CSCI1470

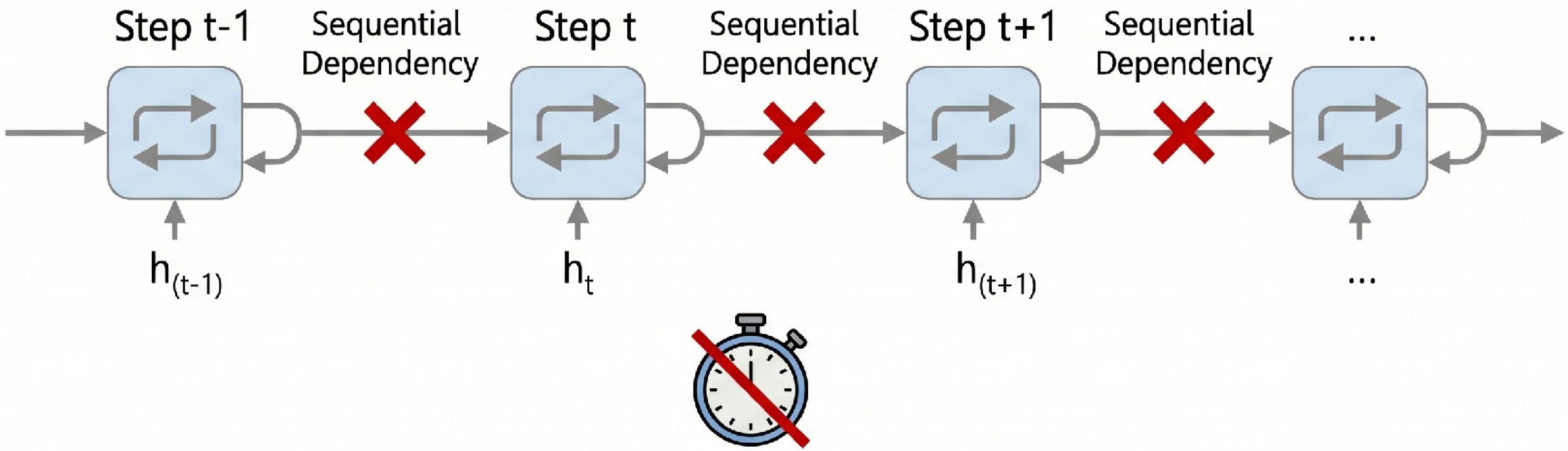
Deep Learning

Randall Balestriero

Recap

Benefits of RNN?

The Bottleneck: Sequential Processing in RNNs/LSTMs

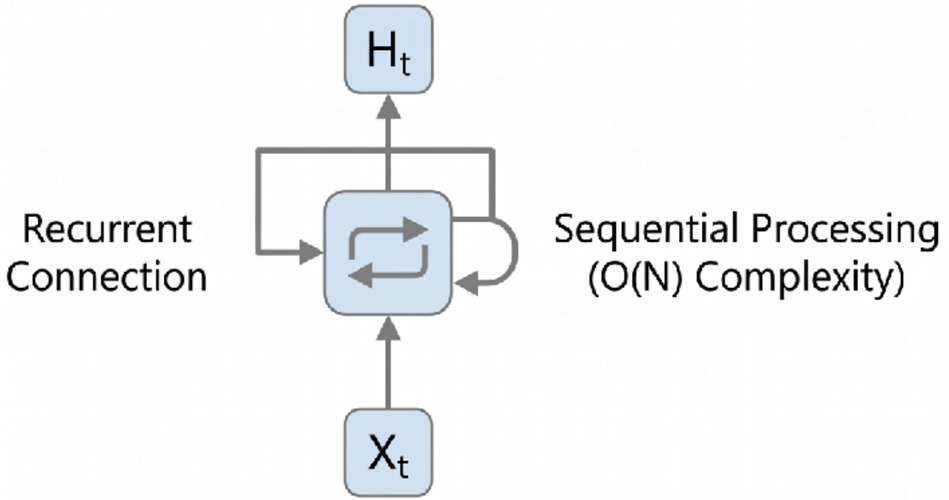


TRAINING BOTTLENECK: Cannot Parallelize (O(N) sequential)

Better alternative?

