

CS11-711 Advanced NLP

Scaling Sequence Length

Sean Welleck

**Carnegie
Mellon
University**



Some 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

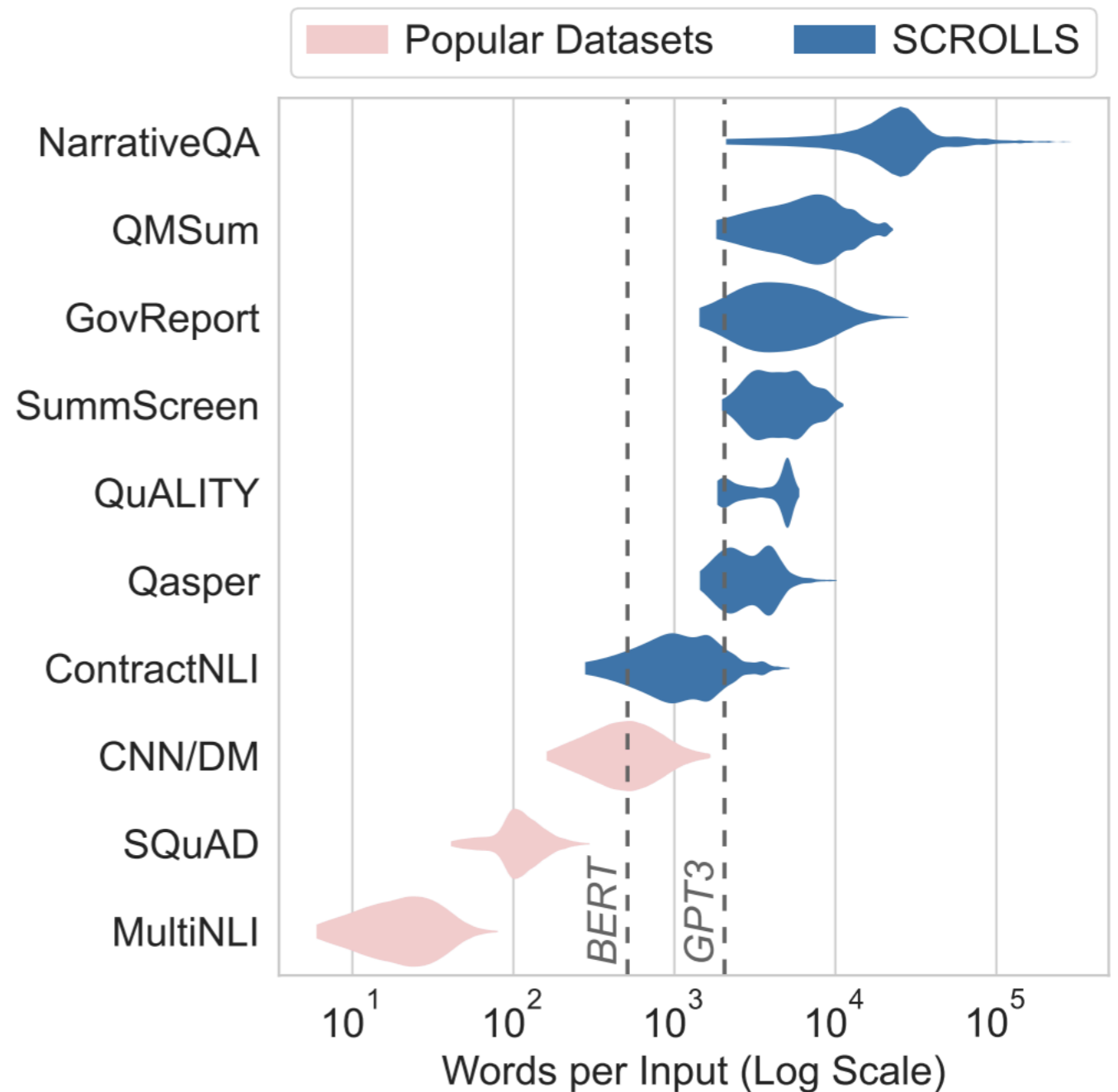
Why is Modeling Long Sequences Hard?

- **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

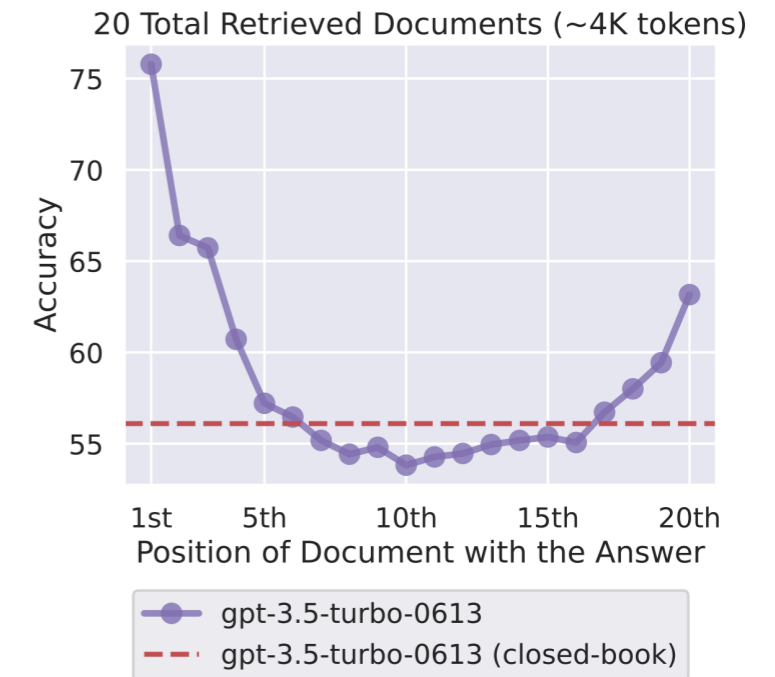
Early benchmarks for long-context models

- **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)



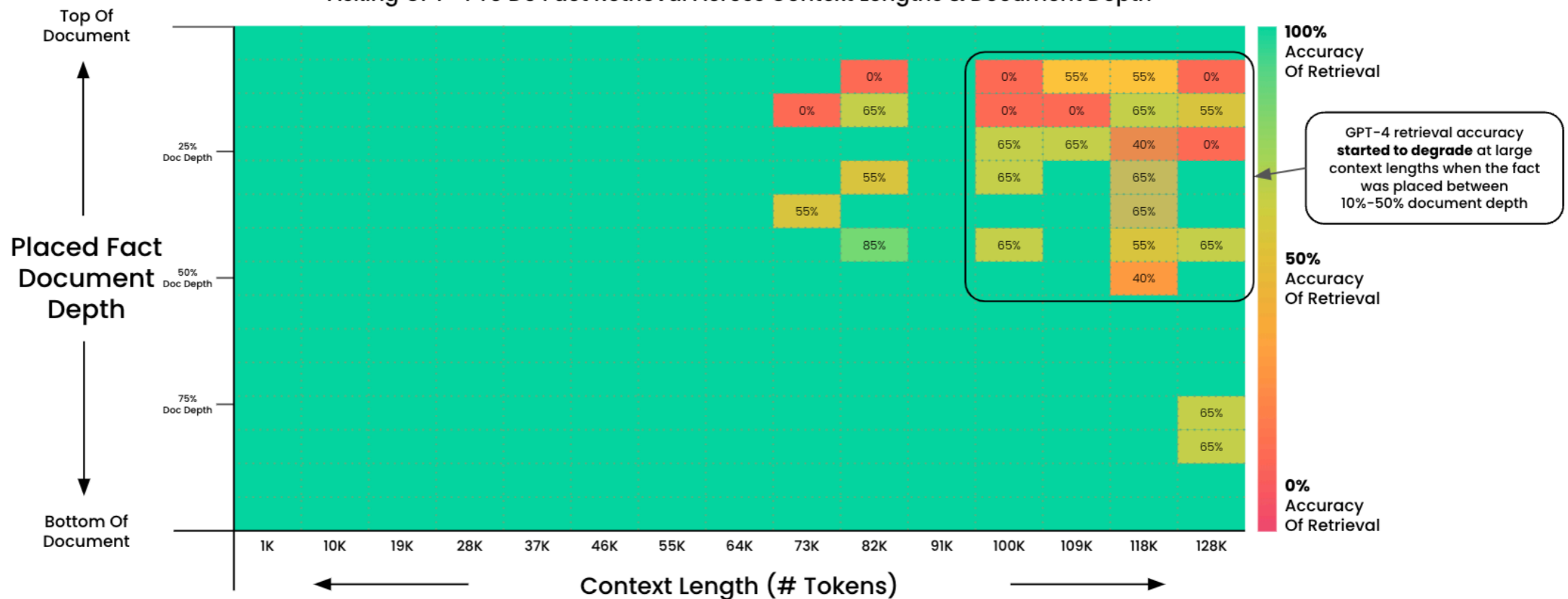
Targeted Analysis Tools

- “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



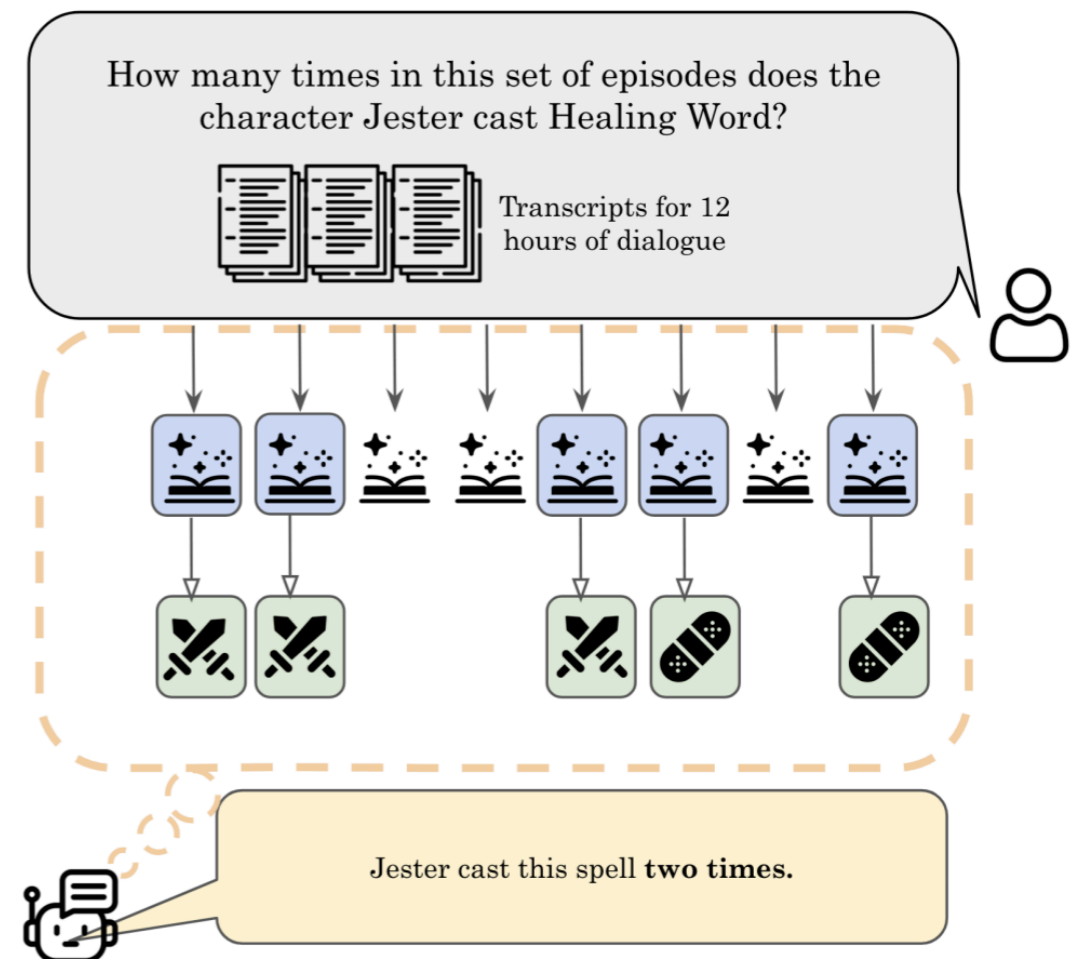
Pressure Testing GPT-4 128K via "Needle In A HayStack"

Asking GPT-4 To Do Fact Retrieval Across Context Lengths & Document Depth



Oolong [Bertsch et al 2025]

- Tests the ability to aggregate multiple pieces of information
- Example: in a transcript from a Dungeons and Dragons show, how many times did a character cast a spell



Oolong [Bertsch et al 2025]

