Gated KalmaNet: A Fading Memory Layer through Test-Time Ridge Regression

Aditya Chattopadhyay
Applied Scientist, AWS Agentic AI

Collaborators: *Liangzu Peng[1], Luca Zancato, Elvis Nunez, Wei Xia, Stefano Soatto*

[1]Work done during an internship at AWS Agentic AI.

Rise of Long-context LLMs

- Long-context processing imperative to unlocking new model capabilities.
  - Code automation over large repositories.
  - Aggregate information across multiple documents.
  - Long-consistent reasoning traces for agentic applications.

Rise of Long-context LLMs

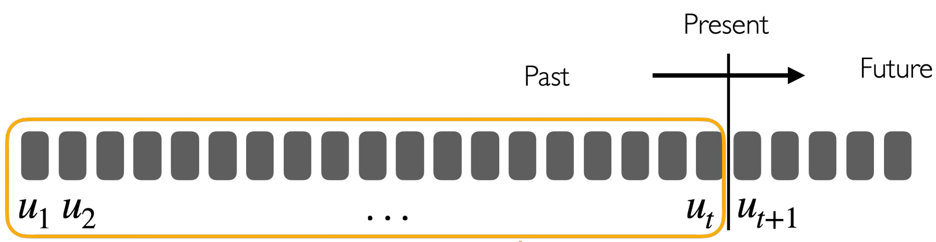
- Significant effort devoted in recent years at increasing context length of LLMs.
  - From 8k in 2023 to over 1M tokens in 2025!

Model	Context Window	Artificial Analysis Intelligence Index
Gemini 3 Pro Preview (high) (Vertex)	1m	73
Gemini 3 Pro Preview (high) (AI Studio)	1m	73
Claude Opus 4.5	200k	70
Claude Opus 4.5 Vertex	200k	70
Claude Opus 4.5	200k	70
GPT-5.1 (high)	400k	70
GPT-5 Codex (high)	400k	68
Kimi K2 Thinking Turbo	262k	67

Image: https://huggingface.co/spaces/ArtificialAnalysis/LLM-Performance-Leaderboard

Memory for sequence models

- Memory is a ***sufficient statistic*** of the past.

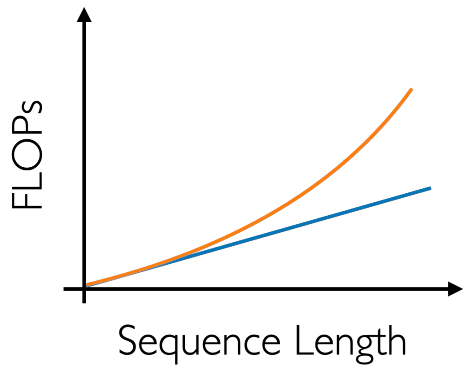


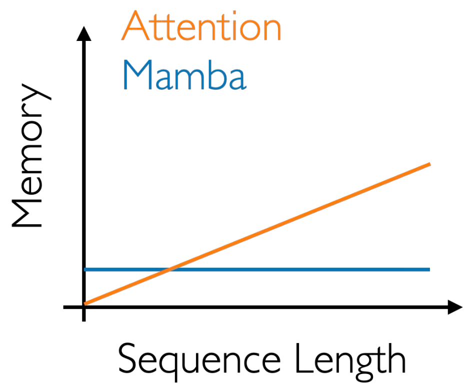
Memory: Function of the past that makes the future conditionally independent of the past

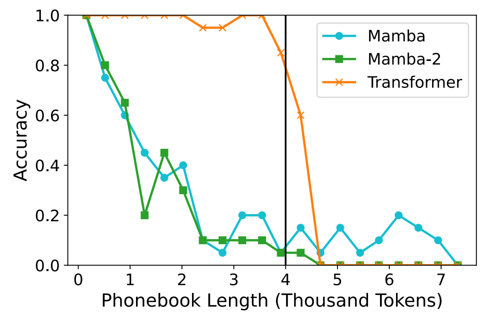
$$P(u_{t+1} \mid x_t) = P(u_{t+1} \mid u^t_{-\infty})$$

- **Challenge:** Data generating mechanism not known *a priori*.

Eidetic vs Fading Memory







- Eidetic (Attention): Keeps entire past verbatim in memory, high compute and storage costs, Perfect recall.
- Fading (SSMs): Compresses the past into a fixed-sized memory, low compute and storage costs, Low recall.

