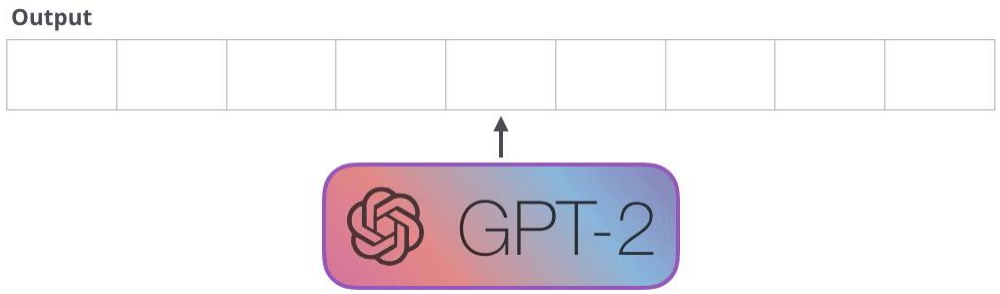
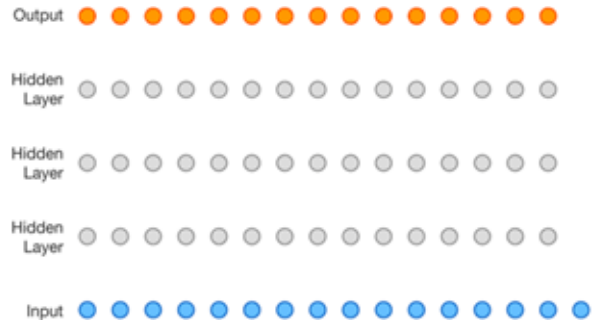
Context Beyond the Window: Architectural Paradigms for Long-Range Sequence Modeling

Albert Gu

MACHINE LEARNING DEPARTMENT

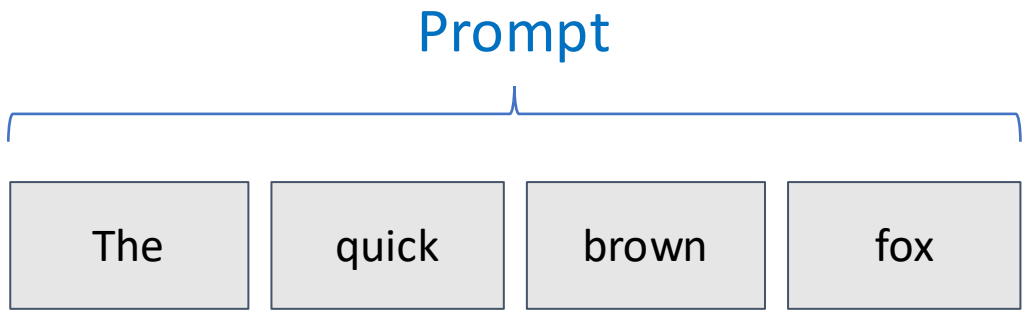
Autoregressive Modeling



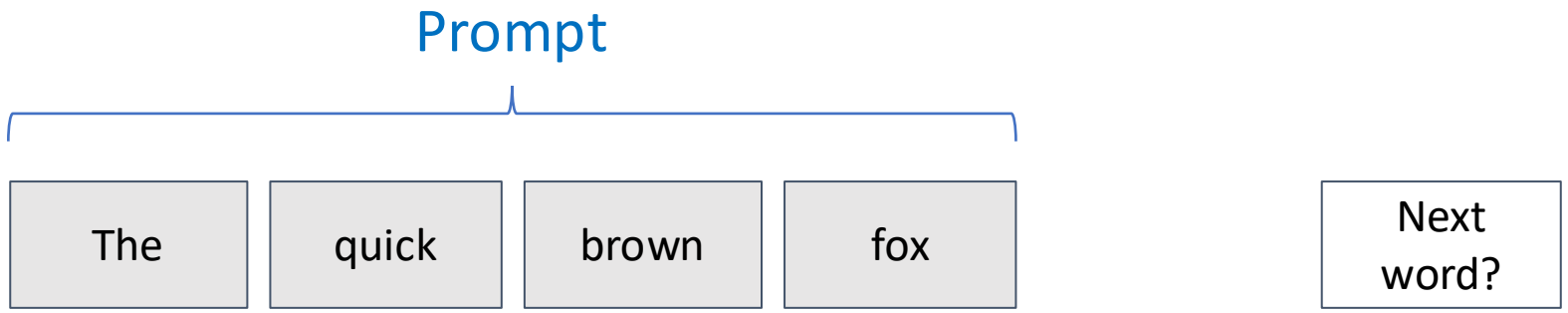


Training time: predict probabilities of next “token” conditioned on history
Inference time: Generate tokens one by one