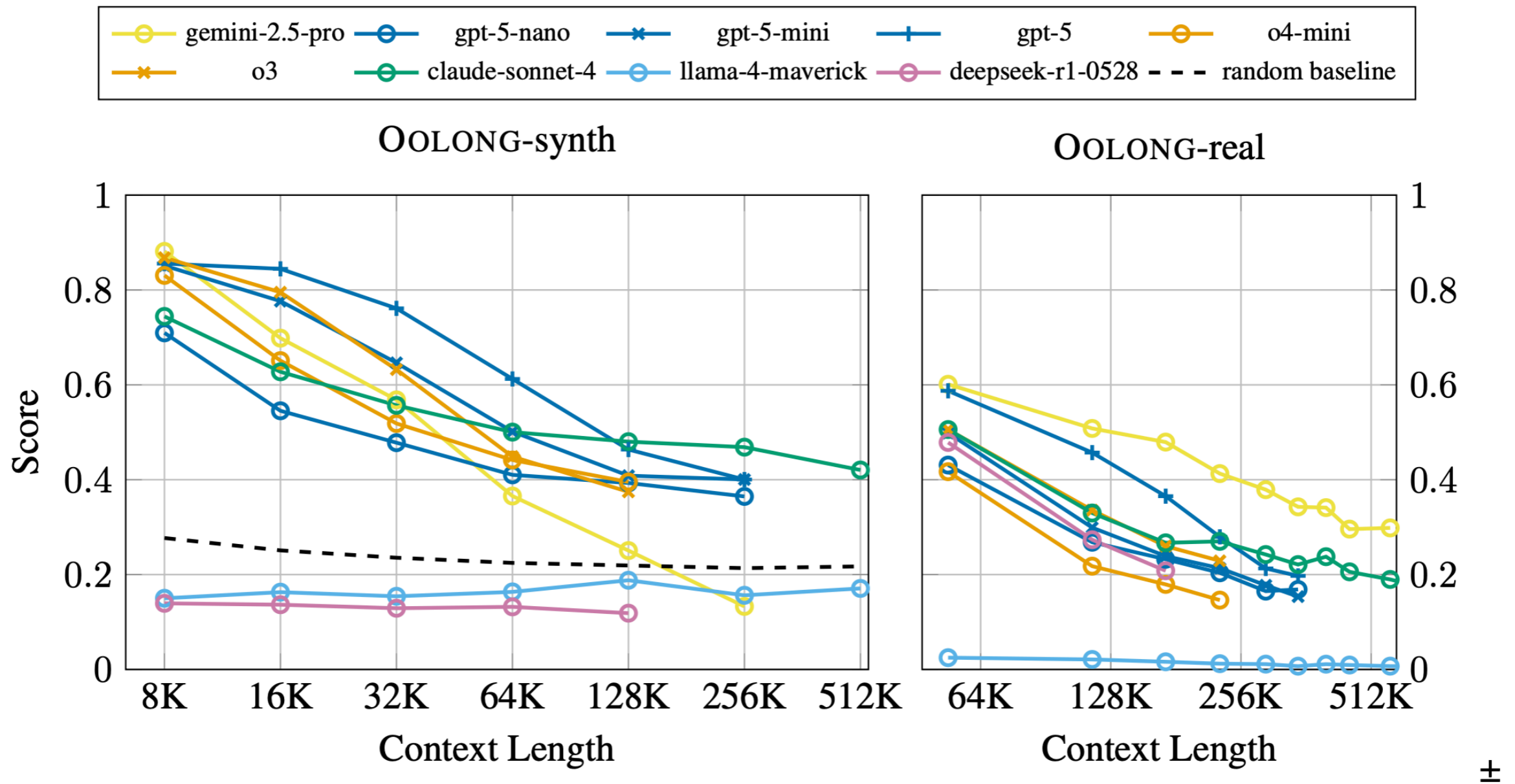
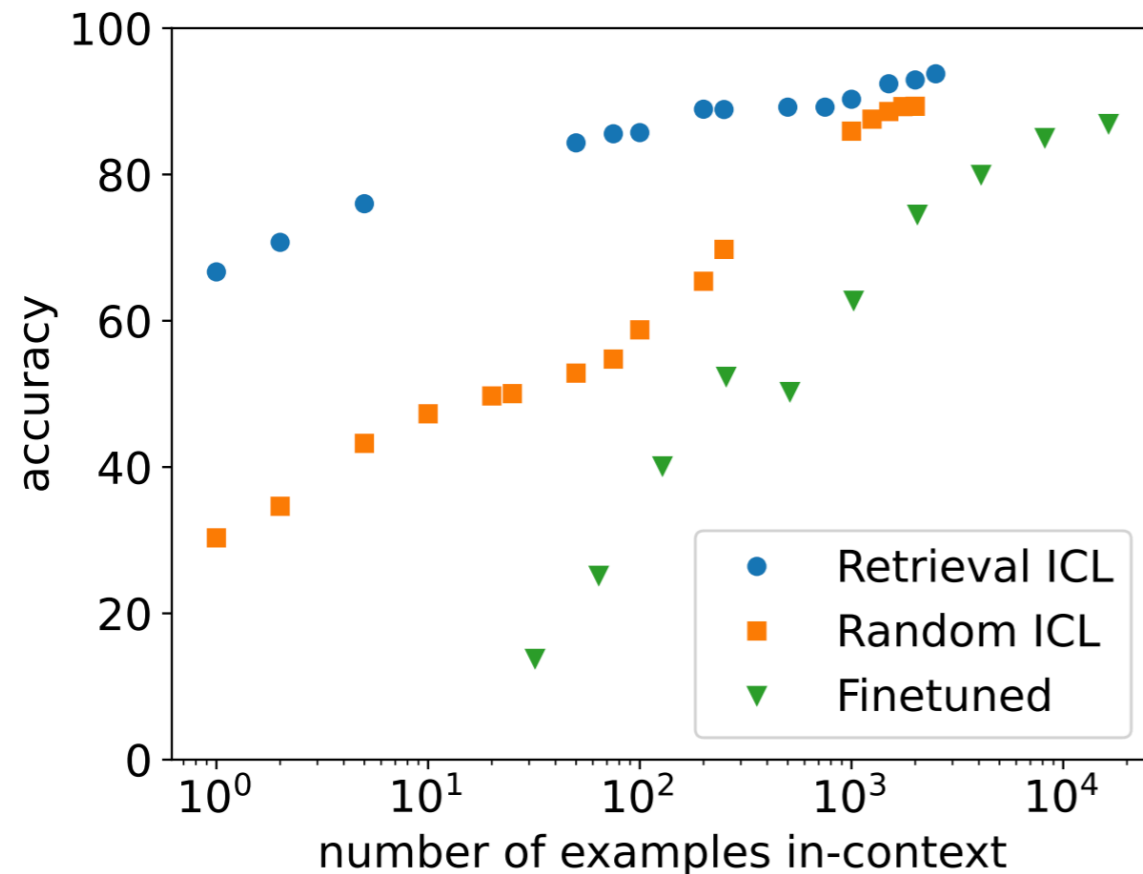


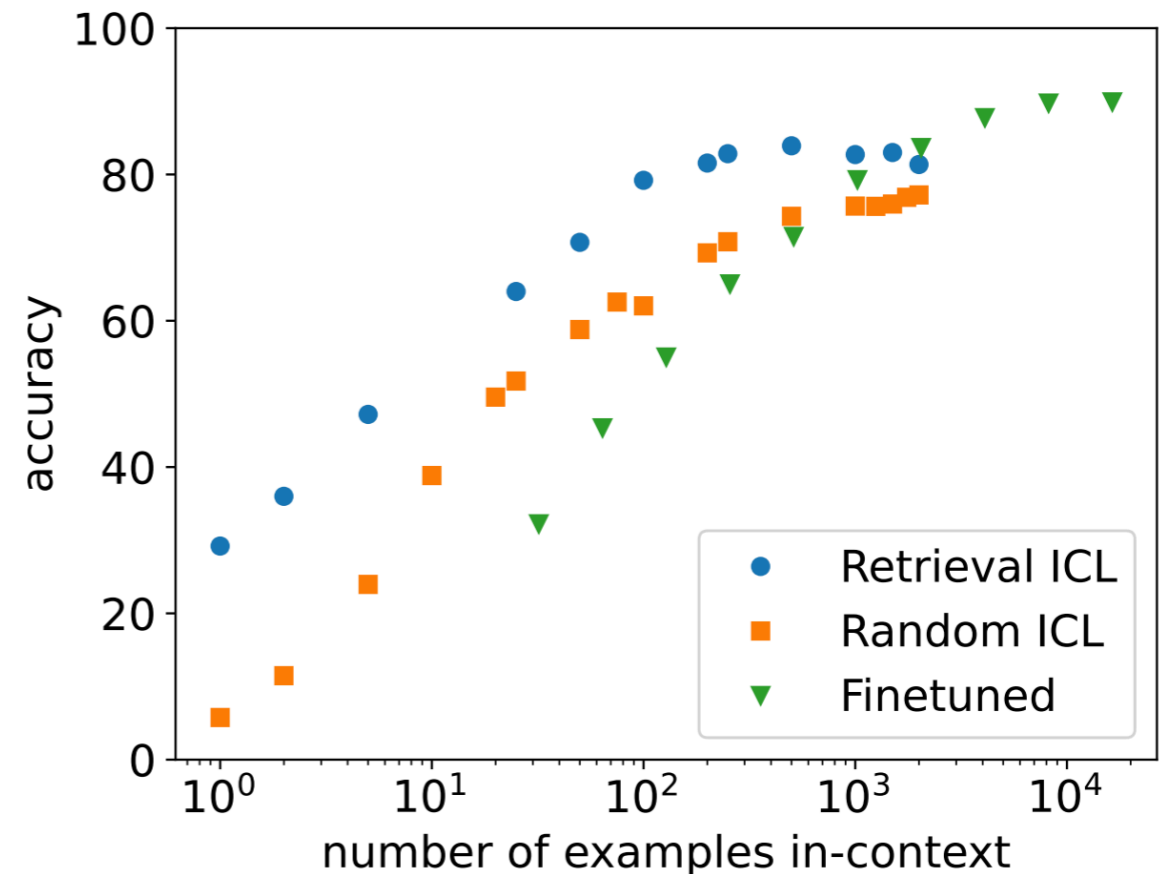
Figure 2: Scores by context window length for OOLONG-synth and OOLONG-real.

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

Today's lecture

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

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

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

Memory: $O(bs^2)$ for all ops

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

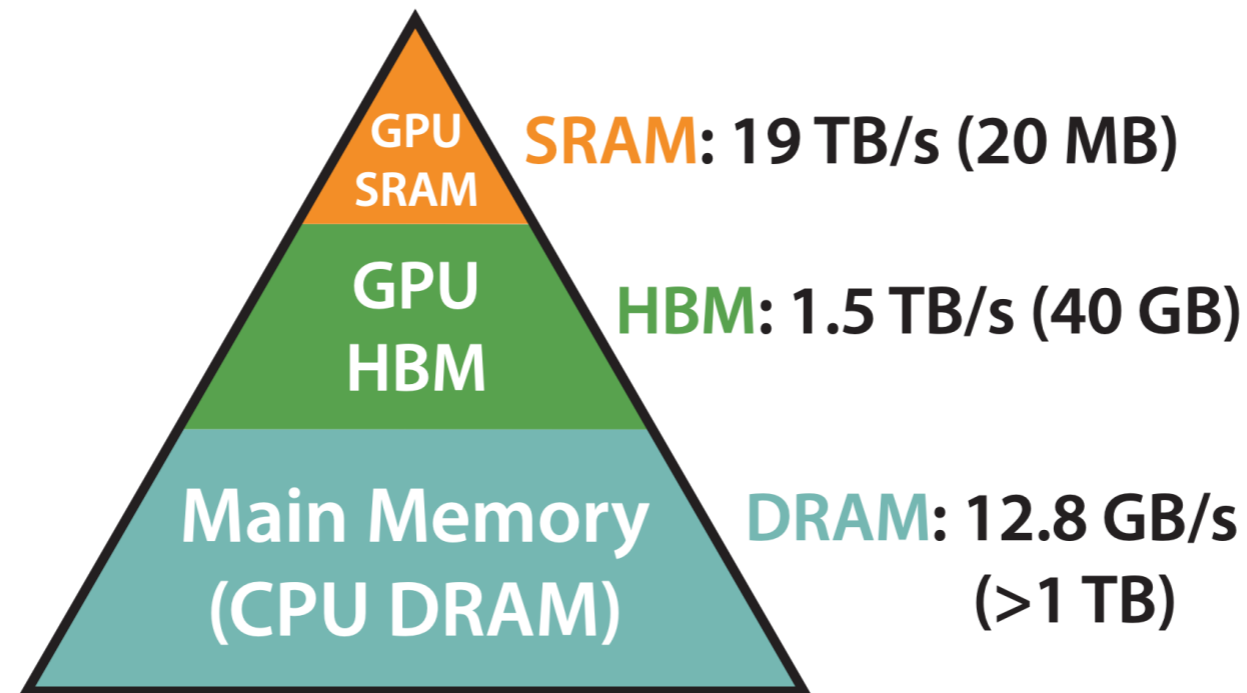
Memory: $O(bs^2h)$ for all ops

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



**Memory Hierarchy with
Bandwidth & Memory Size**

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

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

$$\text{Attention}(Q, K, V) = V^* / S^* \quad \text{Memory: } O(bsd)$$

softmax numerator * V

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

Memory: $O(bsd)$

softmax denominator

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

Memory: $O(bsh)$

Memory-efficient computation

(Jang 2019, Rabe and Staats 2021)

- Online softmax

For each query q :

For $i = 1, \dots, s$:¹

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

$$v^* \leftarrow v^* + v_i \exp(w'_i)$$

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

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

¹In 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	$\pm 5\%$	$\pm 5\%$	$-8 \pm 2\%$	$-13 \pm 2\%$	-	-	-

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

Inference benchmarking from [Rabe and Staats 2021]

Analogous improvements for training

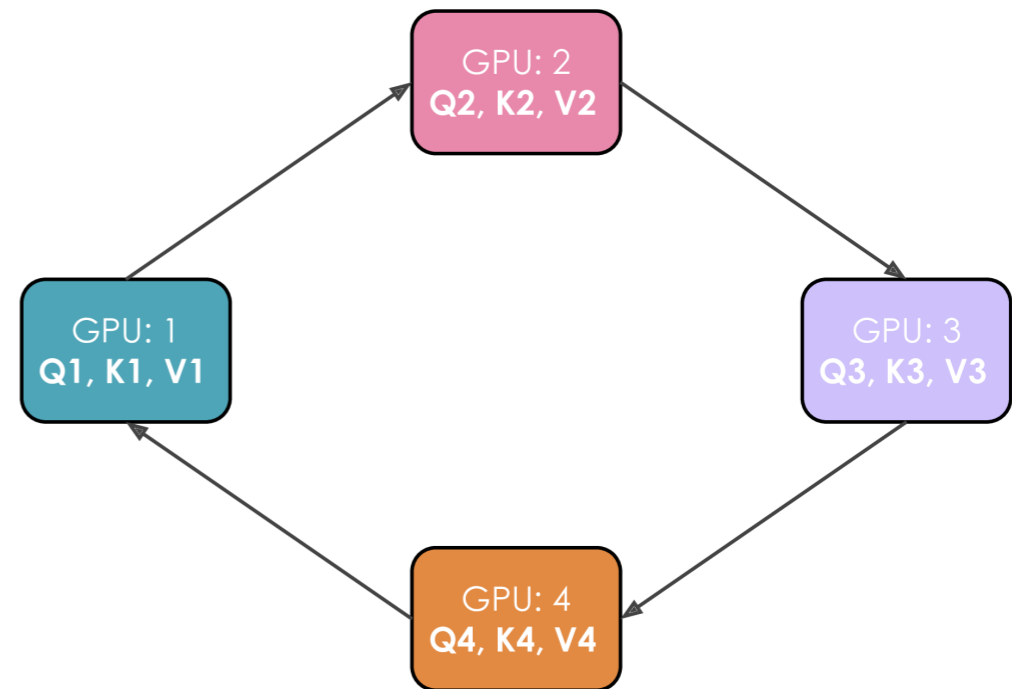
Memory-efficient computation

- 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)

