Introduction to State Space Models

UG BTech - 2nd Year

Vinay, Subhadeep Sing, Vaibhav Mahore, Snehal Biswas

Indian Institute of Science, Bangalore

April 15, 2025

Project Goal and Research Problem

Our project goal is to explore **Sequential Data Modeling**.
Underlying Research Problem: Traditional sequence models each have key limitations:

- **RNNs:** Capture sequences well but struggle with long dependencies due to vanishing gradients and lack parallelism.
- **Transformers:** Handle long dependencies better, but their self-attention mechanism scales quadratically, limiting their efficiency on long sequences.

The research challenge is: *Can we design models that retain the strengths of these approaches—like parallelism, memory efficiency, and long-range modeling—without inheriting their weaknesses?*

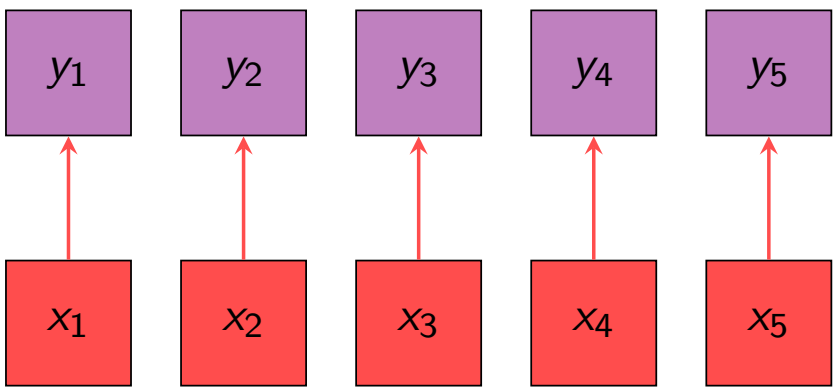
The Need for Sequential Data Modeling

- **What is Sequential Data?**
  - **Ans :** It's the data that comes in a specific order, where the arrangement of the data points matters.
- **Example(NLP):** The dog bites the man vs The man bites the dog.
- **Need for Sequential Data Modeling:** It's crucial because many datasets have an inherent order e.g Language, Time series in which the sequence and context between data points are essential for accurate analysis and predictions.
- **Challenge:** Traditional models (e.g. Feedforward Networks) treat inputs independently and fail to capture such temporal dependencies.

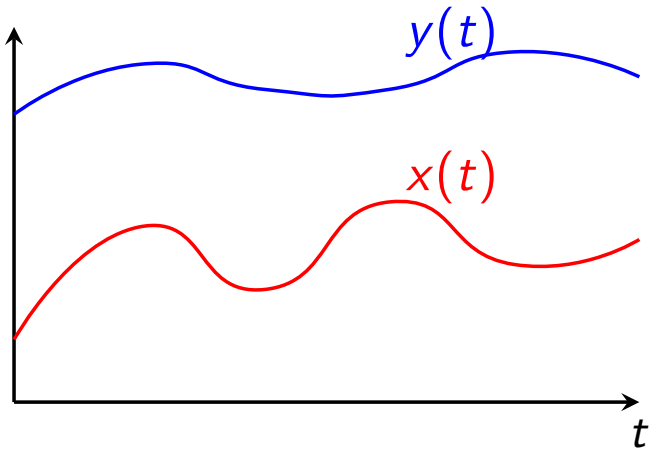
Sequence Modeling

The goal of a sequence model is to map an input sequence to an output sequence:

- Map a continuous input signal $x(t)$ to an output signal $y(t)$
- Or map a discrete input sequence to a discrete output sequence



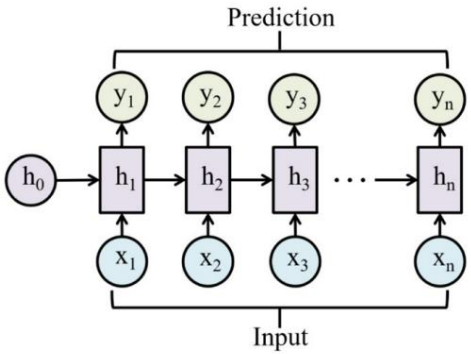
Discrete signal



Continuous signal

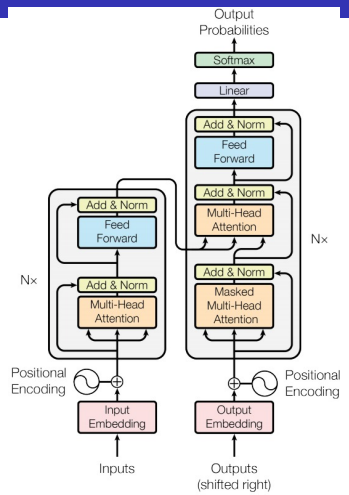
- We can choose among many models to do sequence modeling, let's review them.