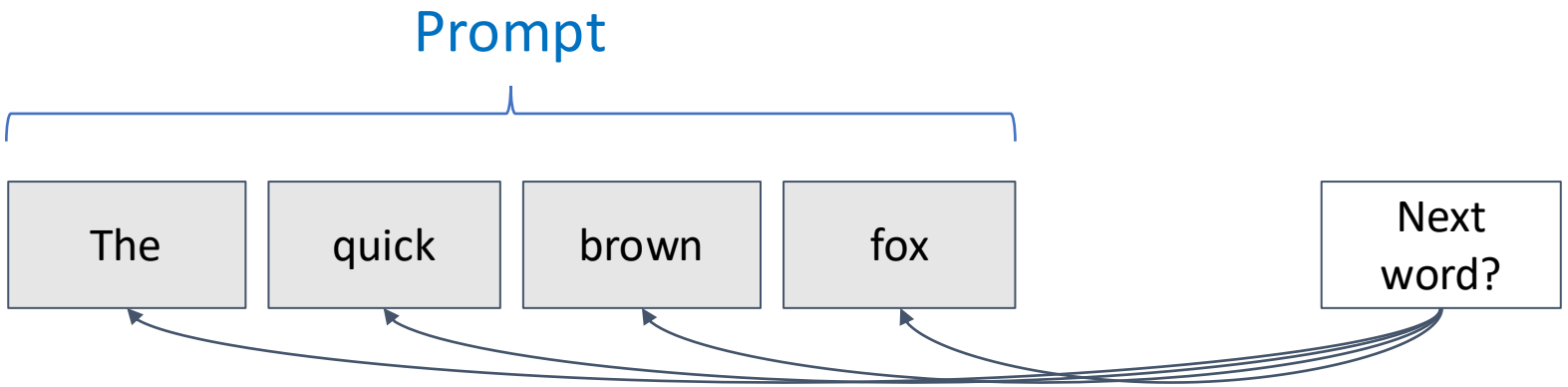
Recap: Attention Inference



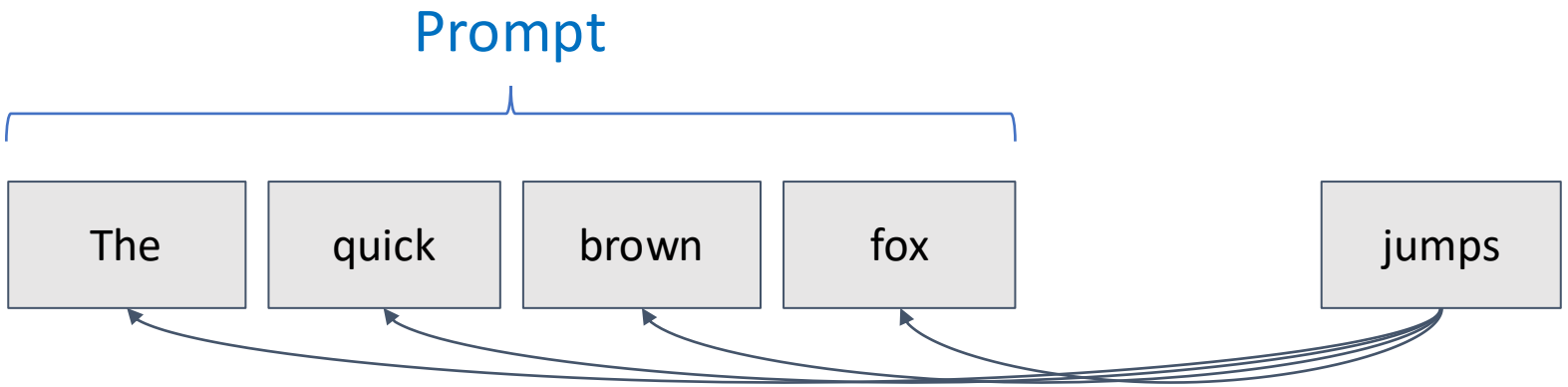
Recap: Attention Inference



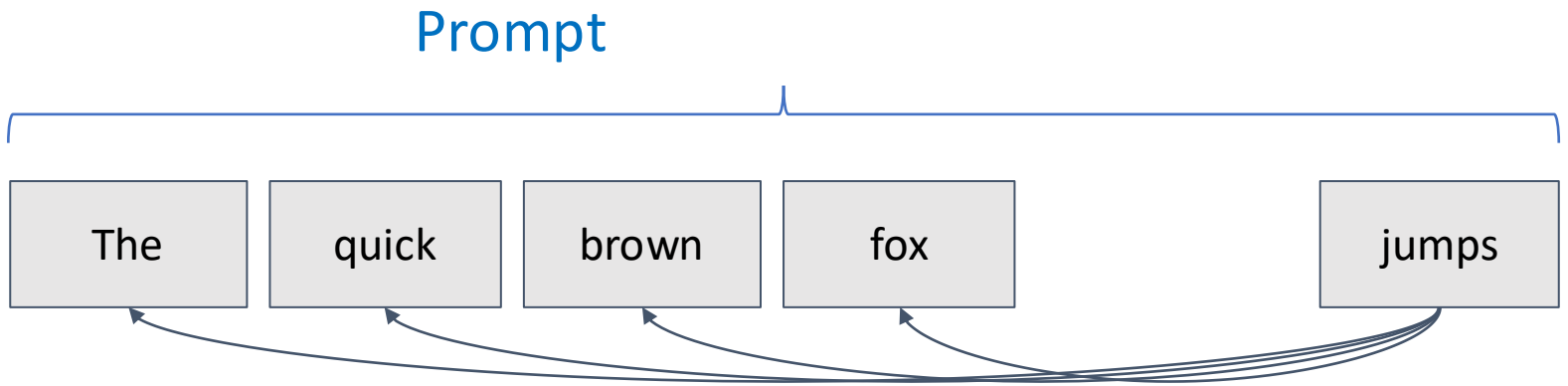
Recap: Attention Inference



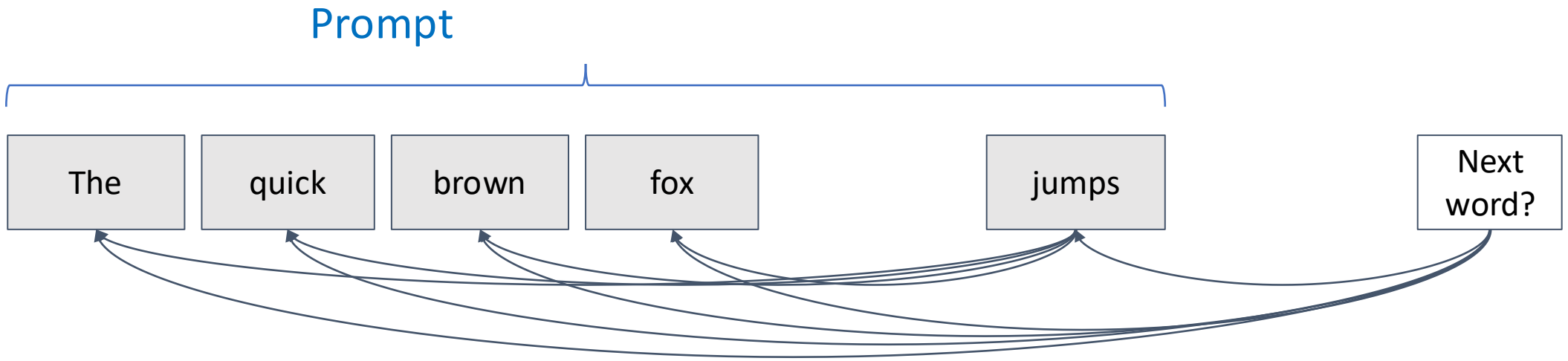
Recap: Attention Inference



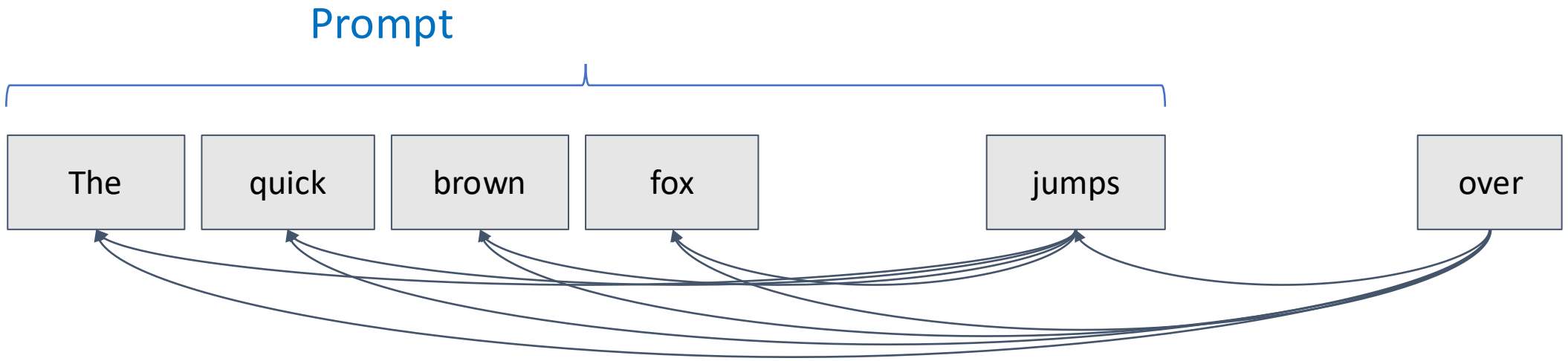
Recap: Attention Inference



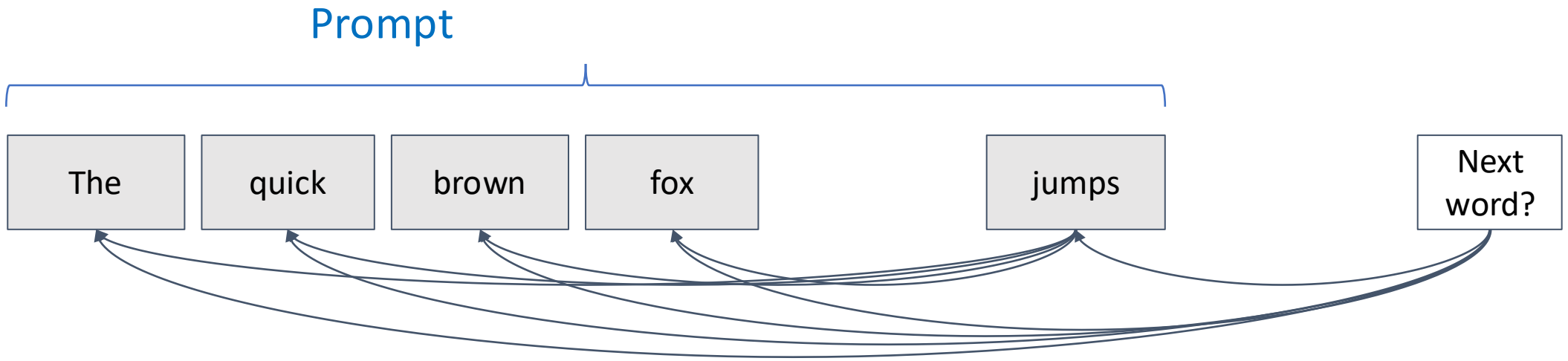
Recap: Attention Inference



Recap: Attention Inference



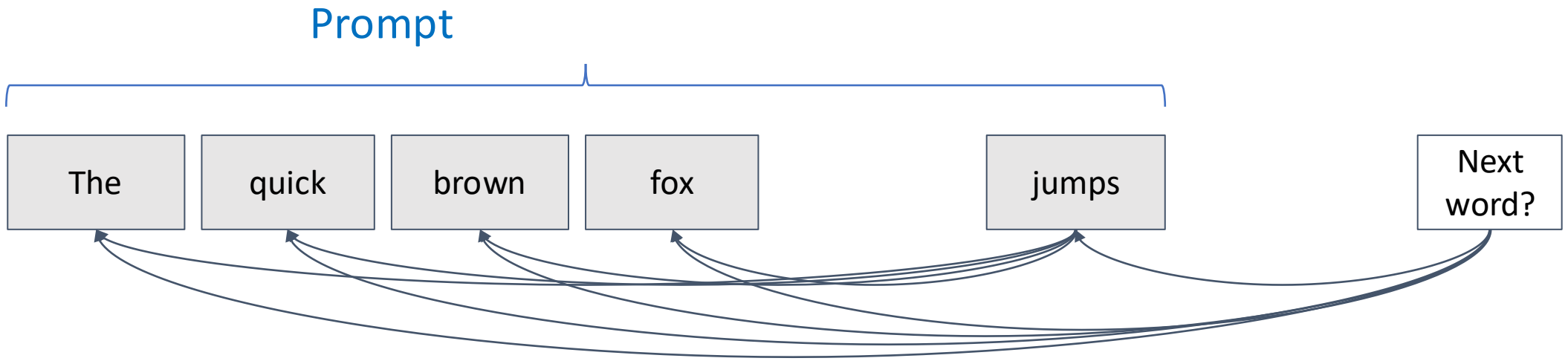
Recap: Attention Inference



In order to be able to attend over every past token, a Transformer must **cache every past token**!

The **"KV cache"** of Transformers is the largest reason for their **inefficiency at inference time**

Recap: Attention Inference

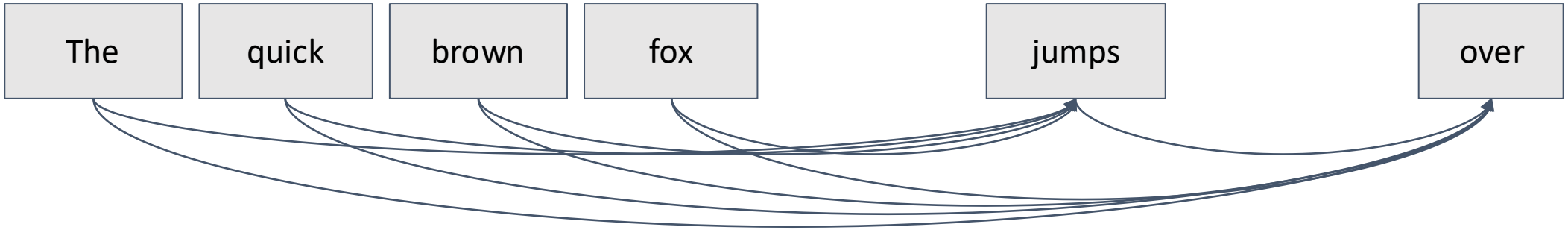


Memory & computation of Transformers increases with context!

