CS11-711 Advanced NLP

Long-Context Models

Sean Welleck

Carnegie Mellon University
Language Technologies Institute

https://cmu-l3.github.io/anlp-spring2025/

Many slides by Graham Neubig from Fall 2024

How Long are Sequences?

- One sentence: ~20 tokens
- One document: 100-10k tokens
- One book: 50k-300k tokens
- One video: 1.5k-1M tokens (~300/sec)
- One codebase: 20k-1B tokens
- One genome: 3B nucleotides

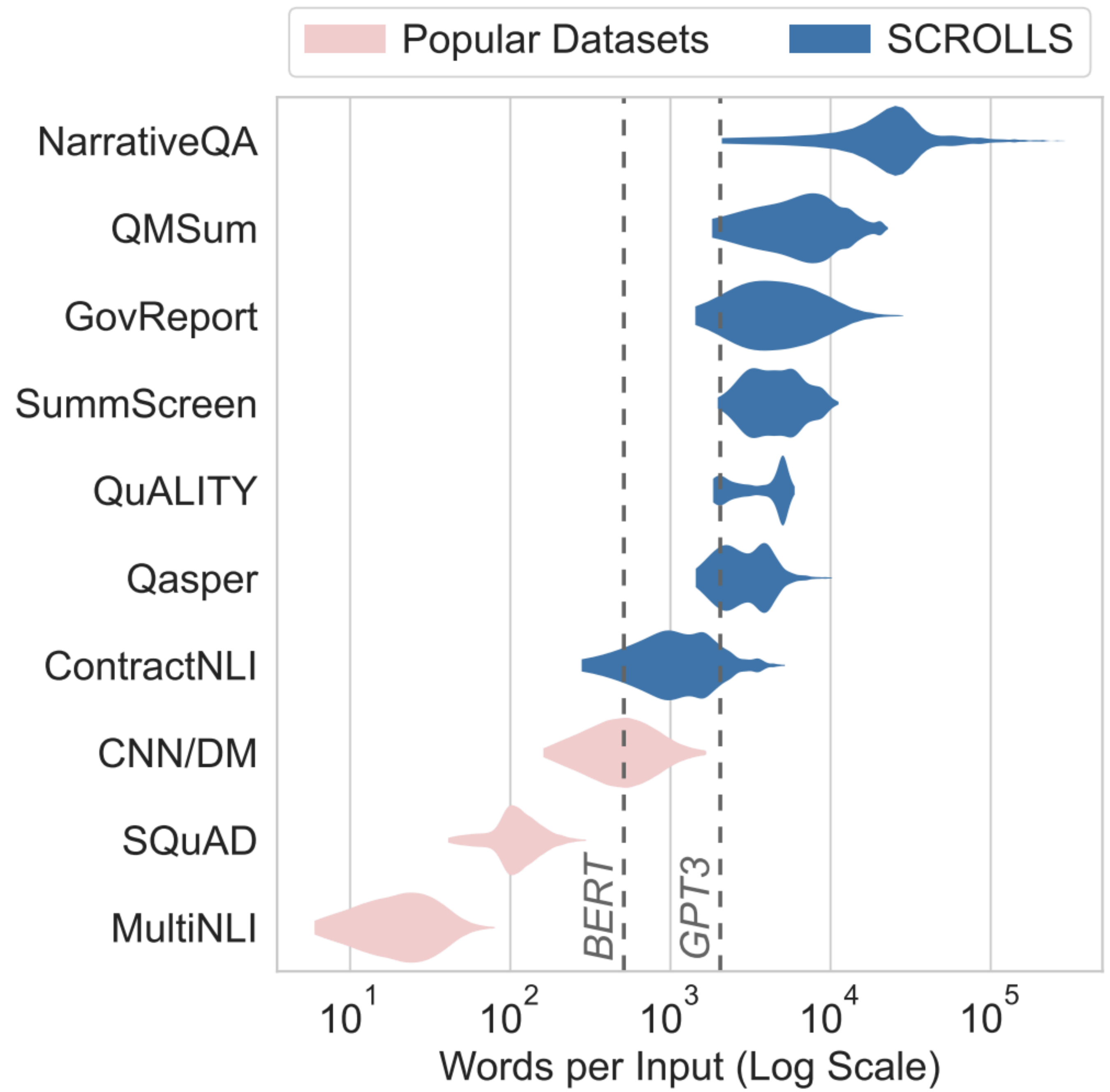
Why is Modeling Long Sequences Hard?

- **Memory Complexity:** Transformer models scale quadratically in memory
- **Compute Complexity:** Transformer models scale quadratically in computation
- **Training:** Data is lacking, training signal is weak, training on long sequences is costly

Long-context Use Cases and Evaluation

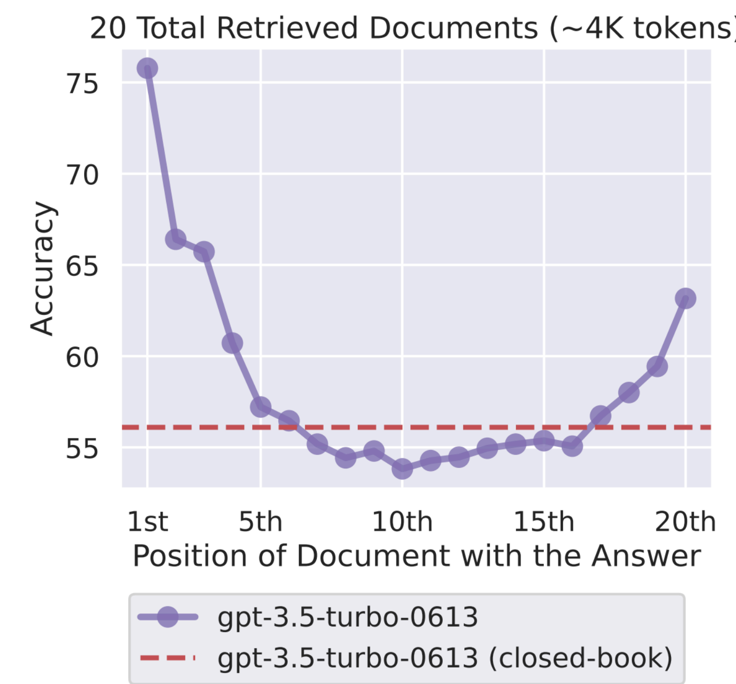
Benchmarks for Long-context Models

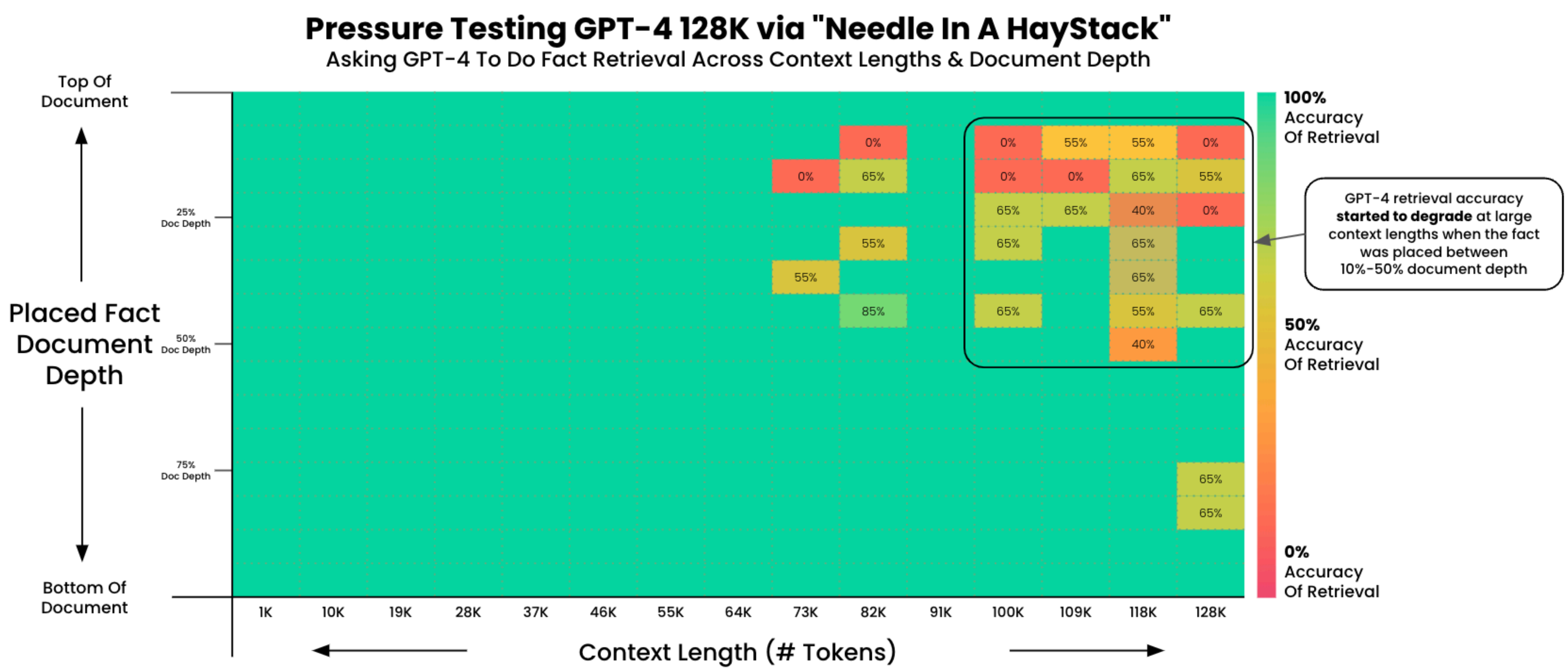
- **Long Range Arena:** Composite benchmark containing mostly non-NLP tasks (Tay et al. 2020)
- **SCROLLS:** Benchmark containing long-context summarization, QA, etc. (Shaham et al. 2022)



Targeted Analysis Tools

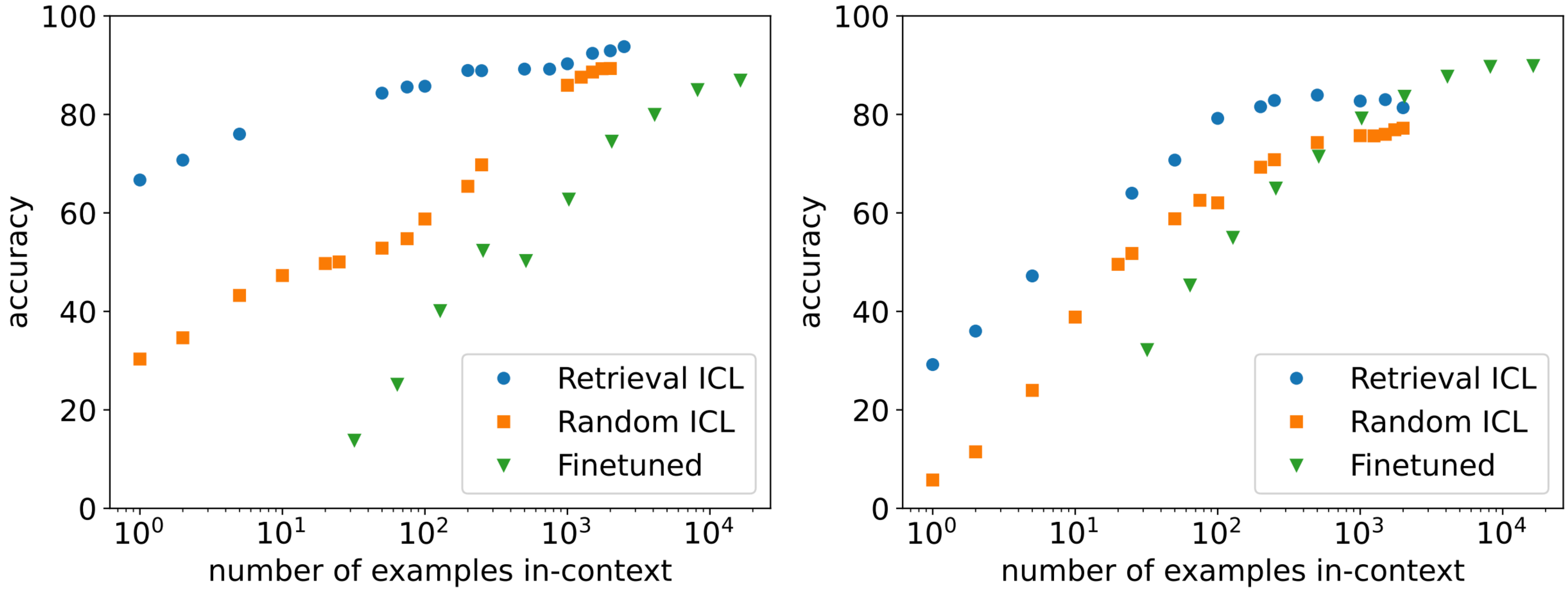
- "lost-in-the-middle" (Liu et al. 2023) demonstrates that models pay less attention to things in middle context
- "needle in a haystack" tests (Kamradt 2023) test across document length/position
- RULER (Hsieh et al. 2024) compiles a number of different NIAH tasks





Long-context In-context Learning

(Bertsch et al. 2024)



