to 16K; thus, moving along a curve changes both cache size and task difficulty. The gold star (★) denotes the ideal operating point of high score and low overhead.

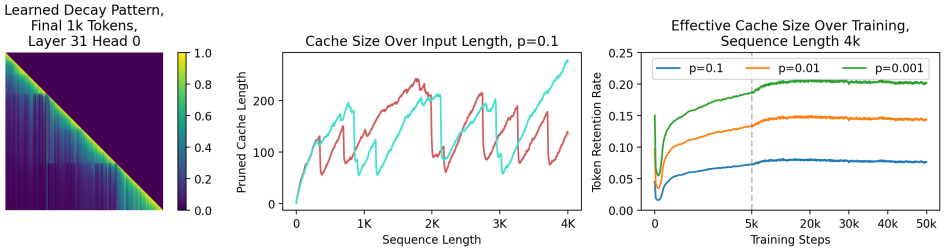

Figure 3: (**Left**) One example head's learned pattern of accumulated forget gates $exp(\mathbf{\Lambda})$, exhibiting behaviors from multiple decay mechanisms. (**Middle**) Progressive UA cache size over two different input sequences. Pruning is highly dynamic and input-adaptive, with spontaneously learned growth-and-reset behaviors. (**Right**) Evolution of effective cache size over the course of training, with an extreme initial dip and gradual recovery.

This notion of learned frugality is reinforced in Fig 3 (right), which plots the effective KV-cache size during training. Here we measure effective size as the percentage of cache entries above p at the final token position / in the final row of the attention matrix. Caches immediately sparsify at the beginning, then the model gradually re-incorporates past information as its language modeling capabilities improve. Eventually, cache sizes stabilize and even fall gently over the last half of training, with the model naturally becoming more efficient. This suggests that longer training regimes could produce further compression.

5.4 Ablations

UA decay mechanisms: We ablate UA's decay mechanisms using models trained on 10B tokens instead of 50B, and report averaged RULER `S1`, `S2`, & `S3` scores across sequence lengths. Table 3 (left) shows that all components contribute to performance, supporting our "universal" approach of combining multiple decay mechanisms. Interestingly, removing the most well-studied, Mamba-style query-aware decay $f^{(\mathrm{q})}$ causes the smallest degradation, while the other decay mechanisms provide substantial complementary benefits. Appendix E provides additional architectural ablations, including each decay mechanism in isolation, Qwen3-style normalized queries and keys, and comparisons with FoX and sliding-window attention.

Pruning threshold: We ablate different pruning thresholds p on our 50B-token model in Table 3 (right), evaluating LongBench-E with inputs truncated to 16K and perplexity on held-out natural-language validation data matching the training distribution. We report both the average compression ratio across KV heads and the maximum compression achieved

on a single sequence. UA degrades gracefully as compression increases: at p=0.15, it achieves 21.4× average compression on LongBench-E and 14.9× on held-out validation data, while changing the LongBench-E score and validation perplexity by less than one point each. Because the learned decay patterns vary across models and the compression–performance trade-off changes smoothly rather than exhibiting a natural cutoff, there is no single universally optimal threshold. We therefore use p=0.05 and p=0.15 as representative operating points for gentler and more aggressive pruning, respectively, while allowing the threshold to be selected according to the desired memory–quality trade-off.

Table 3: **(Left)**: Impact of UA components on downstream retrieval tasks. **(Right)**: Impact of pruning threshold, with performance/compression tradeoff.

	RULER S1, S2, S3 (% ↑)		
Method	**4k**	**8k**	**16k**
Universal Attention	**99.60**	**94.20**	**58.40**
– Query-aware $f^{(q)}$	–0.27	–5.67	–3.47
– Key-aware $f^{(k)}$	–23.20	–11.00	–36.20
– Similarity $f^{(sim)}$	–2.33	–34.27	–31.76
– RoPE & YaRN	–9.07	–11.20	–9.80
– Gated attention	–2.93	–20.07	–8.13

	LongBench-E (16k)			Held-Out Val. (4k)		
p	**Avg.**	**Avg. Comp.**	**Max Comp.**	**PPL**	**Avg. Comp.**	**Max Comp.**
0.00	14.05	1.00	1.00	10.48	1.00	1.00
0.05	**13.85**	**12.94**	**56.16**	**10.54**	**9.99**	**50.88**
0.10	13.71	17.01	107.9	10.61	12.50	73.14
0.15	**13.66**	**21.38**	**113.95**	**10.71**	**14.94**	**95.26**
0.20	13.04	26.32	144.24	10.86	17.20	122.27
0.25	12.24	31.95	174.07	11.09	19.61	151.70

6 Limitations

Aggregation tasks: Universal Attention is comparatively weaker on aggregation tasks, which often require combining weak evidence distributed across many positions rather than preserving a sparse set of individually salient tokens. This contrasts with UA's strengths on retrieval, tracking, and QA, suggesting a trade-off in the types of long-range information emphasized by different architectures. Our hybrid results in Appendix D further show that this weakness can be mitigated by combining UA with complementary memory mechanisms, making UA suitable as a cache-efficient attention component within broader long-context architectures when stronger aggregation capability is required.

Additional computation: UA introduces additional computation for constructing its learned decay and pruning masks. Although this enables adaptive, content-dependent retention, reducing the associated overhead remains an important systems optimization target.

Training from scratch: We evaluate UA as a novel architecture, but given that it generalizes standard softmax attention, a natural question is whether existing models can be adapted to UA, rather than having to pretrain it completely from scratch. Early results in Appendix G suggest that this is possible, but a significant gap remains compared to fully-pretrained UA. Closing this gap would greatly increase the practical feasibility of UA's deployment.

Inference optimization: While preliminary inference experiments in Appendix H demonstrate that UA's reduced cache can translate into lower decode latency, the current implementation does not include a fully fused, inference-optimized dynamic cache manager. Efficiently exploiting dynamically sparse KV caches requires specialized sparse kernels and systems support (Ye et al., 2025), and developing such an implementation remains an important direction for future work.