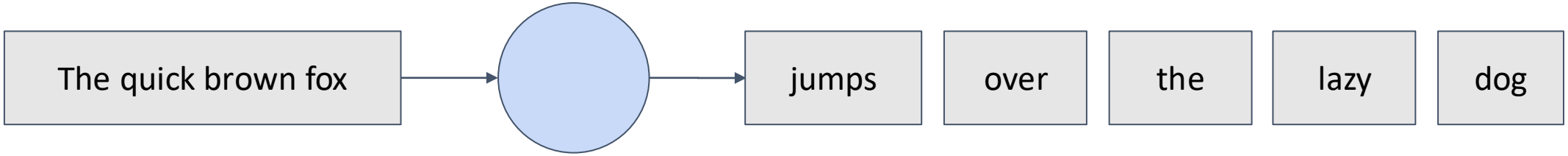
The **“KV cache”** of Transformers is the largest reason for their **inefficiency at inference time**

High Level Tradeoffs

Transformers generate **quadratically** by attending to every past token



In SSMs, tokens stream in, update the model's state and are discarded



State is highly compressed, enabling generation in **linear time**

Key Ingredients of SSMs

1. State size
2. State update
3. Efficiency

$$h_t = \boldsymbol{A_t} h_{t-1} + \boldsymbol{B_t} x_t$$

$$y_t = \boldsymbol{C_t} h_t$$

Key Ingredients of SSMs

1. **State size**
2. State update
3. Efficiency

How large is the autoregressive state? How much information can my model remember?

$$h_t = \boldsymbol{A_t} h_{t-1} + \boldsymbol{B_t} x_t$$
$$y_t = \boldsymbol{C_t} h_t$$

State Expansion

x: 1-dimensional input channel

h: N-dimensional hidden state

Key Ingredients of SSMs

1. State size
2. **State update**
3. Efficiency

How expressive is my state update function $\boldsymbol{h_t = f_\theta(h_{t-1}, x_t)}$? Can the state store exactly what information it needs to?

$$h_t = \boldsymbol{A_t} h_{t-1} + \boldsymbol{B_t} x_t$$
$$y_t = \boldsymbol{C_t} h_t$$

Selectivity / Gating
Input-dependent transitions

Key Ingredients of SSMs

1. State size
2. State update
3. **Efficiency**

Can I compute the forward pass reasonably fast during training?

$$h_t = \boldsymbol{A_t} h_{t-1} + \boldsymbol{B_t} x_t$$
$$y_t = \boldsymbol{C_t} h_t$$

Associative Scan

Linear recurrence is parallelizable

Advanced: rewrite as matmul

Key Ingredients of SSMs

1. State size
2. State update
3. Efficiency

Linear Attention (Katharopoulos et al.)
H3 (Fu, Dao et al.)

Classic RNNs - LSTM, GRU, etc.

Linear RNNs
S5 (Smith et al.), LRU (Orvieto et al.)

Mamba - "Selective" SSM

- Combination of these ingredients
- Demonstrated that recurrent models can be competitive with Transformers on language

Key Ingredients of SSMs

1. State size
2. State update
3. Efficiency

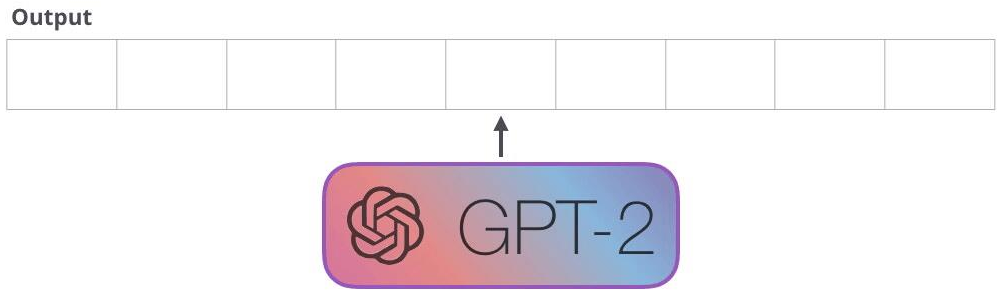
Model	Recurrence	Memory read-out
Linear Attention [40, 39]	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
+ Kernel	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \phi(\boldsymbol{q}_t)$
+ Normalization	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top, \ \ \boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \phi(\boldsymbol{k}_t)$	$\boldsymbol{o}_t = \mathbf{S}_t \phi(\boldsymbol{q}_t) / (\boldsymbol{z}_t^\top \phi(\boldsymbol{q}_t))$
DeltaNet [86]	$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
Gated RFA [69]	$\mathbf{S}_t = g_t \mathbf{S}_{t-1} + (1 - g_t) \boldsymbol{v}_t \boldsymbol{k}_t^\top, \ \ \boldsymbol{z}_t = g_t \boldsymbol{z}_{t-1} + (1 - g_t) \boldsymbol{k}_t$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t / (\boldsymbol{z}_t^\top \boldsymbol{q}_t)$
S4 [26]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \exp(-(\boldsymbol{\alpha} \mathbf{1}^\top) \odot \exp(\boldsymbol{A})) + \boldsymbol{B} \odot (\boldsymbol{v}_t \mathbf{1}^\top)$	$\boldsymbol{o}_t = (\mathbf{S}_t \odot \boldsymbol{C}) \mathbf{1} + \boldsymbol{d} \odot \boldsymbol{v}_t$
DFW [54, 41]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot (\boldsymbol{\beta}_t \boldsymbol{\alpha}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
RetNet [92, 76]	$\mathbf{S}_t = \gamma \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
Mamba [25]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \exp(-(\boldsymbol{\alpha}_t \mathbf{1}^\top) \odot \exp(\boldsymbol{A})) + (\boldsymbol{\alpha}_t \odot \boldsymbol{v}_t) \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t + \boldsymbol{d} \odot \boldsymbol{v}_t$
GLA [101]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot (\mathbf{1} \boldsymbol{\alpha}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top = \mathbf{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
RWKV-6 [68]	$\mathbf{S}_t = \mathbf{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = (\mathbf{S}_{t-1} + (\boldsymbol{d} \odot \boldsymbol{v}_t) \boldsymbol{k}_t^\top) \boldsymbol{q}_t$
HGRN-2 [79]	$\mathbf{S}_t = \mathbf{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t (\mathbf{1} - \boldsymbol{\alpha}_t)^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
mLSTM [7]	$\mathbf{S}_t = f_t \mathbf{S}_{t-1} + i_t \boldsymbol{v}_t \boldsymbol{k}_t^\top, \ \ \boldsymbol{z}_t = f_t \boldsymbol{z}_{t-1} + i_t \boldsymbol{k}_t$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t / \max\{1, \lvert \boldsymbol{z}_t^\top \boldsymbol{q}_t \rvert\}$
Mamba-2 [16]	$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$

Songlin Yang et al. “Parallelizing Linear Transformers with the Delta Rule over Sequence Length”

With these ingredients, most recurrent models are similar!
What are their **fundamental tradeoffs** vs. attention?

Key Concept: Autoregressive States

Tradeoffs of sequence models can be understood through examining their autoregressive state

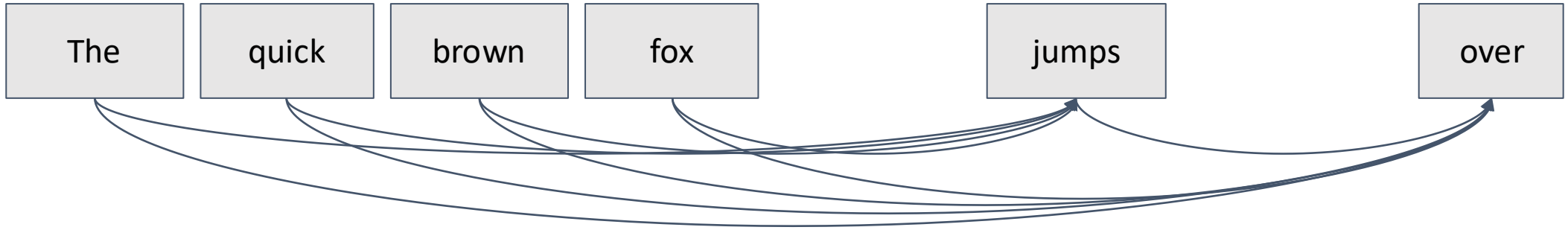


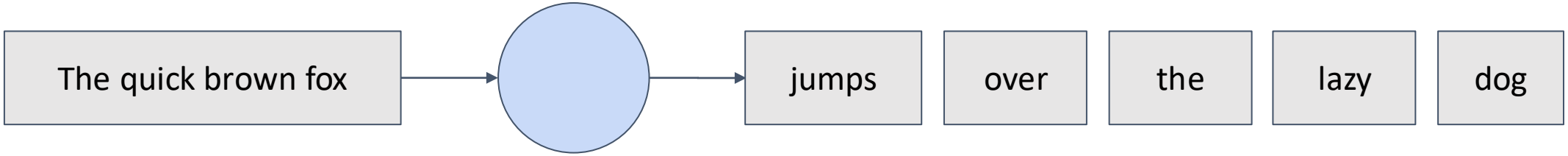
Every autoregressive model is a "state space model"

