KVPOP — RETROFITTING LLMS WITH XLSTM-GUIDED TOKEN EVICTION

Lukas Hauzenberger[1,2] **Niklas Schmidinger**[1,2] **Anamaria-Roberta Hartl**[2] **David Stap**[1]
Thomas Schmied[2] **Sebastian Böck**[1] **Günter Klambauer**[1,2] **Sepp Hochreiter**[1,2]

[1]NXAI [2]Johannes Kepler University Linz, Austria

ABSTRACT

Key-value (KV) cache growth is a major bottleneck in autoregressive decoding, as memory and bandwidth scale linearly with the context length. Existing KV-eviction methods often rely on static heuristics or early retention decisions, which miss downstream context and cause brittle eviction as token relevance shifts. To address this, we introduce KVPOP, which uses stateful delayed importance scoring at eviction time for context-aware cache management under a fixed per-head budget. KVPOP is trained with a novel future-attention importance target that estimates long-term token utility *without* materializing dense attention. We show that KVPOP preserves dense attention performance on challenging mathematical reasoning tasks, while reducing KV cache size by 75%. Even at ∼94% KV compression, KVPOP still retains ∼80% of performance, almost doubling baseline recovery. Our results indicate that timing-aware eviction cuts KV memory costs while maintaining quality.

1 INTRODUCTION & BACKGROUND

In Transformers, each generated token produces a query vector that attends to all previous keys via dot-product attention Vaswani et al. (2017). To avoid recomputing attention scores for every new token during autoregressive generation, Transformer-based-large language models (LLMs) cache the key and value representations of past tokens in a key–value (KV) cache. While caching enables efficient token-by-token generation, the KV cache grows linearly with sequence length and quickly becomes the dominant factor in accelerator memory usage and memory bandwidth, limiting both throughput and the maximum context length that can be served in practice (Kwon et al., 2023).

Prior KV-eviction methods address this bottleneck by compressing or replacing dense dot-product attention with sparse mechanisms in the KV cache. Training-free methods retain recent tokens and a small set of globally important tokens (Zhang et al., 2023), preserve attention sinks (Xiao et al., 2023), or select subsets of tokens using query-aware or similarity-based heuristics (Li et al.; Tang et al., 2024). While effective, such approaches often rely on proxy signals that adapt too slowly as token relevance shifts during generation, and several query-aware methods reduce attention computation without enforcing a strictly bounded persistent cache. Learned approaches retrofit lightweight modules to predict eviction decisions or compress memory online (Nawrot et al., 2024; Zeng et al., 2025; Akhauri et al., 2025; Łańcucki et al., 2025), suggesting token importance is predictable but hard to estimate under tight memory budgets.

To address limitations of existing KV cache management methods, particularly *premature eviction* driven by retrospective or greedy importance signals, we introduce KVPOP , a sparse-attention retrofit that enforces a strict per-head KV budget. KVPOP augments each attention head with a lightweight XLSTM-based scorer (Beck et al., 2024) and maintains a constant-size cache of B tokens at inference time. KVPOP recovers on average 93% of dense attention performance on challenging mathematical reasoning tasks while reducing the KV cache by 75%. Even at 94% compression, it retains 80% of dense performance—nearly twice the recovery of a trained StreamingLLM (Xiao et al., 2024) baseline.

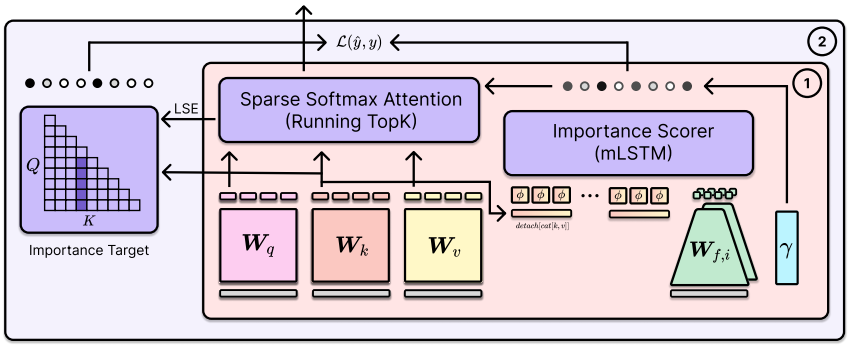

Figure 1: **(1) Training & Inference:** In each attention layer, we compute queries, keys, and values. Keys and values are detached, concatenated as $[k; v]$, and passed to a per-KV-head mLSTM scorer to produce token-importance scores. A fixed-budget KV cache is maintained by always retaining sink tokens and a protected sliding window, and selecting the remaining entries by top-k score. Attention is computed over the retained set. **(2) Training only:** We compute importance targets on-the-fly as the future attention mass of each key and optimize the importance scorer layer-wise.

2 KVPOP

MLSTM Importance Scorer. We provide an overview of our architecture for training and inference in Figure 1. We retrofit a pre-trained LLM by augmenting each attention layer with a lightweight MLSTM scoring module operating *per KV head*. This stateful scorer produces scalar importance scores that drive top-k retention and eviction under a fixed KV budget.