(a) Clinic-150 (b) Trecfine

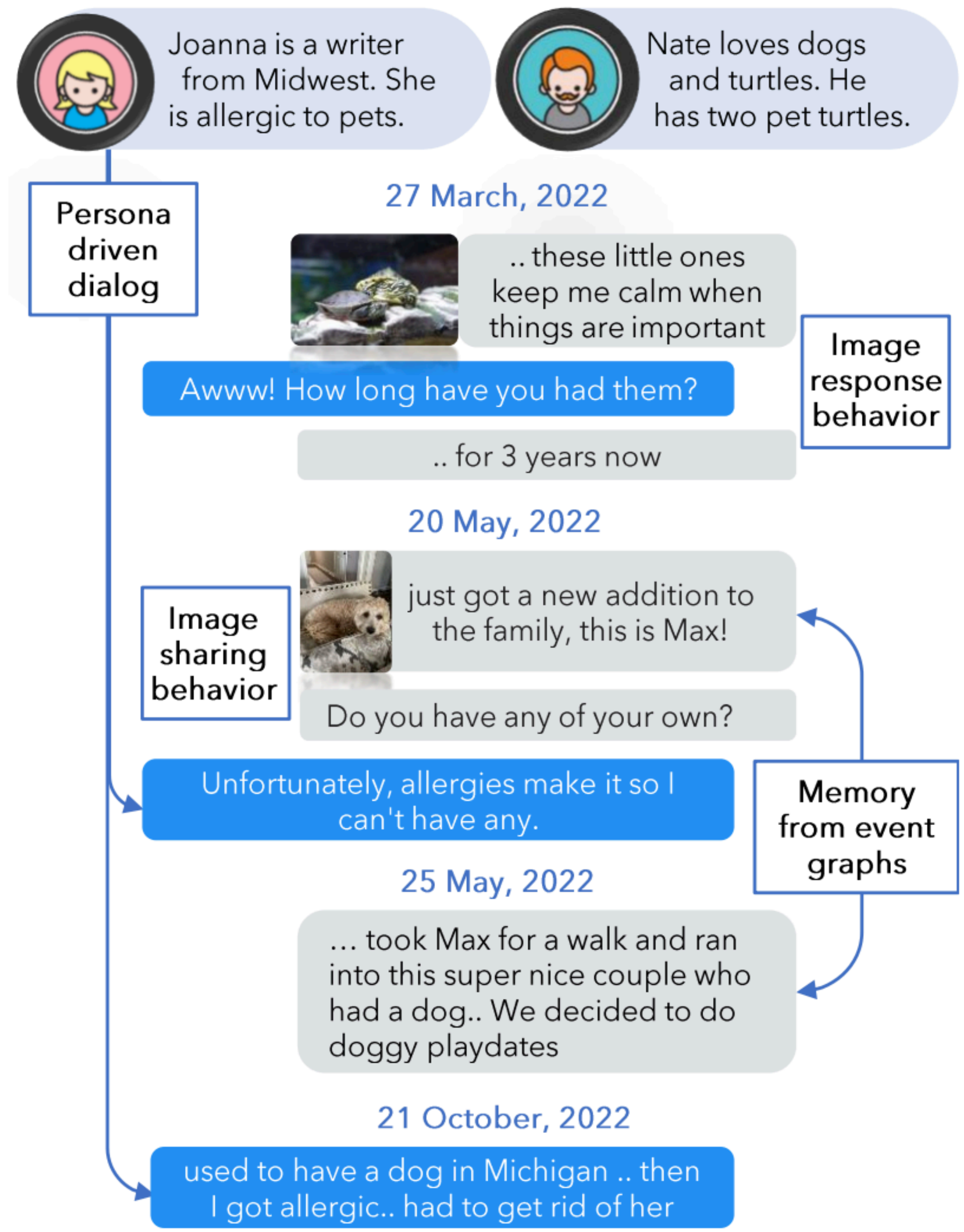
- When many in-context examples are provided, it can be better than fine-tuning

Long-context Dialog

- Chatbots that maintain long-term conversational context

- E.g., Locomo corpus (Maharana et al. 2024)

- Evaluate with question answering, summarization, response generation



Today's lecture

- Long sequence modeling
- **Improving transformers**
  - Memory-efficient computation
  - Extrapolation
  - Transformer modifications
- Transformer alternatives

Vanilla Attention Complexity

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

$$\text{Attention}(Q, K, V) = AV$$

Compute: $O(bs^2d)$ for QK^T

Memory: $\boxed{O(bs^2)}$ for all ops

Compute: $O(bs^2d)$ for AV

Memory: $O(bsd)$

b: batch size, s: sequence length, d: dimension

Multi-head Attention Complexity

- Multi-head attention splits attention heads
- No effect on compute complexity, but effect on memory

Compute: $O(bs^2d)$ for QK^T

Compute: $O(bs^2d)$ for AV

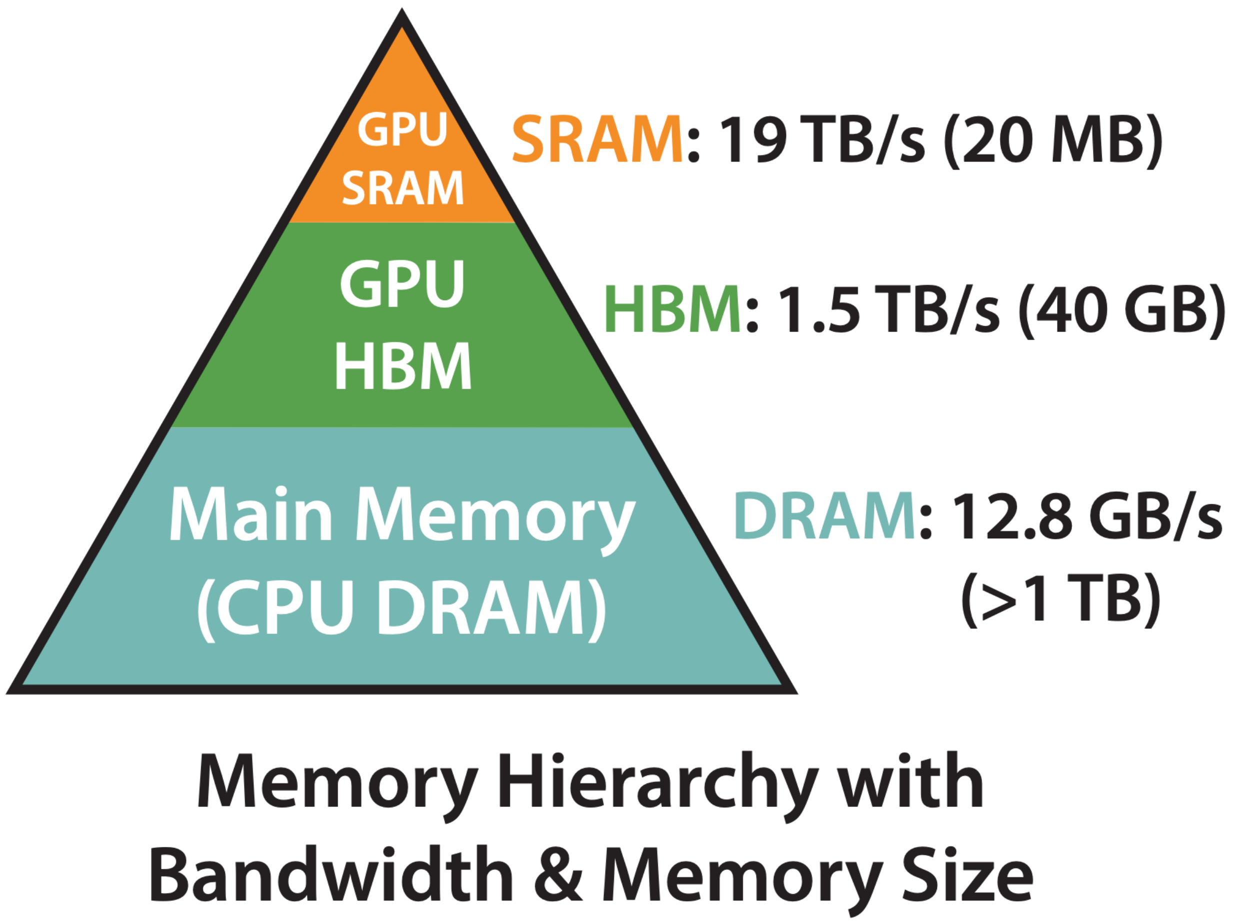
Memory: $\boxed{O(bs^2h)}$ for all ops

Memory: $O(bsd)$

b: batch size, s: sequence length, d: dimension, h: heads

Memory bottlenecks

- Accelerators (e.g., CUDA GPU) have limited memory capacity and bandwidth



$O(bs^2h)$ memory means many slow SRAM ↔ HBM transfers

Image: FlashAttention [Dao et al 2022]

Memory bottlenecks

- Implications:
  - Expensive to **(pre-)train** with a long context length
  - Expensive to **generate** (inference)

- Expensive can mean:
  - Slow: bandwidth leads to transfers
  - Infeasible: simply run out of memory

Memory-efficient computation

(Jang 2019, Rabe and Staats 2021)

- Insight: you can compute softmax "online" to avoid materializing the s^2 matrices

$$\text{Attention}(Q, K, V) = V^*/S^*$$

Memory: *O(bsd)*

<u>*softmax numerator * V*</u>

$$V^* = \exp\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Memory: *O(bsd)*

<u>*softmax denominator*</u>

$$S^* = \text{sum}\left(\exp\left(\frac{QK^T}{\sqrt{d_k}}\right)\right)$$

Memory: *O(bsh)*

Memory-efficient computation

(Jang 2019, Rabe and Staats 2021)

- Online softmax

For each query q:

For $i = 1, \ldots, s$:[1]

$$w'_i = \mathrm{dot}(q, k_i)$$

$$v^* \leftarrow v^* + v_i \,\mathbf{exp}(w'_i)$$

$$s^* \leftarrow s^* + \mathbf{exp}(w'_i)$$

At the end, $\mathrm{Attention}(q) = v^*/s^*$.

[1] In general, segment into "chunks" (aka "blocks" / "tiles").

Memory-efficient computation

~ 1 million tokens

Sequence length	$n = 2^8$	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}
Size of inputs and outputs	160KB	640KB	2.5MB	10MB	40MB	160MB	640MB
Memory overhead of standard attention	270KB	4.0MB	64MB	1GB	OOM	OOM	OOM
Memory overhead of memory-eff. attn.	270KB	4.0MB	16MB	17MB	21MB	64MB	256MB
Compute time on TPUv3	0.06ms	0.11ms	0.7ms	11.3ms	177ms	2.82s	45.2s
Relative compute speed	±5%	±5%	-8±2%	-13±2%	-	-	-

Table 2: Memory and time requirements of self-attention during **inference**.

Inference benchmarking from [Rabe and Staats 2021]

Analogous improvements for training

Memory-efficient computation

- Transformer attention [Rabe & Staats 2021]
- *FlashAttention*: incorporate online softmax into a new CUDA kernel [Dao et al 2022]
- *Ring Attention*: distribute online computation across multiple devices [Liu et al 2023]

Ring Attention (Liu et al. 2023)

Context parallelism

- Split sequence into blocks across devices
- Each host holds one query block, and key-value blocks traverse through a ring
- Different strategies for splitting: contiguous blocks, clever interleaving

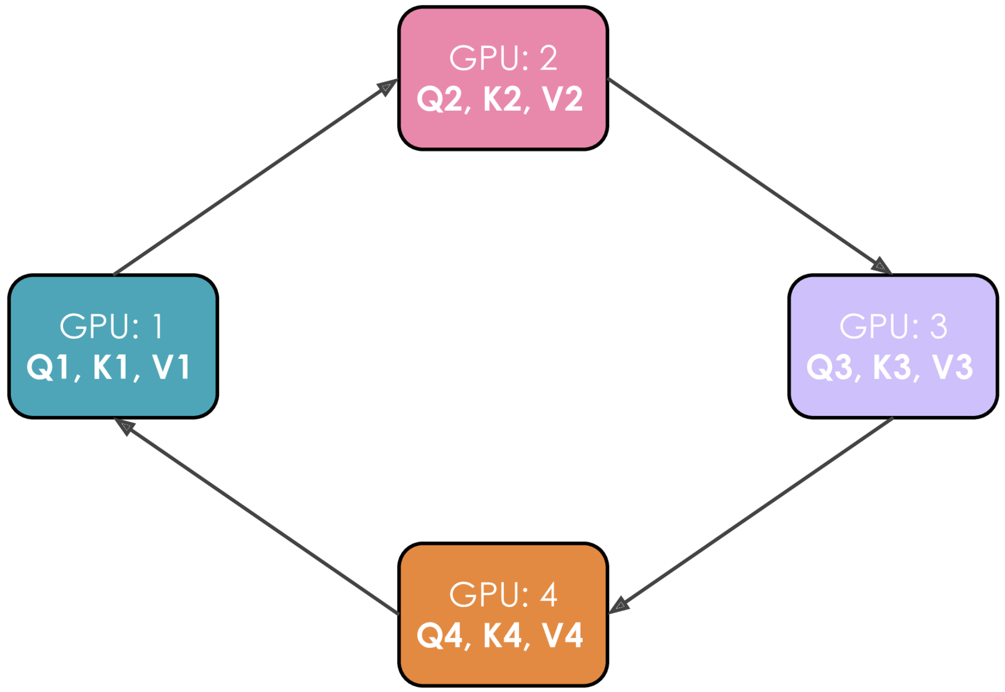


Image: The Ultra-Scale Playbook [Tazi et al 2025]

Ring attention / context parallelism

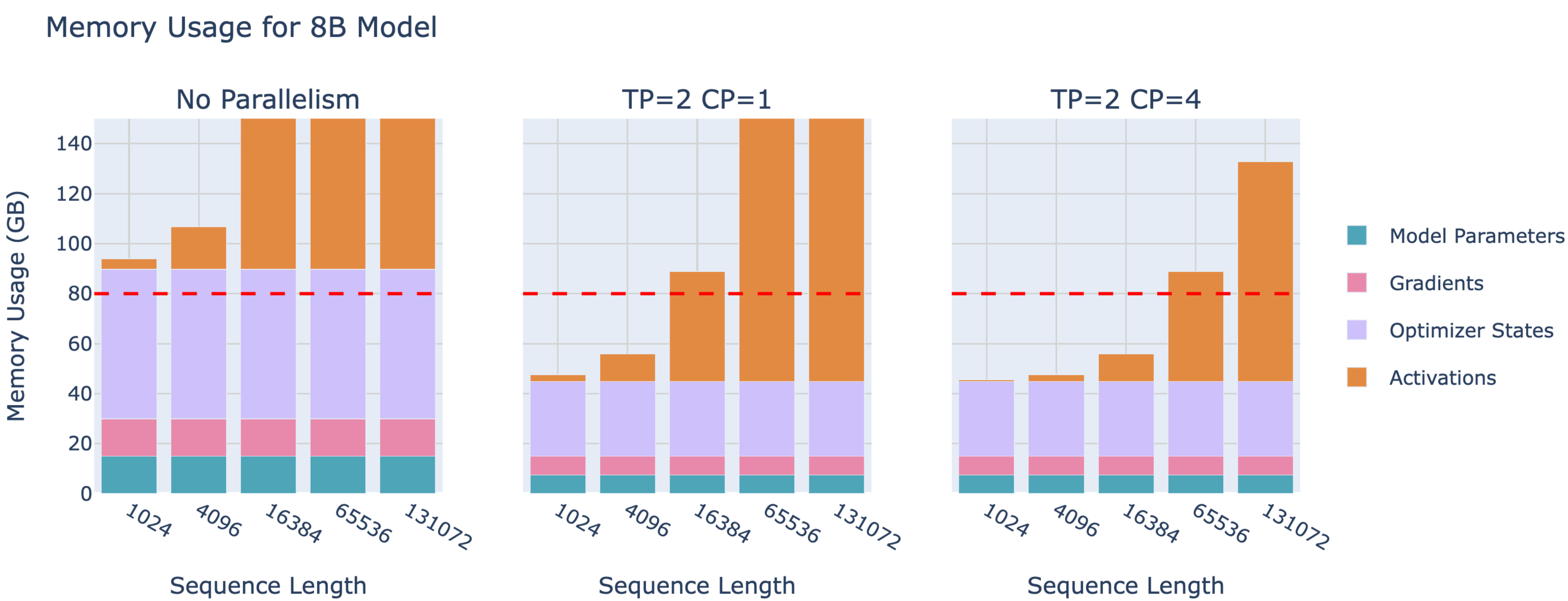


Image: The Ultra-Scale Playbook [Tazi et al 2025]