State: what the model remembers in between time steps

A Coarse Analogy

Transformers generate **quadratically** by attending to every past token

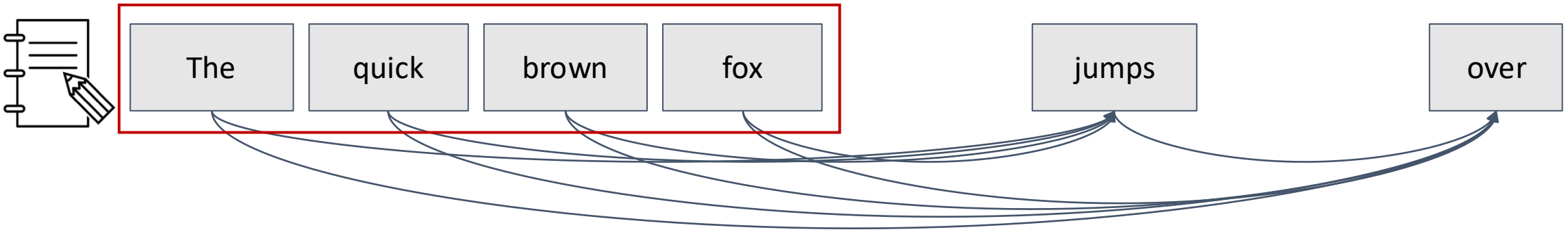


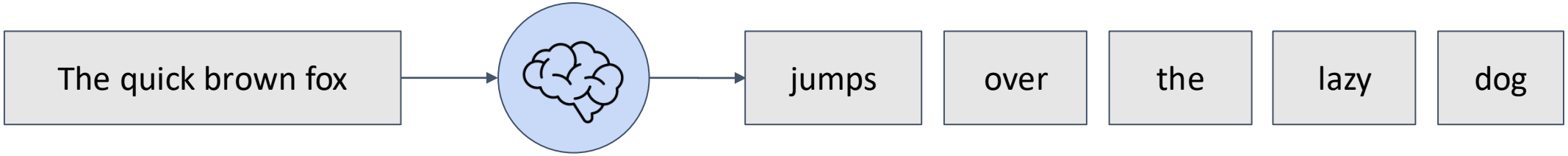


State is highly compressed, enabling generation in **linear time**

A Coarse Analogy

Transformers are like a **database**





State space models are like a **brain**

Tradeoffs of SSMs

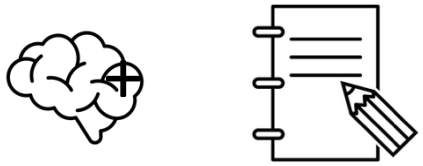
Stateful, compressive model

Interactive, online processing

Lacks fine-grained **recall and retrieval** abilities

Associative recall, needle-in-a-haystack, general QA

Hybrid Models

Intelligence = 

Nemotron 3 Ultra · Tencent Hunyuan · MINIMAX · Kimi · Qwen3.5 · Microsoft

- H3 (Fu, Dao et al. “Hungry Hungry Hippos: Towards Language Modeling with State Space Models”)
- Jamba (Lieber, Lenz et al. “Jamba: A Hybrid Transformer-Mamba Language Model”)
- Zamba (Glorioso et al. “Zamba: A Compact 7B SSM Hybrid Model”)
- Samba (Ren et al. “Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling”)
- Waleffe et al. “An Empirical Study of Mamba-based Language Models”

Is compression a **bug**, or a **feature**?

Bounded State Sequence Models

- State space models
- Linear attention
- Test-time regression
- Slot memories (Raven)
- Parametric attention (Urchin)

Linear Attention

Katharopoulos et al. ’20: kernel view of softmax attention

	Training (“Parallel Form”)	Inference (“Recurrent Form”)
Softmax Attention	$\mathbf{O} = \mathrm{softmax}\left((\mathbf{Q}\mathbf{K}^\top) \odot \mathbf{M}\right)\mathbf{V}$	$\boldsymbol{o}_t = \sum_{j=1}^{t} \frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j)}{\sum_{l=1}^{t} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_l)} \boldsymbol{v}_j$
(Simple) Linear Attention	$\mathbf{O} = \left((\mathbf{Q}\mathbf{K}^\top) \odot \mathbf{M}\right)\mathbf{V}$	$\boldsymbol{o}_t = \sum_{j=1}^{t} (\boldsymbol{q}_t^\top \boldsymbol{k}_j)\boldsymbol{v}_j$

$$S_t = \sum_{j=1}^{t} k_j v_j \qquad o_t = q_t^T S_t$$

Linear Attention

Katharopoulos et al. ’20: kernel view of softmax attention

	Training (“Parallel Form”)	Inference (“Recurrent Form”)
Softmax Attention	$\mathbf{O} = \mathrm{softmax}\left((\mathbf{Q}\mathbf{K}^\top) \odot \mathbf{M}\right)\mathbf{V}$	$\boldsymbol{o}_t = \sum_{j=1}^{t} \frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j)}{\sum_{l=1}^{t} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_l)} \boldsymbol{v}_j$
(Simple) Linear Attention	$\mathbf{O} = \left((\mathbf{Q}\mathbf{K}^\top) \odot \mathbf{M}\right)\mathbf{V}$	$\boldsymbol{o}_t = \sum_{j=1}^{t} (\boldsymbol{q}_t^\top \boldsymbol{k}_j)\boldsymbol{v}_j$

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

SSMs == Linear Attention

State Space Model

x, B, C

"State expansion"

Linear Attention

V, K, Q:

"Matrix-valued states"

$$h_t = \boldsymbol{A_t} h_{t-1} + \boldsymbol{B_t} x_t$$

$$y_t = \boldsymbol{C_t} h_t$$

State Expansion

x: 1-dimensional input channel

h: N-dimensional hidden state

SSMs == Linear Attention

State Space Model

x, B, C

"State expansion"

Linear Attention

V, K, Q:

"Matrix-valued states"

