Models with long memory

Alexey Zaytsev

Some slides are from Zabolotnyi Artem

Benchmarking for Efficient Transformers for transactions data

Skoltech

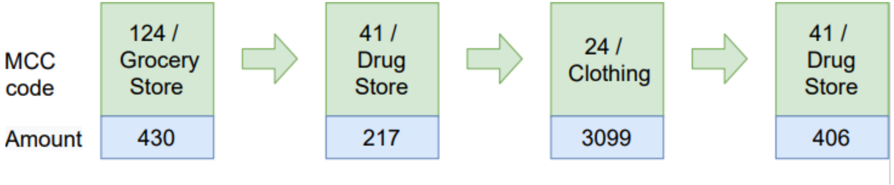
Dataset description
Transaction dataset

Task
Predict the probability of user default

Features
~10 categorical features (MCC code, type etc)
~10 numerical (amount, time etc)

Train test split
Train – 1 million users
Validation – 250k users
Out-of-time – 500k users

Length
700 - mean transaction count
(*Typical transformer length 512)*
Metric
Gini coefficient



3

Hyperparameters search for Longformer

Quality

Longformer model shows better quality than baseline LSTM

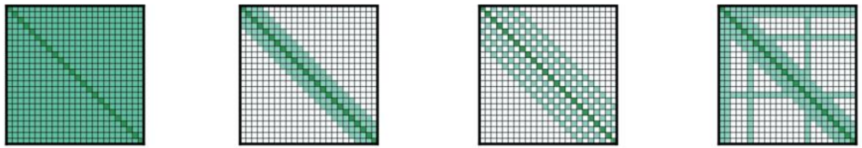
Hyperparameters

- Very deep model required a huge amount of memory for training
- Sliding window size W does not change quality significant

Baseline metric

LSTM - 60.12 ± 0.17

№	Heads	Embedding size	W	Layers	Hidden dropout	OOT Gini	RAM (Mb)
1	8	512	128	1	0.1	59.59	**7647**
2	8	256	128	1	0.1	58.84	4929
3	8	64	128	1	0.1	60.11	4581
4	8	64	128	2	0.1	61.02	6039
5	8	64	256	2	0.1	61.03	10131
6	8	64	64	2	0.1	60.58	4500
7	4	64	128	8	0.3	62.21	13971
8	4	32	128	8	0.3	62.58	13505
9	4	64	128	8	0.5	62.39	13971
10	4	64	128	12	0.3	62.39	20255
11	2	64	128	8	0.5	62.47	12891
12	4	64	128	12	0.6	**62.83**	20255



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Skoltech

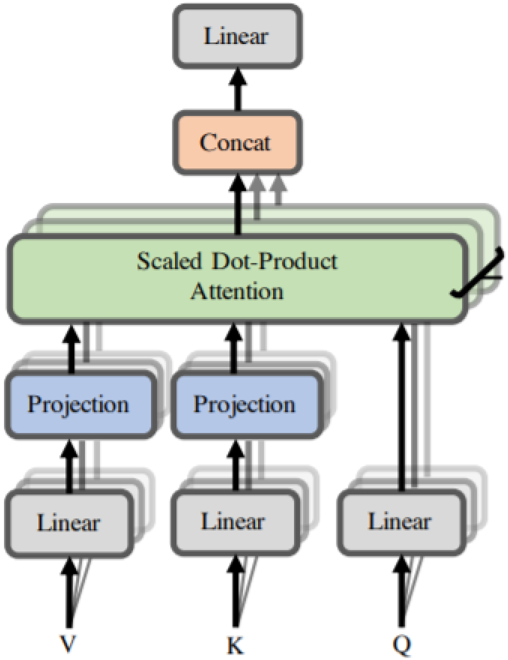
4

Results of the experiment
Matrix projection Linformer

Quality
Linformer with projection matrix shows poor results in comparing with recurrent neural network models

Shift invariance of a projection matrix
The projection matrix not being shifting invariant

Shift invariance test
We train the model to prove this property, choose a subsample, and calculate the metric on it. Then we shift all transactions by one and calculate the metric again.