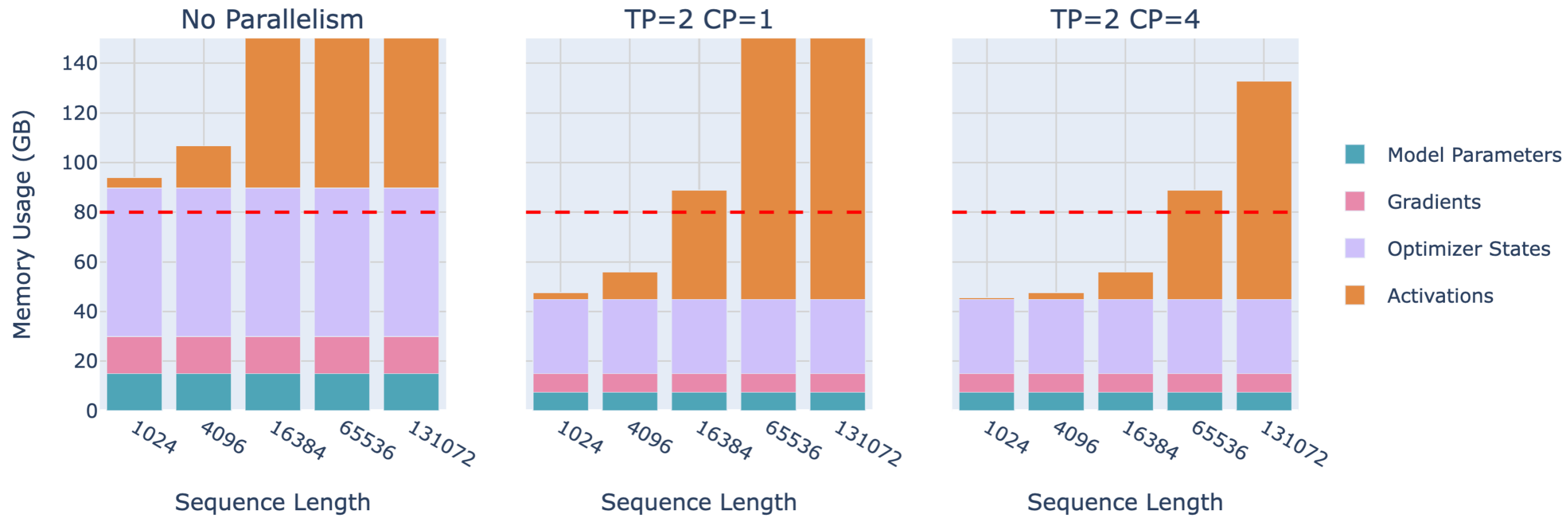
Context parallelism

- 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



Ring attention / context parallelism

Memory Usage for 8B Model



Today's lecture

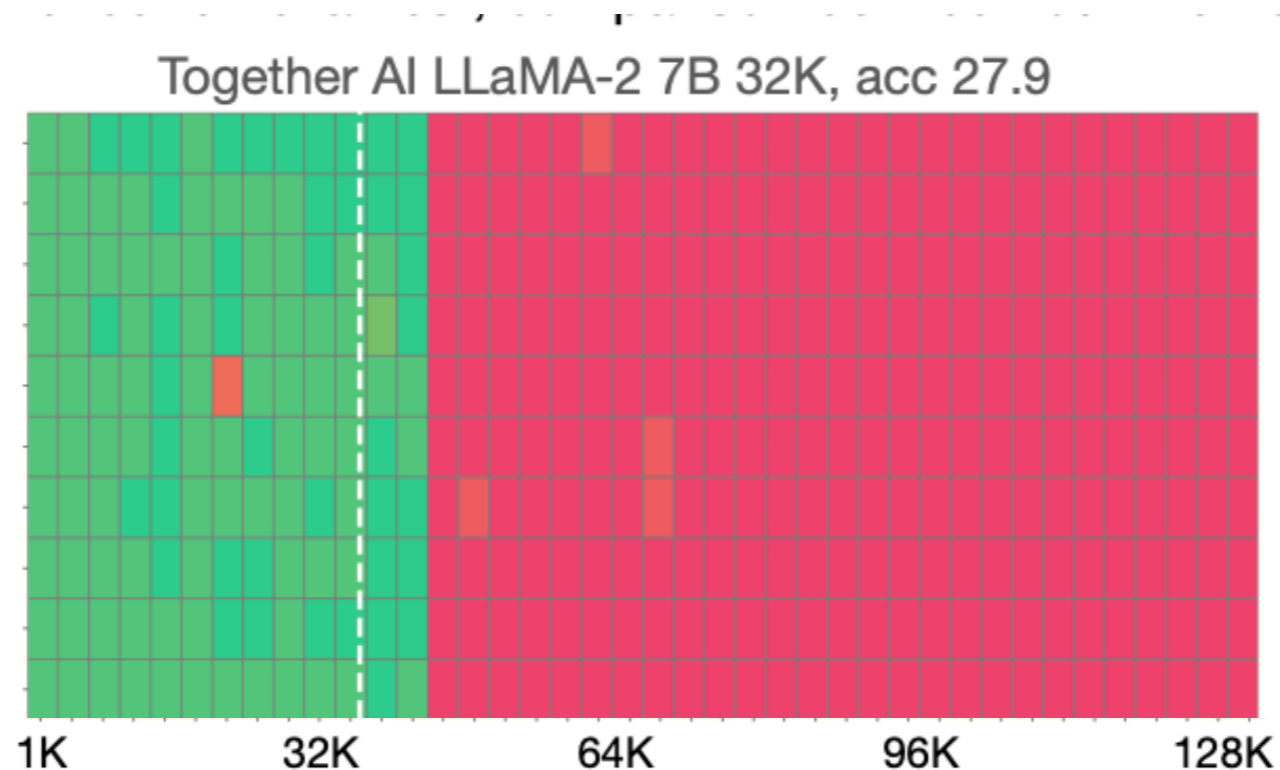
- Long sequence modeling
- Improving transformers
 - Memory-efficient computation
 - **Extrapolation**
 - Transformer modifications
- Transformer alternatives

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)

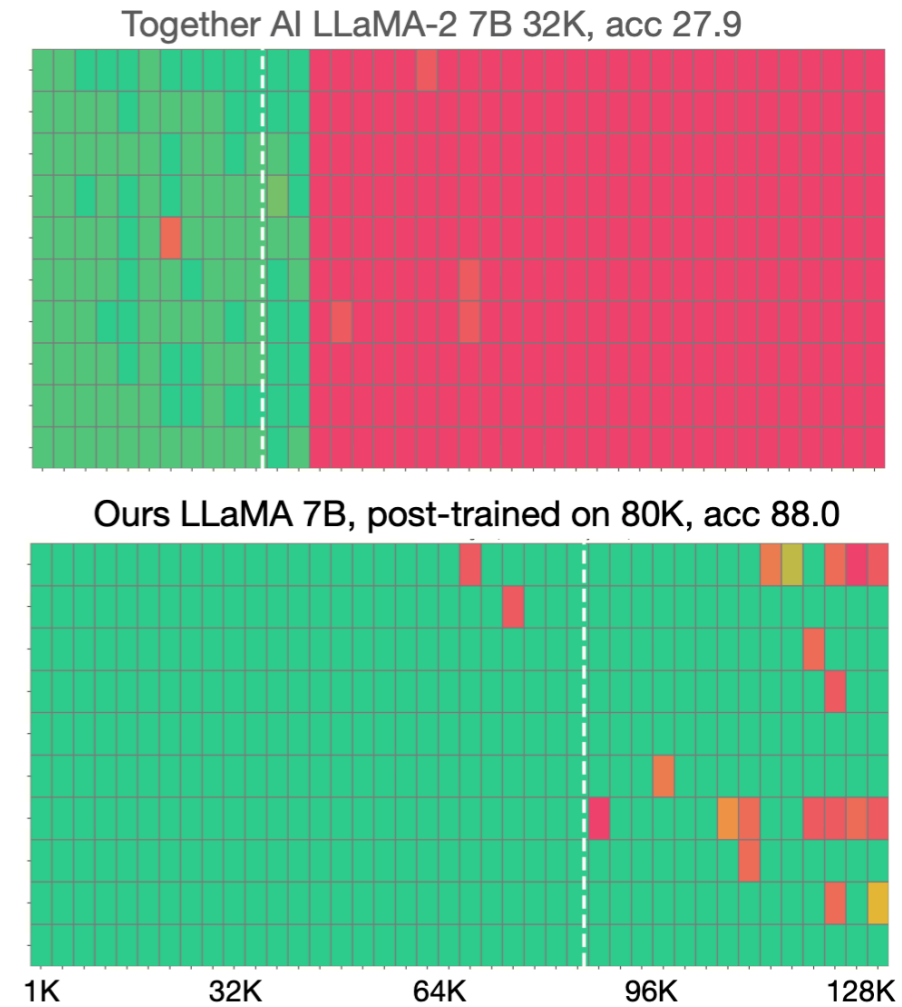
- Llama-2 w/ 32k context (RoPE) can answer questions about sequences up to about 40k, but not beyond



Training with Long Context

(Fu et al. 2024)

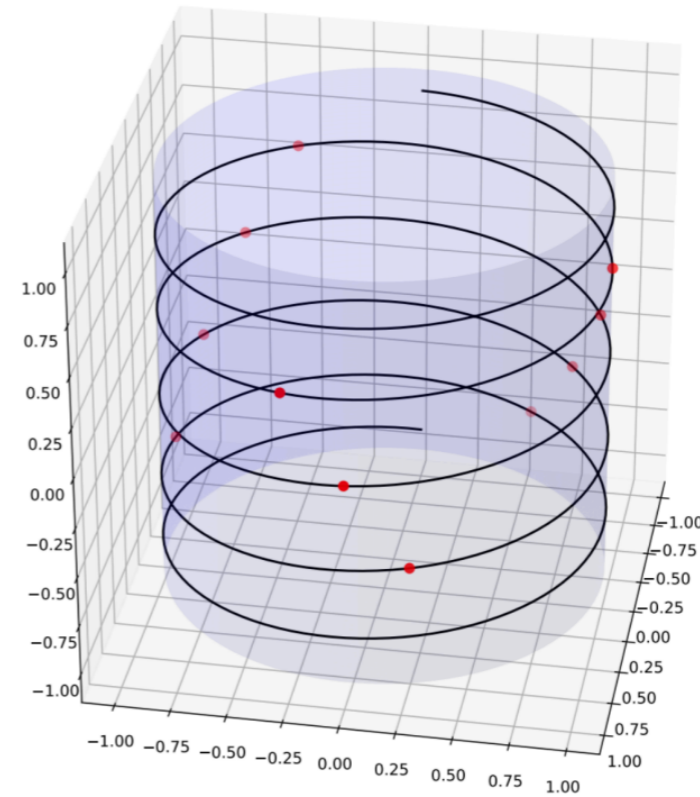
- *A solution:* continually train on longer documents
- Upsample longer documents
- Maintain domain mixture, and upsample long docs in each domain



RoPE Scaling

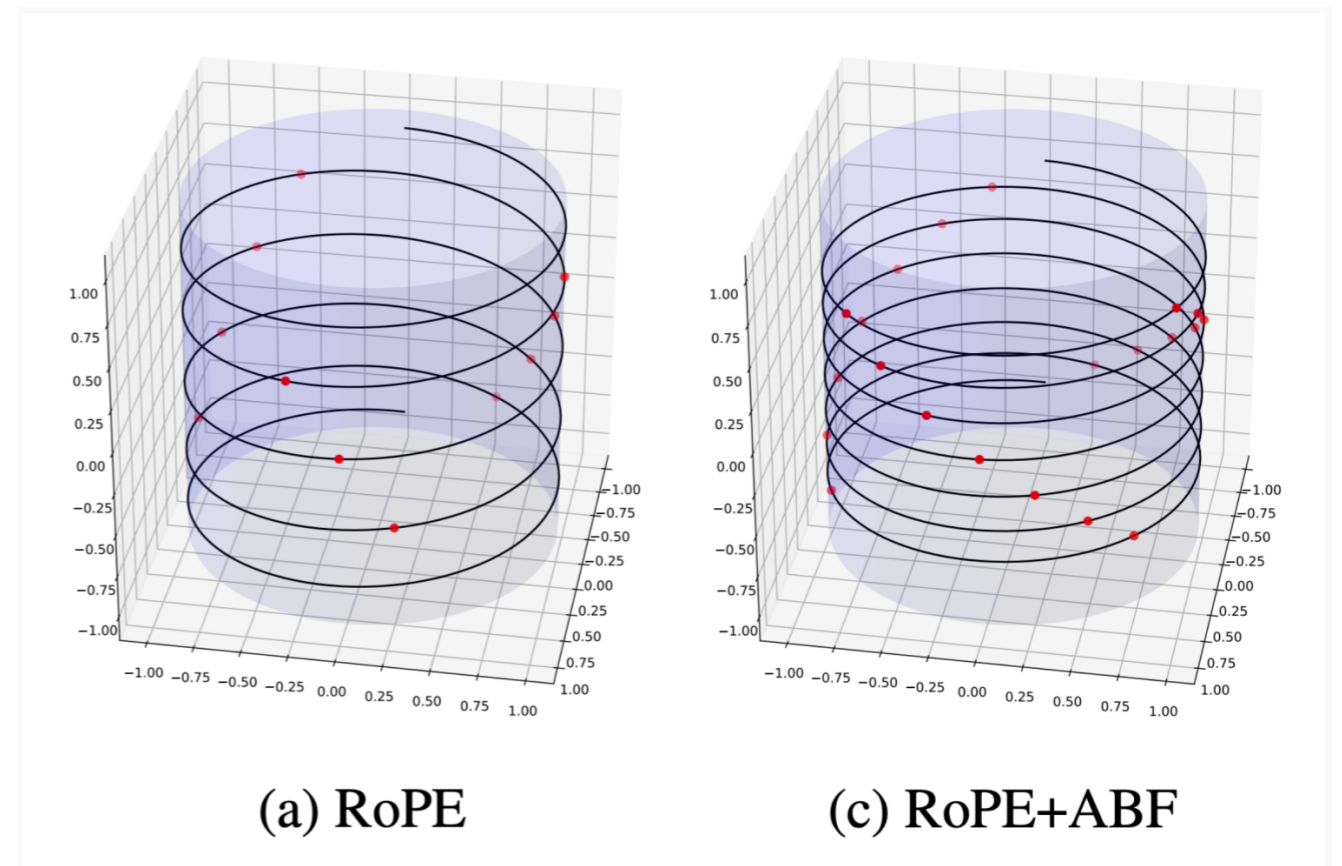
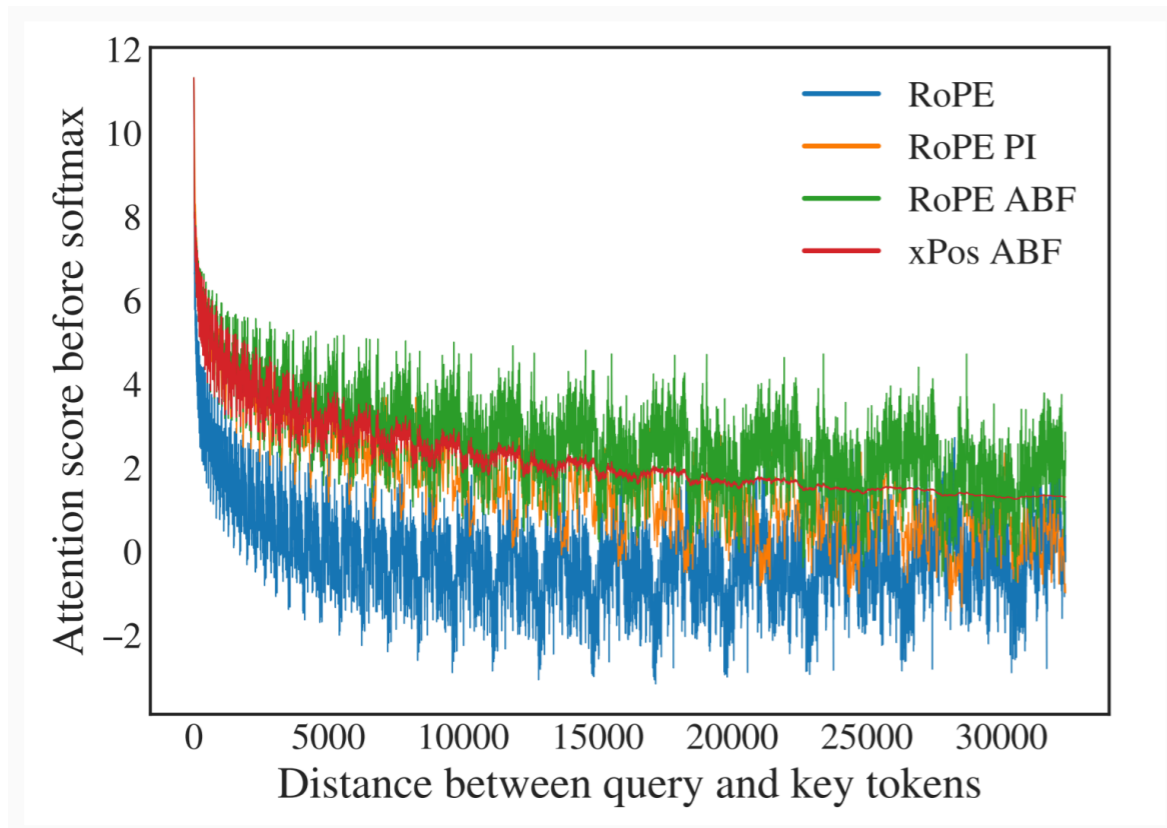
$$\mathbf{R}(\theta, i) = \begin{pmatrix} \cos i\theta_1 & -\sin i\theta_1 & \cdots & 0 & 0 \\ \sin i\theta_1 & \cos i\theta_1 & \cdots & 0 & 0 \\ \vdots & & & & \\ 0 & 0 & \cdots & \cos i\theta \frac{d_k}{2} & -\sin i\theta \frac{d_k}{2} \\ 0 & 0 & \cdots & \sin i\theta \frac{d_k}{2} & \cos i\theta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