Let w be the length of the protected sliding window during which tokens cannot be evicted. At decoding timestep t, the token position that is about to exit the window is $u := t - w + 1$. Intuitively, KVPOP assigns scores to tokens exactly when they become evictable. We define the mLSTM inputs $\tilde{\boldsymbol{q}}_t, \tilde{\boldsymbol{k}}_t, \tilde{\boldsymbol{v}}_t$ via headwise projections of the base attention keys and values:

$$\boldsymbol{x}_t = [\boldsymbol{k}_t; \boldsymbol{v}_t], \quad \tilde{\boldsymbol{q}}_t = \phi(W_q \boldsymbol{x}_u),\ \tilde{\boldsymbol{k}}_t = \phi(W_k \boldsymbol{x}_t),\ \tilde{\boldsymbol{v}}_t = W_v \boldsymbol{x}_t,$$

where $W_{q/k/v} \in \mathbb{R}^{(2d_{qkv}) \times (d_{qkv}/2)}$ are small head-wise adapters and $\phi(\cdot)$ is a Hedgehog non-linearity (Zhang et al., 2024).

We choose $[\boldsymbol{k}; \boldsymbol{v}]$ as input for two reasons. First, most modern LLMs use grouped-query attention (GQA) Ainslie et al. (2023), so scoring naturally aligns with KV heads and avoids an additional query-to-KV-head projection. Second, using only cached quantities enables *delayed scoring*: at timestep t, the mLSTM memory has been updated with information up to t, but the only token whose eviction status changes is u. Since $\boldsymbol{x}_u = [\boldsymbol{k}_u; \boldsymbol{v}_u]$ is available in the KV cache, we can perform a time-shifted readout by setting $\tilde{\boldsymbol{q}}_t$ from $\boldsymbol{x}_u$.

Let $\boldsymbol{h}_t = \tilde{\boldsymbol{q}}_t \boldsymbol{C}_t / \tilde{\boldsymbol{q}}_t \boldsymbol{z}_t$ denote the delayed mLSTM memory readout at timestep t with delayed query $\tilde{\boldsymbol{q}}_t$, where $\boldsymbol{C}_t \in \mathbb{R}^{d_{qkv} \times d_{qkv}}$ denotes the mLSTM's state of the full sequence up to t and $\boldsymbol{z}_t \in \mathbb{R}^{d_{qkv}}$ the mLSTM normalizer. We map $\boldsymbol{h}_t$ to an importance score for token u via a headwise linear projection: $\text{scôre}_u = \text{w}\,\text{SiLU}(\boldsymbol{h}_t) + \text{b}$. We provide additional details on the mLSTM formulation in Appendix B.

Learnable Temporal Decay. We rank tokens using an effective score $\text{score}_u(q) = \text{scôre}_u + \lfloor \frac{q-u}{n} \rfloor \log \gamma_h$, where $\gamma_h \in (0, 1)$ is a learnable, per-head decay factor. We parameterize $\log \gamma_h$ by mapping an unconstrained parameter through a sigmoid into a fixed log-decay range. The sigmoid is used solely to constrain the learnable parameter during training and is not part of the scoring function itself.

Sparse Attention. At each query position q, we maintain a constant-memory KV cache by selecting the top-k keys with the highest effective importance scores among the eligible tokens (i.e., tokens that are neither sinks nor in the protected window). All other eligible tokens are evicted. Two token sets are exempt from eviction: *sink tokens* (the first s tokens of the sequence, typically s=4), motivated by attention sinks (Xiao et al., 2023), and a *sliding window* of the w most recent tokens

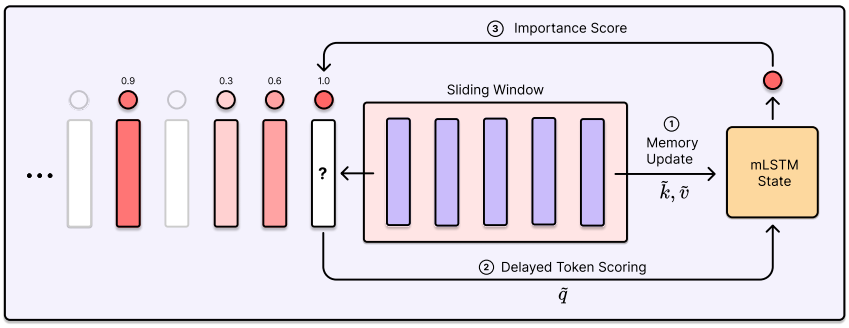

Figure 2: **KVPOP stateful eviction policy**. **(1)** The mLSTM memory is updated with the most recent key value pair at position t. **(2)** For delayed scoring, the memory is read with the token at $t-w+1$ exiting the sliding window cache. **(3)** The mLSTM emits an importance score, and ranking determines whether the token is kept or evicted.

(typically w=64) to preserve local context. Defining the total budget as $B = s + w + k$, attention is computed over the union of sink tokens, window tokens, and the selected top-k tokens. We give an efficient GPU implementation of the running top-k selection using a Fenwick tree (Fenwick, 1994) in Appendix C.