Skoltech

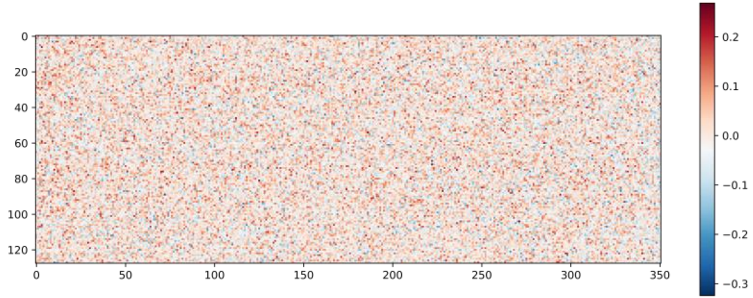
5

Results of the experiment
Matrix projection Linformer

Quality
Linformer with projection matrix shows poor results in comparing with recurrent neural network models

Shift invariance of a projection matrix
The projection matrix not being shifting invariant

Shift invariance test
We train the model to prove this property, choose a subsample, and calculate the metric on it. Then we shift all transactions by one and calculate the metric again.



Linformer projection matrix

Dataset	Gini
Original dataset	53.47
Shift by one transaction	47.31

Skoltech

6

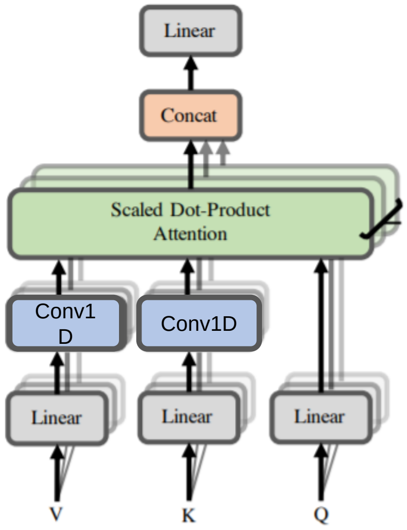
Results of the experiment

Convolution Linformer

Quality
Convolution Linformer shows a better result using less memory and shorter training time.

Convolution modification
To make projection shift-invariant, we replace a projection matrix with a convolution layer.

Cycle-shift augmentation
The result of convolution with kernel and stride equals size could change after adding a new object in sequence. (+1 Gini)



Skoltech

Wang, Sinong, et al. "Linformer: Self-attention with linear complexity." *arXiv preprint arXiv:2006.04768* (2020).

7

Results of the experiment

Convolution Linformer

Metric
LSTM - 60.12±0.17
Longformer - 62.80

Quality
Convolution Linformer shows a better result using less memory and shorter training time.

Convolution modification
To make projection shift-invariant, we replace a projection matrix with a convolution layer.

Cycle-shift augmentation
The result of convolution with kernel and stride equals size could change after adding a new object in sequence. (+1 Gini)

№	Projection size	Heads	Embedding size	Layers	Hidden dropout	dim ff	OOT gini	RAM (Mb)
1	128	8	64	4	0.2	2048	63.5	8757
2	32	8	64	4	0.2	2048	62.02	5124
3	64	8	64	4	0.2	2048	62.40	6789
4	128	4	64	4	0.2	2048	62.73	7159
5	128	8	64	2	0.2	2048	64.04	5333
6	128	8	64	4	0.2	2048	62.49	8715
7	128	8	32	4	0.2	2048	64.17	8179
8	128	8	64	4	0.2	2048	62.98	3635
9	128	8	64	3	0.2	2048	63.53	7051
10	128	8	64	2	0.2	512	**64.3 ± 0.19**	5333
11	128	8	64	2	0.2	256	63.5	**3395**
12	128	8	128	2	0.2	512	63.26	3833

Skoltech

8

Results of the experiment

Convolution Linformer

Curriculum learning: Pre-trained weights
Training first on short sequences and the fine-tune on long show better quality and reduce training time

Non linear complexity
We need to increase the projection size: increasing length of sequence without increasing projection size leads to quality degradation.