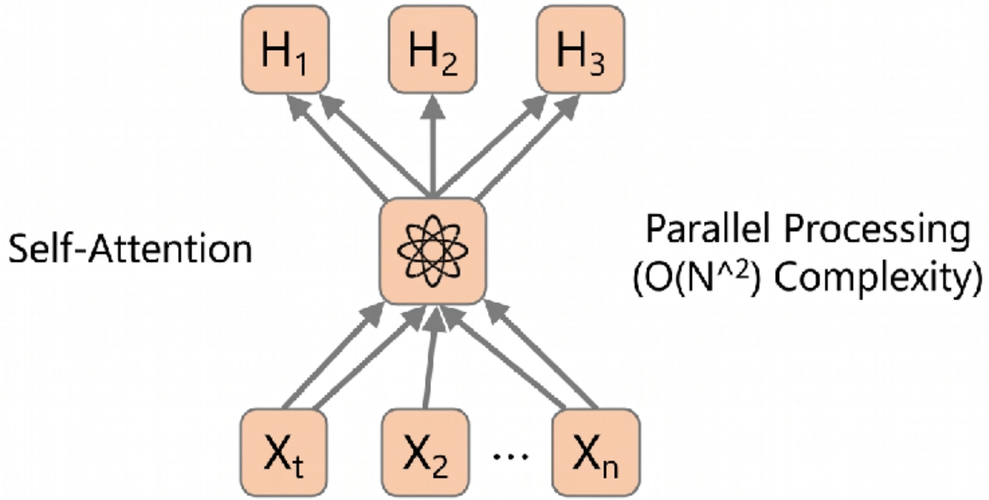
Sequence Modeling: The Shift from RNNs to Transformers

Traditional: Recurrent Architectures (RNN/LSTM)



Sequential Processing
(O(N) Complexity)

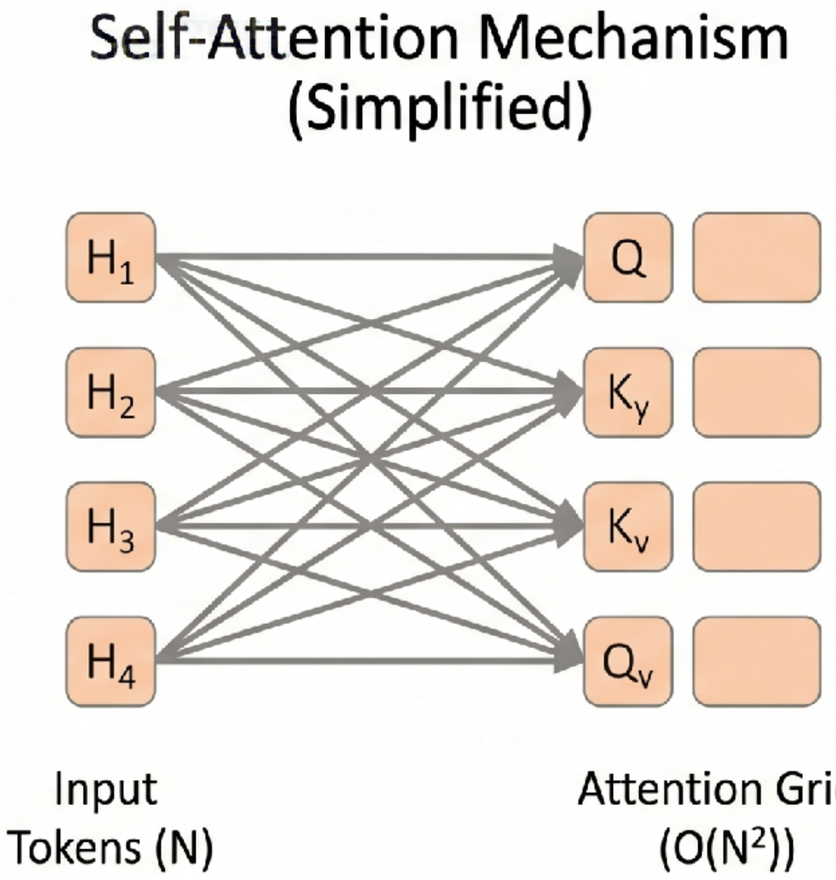
Modern: The Transformer Era (Attention)