- 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



Today's lecture

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

Transformer modifications

Skipped in lecture due to time

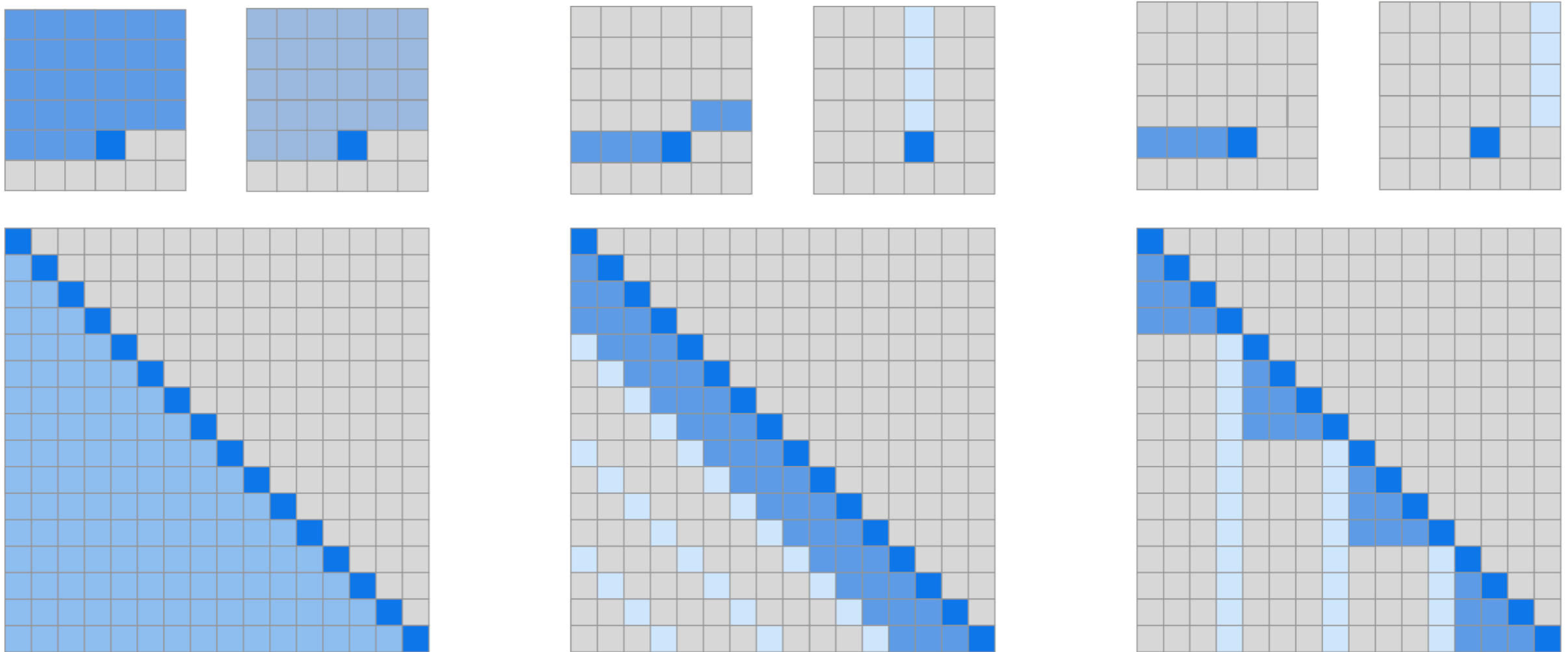
- Sparse Attention
- Sliding Window Attention
- Compression
- Low-rank Approximation

Skipped in lecture due to time

Sparse Transformers

(Child et al. 2019)

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



(a) Transformer

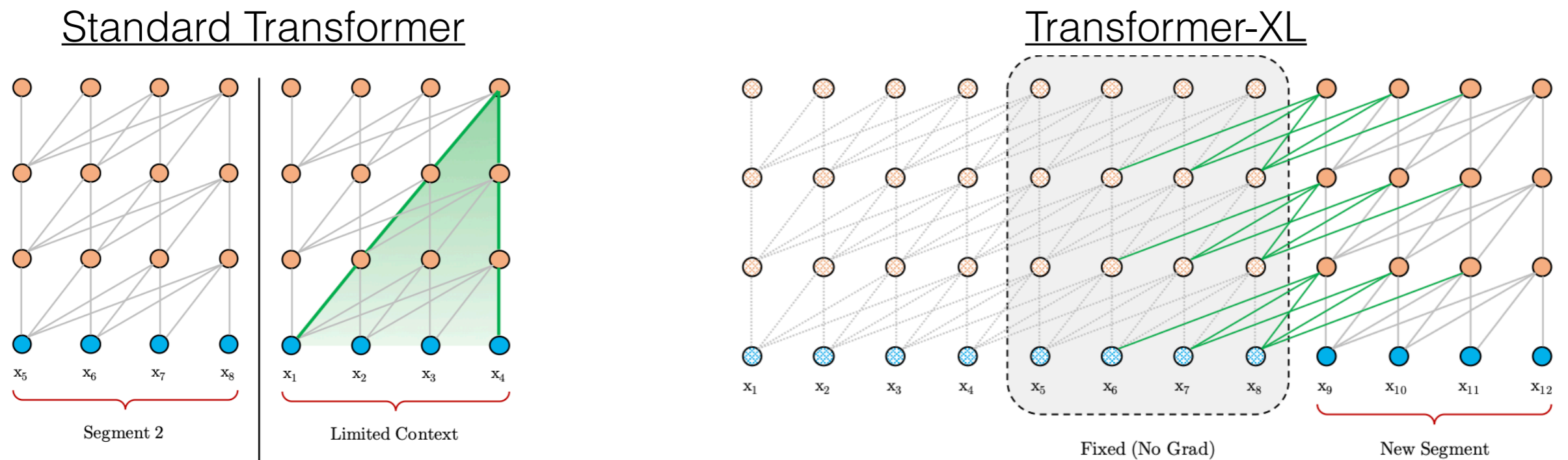
(b) Sparse Transformer (strided)

(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

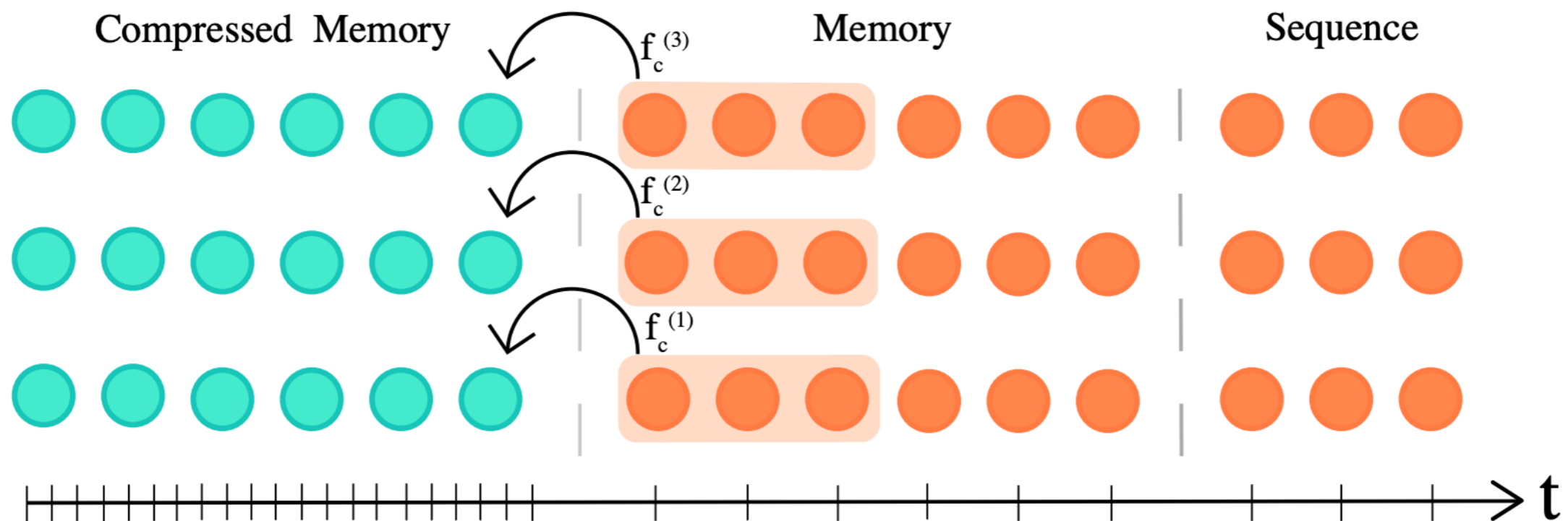


- Like truncated backprop through time for RNNs; can use previous states, but not backprop into them
- See also Mistral's (Jiang et al. 2023) sliding window attention

Skipped in lecture due to time

Compressing Previous States

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

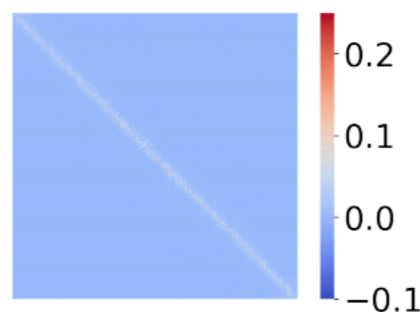


Skipped in lecture due to time

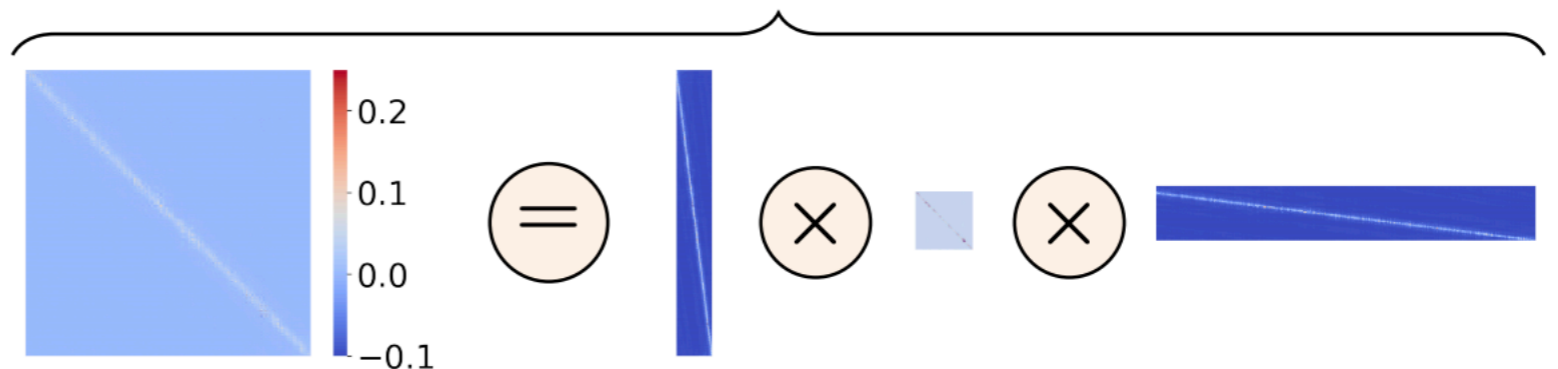
Low-rank Approximation

- 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)