7 Conclusion

We present Universal Attention, a technique that generalizes existing decay mechanisms, introduces a novel decay mechanism, and combines these with RoPE and softmax attention into a single, unified layer. UA's expressive decay mechanism provides a natural adaptive pruning criterion, yielding superior memory-performance tradeoffs, improving downstream performance with lower memory overhead than even $O(1)$ linear attention models. Our results demonstrate that trainable self-pruning architectures provide a practical avenue to mitigate large memory overhead during inference.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Anthropic. Models overview, 2026. URL https://docs.anthropic.com/en/docs/about-claude/models. Claude API Docs. Accessed: 2026-03-31.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 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)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Assaf Ben-Kish, Itamar Zimerman, Shady Abu-Hussein, Nadav Cohen, Amir Globerson, Lior Wolf, and Raja Giryes. Decimamba: Exploring the length extrapolation potential of mamba. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=iWSl5Zyjjw.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *ArXiv*, abs/1604.06174, 2016. URL https://api.semanticscholar.org/CorpusID:15865278.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv e-prints*, art. arXiv:1904.10509, April 2019. doi: 10.48550/arXiv.1904.10509.

Linsong Chu, Divya Kumari, Tri Dao, Albert Gu, Raghu Ganti, Dakshi Agrawal, Mudhakar Srivatsa, Davis Wertheimer, Yu Chin Fabian Lim, Antoni Viros, Nelson Gonzalez, Tuan HoangTrong, Ofir Arviv, Yotam Perlitz, Michal Shmueli, Haochen Shen, Minjia Zhang, Gabe Goodhart, Naigang Wang, Nick Hill, Joshua Rosenkranz, Chi-Chun Liu, Adnan Hoque, Chih-Chieh Yang, Sukriti Sharma, Anh Uong, Jay Gala, Syed Zawad, and Ryan Gordon. Bamba: Inference-efficient hybrid mamba2 model. https://huggingface.co/blog/bamba#bamba-inference-efficient-hybrid-mamba2-model-%F0%9F%90%8D, December 2024. Accessed: 2025-05-06.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025. URL https://arxiv.org/abs/2510.00636.

Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning, 2020. URL https://arxiv.org/abs/2006.10901.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Raghavv Goel, Junyoung Park, Mukul Gagrani, Dalton Jones, Matthew Morse, Harper Langston, Mingu Lee, and Chris Lott. Caote: Kv cache selection for llms via attention output error-based token eviction, 2025. URL https://arxiv.org/abs/2504.14051.

Google DeepMind. Gemini 3, 2026. URL https://deepmind.google/models/gemini/. Accessed: 2026-03-31.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney

Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary

DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mOJgZWkXKW.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

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

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=q2Lnyegkr8.

Zhixuan Lin, Johan Obando-Ceron, Xu Owen He, and Aaron Courville. Adaptive computation pruning for the forgetting transformer. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=xNj14CY5S1.

Jie Liu, Huanzhi Pu, and Zhiru Zhang. Asyncsparse: Accelerating sparse matrix-matrix multiplication on asynchronous gpu architectures, 2026. URL https://arxiv.org/abs/2604.17834.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

NVIDIA, :, Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, Andrew Tao, Anna Shors, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Bobby Chen, Boris Ginsburg, Boxin Wang, Brandon Norick, Brian Butterfield, Bryan Catanzaro, Carlo del Mundo, Chengyu Dong, Christine Harvey, Christopher Parisien, Dan Su, Daniel Korzekwa, Danny Yin,

Daria Gitman, David Mosallanezhad, Deepak Narayanan, Denys Fridman, Dima Rekesh, Ding Ma, Dmytro Pykhtar, Dong Ahn, Duncan Riach, Dusan Stosic, Eileen Long, Elad Segal, Ellie Evans, Eric Chung, Erick Galinkin, Evelina Bakhturina, Ewa Dobrowolska, Fei Jia, Fuxiao Liu, Gargi Prasad, Gerald Shen, Guilin Liu, Guo Chen, Haifeng Qian, Helen Ngo, Hongbin Liu, Hui Li, Igor Gitman, Ilia Karmanov, Ivan Moshkov, Izik Golan, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jarno Seppanen, Jason Lu, Jason Sewall, Jiaqi Zeng, Jiaxuan You, Jimmy Zhang, Jing Zhang, Jining Huang, Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jon Barker, Jonathan Cohen, Joseph Jennings, Jupinder Parmar, Karan Sapra, Kari Briski, Kateryna Chumachenko, Katherine Luna, Keshav Santhanam, Kezhi Kong, Kirthi Sivamani, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Lawrence McAfee, Leon Derczynski, Lindsey Pavao, Luis Vega, Lukas Voegtle, Maciej Bala, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Markus Kliegl, Marta Stepniewska-Dziubinska, Matthieu Le, Matvei Novikov, Mehrzad Samadi, Michael Andersch, Michael Evans, Miguel Martinez, Mike Chrzanowski, Mike Ranzinger, Mikolaj Blaz, Misha Smelyanskiy, Mohamed Fawzy, Mohammad Shoeybi, Mostofa Patwary, Nayeon Lee, Nima Tajbakhsh, Ning Xu, Oleg Rybakov, Oleksii Kuchaiev, Olivier Delalleau, Osvald Nitski, Parth Chadha, Pasha Shamis, Paulius Micikevicius, Pavlo Molchanov, Peter Dykas, Philipp Fischer, Pierre-Yves Aquilanti, Piotr Bialecki, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Rabeeh Karimi, Rahul Kandu, Ran El-Yaniv, Raviraj Joshi, Roger Waleffe, Ruoxi Zhang, Sabrina Kavanaugh, Sahil Jain, Samuel Kriman, Sangkug Lym, Sanjeev Satheesh, Saurav Muralidharan, Sean Narenthiran, Selvaraj Anandaraj, Seonmyeong Bak, Sergey Kashirsky, Seungju Han, Shantanu Acharya, Shaona Ghosh, Sharath Turuvekere Sreenivas, Sharon Clay, Shelby Thomas, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shyamala Prayaga, Siddhartha Jain, Sirshak Das, Slawek Kierat, Somshubra Majumdar, Song Han, Soumye Singhal, Sriharsha Niverty, Stefania Alborghetti, Suseella Panguluri, Swetha Bhendigeri, Syeda Nahida Akter, Szymon Migacz, Tal Shiri, Terry Kong, Timo Roman, Tomer Ronen, Trisha Saar, Tugrul Konuk, Tuomas Rintamaki, Tyler Poon, Ushnish De, Vahid Noroozi, Varun Singh, Vijay Korthikanti, Vitaly Kurin, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenliang Dai, Wonmin Byeon, Xiaowei Ren, Yao Xu, Yejin Choi, Yian Zhang, Ying Lin, Yoshi Suhara, Zhiding Yu, Zhiqi Li, Zhiyu Li, Zhongbo Zhu, Zhuolin Yang, and Zijia Chen. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models, 2025. URL https://arxiv.org/abs/2504.03624.

