An Overview of Deep State Space Models for Long Dependency Learning

Kangyun Chen

Johns Hopkins University | Whiting School of Engineering | Baltimore, MD

Introduction

Modeling long-range dependencies within sequential data presents a formidable challenge in deep learning. Traditional deep learning models, including RNNs, CNNs, and Transformers, offer valuable insights but face inherent limitations. To address this issue, recently **State Space Model (SSM)** was introduced.

This project investigates the promising capabilities of SSMs as a robust alternative. SSMs demonstrate a remarkable proficiency in managing long-range dependencies, overcoming the constraints observed in conventional models.

Objectives

Our goal is to provide readers with an insightful understanding of the significant role and impact of deep SSMs. This review will mainly focus on **three influential papers**. The first lays the foundation of SSMs, explaining their mathematical efficacy. The next two tackle computational challenges within SSMs, presenting solutions for enhanced efficiency and scalability in practical scenarios.

Materials and Methods

Three Representations of SSMs

The state space model maps a 1-D input signal u(t) to an N-D latent state x(t) before projecting to a 1-D output signal y(t).

1. Continuous-time Representation

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

2. Recurrent Discrete-time Representation

After discretizing continuous-time SSM, we can obtain a recurrent expression:

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

Two discretization methods:

$$(\textbf{Bilinear}) \quad \overline{\boldsymbol{A}} = (\boldsymbol{I} - \Delta/2\boldsymbol{A})^{-1}(\boldsymbol{I} + \Delta/2\boldsymbol{A})$$
$$\overline{\boldsymbol{B}} = (\boldsymbol{I} - \Delta/2\boldsymbol{A})^{-1} \cdot \Delta\boldsymbol{B}$$

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

3. Convolutional Discrete-time Representation

Unrolling the recurrent representation yields a convolutional formula

$$x_0 = \overline{\boldsymbol{B}}u_0 \qquad y_0 = \overline{\boldsymbol{C}\boldsymbol{B}}u_0$$
$$x_1 = \overline{\boldsymbol{A}\boldsymbol{B}}u_0 + \overline{\boldsymbol{B}}u_1 \qquad y_1 = \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}\boldsymbol{B}}u_1$$
$$x_2 = \overline{\boldsymbol{A}}^2\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{A}\boldsymbol{B}}u_1 + \overline{\boldsymbol{B}}u_2 \qquad y_2 = \overline{\boldsymbol{C}\boldsymbol{A}}^2\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}u_1 + \overline{\boldsymbol{C}\boldsymbol{B}}u_2$$
$$\vdots \qquad \vdots$$

Vectorizing y:

$$y_k = \overline{\boldsymbol{C}\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}\boldsymbol{A}}^{k-1}\overline{\boldsymbol{B}}u_1 + \cdots + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}\boldsymbol{B}}u_k$$
$$y = \overline{\boldsymbol{K}} * u.$$

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

Experiments

We reproduce the performance evaluation on the **1D pixel-level sequential image classification** on the sequential CIFAR-10 dataset.

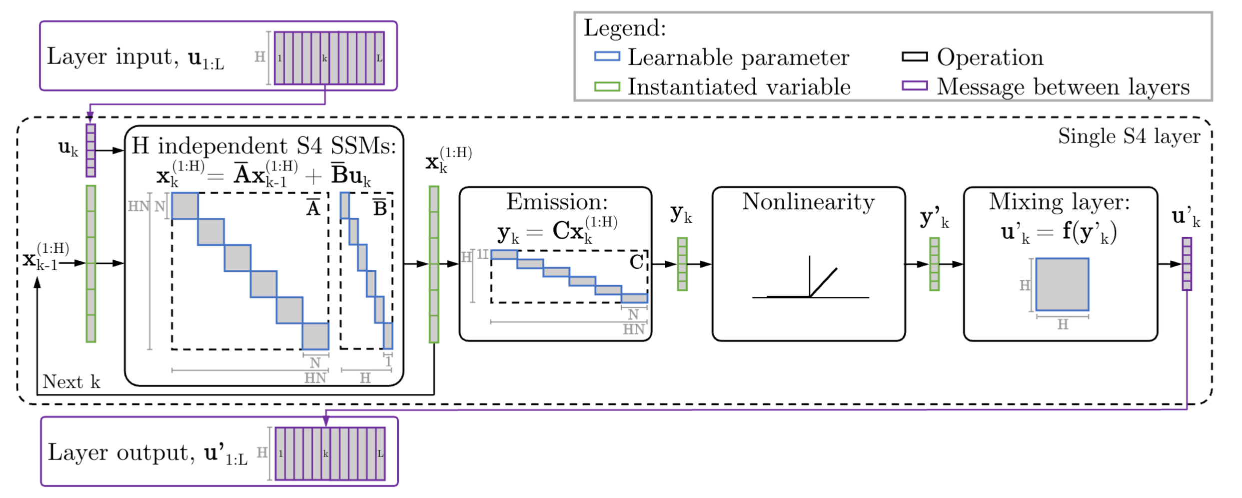
This task feeds image inputs of length 1024 into a model pixel-by-pixel, requiring learning long-term dependencies.