softmax



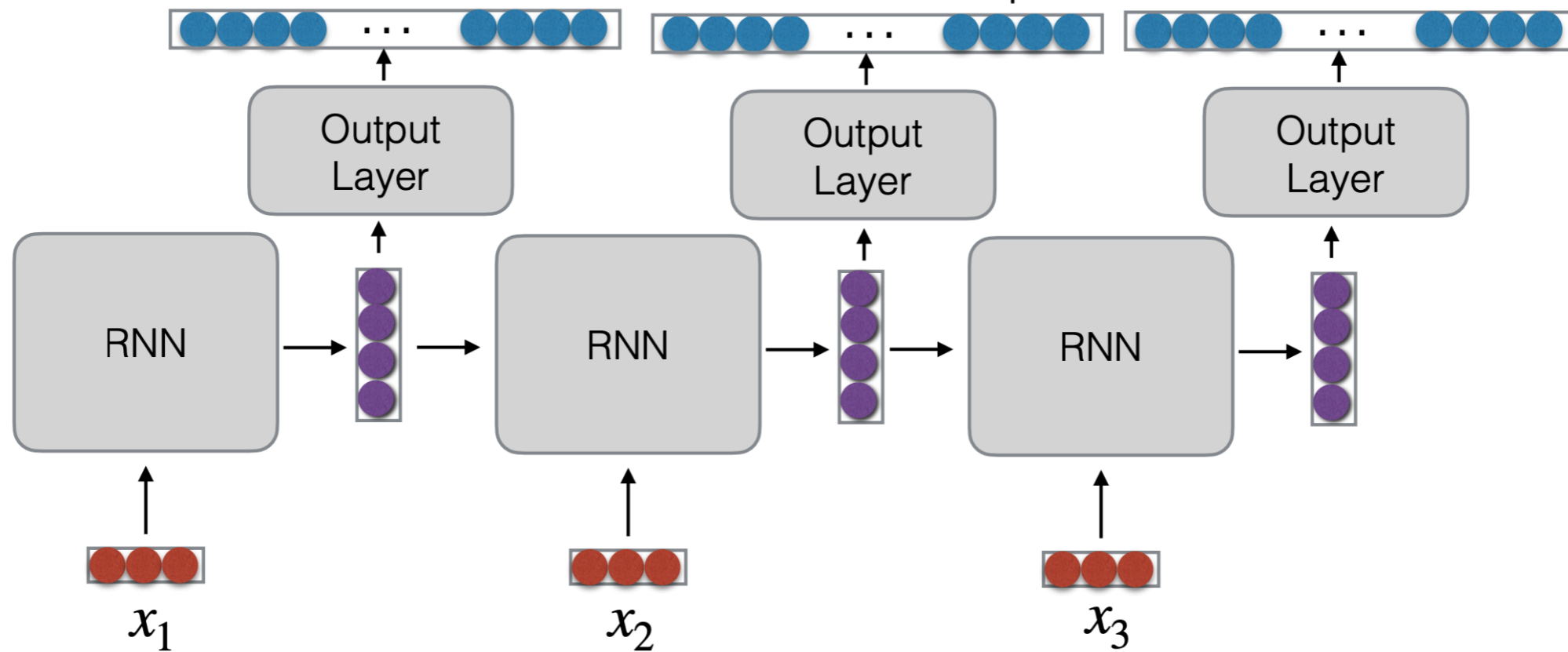
Nyström approximation



Today's lecture

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

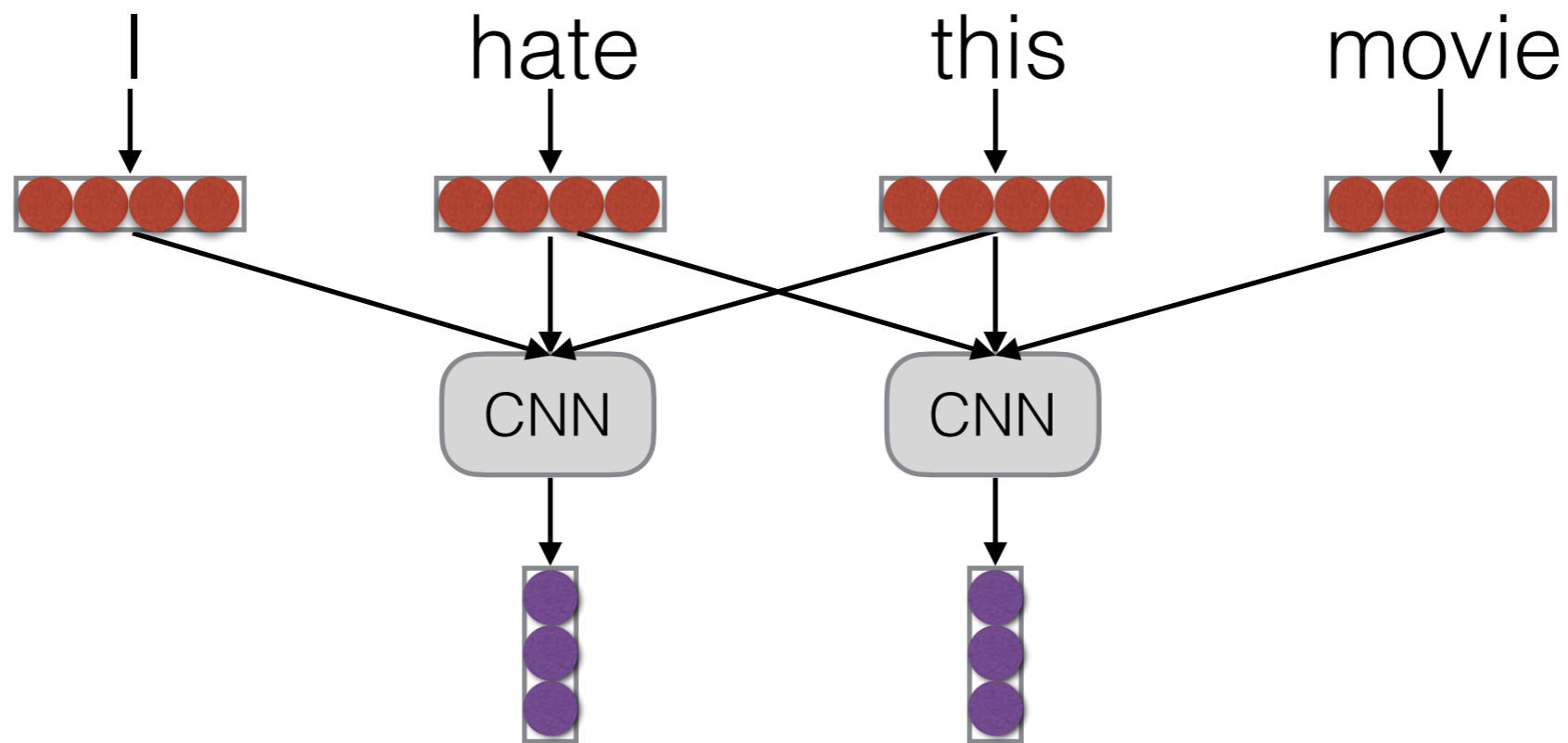
Reminder: RNNs



- Infinite context
- Memory-efficient inference (single hidden state)
- Hard to parallelize training

Convolution

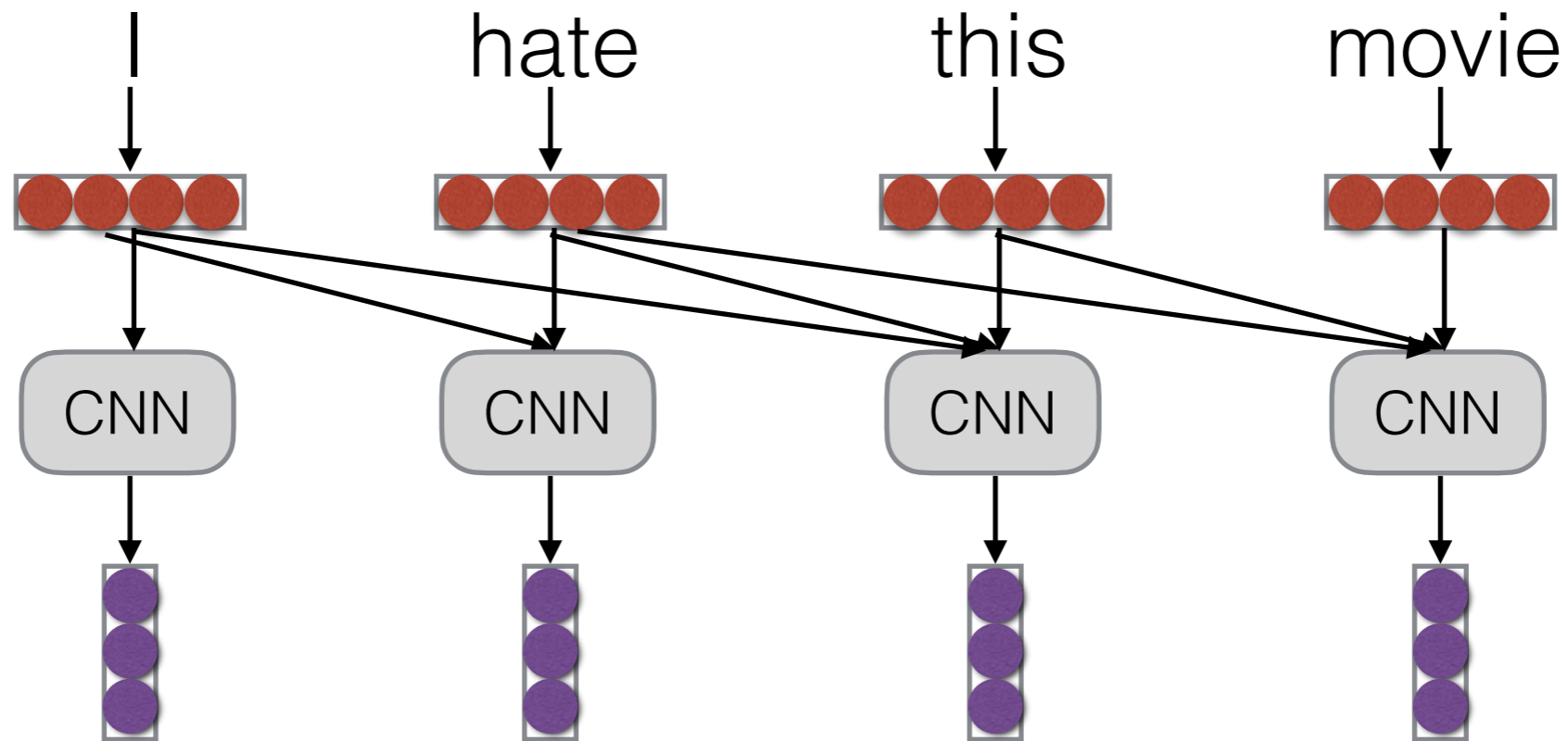
- Calculate based on local context



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

Convolution for Auto-regressive Models

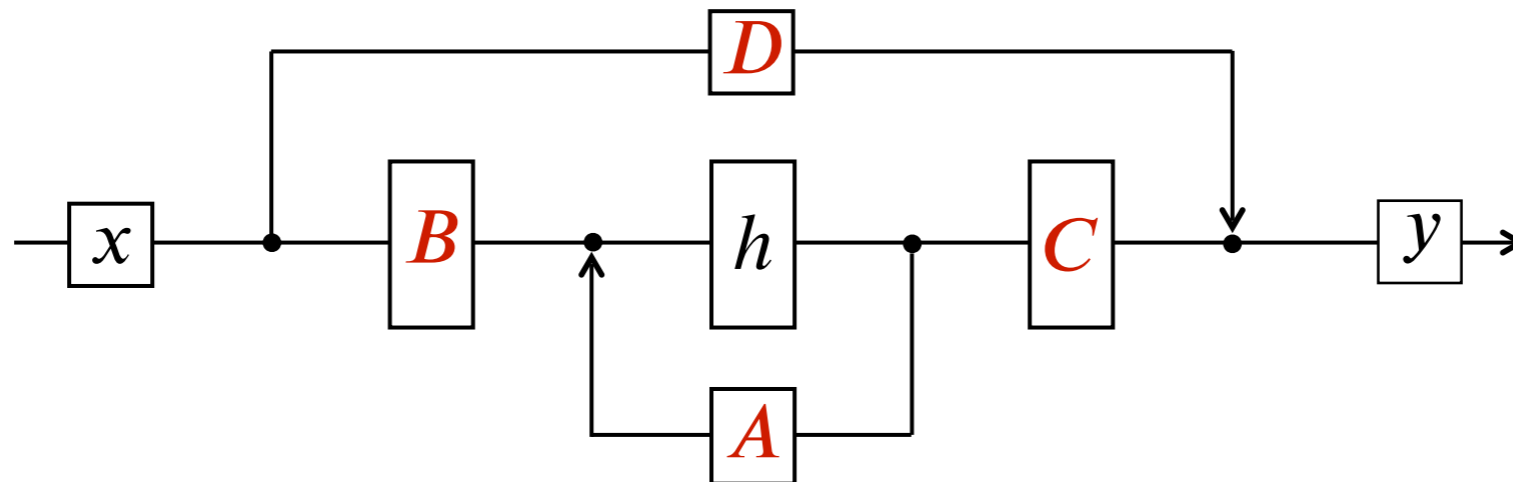
- Functionally identical, just consider previous context



Structured State Space Models

(Gu et al. 2021)