OpenAI. Introducing gpt-5, August 2025. URL https://openai.com/index/introducing-gpt-5/. Accessed: 2026-03-31.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Junyoung Park, Dalton Jones, Matthew J Morse, Raghavv Goel, Mingu Lee, and Chris Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments, 2026. URL https://arxiv.org/abs/2504.15364.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: towards larger convolutional language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated

attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025. URL https://arxiv.org/abs/2505.06708.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Jingze Shi, Yifan Wu, Yiran Peng, Bingheng Wu, Liangdong Wang, Guang Liu, and Yuyu Luo. Trainable dynamic mask sparse attention, 2025. URL https://arxiv.org/abs/2508.02124.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2402.00159.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy

Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Jamba Team, Barak Lenz, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, Daniel Gissin, Daniel Jannai, Dor Muhlgay, Dor Zimberg, Edden M Gerber, Elad Dolev, Eran Krakovsky, Erez Safahi, Erez Schwartz, Gal Cohen, Gal Shachaf, Haim Rozenblum, Hofit Bata, Ido Blass, Inbal Magar, Itay Dalmedigos, Jhonathan Osin, Julie Fadlon, Maria Rozman, Matan Danos, Michael Gokhman, Mor Zusman, Naama Gidron, Nir Ratner, Noam Gat, Noam Rozen, Oded Fried, Ohad Leshno, Omer Antverg, Omri Abend, Opher Lieber, Or Dagan, Orit Cohavi, Raz Alon, Ro'i Belson, Roi Cohen, Rom Gilad, Roman Glozman, Shahar Lev, Shaked Meirom, Tal Delbari, Tal Ness, Tomer Asida, Tom Ben Gal, Tom Braude, Uriya Pumerantz, Yehoshua Cohen, Yonatan Belinkov, Yuval Globerson, Yuval Peleg Levy, and Yoav Shoham. Jamba-1.5: Hybrid transformer-mamba models at scale, 2024. URL https://arxiv.org/abs/2408.12570.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

xAI. Overview — xai docs, 2026. URL https://docs.x.ai/overview. Documents Grok 4.20 as the newest flagship model. Accessed: 2026-03-31.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3258–3270, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.195. URL https://aclanthology.org/2024.findings-acl.195/.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=y8Rm4VNRPH.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=r8H7xhYPwz.

Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. PaTH attention: Position encoding via accumulating

householder transformations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=ZBlHEeSvKd.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for llm inference serving, 2025. URL https://arxiv.org/abs/2501.01005.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling, 2026. URL https://arxiv.org/abs/2603.05451.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Hanzhi Zhang, Heng Fan, Kewei Sha, Yan Huang, and Yunhe Feng. DAM: Dynamic attention mask for long-context large language model inference acceleration. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4663–4676, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.242. URL https://aclanthology.org/2025.findings-acl.242/.

Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, Y. T. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

A Training, Model, and Evaluation Details

Training: All models are trained across 16 NVIDIA A100s in parallel. We employ a learning rate of $3e-4$, with warmup over 2k steps and cosine decay. Optimizer is AdamW with $\lambda = (.9, .95)$ and weight decay 0.1.

The models presented in Section 5.1, 5.2, and 5.3 are trained on 50B tokens, while the models for ablation in Section 5.4 are trained on 10B tokens. The training data is drawn from Dolma v1.7 (Soldaini et al., 2024) using the data-mixture reweighting adopted by Chu et al. (2024), with a context length of 4096 and 0.5M tokens per minibatch.

Model: We use the Llama-3 tokenizer (Grattafiori et al., 2024) (vocabulary size 128,256) for all models. Unless noted below, all models have 32 layers with $d_{\text{model}} = 1280$.

- Base architectures:
  - **Universal Attention / Llama-3:** n_{heads}=16, n_{kv}=4, d_{ff}=4096, $\theta_{\text{RoPE}}=5\times10^5$. Context extension uses YaRN with $(\beta_{\text{slow}}, \beta_{\text{fast}}) = (0.7, 35)$.
  - **Mamba2:** d_{state}=128, d_{conv}=4, d_{ff}=3072.
  - **Gated DeltaNet:** n_{heads}=4, d_{head}=256, d_{ff}=3072.

  For Mamba2 and Gated DeltaNet, d_{ff} is reduced from 4096 to keep total size ≈1B.
- Hybrid variants (GDN-H, Mamba2-H): In both hybrids, layers 4, 12, 20, and 28 are replaced with grouped-query attention (n_{heads}=16, n_{kv}=4). GDN-H retains $\theta_{\text{RoPE}}=5\times10^5$; Mamba2-H uses no positional encoding in its attention layers following NVIDIA et al. (2025).
- KV-cache compression baselines:
  - **SnapKV:** sparsity ratio r=0.1, `window_size` = 32, `kernel_size` = 7, max pooling.
  - **H_2O:** r_{heavy}=0.05, r_{recent}=0.05.

Evaluation: We utilize `EleutherAI\lm-evaluation-harness` (Gao et al., 2024) for evaluating on LongBench-E (Bai et al., 2024) and the common-sense reasoning benchmarks: WikiText (Merity et al., 2016), LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), Social IQa (Sap et al., 2019), and BoolQ (Wang et al., 2019). For RULER (Hsieh et al., 2024), we utilize the official GitHub repository[1], following the default `synthetic` task set.

B Compute State Size & KV-Cache Size

Linear attention models are known for their constant state size, which does not grow as the input sequence length grows. However, at short sequence length, linear attention can often have larger state than a softmax attention with similar number of parameters. Here we present the steps we used to compute the state size in our Mamba2 model and Gated DeltaNet model, as well as the KV-cache size of the GQA model.