№	Length	K	Training type	OOT gini	RAM (Mb)	Train time (hours)
1	350	128	From scratch	64.37	9787	3
2	1500	128	From scratch	63.75	12451	4.5
3	1500	256	From scratch	64.81	16047	11.61
4	1500	256	Pre-trained weights	**65.2**	16081	9.15
5	1500	400	From scratch	65.39	23481	17.3
6	1500	400	Pre-trained/train only conv layer	64.05	2299	7.1
7	1500	400	Pre-trained weights	**66.00 ± 0.21**	23489	8.5

Skoltech

9

Results of the experiment
Comparison LSTM and Transformer

Transaction count	LSTM	Longformer	Linformer	Convolution Linformer
350	60.12 ± 0.17	62.80	53.47	64.37 ± 0.19
1500	63.01 ± 0.14	None	None	66.00 ± 0.21

Skoltech

10

Performer: making transformers faster

Skoltech

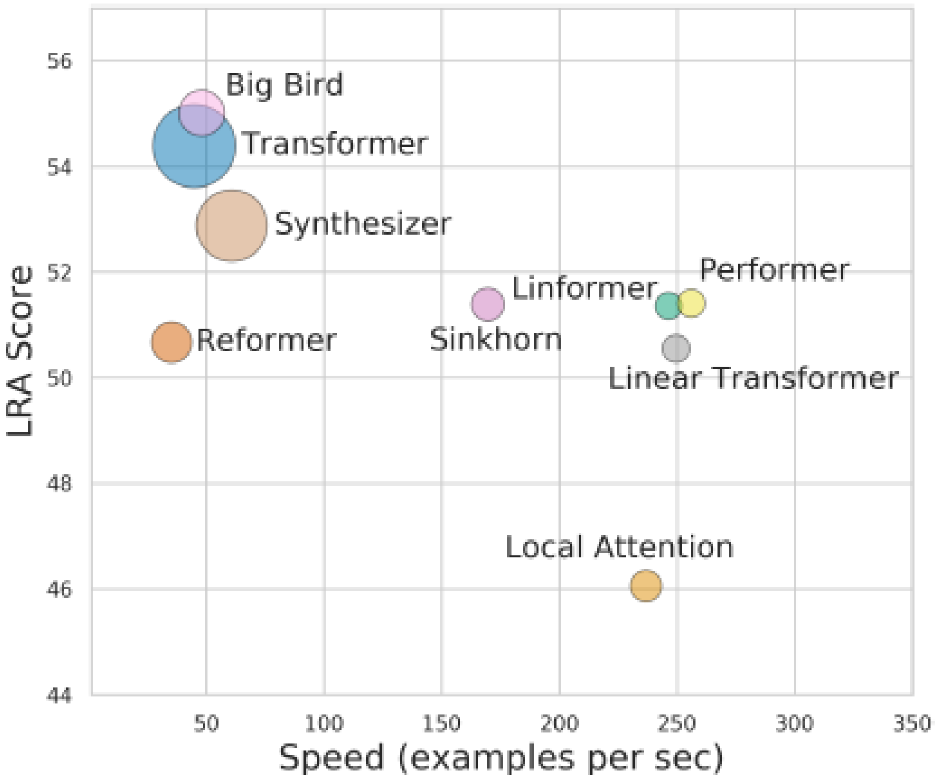
Tay, Yi, et al. "Long range arena: A benchmark for efficient transformers." *arXiv preprint arXiv:2011.04006* (2020).

