中国人民大学
RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院
Gaoling School of Artificial Intelligence

Efficiently Modeling Long Sequences with Structured State Spaces

Albert Gu, Karan Goel, and Christopher Re

Department of Computer Science, Stanford University

Shen Yuan

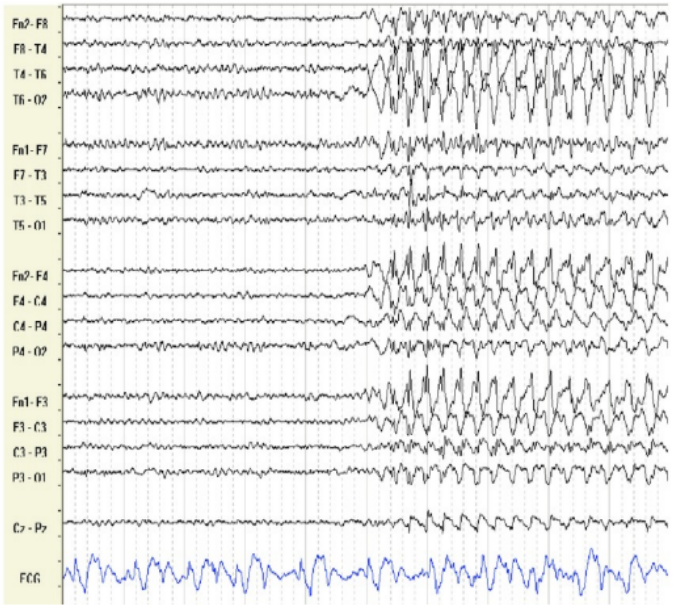
2023-7-19

Outline

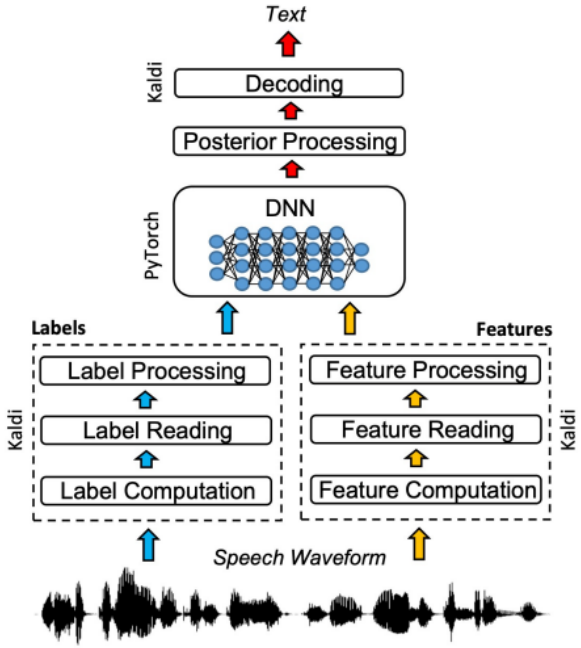
- Introduction
- State space models (SSM) for deep sequence modeling
- Structured state spaces (S4) for long-term dependencies
- Experiments
- Conclusion

- Introduction
- State space models (SSM) for deep sequence modeling
- Structured state spaces (S4) for long-term dependencies
- Experiments
- Conclusion

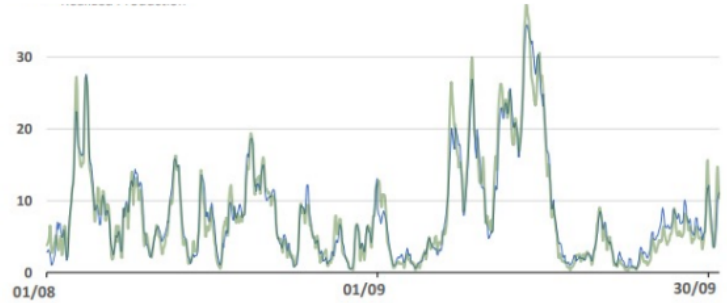
Long "Continuous" Time Series



EEG/ECG



Speech



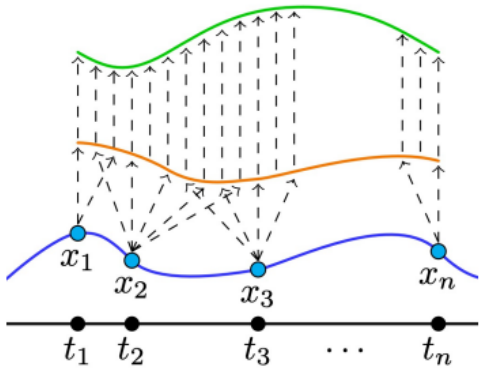
Energy Forecasting

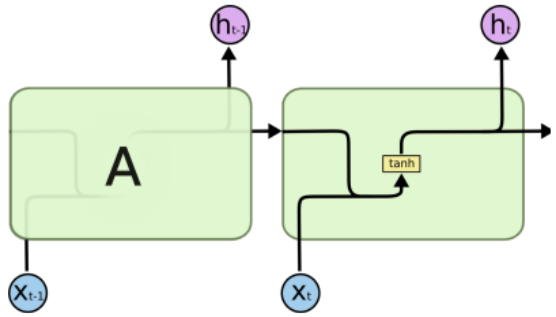
- Time series sampled from an underlying (continuous) physical process

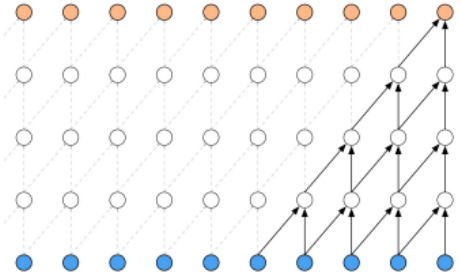
Sequence Modeling Paradigms

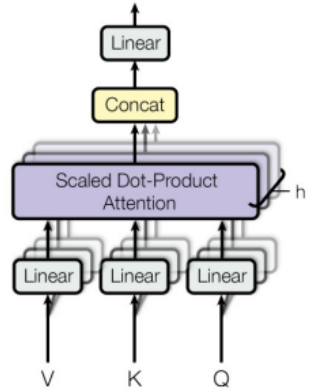
高瓴人工智能学院
Gaoling School of Artificial Intelligence

Continuous-time Model	Recurrent Neural Net.	Convolutional Neural Net.	Transformer