Mamba2 (Dao & Gu, 2024): Following the official implementation of Mamba2[2], we take the following parameters:

- d_{model}
- expand, 2 by default,
- d_{state}, 128 by default.

[1] https://github.com/NVIDIA/RULER
[2] https://github.com/state-spaces/mamba

The state's shape is $(n_{\text{heads}}, d_{\text{state}}, d_{\text{head}})$, where the total number of elements is

$$\begin{aligned} s_{\text{Mamba}} &= n_{\text{heads}} \times d_{\text{state}} \times d_{\text{head}} \\ &= \frac{d_{\text{model}} \times \texttt{expand}}{d_{\text{head}}} \times d_{\text{state}} \times d_{\text{head}} \\ &= \texttt{expand} \times d_{\text{model}} \times d_{\text{state}}. \end{aligned}$$

Gated DeltaNet (Yang et al., 2025a): Following the official implementation listed in `flash-linear-attention`[3], we take the following parameters:

- d_{head}, 256 by default,
- `expand_v`, 2 by default,
- n_{heads}, 6 by default.

The state's shape is $(\texttt{num_heads}, \texttt{v_head_dim}, \texttt{head_dim})$, where the total number of elements is

$$\begin{aligned} s_{\text{GDN}} &= n_{\text{heads}} \times d_{\text{head, v}} \times d_{\text{head}} \\ &= \texttt{expand_v} \times n_{\text{heads}} \times d_{\text{head}}^2. \end{aligned}$$

Grouped-Query Attention (Ainslie et al., 2023): Each entry of the GQA's KV cache contains a key and a value, where both of them share the size of $n_{\text{heads, kv}} \times d_{\text{head}}$. Thus, the total size of the KV-cache after seeing a sequence of length l is

$$s_{\text{GQA}} = 2 \times n_{\text{heads, kv}} \times d_{\text{head}} \times l.$$

To compare the actual memory footprint, we can unify them using the actual size of the stored states. Thus, for a sequence of length l, a hybrid model consisting of n_{Mamba} Mamba2 layers, n_{GDN} Gated DeltaNet layers, and n_{GQA} grouped-query attention layers, has the following size of saved tensors:

$$n_{\text{Mamba}} \times s_{\text{Mamba}} + n_{\text{GDN}} \times s_{\text{GDN}} + n_{\text{GQA}} \times s_{\text{GQA}}.$$

Plugging-in the actual numbers from our setting as listed in Appendix A, we got the following formula for the size of the stored states and KV-caches for each of our model configurations, as listed in Table 4. Here we store all states and KV-caches in bf16.

Table 4: Size of the stored states and KV-caches, w.r.t. sequence length l.

Model	**Stored Size**
Mamba	20MB
Mamba-Hybrid	$l \times 5\text{KB} + 17.5\text{MB}$
Gated-DeltaNet	32MB
Gated-DeltaNet-Hybrid	$l \times 5\text{KB} + 28\text{MB}$
Llama3, Llama3-Gated	$l \times 40\text{KB}$
Llama3 with 10× KV-cache eviction	$l \times 4\text{KB}$

[3] https://github.com/fla-org/flash-linear-attention

C Kernel Design

We detail a memory-efficient Universal Attention layer in listing 1 alongside the triton kernels for both the forward and backwards passes in listings 2 & 3, respectively.

Decay: The decay mask kernels launch one kernel for both the forward and backwards pass. Thread-blocks tile over rows, computing the decay output (in the forward pass), or gradients (in the backwards pass) for a chunk of columns.

Memory-efficient attention: The memory-efficient attention kernel launches one kernel in the forward pass and two in the backwards pass. In the forward pass, the attention kernel launches 2D thread-blocks with a block computing one chunk (consisting of multiple rows) of the output tensor. In the backwards pass, we first launch a preprocessing kernel that computes `o` $\odot$ `dO` similar to FlashAttention. Then we launch the main backwards kernel that computes the gradients of the queries keys and values. The two functions `mem_efficient_bwd_dkv` and `mem_efficient_bwd_dq` are two subroutines of the main kernel and follow logic similar to FlashAttention-2. However, prior to computing softmax normalized values, the decay mask is applied to intermediate attention outputs to correctly compute normalized attention scores. If GQA is applied, our kernel mimics a broadcast on the fly by indexing the correct kv-head of the decay-mask.

```
class UniversalAttention:
  #b (batch), nkv (kv-heads)
  #l (seq length) d (head-dim)
  def forward(qgate,kgate,Q,K,V):
    decay = decay_fwd(qgate,kgate,K) # (b,nkv,l,l)
    o = mem_efficient_fwd(Q,K,V,decay)
    del decay; return o
  def backward(dO):
    decay = decay_fwd(qgate,kgate,K) # AC
    dq,dk,dv,ddecay = mem_efficient_bwd(Q,K,V,decay,dO)
    dqgate,dkgate,dkv2 = decay_bwd(ddecay)
    return dqgate,dkgate,dq,dk+dkv2,dv
```

Listing 1: Pseudocode for a memory-efficient UA layer. A collection of kernels compute the decay-mask and attention output, additively applying the decay mask to the attention matrix in a fused online softmax kernel, avoiding expensive I/O to HBM. Activation-Checkpointing additionally avoids saving the quadratic decay mask per-layer.

```
# Assume block-size [by, bx].
def decay_fwd(src, dest, K):
  b,n,f = tl.program_id(0), tl.program_id(1), tl.program_id(2)
  osum,srcsum,destsum = zeros(by,bx),zeros(by,bx),zeros(by,bx)
  for k in range(l//by+1):
    k, kt = K[b,n,k*by:k*(by+1),:], K[b,n,f*bx:k*(bx+1),:].T
    srcsum += src[b,n,k*bx:k*(bx+1)] # broadcast and add
    destsum += dest[b,n,k*by:k*(by+1)] # broadcast and add
    Γc = relu(dot(k,kt))**2 + srcsum + destsum # [by, bx]
    osum += log(1-Γc)
    decay[b,n,k*by:k*(by+1),k*bx:k*(bx+1)] = osum
  return decay
# Omit batch dimension for brevity.
def memory_efficient_fwd(Q,K,V,decay):
  i,n = tl.program_id(0), tl.program_id(1)
  li, mi = [0]*by, [-inf]*by
  qi = Q[n,i*by:(i+1)*by, :]
  oi = zeros((by, d))
  for j in range(l//bx+1):
    s = tl.matmul(qi, k[n%kvh,j*bx:(j+1)bx,:].T)+\
          decay[n%kvh,i*by:(i+1)by,j*bx:(j+1)bx] # On-the-fly broadcast
    mij = tl.maximum(mi,tl.max(s,1))
    s -= mij
    p = tl.exp(s)
    lij = tl.sum(s, axis=-1)
    alpha = tl.exp(mi-mij)
    li, mi = li*alpha+lij, mij
    oi = oi * alpha + \
          tl.matmul(p,v[j*bx:(j+1)bx, :])
  store(oi)
```

Listing 2: Pseudocode for the decay forward and memory efficient forward kernels illustrating the per thread-block work.