Today's lecture

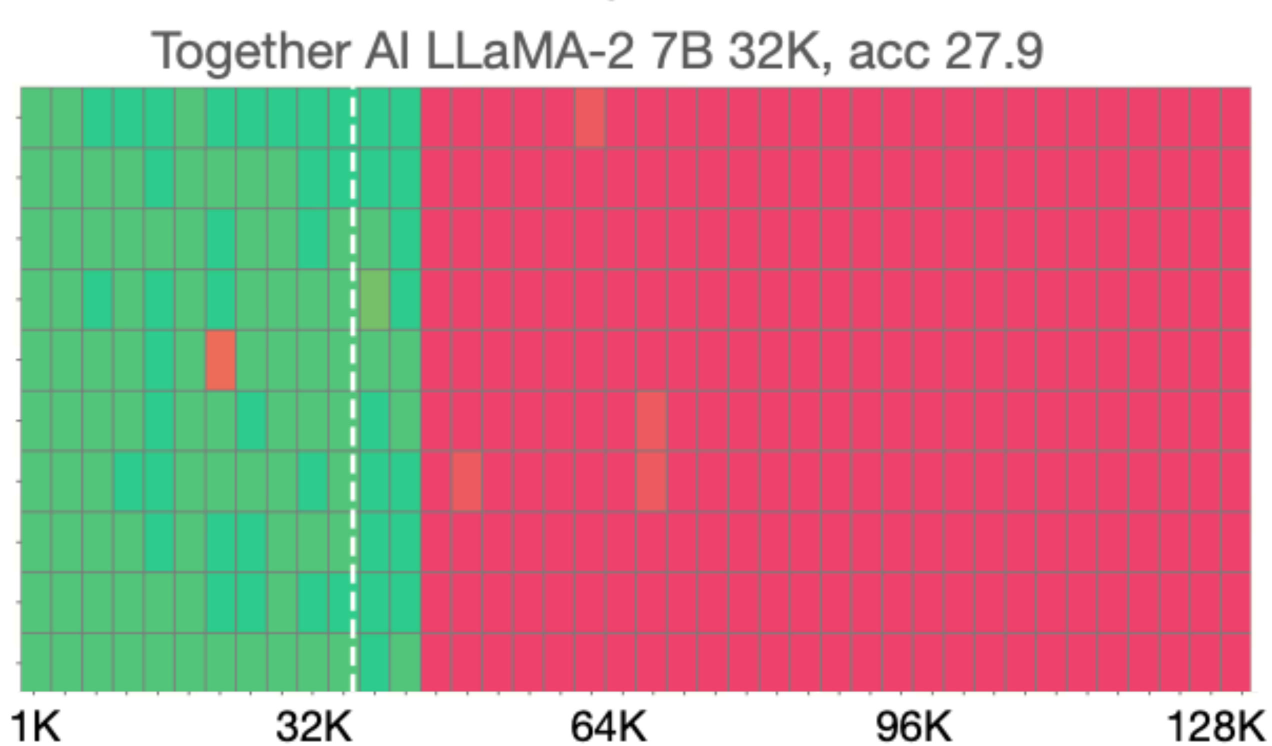
- Long sequence modeling
- Improving transformers
  - Memory-efficient computation
  - **Extrapolation**
  - Transformer modifications
- Transformer alternatives

Trained Models Fail to Extrapolate

- Most transformer models are trained on shorter sequences (4k)
    - If a document is longer than the limit, truncate or chunk
- This poses problems for positional encodings:
    - **Learned absolute encodings:** impossible to extrapolate
    - **Fixed absolute encodings:** move models out of distribution, very bad
    - **Relative encodings:** should extrapolate better in theory, but not really in practice

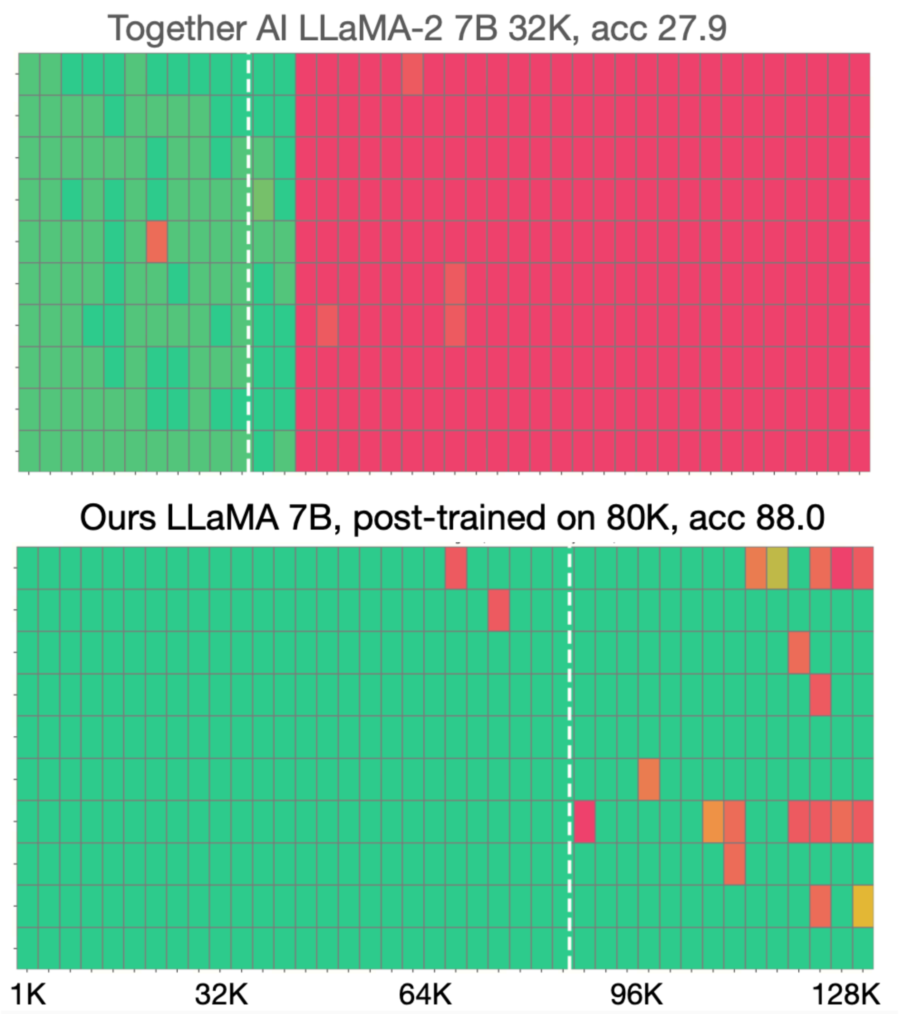
An Example of Failed Extrapolation (Fu et al. 2024)

- Llama-2 w/ 32k context (RoPE) can answer questions about sequences up to about 40k, but not beyond



Training with Long Context
(Fu et al. 2024)

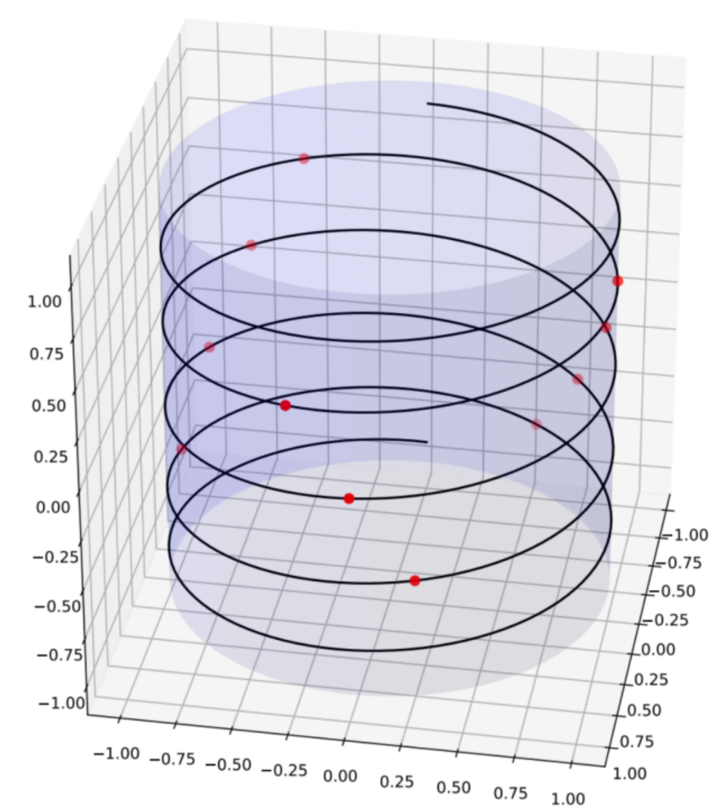
- *A solution:* continually train on longer documents
  - Upsample longer documents
  - Maintain domain mixture, and upsample long docs in each domain



RoPE Scaling

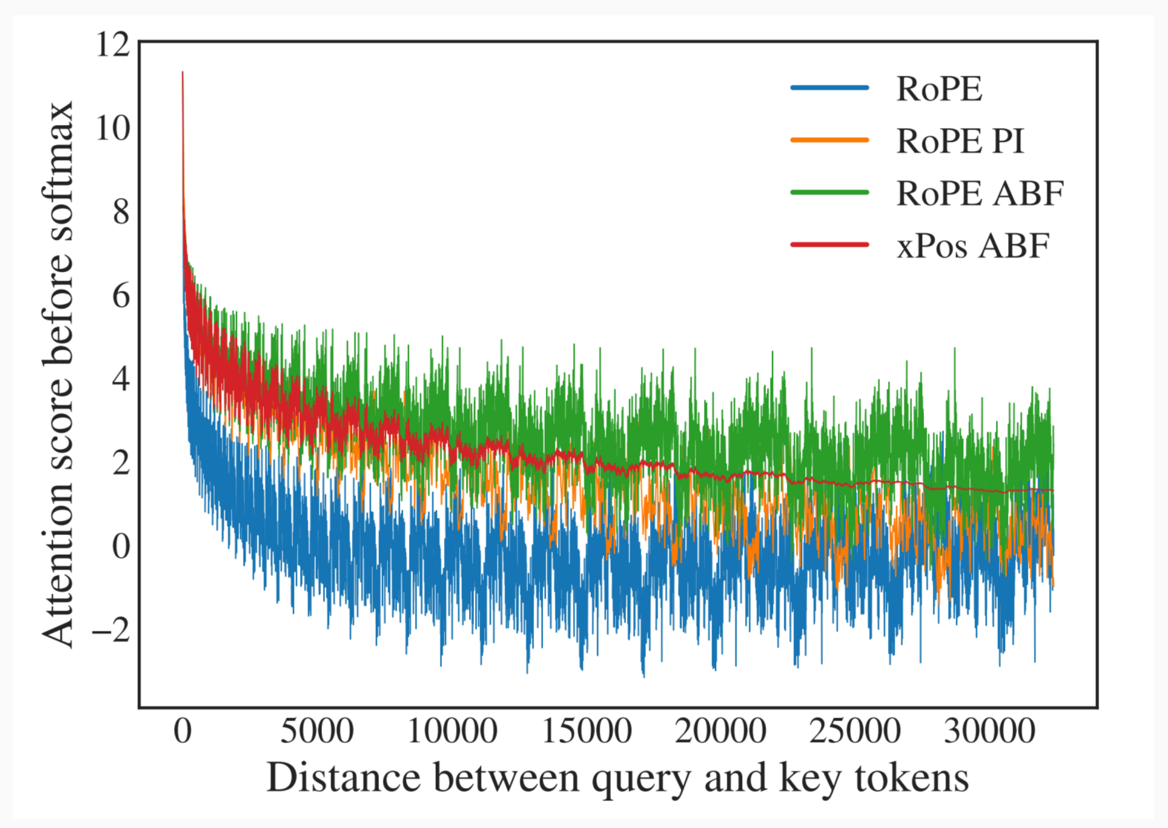
$$\mathbf{R}(\boldsymbol{\theta}, i) = \begin{pmatrix} \cos i\theta_1 & -\sin i\theta_1 & \cdots & 0 & 0 \\ \sin i\theta_1 & \cos i\theta_1 & \cdots & 0 & 0 \\ \vdots & & & & \\ 0 & 0 & \cdots & \cos i\theta_{\frac{d_k}{2}} & -\sin i\theta_{\frac{d_k}{2}} \\ 0 & 0 & \cdots & \sin i\theta_{\frac{d_k}{2}} & \cos i\theta_{\frac{d_k}{2}} \end{pmatrix}$$

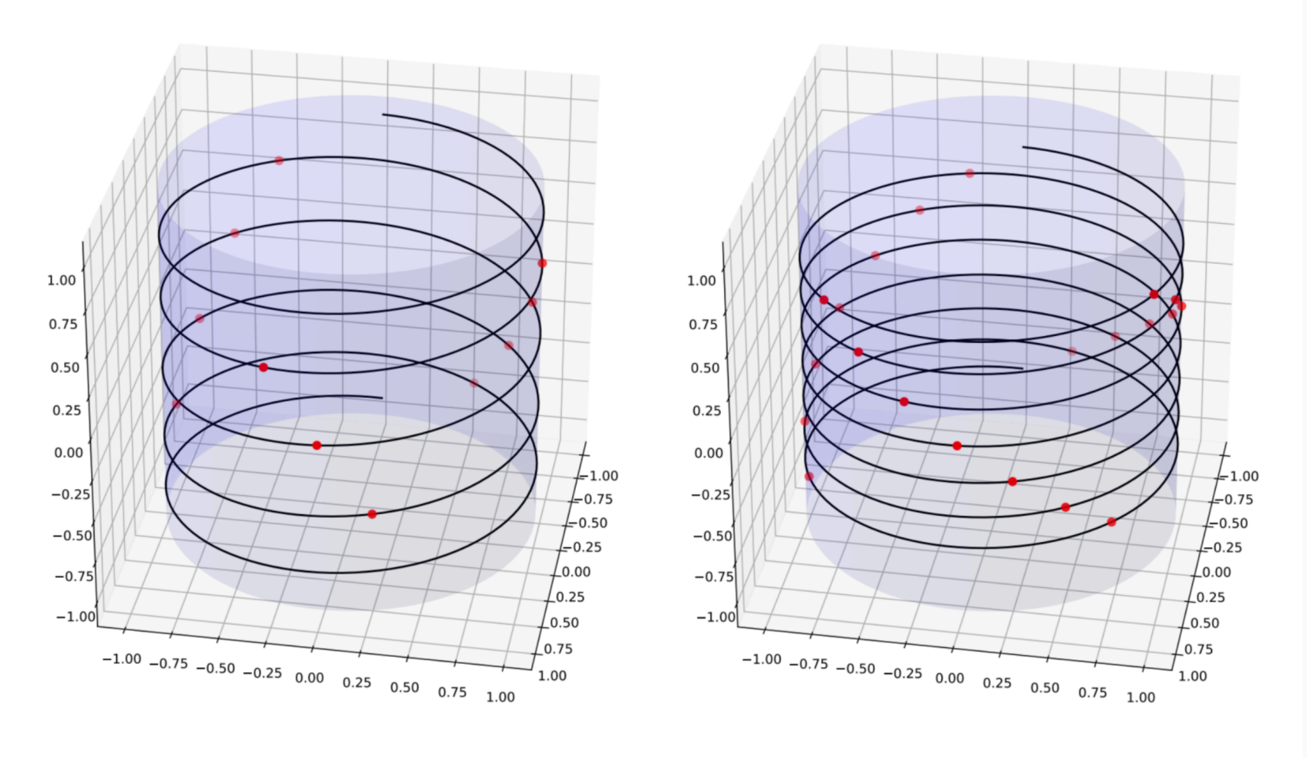
- RoPe embeddings have a periodic structure
- Parameter θ impacts the period, e.g., $\theta_j = b^{-\frac{2j}{d_k}}$ with $b = 10000$