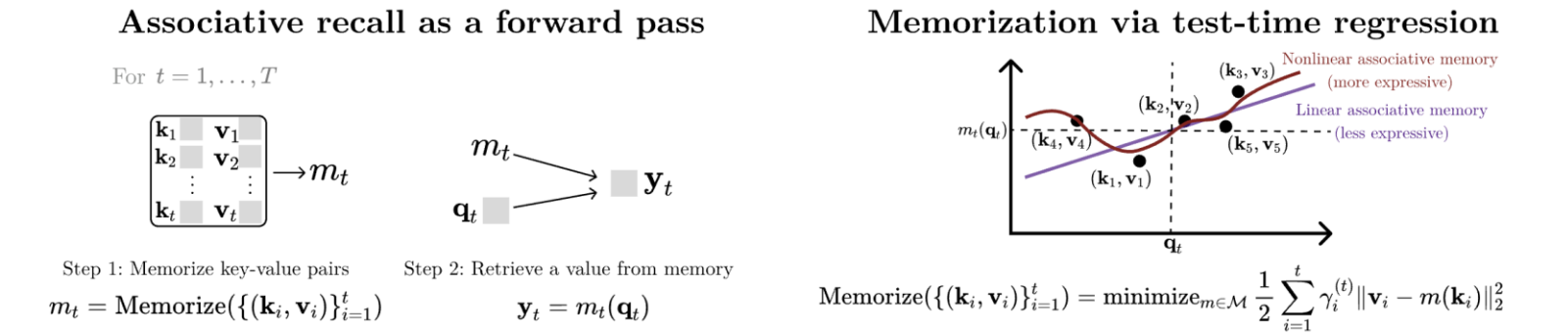
Inspired by dynamical systems

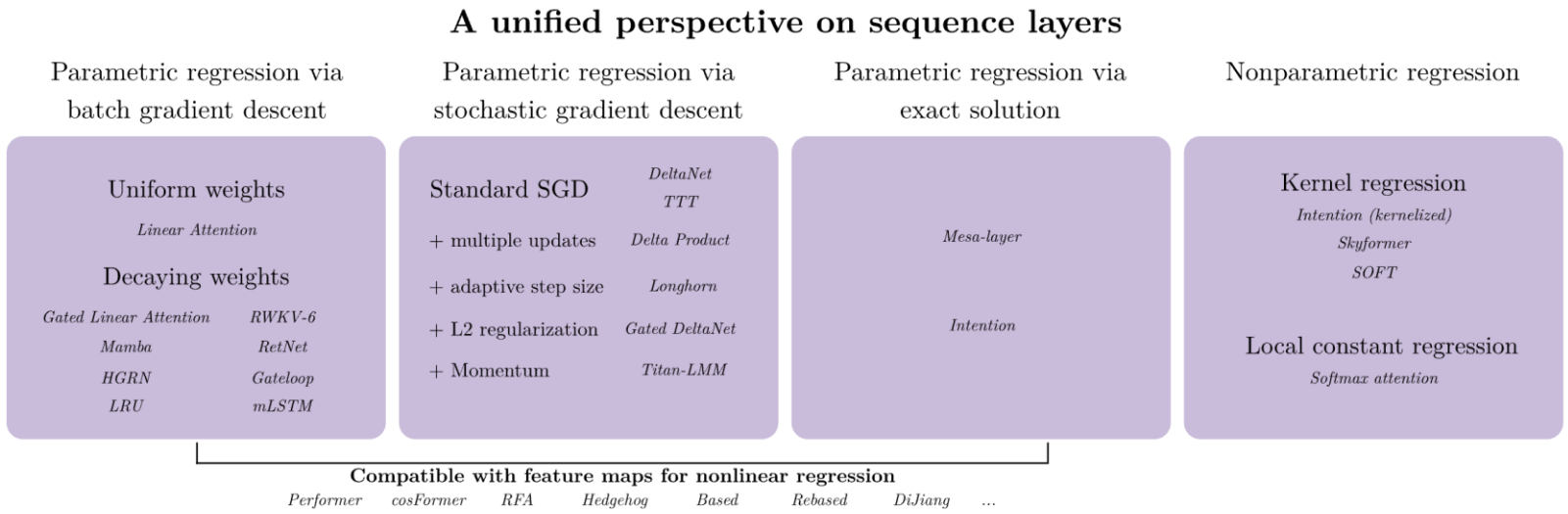
Gating / selectivity

Inspired by associative memory

Chunkwise training algorithm

Test-time Regression





Wang et al. “Test-time regression: a unifying framework for designing sequence models with associative memory”

Test-time Regression

Given data (key-value pairs), build a state that memorizes the data

DeltaNet fits in the TTR framework

$$S_t = S_{t-1}(I - \beta_t k_t k_t^T) + \beta_t k_t v_t$$

Remove existing binding for k

Many extensions (GDN, TTT, etc.)

Test-time Regression

Given data (key-value pairs), build a state that memorizes the data

DeltaNet fits in the TTR framework

$$\boxed{S_t} = S_{t-1}\boxed{(I - \beta_t k_t k_t^T)} + \beta_t \boxed{k_t}\boxed{v_t}$$

h_t A_t B_t x_t

Same form as a state space model!

Slot memories

- Attention’s core strength is caching a representation per token
  - Memory is a **set of slots**

- Sliding window attention (SWA) is an approximation
  - **Pro**: token-level granularity (slots)
  - **Con**: Rigid structure (rolling window)

The Memory Gap

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.

The Memory Gap

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**

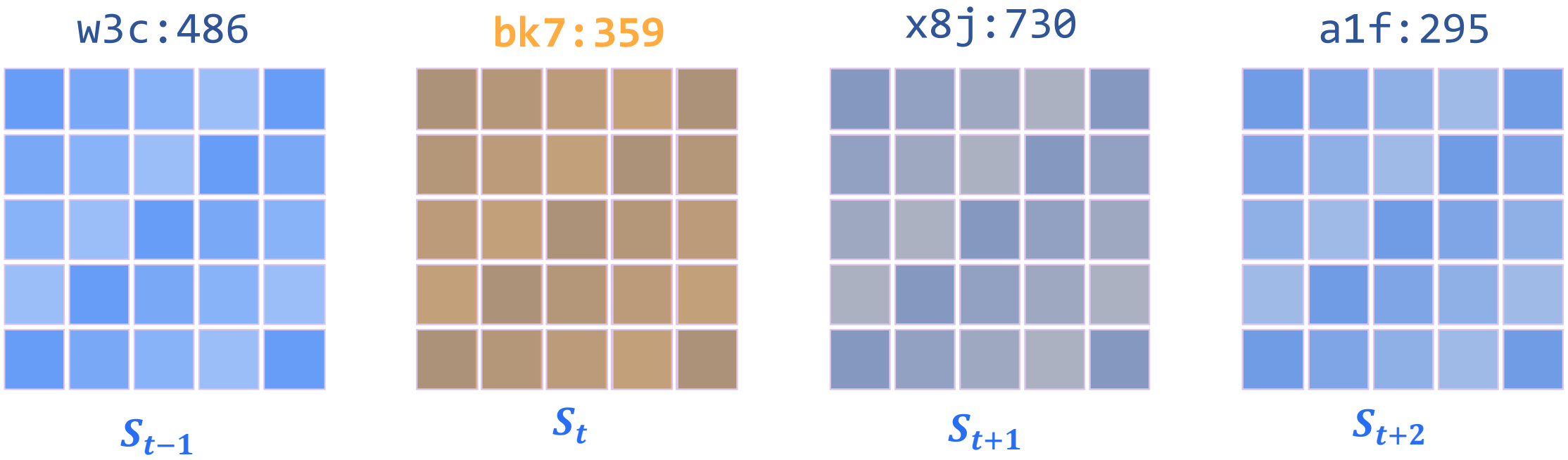
2 Reusing a variable defined 9k lines ago

```
  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)
```

The Memory Gap

Most SSMs fail NIAH even though they have enough memory

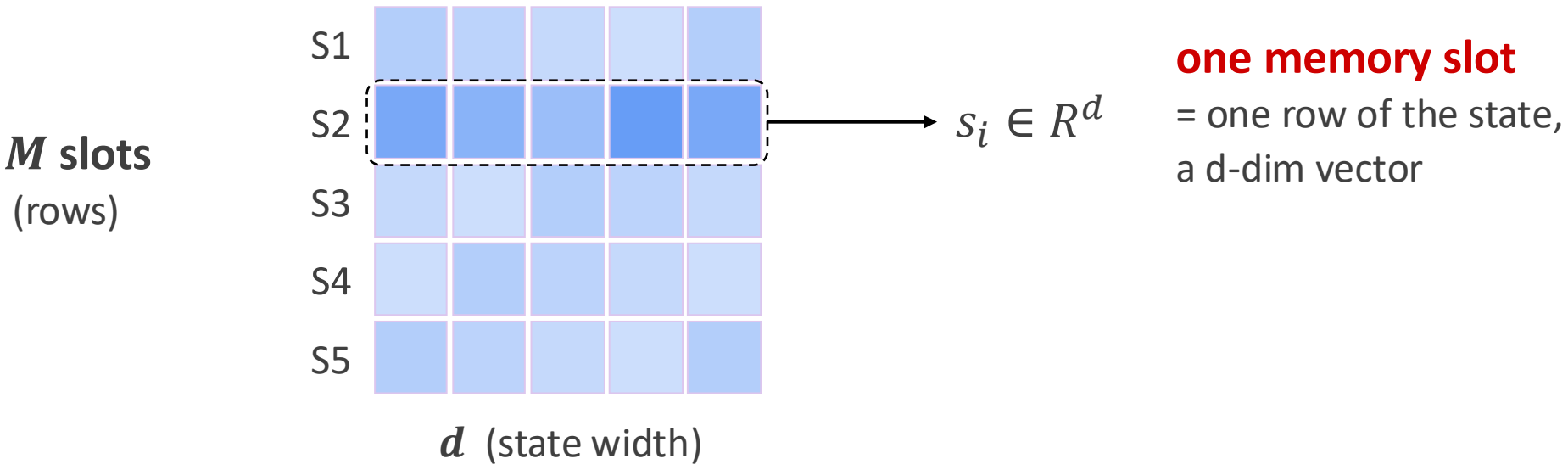
This is due to lack of organization in the state – they lack persistence



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.



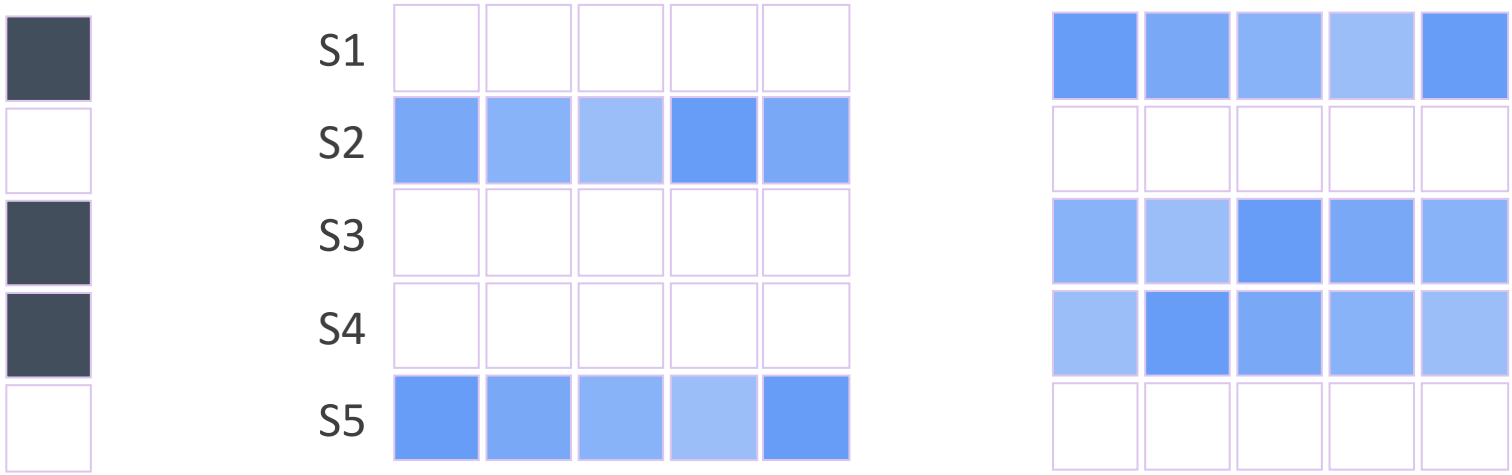
one memory slot
= one row of the state,
a d-dim vector

Towards Persistence

Routing Slot Memories (RSMs):

- Selected slots $(\mathrm{r_t})$ to be written
- Non-selected slots $(1 - \mathrm{r_t})$ 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 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:

$r_{\bar{t}} =$ 

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

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



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!