D Hybrid Model Results

We further evaluate UA in hybrid architectures that combine it with complementary sequence-modeling mechanisms. We consider two variants: **UA-Softmax**, which replaces one out of every eight UA layers with standard full attention, and **UA-Mamba**, which alternates UA and Mamba layers with an even split. All models follow the same training recipe as our main experiments. To isolate the architectural effects of hybridization, we evaluate these models without KV-cache pruning.

Table 5 shows that UA remains effective when incorporated into both hybrid architectures. UA-Softmax achieves performance close to pure UA, with RULER/LongBench-E averages of 48.8/17.6 compared with 48.5/18.1 for UA. UA-Mamba is slightly weaker overall at 47.9/15.9, but reduces the effective cache relative to the full-attention hybrid.

The hybrids also exhibit different capability trade-offs. In particular, introducing full-attention or Mamba layers substantially improves aggregation performance relative to pure UA, while reducing tracking performance. This suggests that UA can serve as a cache-efficient, high-resolution attention component within broader hybrid architectures, with full-attention or recurrent layers providing complementary memory behavior when stronger aggregation capability is desired.

Table 5: Hybrid UA model performance, compared against pure Llama3 and UA models, with no eviction.

	RULER (4k)						LongBench-E (0-4k)	
Model	**Retrieval**	**Tracking**	**Aggregation**	**QA**	**Avg.**	**Cache (MB)**	**Avg.**	**Cache (MB)**
UA-Mamba	63.7	29.8	15.2	28.7	47.9	81.9	15.9	66.7
UA-Softmax	64.3	33.5	11.7	31.5	48.8	143.7	17.6	113.3
Llama3	54.2	8.4	13.6	30.4	40.8	143.7	16.4	113.3
UA, p=0	65.5	37.0	5.1	29.3	48.5	143.7	18.1	113.3

```
def decay_bwd(ddecay, src, dest, K):
  b,n,f = tl.program_id(0), tl.program_id(1), tl.program_id(2)
  dki, dsrci, suff = zeros(by,d), zeros(by), 0
  for j in rev_range(l//bx+1):
    ki = K[b,n,f*by:(f+1)*by,:]
    kj = K[b,n,j*bx:(j+1)*bx,:]
    Γ = tl.dot(ki*sqrt(src), (kj*sqrt(dest)).T)
    Γc = relu(Γ)**(2/3)
    dd = rev_cumsum(ddecay[b,n,f*by:(f+1)*by,j*bx:(j+1)*bx]) + suff
    suff += tl.sum(dd, -1)
    dd = where(Γ>0, -dd/(1-Γc) * (2/3) * relu(Γ)**(-1/3), 0)
    dki += tl.dot(dd, kj*sqrt(dest))
    dsrci += tl.sum(dd*Γ, -1) / (2*src)
  dk_i[b,n,f*by:(f+1)*by,:] = dki * sqrt(src)
  dsrc[b,n,f*by:(f+1)*by] = dsrci
  # Symmetric col-kernel for dk_j, ddest
  return dk_i+dk_j, dsrc, ddest
# Preprocess: Di = rowsum(oi ⊙ doi)
# --- dK, dV kernel (key-parallel) ---
def mem_efficient_bwd_dkv(Q,K,V,decay,dO):
  j,n = tl.program_id(0), tl.program_id(1)
  kj = K[n%kvh,j*bx:(j+1)*bx,:]
  vj = V[n%kvh,j*bx:(j+1)*bx,:]
  dkj, dvj = zeros(bx,d), zeros(bx,d)
  for i in range(l//by+1):
    qi = Q[n,i*by:(i+1)*by,:]
    aff = decay[n%kvh,i*by:(i+1)*by,j*bx:(j+1)*bx]
    s = tl.dot(kj, qi.T) + aff.T
    p = tl.exp(s - M[n,i*by:(i+1)*by])
    dvj += tl.dot(p, dO[n,i*by:(i+1)*by,:])
    dp = tl.dot(vj, dO[n,i*by:(i+1)*by,:].T)
    ds = p * (dp - D[n,i*by:(i+1)*by])
    dkj += tl.dot(ds, qi)
  DK[n%kvh,j*bx:(j+1)*bx,:] = dkj
  DV[n%kvh,j*bx:(j+1)*bx,:] = dvj
# --- dQ, ddecay kernel (query-parallel) ---
def mem_efficient_bwd_dq(Q,K,V,decay,dO):
  i,n = tl.program_id(0), tl.program_id(1)
  qi = Q[n,i*by:(i+1)*by,:]
  doi = dO[n,i*by:(i+1)*by,:]
  mi = M[n,i*by:(i+1)*by]
  dqi = zeros(by, d)
  for j in range(l//bx+1):
    kj = K[n%kvh,j*bx:(j+1)*bx,:]
    vj = V[n%kvh,j*bx:(j+1)*bx,:]
    aff = decay[n%kvh,i*by:(i+1)*by,j*bx:(j+1)*bx]
    s = tl.dot(qi, kj.T) + aff
    p = tl.exp(s - mi)
    dp = tl.dot(doi, vj.T)
    ds = p * (dp - D[n,i*by:(i+1)*by])
    ddecay[n%kvh,i*by:(i+1)*by,j*bx:(j+1)*bx] = ds
    dqi += tl.dot(ds, kj)
  DQ[n,i*by:(i+1)*by,:] = dqi
```

Listing 3: Pseudocode for the decay backward and memory-efficient decay backward kernels illustrating the per thread-block work.