Sequence Modeling: Models



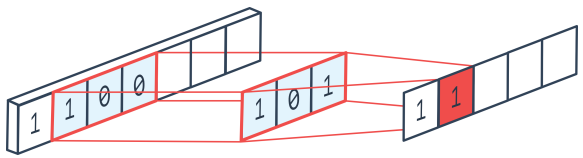
RNN

- Potentially infinite context window
- Practically suffer from vanishing/exploding gradient
- Training O(N), not **not parallelizable**
- Inference: Constant time for each token



Transformer

- Finite context window
- Training: O(N^2), **easily parallelizable**
- Inference: O(N) when using KV-Cache, for each token. This means that if we want to generate the 10th token, we need to perform 10 dot product. To generate the 11th token, we will need to perform 11 dot products, etc.



CNN

- Finite context window (depending on kernel size)
- Need to materialize the kernel before using it.
- Training and inference depend on kernel size.
- **Easily parallelizable**

State Space Models (SSMs)

A State Space Model (SSM) is a mathematical framework used to describe a dynamical system in terms of:

- A hidden state that evolves over time, and
- An observation (output) that depends on that state.

Origin

SSMs originate from control theory and signal processing, but are increasingly used in deep learning — especially for modeling sequences with long-term dependencies (e.g., audio, text, time series).

Linear State Space Layers (LSSL)

LSSL is a specific way to implement SSMs.

Working

It is a simple sequence model that maps a 1-dimensional function or sequence u(t) $\rightarrow$ y(t) through an implicit state $x(t)$ by simulating a linear continuous time state space representation.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$

$$y(t) = Cx(t) + Du(t) \tag{2}$$

where A controls the evolution of the system and B, C, D are projection parameters.

Now we need to **discretize** the equations so that they can be implemented and computed efficiently on digital systems.

Discretization using Generalized Bilinear Transform (GBT)

Continuous System

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$

Approximation and GBT Setup ($\alpha \in [0, 1]$)

Approximate the derivative using backward difference:

$$\dot{x}(t) \approx \frac{x(t) - x(t - \Delta t)}{\Delta t}$$

Assuming $u(t) \approx u(t - \Delta t)$, the GBT approximates the right-hand side as a weighted average:

$$\frac{x(t) - x(t - \Delta t)}{\Delta t} = (1 - \alpha)[Ax(t - \Delta t) + Bu_t] + \alpha[Ax(t) + Bu_t]$$

Discrete-Time Model & Key Cases

Final Discrete System: $\boxed{x_t = \bar{A}\, x_{t-1} + \bar{B}\, u_t}$

$$\bar{A} = (I - \Delta t \alpha A)^{-1}[I + \Delta t(1-\alpha)A], \quad \bar{B} = (I - \Delta t \alpha A)^{-1}\Delta t B$$

Key Cases:

- $\alpha = 0$: Forward Euler $\bar{A} = I + \Delta t\, A, \quad \bar{B} = \Delta t\, B.$
  Simple but may be less stable.
- $\alpha = 1$: Backward Euler $\bar{A} = (I - \Delta t\, A)^{-1}, \quad \bar{B} = (I - \Delta t\, A)^{-1}\Delta t B.$
  Implicit formulation, which enhances stability.
- $\alpha = 0.5$: Trapezoidal

$$\bar{A} = \left(I - \frac{\Delta t}{2}A\right)^{-1}\left(I + \frac{\Delta t}{2}A\right), \quad \bar{B} = \left(I - \frac{\Delta t}{2}A\right)^{-1}\Delta t\, B.$$

  Balances stability and accuracy. Commonly used.

Why Convolution Matters in LSSLs?

We obtained the equation from discretization: $x_t = \bar{A}x_{t-1} + \bar{B}u_t$

- **Unrolling the State Update Equation:**

Assuming $x_0 = 0$:

At $t = 1$**:** $x_1 = \bar{B}u_1$
At $t = 2$**:** $x_2 = \bar{A}\bar{B}u_1 + \bar{B}u_2$
General case: $x_t = \sum_{k=0}^{t-1} \bar{A}^k \bar{B}\, u_{t-k}$

Output Equation via Convolution:

$$\begin{aligned} y_k &= C\bar{A}^k\bar{B}u_0 + C\bar{A}^{k-1}\bar{B}u_1 + \cdots + C\bar{A}\bar{B}u_{k-1} + Du_k \\ &= \underbrace{(C\bar{B}, C\bar{A}\bar{B}, \ldots, C\bar{A}^{L-1}\bar{B})}_{K_L(\bar{A},\bar{B},C)} * u + Du \end{aligned}$$

Why is this useful?

