iTransformer & Mamba

Papers

Attention is all you need – Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

iTransformer: inverted Transformers are effective for time series forecasting

Published as a conference paper at ICLR 2024

iTRANSFORMER: INVERTED TRANSFORMERS ARE EFFECTIVE FOR TIME SERIES FORECASTING

Yong Liu, **Tengge Hu**, **Haoran Zhang**, **Haixu Wu**, **Shiyu Wang**§, **Lintao Ma**§, **Mingsheng Long**✉
School of Software, BNRist, Tsinghua University, Beijing 100084, China
§Ant Group, Hangzhou, China
{liuyong21,htg21,z-hr20,whx20}@mails.tsinghua.edu.cn
{weiming.wsy,lintao.mlt}@antgroup.com,mingsheng@tsinghua.edu.cn

ABSTRACT

The recent boom of linear forecasting models questions the ongoing passion for architectural modifications of Transformer-based forecasters. These forecasters leverage Transformers to model the global dependencies over *temporal tokens* of time series, with each token formed by multiple variates of the same timestamp. However, Transformers are challenged in forecasting series with larger lookback windows due to performance degradation and computation explosion. Besides, the embedding for each temporal token fuses multiple variates that represent potential delayed events and distinct physical measurements, which may fail in learning variate-centric representations and result in meaningless attention maps. In this work, we reflect on the competent duties of Transformer components and repurpose the Transformer architecture without any modification to the basic components. We propose **iTransformer** that simply applies the attention and feed-forward network on the inverted dimensions. Specifically, the time points of individual series are embedded into *variate tokens* which are utilized by the attention mechanism to capture multivariate correlations; meanwhile, the feed-forward network is applied for each variate token to learn nonlinear representations. The iTransformer model achieves state-of-the-art on challenging real-world datasets, which further empowers the Transformer family with promoted performance, generalization ability across different variates, and better utilization of arbitrary lookback windows, making it a nice alternative as the fundamental backbone of time series forecasting. Code is available at this repository: https://github.com/thuml/iTransformer.

1 INTRODUCTION

Transformer (Vaswani et al., 2017) has achieved tremendous success in natural language processing (Brown et al., 2020) and computer vision (Dosovitskiy et al., 2021), growing into the foundation model that follows the scaling law (Kaplan et al., 2020). Inspired by the immense success in extensive fields, Transformer with strong capabilities of depicting pairwise dependencies and extracting multi-level representations in sequences is emerging in time series forecasting (Wu et al., 2021; Nie et al., 2023).

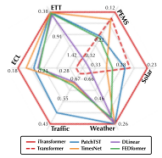


Figure 1: Performance of iTransformer. Average results (MSE) are reported following TimesNet (2023).

However, researchers have recently begun to question the validity of Transformer-based forecasters, which typically embed multiple variates of the same timestamp into indistinguishable channels and apply attention on these *temporal tokens* to capture temporal dependencies. Considering the numerical but less semantic relationship among time points, researchers find that simple linear layers, which can be traced back to statistical forecasters (Box & Jenkins, 1968), have exceeded complicated Transformers on both performance and efficiency (Zeng et al., 2023; Das et al., 2023). Meanwhile, ensuring the independence of variate and utilizing mutual

*Equal Contribution

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu* and **Tri Dao***
*Machine Learning Department, Carnegie Mellon University
†Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

Abstract

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several improvements. First, simply letting the SSM parameters be functions of the input addresses their weakness with discrete modalities, allowing the model to selectively propagate or forget information along the sequence length dimension depending on the current token. Second, even though this change prevents the use of efficient convolutions, we design a hardware-aware parallel algorithm in recurrent mode. We integrate these selective SSMs into a simplified end-to-end neural network architecture without attention or even MLP blocks (Mamba). Mamba enjoys fast inference (5× higher throughput than Transformers) and linear scaling in sequence length, and its performance improves on real data up to million-length sequences. As a general sequence model backbone, Mamba achieves state-of-the-art performance across several modalities such as language, audio, and genomics. On language modeling, our Mamba-3B model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream evaluation.

1 Introduction

Foundation models (FMs), or large models pretrained on massive data then adapted for downstream tasks, have emerged as an effective paradigm in modern machine learning. The backbone of these FMs are often sequence models, operating on arbitrary sequences of inputs from a wide variety of domains such as language, images, speech, audio, time series, and genomics (Brown et al. 2020; Dosovitskiy et al. 2020; Ismail Fawaz et al. 2020; Oord et al. 2016; Poli et al. 2023; Sutskever, Vinyals, and Quoc V Le 2014). While this concept is agnostic to a particular choice of model architecture, modern FMs are predominantly based on a single type of sequence model: the Transformer (Vaswani et al. 2017) and its core attention layer (Bahdanau, Cho, and Bengio 2015). The efficacy of self-attention is attributed to its ability to route information densely within a context window, allowing it to model complex data. However, this property brings fundamental drawbacks: an inability to model anything outside of a finite window, and quadratic scaling with respect to the window length. An enormous body of research has appeared on more efficient variants of attention to overcome these drawbacks (Tay, Dehghani, Bahri, et al. 2022), but often at the expense of the very properties that makes it effective. As of yet, none of these variants have been shown to be empirically effective at scale across domains.

Recently, structured state space sequence models (SSMs) (Gu, Goel, and Ré 2022; Gu, Johnson, Goel, et al. 2021) have emerged as a promising class of architectures for sequence modeling. These models can be interpreted as a combination of recurrent neural networks (RNNs) and convolutional neural networks (CNNs), with inspiration from classical state space models (Kalman 1960). This class of models can be computed very efficiently as either a recurrence or convolution, with linear or near-linear scaling in sequence length. Additionally, they have principled

*Equal contribution.

Transformer review

Transformer Review

structure

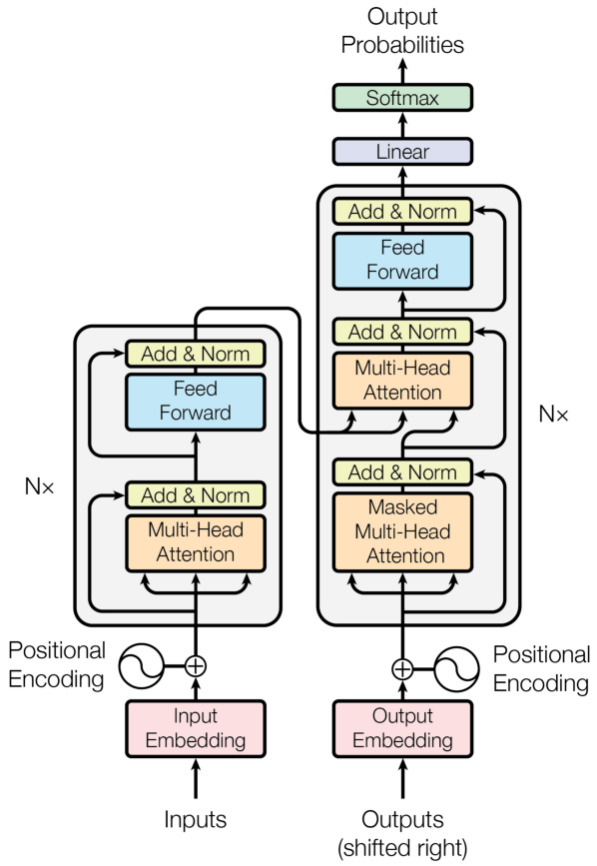


Figure 1: The Transformer - model architecture.

