Efficient Language Models Beyond Transformers

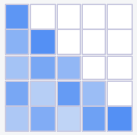
Arshia Afzal

Aviv Bick

A joint work with
Eric Xing,
Volkan Cevher,
Albert Gu

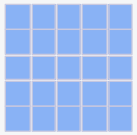
Outline

Linear models don't fail from lack of memory — they fail from lack of organization.



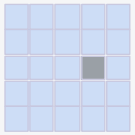
Sequence Modeling

Mixing tokens
along the time axis



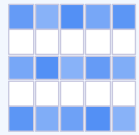
Linear & SSMs

Constant state,
linear time



The Gap

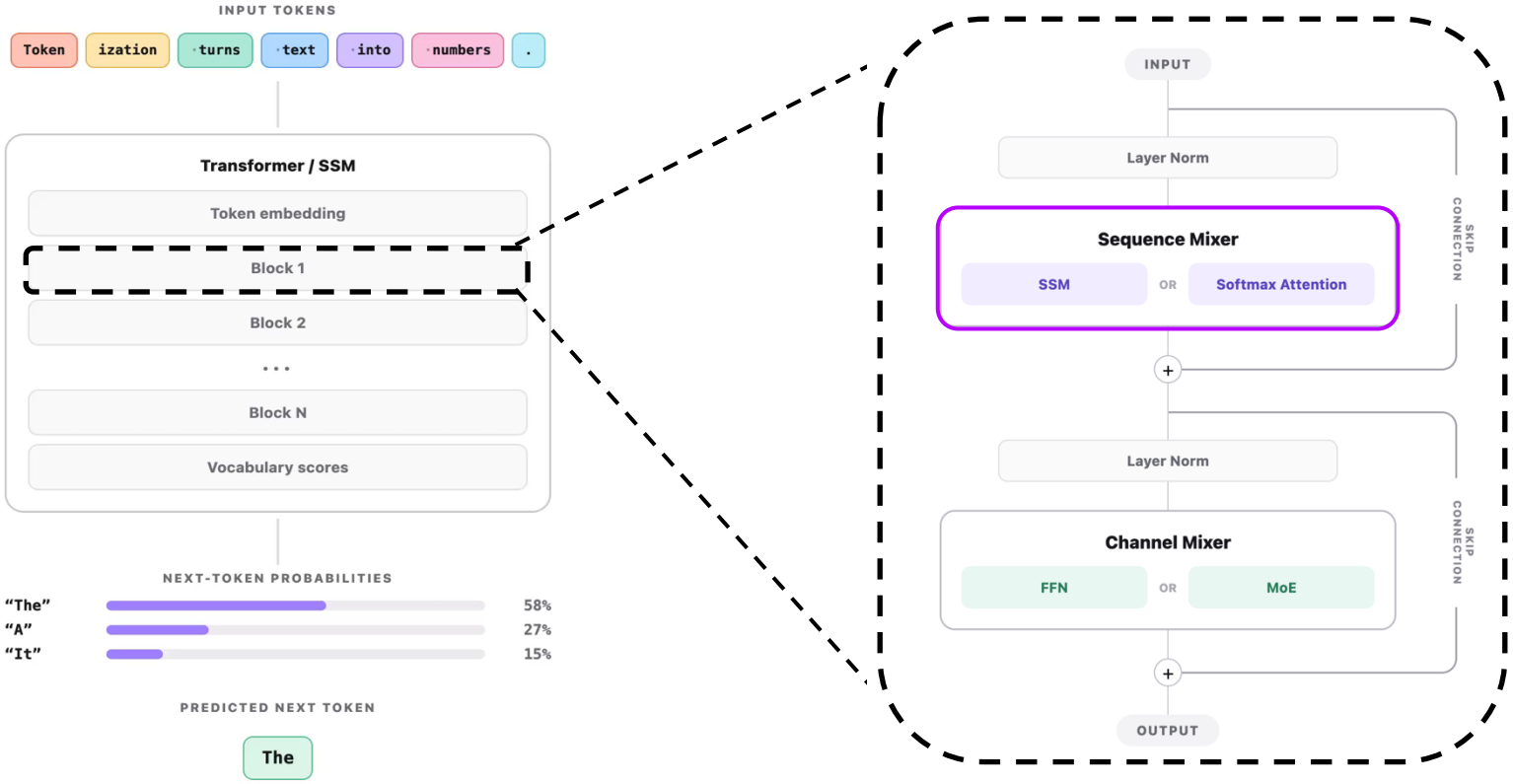
Memory without
organization



Raven

Slots that
persist

Language Modelling Overview

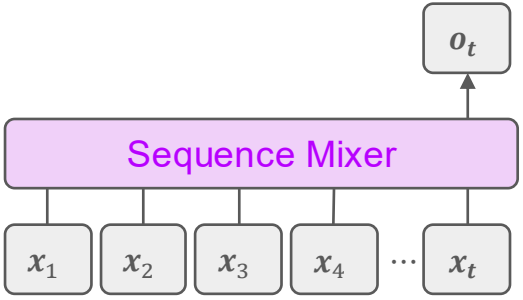


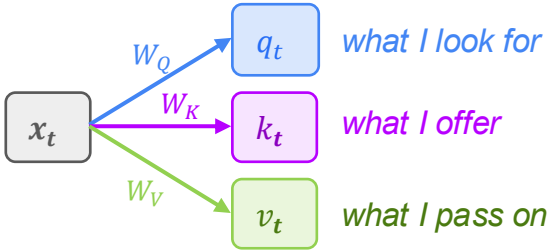
Sequence Mixer

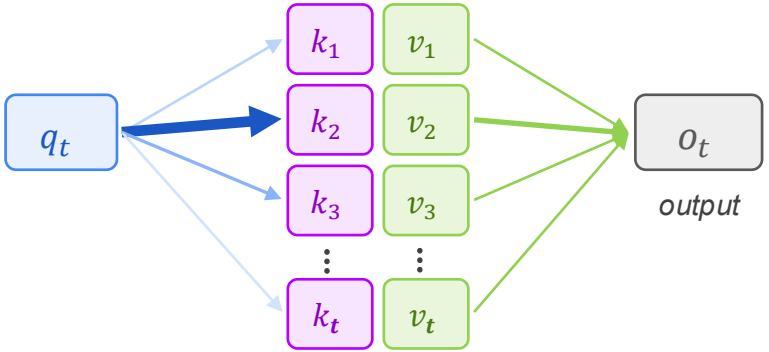
A Sequence Mixer mixes tokens along the time axis.

- Given the past we predict the future
- We use a query q_t, key k_t and value v_t
- We compute the interaction:

$$o_t = v_1 \cdot sim(q_t, k_1) + \cdots + v_t \cdot sim(q_t, k_t)$$







thicker line = higher attention weight

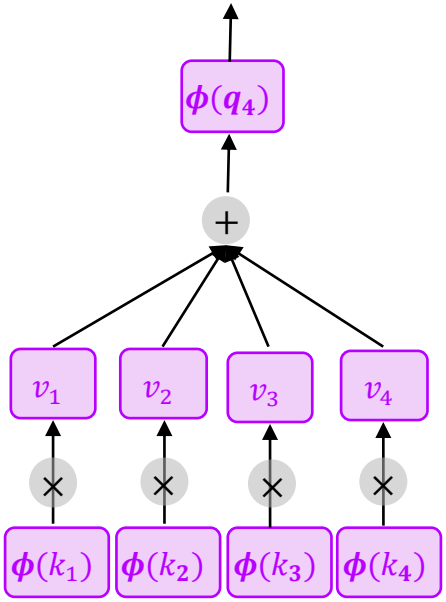
Linear Sequence Mixer

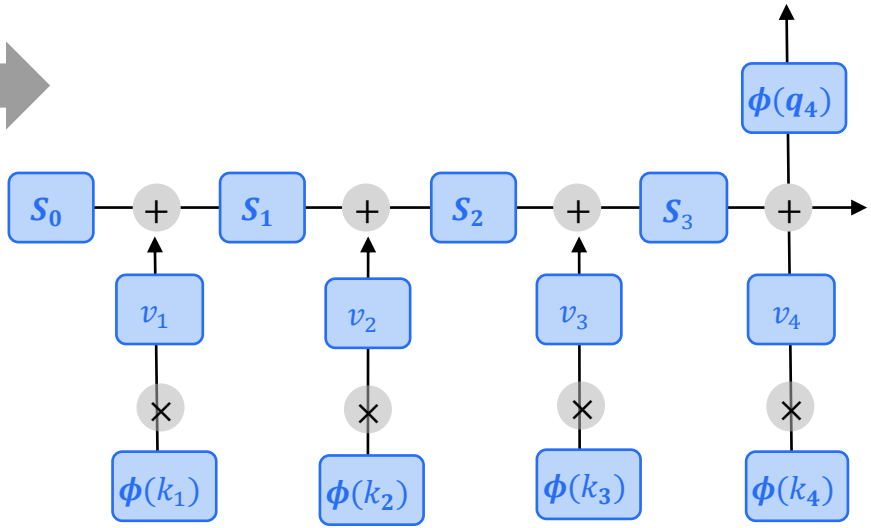
Assume $sim(q_t, k_1) = \phi(k_1)^T \phi(q_t)$:

$$o_t = v_1 \phi(k_1)^T \phi(q_t) + \cdots + v_t \phi(k_t)^T \phi(q_t)$$

$$o_t = v_1 \phi(k_1)^T \phi(q_t) + \cdots + v_t \phi(k_t)^T \phi(q_t)$$

$$o_t = \left(v_1 \phi(k_1)^T + \cdots + v_t \phi(k_t)^T\right) \phi(q_t)$$

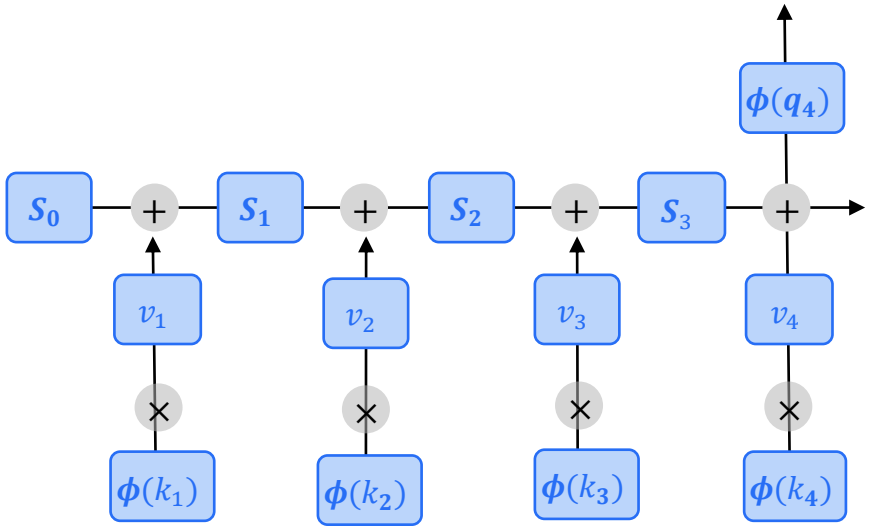




Linear Sequence Mixer

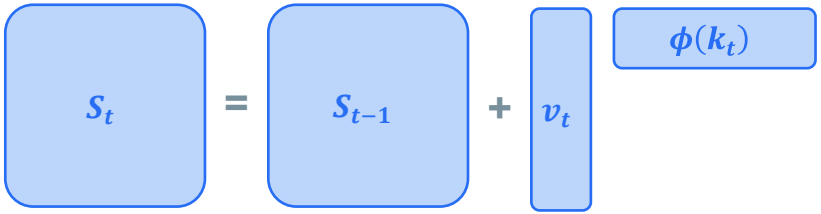
Recurrence

$$S_t = S_{t-1} + v_t \, \phi(k_t)$$



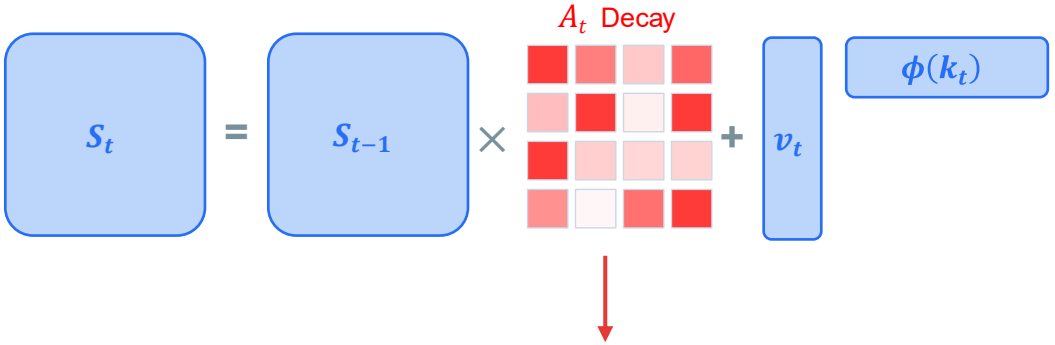
Linear Sequence Mixer

Recurrence



One recurrence, different decays

Recurrence

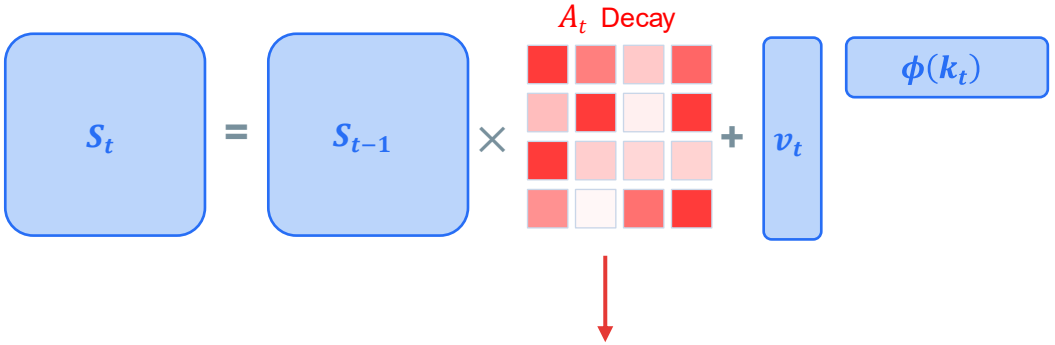


What A_t brings