- **Parallel Computation:** Convolutions compute all x_t and y_t in parallel using FFT-based methods.

The HiPPO Framework: High-order Polynomial Projection Operators

- **Core Problem:** Online function approximation via projection
- Project continuous function history $f(t) : \mathbb{R}^+ \to \mathbb{R}$ onto polynomial basis

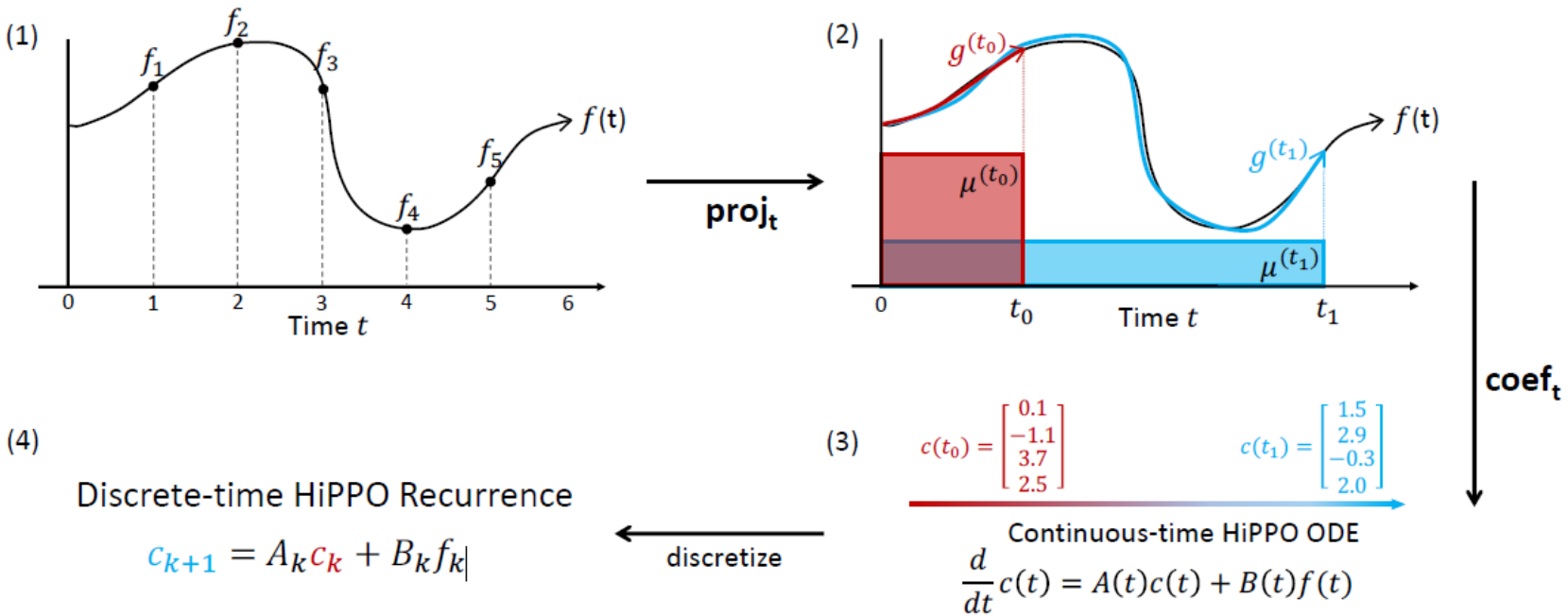
HiPPO Definition

Given a time-varying measure family $\mu^{(t)}$ supported on $(-\infty, t]$, an N-dimensional subspace G of polynomials, and a continuous function f:

- proj$t$: Maps $f_{\leq t}$ to polynomial $g^{(t)} \in G$ minimizing $\|f_{\leq t} - g^{(t)}\|_{L^2(\mu^{(t)})}$
- coef$_t$: Maps $g^{(t)}$ to coefficients $c(t) \in \mathbb{R}^N$ in orthogonal polynomial basis

hippo $=$ coef $\circ$ proj maps f to optimal projection coefficients $c(t)$

HiPPO Framework Visualization and Dynamics



Key Insight

For suitable measures, coefficients $c(t)$ follow: $\frac{d}{dt}c(t) = A(t)c(t) + B(t)f(t)$, where $A(t) \in \mathbb{R}^{N\times N}$, $B(t) \in \mathbb{R}^{N\times 1}$.

HiPPO Variants and Measures

Legendre (LegT)

$\mu^{(t)}(x) = \frac{1}{\theta} I_{[t-\theta,t]}(x)$,

$A_{nk} = \frac{1}{\theta} \begin{cases} (-1)^{n-k}(2n+1) & n \geq k \\ 2n+1 & n \leq k \end{cases}$,

$B_n = \frac{1}{\theta}(2n+1)(-1)^n$

LegT: Legendre polynomials on $[t-\theta, t]$, uniform weight on recent history.