RoPE Scaling

- Vanilla RoPE naturally decays as *s* increases
  - *Rope ABF:* Increase base frequency
  - *Position Interpolation:* Multiply period by a constant





(a) RoPE (c) RoPE+ABF

Today's lecture

- Long sequence modeling
- Improving transformers
  - Memory-efficient computation
  - Extrapolation
  - **Transformer modifications**
- Transformer alternatives

Transformer modifications

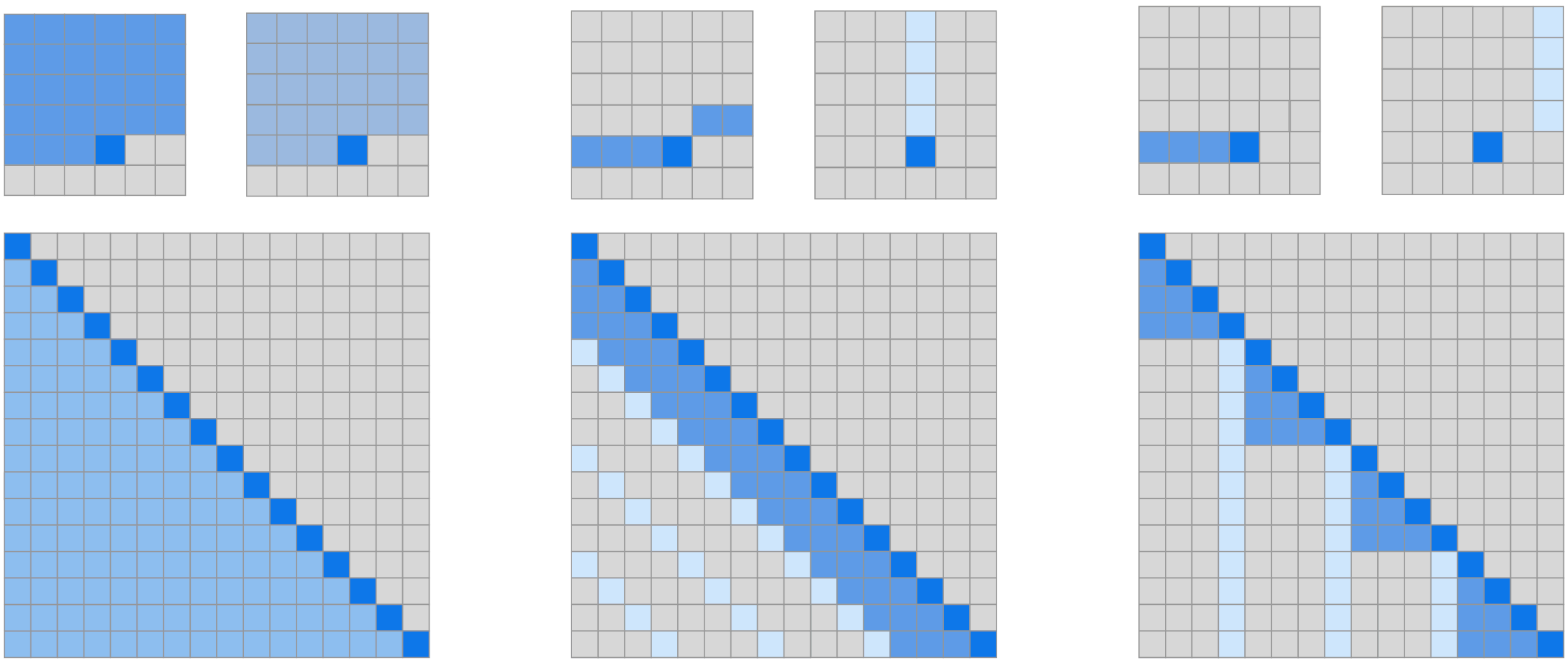
Skipped in lecture due to time

- Sparse Attention
- Sliding Window Attention
- Compression
- Low-rank Approximation

Skipped in lecture due to time

Sparse Transformers
(Child et al. 2019)

- Add "stride", only attending to every *n* previous states



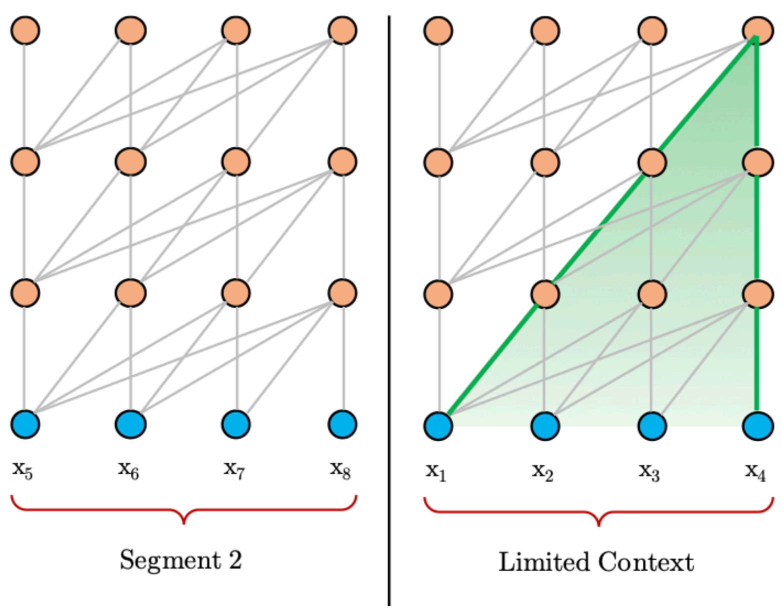
(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Skipped in lecture due to time

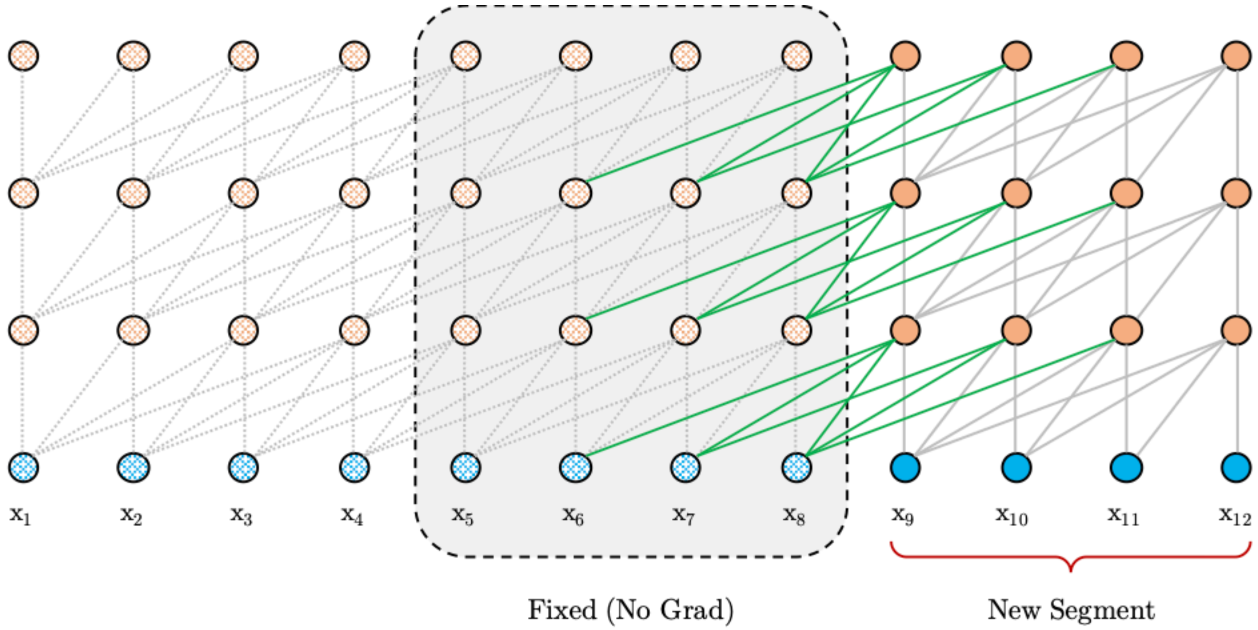
Truncated BPTT+Transformer

- Transformer-XL (Dai et al. 2019) attends to fixed **vectors** from the previous sentence

Standard Transformer



Transformer-XL

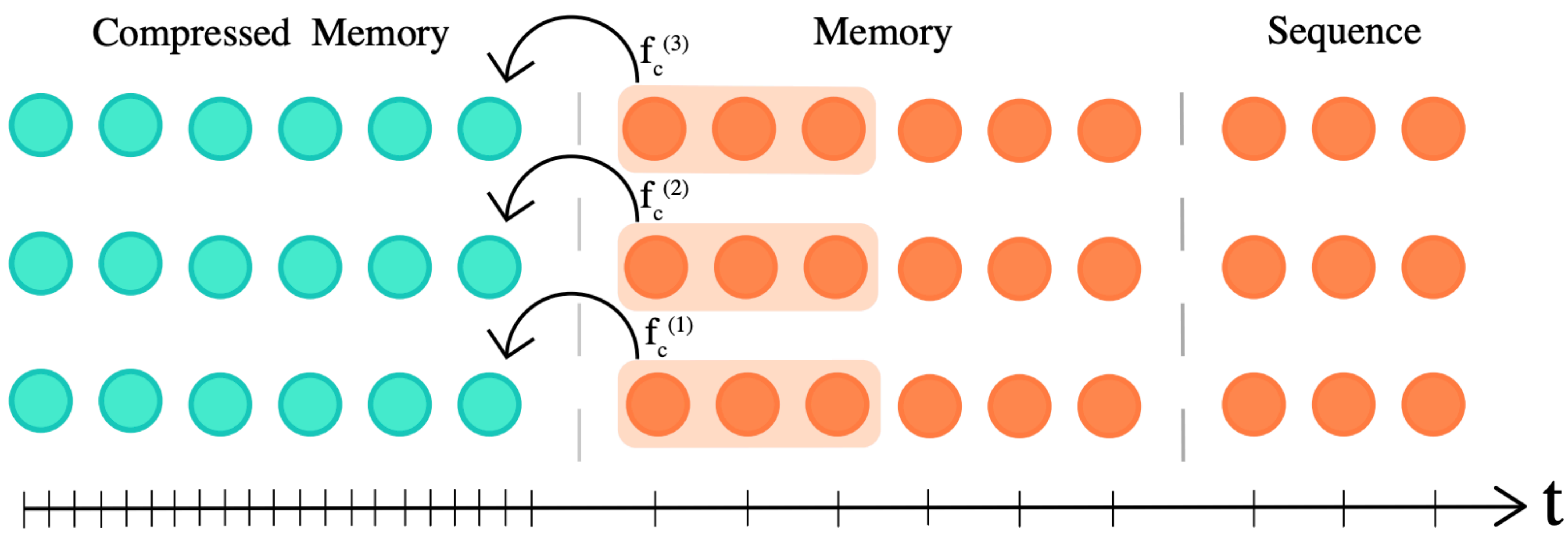


- Like truncated backprop through time for RNNs; can use previous states, but not backprop into them
- See also Mistral's (Jiang et al. 2023) sliding window attention

Skipped in lecture due to time

Compressing Previous States

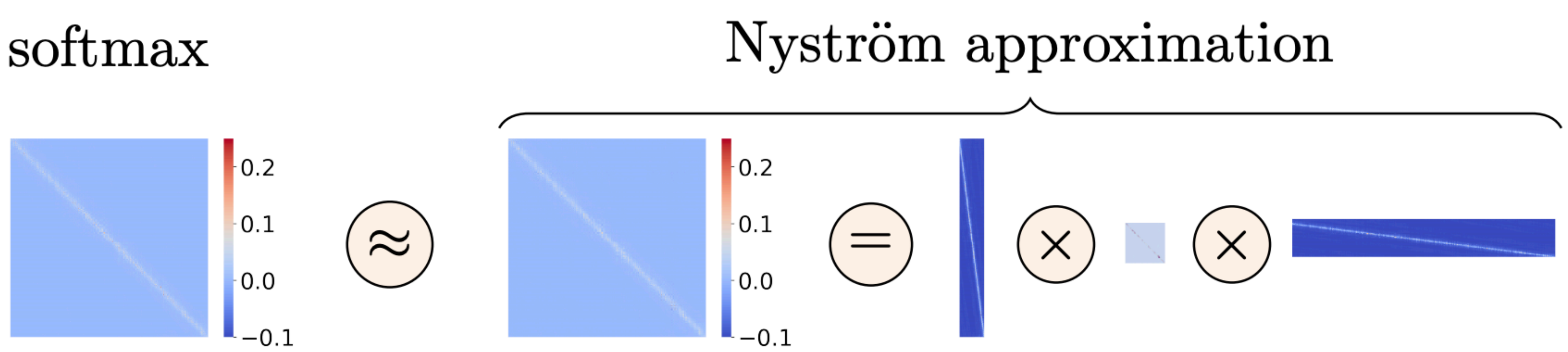
- Add a "strided" compression step over previous states (Rae et al. 2019)



Skipped in lecture due to time

Low-rank Approximation

- Calculating the attention matrix is expensive, can it be predicted with a low-rank matrix?
- **Linformer:** Add low-rank linear projections into model (Wang et al. 2020)
- **Nystromformer:** Approximate using the Nystrom method, sampling "landmark" points (Xiong et al. 2021)