- Removing historical information from hidden state.
- Introducing relative distance $D_{ij} = (A_i \cdots A_j)$ between tokens.
- Important for SSMs to be able to do language modeling.

One recurrence, different decays

Recurrence

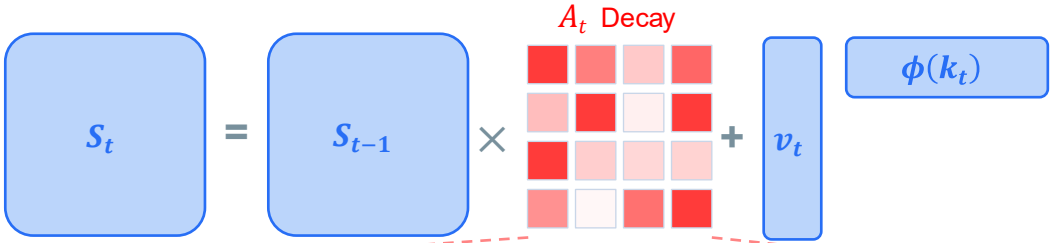


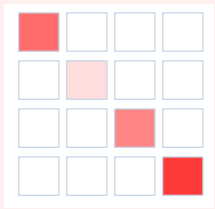
Why A_t can't be dense

- Dense $d \times d$ decay = a matmul every step — $O(d^2)$ per token, and the linear-time win is gone
- It breaks the parallel scan: dense A_t don't commute, so training falls back to sequential
- Repeated dense products explode or vanish over long contexts