Laguerre (LagT)

$\mu^{(t)}(x) = e^{-(t-x)} I_{(-\infty,t]}(x)$,

$A_{nk} = \begin{cases} 1 & n \geq k \\ 0 & n < k \end{cases}$, $B_n = 1$

LagT: Laguerre functions with exponential decay, emphasize recent inputs.

Scaled Legendre (LegS)

$\mu^{(t)} = \frac{1}{t} I_{[0,t]}, \quad \frac{d}{dt} c(t) = -\frac{1}{t} A c(t) + \frac{1}{t} B f(t), \quad c_{k+1} = \left(1 - \frac{A}{k}\right) c_k + \frac{1}{k} B f_k$

$$A_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & n > k \\ n+1 & n = k \\ 0 & n < k \end{cases}, \quad B_n = (2n+1)^{1/2}$$

LegS: Scaled Legendre over $[0, t]$, uniform weight on all history, time-scaled dynamics.

Structured State Spaces (S4)

- **Goal:** Our goal is to efficiently compute the power of $\bar{A}^K$ by reducing time complexity, leveraging techniques like FFT (Fast Fourier Transform) to improve computational efficiency.

Lemma (Conjugation Equivalence)

$(A, B, C) \sim (V^{-1}AV, V^{-1}B, CV)$

- Diagonal A makes computation efficient (Vandermonde structure).
- But diagonalizing HiPPO matrix is unstable:

HiPPO Matrix Diagonalization

$V_{ij} = \binom{i+j}{i-j}, \quad V_{3i,i} = \binom{4i}{2i} \approx 2^{4i}$

$\Rightarrow$ Entries in V can reach up to $2^{4N/3}$

The S4 Parameterization: Normal Plus Low-Rank

- Observation: HiPPO matrix can be decomposed as normal plus low-rank

HiPPO matrices can be expressed as the difference between a normal matrix and a low-rank correction:

$$A = V\Lambda V^* - PQ^\top$$

- $V \in \mathbb{C}^{N\times N}$: unitary matrix
- Λ: diagonal matrix (eigenvalues of normal part)
- $P, Q \in \mathbb{R}^{N\times r}$: low-rank factors
- $r = \text{rank}(PQ^\top) \in \{1, 2\}$

The term $PQ^\top$ introduces a **low-rank correction** (rank 1 or 2) to the normal matrix $V\Lambda V^*$.

Efficient Convolution in S4 via Fourier Methods

S4 performs convolution efficiently by using the Fourier transform instead of computing it directly in the time domain. The steps are:

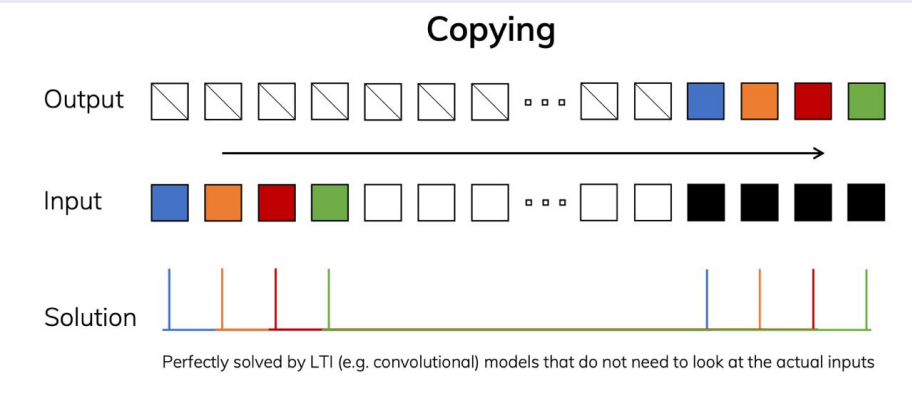
1. **Choose a length L:** Truncate the kernel to length L, since later terms often have minimal impact.
2. **Transform to frequency domain and multiply:** Use FFT to convert both the input and kernel to the frequency domain, where convolution becomes fast elementwise multiplication, avoiding costly matrix operations.
3. **Inverse FFT:** Apply IFFT to recover the final output in the time domain.

Efficiency: This entire process only takes $\mathcal{O}(L \log L)$ operations using FFT/IFFT, compared to the naive $\mathcal{O}(N^2 L)$ time required for direct convolution involving matrix multiplications.