E Additional Ablation Results

We further evaluate the robustness of UA to alternative architectural choices and compare against individual decay mechanisms and sliding-window attention. Table 6 reports averaged RULER S1, S2, and S3 scores across context lengths for 1B models trained on 10B tokens.

UA remains effective when adopting Qwen3-style normalized queries and keys, with slightly improved performance at 16K. We also evaluate each decay mechanism in isolation. In particular, using only the query-aware decay $f^{(q)}$ corresponds to the FoX-style design. While this variant performs reasonably at 4K, its performance degrades substantially with

context length, reaching 40.0 at 16K compared with 58.4 for full UA. The key-aware and similarity-aware mechanisms likewise underperform the complete model when used alone, indicating that the three decay signals provide complementary retention behavior. Sliding-window attention is competitive only at the shortest context length and degrades sharply as context increases. Overall, these results show that UA is robust to alternative query/key normalization choices and that combining multiple decay mechanisms is important for robust long-context performance.

Table 6: Additional ablations on model design choices. 1B models, trained on 10B tokens.

	RULER S1, S2, S3 (% ↑)		
Method	**4k**	**8k**	**16k**
Universal Attention	99.6	94.2	58.4
Normalized QK (Qwen3-style)	100.0	93.8	60.2
Only query-aware $f^{(\mathrm{q})}$ (FoX)	90.4	68.7	40.0
Only key-aware $f^{(\mathrm{k})}$	96.8	65.3	54.8
Only similarity $f^{(\mathrm{sim})}$	82.3	85.4	56.3
Sliding-window attention ($w = 512$)	100.0	68.2	31.5

F 3B Model Results

To evaluate whether UA's benefits persist at larger model scale, we train 3B-parameter Llama3-style softmax and UA models using the same 50B-token training recipe as in our main experiments. Table 7 reports RULER and LongBench-E results together with effective cache size.

At 3B scale, UA continues to improve both model quality and cache efficiency. On RULER at 4K context, UA achieves an average score of 52.6 compared with 48.8 for Llama3-3B, while p=0.05 reduces the effective cache from 341.8 MB to 46.6 MB, corresponding to a 7.3× reduction. On LongBench-E (0–4K), UA reaches 23.5 compared with 21.5 for Llama3-3B while reducing cache from 271.9 MB to 41.5 MB, a 6.5× reduction. These results indicate that UA's quality–memory advantages extend beyond the 1B setting.

Table 7: Llama3-3B and UA-3B model performance, trained on 50B tokens.

	RULER (4k)						LongBench-E (0-4k)	
Model	**Retrieval**	**Tracking**	**Aggregation**	**QA**	**Avg.**	**Cache (MB)**	**Avg.**	**Cache (MB)**
Llama3-3B	64.7	22.8	16.6	30.1	48.8	341.8	21.5	271.9
UA-3B, p=0	69.8	40.3	7.2	35.4	52.6	341.8	23.6	271.9
UA-3B, p=0.05	70.3	38.7	6.8	34.4	52.6	46.6	23.5	41.5

G Post-Training Adaptation

Although UA is designed as a distinct attention architecture and is primarily evaluated through pretraining from scratch, we also investigate whether it can be introduced into an existing pretrained Transformer checkpoint through continued training.

Starting from our pretrained Llama3 model, we convert its attention layers to UA while retaining the pretrained Llama3 weights, randomly initialize the UA-specific parameters—including the low-rank attention-gate projections and decay parameters—and continue training for an additional 10B tokens. Table 8 reports the resulting RULER and LongBench-E performance together with effective cache size.

Post-training adaptation substantially improves over the original Llama3 baseline while enabling significant cache compression. At p=0.05, Llama3-UA improves the RULER

Table 8: Post-training adaptation of a pretrained Llama3-1B checkpoint to UA.

	RULER (4k)						LongBench-E (0-4k)	
Model	**Retrieval**	**Tracking**	**Aggregation**	**QA**	**Avg.**	**Cache (MB)**	**Avg.**	**Cache (MB)**
Llama3	54.2	8.4	13.6	30.4	40.8	143.7	16.4	113.3
Llama3-UA, p=0	58.7	16.2	10.8	32.4	44.0	143.7	17.1	113.3
Llama3-UA, p=0.05	59.3	16.2	9.7	33.0	44.3	38.6	17.2	41.3
UA, p=0	65.5	37.0	5.1	29.3	48.5	143.7	18.1	113.3
UA, p=0.05	68.4	36.4	5.7	29.5	50.3	21.4	17.9	20.2

average from 40.8 to 44.3 while reducing the cache from 142.4 MB to 38.6 MB; on LongBench-E, it improves from 16.4 to 17.2 while reducing the cache from 113.3 MB to 41.3 MB. The adapted model remains below UA trained from scratch, which reaches a RULER average of 50.3 with a 21.4 MB cache at the same pruning threshold. These results suggest that UA can be incorporated into existing checkpoints through continued training, although training from scratch currently yields the strongest performance–memory trade-off.

H Latency Analysis

We evaluate the end-to-end inference-time efficiency of UA. An inference-optimized dynamic cache manager is not implemented in the current system, as efficiently operating on dynamically pruned KV caches requires sparse matrix–vector computation and specialized sparse kernels, which remain active areas of systems research (Liu et al., 2026; Gale et al., 2020). Nevertheless, the current implementation already provides measurable benefits during training and decoding.

For prefill, our custom UA kernels improve end-to-end training throughput by approximately 1.13×; since prefill exercises the same forward attention path, this suggests that kernel optimization can also reduce TTFT, although we do not report a matched inference-prefill baseline here. For decoding, we evaluate a naive UA inference implementation with and without KV-cache pruning. As shown in Table 9, pruning yields approximately 1.1×–1.4× per-token decode speedup (11–29% lower latency), with larger benefits at longer prompt lengths.

Table 9: Prefill latency, per-step decode latency with and without KV-cache pruning, and post-eviction cache size at different sequence lengths.