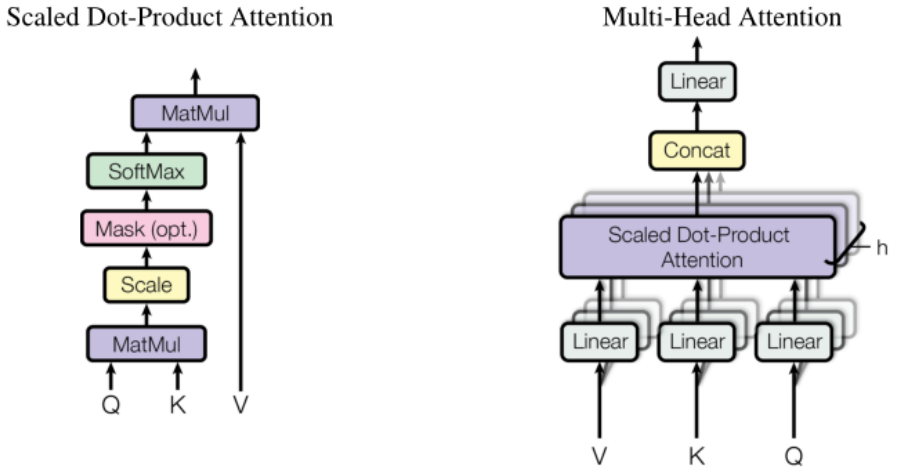


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

在transformer中，Self-Attention关注序列内每个位置对其他所有位置的重要性，而Multi-Head Attention则通过在多个子空间中并行计算注意力，使模型能够同时捕获和整合不同方面的上下文信息

FFN层则是一个两层的MLP，用来学习特征；同时也可以通过激活函数的方式，强化模型的表达能力

Transformer Review

structure

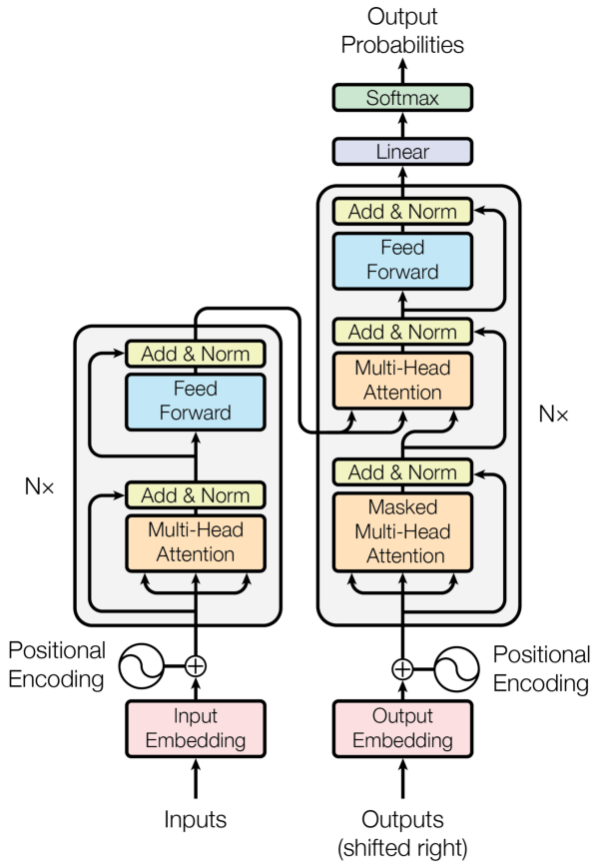


Figure 1: The Transformer - model architecture.

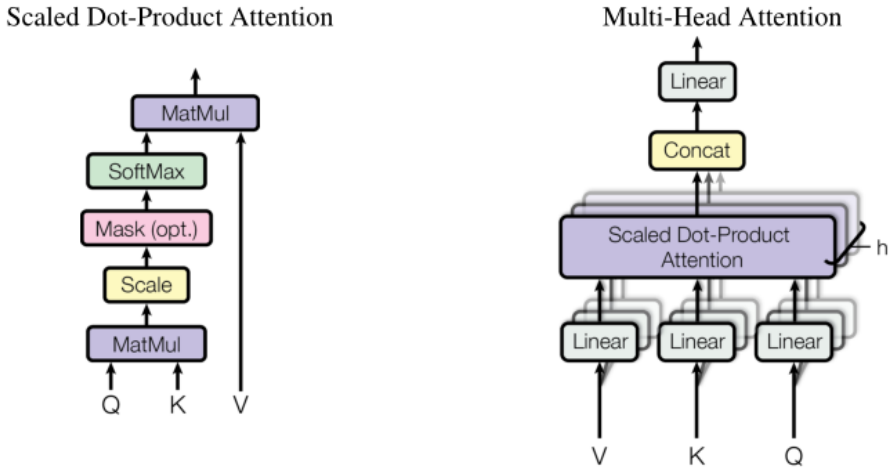


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

- **相比于CNN**

  transformer通过自注意力，可以实现对长距离文本的记忆性，有效避免了CNN的对长距离信息的捕获问题(缺陷是带来的比较大的内存开销)

- **相比于RNN**

  Transformer避免了递归过程，可以完全使用attention的输入输出来建立全局关系，有效实现并行化

Transformer Review

Algorithm

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V$$

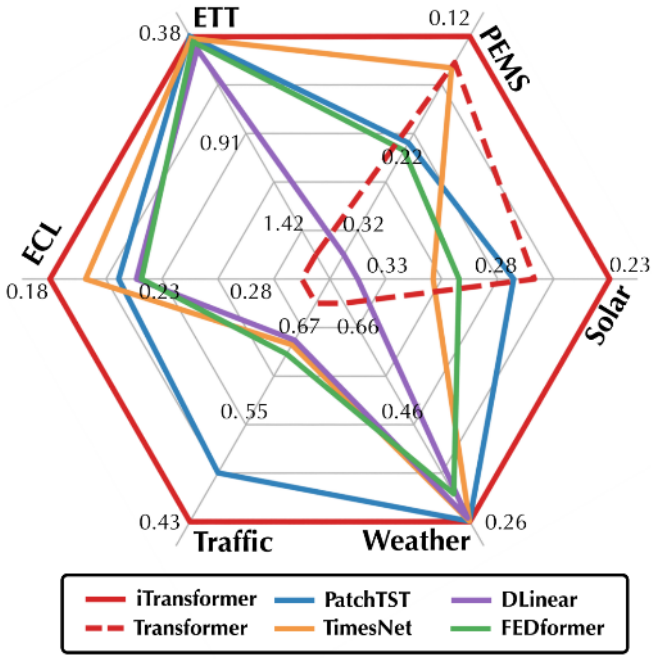
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

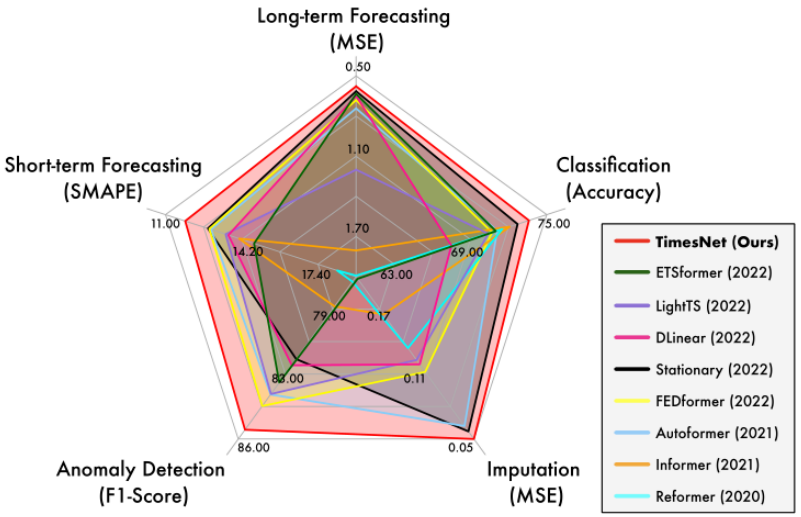
Complexity

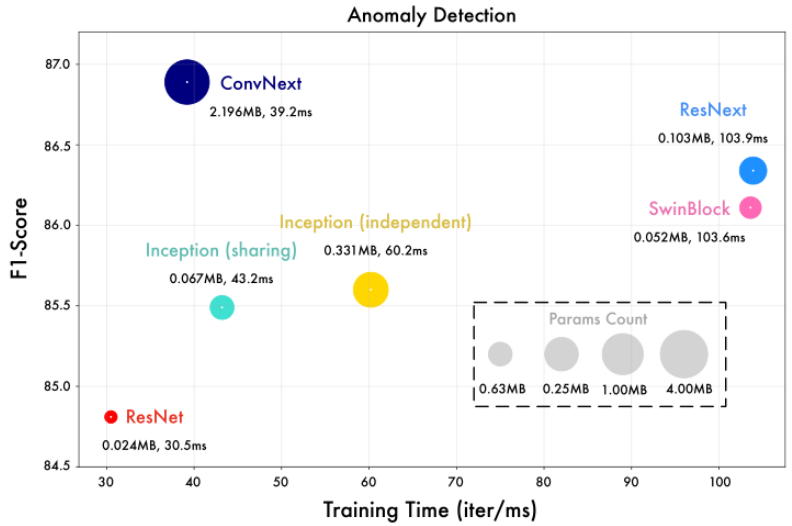
Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$