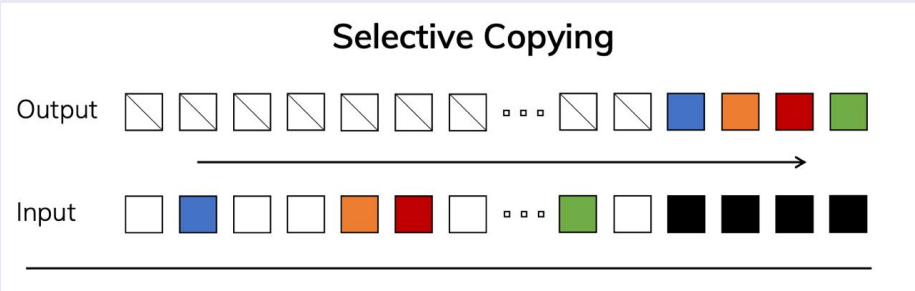
Motivation for Better Models

- The authors identify two tasks on which vanilla SSM or even S4 do not perform well, which motivates the development of MAMBA.
- The **MAMBA** paper proposes **selectivity** as the fundamental principle for building improved sequence models.

Experiment 1: Selective Copying Task

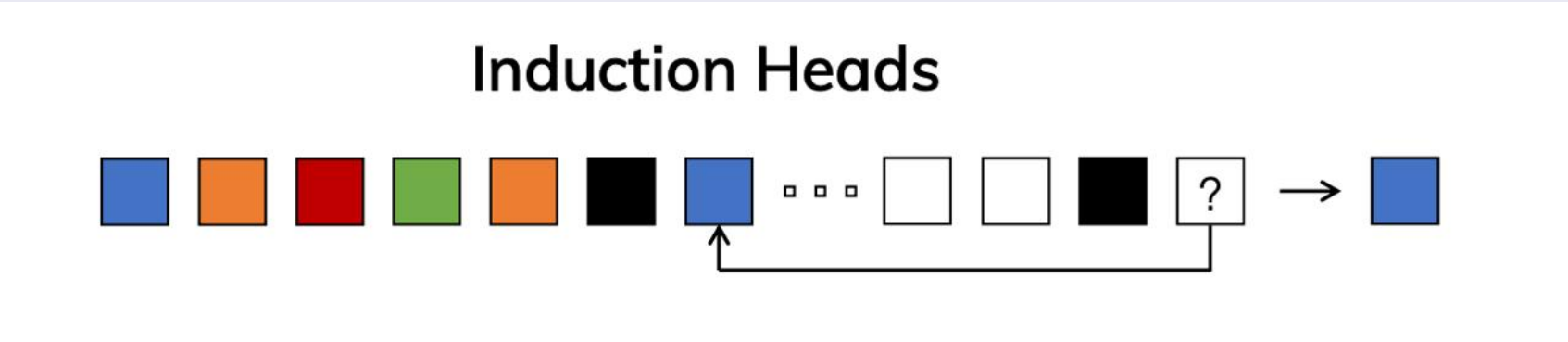


- **Intuition:** Rewrite the input one token at a time, but time-shifted.
- Can be done by a vanilla SSM.
- Time delay can be learned by convolution.



- **Intuition:** Given a comment on Twitter, rewrite the comment by removing all the bad words (the white tokens)
- Cannot be done by vanilla SSM.
- Needs content-aware reasoning, not supported by time-invariant systems.

Experiment 2: Induction Heads Task



Intuition: For example with "few-shot" prompting, we can "teach" LLMs new tasks and how to perform them. Transformer-based models can do this because they attend to previous tokens when generating the current one—so they can "recall previous history".

Limitation: This task **cannot** be performed by a time-invariant SSM, because they cannot *select* which previous token to recall from history.

MAMBA: A Selective SSM

Selection Mechanism

MAMBA implement's selection mechanism by making parameters Δ, B, C **input dependent**