2.1 Training Setup

Importance Score Target. We supervise the scorer with a forward-looking signal: how much future attention a key will receive once it leaves the protected window. Let S denote the sequence length and let $g \in \{1, \dots, G\}$ index the query heads (GQA groups) that share KV head h. For query position d and key position $t \le d$, let $p_{d\to t}^{(h,g)}$ denote the causal attention probability from (h, g) at d to key t. We define the *future attention mass* for key (h, t) as

$$m_{h,t} = \max_{g\in\{1,\dots,G\}} \sum_{d=t+w}^{S-1} p_{d\to t}^{(h,g)}, \qquad p_{d\to t}^{(h,g)} = \frac{\exp\Big(\langle \boldsymbol{q}_d^{(h,g)}, \boldsymbol{k}_t^{(h)}\rangle/\sqrt{d_k}\Big)}{\sum_{t'=0}^{d}\exp\Big(\langle \boldsymbol{q}_d^{(h,g)}, \boldsymbol{k}_{t'}^{(h)}\rangle/\sqrt{d_k}\Big)}. \tag{1}$$

Naively computing $m_{h,t}$ requires the full $S \times S$ attention matrix. Instead, we exploit an identity that reduces the computation to a single *transposed* attention pass. Using FlexAttention Dong et al. (2025), the first (sparse) attention pass returns per-query log-normalizers $\widetilde{\mathrm{LSE}}_d$. A second pass swaps keys and queries and applies the score modifier $\langle \boldsymbol{q}_d^{(h,g)}, \boldsymbol{k}_t^{(h)}\rangle/\sqrt{d_k} - \widetilde{\mathrm{LSE}}_d$, which converts logits into (approximate) log-probabilities.[1] The resulting per-key LSE over $d \ge t + w$ yields an estimate of $\log m_{h,t}$ for all t in parallel. This transposed pass is used only during training (no inference overhead); details and the derivation are in Appendix D.

Importance Score Loss. We train KVPOP with full finetuning and add an auxiliary ranking loss for the importance scorer. This loss directly supervises the eviction decision at each step. When the next token $t_{\text{new}} = q - w$ exits the protected window and becomes eligible for the top-k budget, the policy must decide whether to retain it.

Let τ_q denote the teacher's cutoff rank at query position q (computed from target effective scores), and let t_{bnd} be the boundary key at rank τ_q among eligible keys. The teacher keeps t_{new} if $\mathrm{rank}(t_{\text{new}}) \le \tau_q$, and evicts it otherwise. We train the predictor to match this decision via a pairwise logistic loss:

$$\mathcal{L}_{\text{score}} = \mathbb{E}_q\Big[\mathrm{softplus}\Big(-y_q\,\frac{\text{score}_{\text{new}} - \text{score}_{\text{bnd}}}{\tau}\Big)\Big], \tag{2}$$

where $y_q = +1$ if the teacher keeps t_{new} and $y_q = -1$ otherwise, $\text{score}_{\text{new}}$ and $\text{score}_{\text{bnd}}$ are the predicted *effective* scores (including decay), and τ is a temperature.

[1] The computation is exact if $\widetilde{\mathrm{LSE}}_d$ is replaced by the dense causal LSE_d.

Table 1: Downstream results on **AIME** and **HMMT**. We report the relative **pass@1** performance w.r.t. the dense uptraining checkpoint. Absolute scores are shown in parentheses.

		AIME		HMMT		Avg.
	Variant	2024	2025	Feb 2025	Nov 2025	
–	QWEN3-4B-IT	1.00 (0.49)	.93 (0.40)	1.08 (0.22)	1.01 (0.30)	.99 (0.35)
–	QWEN3-4B-IT Uptrain	1.00 (0.49)	1.00 (0.43)	1.00 (0.20)	1.00 (0.30)	1.00 (0.36)
B=512	QWEN3-4B-IT StreamLLM	.07 (0.03)	.16 (0.07)	.24 (0.05)	.06 (0.02)	.12 (0.04)
B=512	QWEN3-4B-IT StreamLLM+	.40 (0.20)	.44 (0.19)	.43 (0.09)	.28 (0.08)	.39 (0.14)
B=512	QWEN3-4B-IT KVPOP	**.78 (0.38)**	**.76 (0.33)**	**.86 (0.18)**	**.85 (0.25)**	**.80 (0.28)**
B=2048	QWEN3-4B-IT StreamLLM	.75 (0.37)	.66 (0.28)	.98 (0.20)	.73 (0.22)	.75 (0.33)
B=2048	QWEN3-4B-IT StreamLLM+	.86 (0.42)	.74 (0.32)	**1.10 (0.23)**	1.10 (0.33)	.91 (0.43)
B=2048	QWEN3-4B-IT KVPOP	**.93 (0.46)**	**.81 (0.35)**	.90 (0.18)	**1.13 (0.33)**	**.93 (0.43)**

This objective (1) concentrates supervision on the critical boundary decision rather than all pairwise comparisons, (2) naturally incorporates temporal decay through the effective scores, and (3) is $O(1)$ per sampled query position. We optionally apply margin-based weighting using the teacher margin $|\text{score}^{\text{tgt}}_{\text{new}} - \text{score}^{\text{tgt}}_{\text{bnd}}|$ to downweight ambiguous decisions.