Transformer 的缺陷





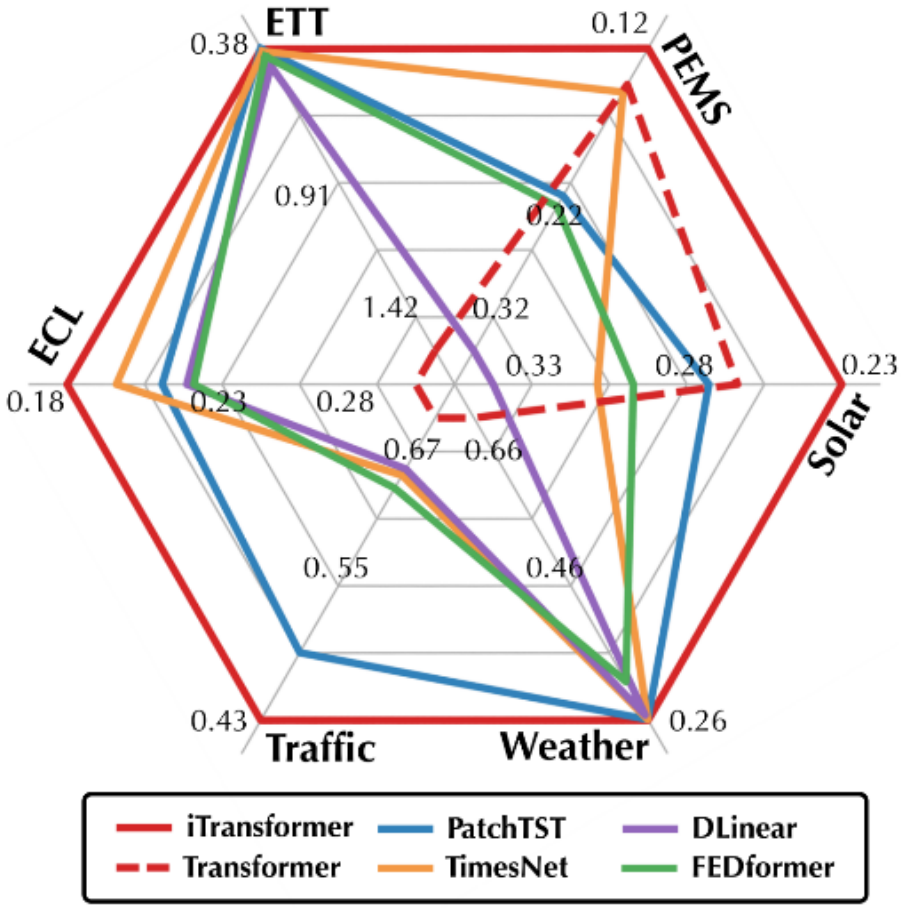
- **长序列上非常消耗显存**
  - 前文可见，Transformer的**计算复杂度与序列长度呈平方关系**，这意味着在处理具有较大回顾窗口的长程序列时，transformer会出现**性能下降和计算量激增**的问题
- **时序预测领域效果不突出**
  - 基于transformer模型的时序预测架构通常会引入时间步，将时间/词序连同变量编码成统一的temporal token，利用自注意力机制去捕捉不同时刻变量之间的时序相关性，这实际上**破坏了变量之间由于不同物理意义带来的相关性**，所以效果并不优异

Transformer的改进↓

- iTransformer
  - 【较为温和】
  - 不改变transformer原有组件，也不添加新的结构，而通过重新设计transformer的结构，从而使transformer在时序预测领域取得更优秀的性能
  - 应用领域主要在transformer的时序预测方向

- Mamba
  - 【崭新模型框架】
  - 基于状态空间理论构建一种新的模型框架，致力于实现比Transformer更快的训练与推理速度
  - 同时，针对Transformer模型所面对的高存储与运行开销问题，Mamba也尝试通过硬件扫描方法，以一种更符合硬件动作的方式来优化内存管理
  - 全新的大模型框架

iTransformer



- 基于Transformer模型设计的预测器，通常会将同一时间步的多个变量嵌入到不可区分的通道中，编码成一个统一的多维Temporal token，并将注意力集中在这些token上来捕获时间相关性。

- **一些局限性：**

1. 对于同一时间步的数据点，他们具有不同的物理意义，采集时间可能不对齐并且尺度差异大，如果强行将它们编码为统一的temporal token也不再区分channels时，多变量间的相关性会被消除，无法学习以变量为基础的高效表征
2. 变量存在时滞性，一个时间点所包含的信息量比较局限，从每个时刻出发，可能并不利于建模全局的时序相关性
3. 在建模数据沿时间方向的长期相关性时，需要考虑历史窗口长度不断增加所带来的性能下降和计算量爆炸问题

iTransformer

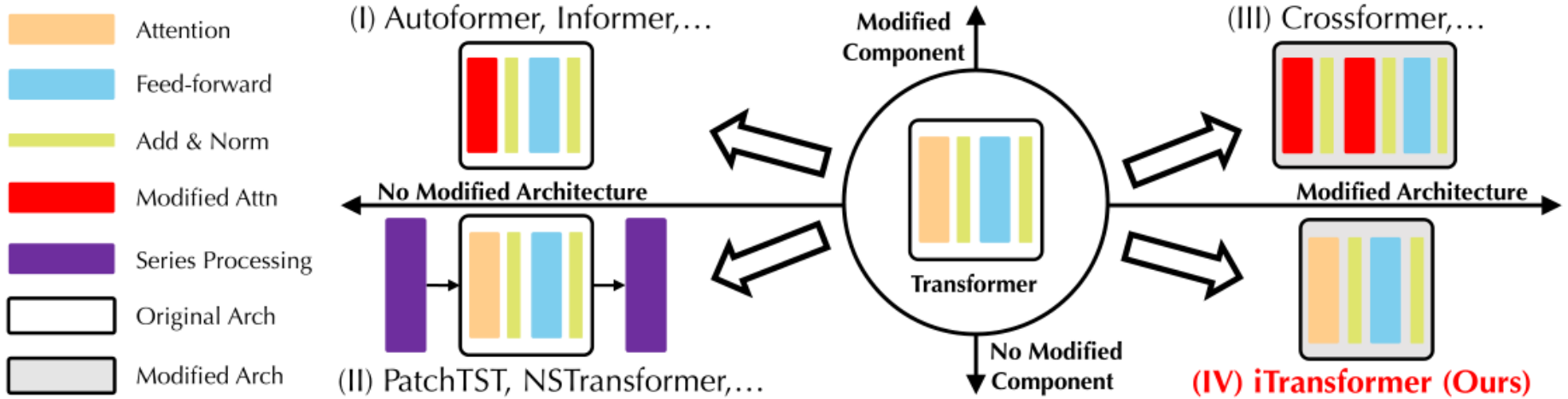


Figure 3: Transformer-based forecasters categorized by component and architecture modifications.