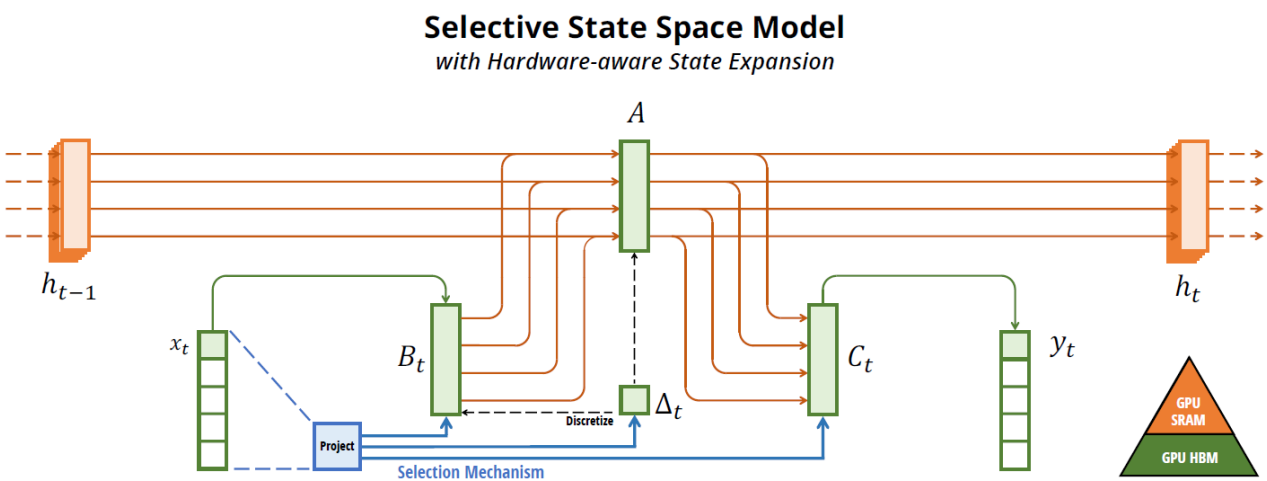
- $s_B(x_t) = Linear_N(x_t)$ generates a token-specific vector B_t from the input embedding x_t via a linear transformation.
- $s_C(x_t) = Linear_N(x_t)$ similarly produces a token-dependent output mapping transformation C_t.
- $\tau_\Delta(\text{Parameter} + s_\Delta(x_t))$: Here $s_\Delta(x_t) = \text{Broadcast}_D\left(\text{Linear}_1(x_t)\right)$ is a learnable linear function applied to each token embedding x_t and then broadcast to match the dimension D. A base parameter is also added to adjust the scale. The combined output passes through τ_Δ , a *softplus* function to ensure that the resulting step size is strictly positive.
- The paper state while $\mathbf{A}$ could also be selective it ultimately affects the model only through its interaction with Δ via $\bar{\mathbf{A}} = exp(\Delta\mathbf{A})$. Thus selectivity in Δ is enough to ensure selectivity in $\bar{\mathbf{A}}$ and is the main source of improvment.

MAMBA: A Selective SSM

Efficient Implementation

The paper introduces several methods for efficient implementation

- Kernel Fusion
- Parallel scan
- Recomputation



1. Selective Copying

Details

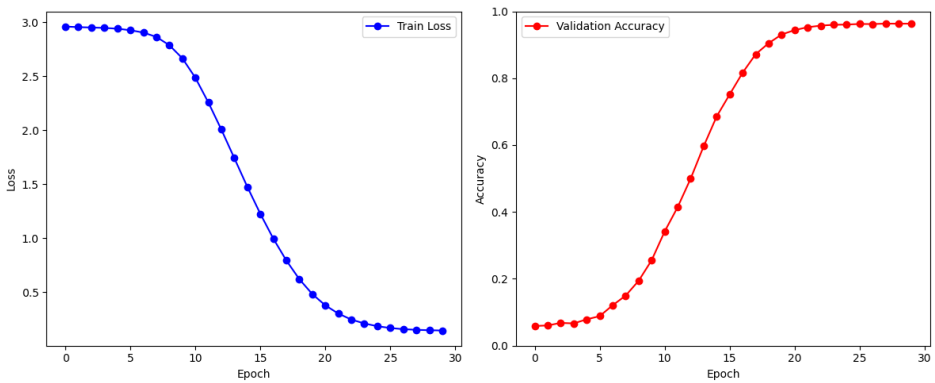
We created a dataset with a vocabulary size of 30 and trained models on sequences of length 64. Each training sequence contains 3 to 5 markers. The training set consists of 10,000 samples. Upon training we got around $\sim$ 96.31 % accuracy at around 30 epoches.

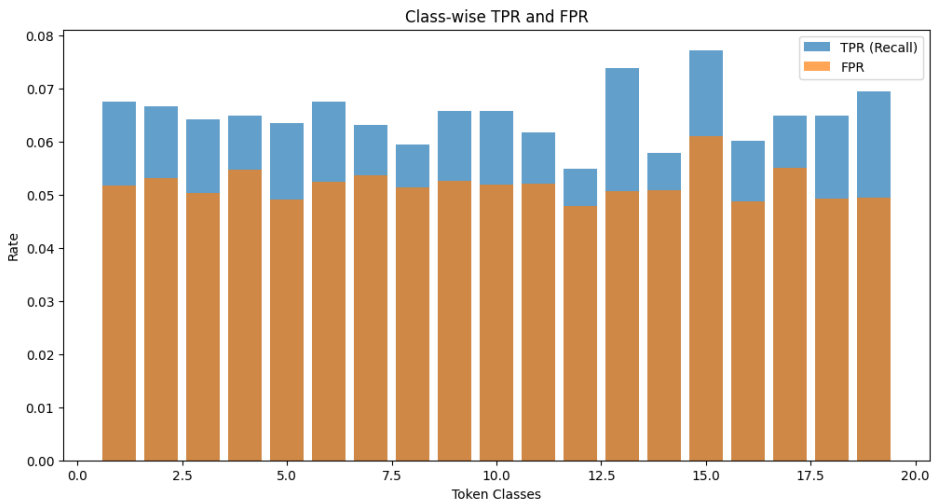
We replicate this task on **MAMBA** with $\sim$ 1 million parameters.