One recurrence, different decays

Recurrence

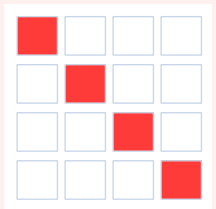




$A_t = \exp(-Wx_t)$

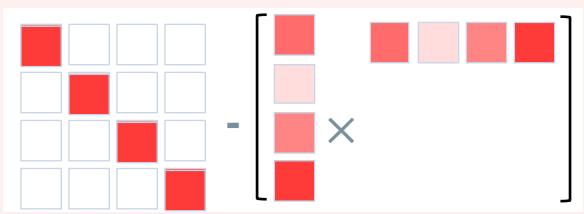
$A_t = \sigma(Wx_t)$

Mamba/GLA



$A_t = \exp(-wx_t)$

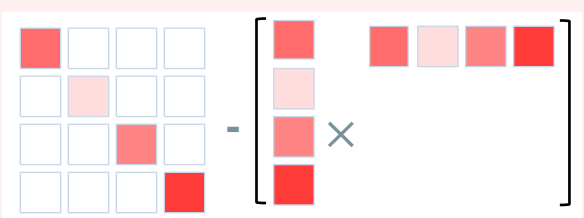
Mamba-2



$A_t = a_t(I - k_t k_t^T)$

$a_t = \exp(-wx_t)$

GDN



$A_t = D_t(I - k_t k_t^T)$

$D_t = diag(\exp(-Wx_t))$

KDA/K3

What SSMs can (not) do?

KV Retrieval **key given AFTER** the dictionary

```
m4q:812   r9v:604   t2m:118   q7d:953   h5n:271
w3c:486   bk7:359   x8j:730   a1f:295   n6p:842
c4t:167   v2k:508   z1x:447   y3g:451   k3y:562
Question: what is the value of  bk7 ?
```

Key arrives last — the model cannot know which entry matters, so it must retain the whole dictionary.

NIAH **key given BEFORE** the dictionary

```
Question: what is the value of  bk7 ?
m4q:812   r9v:604   t2m:118   q7d:953   h5n:271
w3c:486   bk7:359   x8j:730   a1f:295   n6p:842
c4t:167   v2k:508   z1x:447   y3g:451   k3y:562
```

Model reads the key first — it knows what to look for, so a fixed-size state can keep just that one entry.

NIAH in real life: two hard cases

Jump TRADING

1 Honoring a constraint 200 turns later

You Never write to prod — use the staging DB.

⋮ *200 turns of tool calls*

Agent Read prod metrics (allowed)

Agent Looked up prod_url

⋮

Agent Running migration on **prod**

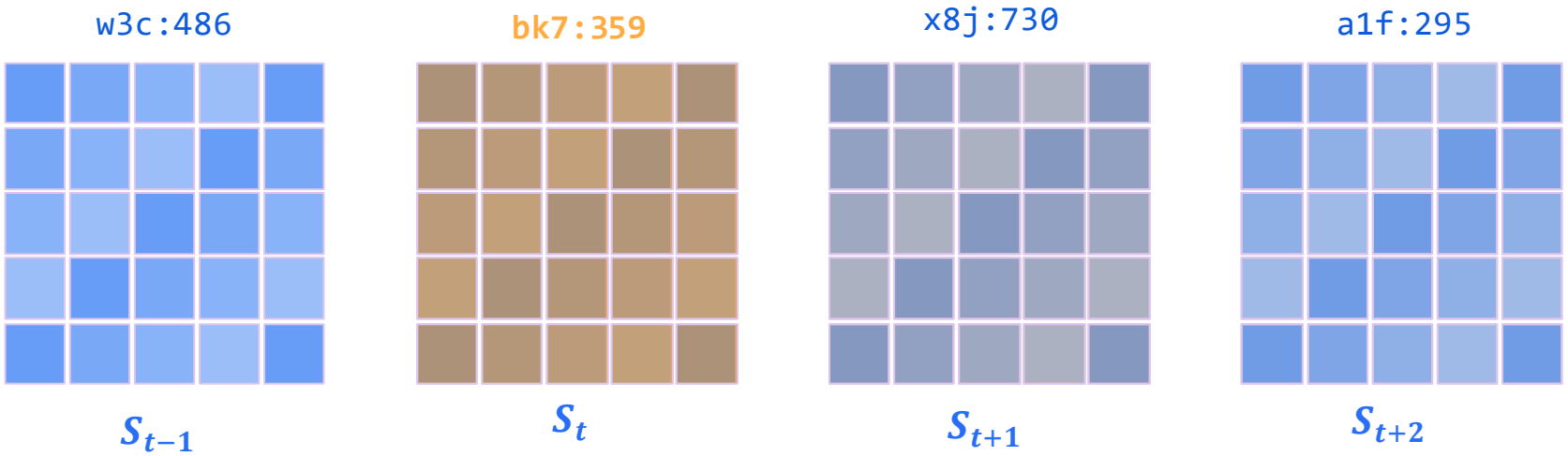
2 Reusing a variable defined 9k lines ago

```
38  canvas_scale = width / 3
⋮
    # 9,000 lines of similar names
    scale_cache = {}
    tmp_scale = ...
    def get_scale(): ...
⋮
    # agent writes next line
9120 render(canvas_scale)
```