Overall results

Speed (x axis), Performance (y axis) and Memory (size of the circles) for different models

BigBird is better than the vanilla transformer

Performer is more efficient



Skoltech

Performer

$$\exp(q_i^T k_j) = \phi(q_i)^T \phi(k_j) = \mathrm{E}(\phi(q_i)^T \phi(k_j))$$

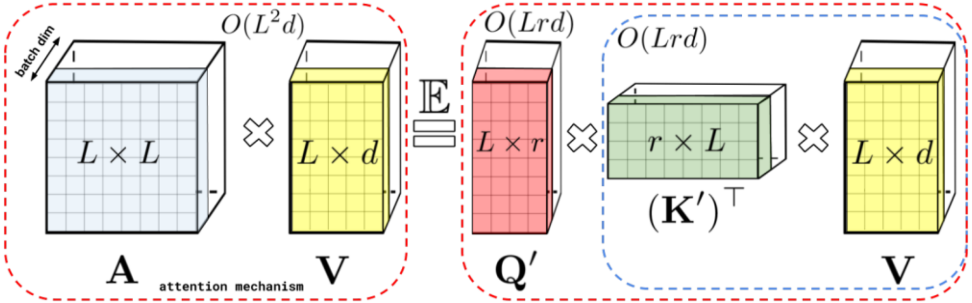
Vanilla attention formula:

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V}, \quad \mathbf{A} = \exp(\mathbf{Q}\mathbf{K}^{\top}/\sqrt{d}), \quad \mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_L).$$

Efficient bidirectional attention:

$$\widehat{\mathrm{Att}_{\leftrightarrow}}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \widehat{\mathbf{D}}^{-1}(\mathbf{Q}'((\mathbf{K}')^{\top}\mathbf{V})), \qquad \widehat{\mathbf{D}} = \mathrm{diag}(\mathbf{Q}'((\mathbf{K}')^{\top}\mathbf{1}_L)).$$

It uses low-rank approximation to query and key matrices: $Q' \in \mathrm{R}^{T\times r}$, $K' \in \mathrm{R}^{r\times T}$



Choromanski, Krzysztof Marcin, et al. *Rethinking Attention with Performers.* ICLR. 2020.

Exponential kernel decomposition

Kernel representation

$$\exp\left(q_i^T k_j\right) = \phi(q_i)^T \phi\left(k_j\right) = \mathrm{E}\left(\phi(q_i)^T \phi\left(k_j\right)\right)$$

Feature vector

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}\left(f_1(\omega_1^\top \mathbf{x}), ..., f_1(\omega_m^\top \mathbf{x}), ..., f_l(\omega_1^\top \mathbf{x}), ..., f_l(\omega_m^\top \mathbf{x})\right)$$

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^d,\ \mathbf{z} = \mathbf{x} + \mathbf{y}$$

$$\Lambda = \exp\left(-\frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2}{2}\right)$$

Our kernel

$$\mathrm{SM}(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\|\mathbf{x}\|^2}{2}\right) \mathrm{K}_{\mathrm{gauss}}(\mathbf{x}, \mathbf{y}) \exp\left(\frac{\|\mathbf{y}\|^2}{2}\right)$$

$$\mathrm{SM}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{\omega \sim \mathcal{N}(0, \mathbf{I}_d)}\left[\exp\left(\omega^\top \mathbf{x} - \frac{\|\mathbf{x}\|^2}{2}\right) \exp\left(\omega^\top \mathbf{y} - \frac{\|\mathbf{y}\|^2}{2}\right)\right] = \Lambda \mathbb{E}_{\omega \sim \mathcal{N}(0, \mathbf{I}_d)} \cosh(\omega^\top \mathbf{z})$$

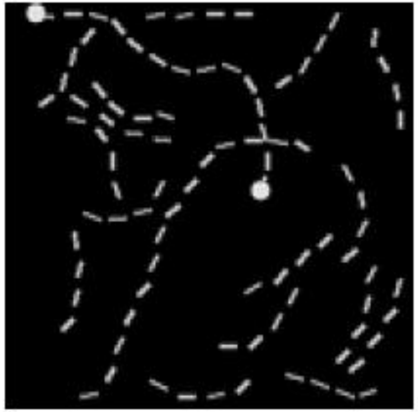
Skoltech

Choromanski, K.M., et al. *Rethinking Attention with Performers.* ICLR. 2020.