Designing memory through test-time regression

- Attention can be seen as the solution to the following regression problem[1].

$$y_t = \operatorname*{argmin}_{v} \sum_{i=1}^{t} \exp\left(\frac{k_i^\top q_t}{\sqrt{D}}\right) \cdot \|v - v_i\|_2^2.$$

$$y_t = \sum_{i=1}^{t} c_i v_i, \qquad c_i := \frac{\exp(\frac{k_i^\top q_t}{\sqrt{D}})}{\sum_{i=1}^{t} \exp(\frac{k_i^\top q_t}{\sqrt{D}})}. \qquad \text{(Attn)}$$

- Uses entire KV-cache to solve its regression objective.

1. Wang et al. "Test-time regression: a unifying framework for designing sequence models with associative memory." arXiv preprint arXiv:2501.12352 (2025).

Designing memory through test-time regression

- Several existing SSMs can be seen as solving an *instantaneous* objective[1,2],
  - depends only on current key, value and previous lossy memory.

- Examples,
  - DeltaNet is 1-step SGD applied to $\mathcal{L}_t = \left|\left|Sk_t - v_t\right|\right|_2^2$

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t\right) \mathbf{k}_t^\top \end{aligned}$$

    - SGD on $\mathcal{L}_t$ with initialization $\gamma_t S_{t-1}$ gives the Gated DeltaNet update, $\beta_t, \gamma_t \in (0,1)$

1. Yang, et al. "Parallelizing linear transformers with the delta rule over sequence length." Advances in neural information processing systems 37 (2024): 115491-115522.
2. Wang et al. "Test-time regression: a unifying framework for designing sequence models with associative memory." arXiv preprint arXiv:2501.12352 (2025).

Attention vs. existing SSM layers

- We hypothesize this myopic view of SSM objectives result in their lower performance and limited long-context abilities.
  - SSMs update memory based on current time-step and lossy previous memory.
  - Attention uses the entire exact KV-cache to solve its objective.

- What is an **objective** that considers the **entire past as Attention** while still being solvable **in linear time as linear SSMs**?

Gated KalmaNet: A Linear SSM Layer Inspired by the Kalman Filter

Motivation from Kalman Filter (KF)

- KF is an established online approach that takes **exact past** into account to **optimally solve** a Weighted **Ridge Regression** (WRR) objective.

$$S_t = \underset{S \in \mathbb{R}^{D \times D}}{\arg\min}\, \lambda \cdot \|S\|_{\mathrm{F}}^2 + \sum_{i=1}^{t} \eta_i \cdot \|Sk_i - v_i\|_2^2, \qquad \text{(WRR)}$$

- Unlike existing SSMs, takes entire past into account ☺
  - More expressive than Linear Attention, Mamba2, Gated DeltaNet etc.
- Unlike Attention, does not need to store the entire KV-cache ☺
  - Thanks to the parametric linear estimator S_t that enables a constant-sized memory.

Contrasting with Attention

Our objective (WRR)

$$S_t = \underset{S \in \mathbb{R}^{D \times D}}{\arg\min}\, \lambda \cdot \|S\|_{\mathrm{F}}^2 + \sum_{i=1}^{t} \eta_i \cdot \|Sk_i - v_i\|_2^2,$$

Attention's objective (Attn)

$$y_t = \underset{v}{\operatorname{argmin}} \sum_{i=1}^{t} \exp\left(\frac{k_i^\top q_t}{\sqrt{D}}\right) \cdot \|v - v_i\|_2^2.$$

- ❖ Attn learns a non-parametric estimator while WRR computes a parametric linear estimator.
  - Thus, no need to store the entire KV-cache.
- ❖ Attn has query-dependent weights, WRR has weights that are input-dependent and exponentially fading time (more on this later).
- ❖ WRR has constant-sized memory, need regularization to prevent fuzzy recall. λ controls memorization capacity of our memory.

aws

Hurdles Towards Scalable Kalman Filter SSMs

- Memory update for KF,

$$S_t = S_{t-1} - \frac{\overbrace{(S_{t-1}k_t - v_t)}^{\text{Innovation/Surprise}} k_t^\top \Phi_{t-1}}{1/\eta_t + k_t^\top \Phi_{t-1} k_t}, \qquad \textbf{(KF)}$$

Φ_{t-1} is inverse of the Hessian of WRR at time t – 1.