Sequence Length	**Prefill Latency**	**Decode Latency per Step**		**Cache Size after eviction (MB)**
		w. full cache	**w. evicted cache**	
3,678	438 ms	1.1 ms	978 µs	21.5
7,875	1.76 s	1.2 ms	998 µs	24.4
15,678	5.44 s	1.4 ms	997 µs	24.3

I Detailed RULER results for Figure 2

We report the complete RULER results underlying Figure 2, including per-task scores for all models at context lengths from 1K to 16K and their corresponding effective cache sizes. In addition to the configurations shown in the main comparison, we include GDN-H with YaRN at 8K and 16K for a consistent long-context comparison, as well as full-cache UA (p=0) to separate the effects of the UA architecture from those of cache pruning. Applying YaRN substantially improves GDN-H at long context, particularly on retrieval, but UA with p=0.05 still achieves a better overall RULER score with a smaller effective cache at both 8K and 16K.

Table 10: Detailed RULER (1k) scores, with effective cache size.

Model	S1	S2	S3	M1	M2	M3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.	Cache (MB)
Mamba2	100.0	100.0	100.0	47.2	4.4	4.2	53.0	54.5	1.8	3.6	43.1	44.0	5.8	43.2	20.0
Mamba2-H	100.0	100.0	96.8	87.0	68.0	37.6	72.4	73.0	5.4	20.0	43.1	44.4	5.0	57.9	21.5
GDN	100.0	100.0	99.8	38.6	6.4	1.4	80.1	81.0	5.4	54.3	17.7	43.8	6.8	48.9	32.0
GDN-H	100.0	100.0	99.2	94.0	97.4	43.2	83.0	74.8	38.5	56.7	21.9	49.0	7.0	66.5	32.0
Llama3	100.0	100.0	96.6	89.6	41.2	23.8	83.6	80.5	8.3	17.4	35.1	49.4	5.8	56.2	32.2
+ SnapKV	95.2	7.4	0.0	3.2	0.8	0.0	0.0	0.0	0.1	0.3	39.7	5.4	5.0	12.1	3.2
+ Pyramid SnapKV	7.6	0.4	0.0	0.8	0.0	0.0	0.0	0.0	0.0	0.1	5.3	5.2	3.8	1.8	3.2
+ H_2O	0.0	0.0	0.0	2.2	0.0	0.0	0.2	0.0	0.0	2.6	41.9	21.8	6.0	5.7	3.2
+ Pyramid H_2O	0.0	0.0	0.0	2.2	0.0	0.0	0.2	0.0	0.0	2.5	38.5	18.4	5.0	5.1	3.2
Llama3-G	100.0	100.0	99.8	82.2	51.0	31.8	83.4	80.3	25.3	13.3	47.3	49.6	5.2	59.2	32.2
UA, p=0	100.0	100.0	100.0	93.6	90.4	74.6	90.6	80.2	53.0	11.6	22.3	48.8	5.2	66.9	32.2
UA, p=0.05	100.0	100.0	99.8	89.0	92.2	71.2	87.0	80.4	53.0	11.6	23.3	46.2	5.6	66.1	10.2
UA, p=0.15	100.0	100.0	100.0	83.2	86.0	63.2	79.5	74.6	53.8	11.6	17.5	47.2	5.4	63.2	7.2

Table 11: Detailed RULER (2k) scores, with effective cache size.

Model	S1	S2	S3	M1	M2	M3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.	Cache (MB)
Mamba2	100.0	96.0	98.6	32.8	1.2	0.0	29.8	29.9	0.9	1.3	30.7	22.2	23.0	35.9	20.0
Mamba2-H	100.0	100.0	91.6	84.4	38.2	27.4	64.5	63.3	2.1	6.9	39.3	36.8	21.0	52.0	25.8
GDN	100.0	100.0	98.8	34.6	2.0	0.2	58.2	65.7	4.5	9.8	22.1	17.6	23.6	41.3	32.0
GDN-H	100.0	100.0	99.8	92.2	85.8	31.8	70.6	66.5	6.9	5.4	10.3	26.8	27.4	55.7	36.3
Llama3	100.0	100.0	94.0	82.2	16.4	9.6	72.9	69.5	3.1	4.7	30.6	35.2	24.4	49.4	66.3
+ SnapKV	100.0	29.0	0.0	18.8	0.6	0.0	0.2	0.1	6.1	1.9	27.1	24.0	20.4	17.6	6.6
+ Pyramid SnapKV	100.0	28.8	0.0	16.0	0.4	0.0	0.0	0.0	12.1	0.6	19.3	15.4	19.0	16.3	6.6
+ H_2O	2.4	2.4	2.4	3.0	2.4	0.0	0.5	0.5	0.0	2.7	24.0	15.6	21.0	5.9	6.6
+ Pyramid H_2O	2.4	2.4	2.4	3.0	2.2	0.0	0.7	0.9	0.1	2.9	23.5	15.6	20.6	5.9	6.6
Llama3-G	100.0	100.0	99.8	73.8	19.6	14.0	60.6	64.0	12.9	5.3	44.1	34.8	26.0	50.4	66.3
UA, p=0	100.0	100.0	99.8	63.6	72.6	68.5	80.2	63.5	36.1	5.7	13.2	37.8	28.4	59.2	66.3
UA, p=0.05	100.0	100.0	99.4	62.4	68.0	63.2	77.8	67.6	38.1	6.3	7.6	37.4	27.0	58.1	14.7
UA, p=0.15	100.0	100.0	100.0	60.4	60.8	55.0	74.9	65.7	41.8	5.5	2.3	37.8	26.4	56.2	9.6

Table 12: Detailed RULER (4k) scores, with effective cache size.