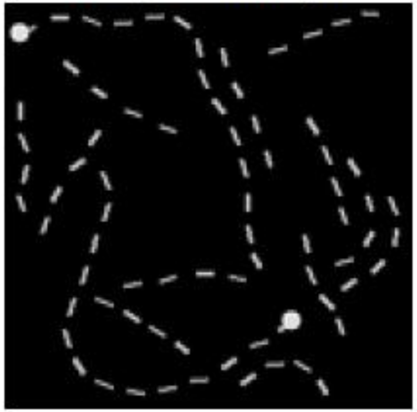
Pathfinder problem

For an image reshaped to the image find out, if two points are connected by a dashed line or not.

- Pathfinder: image 32x32 -> sequence 1024
- Pathfinder-X: images128x128 -> sequence 16K, extreme length (c.t. 1K limit for GPT-3)



Positive example



Negative example

Tay, Yi, et al. "Long range arena: A benchmark for efficient transformers." *arXiv preprint arXiv:2011.04006* (2020).

All transformers fail for Path-X problem

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	FAIL	54.39
Local Attention	15.82	52.98	53.39	41.46	66.63	FAIL	46.06
Sparse Trans.	17.07	63.58	**59.59**	**44.24**	71.71	FAIL	51.24
Longformer	35.63	62.85	56.89	42.22	69.71	FAIL	53.46
Linformer	35.70	53.94	52.27	38.56	76.34	FAIL	51.36
Reformer	**37.27**	56.10	53.40	38.07	68.50	FAIL	50.67
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	FAIL	51.39
Synthesizer	36.99	61.68	54.67	41.61	69.45	FAIL	52.88
BigBird	36.05	64.02	59.29	40.83	74.87	FAIL	**55.01**
Linear Trans.	16.13	**65.90**	53.09	42.34	75.30	FAIL	50.55
Performer	18.01	65.40	53.82	42.77	**77.05**	FAIL	51.41
Task Avg (Std)	29 (9.7)	61 (4.6)	55 (2.6)	41 (1.8)	72 (3.7)	FAIL	52 (2.4)

Experimental results for Long-Range Arena Benchmark

Skoltech

Tay, Yi, et al. "Long range arena: A benchmark for efficient transformers." *arXiv preprint arXiv:2011.04006* (2020).

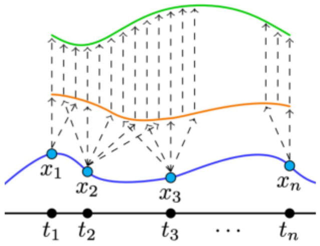
Efficiently Modeling Long Sequences with Structured State Spaces

Skoltech

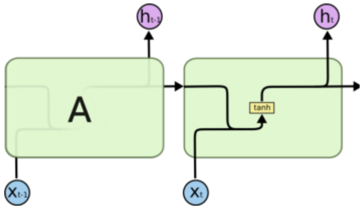
Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Paradigms for sequence modelling

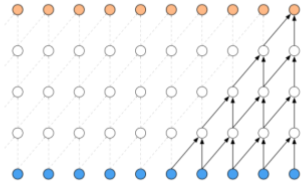
Continuous-time Model



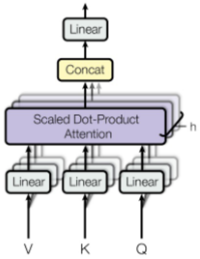
Recurrent Neural Net.



Convolutional Neural Net.



Transformer



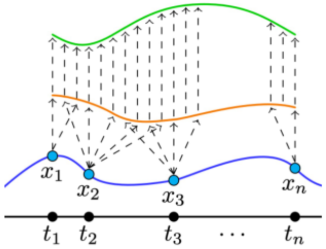
Deep Sequence Model
Sequence-to-sequence map

(batch, length, dim)
↑
Sequence Model Layer
|
(batch, length, dim)

Skoltech

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

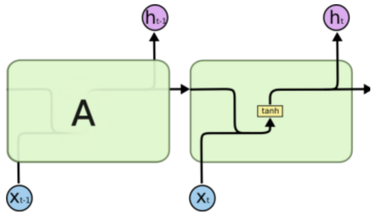
Pros and cons for each paradigm



✓ Continuous data
Irregular sampling

✗ Complex, very inefficient
Vanishing gradients

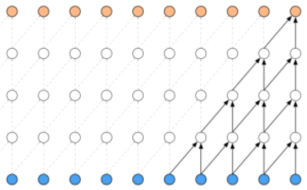
Continuous-time (CTM)