2.2 EXPERIMENTS

All experiments start from QWEN3-4B-IT (Yang et al., 2025) and use sequence length $S = 8192$. For reasoning uptraining, we use the Nemotron-Math v2 dataset Du et al. (2025), selecting the subset with medium-length thinking traces, which fit well within $S = 8192$ with the Qwen tokenizer. We apply sequence packing to improve token utilization. We train using a two-stage protocol:

Phase 1 (reasoning uptraining). We fully finetune for 5,000 steps (global batch 128) with a constant learning rate of 8×10^{-5}, optimizing next-token cross-entropy.

Phase 2 (KVPOP sparsification). Starting from the Phase-1 checkpoint, we train for 2,000 additional steps on the same data (new seed) and initialize per-layer, per-KV-head mLSTM scorers. We use a cosine schedule for the scorer parameters (peak 10^{-3}) while keeping the base-model learning rate at 8×10^{-5}, and optimize a mixed objective: 0.9 cross-entropy + 0.1 KL distillation. Full hyperparameters are in Appendix E.

Results. We report **pass@1** on AIME24/25 and HMMT (Feb/Nov 2025) in Table 1, comparing dense attention to StreamingLLM (Xiao et al., 2024) and KVPOP under two cache budgets ($B \in \{512, 2048\}$). For sparse variants we report performance relative to the uptrained dense checkpoint, with absolute scores in parentheses. At B=512 ($\sim$94% compression), applying StreamingLLM without additional training collapses performance (Avg. 0.12), and even training under the fixed mask (StreamLLM+) recovers only partially (Avg. 0.39). In contrast, KVPOP retains a large fraction of the dense baseline at the same budget (Avg. 0.80), indicating that learned, context-aware retention is critical under aggressive compression. At B=2048 ($\sim$75% compression), all methods improve; KVPOP remains best on average (Avg. 0.93), closely matching dense attention and slightly outperforming StreamLLM+ (Avg. 0.91). Overall, fixed window+sinks masking can be brittle for long-horizon reasoning, while KVPOP's delayed mLSTM scoring yields robust retention decisions, especially at small budgets.

3 CONCLUSION

We introduce KVPOP, a lightweight retrofit for fixed-budget KV-cache management. Our key design is stateful delayed scoring at eviction time, which lets the importance scorer incorporate near-future context while preserving causality. We also introduce a novel future-attention importance target to supervise eviction decisions. Combined, these components keep inference memory bounded and make training practical with efficient sparse selection. Across long-context math reasoning evaluations, KVPOP preserves most gains from math uptraining while reducing the KV cache by 93.75%. Consequently, KVPOP represents a promising alternative to dense attention mechanisms.

ACKNOWLEDGMENTS

We acknowledge EuroHPC Joint Undertaking for awarding us access to Leonardo at CINECA, Italy, Deucalion at MACC, Portugal, Discoverer at SofiaTech, Bulgaria.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

Yash Akhauri, Ahmed F. AbouElhamayed, Yifei Gao, Chi-Chih Chang, Nilesh Jain, and Mohamed S. Abdelfattah. TokenButler: Token Importance is Predictable, March 2025. arXiv:2503.07518 [cs].

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended Long Short-Term Memory. *Advances in Neural Information Processing Systems*, 37: 107547–107603, December 2024. doi: 10.52202/079017-3417.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

Ricardo Buitrago and Albert Gu. Understanding and improving length generalization in recurrent models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2OEb20dy7B.

Juechu Dong, BOYUAN FENG, Driss Guessous, Yanbo Liang, and Horace He. Flexattention: A programming model for generating fused attention variants. In *Eighth Conference on Machine Learning and Systems*, 2025.

Wei Du, Shubham Toshniwal, Branislav Kisacanin, Sadegh Mahdavi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. Nemotron-math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision. *arXiv preprint arXiv:2512.15489*, 2025.

Peter M. Fenwick. A new data structure for cumulative frequency tables. *Software—Practice and Experience*, 24(3):327–336, 1994. doi: 10.1002/SPE.4380240306.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *Conference on Language Modeling*, volume 1, Philadelphia, PA, USA, 2024. OpenReview.

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

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*, volume 119, pp. 5156–5165, virtual, 2020. PMLR. arXiv: 2006.16236.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, October 2023. Association for Computing Machinery. ISBN 979-8-4007-0229-7. doi: 10.1145/3600006.3613165.

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.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic Memory Compression: Retrofitting LLMs for Accelerated Inference. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 37396–37412. PMLR, July 2024. ISSN: 2640-3498.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-Aware Sparsity for Efficient Long-Context LLM Inference. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 47901–47911. PMLR, July 2024. ISSN: 2640-3498.