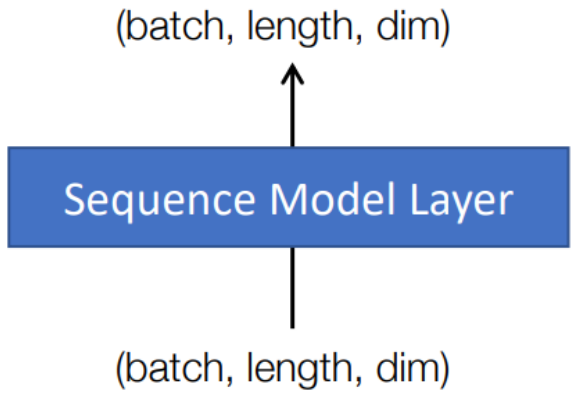




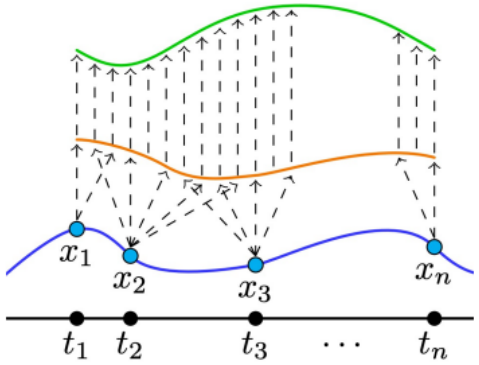




Deep Sequence Model
Sequence-to-sequence map



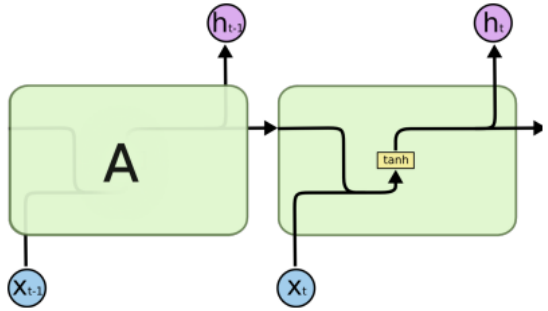
Paradigms for Long Time Series



✓ Continuous data
Irregular sampling

✗ Complex, very inefficient
Vanishing gradients

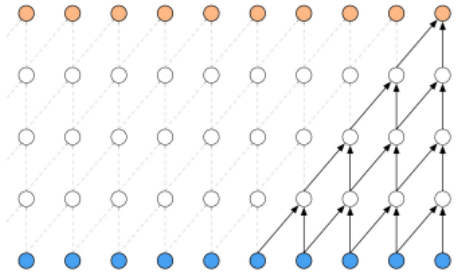
Continuous-time (CTM)



✓ Unbounded context
Stateful inference

✗ Inefficient training
Vanishing gradients

Recurrent (RNN)



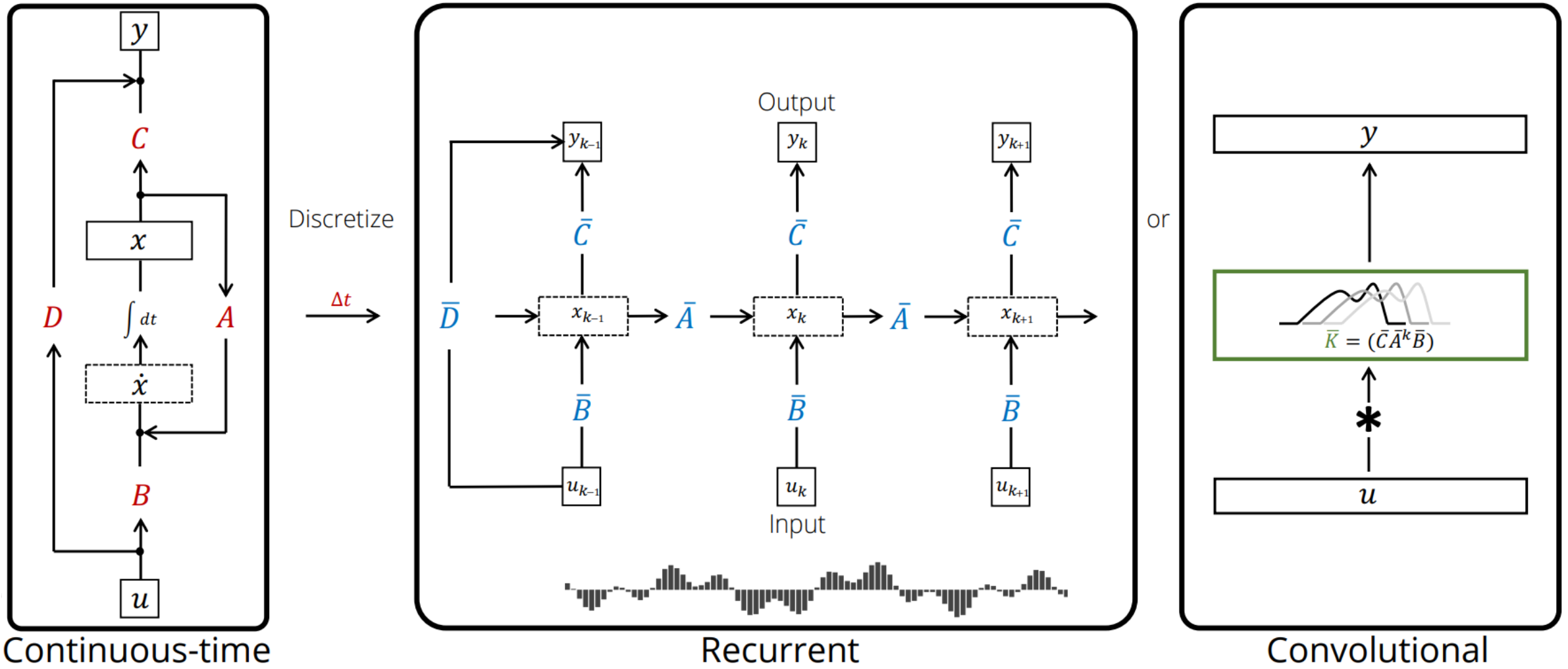
✓ Easy optimization
Parallelizable training

✗ Inefficient inference
Bounded context

Convolutional (CNN)

- Existing model families have clear tradeoffs
- All struggle with long-range dependencies (LRD)

Structured State Spaces (S4)



- Introduction
- State space models (SSM) for deep sequence modeling
- Structured state spaces (S4) for long-term dependencies
- Experiments
- Conclusion