✓ Unbounded context
Stateful inference

✗ Inefficient training
Vanishing gradients

Recurrent (RNN)



✓ Easy optimization
Parallelizable training

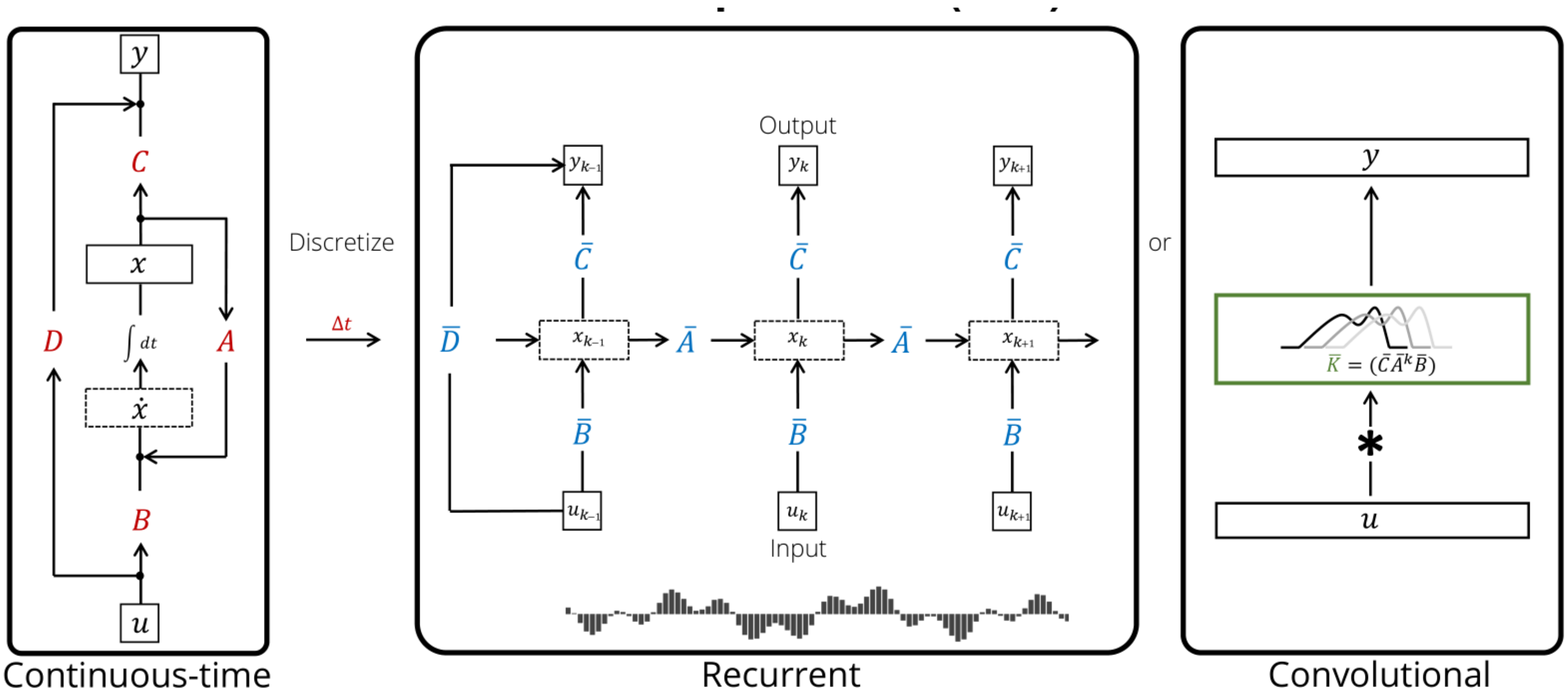
✗ Inefficient inference
Bounded context

Convolutional (CNN)