Afzal et al. “Raven: High-Recall Sequence Modeling with Sparse Memory Routing”

Routing Slot Memories

SWA



Raven



Mamba-2

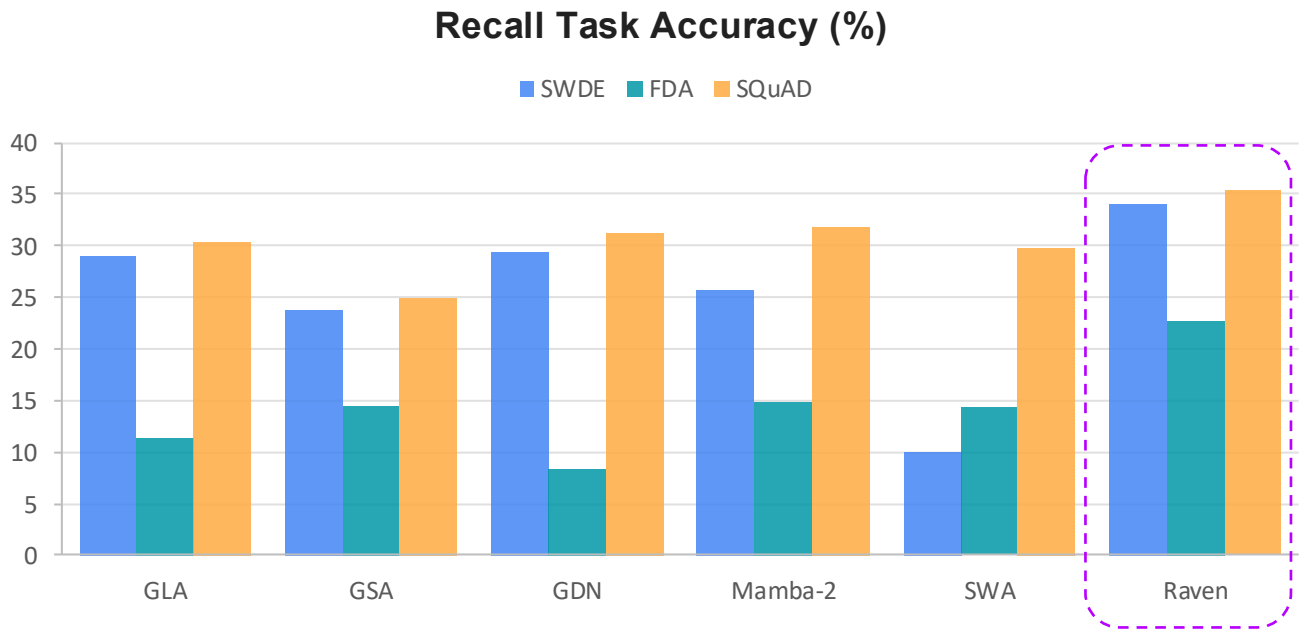


Routing Slot Memories

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	$(S_t^v)^\top f(S_t^k q_t)$

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**

Memory Visualization



Attention as a Mixture Model

Frameworks that approximate attention have had great success
Are there any other useful interpretations?

$$\boldsymbol{q}_t = \mathbf{W}_q \boldsymbol{x}_t \in \mathbb{R}^{d_k}, \qquad \boldsymbol{k}_t = \mathbf{W}_k \boldsymbol{x}_t \in \mathbb{R}^{d_k}, \qquad \boldsymbol{v}_t = \mathbf{W}_v \boldsymbol{x}_t \in \mathbb{R}^{d_v}$$

$$\boldsymbol{y}_t = \sum_{i=1}^{t} \frac{\exp\left(\boldsymbol{q}_t^\top \boldsymbol{k}_i / \sqrt{d_k}\right)}{\sum_{j=1}^{t} \exp\left(\boldsymbol{q}_t^\top \boldsymbol{k}_j / \sqrt{d_k}\right)} \boldsymbol{v}_i \in \mathbb{R}^{d_v}$$

Claim: softmax attention is the nonparametric GMM

$$\boldsymbol{y}_t = \mathbb{E}\left[\mathsf{V} \mid \mathsf{K} = \boldsymbol{q}_t\right] \in \mathbb{R}^{d_v}$$

Attention as a Mixture Model (Urchin)

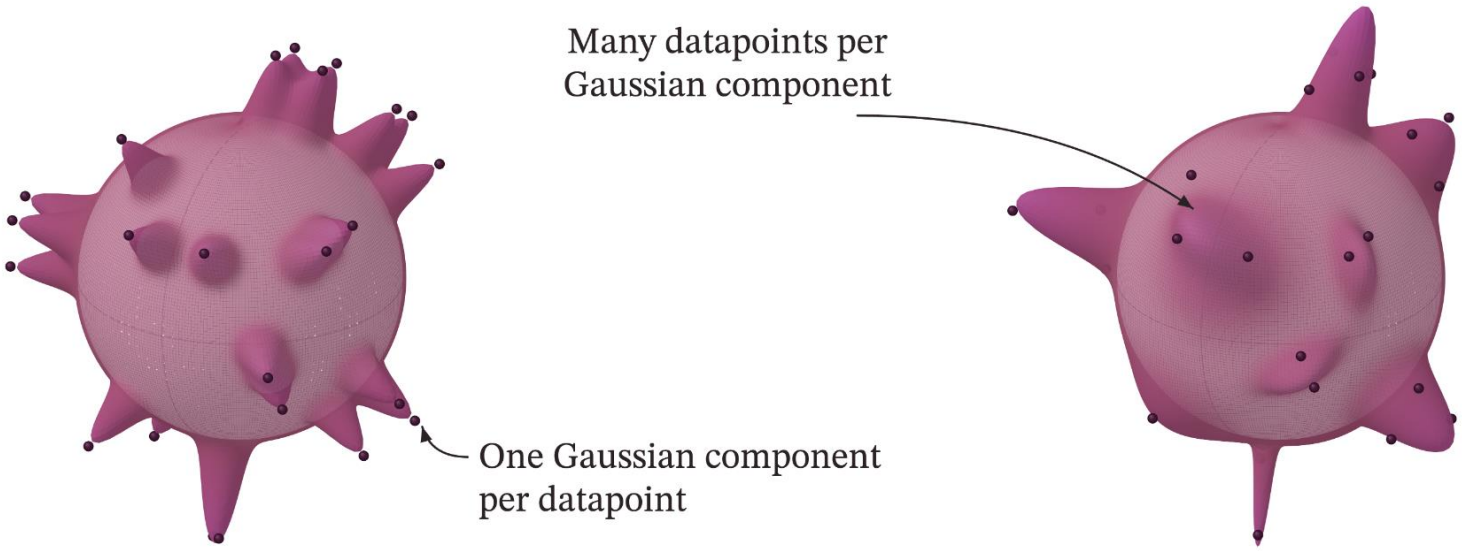


Figure 1: (Left) Attention can be viewed as modeling the probability density of key-value pairs (points) with a Gaussian mixture component centered at each point. (Right) We introduce a class of fixed-state models which parametrically model the density with a fixed number of components.

McDermott et al. “Urchin: Parametrizing Attention for Long Context Recall”

Attention as a Mixture Model (Urchin)

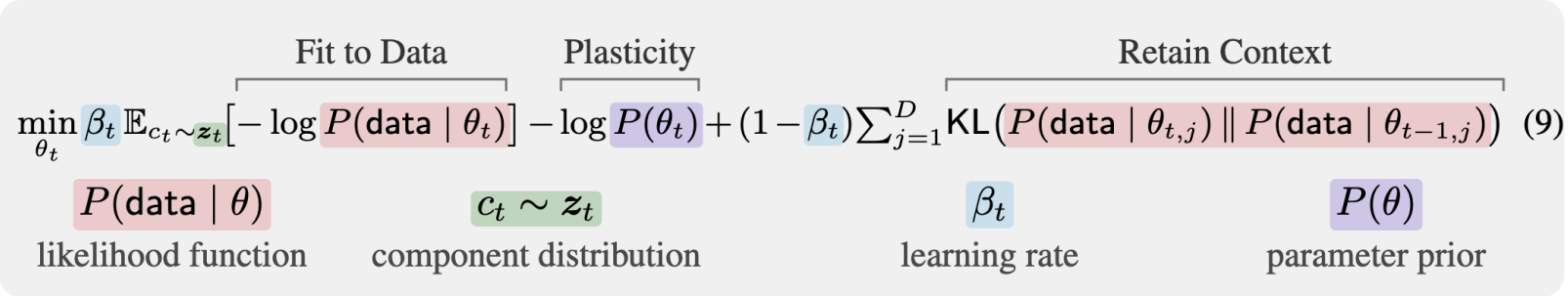


Figure 2: In our framework, fixed-state layers are a choice of four things, based on the three desiderata of fitting to data, maintaining plasticity, and retaining context.