Computing with SSMs: Continuous View

Input → State

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$

$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)$$

Parameters

$\boldsymbol{A} \in \mathbb{R}^{N \times N}$

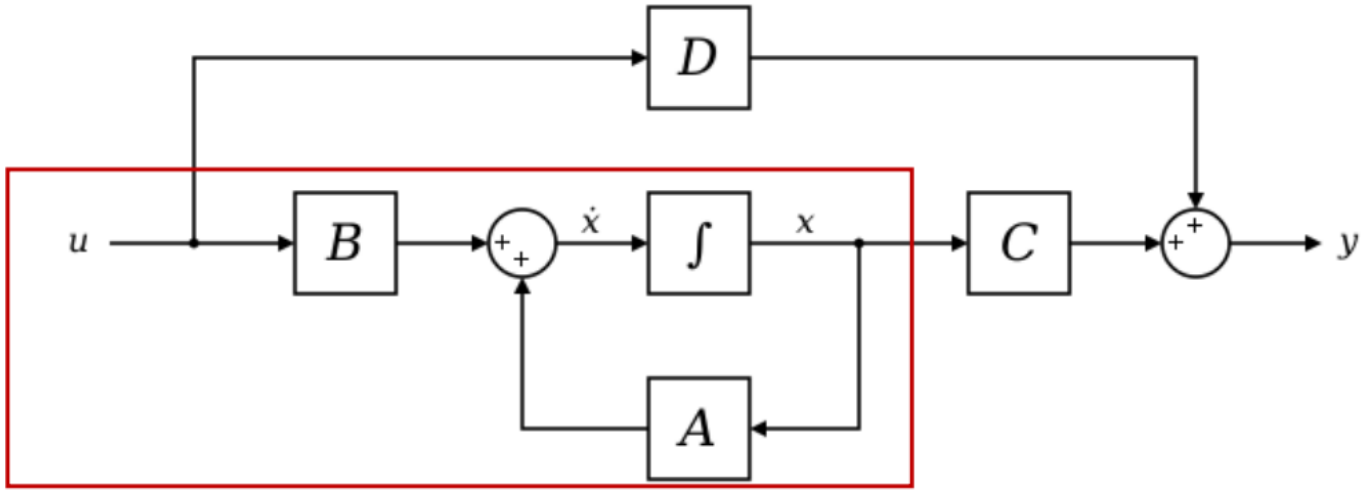
$\boldsymbol{B} \in \mathbb{R}^{N \times 1}$

$\boldsymbol{C} \in \mathbb{R}^{1 \times N}$

$\boldsymbol{D} \in \mathbb{R}^{1 \times 1}$

Function-to-function map $u(t) \mapsto y(t)$

1D input
u(t)



1D output
y(t)

Computing with SSMs: Continuous View

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$

State → Output

$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)$$

Parameters

$\boldsymbol{A} \in \mathbb{R}^{N \times N}$

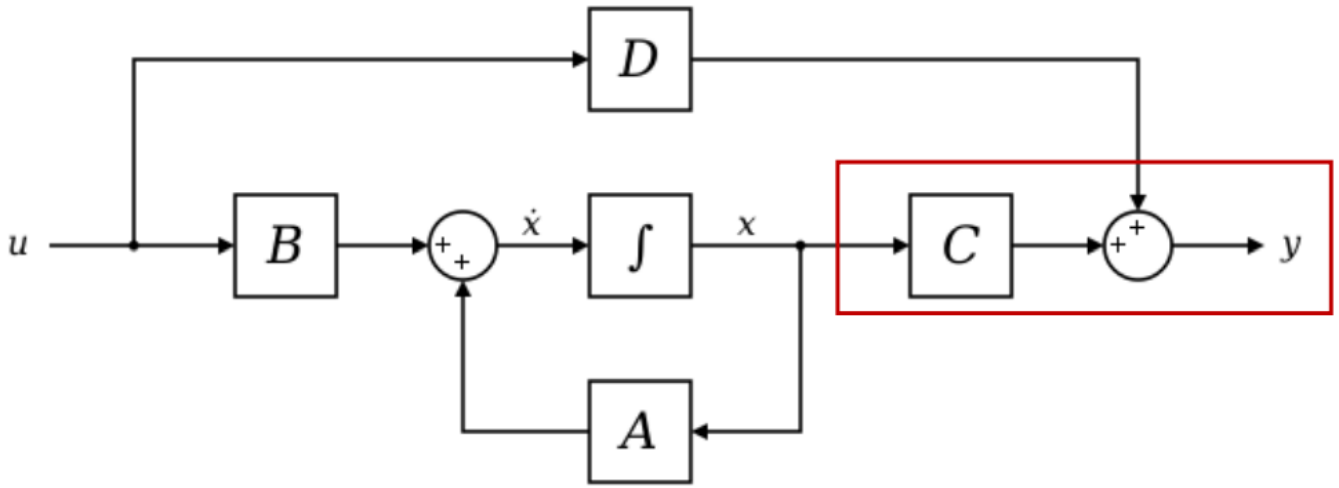
$\boldsymbol{B} \in \mathbb{R}^{N \times 1}$