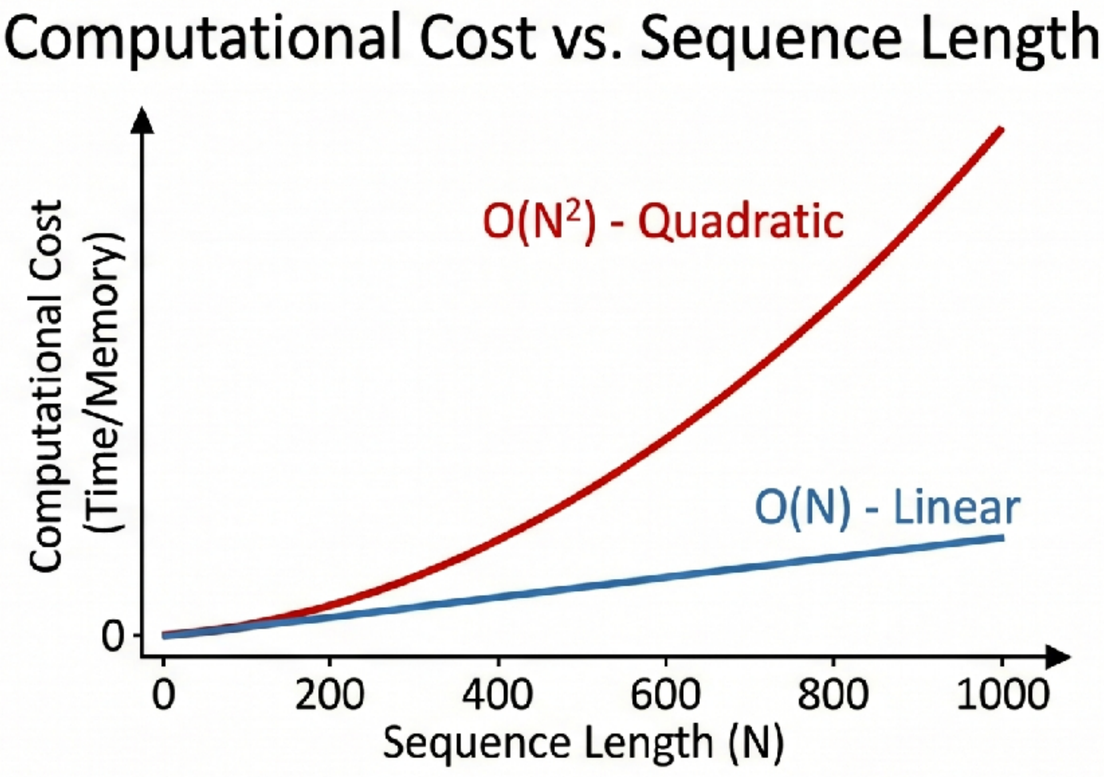


Parallel Processing
(O(N^2) Complexity)

Any problem?

The Cost of Attention: Quadratic Complexity

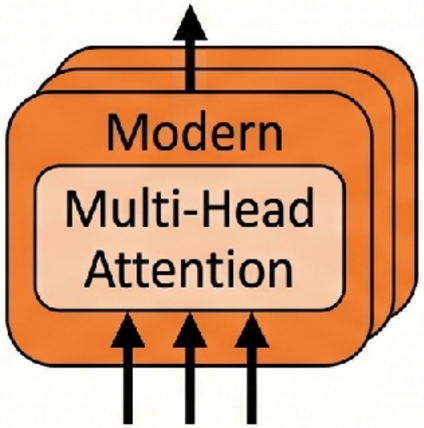




LONG SEQUENCES ARE PROHIBITIVE

The Expressivity vs. Efficiency Trade-off

Transformers
(Attention)

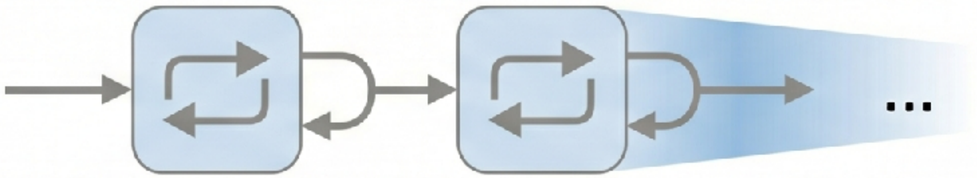


High Expressivity (Long Context)

High Complexity (Quadratic)

?

RNNs
(LSTMs)



Fixed Context (O(1) Memory State)

Low Complexity (Linear)

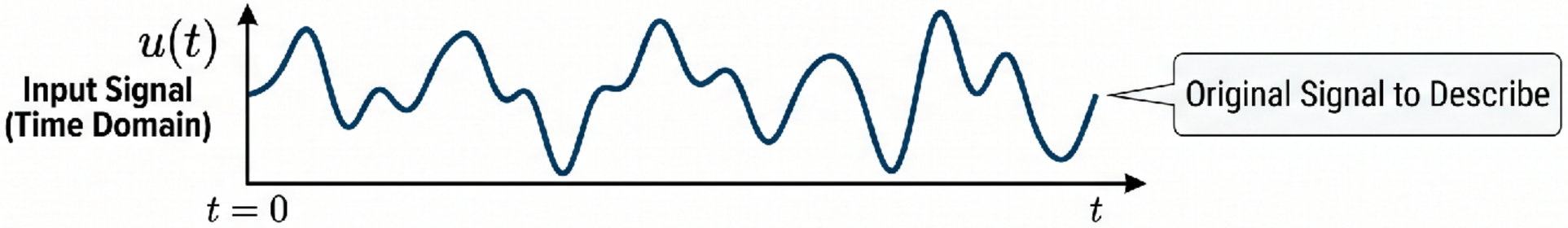
CAN WE ACHIEVE BOTH? (O(N) with High Quality)

What does it mean to “not forget”?