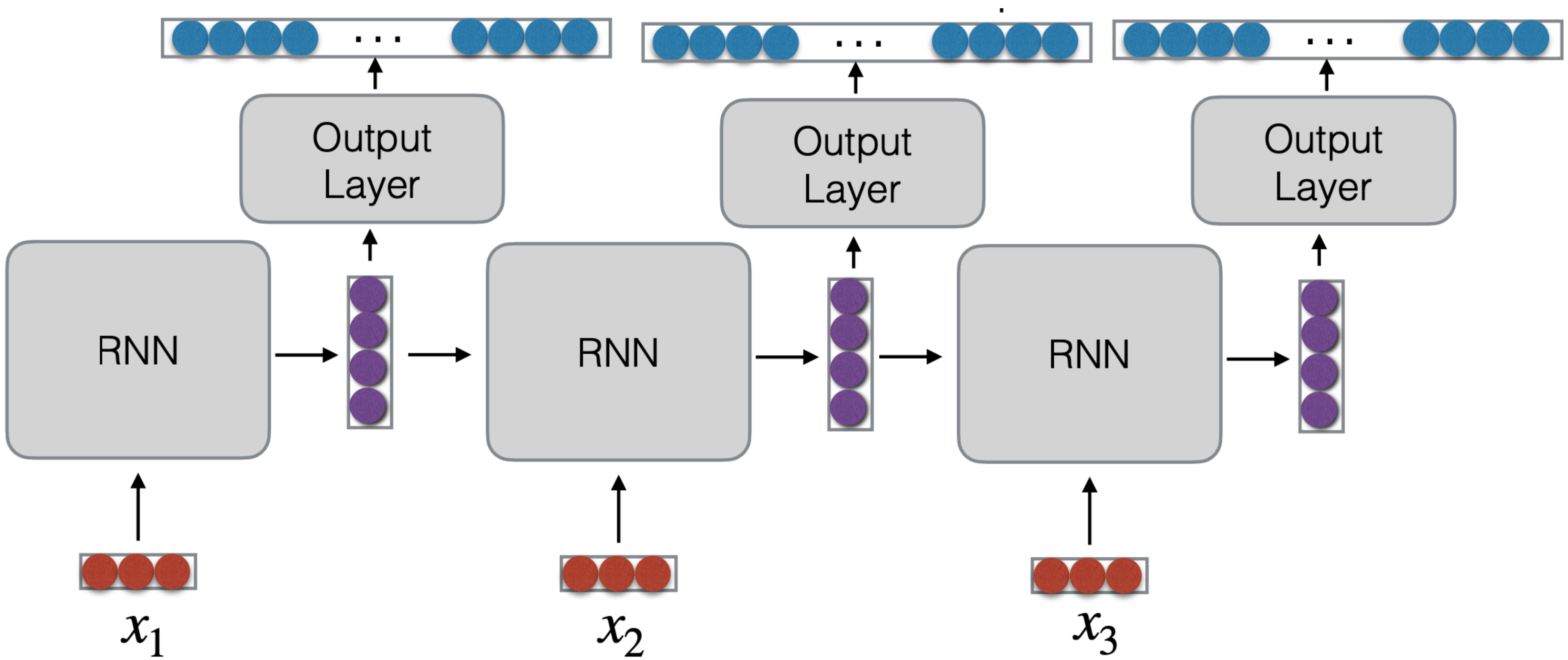


Today's lecture

- Long sequence modeling
- Improving transformers
- **Transformer alternatives**
    - State-space models

Reminder: RNNs

Lecture 4

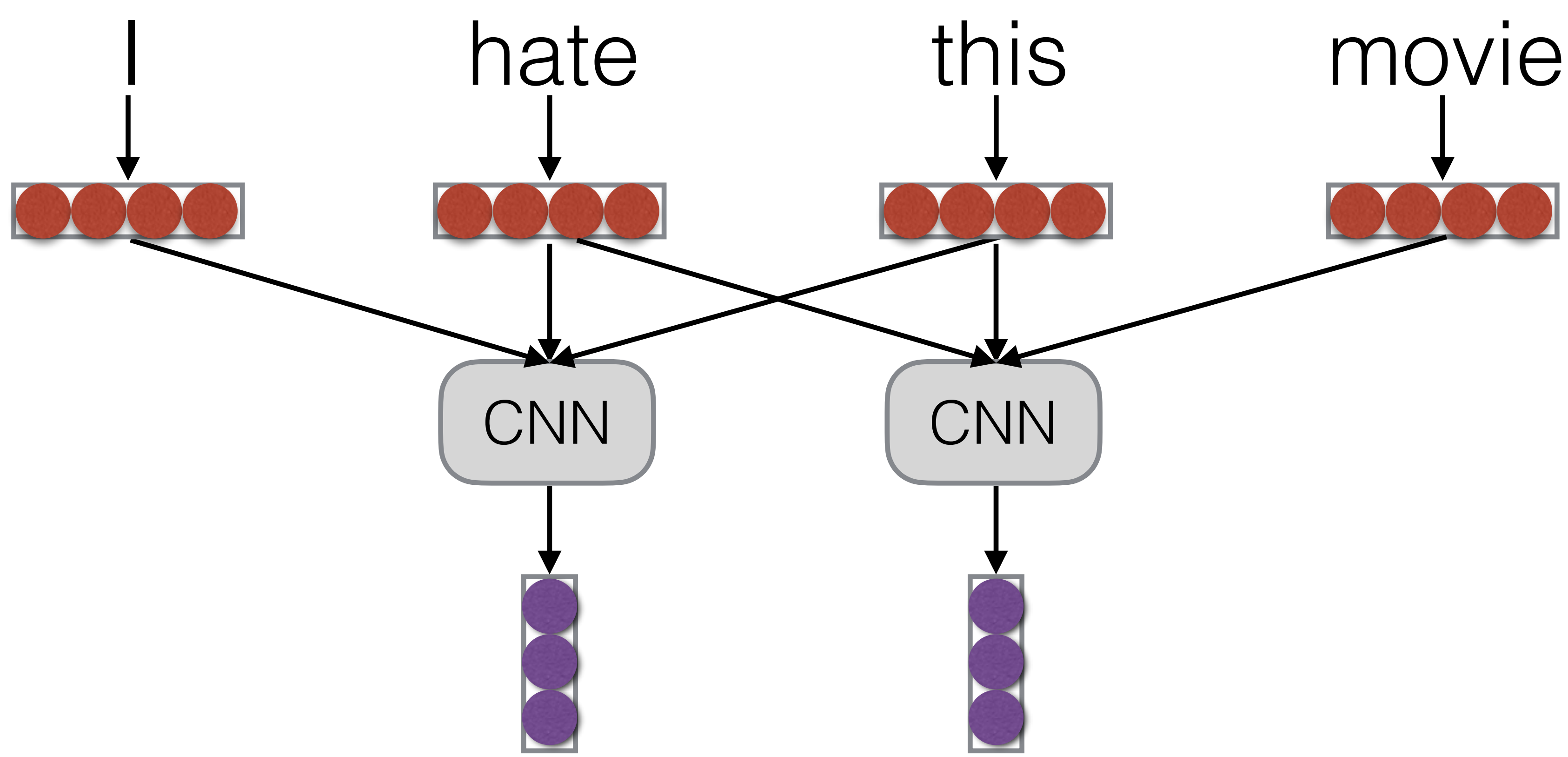


- Infinite context
- Memory-efficient inference (single hidden state)

- Hard to parallelize training

Convolution

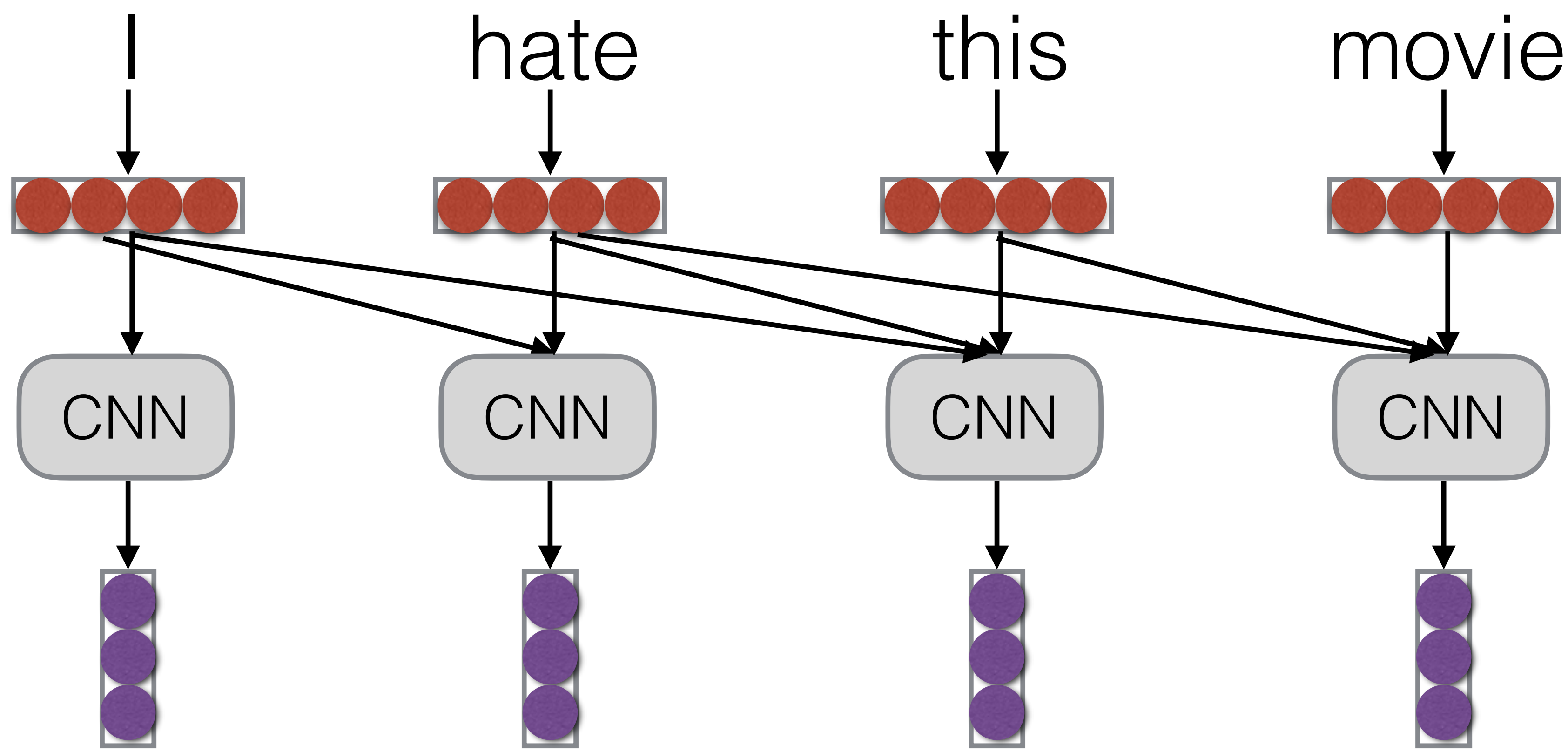
- Calculate based on local context



$$h_t = \mathrm{f}(W[x_{t-1}; x_t; x_{t+1}])$$

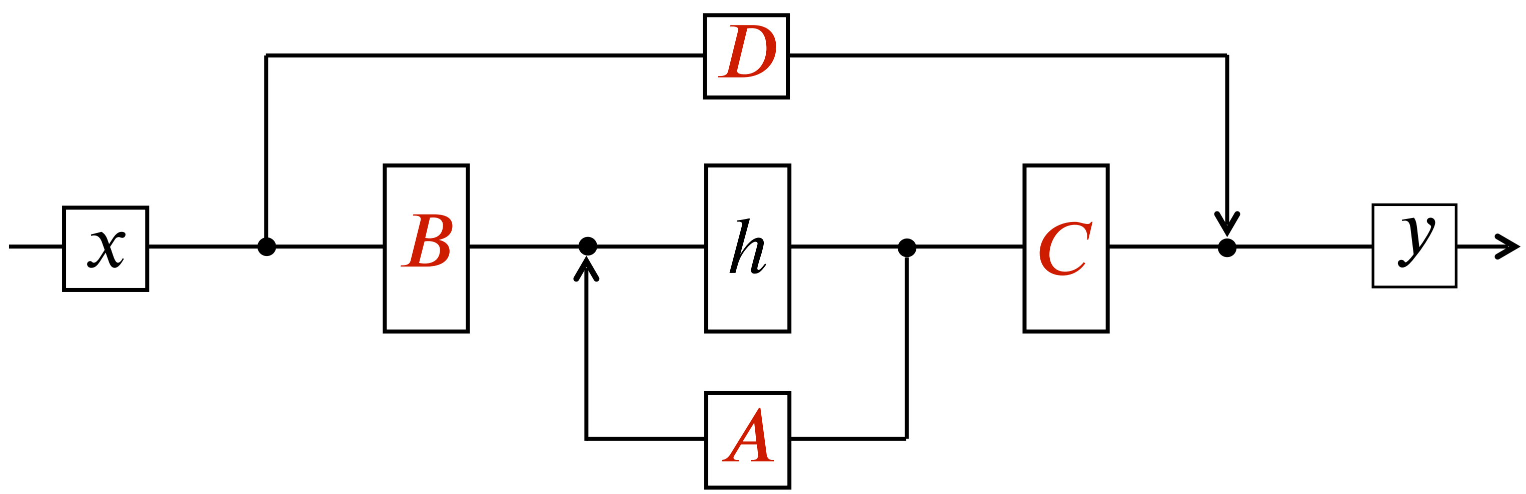
Convolution for Auto-regressive Models

- Functionally identical, just consider previous context



Structured State Space Models
(Gu et al. 2021)