$\boldsymbol{C} \in \mathbb{R}^{1 \times N}$

$\boldsymbol{D} \in \mathbb{R}^{1 \times 1}$

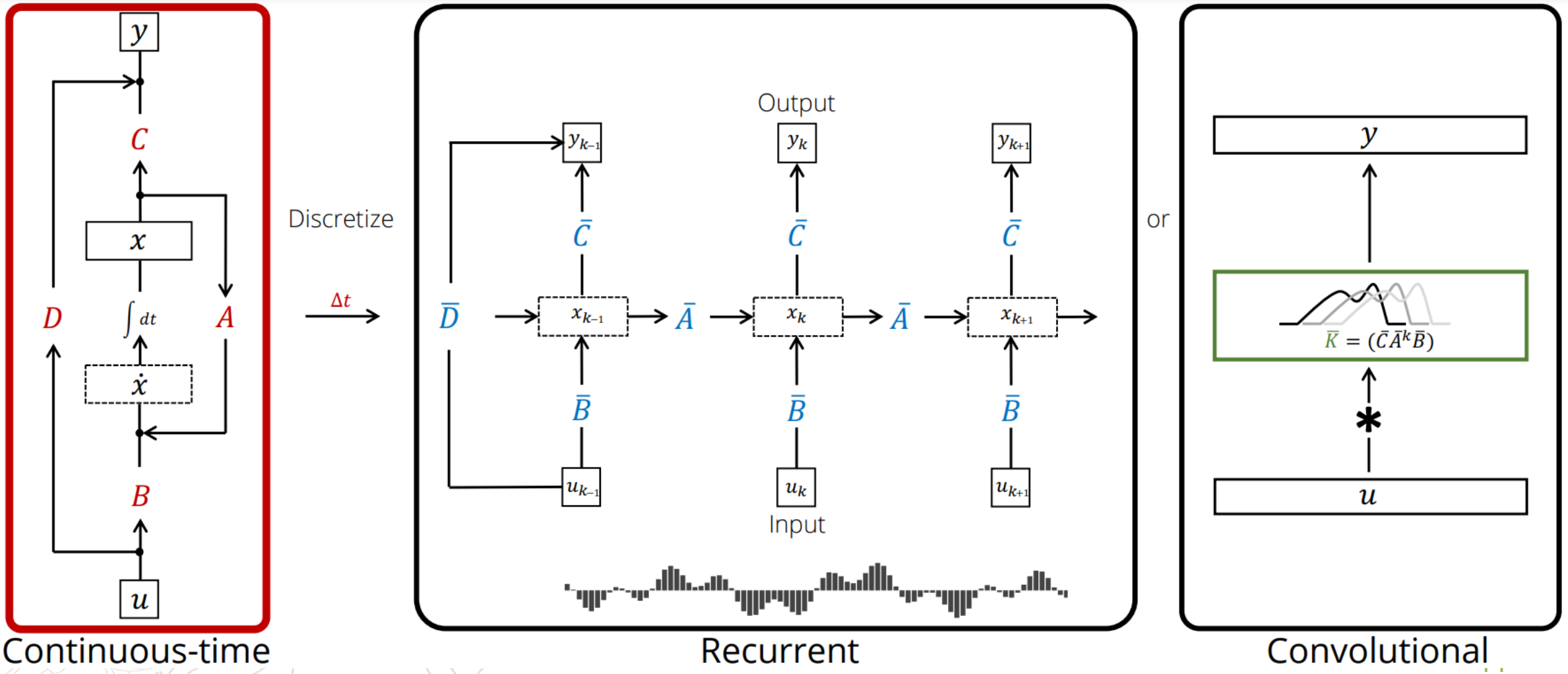
Function-to-function map $u(t) \mapsto y(t)$

1D input
u(t)



1D output
y(t)

SSM: Continuous Representation



Computing with SSMs: Recurrent View

Conceptually, the inputs u_k can be viewed as **sampling an implicit underlying continuous signal** u(t), where $u_k = u(k\Delta)$.

To discretize the continuous-time SSM, we follow prior work in using **the bilinear method**, which converts **the state matrix** $\boldsymbol{A}$ into an **approximation** $\overline{\boldsymbol{A}}$. The discrete SSM is

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k \qquad \overline{\boldsymbol{A}} = (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1}(\boldsymbol{I} + \Delta/2 \cdot \boldsymbol{A})$$
$$y_k = \overline{\boldsymbol{C}} x_k \qquad \overline{\boldsymbol{B}} = (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1} \Delta \boldsymbol{B} \qquad \overline{\boldsymbol{C}} = \boldsymbol{C}.$$

Parameters

$\boldsymbol{A} \in \mathbb{R}^{N \times N}$

$\boldsymbol{B} \in \mathbb{R}^{N \times 1}$

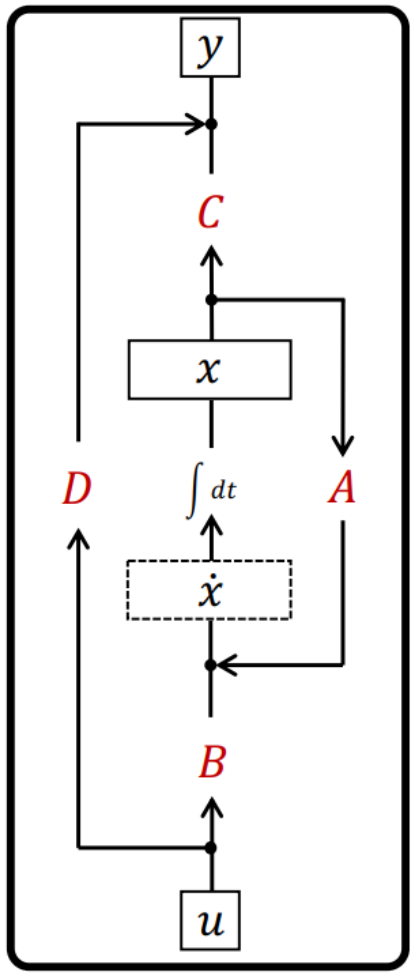
$\boldsymbol{C} \in \mathbb{R}^{1 \times N}$

$\boldsymbol{D} \in \mathbb{R}^{1 \times 1}$

$\Delta \in \mathbb{R}$

Now it's a *sequence-to-sequence* map $u(t) \mapsto y(t)$ instead of *function-to-function map*.