We have information to “reconstruct” all the past!

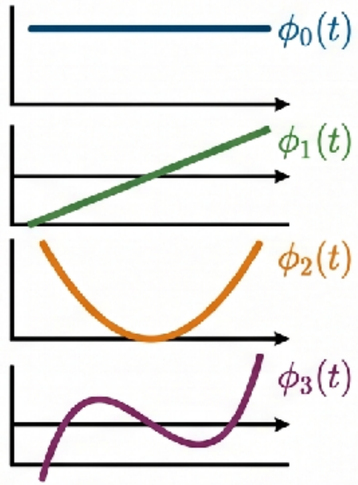
GENERAL SIGNAL REPRESENTATION: EXPLORING BASIS AND PROJECTION

Understanding how basic functions describe complex data.

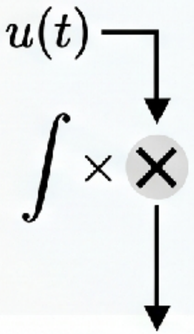


POLYNOMIAL BASIS

Basis Functions $\phi_n(t)$



Family of Base Functions



Projection
(Measuring Contribution)

$$x_n(t) = \int_0^t u(t) \cdot \phi_n(\tau; t)\, d\tau$$

Inner Product: Measures how well the signal matches the basis function

STATE VECTOR $x(t)$
(Coefficients)

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

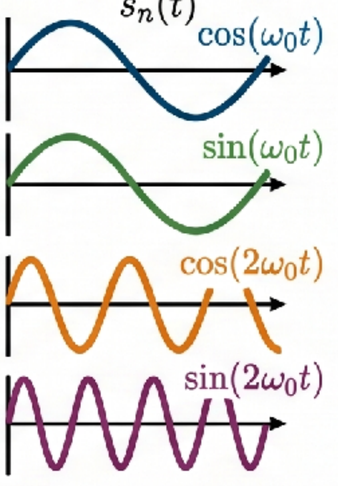
Compact Vector of Learned Coefficients

Description:

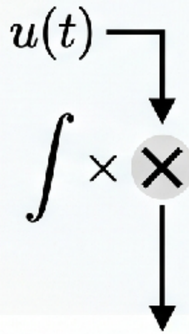
The state contains all info for description

FOURIER-LIKE PERIODIC BASIS

Basis Functions $c_n(t)$, $s_n(t)$



Orthogonal Sine/Cosine Basis Functions



Projection (Frequency Component Measurement)

$$a_n = \frac{1}{T}\int_0^T u(t) \cdot \cos(n\omega_0 t)\, dt$$

$$b_n = \frac{1}{T}\int_0^T u(t) \cdot \sin(n\omega_0 t)\, dt$$

PERIODIC COEFFICIENTS
(a_n, b_n)

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_1 \\ a_2 & b_2 \\ a_3 & b_2 \end{bmatrix}$$

The Vector of Amplitude/ Phase Information

Description:

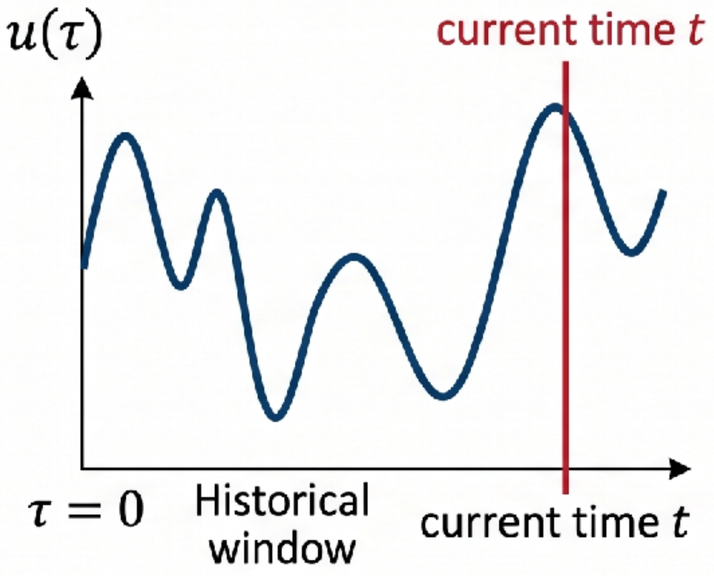
These coefficients also contain all needed info

GENERAL CONCEPT: Any complex signal can be described by choosing a set of basis functions. The projection process creates unique coefficients (the 'state') that encode all information about the original input. This state contains everything needed to fully describe the signal.