NIAH: capacity is not the bottleneck

Most SSMs fail NIAH even though they have enough memory

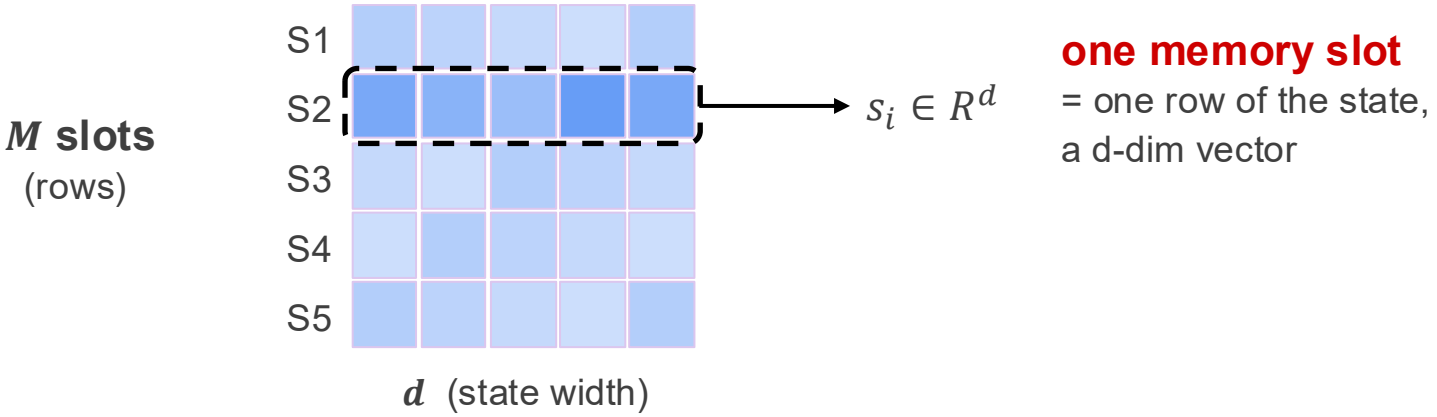
This is due to lack of organization in the state – they lack persistence



A Step Towards Persistence

Memory a set of **slots**

- This **structures** the state $S_t \in R^{M \times d}$ to sub-units.
- Can be written to and read from independently of the other rows.

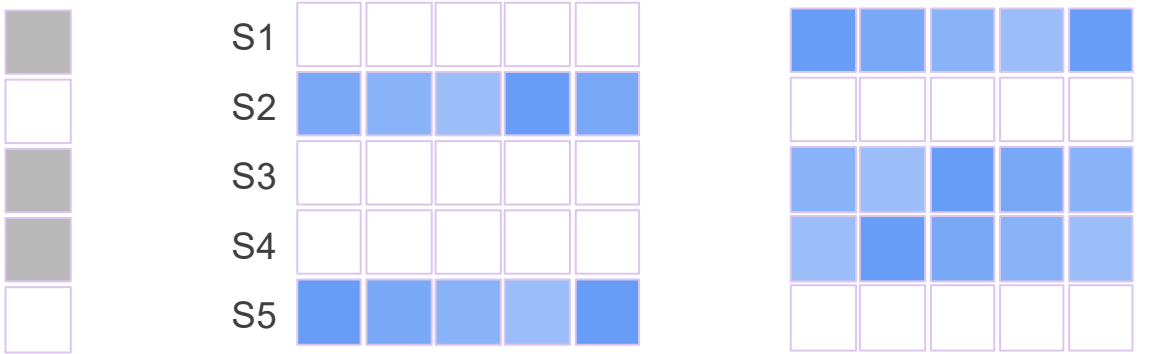


A Step Towards Persistence

Routing Slot Memories (RSMs):

- Selected slots (r_t) to be written
- Non-selected slots $(1 - r_t)$ remain unchanged – therefore persist

$$S_t = (1 - r_t) \odot S_{t-1} + r_t \odot (S_{t-1} A_t + v_t k_t^T)$$



Router r_t | Persistent (unchanged) | Updated (slot decayed)

Sliding Window Attention (SWA)

SWA partly solves the persistence problem

Attention, restricted to the last W tokens.