- Challenges:
  - *Parallelizable Training:* KF is sequential and lacks a hardware-aware implementation.
  - *Numerical Stability:* KF involves matrix inversion that can be numerical unstable in low-precision LLM training environments.

Gated KalmaNet (GKA)

- Our technical contribution in GKA is to solve the aforementioned challenges for implementing KF at scale.

- First step is to observe that WRR admits a closed form solution.

$$S_t = \underset{S \in \mathbb{R}^{D \times D}}{\arg\min}\, \lambda \cdot \|S\|_{\mathrm{F}}^2 + \sum_{i=1}^{t} \eta_i \cdot \|Sk_i - v_i\|_2^2, \quad \text{(WRR)}$$

$$S_t = \left(\sum_{i=1}^{t} \eta_i v_i k_i^\top\right)\left(\sum_{i=1}^{t} \eta_i k_i k_i^\top + \lambda I\right)^{-1}$$

Input-dependent gates

Gated KalmaNet (GKA): Forward Pass

- Thus, the complete GKA forward pass at time t becomes.

$$S_t = \underbrace{\left(\sum_{i=1}^{t} \eta_i v_i k_i^\top\right)}_{U_t} \left(\underbrace{\sum_{i=1}^{t} \eta_i k_i k_i^\top}_{H_t} + \lambda I\right)^{-1} \quad \text{(Memory update)}$$

$$y_t = S_t q_t \quad \text{(Readout)}$$

- Steps to compute y_t,
  - Solve for x, $(\mathrm{H_t} + \lambda I)x = q_t$. ← Naïve computation takes $O(d^3)$, where $\mathrm{q_t} \in R^d$
  - Compute $y_t = U_t x$

Innovation 1: Parallel Training

- Steps to compute y_t,
  - Solve for x, $(\mathrm{H_t} + \lambda I)x = q_t$. ← Naïve computation takes $O(d^3)$, where $\mathrm{q_t} \in R^d$
  - Compute $y_t = U_t x$

- We employ Chebyshev Iteration (CH), a first-order iterative method to solve for x.
  - Reduces complexity from $O(d^3) \to O(d^2 r)$, where r is number of iterations.
    - $r \leq 30$ iterations in our experiments in the paper, compared to $d = 128$.
  - Allows for an efficient parallel implementation via matrix-vector products.

Innovation 2: Adaptive Regularization

- Recall, KF involves matrix inversion that is sensitive to condition number.
  - This is the step where we solve for x in, $(\mathrm{H_t} + \lambda I)x = q_t$.

$$\sum_{i=1}^{t} \eta_i k_i k_i^\top$$

- Propose **adaptive regularization** to control the condition number.
  - Specifically, we set, $\lambda_t = a||H_t||_F$, then the condition number κ_t can be bounded by

$$\kappa_t = \frac{\lambda_{\max}(H_t) + \lambda_t}{\lambda_{\min}(H_t) + \lambda_t} \leq \frac{\|H_t\|_{\mathrm{F}} + \lambda_t}{\lambda_t} = \frac{a+1}{a}.$$

Innovation 3: Adaptive Weighting

- Recall our WRR objective to compute our memory at each time step

$$S_t = \underset{S \in \mathbb{R}^{D \times D}}{\arg\min}\, \lambda \cdot \|S\|_{\mathrm{F}}^2 + \sum_{i=1}^{t} \eta_i \cdot \|Sk_i - v_i\|_2^2, \quad \text{(WRR)}$$

- To make it more expressive, we make the weights input and time-dependent, that is, $\eta_i \to \eta_{i,t}$.

Function of the current input

- The weights are defined recursively, $\eta_{i,t} = \gamma_t\ \eta_{i,t-1}$

Connections with existing SSM layers

A Dynamical System for Fading Memory

- We posit a Linear Gaussian Model for fading memory.

$$s_t = A_t s_{t-1} + B_t u_t + w_t, \qquad w_t \sim \mathcal{N}(0, Q_t) \quad \text{State Transition Equation}$$

$$v_t = k_t^\top s_t + \mu_t, \qquad \mu_t \sim \mathcal{N}(0, r_t), \quad \text{Measurement Equation}$$

- $s_t \in \mathbb{R}^n$ is a latent state that summarizes the past.
- $u_t \in \mathbb{R}^n$ is the control input that updates the state.
- A_t controls "how much of the past to forget".
- B_t controls "how much of the current input to remember".
- k_t, v_t are the keys and values observed at time t.