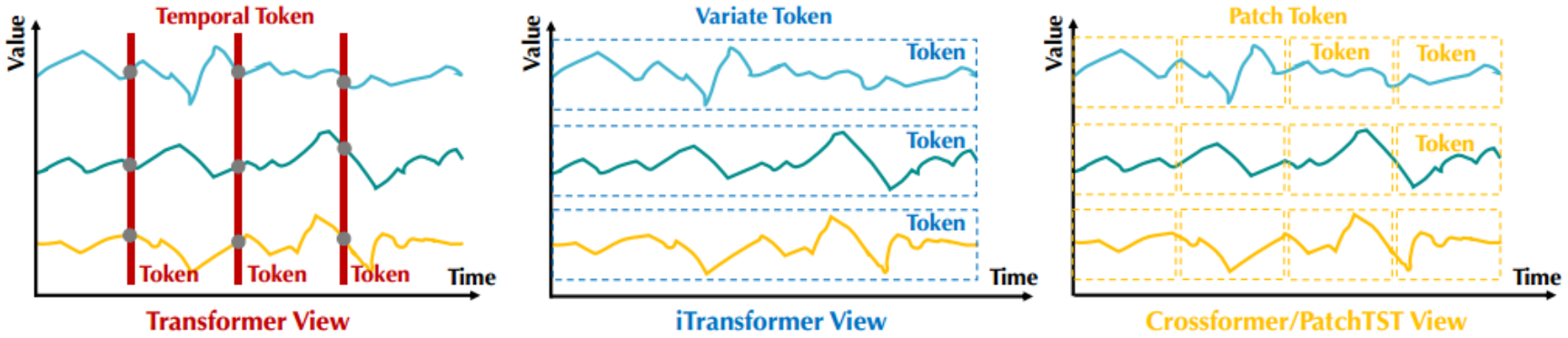
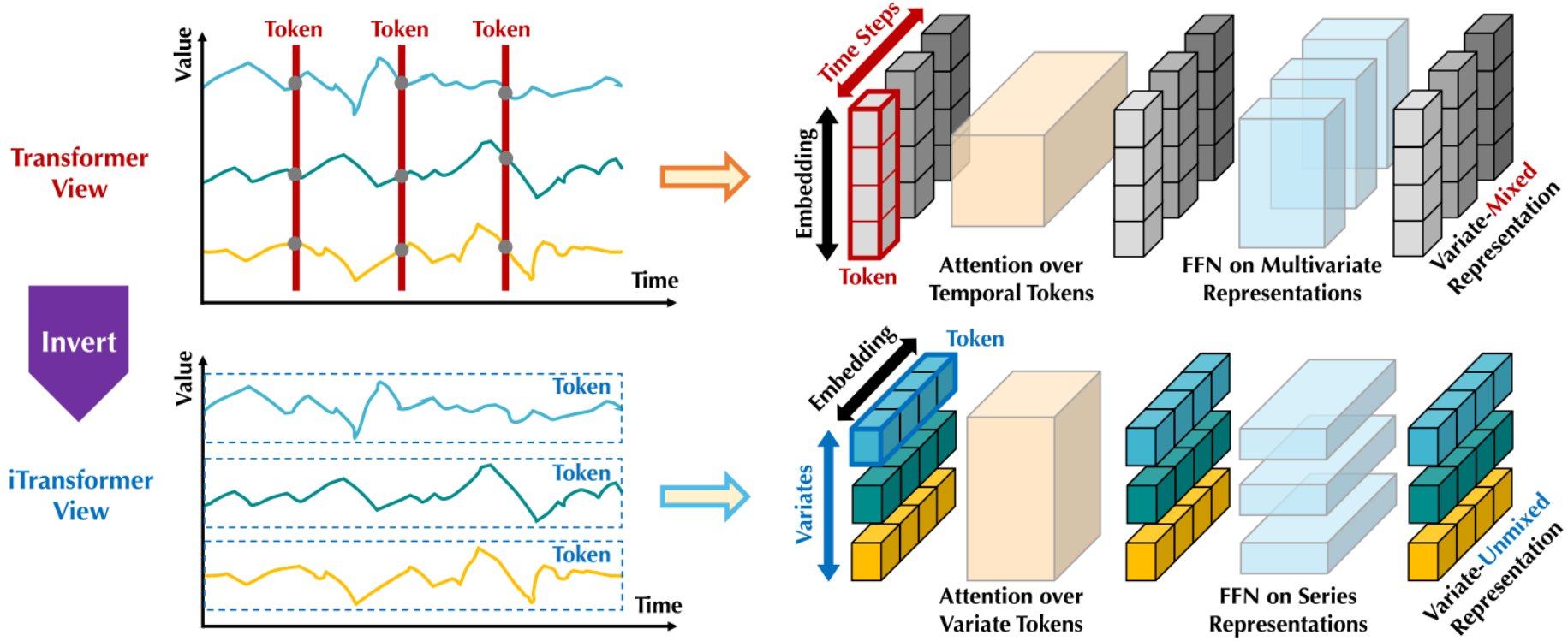


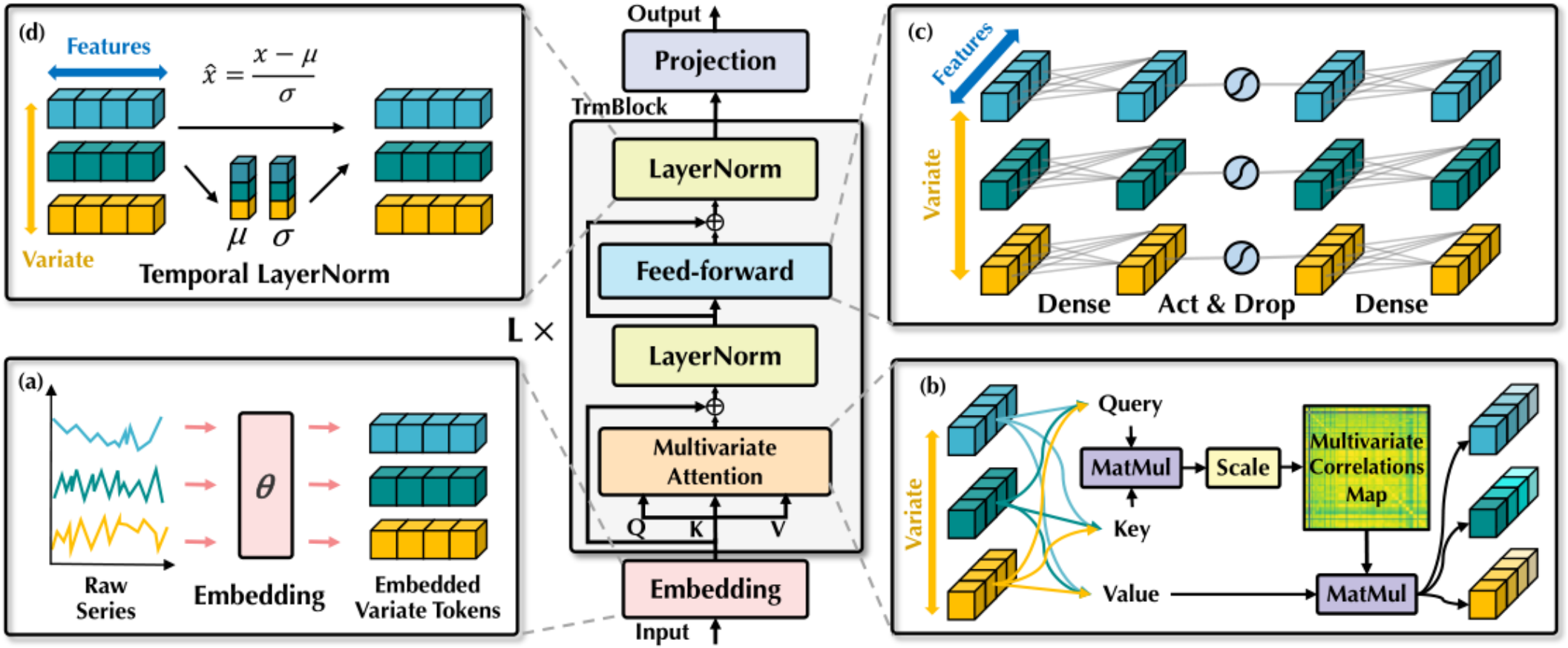
Figure 19: Tokenizations for multivariate time series modality of representative Transformers.