- Models that take a form like the following

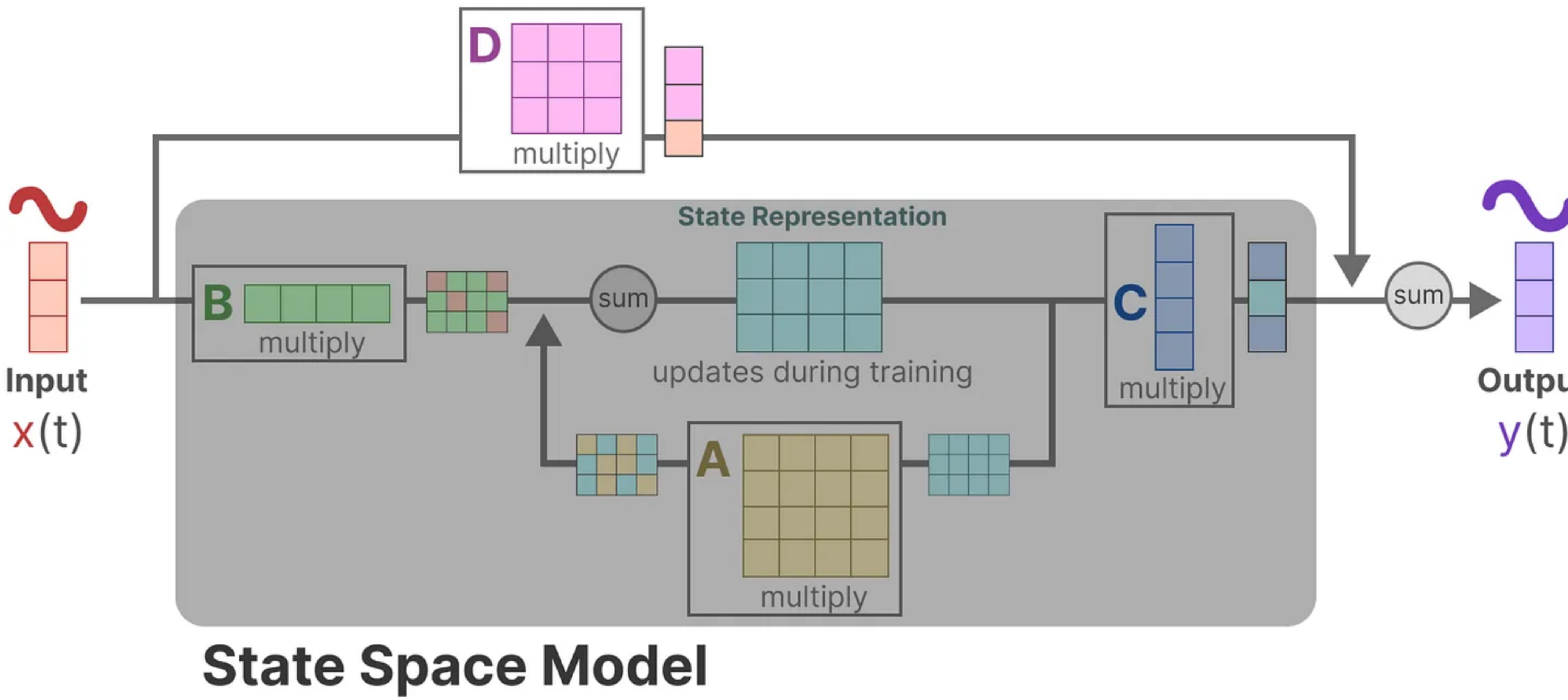


$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t)$$
$$y(t) = \boldsymbol{C}h(t) + \boldsymbol{D}x(t)$$

- **Key idea**: we can compute *h(t)* in parallel (fast training), or compute *h(t)* like in an RNN (fast inference)

Slide Credit: Albert Gu

Structured State Space Models



$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t)$$
$$y(t) = \boldsymbol{C}h(t) + \boldsymbol{D}x(t)$$

Figure: A Visual Guide to Mamba

Structured State Space Models

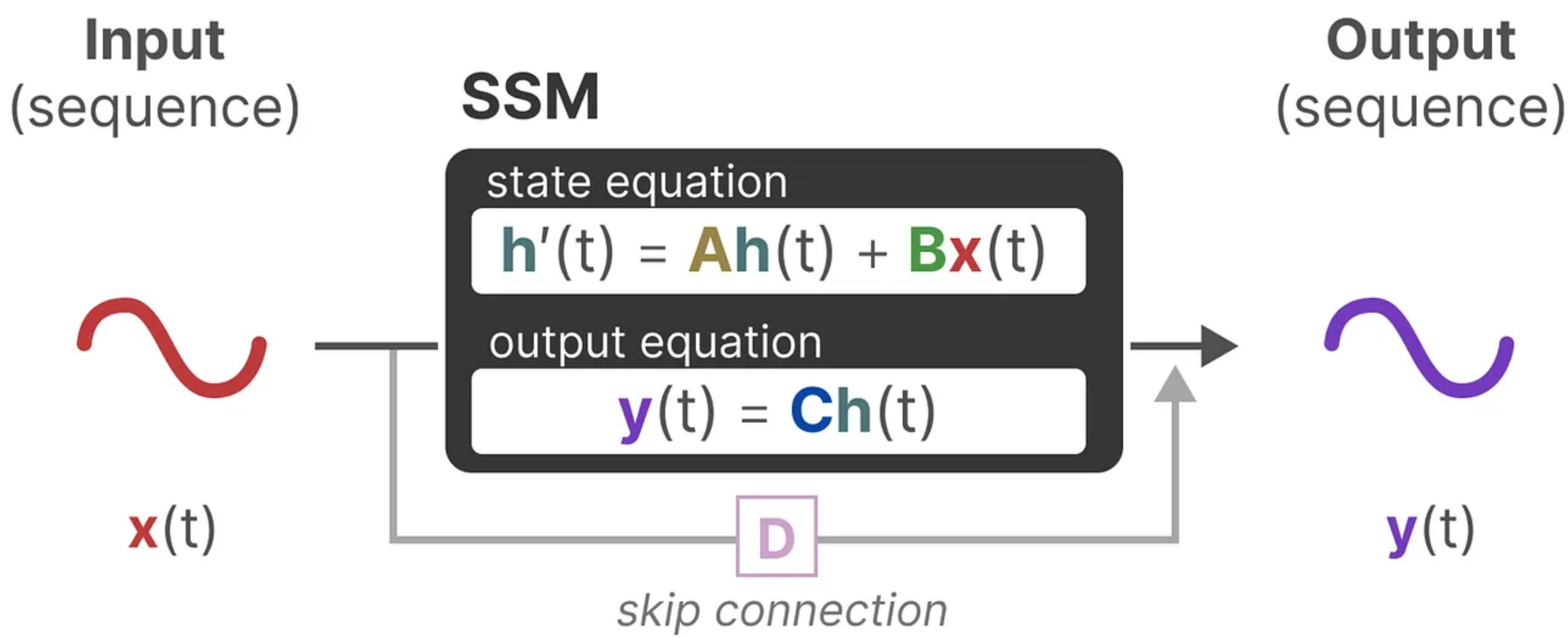
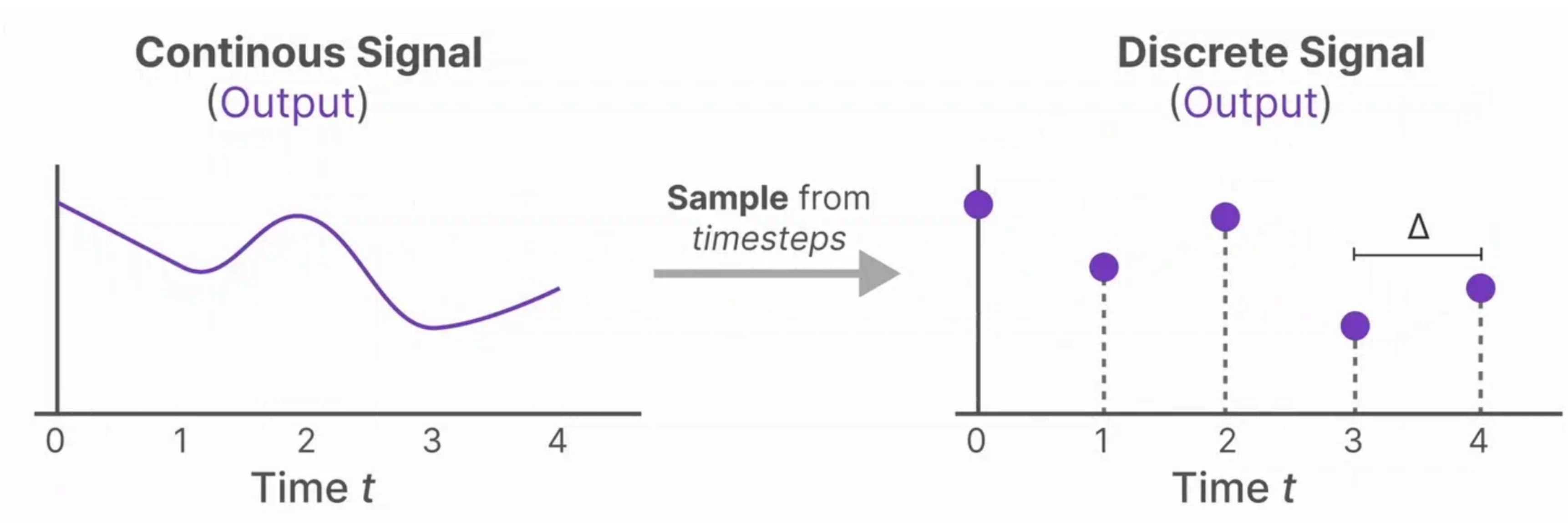
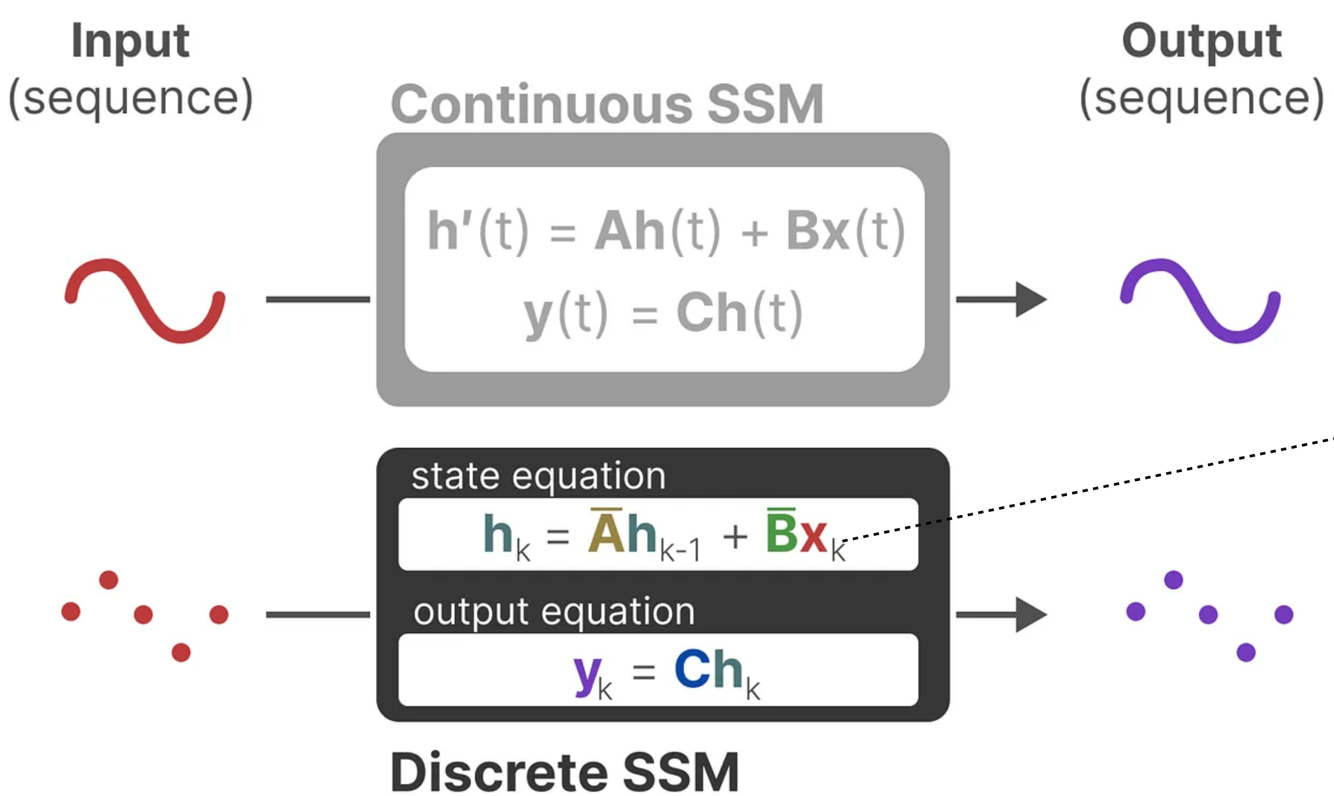


Figure: A Visual Guide to Mamba

SSMs: Discretization



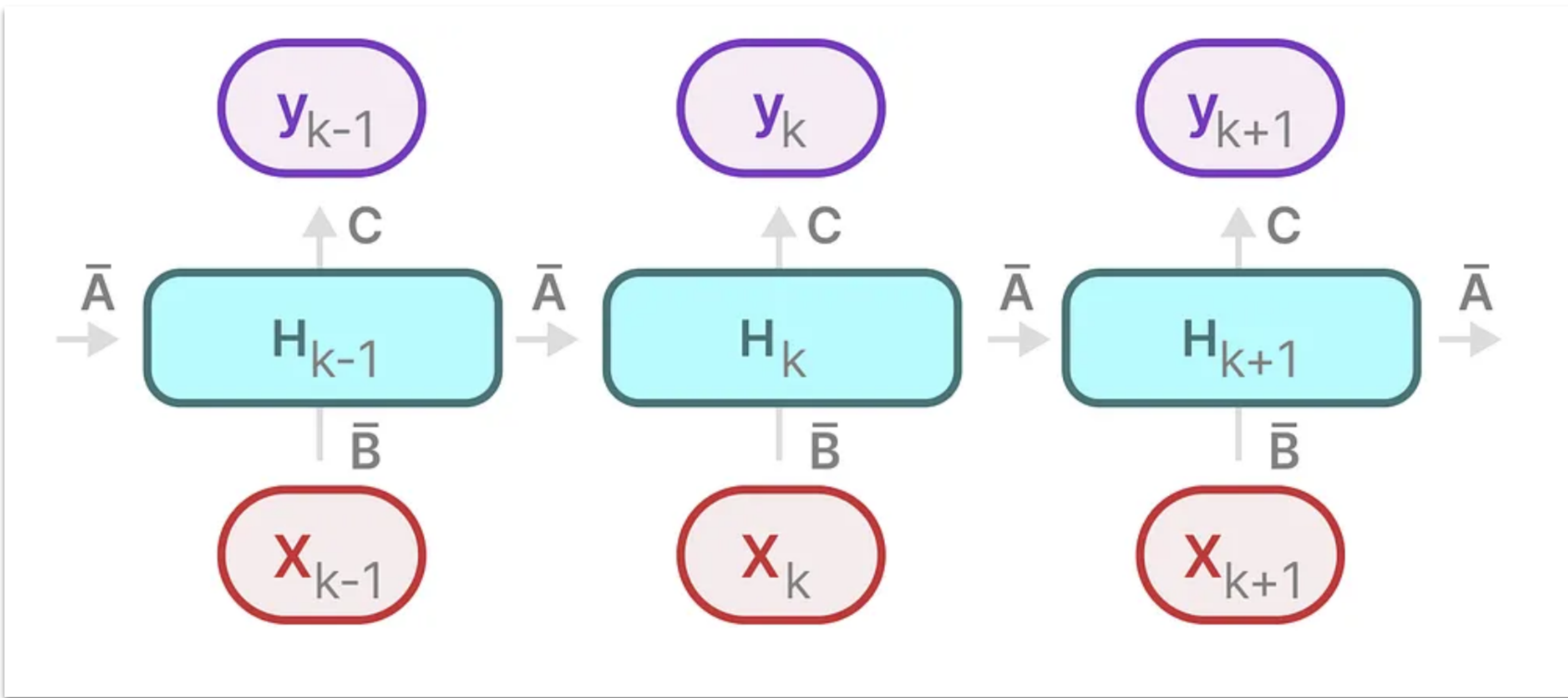


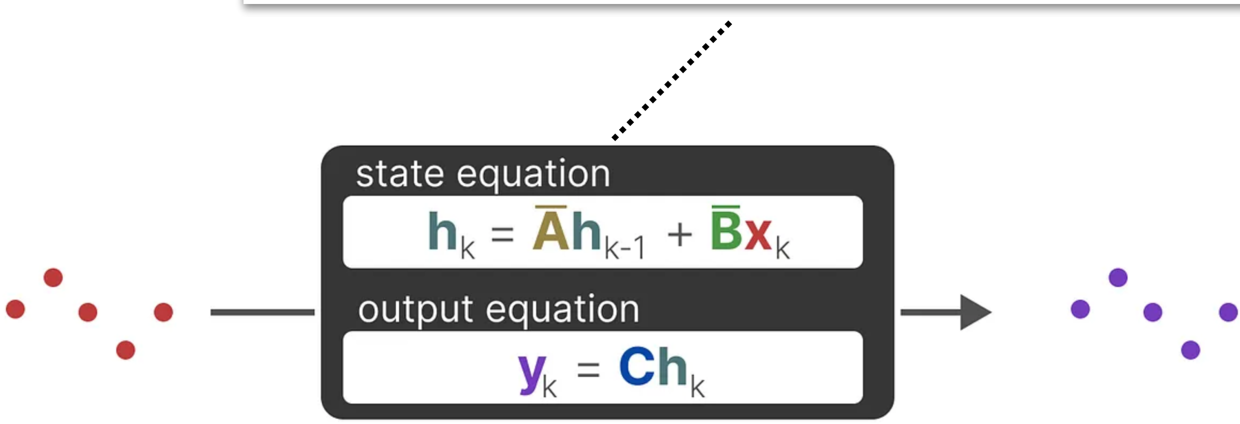
$$\bar{\mathbf{A}} = \exp(\Delta \mathbf{A})$$

$$\bar{\mathbf{B}} = (\Delta \mathbf{A})^{-1}(\exp(\Delta \mathbf{A}) - I) \cdot \Delta \mathbf{B}$$