DeepSeek Team. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models, December 2025. arXiv:2512.02556 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, Long Beach, CA, USA, 2017. Curran Associates, Inc. arXiv: 1706.03762.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. October 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

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

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 56501–56523, Vienna, Austria, 2024b. PMLR. ISSN: 2640-3498.

Zihao Zeng, Bokai Lin, Tianqi Hou, Hao Zhang, and Zhijie Deng. In-context KV-Cache Eviction for LLMs via Attention-Gate, April 2025. arXiv:2410.12876 [cs].

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4g02l2N2Nx.

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. *Advances in Neural Information Processing Systems*, 36:34661–34710, December 2023.

Adrian Łańcucki, Konrad Staniszewski, Piotr Nawrot, and Edoardo Ponti. Inference-Time Hyper-Scaling with KV Cache Compression. October 2025.

A Related work: Learned KV Cache Eviction and Compression

Selection vs. eviction. A useful distinction is whether a method sparsifies *attention computation* or *cache memory*. *Token selection* methods restrict each query to attend to a subset of keys (reducing compute and bandwidth), but under standard autoregressive decoding, they do not necessarily bound the size of the persistent KV cache, since keys/values for all past tokens may remain stored for potential future use. Classic sparse-attention architectures such as Longformer use fixed sparse patterns (local windows plus global tokens) to obtain near-linear attention complexity for long sequences Beltagy et al. (2020). More recent long-context models adopt learned or indexed token selection to reduce attention cost at scale; e.g., DeepSeek-V3.2 introduces DeepSeek Sparse Attention (DSA) to substantially reduce long-context attention complexity while preserving quality Team (2025). In contrast, *eviction* methods explicitly *drop* cached tokens to enforce a hard memory budget, directly reducing the KV footprint during long-context generation.

Training-free eviction and prompt-side compression. Several influential methods reduce KV memory without additional training by combining recency with heuristic notions of global importance. H_2O observes that attention patterns exhibit "heavy hitter" tokens that contribute disproportionately to generation quality and proposes an eviction policy that retains a mixture of recent tokens and heavy hitters, with theoretical motivation via a submodular formulation Zhang et al. (2023). SnapKV proposes a fine-tuning-free strategy that compresses the KV cache by leveraging prompt attention features extracted from an "observation window" near the end of the prompt and forming a reduced cache used for subsequent generation Li et al.. These approaches are effective baselines, but their importance signals are either retrospective (e.g., accumulated attention) or derived from a prompt-only structure, which can make them slower to adapt when token utility changes over the course of generation.

Learned KV cache eviction/compression. A growing line of work trains lightweight mechanisms to make cache management decisions that better preserve accuracy under tight budgets. Inference-Time Hyper-Scaling with KV Cache Compression argues that KV size, rather than token count, is a primary bottleneck for inference-time scaling, and introduces Dynamic Memory Sparsification (DMS), which can be trained briefly to achieve high compression while maintaining accuracy Łańcucki et al. (2025). Notably, DMS emphasizes *delayed eviction* and implicit information merging to avoid prematurely discarding useful context Łańcucki et al. (2025). Overall, these results support the hypothesis that token importance is predictable and that small learned modules can outperform purely heuristic retention policies at high compression ratios.

Positioning of KV Pop. KVPOP falls in the *learned eviction* category: it enforces a constant-size cache by evicting tokens, while computing attention only over the retained set. Unlike prompt-only compression, KVPOP predicts importance *online* during generation, and unlike purely retrospective scoring, it trains a forward-looking predictor using supervision derived from future attention mass. Crucially, KVPOP uses a protected sliding window and *delayed scoring*: each token is scored right before it becomes evictable, allowing the scorer to exploit near-future context already summarized in its recurrent state at decision time.

B Background: Linear Attention and mLSTM

Linear attention replaces the softmax similarity $\kappa_{\exp}(\boldsymbol{q}, \boldsymbol{k}) = \exp\big(\boldsymbol{q}^\top \boldsymbol{k}/\sqrt{d_{qk}}\big)$ with a kernel $\kappa_\phi(\boldsymbol{q}, \boldsymbol{k}) = \phi(\boldsymbol{q})^\top \phi(\boldsymbol{k})$ that admits an explicit feature representation, where $\phi : \mathbb{R}^{d_{qk}} \to \mathbb{R}^{d_{qk}}$ (Katharopoulos et al., 2020).

Through associativity, this factorization yields two mathematically equivalent causal attention implementations: (i) a (chunkwise) parallel computation suited for training and prefill, and (ii) an online recurrent update used for step-by-step decoding. By switching between these views, one obtains linear-time prefill and training and constant-memory autoregressive generation (Yang et al., 2024b).