iTransformer

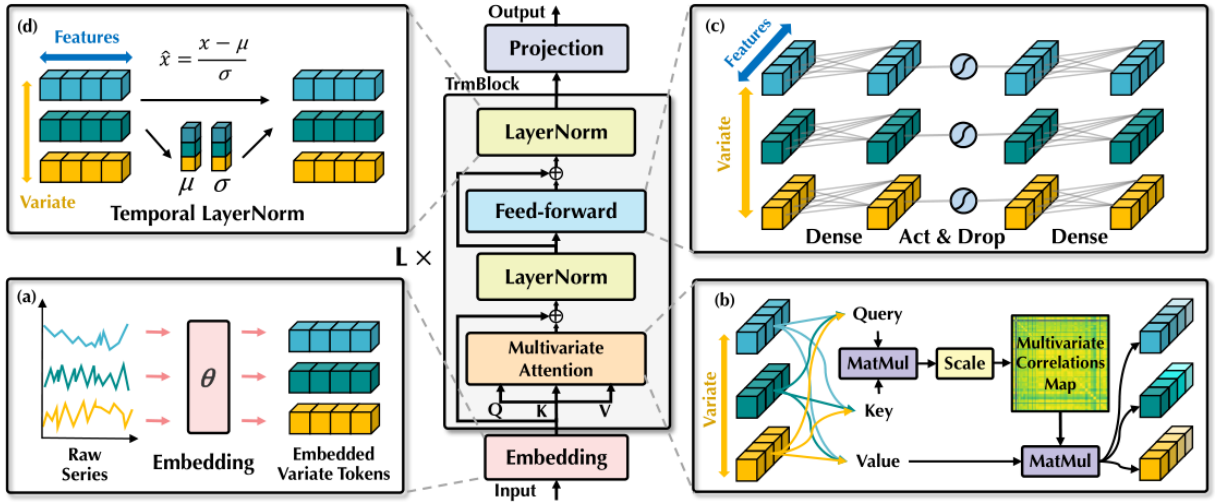


基于前文时序预测中遇到的问题，本文提出了一种全新的transformer架构，在不改变transformer原有的网络架构下，通过重新设计token的选取原则，以及倒置注意力机制和前馈神经网络的作用，来实现更优异的时间序列分析预测任务

iTransformer



iTransformer



普遍近似定理（universal approximation theorem）(Hornik et al., 1989;Cybenko, 1989) 表明，一个前馈神经网络如果具有线性输出层和至少一层具有任何一种‘‘挤压’’性质的激活函数（例如logistic sigmoid激活函数）的隐藏层，只要给予网络足够数量的隐藏单元，它可以以任意的精度来近似任何从一个有限维空间到另一个有限维空间的Borel 可测函数。

万能近似定理意味着无论我们试图学习什么函数，我们知道一个大的MLP 一定能够表示这个函数。

在反向版本中，FFN针对每个token的序列表示，利用普遍近似定理(Hornik, 1991)，提取复杂的函数表示来描述时间序列（t,w,f,...）

通过倒置模块层次的堆叠，文章将时间序列编码后使用密集的非线性连接来解码未来序列的表示，通过这种方式有效地提高时序预测的性能。

该过程可描述为以下内容：

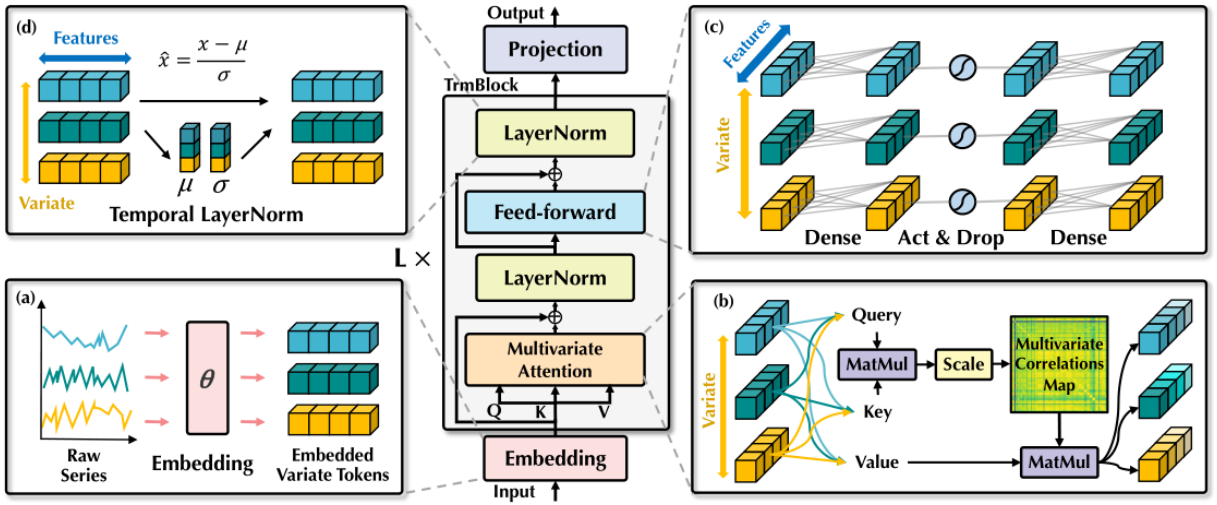
$$\mathbf{h}_n^0 = \mathrm{Embedding}(\mathbf{X}_{:,n}),$$
$$\mathbf{H}^{l+1} = \mathrm{TrmBlock}(\mathbf{H}^l),\ l = 0, \ldots, L-1,$$
$$\hat{\mathbf{Y}}_{:,n} = \mathrm{Projection}(\mathbf{h}_n^L),$$

其中，Embedding和Projection都由MLP实现

得到的变量token通过self-attention相互作用，并在TrmBlock中由FFN独立处理

由于序列的顺序隐式地存储在前馈神经网络的神经元排列中，因此这里不再需要在普通的transformer中嵌入位置

iTransformer



Algorithm 1 iTransformer - Overall Architecture.

Require: Input lookback time series $\mathbf{X} \in \mathbb{R}^{T \times N}$; input Length T; predicted length S; variates number N; token dimension D; iTransformer block number L.

$\mathbf{X} = \mathbf{X}.\texttt{transpose}$ ▷ $\mathbf{X} \in \mathbb{R}^{N \times T}$

▷ Multi-layer Perceptron works on the last dimension to embed series into variate tokens.

$\mathbf{H}^0 = \texttt{MLP}(\mathbf{X})$ ▷ $\mathbf{H}^0 \in \mathbb{R}^{N \times D}$

for l **in** $\{1, \ldots, L\}$**:** ▷ Run through iTransformer blocks.

    ▷ Self-attention layer is applied on variate tokens.

    $\mathbf{H}^{l-1} = \texttt{LayerNorm}\big(\mathbf{H}^{l-1} + \texttt{Self-Attn}(\mathbf{H}^{l-1})\big)$ ▷ $\mathbf{H}^{l-1} \in \mathbb{R}^{N \times D}$

    ▷ Feed-forward network is utilized for series representations, broadcasting to each token.

    $\mathbf{H}^{l} = \texttt{LayerNorm}\big(\mathbf{H}^{l-1} + \texttt{Feed-Forward}(\mathbf{H}^{l-1})\big)$ ▷ $\mathbf{H}^{l} \in \mathbb{R}^{N \times D}$

    ▷ LayerNorm is adopted on series representations to reduce variates discrepancies.

End for

$\hat{\mathbf{Y}} = \texttt{MLP}(\mathbf{H}^L)$ ▷ Project tokens back to predicted series, $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times S}$

$\hat{\mathbf{Y}} = \hat{\mathbf{Y}}.\texttt{transpose}$ ▷ $\hat{\mathbf{Y}} \in \mathbb{R}^{S \times N}$

Return $\hat{\mathbf{Y}}$ ▷ Return the prediction result $\hat{\mathbf{Y}}$

iTransformer

在本节中，作者进行了广泛的实验，以评估提出的模型与高级深度预测器的预测性能。

Table 1: Multivariate forecasting results with prediction lengths $S \in \{12, 24, 36, 48\}$ for PEMS and $S \in \{96, 192, 336, 720\}$ for others and fixed lookback length $T = 96$. Results are averaged from all prediction lengths. *Avg* means further averaged by subsets. Full results are listed in Appendix F.4.