How to turn that into a Deep Network?

SSM: HISTORICAL COMPRESSION VIA LEGENDRE PROJECTION

Input History $u(\tau)$ (up to t)

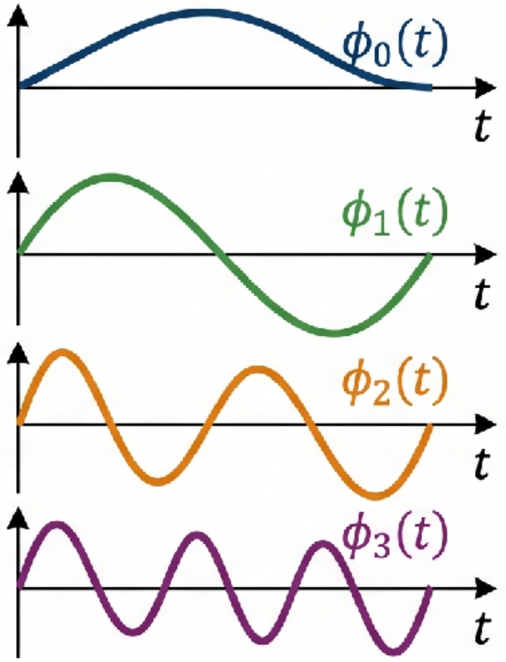


Integral Projection

$\otimes \int \otimes$

Project onto structured basis

LEGENDRE POLYNOMIAL BASIS ($\phi_n(\tau)$)



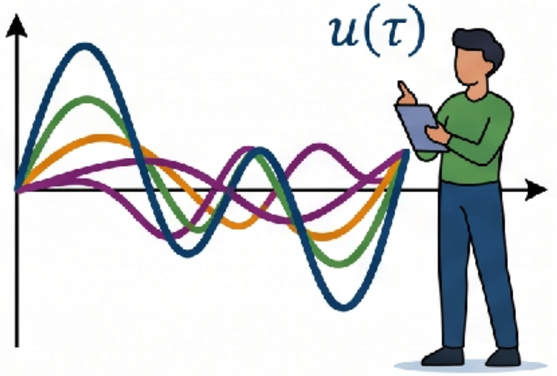
Derived from standard Legendre shapes and so scaled to the window [0, t]

$$\begin{bmatrix} \phi_{00} \\ \phi_{11} \\ \phi_{22} \\ \phi_{33} \\ \vdots \\ \phi_{3n} \end{bmatrix}$$

$x(t)$: THE COMPRESSED STATE (Coefficients)

State Vector

$$x(t) = \begin{bmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \\ \cdots \end{bmatrix}^T$$



Result: A **compact vector** $x(t)$ that 'remembers' everything (low-frequency trends & high-frequency details)

Inner Product Projection:

$$x_n(t) = \int_0^t u(\tau) \cdot \phi_n(\tau; t)\, d\tau$$

(scaled LegS example)

State Vector:

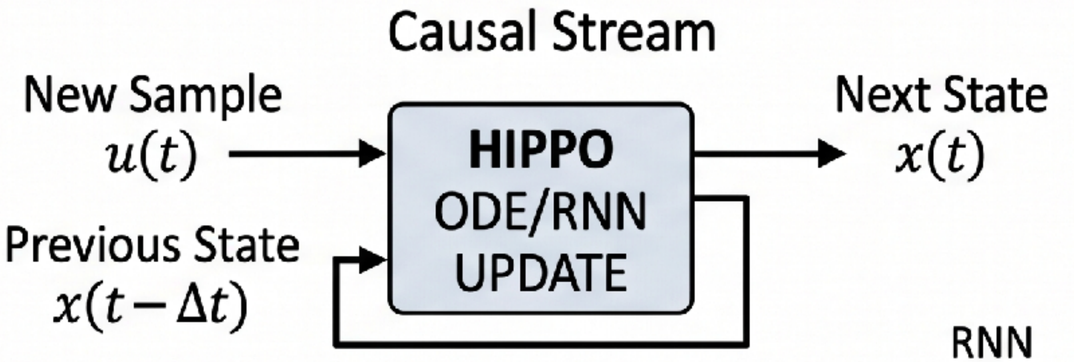
$$x(t) = [x_0(t), x_1(t), x_2(t), x_3(t), \ldots]^T$$

Intuition: The State Vector $x(t)$ is like a 'lossy zip file' of the past, **organizing memory** by frequency components.