Let $\bar{t} = t \bmod W$, this can be written as a recurrence:

$r_{\bar{t}} =$ 

$$S_t^K = \underbrace{(1 - r_{\bar{t}}) \odot S_{t-1}^K}_{delete\ row\ \bar{t}} + \underbrace{r_{\bar{t}} k_t^T}_{new} \qquad S_t^V = \underbrace{(1 - r_{\bar{t}}) \odot S_{t-1}^V}_{delete\ row\ \bar{t}} + \underbrace{r_{\bar{t}} v_t^T}_{new}$$

$$o_t = (S_t^V)^T softmax(S_t^K q_t)$$

Inside the window: recall is exact. Beyond it: the token is gone, not faded.

Raven

A linear model that stores context wisely, just as a raven stores its walnuts!

Raven

SWA

$$S_t^K = (1 - e^{r_{\bar{t}}}) \odot S_{t-1}^K + e^{r_{\bar{t}}} {k_t}^T$$
$$S_t^V = (1 - e^{r_{\bar{t}}}) \odot S_{t-1}^V + e^{r_{\bar{t}}} {v_t}^T$$
$$o_t = (S_t^V)^T softmax(S_t^K q_t)$$

Raven

$$S_t^k = (1 - e^{\alpha_t r_t}) \odot S_{t-1}^k + e^{\alpha_t r_t} k_t^T$$
$$S_t^v = (1 - e^{\alpha_t r_t}) \odot S_{t-1}^v + e^{\alpha_t r_t} v_t^T$$
$$o_t = (S_t^V)^T softmax(S_t^K q_t)$$

Mamba-2

$$S_t = e^{\alpha_t} \cdot S_{t-1} + v_t k_t^T$$
$$o_t = S_t q_t$$

Raven

SWA

$$S_t^K = (1 - e^{r_{\bar{t}}}) \odot S_{t-1}^K + e^{r_{\bar{t}}} {k_t}^T$$
$$S_t^V = (1 - e^{r_{\bar{t}}}) \odot S_{t-1}^V + e^{r_{\bar{t}}} {v_t}^T$$
$$o_t = (S_t^V)^T softmax(S_t^K q_t)$$

$$r_{\bar{t}} = \log e_{\bar{t}}$$

Raven

$$S_t^k = (1 - e^{\alpha_t r_t}) \odot S_{t-1}^k + e^{\alpha_t r_t} k_t^T$$
$$S_t^v = (1 - e^{\alpha_t r_t}) \odot S_{t-1}^v + e^{\alpha_t r_t} v_t^T$$
$$o_t = (S_t^V)^T softmax(S_t^K q_t)$$

$$r_t = \frac{g_t}{\sum_i^M g_t[i]},$$
$$g_t = TopK\big(\sigma(W x_t)\big)$$

Mamba-2

$$S_t = e^{\alpha_t} \cdot S_{t-1} + v_t k_t^T$$
$$o_t = S_t q_t$$

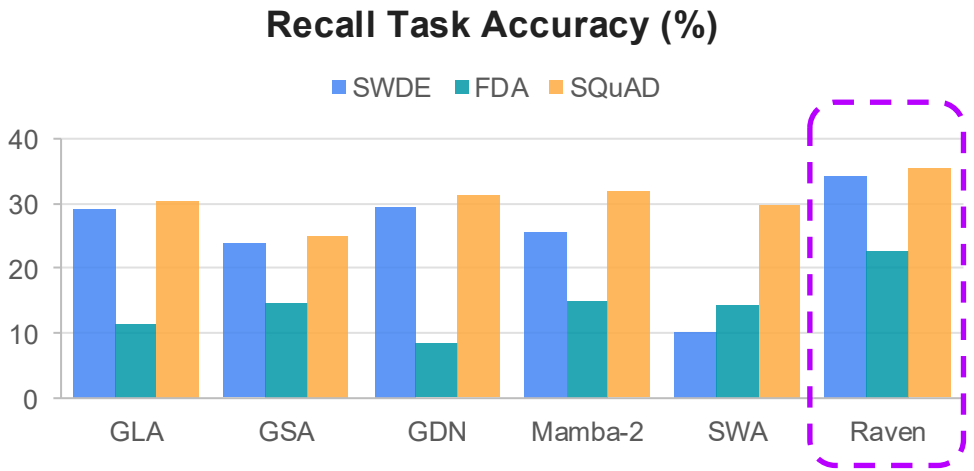
$$\alpha_t = -softplus(w^T x_t) e^{\Delta}$$