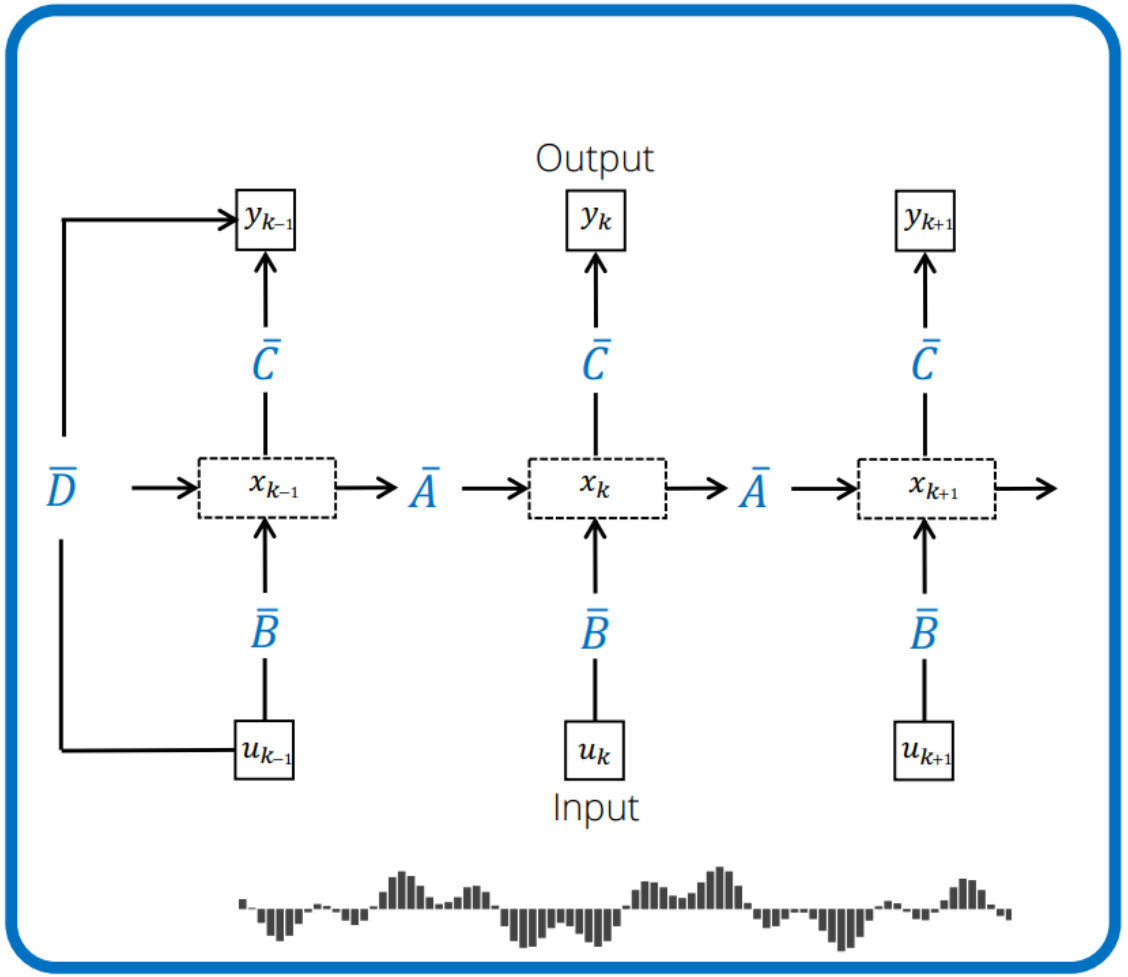
SSM: Recurrent Representation



Continuous-time

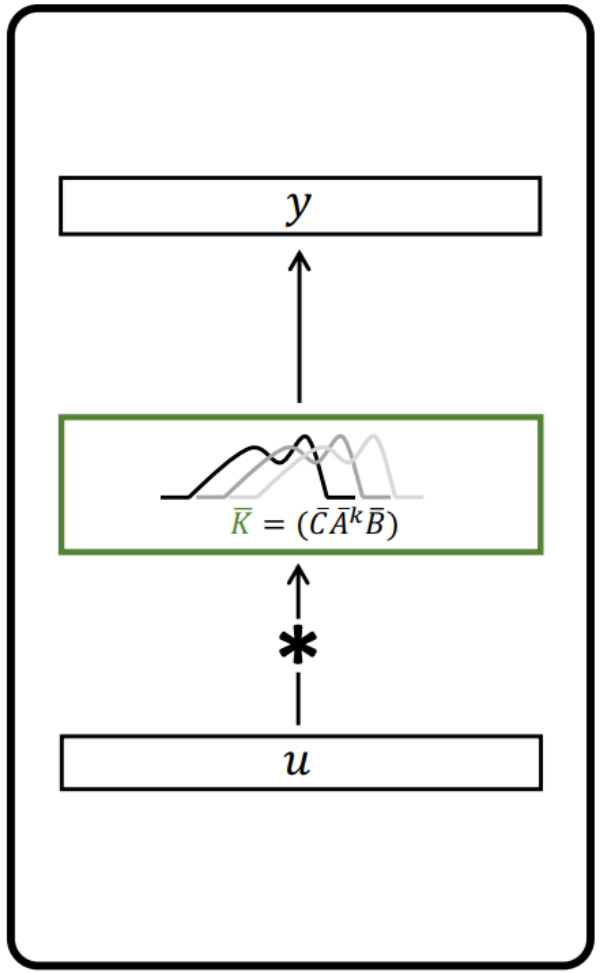
Discretize

Δt



Recurrent

or



Convolutional

Computing with SSMs: Convolutional View

$$x_k = \overline{A}x_{k-1} + \overline{B}u_k$$
$$y_k = \overline{C}x_k$$

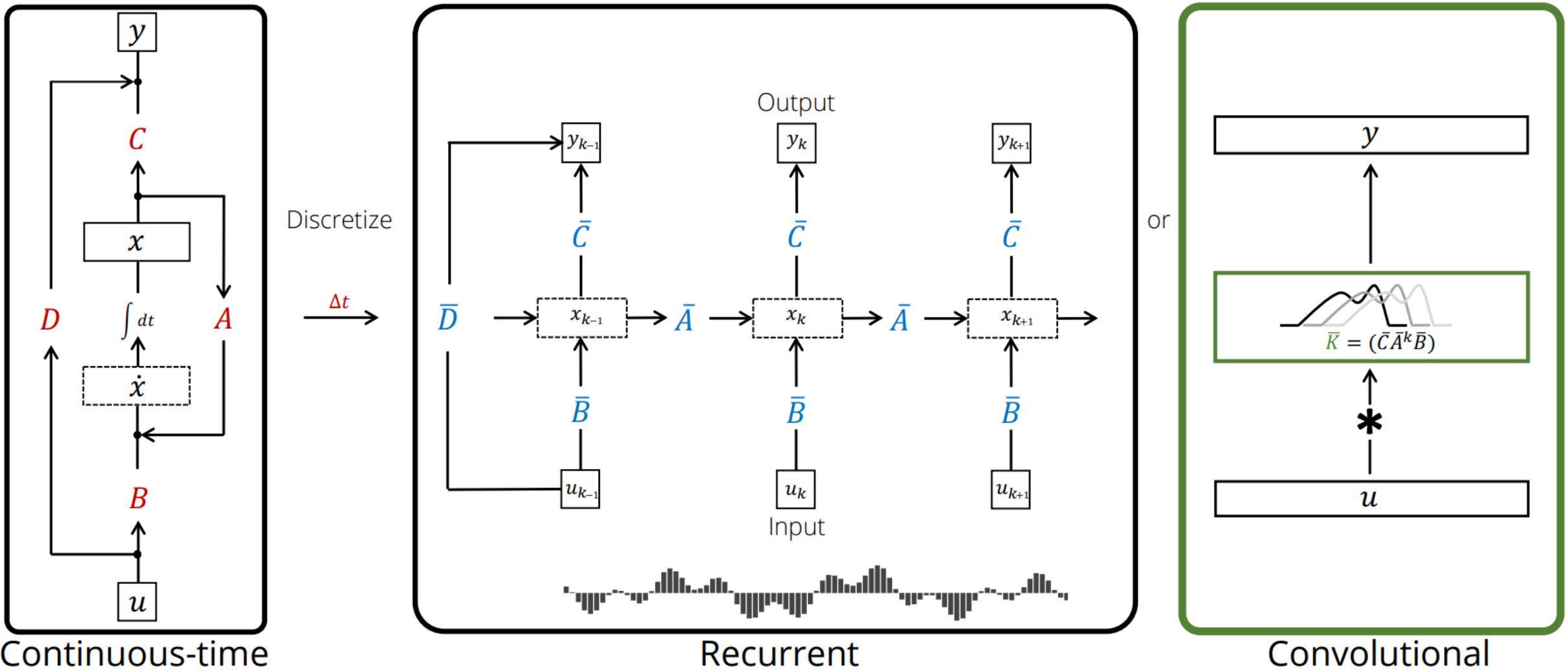
$$x_0 = \overline{B}u_0 \qquad x_1 = \overline{AB}u_0 + \overline{B}u_1 \qquad x_2 = \overline{A}^2\overline{B}u_0 + \overline{AB}u_1 + \overline{B}u_2 \qquad \dots$$
$$y_0 = \overline{CB}u_0 \qquad y_1 = \overline{CAB}u_0 + \overline{CB}u_1 \qquad y_2 = \overline{CA}^2\overline{B}u_0 + \overline{CAB}u_1 + \overline{CB}u_2 \qquad \dots$$