Models	**iTransformer (Ours)**		RLinear (2023)		PatchTST (2023)		Crossformer (2023)		TiDE (2023)		TimesNet (2023)		DLinear (2023)		SCINet (2022a)		FEDformer (2022)		Stationary (2022b)		Autoformer (2021)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	**0.178**	**0.270**	0.219	0.298	0.205	0.290	0.244	0.334	0.251	0.344	0.192	0.295	0.212	0.300	0.268	0.365	0.214	0.327	0.193	0.296	0.227	0.338
ETT (Avg)	0.383	0.399	**0.380**	**0.392**	0.381	0.397	0.685	0.578	0.482	0.470	0.391	0.404	0.442	0.444	0.689	0.597	0.408	0.428	0.471	0.464	0.465	0.459
Exchange	0.360	**0.403**	0.378	0.417	0.367	0.404	0.940	0.707	0.370	0.413	0.416	0.443	**0.354**	0.414	0.750	0.626	0.519	0.429	0.461	0.454	0.613	0.539
Traffic	**0.428**	**0.282**	0.626	0.378	0.481	0.304	0.550	0.304	0.760	0.473	0.620	0.336	0.625	0.383	0.804	0.509	0.610	0.376	0.624	0.340	0.628	0.379
Weather	**0.258**	**0.278**	0.272	0.291	0.259	0.281	0.259	0.315	0.271	0.320	0.259	0.287	0.265	0.317	0.292	0.363	0.309	0.360	0.288	0.314	0.338	0.382
Solar-Energy	**0.233**	**0.262**	0.369	0.356	0.270	0.307	0.641	0.639	0.347	0.417	0.301	0.319	0.330	0.401	0.282	0.375	0.291	0.381	0.261	0.381	0.885	0.711
PEMS (Avg)	**0.119**	**0.218**	0.514	0.482	0.217	0.305	0.220	0.304	0.375	0.440	0.148	0.246	0.320	0.394	0.121	0.222	0.224	0.327	0.151	0.249	0.614	0.575

iTransformer

在本节中，作者通过将框架应用于Transformer及其变体来评估itransformer预测精度

Table 2: Performance promotion obtained by our inverted framework. Flashformer means Transformer equipped with hardware-accelerated FlashAttention (Dao et al., 2022). We report the average performance and the relative MSE reduction (Promotion). Full results can be found in Appendix F.2.

Models		Transformer (2017)		Reformer (2020)		Informer (2021)		Flowformer (2022)		Flashformer (2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	Original	0.277	0.372	0.338	0.422	0.311	0.397	0.267	0.359	0.285	0.377
	+Inverted	**0.178**	**0.270**	**0.208**	**0.301**	**0.216**	**0.311**	**0.210**	**0.293**	**0.206**	**0.291**
	Promotion	35.6%	27.4%	38.4%	28.7%	30.5%	21.6%	21.3%	18.6%	27.8%	22.9%
Traffic	Original	0.665	0.363	0.741	0.422	0.764	0.416	0.750	0.421	0.658	0.356
	+Inverted	**0.428**	**0.282**	**0.647**	**0.370**	**0.662**	**0.380**	**0.524**	**0.355**	**0.492**	**0.333**
	Promotion	35.6%	22.3%	12.7%	12.3%	13.3%	8.6%	30.1%	15.6%	25.2%	6.4%
Weather	Original	0.657	0.572	0.803	0.656	0.634	0.548	0.286	0.308	0.659	0.574
	+Inverted	**0.258**	**0.279**	**0.248**	**0.292**	**0.271**	**0.330**	**0.266**	**0.285**	**0.262**	**0.282**
	Promotion	60.2%	50.8%	69.2%	55.5%	57.3%	39.8%	7.2%	7.7%	60.2%	50.8%

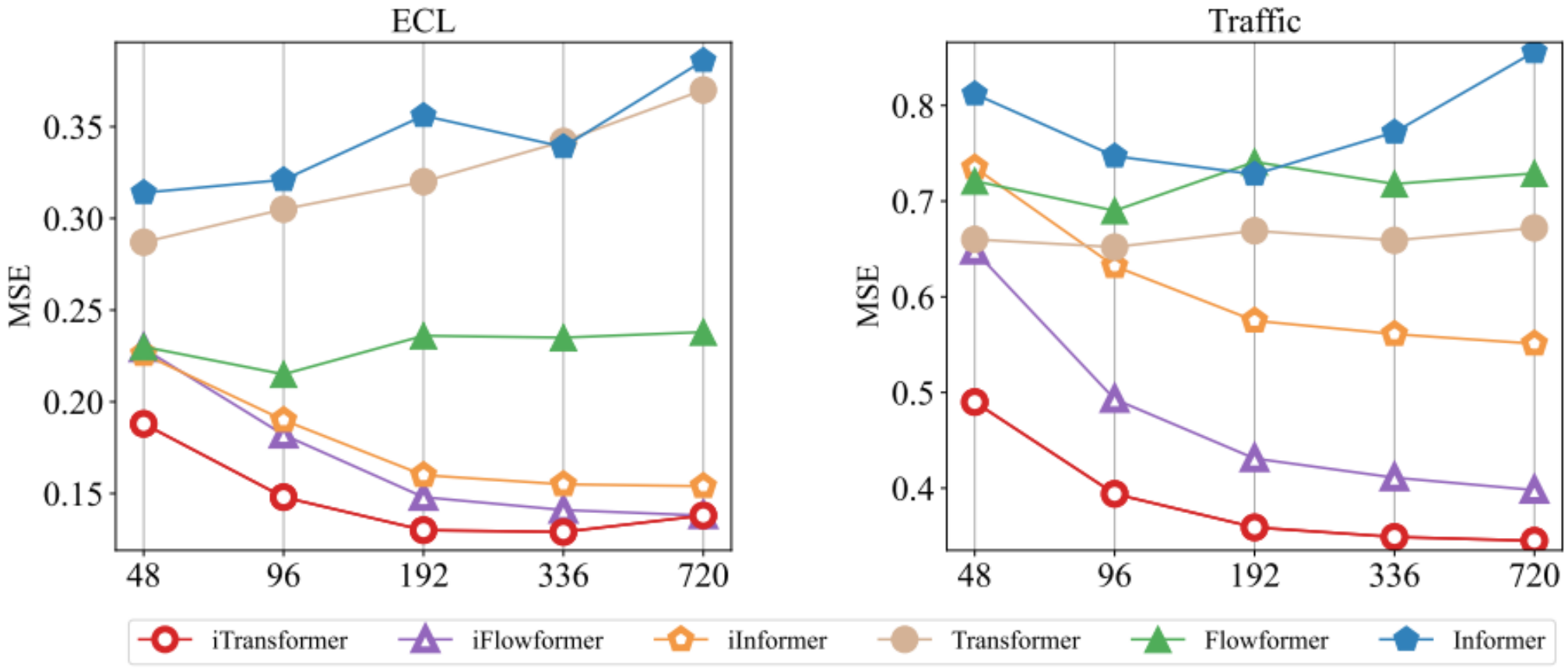
iTransformer

泛化能力

Figure 5: Performance of generalization on unseen variates. We partition the variates of each dataset into five folders, train models with 20% variates, and use the partially trained model to forecast all varieties. iTransformers can be trained efficiently and forecast with good generalizability.

iTransformer

回顾窗口长度



消融实验

Table 3: Ablations on iTransformer. We replace different components on the respective dimension to learn multivariate correlations (Variate) and series representations (Temporal), in addition to component removal. The average results of all predicted lengths are listed here.