McDermott et al. “Urchin: Parametrizing Attention for Long Context Recall”

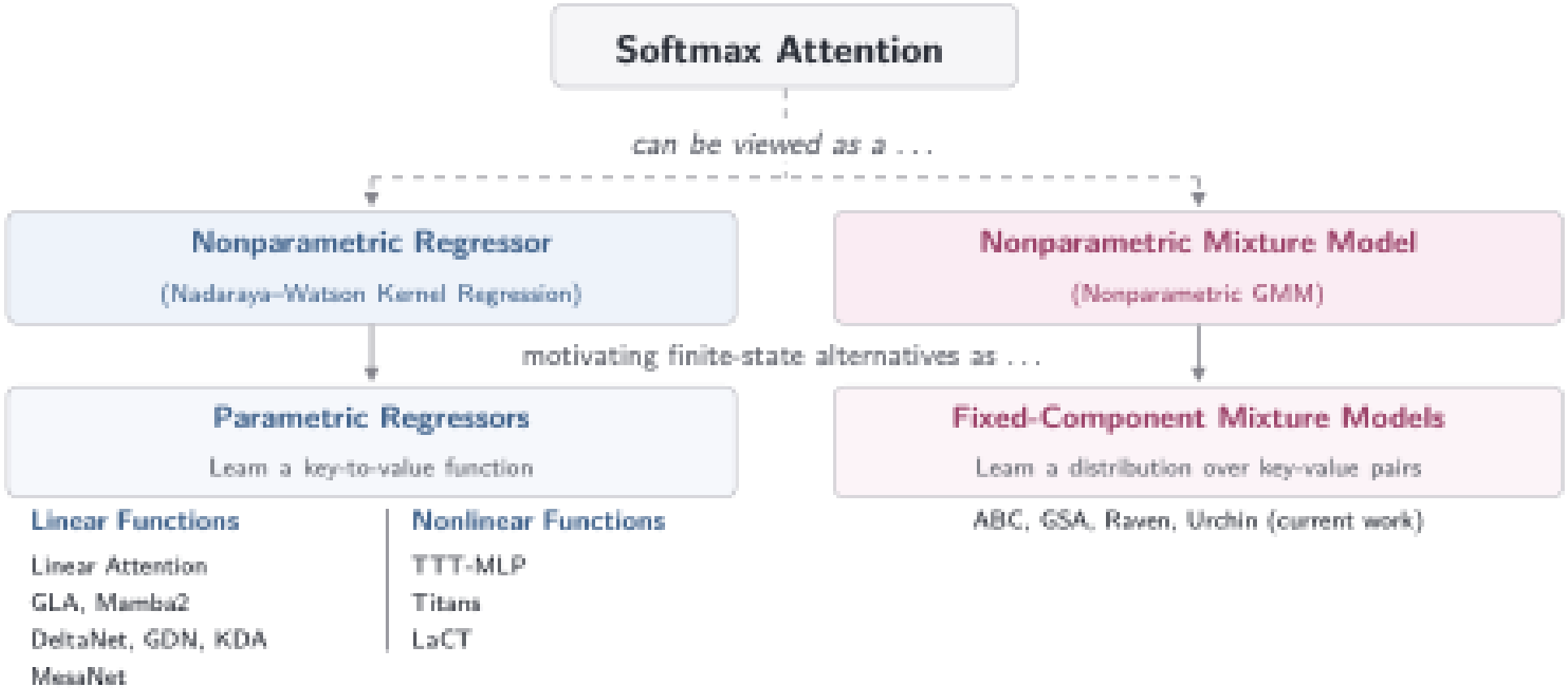
Table 1: **Long Context Recall** on RULER S-NIAH-1 at extreme context lengths. We report the average of three training seeds for 120M and 400M models. Accuracies are averaged over 500 samples up to 128K, 250 at 256K, 100 at 384K/512K and 50 at 768K/1M.

		S-NIAH-1 Across Context Lengths												
Model	**Params**	1K	2K	4K	8K	16K	32K	64K	128K	256K	384K	512K	768K	1M
~120M params, 15B tokens														
Mamba-2	121M	99.3 ±0	69.2 ±20	25.9 ±11	11.6 ±6	6.3 ±2	3.3 ±2	1.7 ±1	0.9 ±0	0.2 ±0	0.0 ±0	0.0 ±0	0.0 ±0	0.0 ±0
GLA	121M	43.8 ±5	15.6 ±1	4.1 ±1	0.9 ±1	0.9 ±1	0.4 ±0	0.2 ±0	0.0 ±0	0.0 ±0	0.0 ±0	0.0 ±0	0.0 ±0	0.0 ±0
GSA	121M	96.7 ±0	95.8 ±1	90.2 ±5	76.2 ±21	60.0 ±39	36.9 ±30	18.7 ±15	10.7 ±9	5.2 ±5	3.5 ±4	0.5 ±1	2.0 ±3	0.0 ±0
Gated DeltaNet	121M	97.9 ±4	96.0 ±6	91.3 ±13	75.4 ±23	35.3 ±16	17.9 ±8	10.3 ±4	4.9 ±3	2.3 ±1	1.7 ±1	0.0 ±0	0.0 ±0	0.0 ±0
KDA	127M	**99.9** ±0	99.7 ±0	94.0 ±6	73.9 ±23	46.1 ±31	18.6 ±12	4.6 ±2	1.3 ±1	0.8 ±1	0.0 ±0	0.3 ±1	0.0 ±0	0.0 ±0
Raven	127M	97.7 ±2	96.9 ±3	92.7 ±11	84.8 ±22	72.9 ±35	59.7 ±33	45.9 ±39	30.1 ±32	13.9 ±17	9.0 ±8	5.0 ±6	2.7 ±3	1.3 ±1
Urchin	124M	99.7 ±1	99.7 ±1	98.1 ±3	96.3 ±5	89.9 ±10	**84.8** ±14	**80.8** ±17	**80.9** ±18	**80.3** ±19	**77.3** ±22	**75.7** ±22	**78.0** ±20	**82.0** ±18
Urchin-KDA	126M	**99.9** ±0	**99.9** ±0	**99.8** ±0	**98.5** ±2	**92.2** ±11	77.1 ±28	54.6 ±38	43.3 ±46	34.9 ±47	33.3 ±47	28.7 ±48	22.7 ±38	26.7 ±43
~400M params, 15B tokens														
Mamba-2	400M	89.1 ±8	51.3 ±19	10.4 ±8	3.0 ±3	1.9 ±2	0.5 ±1	0.5 ±1	0.2 ±0	0.3 ±0	0.0 ±0	0.0 ±0	0.0 ±0	–
GLA	400M	84.6 ±16	30.3 ±13	9.9 ±3	3.4 ±1	2.3 ±2	0.4 ±0	0.3 ±0	0.0 ±0	0.0 ±0	0.0 ±0	0.0 ±0	0.0 ±0	0.0 ±0
GSA	399M	99.1 ±1	98.1 ±2	94.3 ±4	74.3 ±6	39.4 ±4	17.6 ±3	9.5 ±3	5.7 ±1	1.9 ±0	1.7 ±1	0.0 ±0	0.0 ±0	0.0 ±0
Gated DeltaNet	400M	**100.0** ±0	**100.0** ±0	98.2 ±2	82.7 ±15	55.9 ±34	34.5 ±27	19.4 ±15	10.5 ±11	5.1 ±6	5.3 ±5	0.7 ±1	1.3 ±2	1.3 ±2
KDA	400M	**100.0** ±0	99.9 ±0	97.0 ±5	85.1 ±19	62.3 ±30	40.7 ±31	27.0 ±28	17.0 ±22	8.8 ±12	5.3 ±7	2.0 ±4	2.0 ±4	2.7 ±5
Raven	424M	99.2 ±1	99.1 ±1	98.7 ±1	97.9 ±1	93.7 ±1	83.0 ±7	57.9 ±28	42.1 ±39	33.2 ±36	21.7 ±27	13.3 ±14	6.7 ±6	5.3 ±8
Urchin	406M	**100.0** ±0	**100.0** ±0	**100.0** ±0	**99.9** ±0	**99.3** ±1	**98.3** ±2	**96.8** ±3	**96.4** ±4	**95.9** ±4	**95.7** ±5	**97.3** ±2	**94.7** ±5	**97.3** ±3
Urchin-KDA	402M	**100.0** ±0	**100.0** ±0	**100.0** ±0	99.5 ±1	94.5 ±10	82.9 ±30	74.1 ±44	67.6 ±49	49.5 ±42	40.0 ±40	28.7 ±33	16.7 ±19	11.3 ±18
~850M params, 30B tokens														
Mamba-2	835M	**100.0**	**100.0**	99.0	52.6	19.4	8.0	2.6	0.4	0.4	–	–	–	–
GLA	836M	**100.0**	95.8	40.6	9.0	2.6	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GSA	834M	**100.0**	**100.0**	99.6	98.0	88.0	38.6	10.6	6.0	1.6	0.0	0.0	0.0	0.0
Gated DeltaNet	835M	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	99.4	93.8	63.6	41.0	30.0	16.0	18.0
KDA	816M	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	99.8	96.2	66.8	28.0	13.0	7.0	4.0	4.0
Raven	891M	99.6	99.6	99.6	99.6	**100.0**	84.0	91.8	90.8	87.6	84.0	75.0	74.0	64.0
Urchin	845M	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Urchin-KDA	826M	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	95.2	58.8	32.8	21.0	11.0	10.0	6.0

Table 2: **Beyond 1 Million Token Haystacks**. Accuracies are averaged over 50 samples.