Kalman Filter (KF) for Optimal Inference

- Given the following model for Fading memory,

$$s_t = A_t s_{t-1} + B_t u_t + w_t, \qquad w_t \sim \mathcal{N}(0, Q_t) \quad \text{State Transition Equation}$$

$$v_t = k_t^\top s_t + \mu_t, \qquad \mu_t \sim \mathcal{N}(0, r_t), \quad \text{Measurement Equation}$$

- KF is a classical algorithm to perform online optimal inference for this model.
- Specifically, given a sequence of keys and values observed, KF computes the MAP estimate for s_t.

$$\hat{s}_t = \arg\max_s P\Big(s \mid \{k_1, v_1\}, \ldots, \{k_t, v_t\}\Big)$$

Kalman Filter (KF) recursion

➢ At a high-level the KF recursion can be understood as follows.

$$\hat{s}_t = \underbrace{A_t \hat{s}_{t-1} + B_t u_t}_{\text{Predicted state}} + G_t(\overbrace{v_t \quad - k_t^\top \underbrace{\left[A_t \hat{s}_{t-1} + B_t u_t\right]}_{\text{Predicted state}}}^{\text{Innovation}}),$$

- ❖ Predicted state is the state transition equation applied to $\hat{s}_{t-1}$.
- ❖ G_t is the Kalman Gain which accounts for feedback from the true measurement v_t and predicted measurement.
  - Kalman Gain depends on the whole history via the error covariance → uncertainty in state estimate based on key-value pairs observed so far.

Existing SSMs = Approximate Kalman Filters

- DeltaNet assumes steady-state model,

$$
\begin{aligned}
s_t &= s_{t-1} \\
v_{t,i} &= k_t^\top s_t + \mu_t, \qquad \mu_t \sim \mathcal{N}(0, r_t),
\end{aligned} \tag{15}
$$

where $A_t = I_n$, $B_t = 0$, and $w_t = 0$ (i.e., no state evolution, no control input, and no process noise).

- DeltaNet state update approximates the Kalman Gain by assuming identity error covariance matrix.
  - Avoids tracking uncertainty in state over time.

Existing SSMs = Approximate Kalman Filters

- Gated DeltaNet (GDN) assumes a simplified fading memory model,

$$s_t = \alpha_t s_{t-1} + w_t \qquad w_t \sim \mathcal{N}(0, I_n)$$

$$v_{t,i} = k_t^\top s_t + \mu_t, \qquad \mu_t \sim \mathcal{N}(0, r_t),$$

- Like DeltaNet, GDN's state update also approximates the Kalman Gain by assuming identity error covariance matrix.
- Kimi Delta Attention, extends GDN by using channel-specific decay factors $\alpha_{t,i}$ instead of a global α_t.
  - Still assumes identity error covariance matrix!

GKA: Exact Kalman Filter for the Steady-State model

- The KF recursion in its most general form is not amenable to parallelization

- In GKA, we assume the same steady-state model as DeltaNet, but implement **the exact KF recursion.**
  - The Kalman Gain accounts for the full history via tracking the exact error covariance matrix.

Connections with MesaNet

- MesaNet[1] also proposes to solve a WRR objective to update the state

$$\hat{\Phi}_t^{\text{mesa}} = \arg\min_{\Phi} \mathcal{L}_t(\Phi), \qquad \textbf{with} \qquad \mathcal{L}_t(\Phi) = \frac{1}{2}\sum_{t'=1}^{t} ||v_{t'} - \Phi k_{t'}||^2 + \frac{\mathrm{Tr}(\Phi^\top \Lambda \Phi)}{2}.$$

- Key Difference 1:
  - Learns a time-independent regularizer Λ from data,
    - Can result in training instabilities as the condition number is not controlled.
      - We explicitly control for it with adaptive regularization $\lambda_t = a||H||_F$.

1. von Oswald, Johannes, et al. "MesaNet: Sequence Modeling by Locally Optimal Test-Time Training." arXiv preprint arXiv:2506.05233 (2025).