Spectrum of RSMs

Model	Router (r_t)	Decay (A_t)	Readout (o_t)
LinAtt	1_M	I	$S_t q_t$
RetNet	1_M	γ	$S_t q_t$
GLA	1_M	$diag(\sigma(Wx_t))^{1/\tau}$	$S_t q_t$
Mamba-2	1_M	a_t	$S_t q_t$
GDN	1_M	$a_t(\mathrm{I} - k_t k_t^\top)$	$S_t q_t$
Raven	$g_t / 1^\top g_t$	I	$(S_t^v)^\top f(S_t^k q_t)$
GSA	$1_M - \sigma(Wx_t)^{1/\tau}$	I	$(S_t^v)^\top f(S_t^k q_t)$
ABC	$softmax(Wx_t)$	I	$(S_t^v)^\top f(S_t^k q_t)$
SWA	e_t	I	$(S_t^v)^\top f(S_t^k q_t)$

Recall-Heavy Results

- **Highest** NIAH Results on all tested scales.
- Extrapolation to **16x** beyond 2048 training sequence length.
- **Strong** recall heavy performance on FDA, SWDE, and SQuAD.



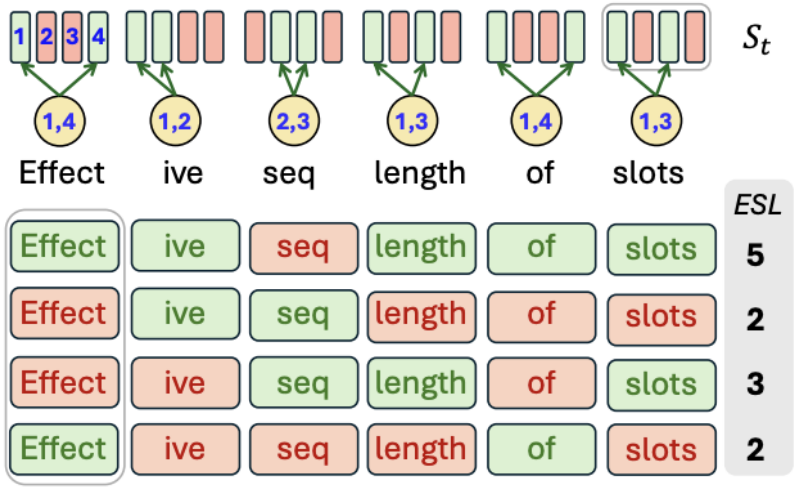
Model	Needle-In-The-Haystack (NIAH-1)					
	1K	2K	4K	8K	16K	32K
~400M / 15B tokens						
Transformer						
w. RoPE	**100**	**100**	0.0	0.0	0.0	0.0
w. Gate (FoX)	**100**	**100**	**32.2**	**8.0**	**4.2**	0.0
SSM						
GLA	74.6	25.1	8.2	2.2	0.0	0.0
GSA	99.2	97.1	90.0	67.4	29.6	11.0
GDN	99.2	**100**	**99.8**	92.0	41.8	22.1
Mamba-2	99.2	95.6	52.2	12.8	5.4	2.8
SWA	29.8	11.0	6.2	3.4	1.2	0.0
Raven	**99.8**	**100**	**99.8**	**99.8**	**99.4**	**91.4**

Raven's Memory Visualization

Effective Sequence Length

Different slots see different effective context lengths

- In NIAH, a recall slot only attends to the needle — an effective length of ~1, no matter the nominal context
- This gap between nominal and effective length is the intuition behind length generalization



Open Questions

- Can newer update rules (e.g., the delta rule) improve memory-slot writes?
- Can sparsity be exploited for faster, more efficient kernels?
- Can router-based models scale to frontier sizes?

@rshia_afz arshia.afzal@epfl.ch

@avivbick aviv@cmu.edu