Design	Variate	Temporal	ECL		Traffic		Weather		Solar-Energy	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
iTransformer	**Attention**	**FFN**	**0.178**	**0.270**	**0.428**	**0.282**	0.258	0.278	**0.233**	**0.262**
Replace	Attention	Attention	0.193	0.293	0.913	0.500	0.255	0.280	0.261	0.291
	FFN	Attention	0.202	0.300	0.863	0.499	0.258	0.283	0.285	0.317
	FFN	FFN	0.182	0.287	0.599	0.348	**0.248**	**0.274**	0.269	0.287
w/o	Attention	w/o	0.189	0.278	0.456	0.306	0.261	0.281	0.258	0.289
	w/o	FFN	0.193	0.276	0.461	0.294	0.265	0.283	0.261	0.283

多元相关性

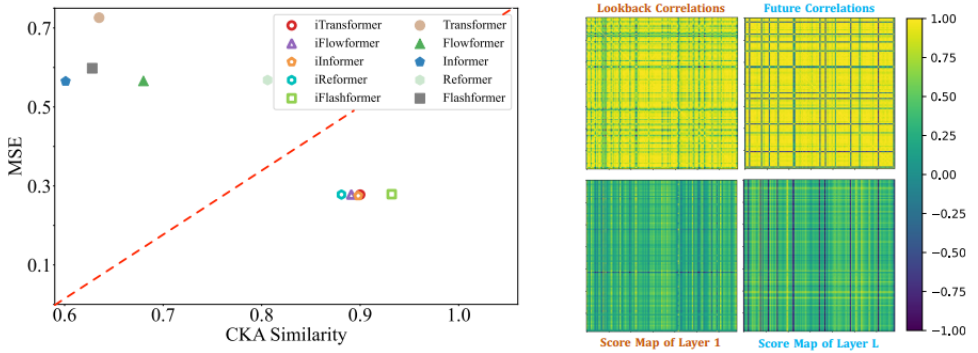


Figure 7: Analysis of series representations and multivariate correlations. Left: MSE and CKA similarity of representations comparison between Transformers and iTransformers. A higher CKA similarity indicates more favored representations for accurate predictions. Right: A case visualization of multivariate correlations of raw time series and the learned score maps by inverted self-attention.

具体来说，我们在每批数据中随机选择一部分变量，只使用选定的变量来训练模型。

由于我们的反演使得变量通道的数量是灵活的，所以模型可以对所有的变量进行预测。如图8所示，我们提出的策略的性能仍然可以与全变量训练相媲美，而内存占用可以显着减少

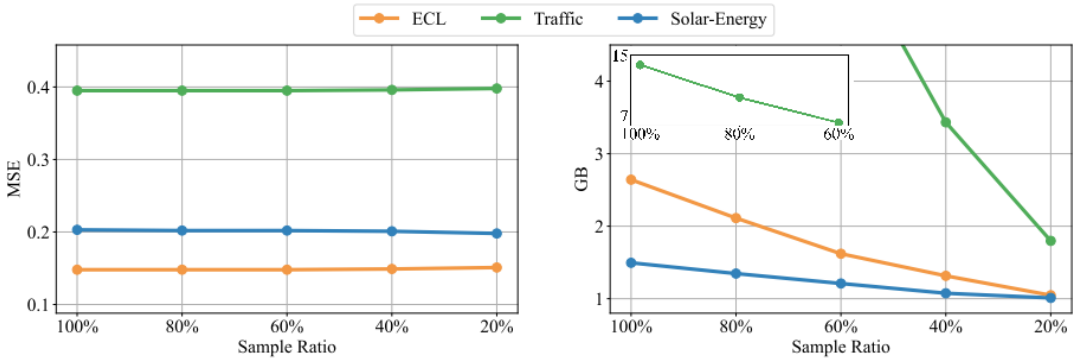


Figure 8: Analysis of the efficient training strategy. While the performance (left) remains stable on partially trained variates of each batch with different sampled ratios, the memory footprint (right) can be cut off greatly. We provide the comprehensive model efficiency analysis in Appendix D.

iTransformer

- iTransformer小结
  - 考虑到多元时间序列的特点，作者提出了在不修改任何固有模块的情况下对Transformer的结构进行翻转的iTransformer。
  - iTransformer将独立序列作为变量token，通过注意力来捕获多变量相关性，并利用LayerNorm和FFN学习序列表示，取得了最先进的性能。
  - 一些启发：Transformer本身并不一定是不适合来做多元时间序列预测任务，可能是原有的结构设计不恰当，通过合理的结构设计可以使模型得到更优化的结果

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Submission history

From: Albert Gu [view email]

[v1] Fri, 1 Dec 2023 18:01:34 UTC (1,264 KB)

卡内基梅隆大学机器学习系助理教授 Albert Gu和普林斯顿大学计算机科学系即将上任的助理教授Tri Dao，联合提出一项名为「MAMBA」的研究

Mamba研究背景

Transformer架构的模型有下述两个局限：

1. **计算复杂度问题**。O（N^2）的复杂度带来的是在长序列中的注意力散失，带来性能下降以及计算量的爆炸
2. **可解释性差**。尽管Transformer的数学表示比较简单，但是目前仍无法精准理解Transformer的输出选择

为了解决Transformer的这类问题，各种架构变体涌现（参考前面讲的iTransformer），但是往往这类解决方案会降低框架在其他方面的原有优点，或者在实践中效果不佳。

另一个研究前景就是去开发全新于Transformer的框架。Mamba就是在此基础上进行的研发。

前置知识

- SSM状态空间模型
- S4
- 连续空间离散化
- 递归和卷积设计

Efficiently Modeling Long Sequences with Structured State Spaces

Albert Gu, Karan Goel, and Christopher Ré

Department of Computer Science, Stanford University

{albertgu,krng}@stanford.edu, chrismre@cs.stanford.edu

Abstract

A central goal of sequence modeling is designing a single principled model that can address sequence data across a range of modalities and tasks, particularly on long-range dependencies. Although conventional models including RNNs, CNNs, and Transformers have specialized variants for capturing long dependencies, they still struggle to scale to very long sequences of 10000 or more steps. A promising recent approach proposed modeling sequences by simulating the fundamental state space model (SSM) $x'(t) = Ax(t) + Bu(t), y(t) = Cx(t) + Du(t)$, and showed that for appropriate choices of the state matrix A, this system could handle long-range dependencies mathematically and empirically. However, this method has prohibitive computation and memory requirements, rendering it infeasible as a general sequence modeling solution. We propose the Structured State Space sequence model (S4) based on a new parameterization for the SSM, and show that it can be computed much more efficiently than prior approaches while preserving their theoretical strengths. Our technique involves conditioning A with a low-rank correction, allowing it to be diagonalized stably and reducing the SSM to the well-studied computation of a Cauchy kernel. S4 achieves strong empirical results across a diverse range of established benchmarks, including (i) 91% accuracy on sequential CIFAR-10 with no data augmentation or auxiliary losses, on par with a larger 2-D ResNet, (ii) substantially closing the gap to Transformers on image and language modeling tasks, while performing generation 60× faster (iii) SoTA on every task from the Long Range Arena benchmark, including solving the challenging Path-X task of length 16k that all prior work fails on, while being as efficient as all competitors.

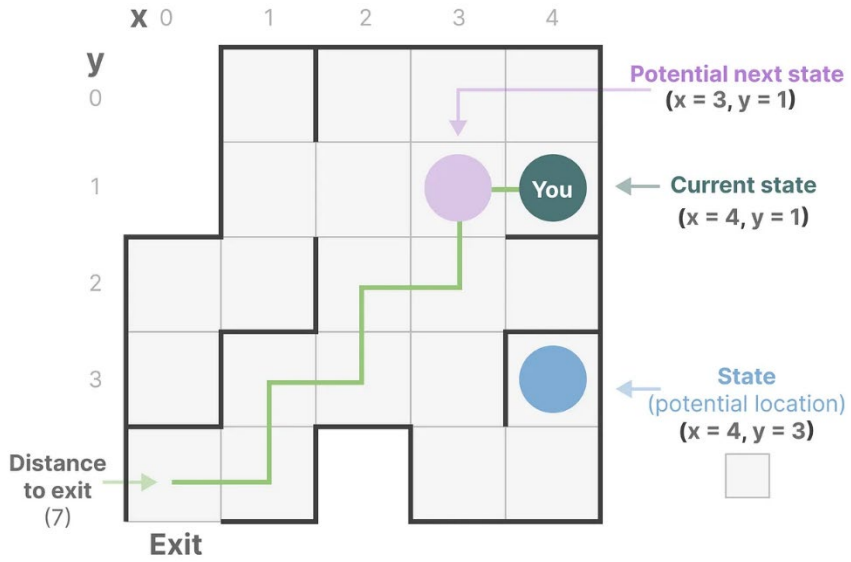
HiPPO: Recurrent Memory with Optimal Polynomial Projections

Albert Gu*†, Tri Dao*†, Stefano Ermon †, Atri Rudra ‡, Christopher Ré †
† Department of Computer Science, Stanford University
‡ Department of Computer Science and Engineering, University at Buffalo, SUNY
{albertgu,trid}@stanford.edu, ermon@cs.stanford.edu, atri@buffalo.edu, chrismre@cs.stanford.edu

Abstract

A central problem in learning from sequential data is representing cumulative history in an incremental fashion as more data is processed. We introduce a general framework (HiPPO) for the online compression of continuous signals and discrete time series by projection onto polynomial bases. Given a measure that specifies the importance of each time step in the past, HiPPO produces an optimal solution to a natural *online function approximation* problem. As special cases, our framework yields a short derivation of the recent Legendre Memory Unit (LMU) from first principles, and generalizes the ubiquitous gating mechanism of recurrent neural networks such as GRUs. This formal framework yields a new memory update mechanism (HiPPO-LegS) that scales through time to remember all history, avoiding priors on the timescale. HiPPO-LegS enjoys the theoretical benefits of timescale robustness, fast updates, and bounded gradients. By incorporating the memory dynamics into recurrent neural networks, HiPPO RNNs can empirically capture complex temporal dependencies. On the benchmark permuted MNIST dataset, HiPPO-LegS sets a new state-of-the-art accuracy of 98.3%. Finally, on a novel trajectory classification task testing robustness to out-of-distribution timescales and missing data, HiPPO-LegS outperforms RNN and neural ODE baselines by 25-40% accuracy.