This can be vectorized into a **convolution** with an explicit formula for the **convolution kernel:**

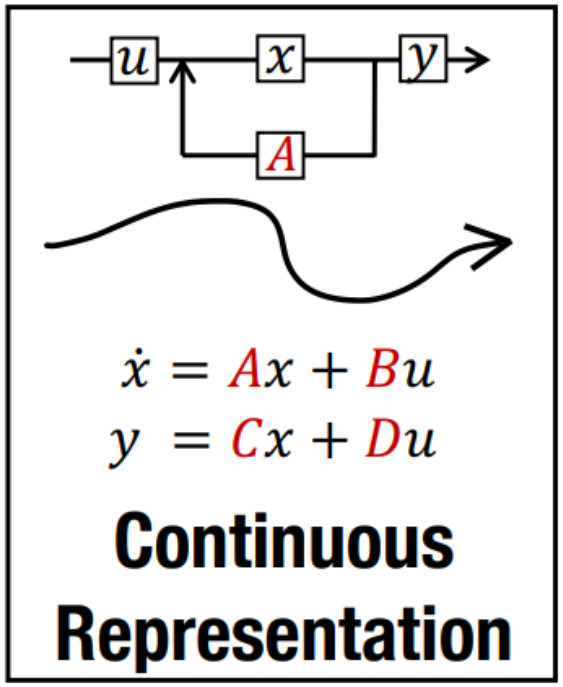
$$y_k = \overline{CA}^k\overline{B}u_0 + \overline{CA}^{k-1}\overline{B}u_1 + \cdots + \overline{CAB}u_{k-1} + \overline{CB}u_k$$
$$y = \overline{K} * u.$$

$$\overline{K} \in \mathbb{R}^L := \mathcal{K}_L(\overline{A}, \overline{B}, \overline{C}) := \left(\overline{CA}^i\overline{B}\right)_{i\in[L]} = (\overline{CB}, \overline{CAB}, \dots, \overline{CA}^{L-1}\overline{B}).$$

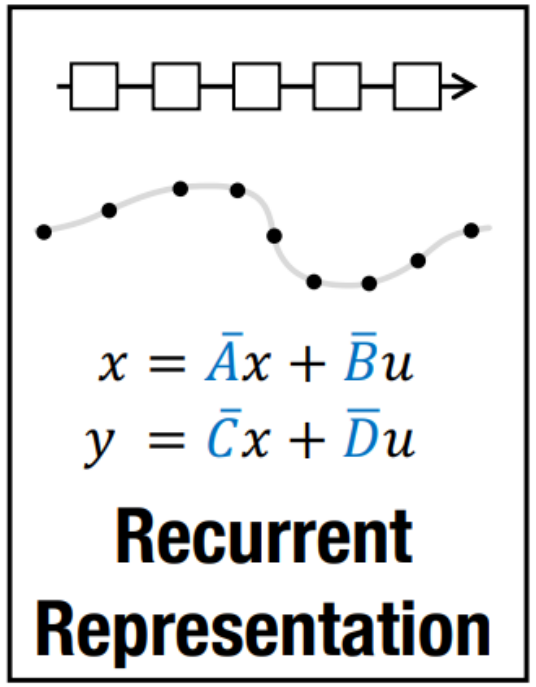
SSM: Convolutional Representation



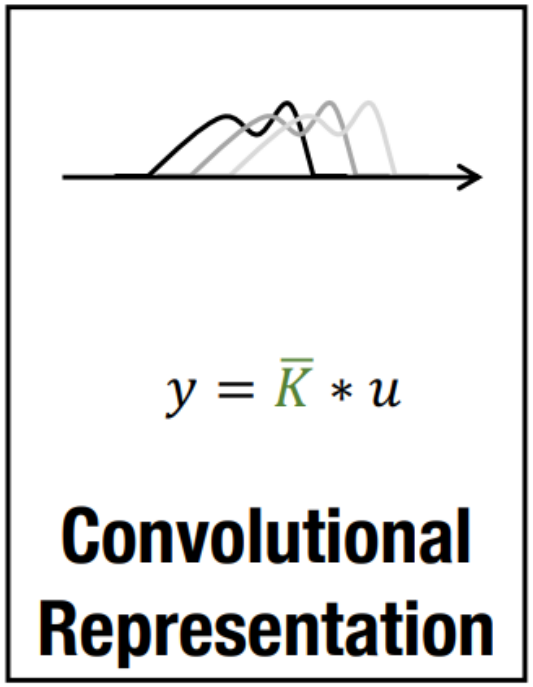
Summary: Properties of SSMs



Discretize



Unroll



- ✓ *mathematically tractable*
- ✓ *handles irregular data*

- ✓ *unbounded context*
- ✓ *efficient inference*

- ✓ *easy optimization*
- ✓ *parallelizable training*

Simple & principled... why haven't SSMs been used in deep learning?

- Introduction
- State space models (SSM) for deep sequence modeling
- Structured state spaces (S4) for long-term dependencies
- Experiments
- Conclusion

Challenges of Deep SSMs

- **Modeling Challenge**: SSMs inherit properties of CTMs, RNNs, CNNs... including problems with LRDs

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

with random matrix A: 50% on MNIST

$$\overline{\boldsymbol{K}} = (\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}})$$

Powering up $\overline{\boldsymbol{A}}$ → vanishing gradients?