Connections with MesaNet

- MesaNet[1] also proposes to solve a WRR objective to update the state

$$\hat{\Phi}_t^{\text{mesa}} = \arg\min_{\Phi} \mathcal{L}_t(\Phi), \qquad \text{with} \qquad \mathcal{L}_t(\Phi) = \frac{1}{2}\sum_{t'=1}^{t} ||v_{t'} - \Phi k_{t'}||^2 + \frac{\mathrm{Tr}(\Phi^\top \Lambda \Phi)}{2}.$$

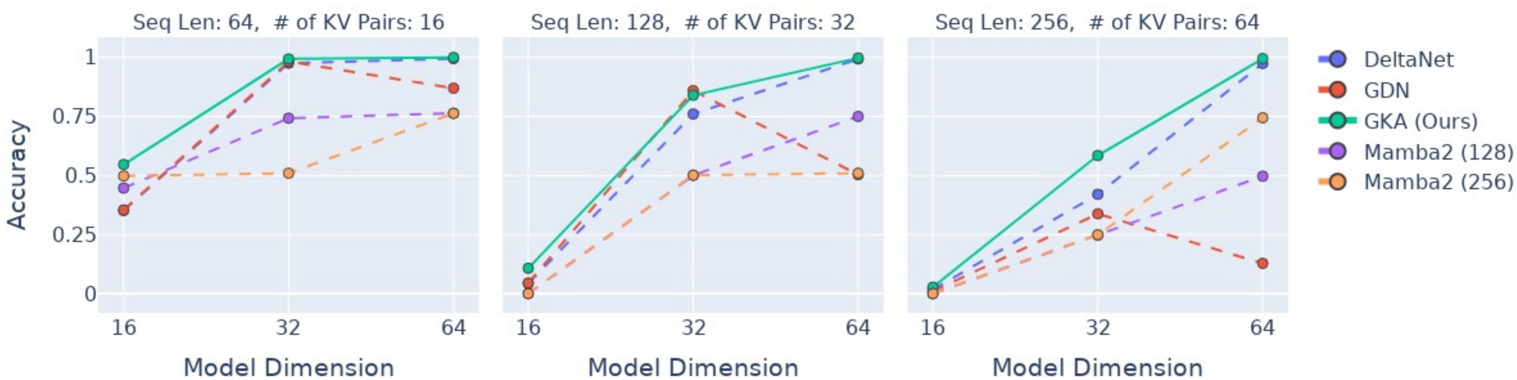
- Key Difference 2:
  - Employs Conjugate Gradient (CG) as the iterative solver for parallel training.
    - CG is unstable in low-precision environments and leads to **erroneous** gradients.
      - We employ Chebyshev Iteration (CH) which we show is more stable and has “exact” gradients.

1. von Oswald, Johannes, et al. "MesaNet: Sequence Modeling by Locally Optimal Test-Time Training." arXiv preprint arXiv:2506.05233 (2025).

Experiments

Multi-Query Associative Recall Tasks

➢ GKA outperforms other fading memory layers on challenging synthetic recall tasks.



GKA scaling laws

Table 5. **GKA shows stronger scaling with compute that other SSM baseline models.** LM-Harness results for models at different scales: 440M, 1B and 2.8B. All models were trained from scratch. 440M and 1B models were trained on 8B and 20B tokens respectively in accordance to the Chinchila scaling laws [24]. For the 2.8B model we trained on 100B tokens.

Model	ARC-C acc_n ↑	ARC-E acc_n ↑	BoolQ acc ↑	COPA acc ↑	HellaSWAG acc_n ↑	PIQA acc_n ↑	SciQ acc_n ↑	Winogrande acc ↑	FDA contains ↑	SWDE contains ↑	Avg
440M Models											
Transformer	24.40	<u>42.26</u>	<u>59.88</u>	70.00	36.19	64.15	61.50	**51.70**	**5.17**	**35.64**	**45.09**
Gated Linear Attention	24.06	40.28	56.57	<u>71.00</u>	32.70	62.24	57.80	50.67	1.00	9.18	40.55
Gated DeltaNet	**25.17**	41.96	58.23	**72.00**	36.96	**64.69**	<u>63.6</u>	**51.7**	1.91	11.88	<u>42.81</u>
DeltaNet	<u>25.09</u>	41.92	**61.13**	65.00	<u>37.20</u>	<u>64.47</u>	**64.00**	49.49	<u>2.81</u>	<u>14.31</u>	42.54
Gated KalmaNet (Ours)	24.57	**43.22**	56.94	<u>71.00</u>	**37.22**	<u>64.47</u>	62.8	50.83	1.45	14.04	42.65

GKA scaling laws

Table 5. **GKA shows stronger scaling with compute that other SSM baseline models.** LM-Harness results for models at different scales: 440M, 1B and 2.8B. All models were trained from scratch. 440M and 1B models were trained on 8B and 20B tokens respectively in accordance to the Chinchila scaling laws [24]. For the 2.8B model we trained on 100B tokens.