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

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Structured state-space models



They are all similar! And you will soon know how…

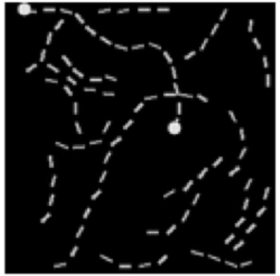
Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Long-range arena benchmark

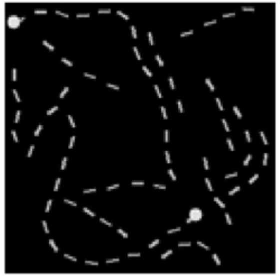
Benchmark spanning text, images, symbolic reasoning (length 1K-16K)

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**

Path-X



(a) A positive example.



(b) A negative example.

Skoltech

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

State Space Models (SSM)

Input → State

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

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

Parameters

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

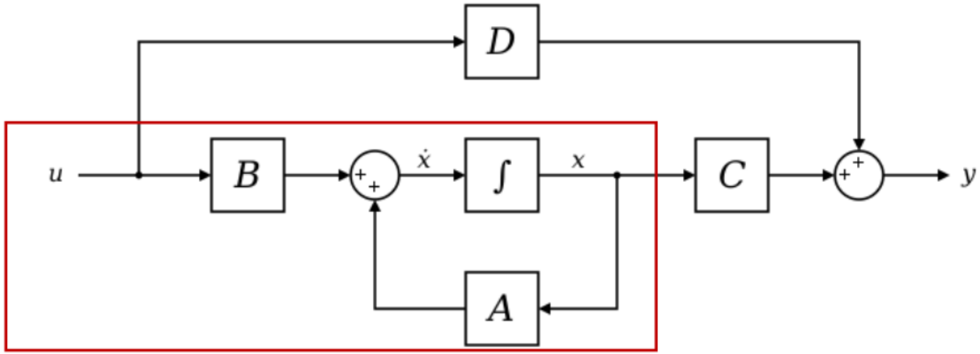
$\boldsymbol{B} \in \mathbb{R}^{N\times 1}$

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

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

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

1D input
u(t)



1D output
y(t)

Skoltech

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

State Space Models (SSM)

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

State → Output $$\boxed{y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)}$$

Parameters

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

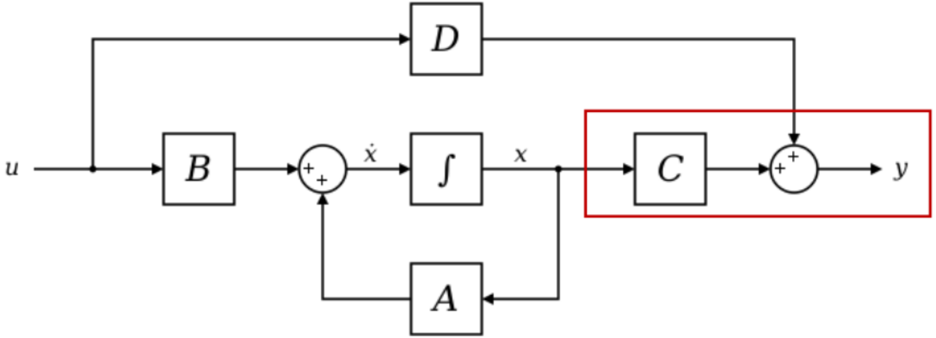
$\boldsymbol{B} \in \mathbb{R}^{N \times 1}$

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

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

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

1D input
u(t)



1D output
y(t)

Skoltech

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Computing with SSMs: Recurrent View

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

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

Parameters

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

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

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

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

$\Delta \in \mathbb{R}$

1. Discretize

$$\overline{\boldsymbol{A}} = \boldsymbol{I} + \Delta \boldsymbol{A}$$

2. Recurrent "hidden state"

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

3. Out projection

$$y_k = \overline{\boldsymbol{C}} x_k + \overline{\boldsymbol{D}} u_k$$

Can be computed with **linear recurrence**, similar to RNNs

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Computing with SSMs: Convolution View