Table 1: Pixel-level image classification. The models in the top part are baselines from the literature.

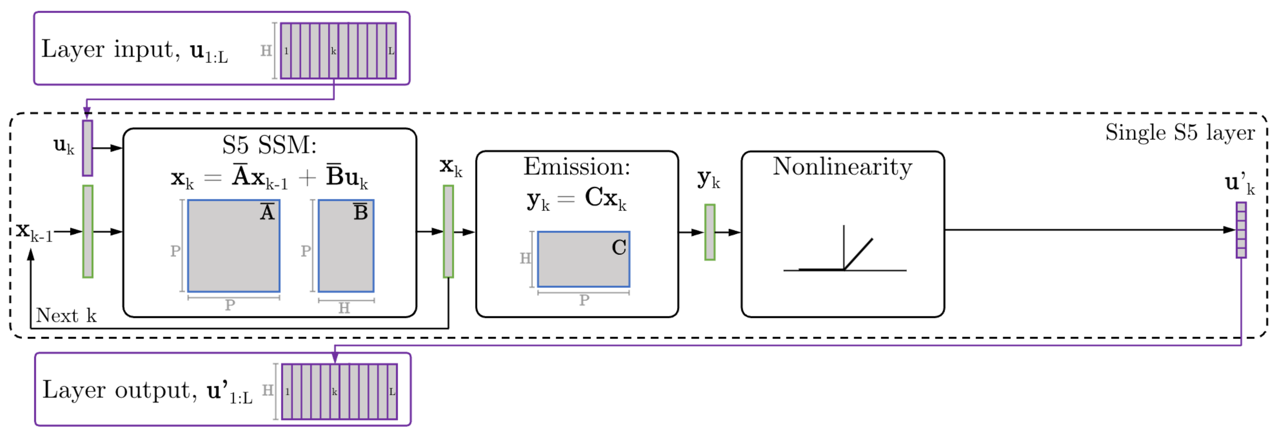
Model	Train Acc	Val Acc	Test Acc
CKConv	49.8	48.5	48.23
LSTM	57.91	53.82	53.23
Transformer 2.4027	99.78	61.79	61.55
HiPPO	94.69	53.8	53.52
S4	92.56	71.56	**72.57**
S4D	91.98	73.64	**73.75**

Results

How SSM is incorporated into Deep Neural Networks



(a) S4ND: H copies of Independent SSMs



(b) S5: One Multi-input Multi-output SSM

Figure 1: Two Different Approaches to Form a SSM Layer. H is the input feature. This figure is from S5 paper {arXiv: 2208.04933}.