COMPUTING THE PROJECTION: TWO IMPLEMENTATIONS

We know the answer $x(t)$. Now, how do we get it efficiently?

HIPPO): INCREMENTAL ONLINE UPDATE (HIPPO)



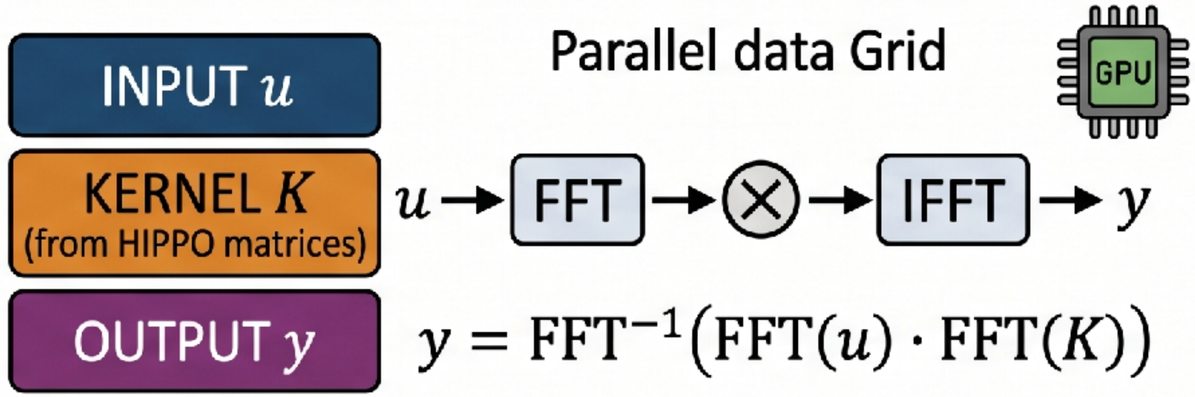
RNN

$$\dot{x}(t) = A_{HIPPO}x(t) + B_{HIPPO}u(t)$$

Good for: Real-time, streaming data, inference.

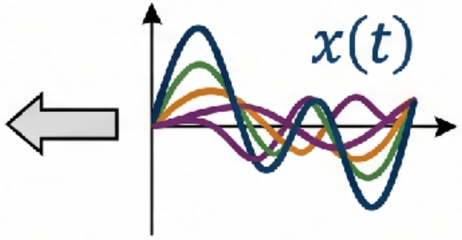
- Computed continuously in time.
- Fixed-size state update.
- Low latency inference.

S4): PARALLEL BATCH COMPUTE (S4)



$$y = \mathrm{FFT}^{-1}(\mathrm{FFT}(u) \cdot \mathrm{FFT}(K))$$

Good for: Parallel training on long sequences.



- Computed globally in parallel.
- Utilizes the Fast Fourier Transform trick for $O(L \log L)$ complexity. Fast parallel training.

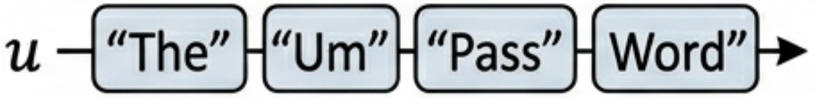
Intuition: HIPPO targets 'Brains' (incremental memory). S4 targets 'Speed' (parallel computation).

DATA-DEPENDENT WEIGHTING: HOW MAMBA SELECTS

S4 - BASELINE:

CONSTANT SAMPLING: NO SELECTION (LTI)

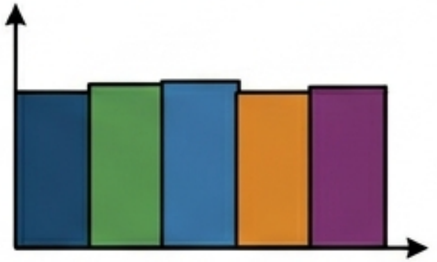
Steady, ticking metronome- fixed Δ



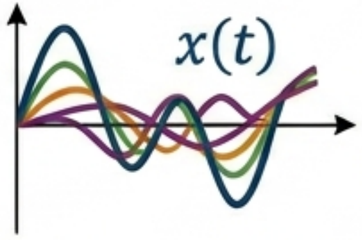
Sampled at an even rate.

$$\bar{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Fixed $\bar{B}$ (derived from constant Δ) applies equal weight al words.



