Efficiently Modeling Long Sequences with Structured State Spaces (S4) - Albert Gu, Karan Goel, Christopher Re (Stanford)

Mathieu Ravaut

Nanyang Technological University

Overview

- ICLR 2022 with impressive review scores : 8, 8, 8.
- An alternative way of sequence modeling without Transformer or even self-attention.
- Uses **state-space models**.
- Combined with *multiple linear algebra tricks* to keep computation cost low. There'll be *a lot* of maths !
- Extremely good at long sequence modeling : SOTA in Long Range Arena with much faster inference.
- *The Annotated S4*, a dedicated illustrated notebook by Sasha Rush (March 2022).

Table of content

State-space models (SSMs)

- State-space models are a classical mathematical tool from the 1970s and 1980s, used for instance in control engineering.
- Framework :
  - A 1-D input signal $u(t)$
  - An N-D latent representation $x(t)$
  - A 1-D output signal $y(t)$
- The state-space model is defined by :

$$x'(t) = \mathbf{A}.x(t) + \mathbf{B}.u(t) \tag{1}$$
$$y(t) = \mathbf{C}.x(t) + \mathbf{D}.u(t) \tag{2}$$

State-space models (SSMs)

- In practice, they ignore **D** because it's simply a skip-connection and set the matrix to 0, so the SSM equation becomes :

$$x'(t) = \mathbf{A}.x(t) + \mathbf{B}.u(t) \tag{3}$$
$$y(t) = \mathbf{C}.x(t) \tag{4}$$

Hippo Matrix

- The basic SSM is a linear first-order ODE.
- It solves with an exponential function.
- Thus, it performs poorly in practice, due to **vanishing/exploding gradient** issues.
- The same authors propose to use the **HiPPO Matrix** as matrix A, from their previous paper *HiPPO : Recurrent memory with optimal polynomial projections* (Neurips 2020).
- This special class of matrices allows the state $x(t)$ to memorize the input $u(t)$ (hard to get the intuition..).

HiPPO Matrix

The HiPPO matrix :

$$\mathbf{A}_{n,k} = -\begin{cases} (2n+1)^{1/2}.(2k+1)^{1/2} \text{ if } n > k \\ (n+1) \text{ if } n = k \\ 0 \text{ if } n < k \end{cases} \tag{5}$$

$$\mathbf{A} = -\begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ \sqrt{3} & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{2N-1} & \sqrt{(2N-1).3} & \dots & \sqrt{(2N-1)(2N-3)} & N \end{bmatrix} \tag{6}$$

Discrete SSM

- So far everything was continuous.
- But in practice, the input signal is discrete $u = (u_0, u_1, \dots)$.
- The *bilinear method* enables to derive a **discrete SSM** :

$$x_k = \overline{\mathbf{A}}.x_{k-1} + \overline{\mathbf{B}}.u_k \tag{7}$$

$$y_k = \overline{\mathbf{C}}.x_k = \mathbf{C}.x_k \tag{8}$$

with the following new matrices (Δ is the step size) :

$$\overline{\mathbf{A}} = (\mathbf{I} - \frac{\Delta}{2}.\mathbf{A})^{-1}.(\mathbf{I} + \frac{\Delta}{2}.\mathbf{A}) \tag{9}$$

$$\overline{\mathbf{B}} = (\mathbf{I} - \frac{\Delta}{2}.\mathbf{A})^{-1}.\Delta.\mathbf{B} \tag{10}$$

- The discrete SSM converts the problem to a *sequence-to-sequence* mapping, useful for NLP.

Structured state-spaces

- The previous recurrent equation is not practical on modern hardware due to sequentiality.
- Instead, we write it as a **convolution**.
- If we enroll the discrete SSM :

$$x_k = \overline{\mathbf{A}^k.\mathbf{B}}.u_0 + \overline{\mathbf{A}^{k-1}.\mathbf{B}}.u_1 + \cdots + \overline{\mathbf{B}}.u_k \quad (11)$$

$$y_k = \overline{\mathbf{C}.\mathbf{A}^k.\mathbf{B}}.u_0 + \overline{\mathbf{C}.\mathbf{A}^{k-1}.\mathbf{B}}.u_1 + \cdots + \overline{\mathbf{C}.\mathbf{B}}.u_k \quad (12)$$

$$(13)$$

- As a convolution :

$$y = \overline{\mathbf{K}} * u$$

  with the kernel

$$\overline{\mathbf{K}} = (\overline{\mathbf{C}.\mathbf{B}}, \overline{\mathbf{C}.\mathbf{A}.\mathbf{B}}, \ldots, \overline{\mathbf{C}.\mathbf{A}^{L-1}.\mathbf{B}}) = (\overline{\mathbf{C}.\mathbf{A}^i.\mathbf{B}})_i$$