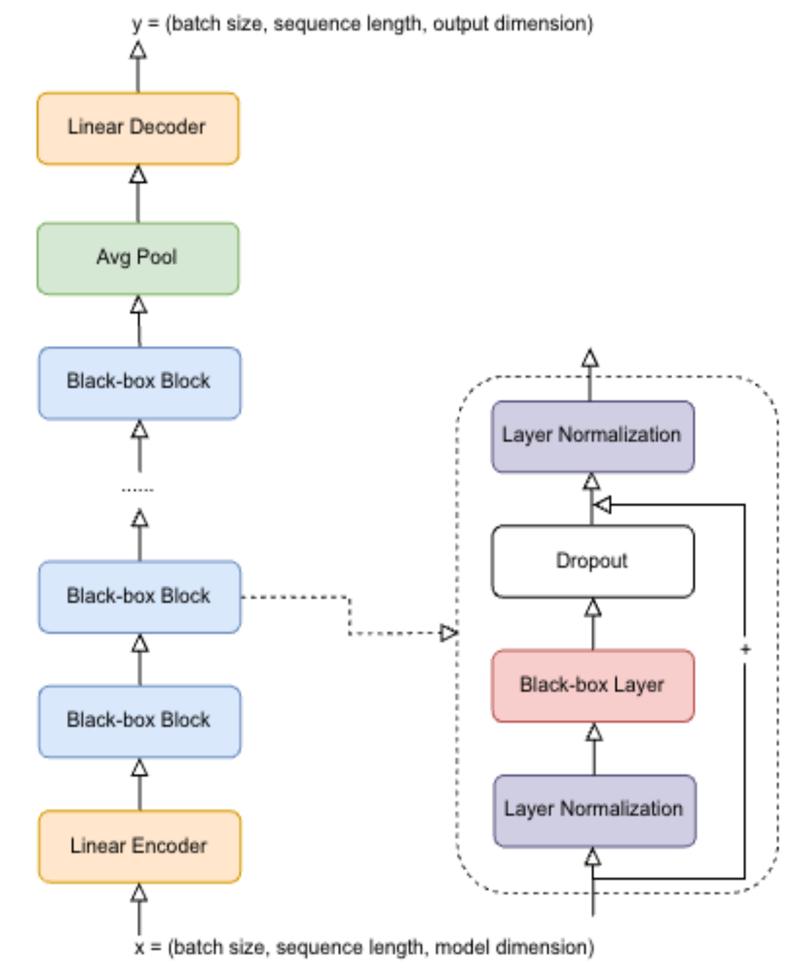


Figure 2: Backbone of Standard Sequence Model. Black-box layer can be SSM layer, RNN cell, attention and convolutions.

HiPPO {arXiv: 2008.07669}

The motivation of HiPPO is to learn from sequential data by creating a memory representation of cumulative history. This is achieved by projecting the data onto polynomial bases under some measure.

In other words, given a continuous function f(t), we aim to approximate f at every time t using a function g(t).

$$g^{(t)} := \mathrm{argmin}_{g\in\mathcal{G}}\|f_{\le t} - g\|_{\mu^{(t)}}, \qquad \text{and} \qquad g^{(t)} = \sum_{n=0}^{N-1} c_n(t) g_n^{(t)}$$

Under some specific measures and bases, the coefficients satisfy a linear ODE:

$$\frac{d}{dt}c(t) = A(t)c(t) + B(t)f(t)$$

This linear ODE motivates the subsequent SSMs. Also under these measures and bases, A and B have some special structure. These special structured matrices are called HiPPO matrices.

S4 {arXiv: 2111.00396}

S4 proposes one kind of special structured state matrix - Diagonal Plus Low-Rank (DPLR), $\boldsymbol{A} = \boldsymbol{\Lambda} - \boldsymbol{P}\boldsymbol{Q}^*$ which makes computing the convolutional kernel tractable.

- Compute generating function from convolutional kernel by FFT
- With DPLR, the generating function has a nice expression involving Cauchy matrices
- Compute convolutional kernel by iFFT