Testing

We test our model on 5000 samples with sequence length of 128 where each sequence contains 5 to 10 markers.

Selective Copying: Results





2. IMDb Sentiment Analysis

We classified 50,000 IMDb reviews as positive or negative using the S4 and Mamba models

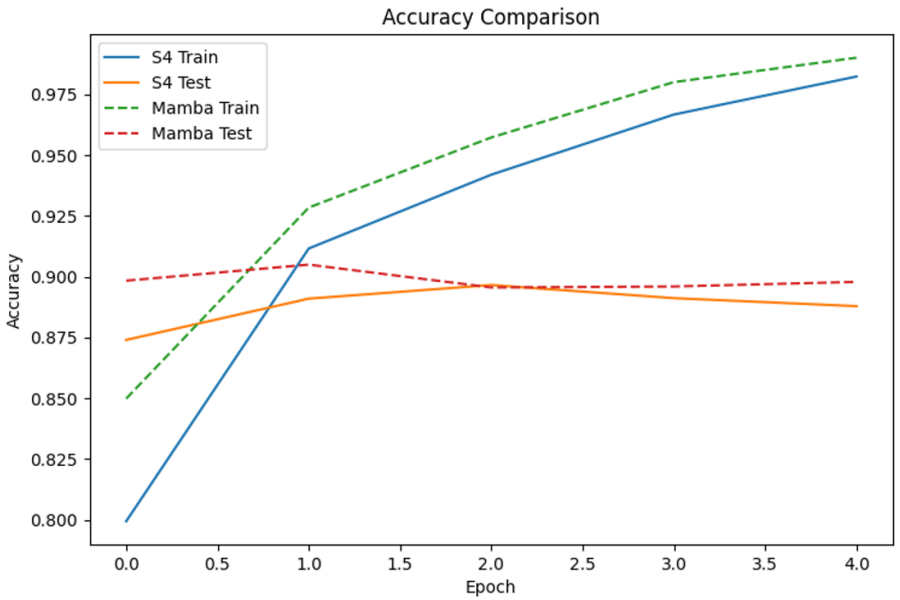


Figure: Accuracy Comparison

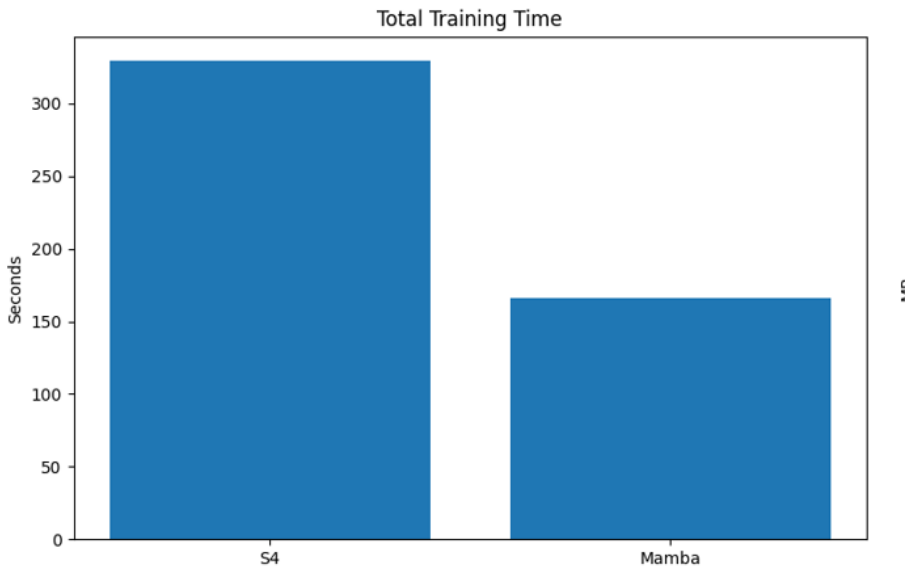


Figure: Training Time Comparison

Confusion Matrix and ROC Curve

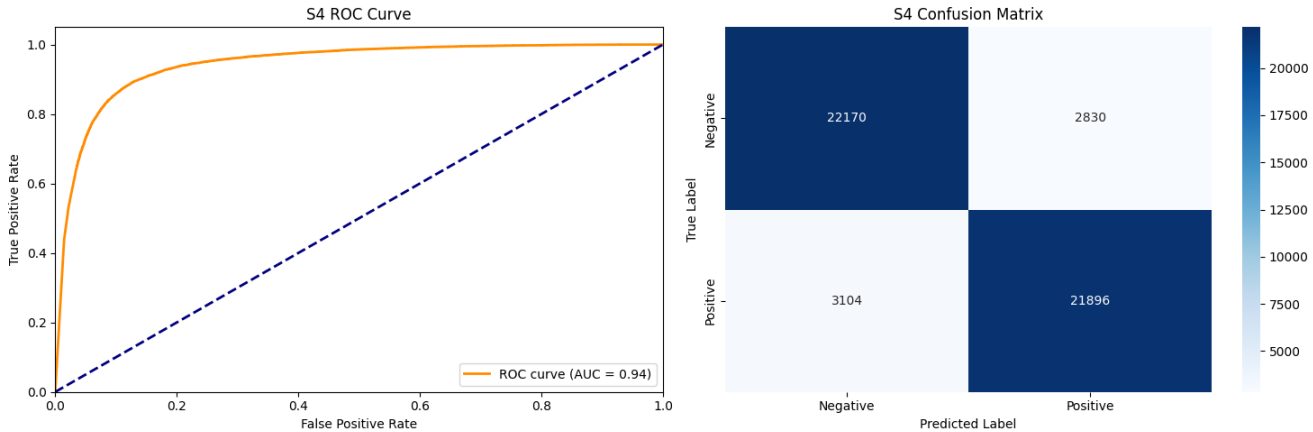


Figure: S4 – Confusion Matrix and ROC Curve

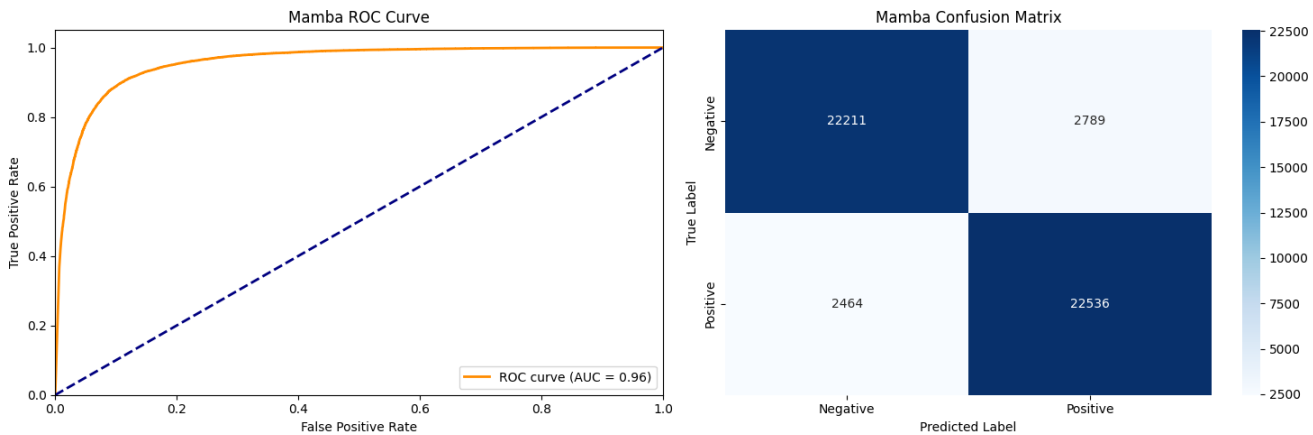


Figure: Mamba– Confusion Matrix and ROC Curve

To view the code, check out the GitHub repo:

`https://github.com/Vinay-Karman/Intro-to-SSM.git`

Project Progress

Step 1: From LSSL to HiPPO

- Started with LSSL for modeling sequential data.
- Used HiPPO matrices **A** to maintain memory and capture long-range dependencies.

Step 2: Efficient Computation via S4

- Developed S4, enabling fast and scalable sequence modeling.
- Leveraged FFT and *NPLR* to efficiently compute $\bar{A}^k$

Step 3: Advancing with Mamba

- Introduced Mamba for selectivity in sequence modeling.
- The MAMBA paper states that, on language modeling,Mamba-3B model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream.

Team Member Contributions

Name	**Contribution**
Vinay	LSSL (30%), S4 (60%), MAMBA (100%), Experiments - Selective Copying (100%), Project Report (60%), Presentation (40%)
Subhadeep Sing	LSSL (60%), HiPPO (80%), S4 (40%), Experiments - IMDb (100%), Project Report (30%), Presentation (20%)
Vaibhav Mahore	HiPPO (10%), Presentation (40%)
Snehal Biswas	HiPPO (10%),

Table: Individual contributions to different components of the project

Thank You

Any Questions?