In the recurrent formulation, each head maintains a running key–value summary $\boldsymbol{C}_t \in \mathbb{R}^{d_{qk} \times d_v}$, optionally together with a normalizer $\boldsymbol{z}_t \in \mathbb{R}^{d_{qk}}$. Given the token at time t, the state is updated via

rank-one outer products:

$$\boldsymbol{C}_t = \boldsymbol{C}_{t-1} + \phi(\boldsymbol{k}_t) \otimes \boldsymbol{v}_t,$$
$$\boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \phi(\boldsymbol{k}_t),$$

where $\otimes$ denotes the outer product. For a query $\boldsymbol{q}_t$, the head output is obtained by reading from the current summary with normalization:

$$\boldsymbol{h}_t = \frac{\phi(\boldsymbol{q}_t)\boldsymbol{C}_t}{\phi(\boldsymbol{q}_t)\boldsymbol{z}_t}.$$

Here $\boldsymbol{q}_t, \boldsymbol{k}_t \in \mathbb{R}^{d_{qk}}$ and $\boldsymbol{v}_t \in \mathbb{R}^{d_v}$.

mLSTM. Recently, modern recurrent architectures, such as mLSTM (Beck et al., 2024), Mamba (Gu & Dao, 2024), and Gated Delta Networks (Yang et al., 2024a) have emerged as competitive linear-complexity alternatives to Transformers. Inspired by the gating structure of the original LSTM cell (Hochreiter & Schmidhuber, 1997), these operators augment the outer product update of linear attention with expressive gates. In this work, we choose mLSTM, which was introduced as a sub-layer of xLSTM, as a stateful importance scorer. The mLSTM introduces three input-dependent gates into the state update of linear attention, each governing a different part of the computation. Let $\boldsymbol{w}_i \in \mathbb{R}^{d\times 1}$ and $\boldsymbol{w}_f \in \mathbb{R}^{d\times 1}$ parameterize scalar input and forget gates, and let $\boldsymbol{W}_{og} \in \mathbb{R}^{d\times d_v}$ parameterize a vector-valued output gate. Given a token representation $\boldsymbol{x}_t$, we compute

$$i_t = \exp(\boldsymbol{x}_t \boldsymbol{w}_i), \qquad f_t = \sigma(\boldsymbol{x}_t \boldsymbol{w}_f), \qquad \boldsymbol{o}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W}_{og}),$$

where i_t scales the new key–value write, f_t attenuates the running state, and $\boldsymbol{o}_t$ gates the readout. The resulting recurrent updates are

$$\boldsymbol{C}_t = f_t\, \boldsymbol{C}_{t-1} + i_t\, \phi(\boldsymbol{k}_t) \otimes \boldsymbol{v}_t,$$
$$\boldsymbol{z}_t = f_t\, \boldsymbol{z}_{t-1} + i_t\, \phi(\boldsymbol{k}_t),$$

with numerical stabilization for the exponential input gate omitted here for clarity. A query then performs the usual normalized retrieval, and the output gate modulates it elementwise:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \frac{\phi(\boldsymbol{q}_t)\boldsymbol{C}_t}{\phi(\boldsymbol{q}_t)\boldsymbol{z}_t}. \tag{3}$$

Lightweight stateful scoring with mLSTM. We tailor the mLSTM scorer for minimal retrofit overhead in a pretrained Transformer. We remove the mLSTM output gate, reuse the Transformer's existing attention projections, and build scorer inputs solely from cached quantities. For each KV head, we form $x_t = [k_t; v_t] \in \mathbb{R}^{2d_{qkv}}$ and apply a small head-specific linear projection $W_{q/k/v} \in \mathbb{R}^{(2d_{qkv})\times(d_{qkv}/2)}$ followed by a *Hedgehog* activation ϕ in its softmax-stabilized form (Zhang et al., 2024):

$$\phi(\mathbf{x}_t) = \big[\mathrm{softmax}(\mathbf{x}_t); \mathrm{softmax}(-\mathbf{x}_t)\big] \in \mathbb{R}^{d_{qkv}},$$

where the softmax is taken over the feature dimension. Crucially, this adds only one small matrix for each query, key, and value head.

C Efficient Running Top-k Attention

While top-k attention during inference can be implemented by attending only to the non-evicted KV-cache entries, training benefits from a parallel implementation that avoids explicitly constructing an $S \times S$ sparse mask.

Let s be the number of sink tokens and w the protected sliding-window length. For each query position q, only keys in the *eligible* set $\mathcal{E}(q) = \{t \mid s \le t \le q - w\}$ compete for the top-k budget (sink and window tokens are always kept). We rank keys by their *effective score* (predicted score plus decay) and define $\mathrm{rank}(t) \in \{0, \dots, S-1\}$ as the position of key t in the descending sort (smaller is better).

We compute, for each q, a *threshold rank* τ_q such that the selected long-range keys are exactly those in $\mathcal{E}(q)$ with $\mathrm{rank}(t) \le \tau_q$. To do so efficiently, we use an online Fenwick tree (Binary Indexed Tree) (Fenwick, 1994):

- **Precompute ranks:** sort effective scores once to obtain $\mathrm{rank}(t)$ for all t.
- **Online thresholds:** as q increases, exactly one new key $t = q - w$ enters $\mathcal{E}(q)$; we insert its rank into the BIT. The cutoff τ_q is the rank of the k-th best inserted element, found by binary lifting on BIT prefix sums.