- Models that take a form like the following

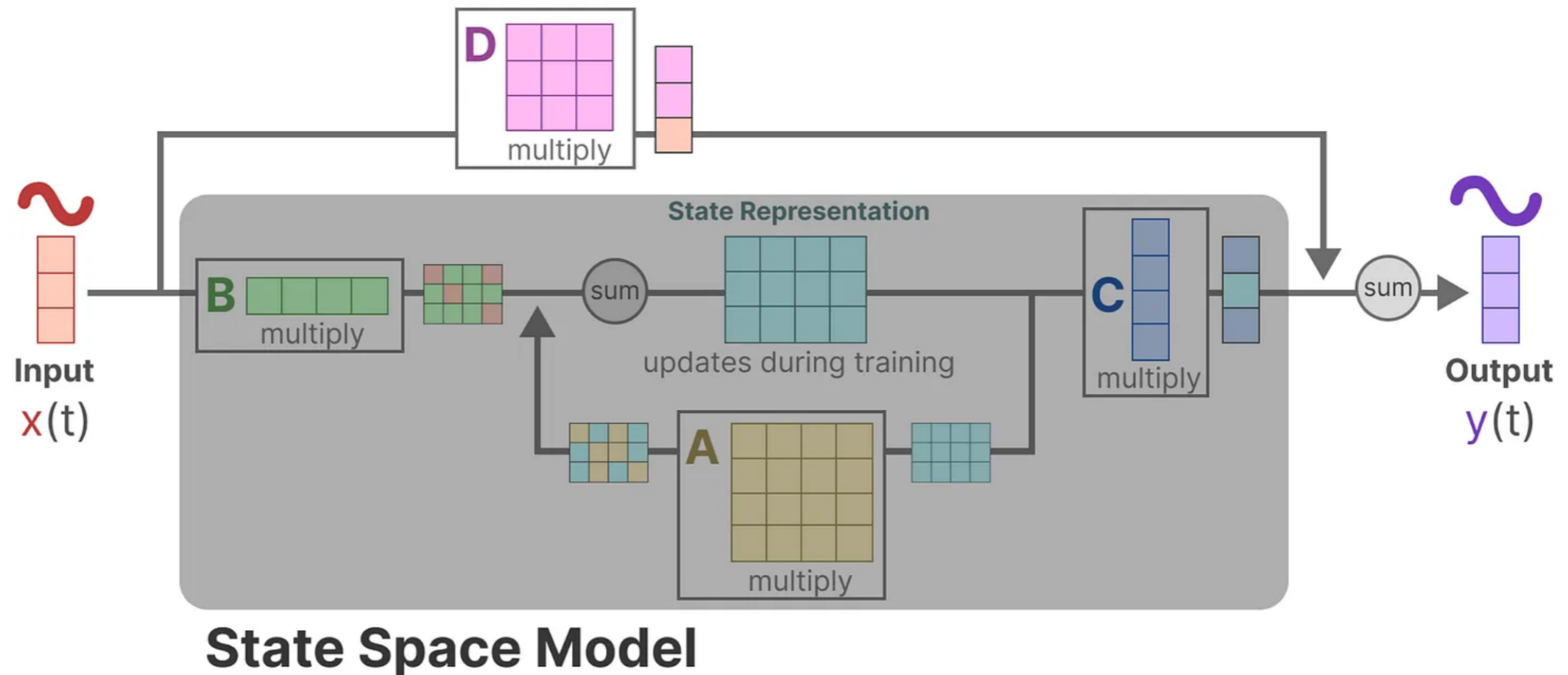


$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$$

$$y(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$$

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

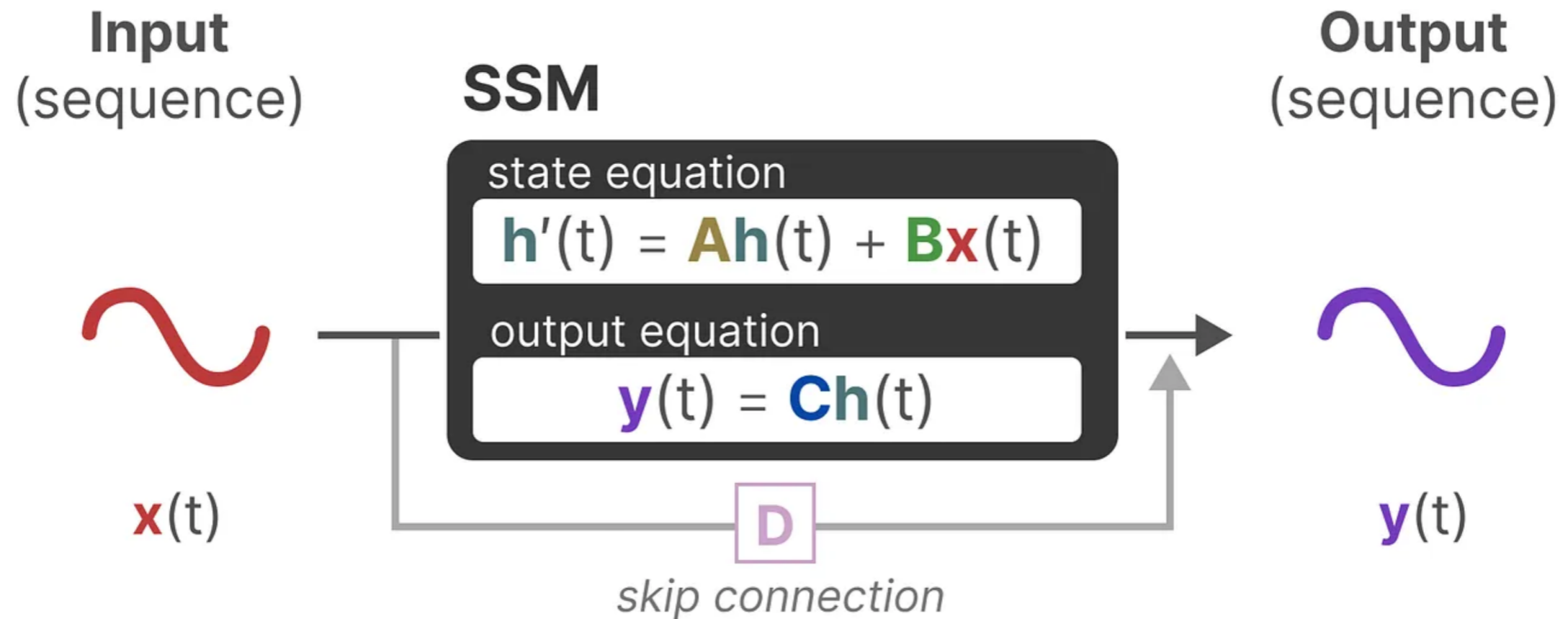
Structured State Space Models



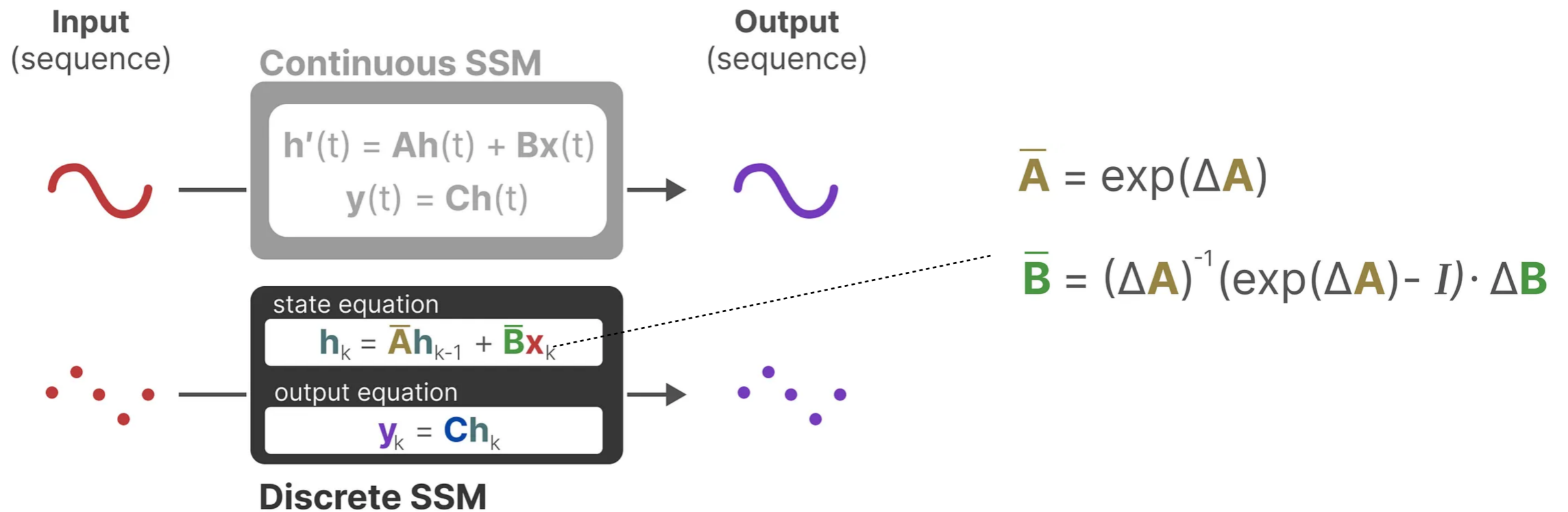
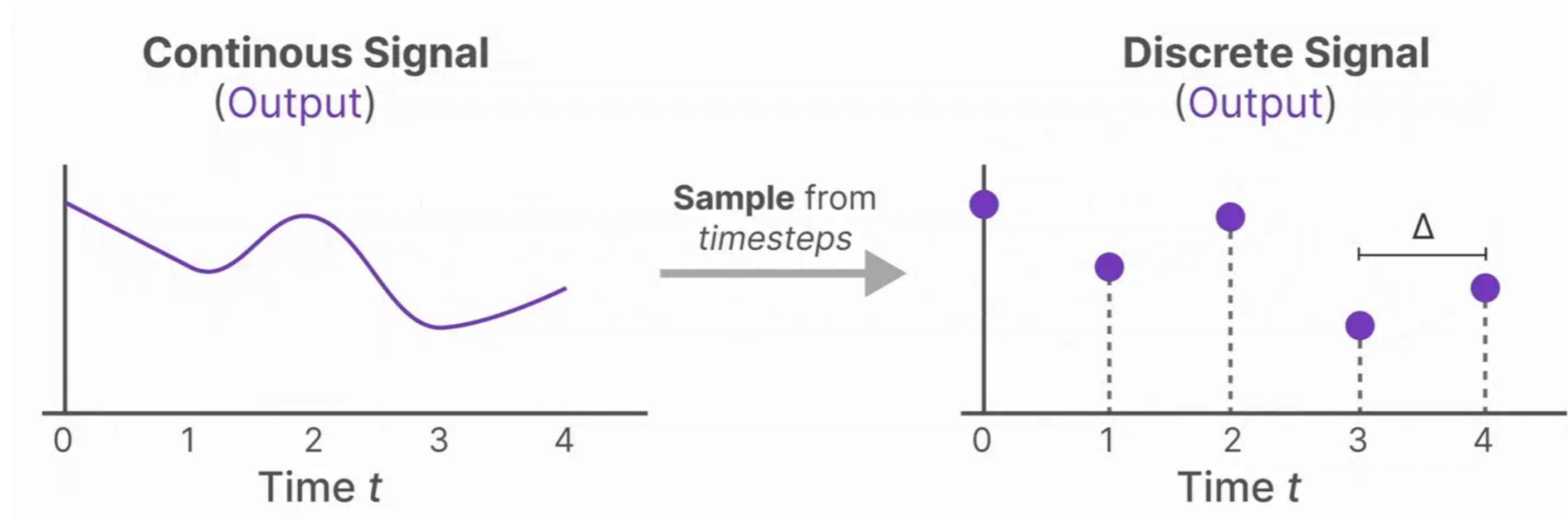
$$h'(t) = Ah(t) + Bx(t)$$

$$y(t) = Ch(t) + Dx(t)$$

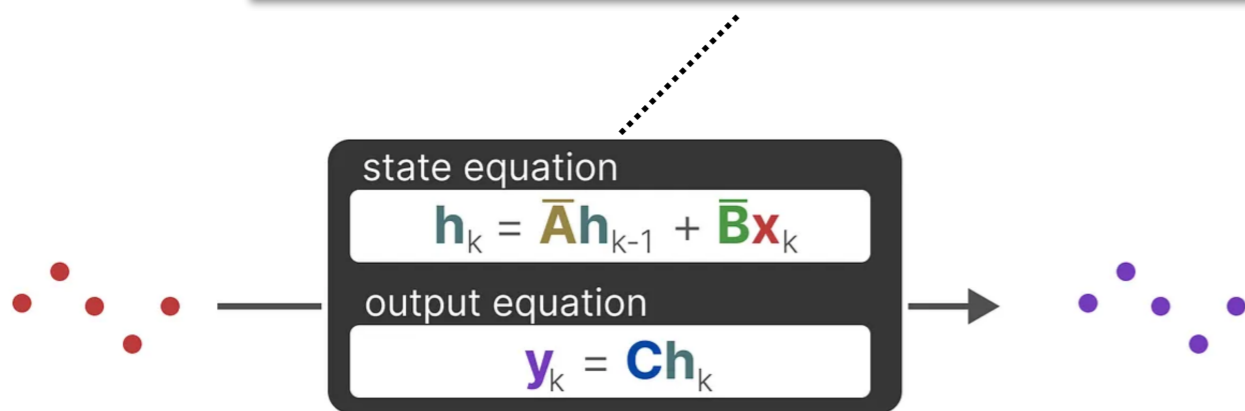
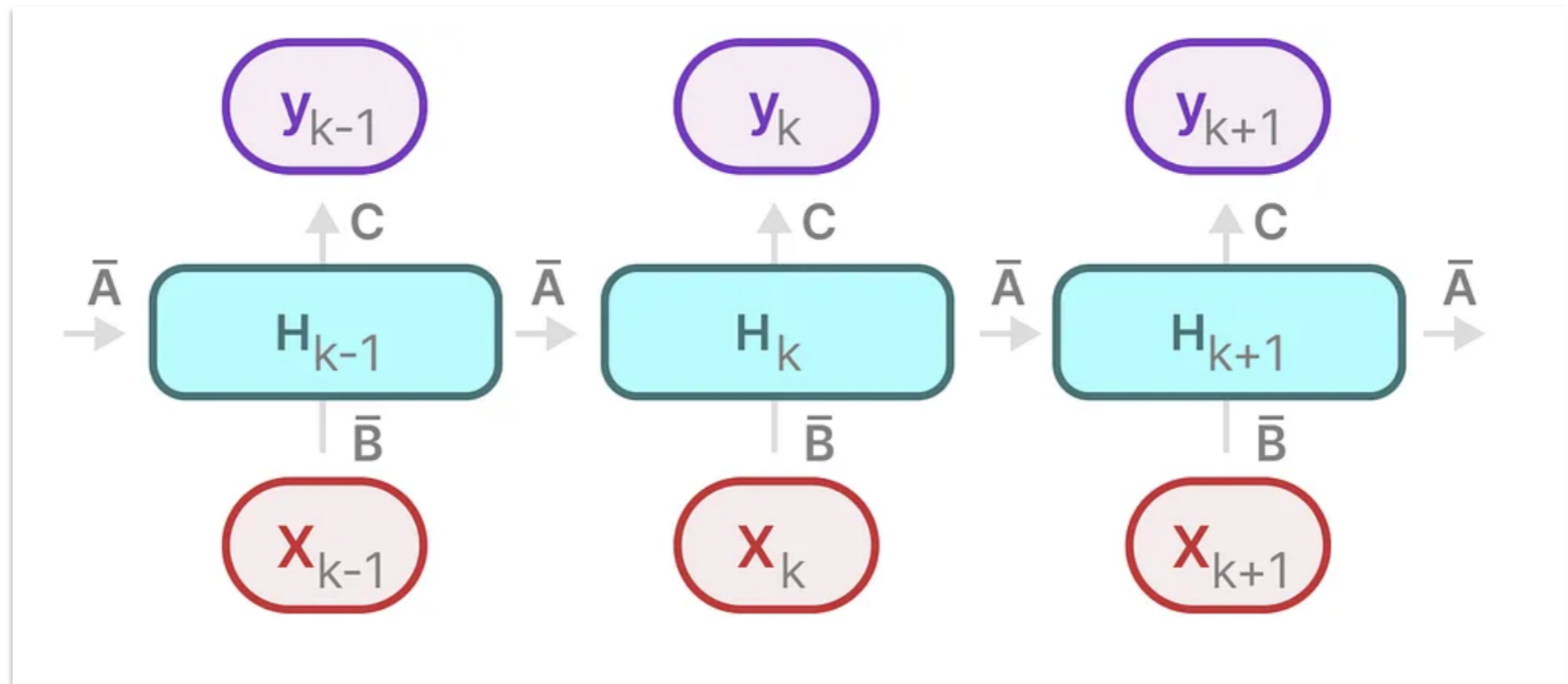
Structured State Space Models