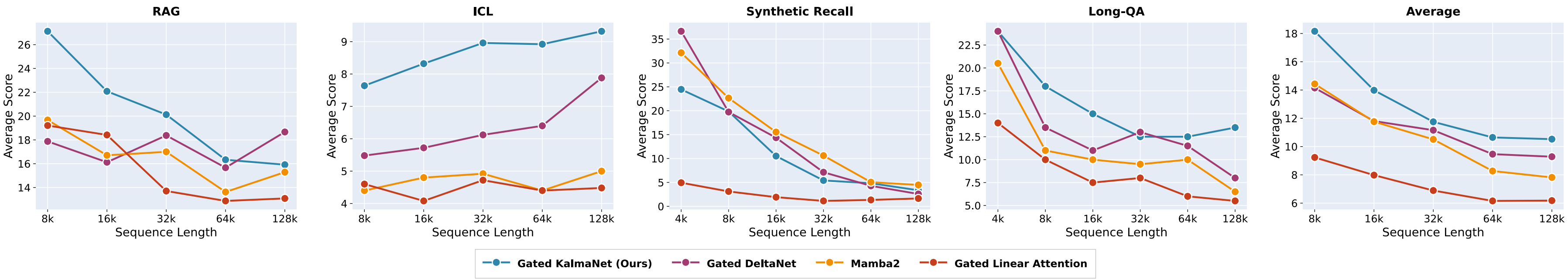
Model	**ARC-C** acc_n ↑	**ARC-E** acc_n ↑	**BoolQ** acc ↑	**COPA** acc ↑	**HellaSWAG** acc_n ↑	**PIQA** acc_n ↑	**SciQ** acc_n ↑	**Winogrande** acc ↑	**FDA** contains ↑	**SWDE** contains ↑	**Avg**
					440M Models						
Transformer	24.40	42.26	59.88	70.00	36.19	64.15	61.50	**51.70**	**5.17**	**35.64**	**45.09**
Gated Linear Attention	24.06	40.28	56.57	71.00	32.70	62.24	57.80	50.67	1.00	9.18	40.55
Gated DeltaNet	**25.17**	41.96	58.23	**72.00**	36.96	**64.69**	63.6	**51.7**	1.91	11.88	42.81
DeltaNet	25.09	41.92	**61.13**	65.00	37.20	64.47	**64.00**	49.49	2.81	14.31	42.54
Gated KalmaNet (Ours)	24.57	**43.22**	56.94	71.00	**37.22**	64.47	62.8	50.83	1.45	14.04	42.65
					1B Models						
Transformer	26.62	46.42	59.94	**77.00**	44.01	67.14	**68.30**	54.06	**8.35**	**45.18**	**49.70**
Mamba2	**28.07**	46.63	60.21	70.00	44.57	67.57	65.50	54.30	1.45	15.75	45.40
Gated Linear Attention	25.94	42.00	58.84	70.00	36.34	63.60	58.20	51.85	1.45	10.53	41.88
Gated DeltaNet	27.05	**47.98**	59.54	74.00	44.27	67.36	66.2	53.83	2.18	17.82	46.02
DeltaNet	27.56	46.25	59.97	71.00	43.18	67.74	65.90	**55.41**	3.09	20.61	46.07
Gated KalmaNet (Ours)	25.43	46.55	**60.73**	74.00	**44.59**	**68.88**	67.60	52.41	6.17	21.87	46.82

GKA scaling laws

Table 5. **GKA shows stronger scaling with compute that other SSM baseline models.** LM-Harness results for models at different scales: 440M, 1B and 2.8B. All models were trained from scratch. 440M and 1B models were trained on 8B and 20B tokens respectively in accordance to the Chinchila scaling laws [24]. For the 2.8B model we trained on 100B tokens.