Figure: A Visual Guide to Mamba

SSMs: Recurrent View



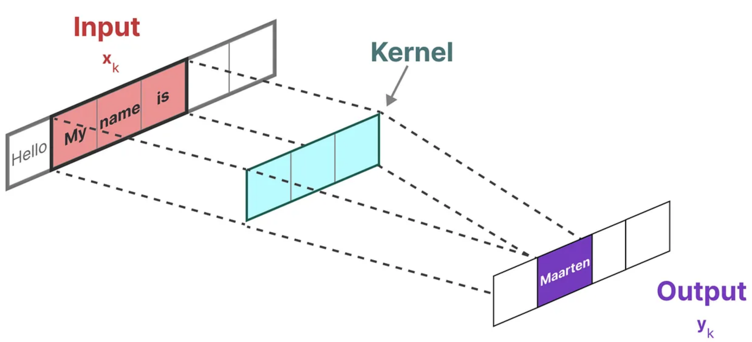


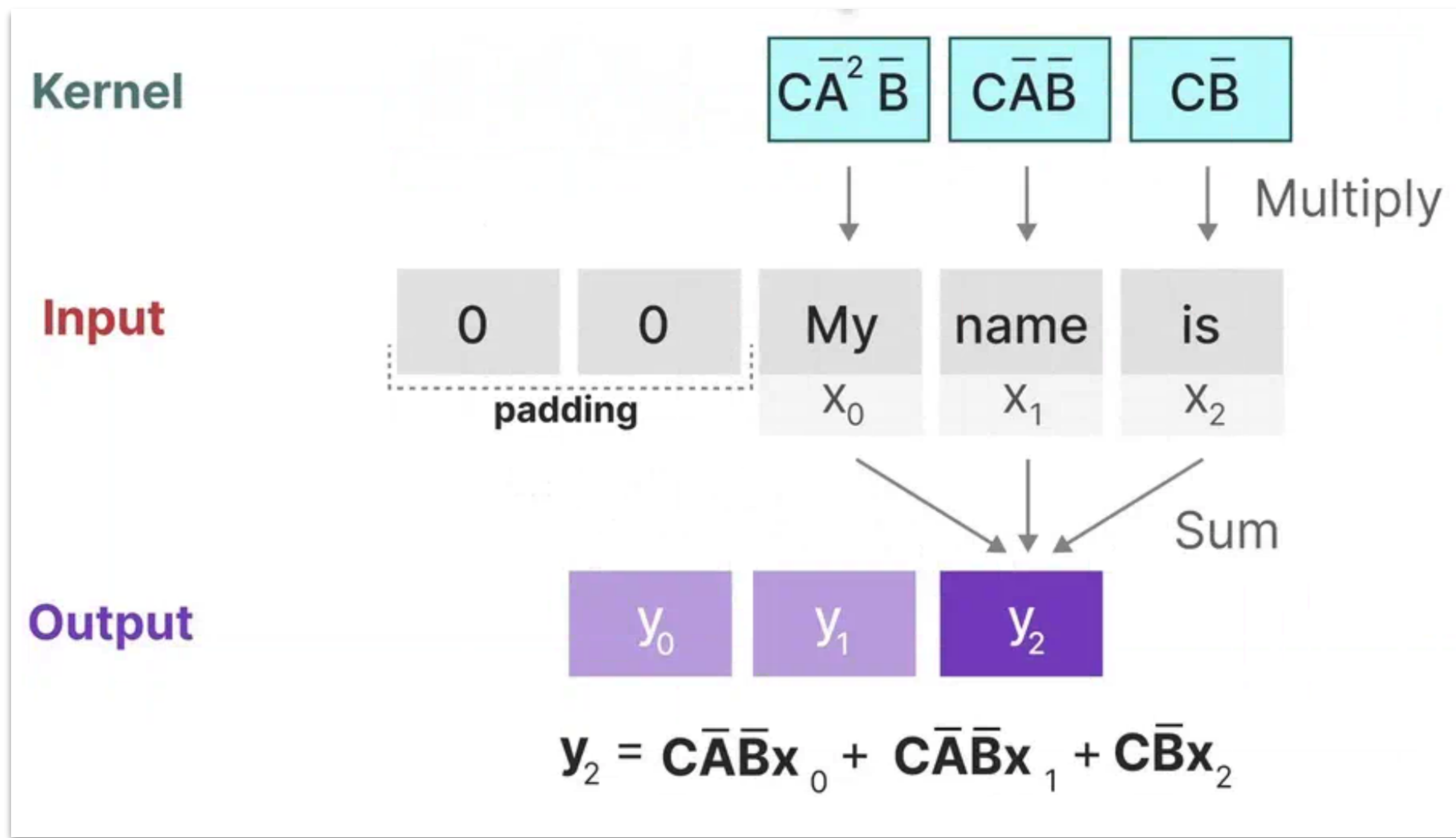
Efficient inference

Parallelizable training

Figure: A Visual Guide to Mamba

SSM: Convolution View





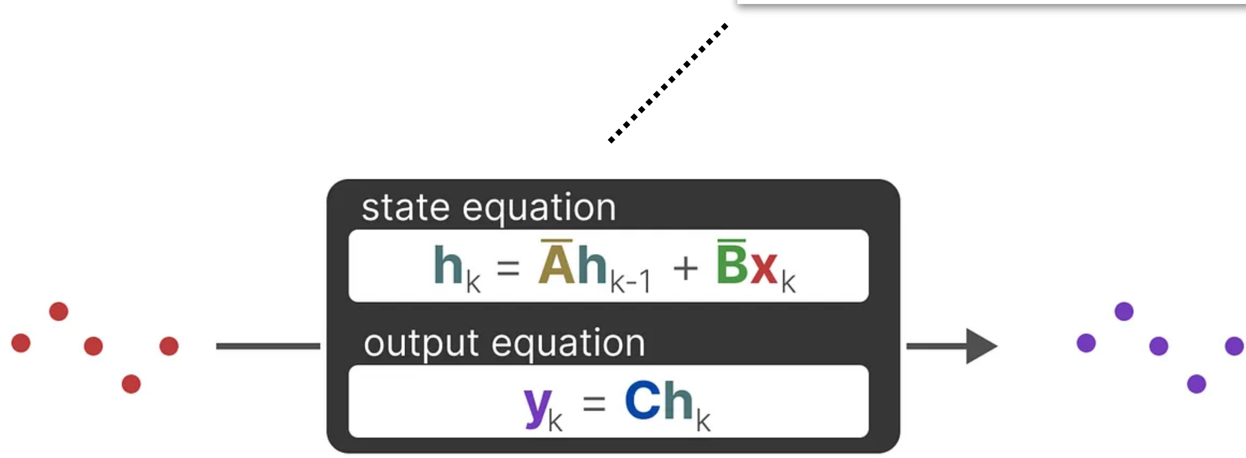


Figure: A Visual Guide to Mamba

SSM: Convolution View

$$\textit{kernel} \longrightarrow \overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}}\overline{\mathbf{B}}, \dots, \mathbf{C}\overline{\mathbf{A}}^{k}\overline{\mathbf{B}}, \dots)$$

$$\underset{\text{output}}{\mathbf{y}} = \underset{\text{input}}{\mathbf{x}} * \underset{\text{kernel}}{\overline{\mathbf{K}}}$$

- ✔ Parallelizable training
- ✘ Unbounded context

Figure: A Visual Guide to Mamba

SSM Variants

- **S4**: Discrete SSM with a structured form of the recurrent update matrix *A* (“HiPPO”) for better memory retention

Efficiently Modeling Long Sequences with Structured State Spaces

Albert Gu, Karan Goel, and Christopher Ré
Department of Computer Science, Stanford University
{albertgu,krng}@stanford.edu, chrismre@cs.stanford.edu

- **Mamba**: S4 + *selectively* retain information

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu*[1] and Tri Dao*[2]
[1]Machine Learning Department, Carnegie Mellon University
[2]Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

S4: SSM + Structured Matrix

- **S4**: Discrete SSM with a structured form of the recurrent update matrix *A* (“HiPPO”) for better memory retention

---

Algorithm 1 SSM (S4)

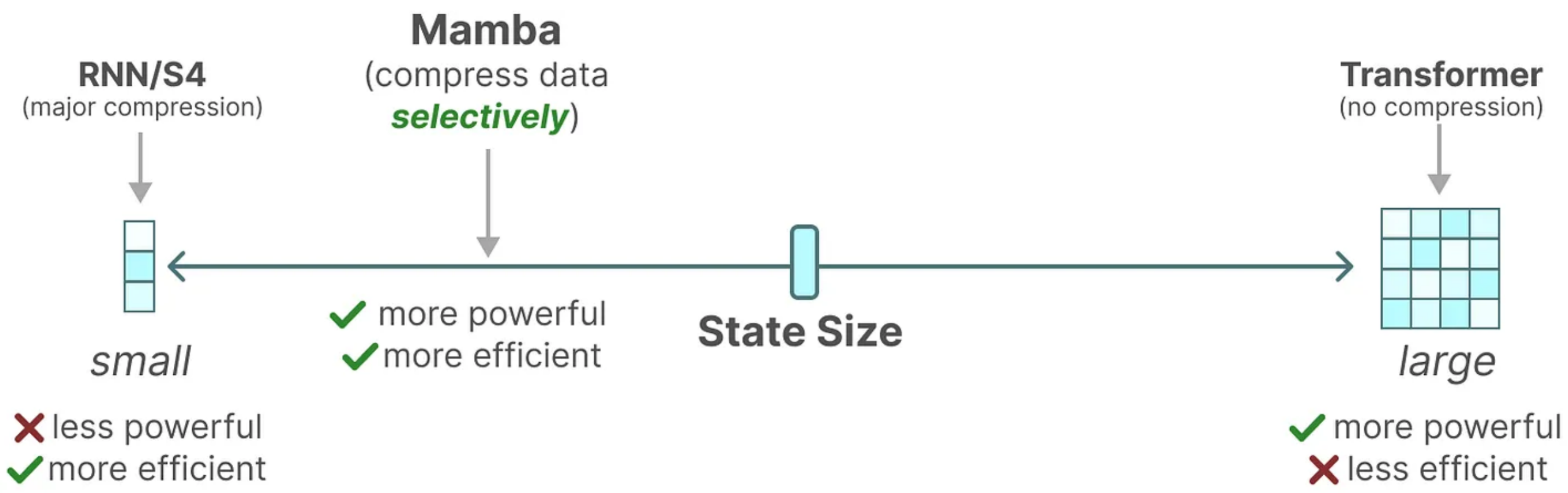
Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$

$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\Delta : (\mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter})$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{D}, \mathtt{N}) \leftarrow \text{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \text{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
return y

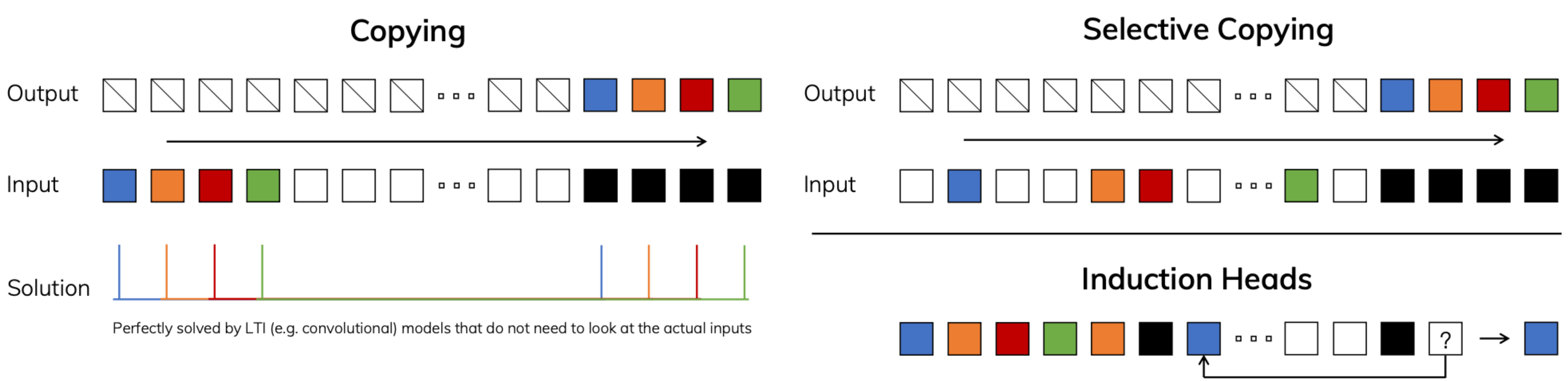
$$\overline{\boldsymbol{A}} = \exp(\Delta \boldsymbol{A})$$

$$\overline{\boldsymbol{B}} = (\Delta \boldsymbol{A})^{-1}(\exp(\Delta \boldsymbol{A}) - \boldsymbol{I}) \cdot \Delta \boldsymbol{B}$$