SSMs: Discretization



SSMs: Recurrent View



- ✓ Efficient inference
- ✗ Parallelizable training

SSM: Convolution View

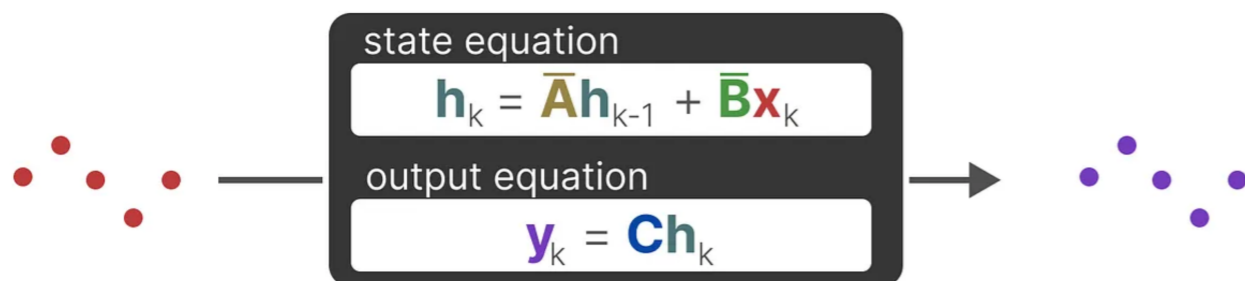
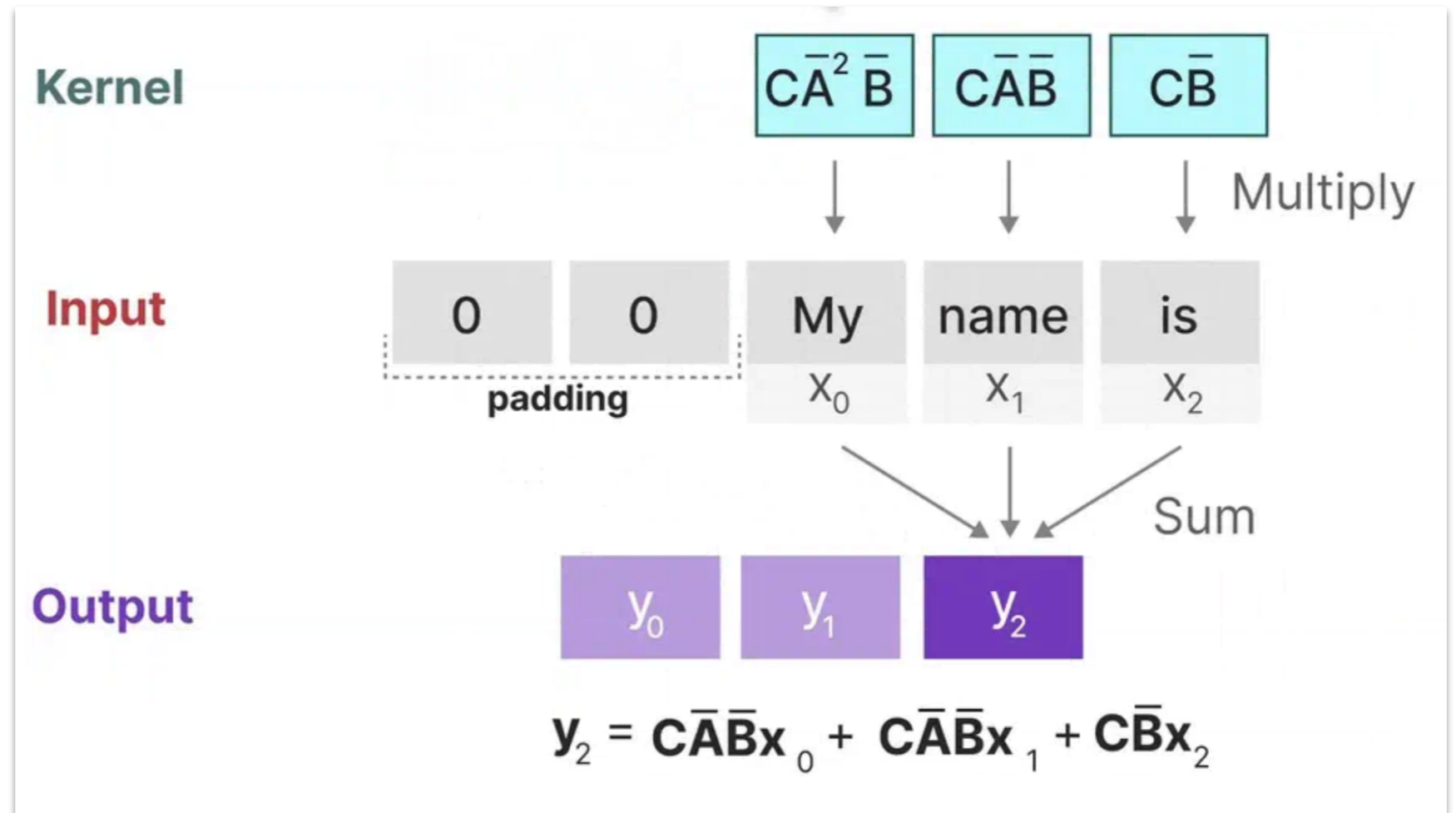
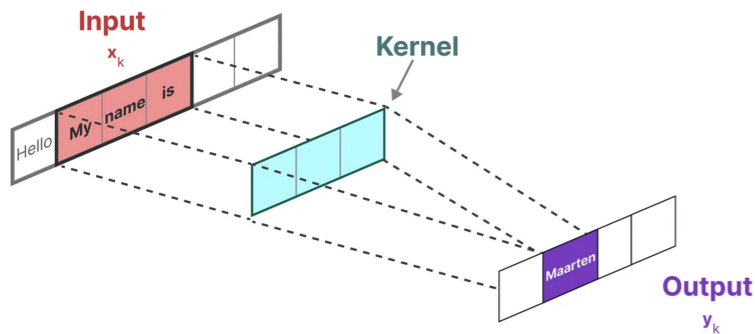


Figure: A Visual Guide to Mamba

SSM: Convolution View

$$\text{kernel} \rightarrow \bar{\mathbf{K}} = (\mathbf{CB}, \mathbf{CAB}, \dots, \mathbf{CA}^k \mathbf{B}, \dots)$$

$$\mathbf{y} = \mathbf{x} * \bar{\mathbf{K}}$$

output input kernel

- ✓ Parallelizable training
- ✗ Unbounded context

SSM Variants

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

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

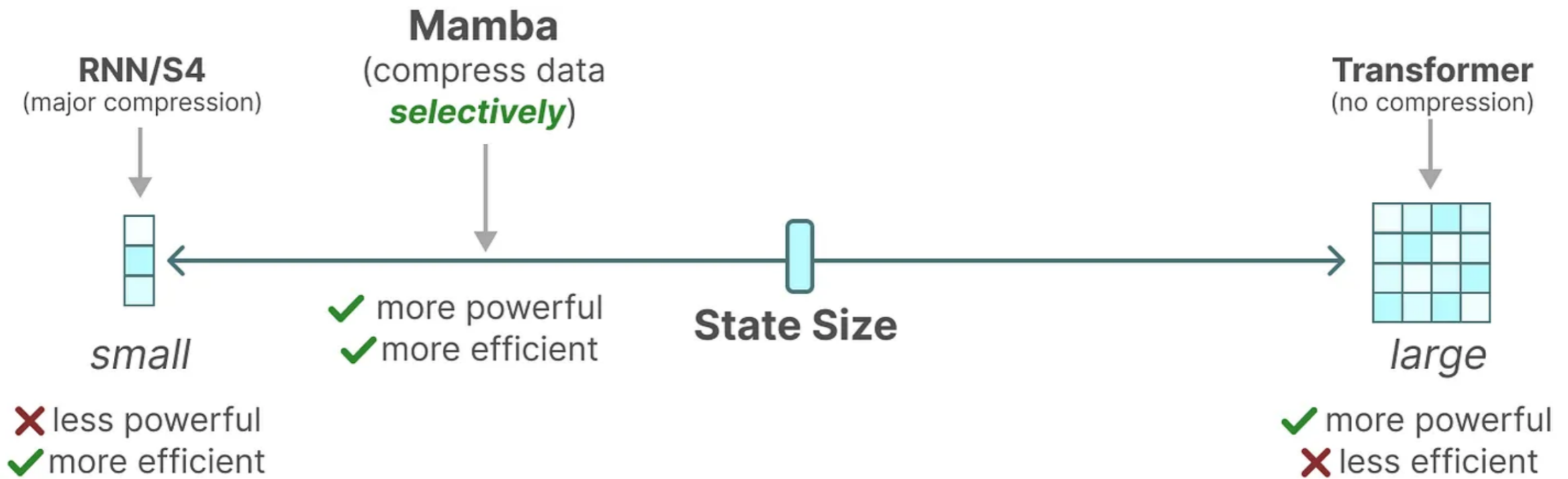
Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu^{*1} and Tri Dao^{*2}

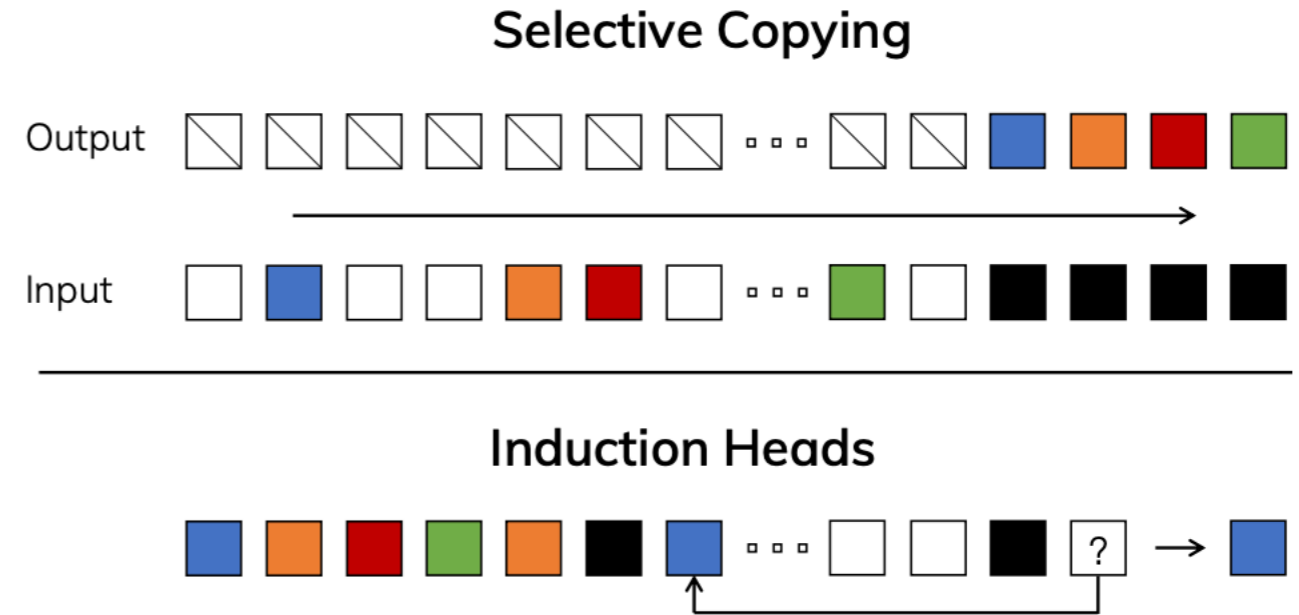
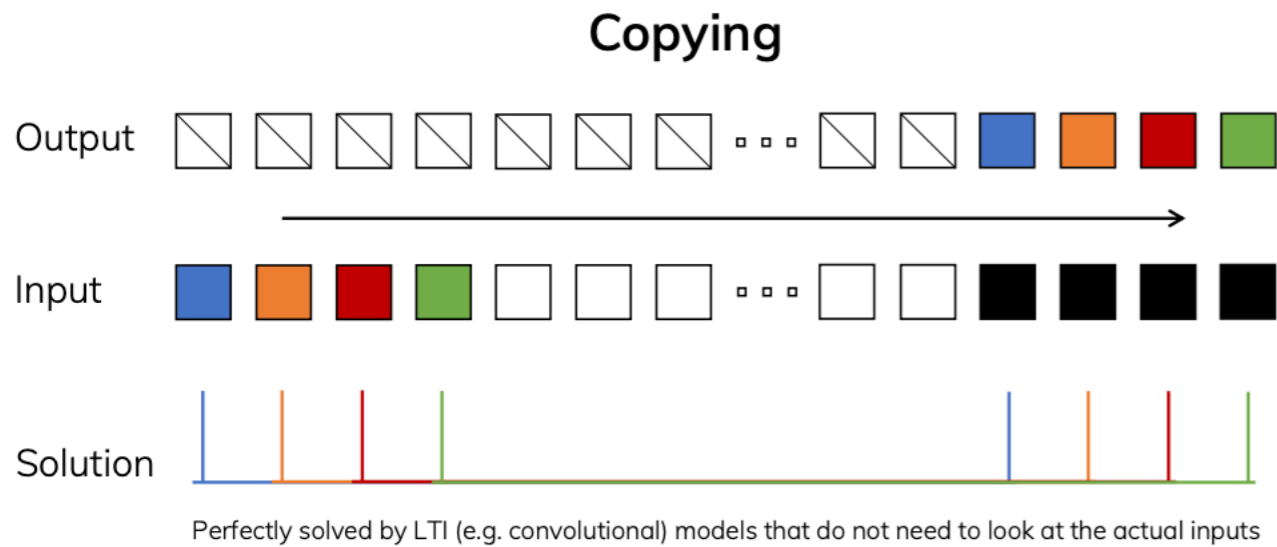
¹Machine Learning Department, Carnegie Mellon University

²Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

Mamba: Selective SSMs



Mamba: Selective SSMs



Mamba: Selective SSMs

- **S6**: S4 with time-varying parameters (B, C, Δ)

Algorithm 2 SSM + Selection (S6)