Structured state-spaces

- This was just the pre-requisite to the paper !
- We reduced the problem to a single convolution.
- It can be computed very efficiently using **Fast Fourier Transform** (FFT).
- ***The main contribution of this paper is computing the SSM convolution kernel $\overline{\mathbf{K}}$ fast.***

Kernel computation

- Computing the kernel $\overline{\mathbf{K}}$ requires raising the matrix $\overline{\mathbf{A}}$ to several powers.
- That is fine if we diagonalize $\overline{\mathbf{A}}$, because conjugation is an equivalence relation in state-space models :

$$(\mathbf{A}, \mathbf{B}, \mathbf{C}) \sim (\mathbf{V}^{-1}.\mathbf{A}.\mathbf{V}, \mathbf{V}^{-1}.\mathbf{B}, \mathbf{C}.\mathbf{V}) \tag{14}$$

- The problem is that there exists a matrix $\mathbf{V}$ diagonalizing the HiPPO matrix $\overline{\mathbf{A}}$, but it has entries exponentially large in the state space N ! Not stable numerically.

Diagonal plus low-rank

Instead of directly diagonalizing $\overline{\mathbf{A}}$, we decompose it into a **normal plus low-rank** matrix (linear algebra property) :

$$\mathbf{A} = \mathbf{V}.\mathbf{D}.\mathbf{V}^* - \mathbf{P}.\mathbf{Q}^{\mathbf{T}} \tag{15}$$

which means conjugating to a ***diagonal plus low-rank*** (DPLR) :

$$\mathbf{A} = \mathbf{V}.(\mathbf{D}. - (\mathbf{V}^*\mathbf{P}).(\mathbf{V}^*.\mathbf{Q})^*).\mathbf{V}^* \tag{16}$$

The above is **Theorem 1** of the paper.

S4 Recurrence

Remember that we care about $\overline{\mathbf{A}}$, not $\mathbf{A}$:

$$\overline{\mathbf{A}} = (\mathbf{I} - \frac{\Delta}{2}.\mathbf{A})^{-1}.(\mathbf{I} + \frac{\Delta}{2}.\mathbf{A}) \tag{17}$$

- Since $\mathbf{A}$ is DPLR, so is the second term (obvious).
- The first term is also DPLR because of **Woodbury identity**.
- Thus, we obtain $\overline{\mathbf{A}}$ from $\mathbf{A}$ in $O(N)$.
- In other words, one recurrent step of the discrete SSM has complexity $O(N)$ (**Theorem 2**).

S4 Convolution

- Instead of computing K directly, we compute its spectrum by evaluating its truncated generating function $\Sigma_{j=0}^{L-1}\overline{\mathbf{K}_j}\zeta^j$ at the roots of unity ζ^j.
- $\overline{\mathbf{K}}$ is then retrieved with an inverse FFT.
- We can replace the matrix powers in the kernel with just an inverse :

$$\overline{\mathbf{K}}(z) = \Sigma_{i=1}^{L-1}\overline{\mathbf{C}.\mathbf{A}^i.\mathbf{B}}.z^i = \cdots = \tilde{\mathbf{C}}.(\mathbf{I} - \overline{\mathbf{A}}.z)^{-1}.\overline{\mathbf{B}} \quad (18)$$

- However, this inverse needs to be calculated L times (once for each root of the unity).

S4 Convolution

- Evaluating the kernel $\overline{\mathbf{A}}$ to the roots of the unity, because of the special structure of the matrix A that we explored before, ends up becoming a **Cauchy kernel**.
- A Cauchy kernel can be evaluated efficiently in $O(N+L).\log^2(N+L)$ operations (while it would be $O(N.L)$ naively).
- The S4 convolutional kernel involves 4 Cauchy kernels, requiring $O(N+L)$ operations and $O(N+L)$ space (**Theorem 3**).

S4 Layer

- One S4 layer has the following learnable parameters : **D**, **P**, **Q**, **B** and **C**.
- An S4 layer maps a $\mathbb{R}^L$ signal to another $\mathbb{R}^L$ signal.
- We make it process H features instead of 1 :
  - Define H independant copies of the S4 layer.
  - Mix the features with a position-wise linear layer.
  - Add non-linear activations between these layers.
- After this process, S4 defines a sequence-to-sequence map of shape (batch size, sequence length, hidden dimension), just like Transformer.