Mamba: Selective SSMs



Mamba: Selective SSMs



Mamba: Selective SSMs

- **S6**: S4 with time-varying parameters (B,C,$\boldsymbol{\Delta}$)

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
1: $\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_B(x)$
3: $\boldsymbol{C} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_C(x)$
4: $\Delta : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter} + s_\Delta(x))$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-varying: recurrence (*scan*) only
7: **return** y

Mamba: Selective SSMs

- Parallel scan algorithm due to sequential computation

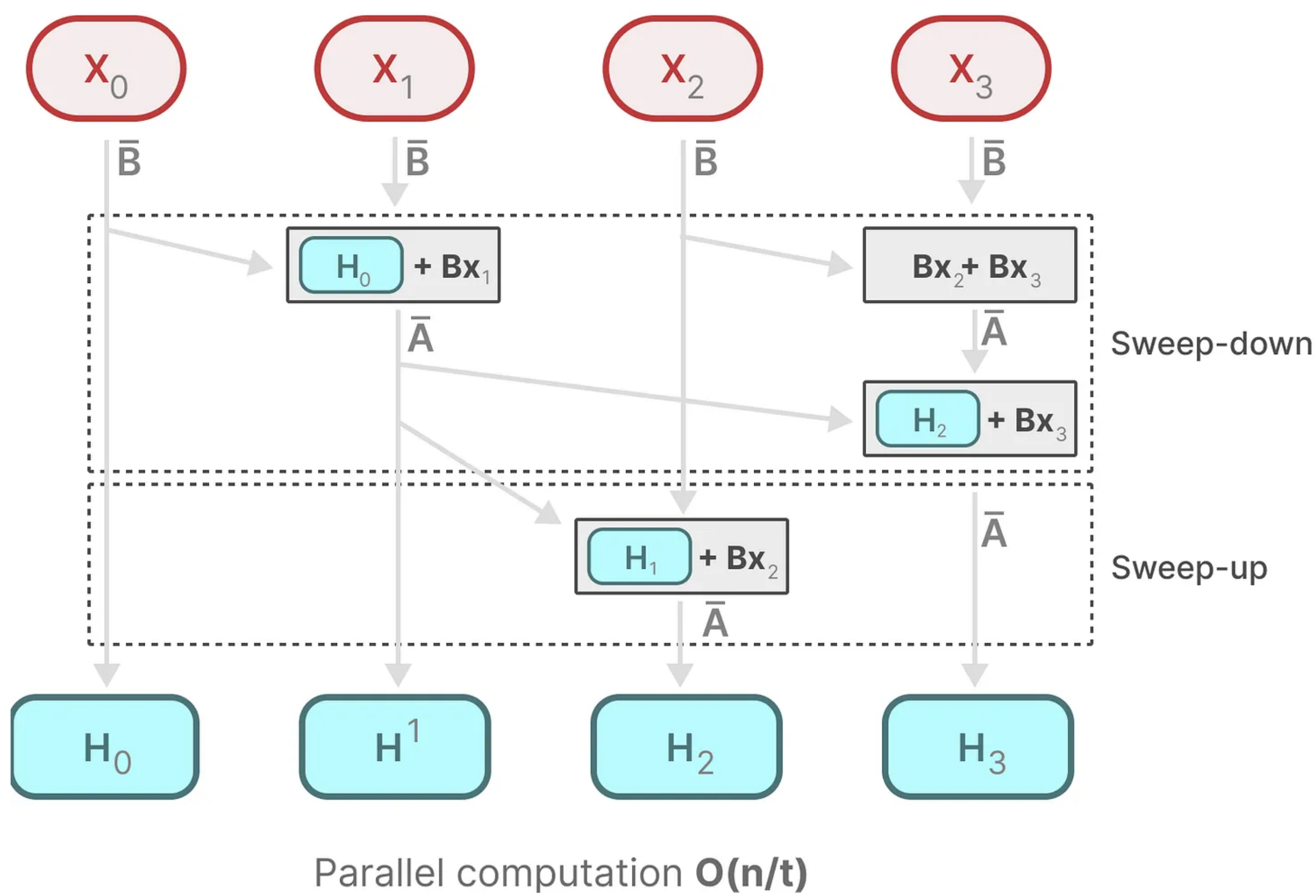
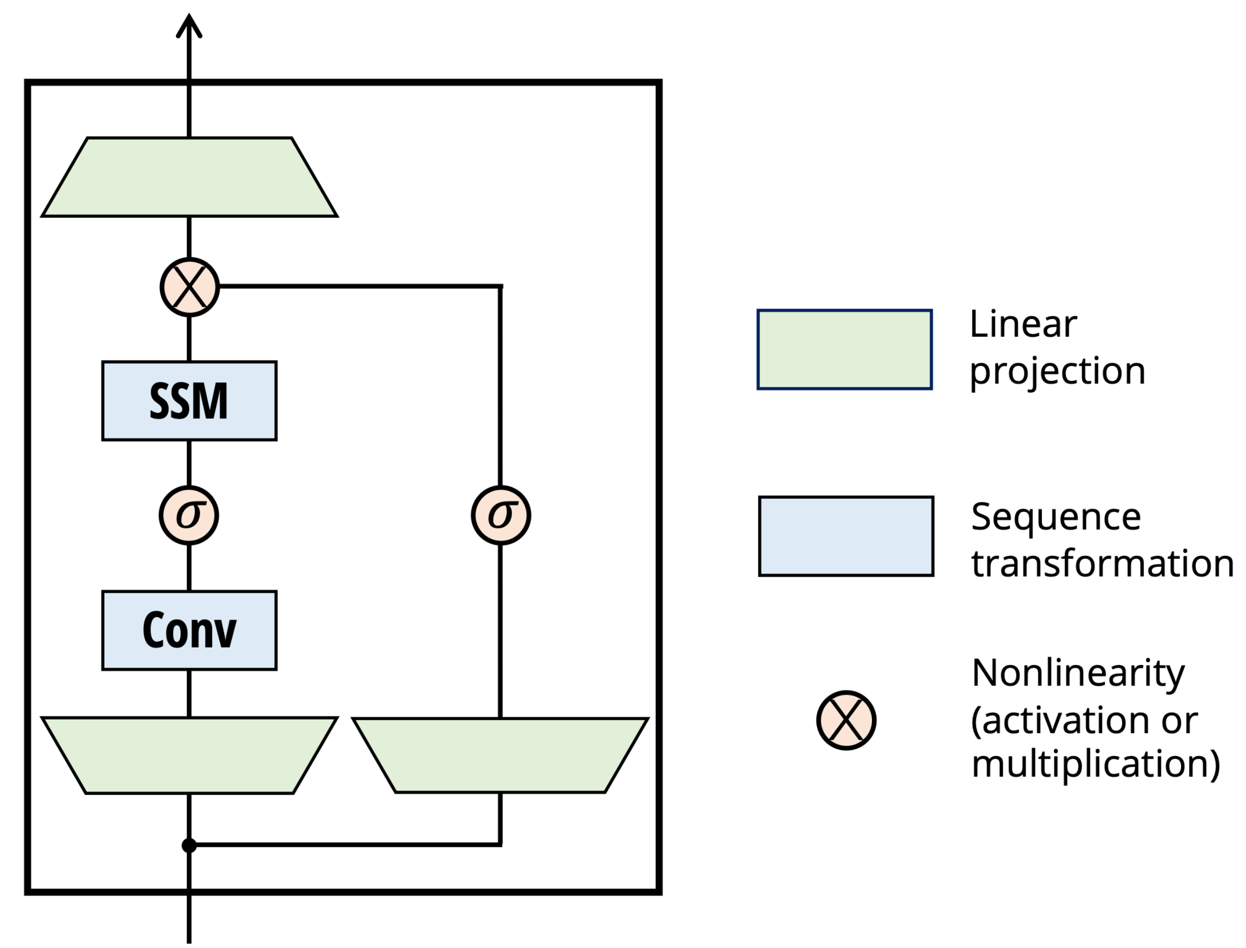


Figure: A Visual Guide to Mamba

Mamba

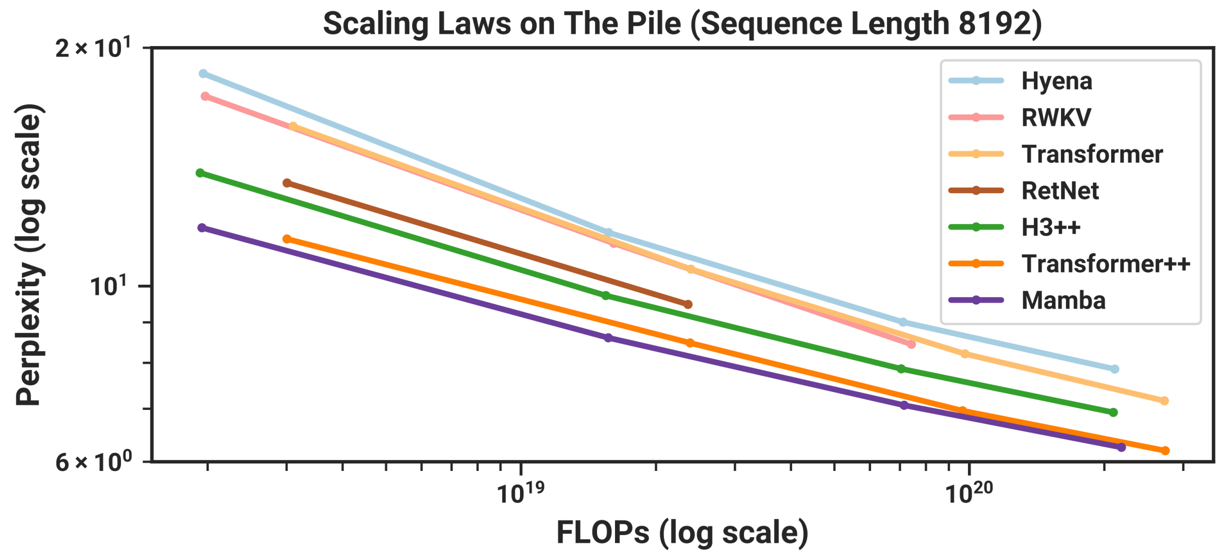
- Block/layer that incorporates S6



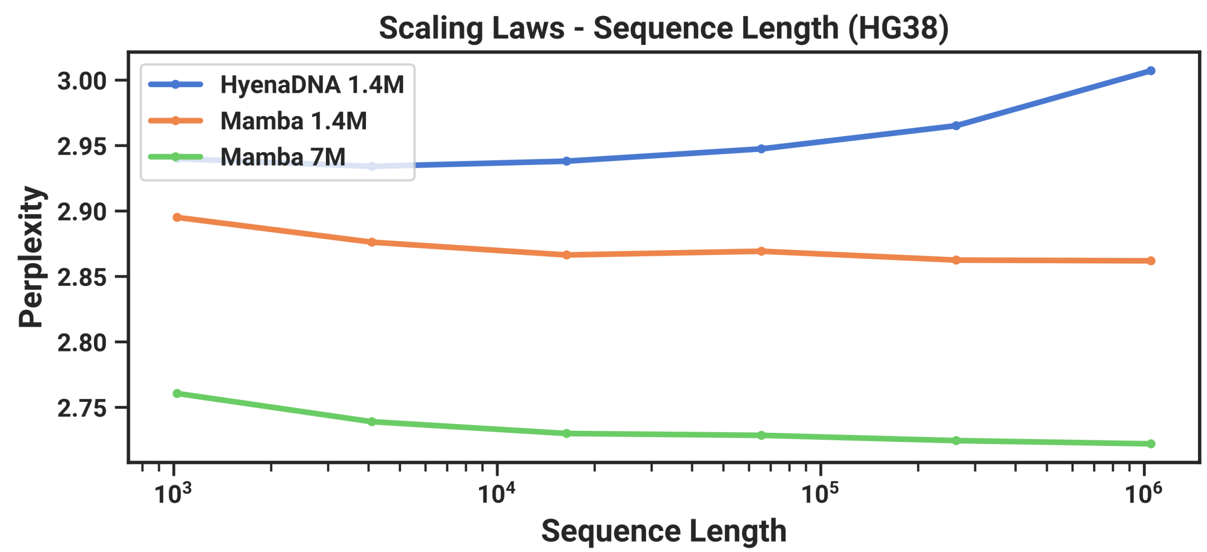
→ **Mamba**

Mamba

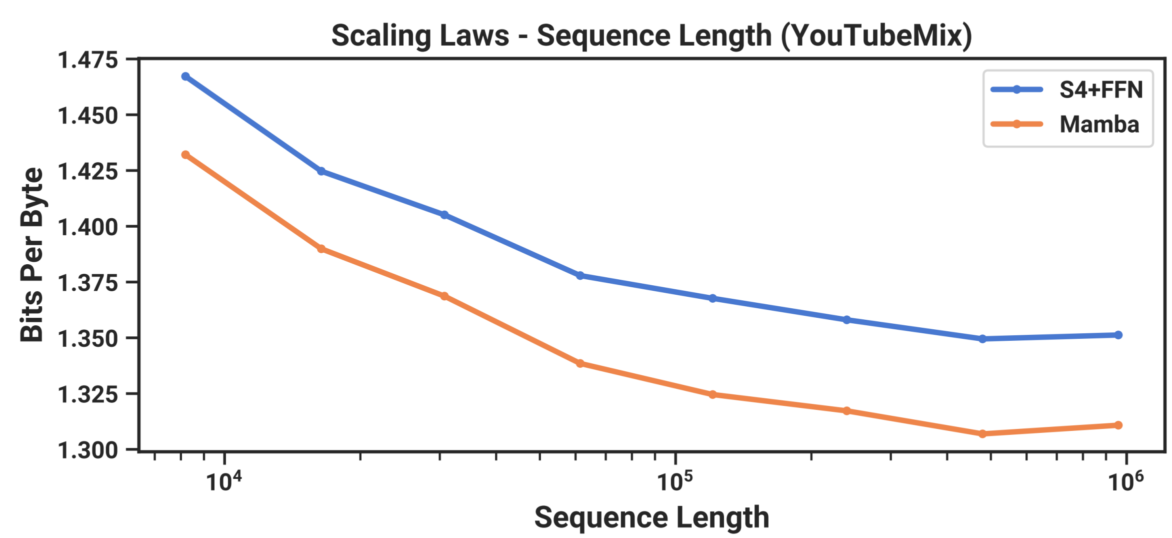
Text Data



Human Genome



Audio



Recap: SSMs

- Combine insights from recurrent models and convolutional models
  - Enables efficient training and inference
    - Scales linearly in sequence length

Recap: long context models

- Long sequence modeling
- Improving transformers
  - Memory efficient computation
  - Extrapolation: training and embeddings
- Transformer alternatives
  - State-space models

Questions?