Input: $x : (B, L, D)$

Output: $y : (B, L, D)$

1: $A : (D, N) \leftarrow \text{Parameter}$

▸ Represents structured $N \times N$ matrix

2: $B : (B, L, N) \leftarrow s_B(x)$

3: $C : (B, L, N) \leftarrow s_C(x)$

4: $\Delta : (B, L, D) \leftarrow \tau_\Delta(\text{Parameter} + s_\Delta(x))$

5: $\bar{A}, \bar{B} : (B, L, D, N) \leftarrow \text{discretize}(\Delta, A, B)$

6: $y \leftarrow \text{SSM}(\bar{A}, \bar{B}, C)(x)$

▸ **Time-varying**: recurrence (*scan*) only

7: **return** y

Mamba: Selective SSMs

- Parallel scan algorithm due to sequential computation

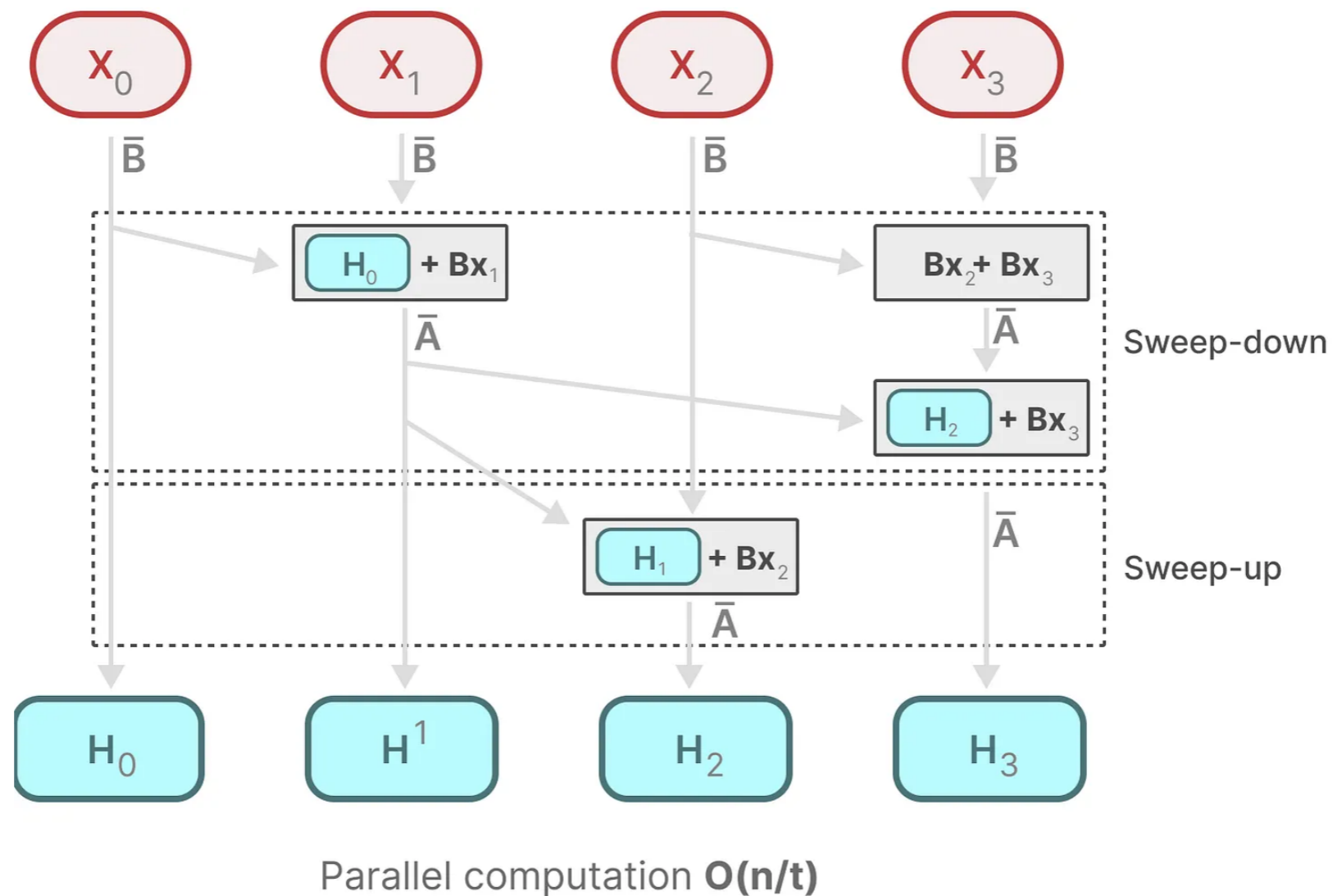
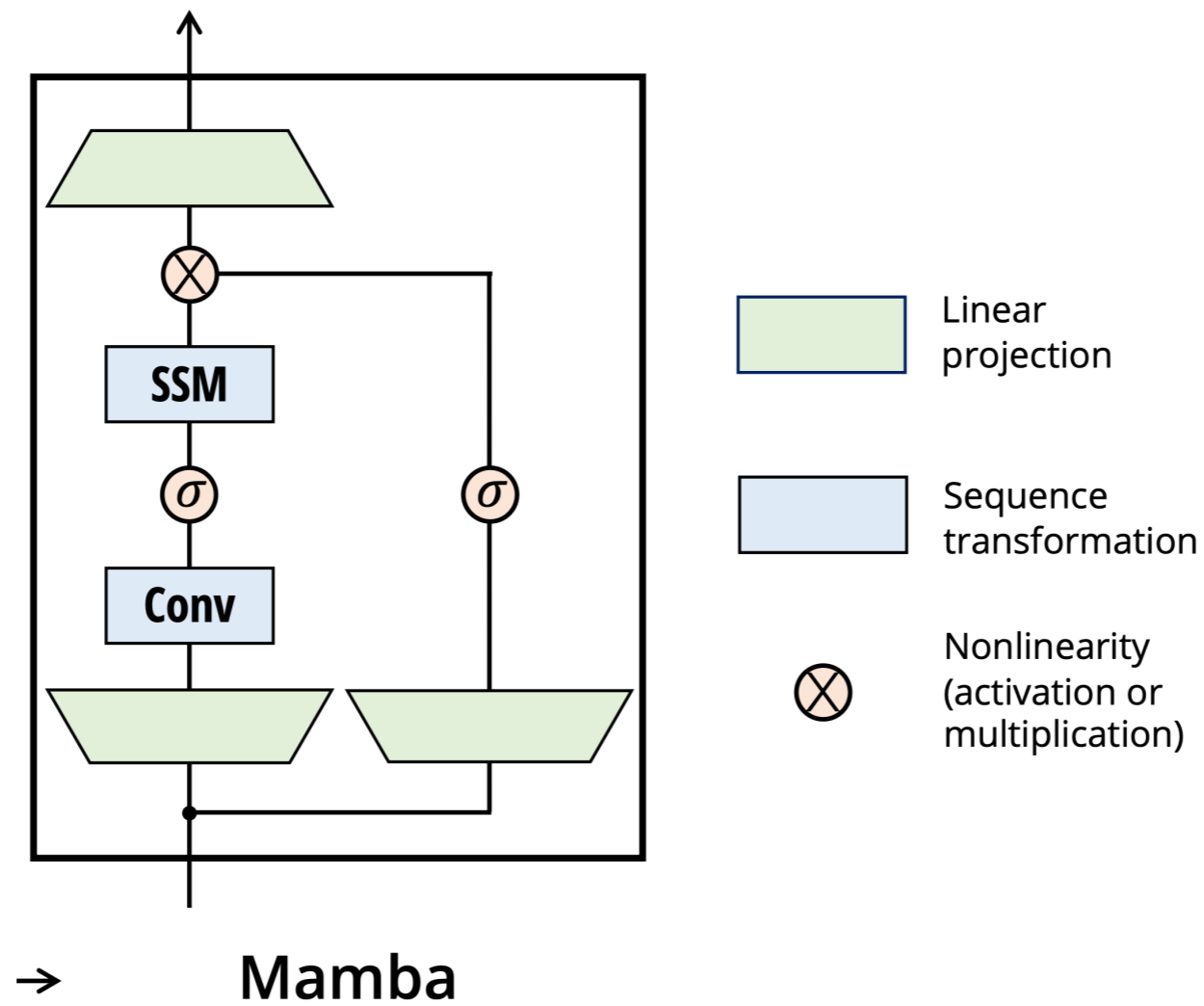


Figure: [A Visual Guide to Mamba](#)

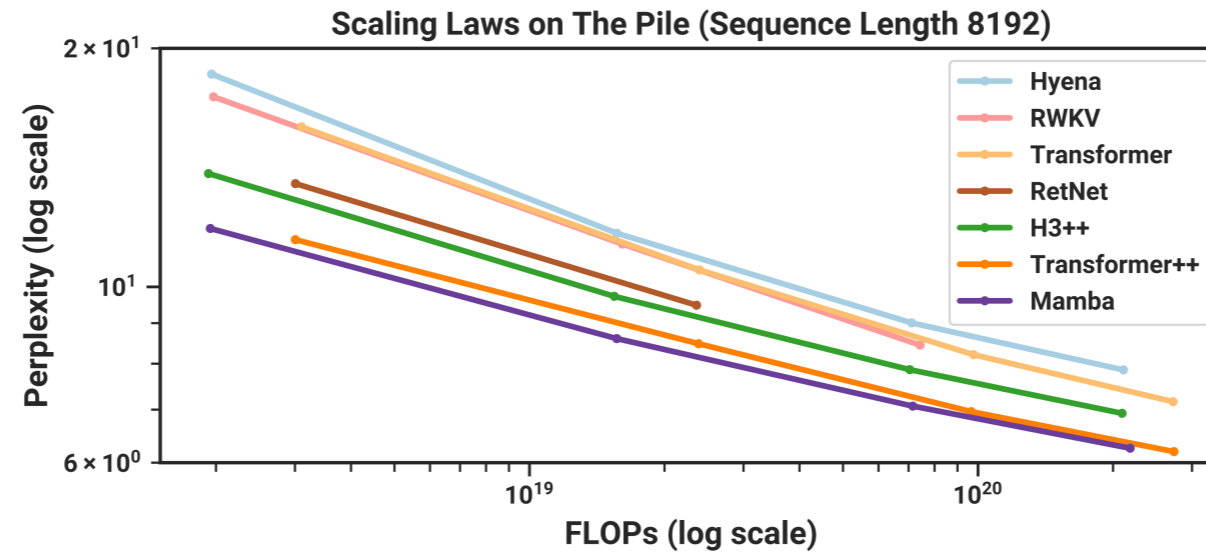
Mamba

- Block/layer that incorporates S6

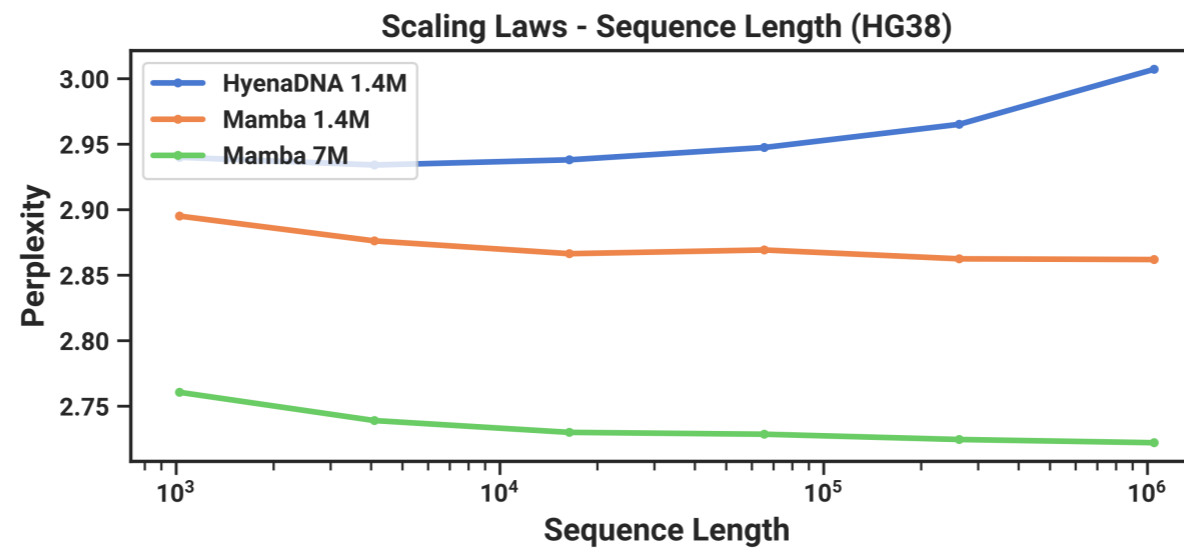


Mamba

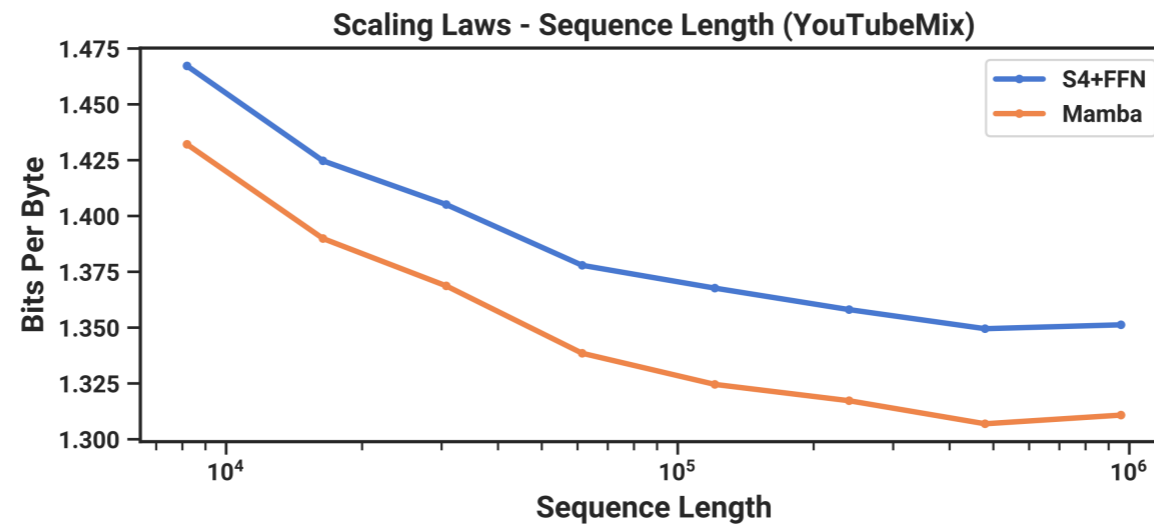
Text Data



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

Thank you