		S-NIAH-1 Across Context Lengths								
Model	**Params**	1.25M	1.5M	1.75M	2M	2.5M	3M	5M	7.5M	10M
Urchin	406M	94.0 ±3.5	92.0 ±4.0	95.3 ±2.3	93.3 ±1.2	93.3 ±4.6	91.3 ±3.1	93.3 ±4.6	94.0 ±4.0	93.3 ±6.4

Frameworks for Parametric Attention

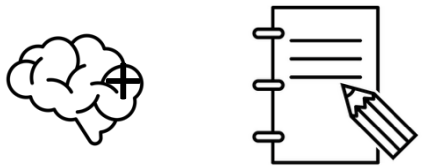


Bounded State Sequence Models

• State space models	Dynamical systems
• Linear attention	Softmax kernel approximation
• Test-time regression	Associative memories
• Slot memories (Raven)	KV cache slots
• Parametric attention (Urchin)	Mixture models

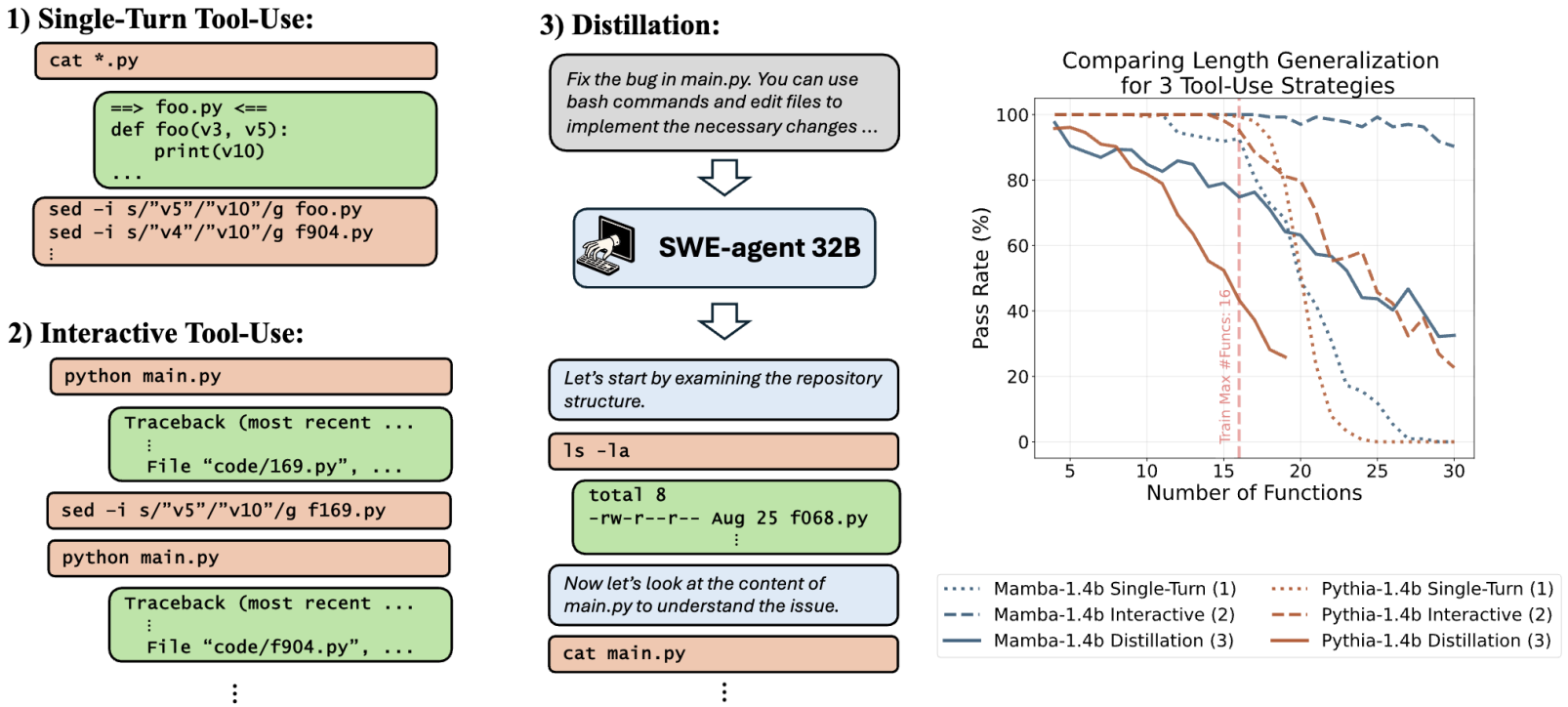
Beyond the Read-once Paradigm

- We care about long context, retrieval etc. in the context of the “read once” sequence model paradigm
- Fundamentally impossible to solve with bounded state
- Direction 1: Sequence models with **external databases** and **explicit tool use**

Intelligence = 

Beyond the Read-once Paradigm

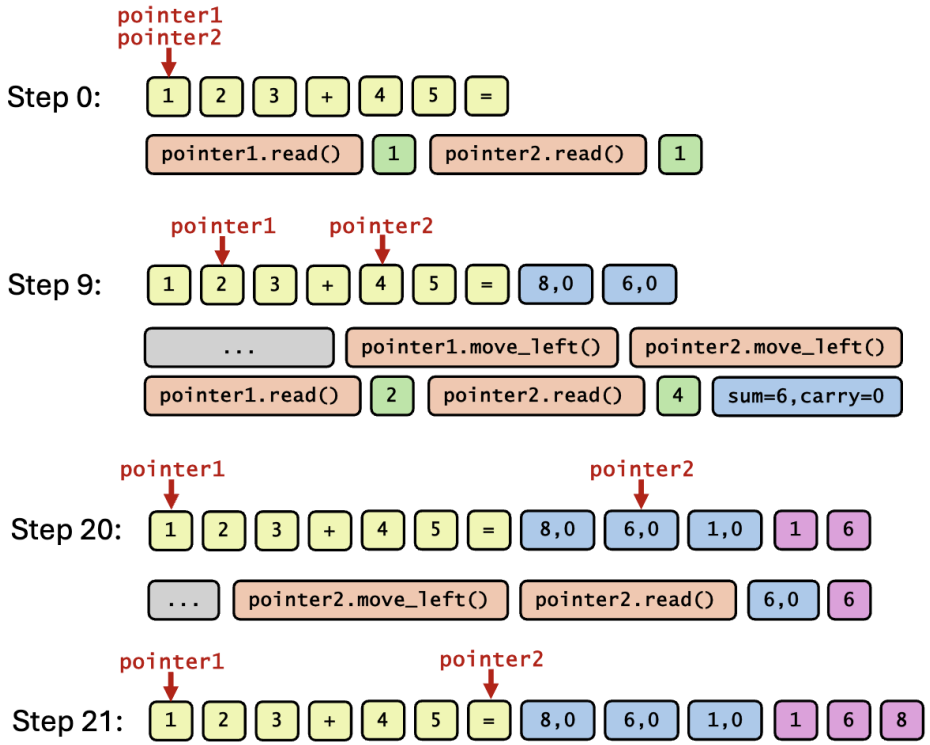
- Direction 1: Sequence models with **external databases** and **explicit tool use**

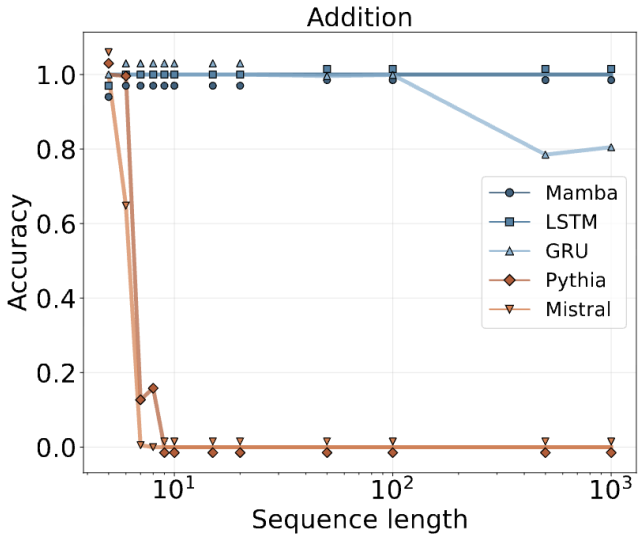


Malach et al. "To Infinity and Beyond: Tool-Use Unlocks Length Generalization in State Space Models"

Beyond the Read-once Paradigm

- Direction 1: Sequence models with **external databases** and **explicit tool use**





Beyond the Read-once Paradigm

- Direction 1: Sequence models with **external databases** and **explicit tool use**

- Direction 2: Fundamental **limits of state size** and **memory**
  - Imagine we're still in the read-once paradigm and need to cache the context
  - Attention requires ~10000 bytes per token
  - Storing the original data requires ~1 byte per token
  - Question: design a sequence model layer with as small a state possible that has the same capabilities as quadratic attention

Takeaways

- State space models have a fundamental tradeoff versus attention (brain vs database)
  - Particularly linear recurrent models (SSMs, LA, TTR)
  - More efficient, worse retrieval
- Approximating attention is a fruitful direction to developing strong bounded-state models
  - Linear attention, test-time regression, slot memories, mixture models
- There are many underexplored directions for context compaction, dynamic context processing, and so on