Algorithm

Overview of S4 algorithm to compute the convolutional kernel :

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

Complexity

L is sequence length, B is the batch size, H is the feature size.

	Convolution[3]	Recurrence	Attention	S4
Parameters	LH	$\boldsymbol{H^2}$	$\boldsymbol{H^2}$	$\boldsymbol{H^2}$
Training	$\boldsymbol{\tilde{L}H(B+H)}$	BLH^2	$B(L^2H+LH^2)$	$\boldsymbol{BH(\tilde{H}+\tilde{L})+B\tilde{L}H}$
Space	$\boldsymbol{BLH}$	$\boldsymbol{BLH}$	$B(L^2+HL)$	$\boldsymbol{BLH}$
Parallel	**Yes**	No	**Yes**	**Yes**
Inference	LH^2	$\boldsymbol{H^2}$	L^2H+H^2L	$\boldsymbol{H^2}$

S4 is optimal or near-optimal on all aspects.

Long-Range Arena

LRA is a benchmark of several tasks with very long-range dependencies :

MODEL	LISTOPS	TEXT	RETRIEVAL	IMAGE	PATHFINDER	PATH-X	AVG
Transformer	36.37	64.27	57.46	42.44	71.40	✗	53.66
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.56
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**

S4 is SOTA by a very large margin, and is the first model ever to beat random chance on Path-X (Path-X is a Pathfinder task, aka identifying the correct target contour of a shape, but with very high resolution (16k) images).

Speech classification

Speech classification accuracy with standard MFCC features, unprocessed signals (Raw), and frequency change at test time (x0.5).

	MFCC	Raw	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	65.96
WaveGAN-D	✗	96.25	✗
LSSL	93.58	✗	✗
S4	93.96	**98.32**	**96.30**

Image classification

Pixel-level sequential image classification, sliddding non-overlapping windows of 1024 pixels.

	sMNIST	pMNIST	sCIFAR
Transformer	98.9	97.9	62.2
LSTM	98.9	95.11	63.01
r-LSTM	98.4	95.2	72.2
UR-LSTM	99.28	96.96	71.00
UR-GRU	99.27	96.51	74.4
HiPPO-RNN	98.9	98.3	61.1
LMU-FFT	-	98.49	-
LipschitzRNN	99.4	96.3	64.2
TCN	99.0	97.2	-
TrellisNet	99.20	98.13	73.42
CKConv	99.32	98.54	63.74
LSSL	99.53	**98.76**	84.65
S4	**99.63**	98.70	**91.13**

S4 is better than Transformer, LSTMs, and CNNs.

Image density estimation

Density estimation on CIFAR-10 :

Model	bpd	2D bias	Images / sec
Transformer	3.47	**None**	0.32 (1×)
Linear Transf.	3.40	**None**	17.85 (56×)
PixelCNN	3.14	2D conv.	-
Row PixelRNN	3.00	2D BiLSTM	-
PixelCNN++	2.92	2D conv.	19.19 (59.97×)
Image Transf.	2.90	2D local attn.	0.54 (1.7×)
PixelSNAIL	2.85	2D conv. + attn.	0.13 (0.4×)
Sparse Transf.	**2.80**	2D sparse attn.	-
S4 (base)	2.92	**None**	**20.84 (65.1×)**
S4 (large)	2.85	**None**	3.36 (10.5×)

S4 is competitive with SOTA in terms of bits per dimension.
S4-base is the fastest overall.

Language modeling

Word-level language modeling (perplexity) on WikiText-103 :

Model	Params	Test ppl.	Tokens / sec
Transformer	247M	**20.51**	0.8K (1×)
GLU CNN	229M	37.2	-
AWD-QRNN	151M	33.0	-
LSTM + Hebb.	-	29.2	-
TrellisNet	180M	29.19	-
Dynamic Conv.	255M	25.0	-
TaLK Conv.	240M	23.3	-
S4	249M	**21.28**	**48K** (**60**×)

A Transformer with attention blocks replaced by S4 blocks provides the best perplexity among attention-free models, very close to Transformer with self-attention.

Besides S4 processes 60x more tokens per second.

Conclusion

- A fresh departure from the classical Transformer layers and self-attention.
- Very high performing on long sequence modeling.
- Also computationnally efficient, as it's not limited by the $O(L^2)$ complexity of self-attention.
- A method derived from an old tool (state-space models), rendered useful through applying multiple linear algebra tricks.
- I am currently working with an intern at A*STAR on how to extend S4 to an encoder-decoder model, with the goal of applying it to long-input summarization (e.g, Arxiv, PubMed), perhaps document-level translation too.