Model	S1	S2	S3	M1	M2	M3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.	Cache (MB)
Mamba2	100.0	34.4	49.2	24.4	0.2	0.0	20.5	19.3	0.8	3.0	33.3	22.2	17.8	25.0	20.0
Mamba2-H	100.0	100.0	60.4	69.6	2.4	9.2	58.8	57.3	3.2	18.4	23.3	34.2	25.2	43.2	35.5
GDN	100.0	65.2	56.2	29.2	0.2	0.0	25.7	25.3	2.6	14.4	21.9	17.4	19.4	29.0	32.0
GDN-H	100.0	100.0	91.2	74.8	25.0	17.6	34.8	36.0	4.4	36.3	2.9	27.0	22.8	44.1	46.0
Llama3	100.0	99.8	87.2	66.8	4.8	6.4	31.0	37.5	8.4	7.3	19.9	34.6	26.2	40.8	143.7
+ SnapKV	99.8	19.4	0.0	38.8	0.4	0.0	10.0	5.0	13.4	1.1	13.9	29.4	21.8	19.5	14.4
+ Pyramid SnapKV	100.0	23.2	0.2	33.2	0.2	0.0	6.3	6.3	20.3	0.6	11.9	28.0	22.2	19.4	14.4
+ H_2O	4.6	6.2	5.6	7.2	1.2	0.0	5.3	3.1	0.2	4.8	16.5	18.6	21.2	7.3	14.4
+ Pyramid H_2O	4.6	6.2	5.6	7.2	1.2	0.0	5.9	3.7	0.2	4.0	15.3	18.2	21.8	7.2	14.4
Llama3-G	100.0	100.0	97.6	61.2	2.4	7.2	26.5	34.7	16.3	9.8	33.0	33.4	23.2	41.9	143.7
UA, p=0	100.0	100.0	82.8	64.4	77.6	54.0	30.0	15.4	37.0	10.0	0.2	35.8	22.8	48.5	143.7
UA, p=0.05	100.0	100.0	86.2	73.6	73.8	57.0	34.7	22.2	36.4	11.4	0.1	36.2	22.8	50.3	21.4
UA, p=0.15	100.0	100.0	86.6	80.2	66.6	46.0	33.9	25.2	30.6	13.8	0.0	36.6	21.6	49.3	12.9

Table 13: Detailed RULER (8k) scores, with effective cache size.

Model	S1	S2	S3	M1	M2	M3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.	Cache (MB)
Mamba2	100.0	24.8	24.0	12.8	0.0	0.0	7.4	6.6	1.6	2.2	29.8	11.2	15.2	18.1	20.0
Mamba2-H	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	3.5	3.6	1.8	0.8	54.9
GDN	100.0	47.4	30.8	27.0	0.0	0.0	23.3	23.0	2.7	1.3	21.7	9.8	15.2	23.2	32.0
GDN-H	56.6	52.0	36.8	45.2	44.8	1.6	24.9	24.6	17.2	1.2	0.1	12.2	19.2	25.9	65.4
+ YaRN	95.4	72.4	40.0	30.2	44.8	1.0	28.7	14.8	15.3	1.0	0.0	16.6	18.5	29.1	65.4
Llama3	99.6	95.0	73.0	51.2	2.2	1.2	24.7	27.6	2.1	5.2	13.7	15.6	18.4	33.0	299.0
+ SnapKV	99.6	16.2	1.2	31.6	1.4	0.0	10.6	4.0	3.7	1.4	7.4	12.8	19.2	16.1	29.9
+ Pyramid SnapKV	99.8	21.2	1.4	29.8	0.8	0.0	12.3	5.5	3.4	0.8	6.4	12.8	20.0	16.5	29.9
+ H_2O	2.8	3.8	4.8	5.4	0.4	0.0	2.9	2.7	0.0	3.9	10.0	12.4	18.6	5.2	29.9
+ Pyramid H_2O	2.8	3.8	4.8	5.4	0.4	0.0	3.4	3.1	0.0	3.3	9.0	10.4	18.4	5.0	29.9
Llama3-G	96.8	86.6	77.6	46.0	1.2	1.4	17.2	22.9	4.1	4.8	27.9	14.0	18.8	32.2	299.0
UA, p=0	99.6	78.0	65.2	49.0	42.8	14.6	20.2	7.5	32.2	6.1	0.0	13.0	20.0	34.5	299.0
UA, p=0.05	99.4	75.2	64.2	46.8	47.2	12.2	19.6	9.0	32.6	6.0	0.0	12.0	19.0	34.1	22.7
UA, p=0.15	97.8	53.0	48.2	40.4	33.8	9.2	24.2	11.1	32.7	2.4	0.0	10.4	16.4	29.2	13.6

Table 14: Detailed RULER (16k) scores, with effective cache size.

Model	S1	S2	S3	M1	M2	M3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.	Cache (MB)
Mamba2	96.0	5.0	9.0	6.2	0.0	0.0	6.6	6.0	0.4	0.6	15.9	8.2	11.0	12.7	20.0
Mamba2-H	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	18.7	1.8	1.6	1.7	94.2
GDN	99.6	22.8	12.4	18.4	0.0	0.0	17.9	16.6	5.9	0.0	17.5	7.0	13.0	17.8	32.0
GDN-H	28.4	20.2	14.6	20.6	8.0	0.8	11.5	9.6	0.0	0.1	0.2	10.6	16.0	10.8	104.7
+ YaRN	92.6	27.2	25.4	18.9	9.2	1.0	15.4	6.2	0.0	0.4	0.0	14.3	16.4	17.5	104.7
Llama3	99.4	68.6	35.8	26.4	0.2	0.0	15.3	7.1	1.6	3.8	11.3	10.4	17.0	22.8	613.3
+ SnapKV	99.8	6.0	0.2	10.8	0.4	0.0	5.3	1.1	2.6	2.2	7.0	11.6	17.4	12.6	61.3
+ Pyramid SnapKV	99.8	5.6	0.2	11.8	0.2	0.0	5.3	1.1	2.2	2.5	6.4	11.2	17.4	12.6	61.3
+ H_2O	4.2	4.2	4.2	4.2	0.0	0.0	0.8	0.4	0.3	1.2	7.1	8.6	17.4	4.0	61.3
+ Pyramid H_2O	4.2	4.2	4.2	4.6	0.0	0.0	1.4	0.6	0.3	0.9	6.5	9.6	17.4	4.1	61.3
Llama3-G	99.4	18.0	21.0	13.0	0.0	0.2	5.3	2.1	1.6	3.2	12.7	9.6	14.6	15.4	613.3
UA, p=0	100.0	36.2	36.4	26.4	6.4	1.0	16.1	5.7	30.5	1.3	0.0	9.8	12.2	21.7	613.3
UA, p=0.05	98.0	25.8	32.8	23.4	6.8	1.0	12.5	4.9	31.0	0.2	0.0	8.2	13.6	19.9	24.0
UA, p=0.15	87.4	6.8	13.0	11.4	3.6	1.0	8.5	4.3	35.0	0.0	0.0	7.4	12.2	14.7	13.5