Riemann sum result state is messy.



MAMBA - THE SOLUTION:

DYNAMIC WEIGHTING: SELECTIVE SAMPLING (LTV)

"Physical knob" (or. changes its level for each word based on its importance.

"The" "Um" "Pass" "Word"

$$\bar{B}_t = \text{Discretize}(B, \Delta_t)$$

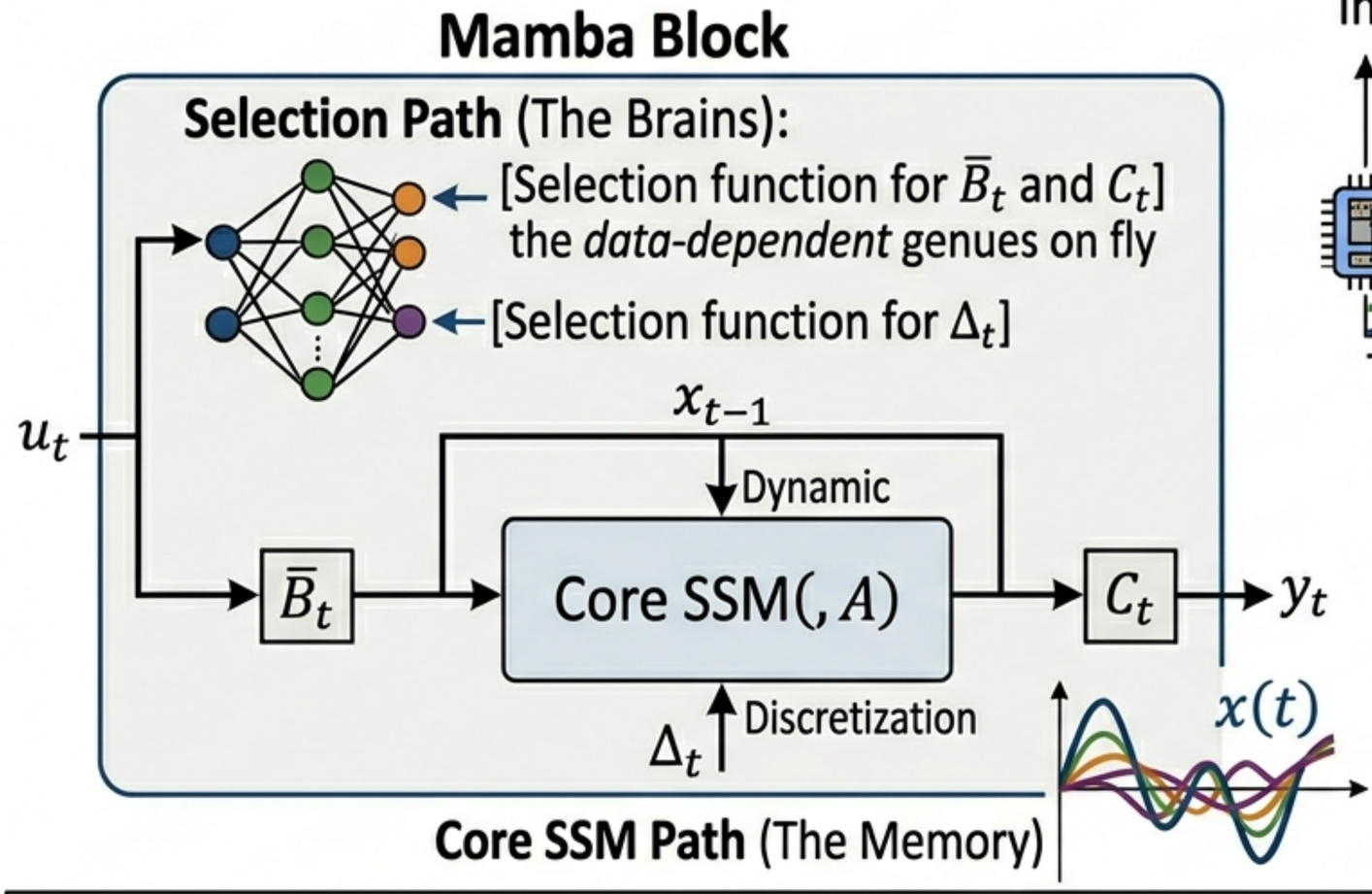
The $\bar{B}_t$ matrix is now a dynamic "gatekeeper" before memory, learned to crush irrelevant data and amplify critical data.