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

$$\boxed{y_k = \overline{\boldsymbol{C}} x_k}$$

Can explicitly unroll the linear recurrence in closed form

$$x_0 = \overline{\boldsymbol{B}} u_0 \qquad x_1 = \overline{\boldsymbol{A}\boldsymbol{B}} u_0 + \overline{\boldsymbol{B}} u_1 \qquad x_2 = \overline{\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{A}\boldsymbol{B}} u_1 + \overline{\boldsymbol{B}} u_2 \qquad \dots$$

$$y_0 = \overline{\boldsymbol{C}\boldsymbol{B}} u_0 \quad y_1 = \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{B}} u_1 \quad y_2 = \overline{\boldsymbol{C}\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_1 + \overline{\boldsymbol{C}\boldsymbol{B}} u_2 \quad \dots$$

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Computing with SSMs: Convolution View

$$y_k = \overline{\boldsymbol{C}}\,\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}}\,\overline{\boldsymbol{A}}^{k-1}\overline{\boldsymbol{B}}u_1 + \cdots + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}\boldsymbol{B}}u_k$$

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

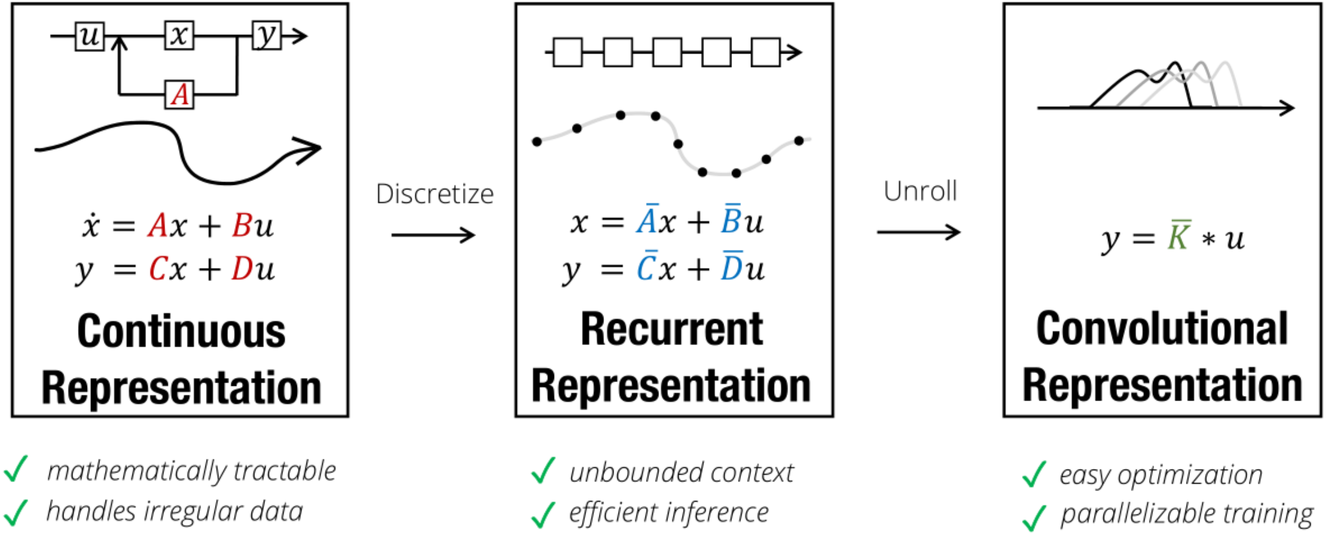
$$y = \overline{\boldsymbol{K}} * u$$

Skoltech

Can be computed with **convolutions**, similar to CNNs

Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Summary: Properties of SSMs



Gu, Albert, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. *ICLR*. 2021.

Conclusions

- We can improve Transformers memory and computational requirements with some heuristics and different types of attention

- The main ideas are:
  - combine different sparse attention mechanisms
  - bucket processing

- But there are some limitations in terms of reducing number of parameters and memory as we have large window sizes and large number of global attention nodes