Powering up $\overline{\boldsymbol{A}}$ → $O(N^2L)$ computation complexity

- **Computation Challenge**: SSMs have nice properties provided that representations $\overline{\boldsymbol{A}}$ and $\overline{\boldsymbol{K}}$ are known, but computing them is extremely hard!

Addressing Long-Range Dependencies with HiPPO Matrix

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \quad (1)$$

- HiPPO specifies a class of certain matrices $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ that when incorporated into (1), allows the state x(t) to memorize the history of the input u(t).

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

G.*, Dao*, Ermon, Rudra, Ré. "HiPPO: Recurrent Memory with Optimal Polynomial Projections." NeurIPS 2020
G., Johnson, Goel, Saab, Dao, Rudra, Ré. "Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers." NeurIPS 2021

Structured State Spaces (S4)

Computing $\overline{\boldsymbol{K}}$ is **non-trivial** and is the focus of technical contributions of this paper.

$$\overline{\boldsymbol{K}} \in \mathbb{R}^{L} := (\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}})$$

Algorithm 1 S4 Convolution Kernel (Sketch)

Input: S4 parameters $\boldsymbol{\Lambda}, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{C}^N$ and step size Δ
Output: SSM convolution kernel $\overline{\boldsymbol{K}} = \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}})$ for $\boldsymbol{A} = \boldsymbol{\Lambda} - \boldsymbol{p}\boldsymbol{q}^*$ (equation (5))

1: $\widetilde{\boldsymbol{C}} \leftarrow \left(\boldsymbol{I} - \overline{\boldsymbol{A}}^L\right)^* \overline{\boldsymbol{C}}$ ▷ Truncate SSM generating function (SSMGF) to length L

2: $\begin{bmatrix} k_{00}(\omega) & k_{01}(\omega) \\ k_{10}(\omega) & k_{11}(\omega) \end{bmatrix} \leftarrow \left[\widetilde{\boldsymbol{C}}\ \boldsymbol{q}\right]^* \left(\frac{2}{\Delta}\frac{1-\omega}{1+\omega} - \boldsymbol{\Lambda}\right)^{-1} [\boldsymbol{B}\ \boldsymbol{p}]$ ▷ Black-box Cauchy kernel

3: $\hat{\boldsymbol{K}}(\omega) \leftarrow \frac{2}{1+\omega}\left[k_{00}(\omega) - k_{01}(\omega)(1 + k_{11}(\omega))^{-1}k_{10}(\omega)\right]$ ▷ Woodbury Identity

4: $\hat{\boldsymbol{K}} = \{\hat{\boldsymbol{K}}(\omega) : \omega = \exp(2\pi i \frac{k}{L})\}$ ▷ Evaluate SSMGF at all roots of unity $\omega \in \Omega_L$

5: $\overline{\boldsymbol{K}} \leftarrow \mathsf{iFFT}(\hat{\boldsymbol{K}})$ ▷ Inverse Fourier Transform

Challenges of Deep SSMs

- **Modeling Challenge**: SSMs inherit properties of CTMs, RNNs, CNNs... including problems with LRDs

generic $\boldsymbol{A}$ → 50% on MNIST ✗

HiPPO $\boldsymbol{A}$ → 99% on MNIST ✓

- **Computation Challenge**: SSMs have nice properties provided that representations $\overline{\boldsymbol{A}}$ and $\overline{\boldsymbol{K}}$ are known, but computing them is extremely hard!

generic $\boldsymbol{A} \rightarrow O(N^2 L)$ computation ✗

HiPPO $\boldsymbol{A} \rightarrow \tilde{O}(N + L)$ computation ✓

- Introduction
- State space models (SSM) for deep sequence modeling
- Structured state spaces (S4) for long-term dependencies
- **Experiments**
- Conclusion

Long Range Arena

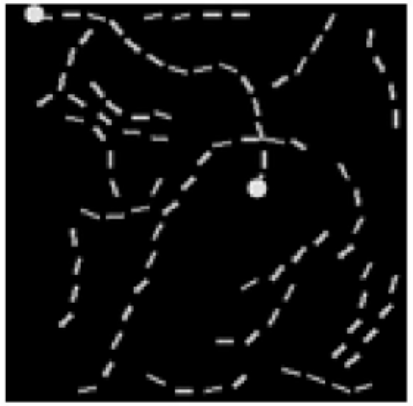
Benchmark spanning text, images, symbolic reasoning (length 1K-16K)

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Random	10.00	50.00	50.00	10.00	50.00	50.00	36.67
Transformer	36.37	64.27	57.46	42.44	71.40	✗	53.66
Local Attention	15.82	52.98	53.39	41.46	66.63	✗	46.71
Sparse Trans.	17.07	63.58	59.59	44.24	71.71	✗	51.03
Longformer	35.63	62.85	56.89	42.22	69.71	✗	52.88
Linformer	35.70	53.94	52.27	38.56	76.34	✗	51.14
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.56
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	✗	51.23
Synthesizer	36.99	61.68	54.67	41.61	69.45	✗	52.40
BigBird	36.05	64.02	59.29	40.83	74.87	✗	54.17
Linear Trans.	16.13	65.90	53.09	42.34	75.30	✗	50.46
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.18
FNet	35.33	65.11	59.61	38.67	77.80	✗	54.42
Nyströmformer	37.15	65.52	79.56	41.58	70.94	✗	57.46
Luna-256	37.25	64.57	79.29	47.38	77.72	✗	59.37
S4	**58.35**	**76.02**	**87.09**	**87.26**	**86.05**	**88.10**	**80.48**

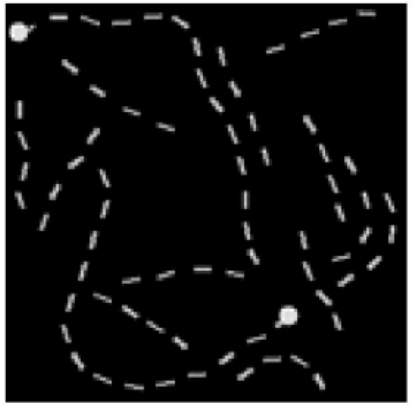
	Length 1024		Length 4096	
	Speed	Mem.	Speed	Mem.
Transformer	1×	1×	1×	1×
Performer	1.23×	0.43×	3.79×	0.086×
Linear Trans.	**1.58**×	**0.37**×	**5.35**×	**0.067**×
S4	**1.58**×	0.43×	5.19×	0.091×