The resulting attention mask can be written as

$$M_{q,t} = \mathbf{1}[t < s] \ \vee \ \mathbf{1}[q - t < w] \ \vee \ \Big(\mathbf{1}[t \in \mathcal{E}(q)] \wedge \mathbf{1}[\mathrm{rank}(t) \le \tau_q]\Big). \tag{4}$$

This procedure runs in $O(S \log S)$ time and $O(S)$ space per head, and enables constructing the mask on-the-fly inside a sparse attention kernel (we use FLEXATTENTION Dong et al. (2025)) without materializing an $O(S^2)$ matrix.

D EFFICIENT TARGET IMPLEMENTATION

This appendix derives an efficient computation of the future-attention target in Eq. equation **??** without materializing an $S \times S$ attention matrix.

Setup. Fix a KV head h and (optionally) a query group g mapping to h. Let

$$s^{(h,g)}(d,t) := \langle \boldsymbol{q}_d^{(h,g)}, \boldsymbol{k}_t^{(h)} \rangle / \sqrt{d_k}$$

be the pre-softmax score, and define the causal per-query log-normalizer

$$\mathrm{LSE}_d^{(h,g)} := \log \sum_{t'=0}^{d} \exp\big(s^{(h,g)}(d,t')\big). \tag{5}$$

The causal attention probability is then

$$p_{d\to t}^{(h,g)} = \exp\Big(s^{(h,g)}(d,t) - \mathrm{LSE}_d^{(h,g)}\Big), \qquad t \le d. \tag{6}$$

Future attention mass. For each key position t, we aggregate attention from future queries outside the protected window:

$$m_t^{(h,g)} := \sum_{d=t+w}^{S-1} p_{d\to t}^{(h,g)} = \sum_{d=t+w}^{S-1} \exp\Big(s^{(h,g)}(d,t) - \mathrm{LSE}_d^{(h,g)}\Big). \tag{7}$$

In the main paper, we take an existential aggregation across query groups, $m_{h,t} = \max_g m_t^{(h,g)}$, which matches the intuition that if *any* query head strongly relies on a token, the KV head should retain it.

Transposed-attention identity. Equation equation 7 is a log-sum-exp over d for each fixed t:

$$\log m_t^{(h,g)} = \log \sum_{d=t+w}^{S-1} \exp\Big(s^{(h,g)}(d,t) - \mathrm{LSE}_d^{(h,g)}\Big). \tag{8}$$

This can be computed by a *transposed* attention-like operation where key positions t act as "queries" and time indices d act as "keys":

$$\log m_t^{(h,g)} = \mathrm{LSE}_t^{\mathrm{T}}\left(s^{(h,g)}(d,t) - \mathrm{LSE}_d^{(h,g)}\right), \qquad \text{with } d \ge t + w. \tag{9}$$

Concretely, define a second attention call with query vectors $\boldsymbol{q}'_t := \boldsymbol{k}_t^{(h)}$ and key vectors $\boldsymbol{k}'_d := \boldsymbol{q}_d^{(h,g)}$ (i.e., swap the roles of $\boldsymbol{Q}$ and $\boldsymbol{K}$). Using the same dot-product kernel yields logits $s^{(h,g)}(d,t)$. Applying the score modifier $-\mathrm{LSE}_d^{(h,g)}$ converts these logits into log-probabilities (Eq. equation 6), and taking the log-sum-exp over the masked set $\{d : d \ge t + w\}$ yields Eq. equation 8.

Dense vs. sparse log-normalizers. If $\mathrm{LSE}_d^{(h,g)}$ in Eq. equation 5 is computed from *dense* causal attention, then Eq. equation 9 yields the *exact* $\log m_t^{(h,g)}$. In practice, our first pass is sparse top-k attention and returns an approximate log-normalizer

$$\widetilde{\mathrm{LSE}}_d^{(h,g)} = \log \sum_{t'=0}^{d} M_{d,t'} \exp\big(s^{(h,g)}(d,t')\big),$$

where M is the sparse attention mask. Replacing $\mathrm{LSE}_d^{(h,g)}$ with $\widetilde{\mathrm{LSE}}_d^{(h,g)}$ yields an approximation of $\log m_t^{(h,g)}$ that we find empirically sufficient, as the top-k set captures most of the softmax mass.

Implementation with FLEXATTENTION. We compute the target in four steps:

1. Run the first (sparse) attention pass and obtain per-query $\widetilde{\mathrm{LSE}}_d^{(h,g)}$ from FLEXATTENTION.
2. Run a second *transposed* FLEXATTENTION pass with swapped inputs ($\boldsymbol{K} \to \boldsymbol{Q}'$, $\boldsymbol{Q} \to \boldsymbol{K}'$).
3. In the transposed pass, apply the score modifier $s^{(h,g)}(d,t) \mapsto s^{(h,g)}(d,t) - \widetilde{\mathrm{LSE}}_d^{(h,g)}$ and a block mask enforcing $d \geq t + w$.
4. The auxiliary output log-sum-exp from this transposed pass yields $\log m_t^{(h,g)}$ for all t in parallel. We aggregate across query groups by $\log m_{h,t} = \max_g \log m_t^{(h,g)}$ (equivalently $m_{h,t} = \max_g m_t^{(h,g)}$).