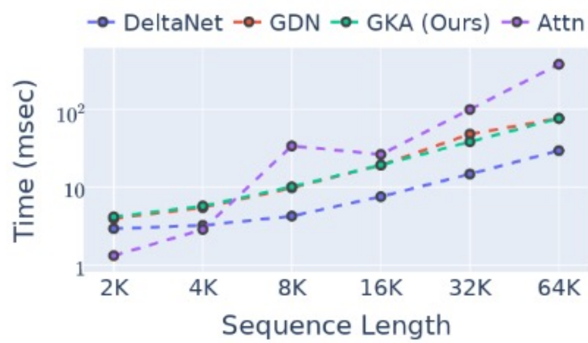
Model	ARC-C acc_n ↑	ARC-E acc_n ↑	BoolQ acc ↑	COPA acc ↑	HellaSWAG acc_n ↑	PIQA acc_n ↑	SciQ acc_n ↑	Winogrande acc ↑	FDA contains ↑	SWDE contains ↑	Avg
440M Models											
Transformer	24.40	42.26	59.88	70.00	36.19	64.15	61.50	**51.70**	**5.17**	**35.64**	**45.09**
Gated Linear Attention	24.06	40.28	56.57	71.00	32.70	62.24	57.80	50.67	1.00	9.18	40.55
Gated DeltaNet	**25.17**	41.96	58.23	**72.00**	36.96	**64.69**	63.6	**51.7**	1.91	11.88	42.81
DeltaNet	25.09	41.92	**61.13**	65.00	37.20	64.47	**64.00**	49.49	2.81	14.31	42.54
Gated KalmaNet (Ours)	24.57	**43.22**	56.94	71.00	**37.22**	64.47	62.8	50.83	1.45	14.04	42.65
1B Models											
Transformer	26.62	46.42	59.94	**77.00**	44.01	67.14	**68.30**	54.06	**8.35**	**45.18**	**49.70**
Mamba2	**28.07**	46.63	60.21	70.00	44.57	67.57	65.50	54.30	1.45	15.75	45.40
Gated Linear Attention	25.94	42.00	58.84	70.00	36.34	63.60	58.20	51.85	1.45	10.53	41.88
Gated DeltaNet	27.05	**47.98**	59.54	74.00	44.27	67.36	66.2	53.83	2.18	17.82	46.02
DeltaNet	27.56	46.25	59.97	71.00	43.18	67.74	65.90	**55.41**	3.09	20.61	46.07
Gated KalmaNet (Ours)	25.43	46.55	**60.73**	74.00	**44.59**	**68.88**	67.60	52.41	6.17	21.87	46.82
2.8B Models											
Transformer	32.25	56.10	**64.28**	80.00	60.96	73.56	79.50	61.72	**58.53**	**72.28**	**63.92**
Mamba2	32.24	59.64	58.72	82.00	62.23	73.78	79.80	62.19	7.71	41.13	55.94
Gated Linear Attention	27.82	50.80	52.57	78.00	48.83	70.13	69.60	54.54	2.81	20.43	47.55
Gated DeltaNet	32.59	**60.02**	62.75	82.00	62.8	74.32	80.6	62.35	8.26	44.28	57.00
DeltaNet	**32.85**	58.16	42.51	81.00	61.13	73.78	43.90	61.72	11.80	46.08	51.29
Gated KalmaNet (Ours)	32.51	59.89	61.68	**85.00**	**63.84**	**74.81**	**83.2**	**64.17**	12.89	50.95	58.89

Long context performance of GKA



- ➢ GKA shows strong RAG and Long-QA capabilities.
  - ❖ Outperforms all fading memory baselines on average.

GKA as comparable runtime with existing SSMs



(b) Runtime of a single memory layer

- GKA has linear time-complexity with sequence length.
- Comparable to GDN in (forward+backward) pass.
- Our parallel Triton implementation of GKA is fast.

Next steps

- We presented a fading memory layer that takes the entire past into account and implemented it efficiently on hardware.

- Although it improves over existing SSM layers, gap with Attention exists, especially on recall-oriented tasks.
  - This is inevitable for any compression-based memory layer.
    - What is relevant is not known a priori.

- It's possible to combine GKA with Attention to build Hybrid models (eidetic + fading) or Hybrid layers (à la B'MOJO).

Coming Soon: Hybrid Model Library

- Comprehensive library for efficiently training Hybrid models at scale (large # parameters + long sequences)
- Hybridize pre-trained Transformers → Hybrid variants (Mamba2, GDN, KDA, ...)
- GKA kernels and GKA-based models will be released too.

Foster new cutting-edge research in Hybrid models

… stay tuned!!!

Thank you!