Long Range Arena Path-X

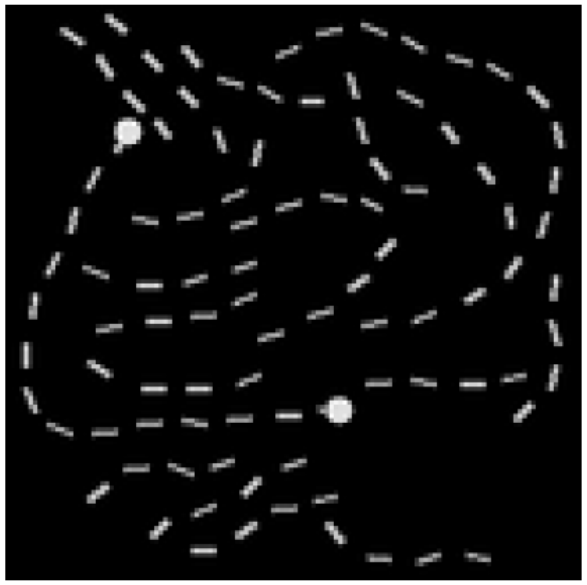
Path-X



(a) A positive example.



(b) A negative example.



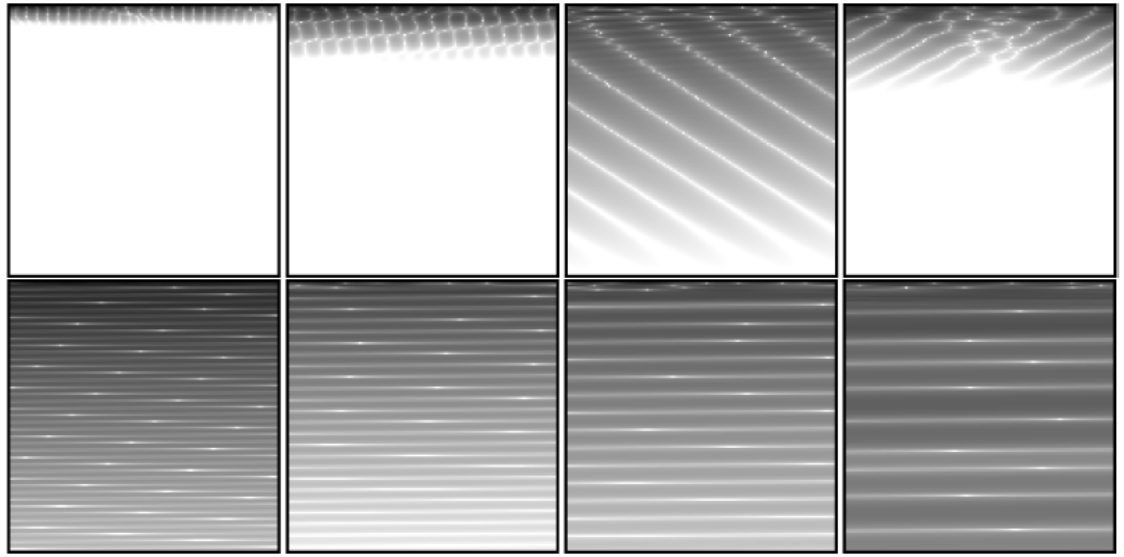


Figure 2: Visualizations of a trained S4 model on LRA Path-X. SSM convolution kernels $\overline{\boldsymbol{K}} \in \mathbb{R}^{16384}$ are reshaped into a 128×128 image. (*Left*) Example from the Path-X task, which involves deducing if the markers are connected by a path (*Top*) Filters from the first layer (*Bottom*) Filters from the last layer.

Sequential Image Classification

	Model	sMNIST	pMNIST	sCIFAR
Transformers	Transformer (Vaswani et al., 2017; Trinh et al., 2018)	98.9	97.9	62.2
CNNs	CKConv (Romero et al., 2021)	99.32	<u>98.54</u>	63.74
	TrellisNet (Bai et al., 2019)	99.20	98.13	73.42
	TCN (Bai et al., 2018)	99.0	97.2	-
RNNs	LSTM (Hochreiter & Schmidhuber, 1997; Gu et al., 2020b)	98.9	95.11	63.01
	r-LSTM (Trinh et al., 2018)	98.4	95.2	72.2
	Dilated GRU (Chang et al., 2017)	99.0	94.6	-
	Dilated RNN (Chang et al., 2017)	98.0	96.1	-
	IndRNN (Li et al., 2018)	99.0	96.0	-
	expRNN (Lezcano-Casado & Martínez-Rubio, 2019)	98.7	96.6	-
	UR-LSTM	99.28	96.96	71.00
	UR-GRU (Gu et al., 2020b)	99.27	96.51	<u>74.4</u>
	LMU (Voelker et al., 2019)	-	97.15	-
	HiPPO-RNN (Gu et al., 2020a)	98.9	98.3	61.1
	UNIcoRNN (Rusch & Mishra, 2021)	-	98.4	-
	LMUFFT (Chilkuri & Eliasmith, 2021)	-	98.49	-
	LipschitzRNN (Erichson et al., 2021)	<u>99.4</u>	96.3	64.2
SSMs	**S4**	**99.63**	**98.70**	**91.13**

Speech Classification(keywords spotting)

- Speech is difficult because LRD:

  1 second clip = length-16000

- Mel-frequency Cepstrum Coefficients (MFCC) (length 160)

	L=160 MFCC	L=16K Raw	L=8K 0.5×
Transformer	90.75	✗	✗
Performer	80.85	30.77	30.68
ODE-RNN	65.9	✗	✗
NRDE	89.8	16.49	15.12
ExpRNN	82.13	11.6	10.8
LipschitzRNN	88.38	✗	✗
CKConv	**95.3**	71.66	<u>65.96</u>
WaveGAN-D	✗	<u>96.25</u>	✗
LSSL	93.58	✗	✗
S4	<u>93.96</u>	**98.32**	**96.30**

- Introduction
- State space models (SSM) for deep sequence modeling
- Structured state spaces (S4) for long-term dependencies
- Experiments
- Conclusion

Conclusion

- This paper proposed a surprising sequence model S4, which can handle LRDs, and move between **continuous-time**, **convolutional**, and **recurrent** model representations, each with distinct capabilities.
- This approach enables SSMs to be applied successfully to a varied set of benchmarks with minimal modification.

Thank You for listening!

Shen Yuan

2023-7-19