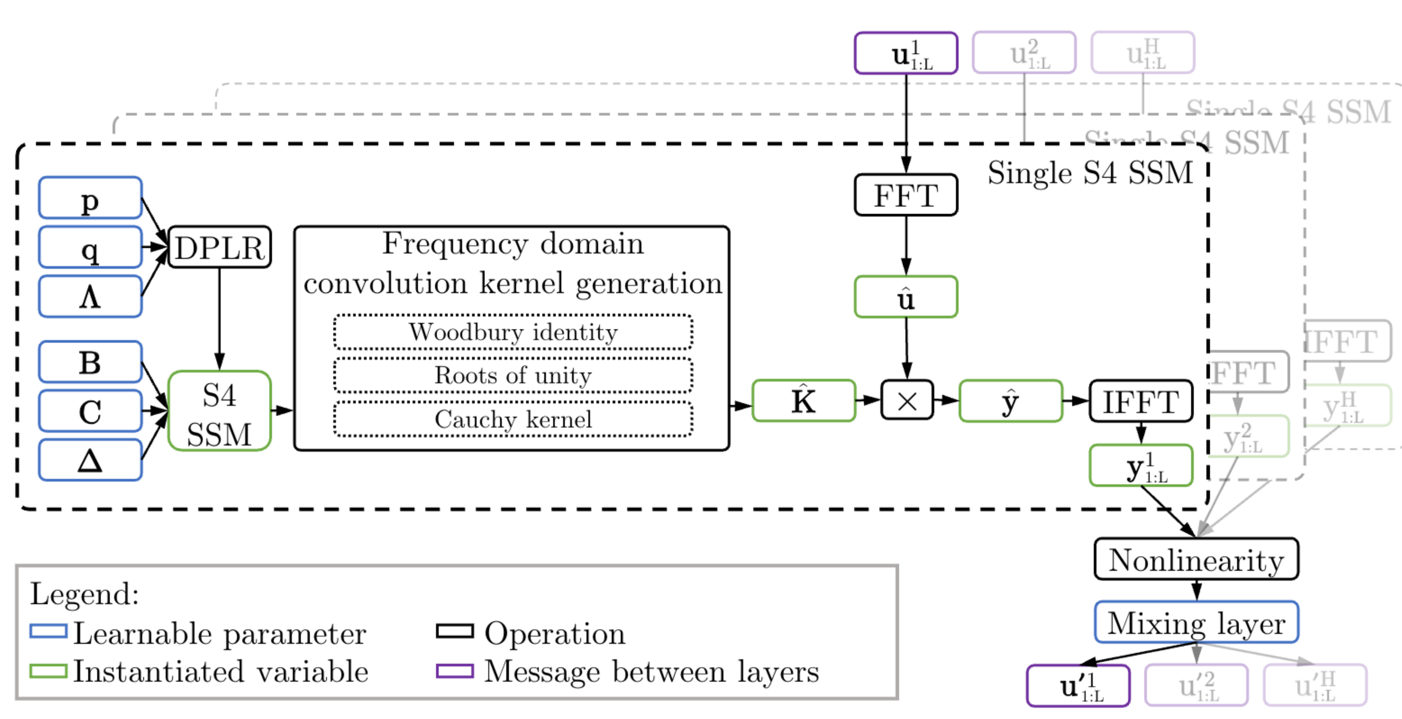


Figure 3: Processing of S4. This figure is from S5 paper {arXiv: 2208.04933}.

S4D {arXiv: 2206.11893}

This is a further improvement via simpler parameterization on the state matrix. With a purely diagonal matrix, the kernel can be computed trivially.

$$\overline{\boldsymbol{K}}_\ell = \sum_{n=0}^{N-1}\boldsymbol{C}_n\overline{\boldsymbol{A}}_n^\ell\overline{\boldsymbol{B}}_n \implies \overline{\boldsymbol{K}} = (\overline{\boldsymbol{C}\boldsymbol{B}}, \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}) = (\overline{\boldsymbol{B}}^\top \circ \boldsymbol{C})\cdot\mathcal{V}_L(\overline{\boldsymbol{A}})$$

$$\overline{\boldsymbol{K}} = \begin{bmatrix}\overline{\boldsymbol{B}}_0\boldsymbol{C}_0 & \ldots & \overline{\boldsymbol{B}}_{N-1}\boldsymbol{C}_{N-1}\end{bmatrix}\begin{bmatrix}1 & \overline{\boldsymbol{A}}_0 & \overline{\boldsymbol{A}}_0^2 & \ldots & \overline{\boldsymbol{A}}_0^{L-1}\\ 1 & \overline{\boldsymbol{A}}_1 & \overline{\boldsymbol{A}}_1^2 & \ldots & \overline{\boldsymbol{A}}_1^{L-1}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \overline{\boldsymbol{A}}_{N-1} & \overline{\boldsymbol{A}}_{N-1}^2 & \ldots & \overline{\boldsymbol{A}}_{N-1}^{L-1}\end{bmatrix}$$

Conclusion

We provide an in-depth exploration of Deep State Space Models and their pivotal role in addressing long-range dependency challenges in sequential data. We illustrate the methodology, provide further exposition and connections between these works, reconstruct the methods, and show them in examples.

References

1. Gu, A., Dao, T., Ermon, S., Rudra, A., & Ré, C. (2020). Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, *33*, 1474-1487.
2. Gu, A., Goel, K., & Ré, C. (2021). Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396.*
3. Gu, A., Goel, K., Gupta, A., & Ré, C. (2022). On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, *35*, 35971-35983.
4. Smith, J. T., Warrington, A., & Linderman, S. W. (2022). Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933.*