This transposed pass is used only during training to compute targets and is not required at inference time.

E EXPERIMENT DETAILS

Training details. All experiments were run on 8 H100 GPUs using PyTorch FSDP. We used a global batch size of 128 via gradient accumulation, mixed precision (bfloat16 for model parameters and most compute; float32 for gradient all-gather/reductions), and gradient clipping to 1.0 for full finetuning. For numerical stability, we compute score decay and top-k ranking/selection in float32; all other activations remain in bfloat16.

To maximize GPU utilization, we pack multiple samples into a single sequence up to the maximum context length. We found that *preserving the attention mask across packed segments* (i.e., not resetting attention at packing boundaries) improved performance for our hybrid architecture, consistent with observations in prior work (Buitrago & Gu, 2025).

Uptraining. In the uptraining phase, we train for 5,000 steps with global batch size 128 and a constant learning rate. We optimize the model using the standard cross-entropy (CE) next-token objective.

Sparsification. In the sparsification phase, we train KV Pop on top of the uptrained checkpoint using a mixed objective that includes a KL-divergence term to keep the student close to the teacher distribution. We compute the KL term over the top-256 teacher logits for efficiency. This truncation substantially reduces the cost of teacher supervision and suggests a variant where teacher logits could be precomputed offline, avoiding loading the teacher model during training. We leave this optimization for future work.

Table 2 summarizes the hyperparameters used throughout our pipeline.

F ARCHITECTURE ABLATION

We ablate architectural components of the mLSTM scorer on LLAMA3.2-3B-IT (Figure 3). We train for 3,000 steps with sequence length 4,096 on the Longmino-3 midtraining mixture Olmo et al.

Table 2: Training Hyperparameters.

Setting	Value
General	
Context size	8192
Batch size	128
Weight decay	0.0001
Optimizer	AdamW
Adam Betas	(0.9, 0.95)
Trainable Parameters	all
Sequence packing	true
Phase I: Uptraining	
Token budget	5B
Learning rate	Constant LR $8e{-5}$
CE loss weight	1.0
KL loss weight	0.0
Phase II: Sparsification	
Token budget	2B
Learning rate (existing params)	Constant LR $8e{-5}$
Learning rate (new params)	100 Steps Warmup + Cosine Schedule ($1e{-3}$ to $8e{-5}$)
CE loss weight	0.9
KL loss weight	0.1
KL temperature	1
KL reverse	false
KL Top-k	256
Pairwise loss temperature	1.0
Pairwise loss margin weighting	true
KVPOP sliding window	64
KVPOP sink tokens	4
KVPOP top-k budget	444 & 1980
KVPOP decay min	0.999
KVPOP decay max	0.999999

(2025). We use a sliding window size of 64 and a total attention budget of 512. Delayed scoring, i.e., scoring tokens when they exit the protected window rather than at insertion, yields a substantial improvement over immediate scoring. Other architectural variants have comparable performance.

Based on these results, we omit the headwise LayerNorm used in xLSTM Beck et al. (2024) to reduce parameter overhead, and use hedgehog feature maps Zhang et al. (2024) which halve the projection parameter count by reducing the feature dimension in the headwise maps from $[k; v]$ to $\tilde{q}$, $\tilde{k}$, and $\tilde{v}$. We retain gate softcapping since it introduces no additional parameters and can be implemented efficiently (e.g., fused into the mLSTM kernel).

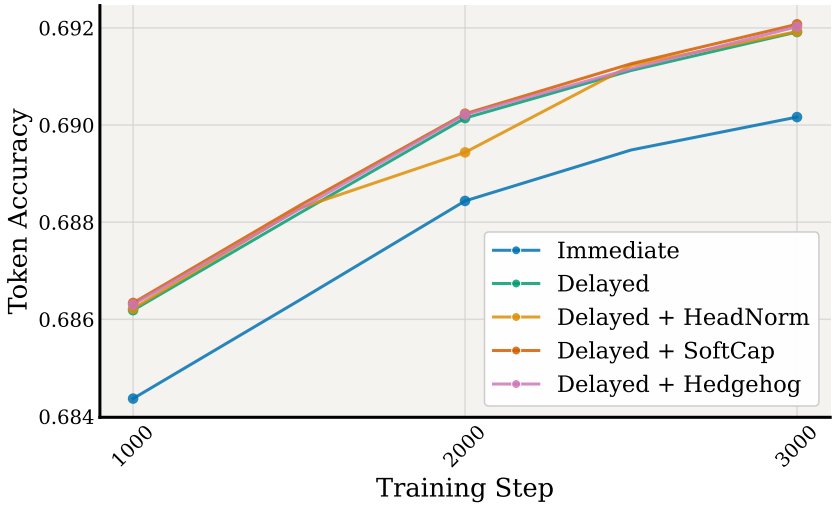

Figure 3: Different configurations of the mLSTM scoring module for LLAMA3.2-3B-IT on the Longmino-3 pretraining mix.