$$\Delta_t = \text{Linear}(u_t)$$

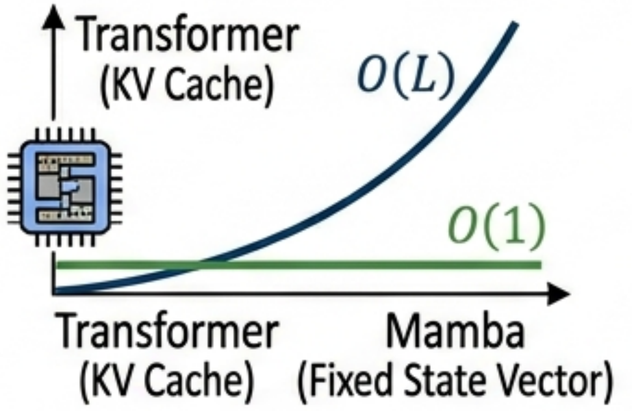
Intuition: Mamba uses Δ_t as a learned 'volume knob' applied to each sample BEFORE projection into memory, effectively acting as dynamic sample weighting.

MAMBA: SELECTIVE STATE SPACE MODEL (SSM) ARCHITECTURE

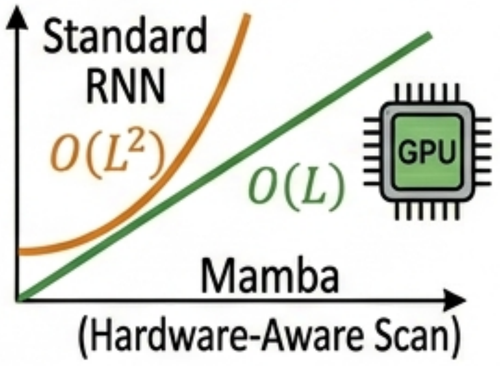
Introducing the complete Mamba architecture.



Memory:
Inference Memory Usage



Speed:
Training Speed vs.
Sequence Length



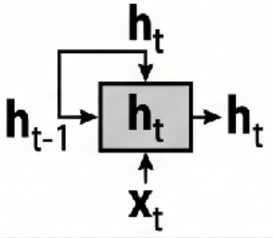
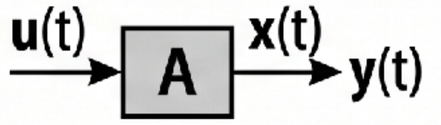
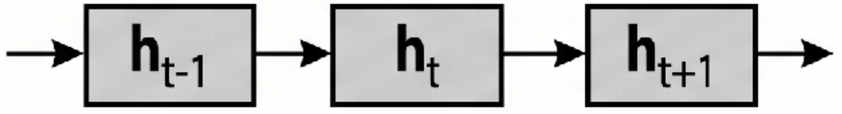
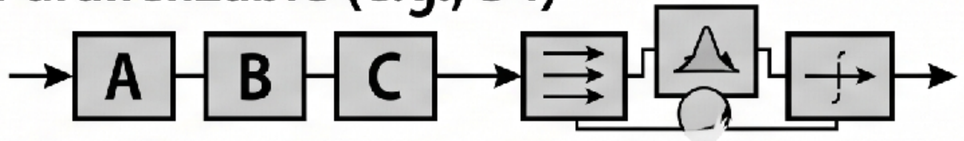
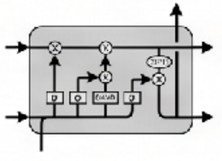
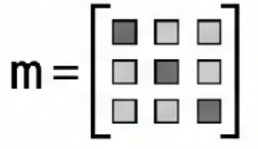
Mamba: The Best of All Worlds

Model	Performance	Memory Efficiency	Selective Memory
Transformer	Low	Medium	Low
S4	High	Medium	Medium
Mamba	**High**	**High**	**High**

Key Innovation: Mamba uses data-dependent parameters for selection and a Hardware-Aware Parallel Scan to bypass the slow RNN loop, creating a linear-time sequence model with Transformer-level performance.

Comparison: RNN vs. SSM

Contrasting key sequential modeling paradigms.

	RNN (Recurrent Neural Network)	SSM (State Space Model)
Core Concept	Vector-based Memory $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$ 	Integral-based State $\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$ 
Inference & Computation	Sequential (O(N))  Mekey Memory Bottleneck	Parallelizable (e.g., S4)  O(log N) training, O(1) inference with structured matrices
Example Models	Vanilla RNN, LSTM, GRU 	S4, HiPPO, Mamba 
Target Applications	NLP, Short Sequences, Standard Time Series	Long Context, Ultra-long Sequences, Interpretable Dynamics, Parallel Training

Thank you!
See you Friday!