Mamba – 状态空间模型



状态空间模型

信息

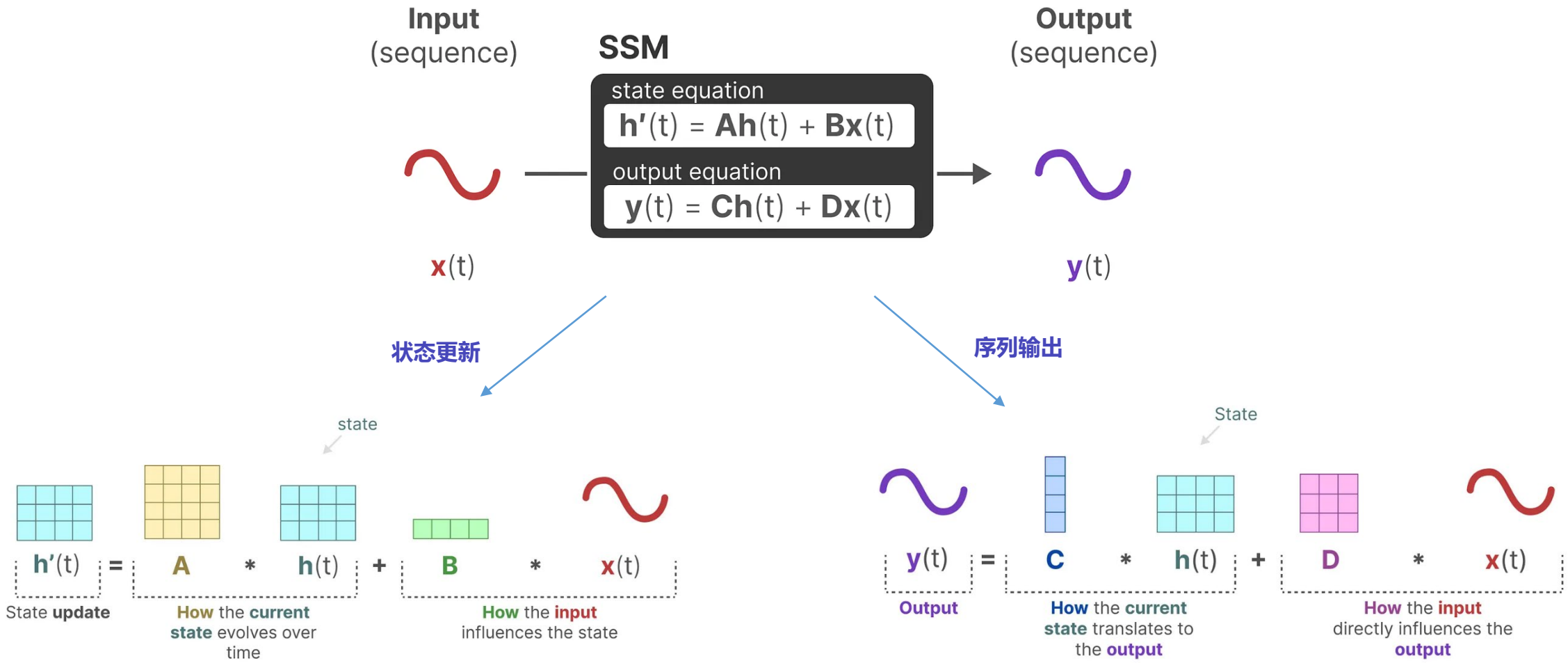
- 当前状态
- 未来可能状态
- 可执行动作

数学表示

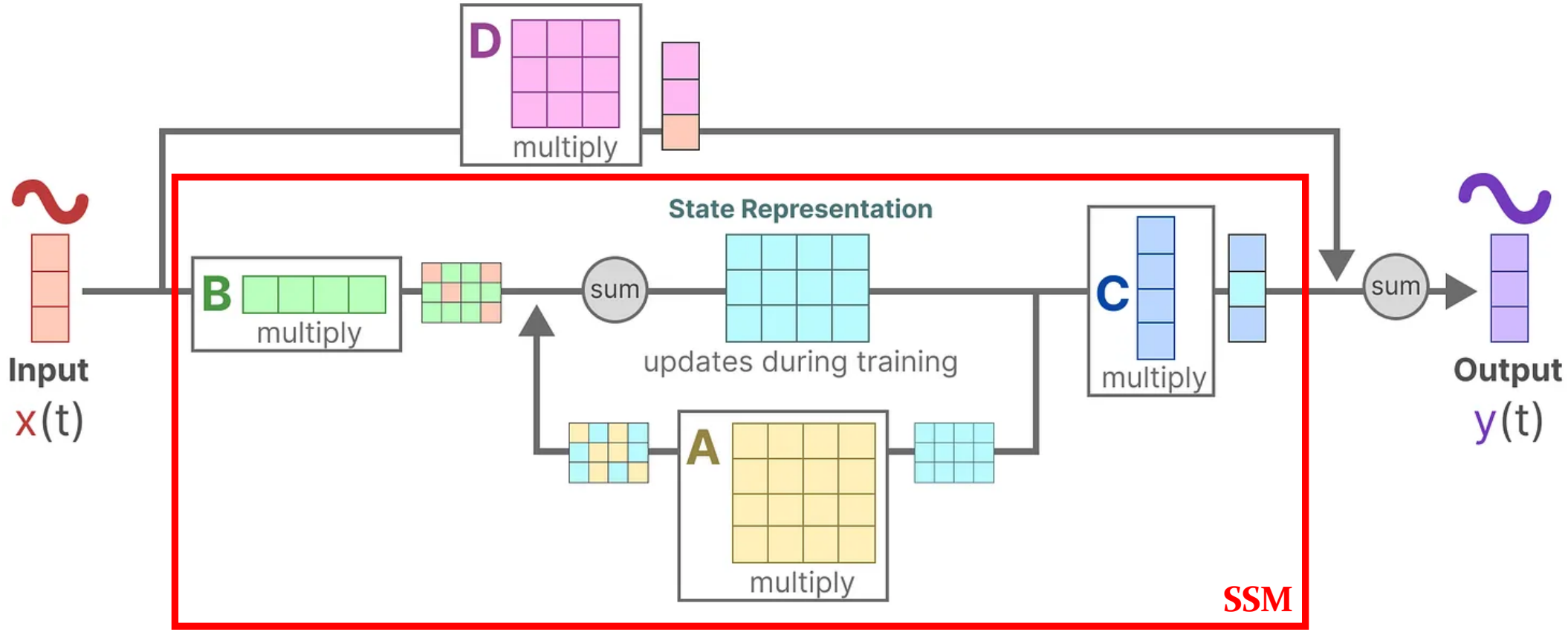
State equation $\mathbf{h'}(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$

Output equation $\mathbf{y}(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$

Mamba – SSM


Input
(sequence)
SSM
state equation
h'(t) = Ah(t) + Bx(t)
output equation
y(t) = Ch(t) + Dx(t)
Output
(sequence)
x(t)
y(t)
状态更新
序列输出
state
h'(t) = A * h(t) + B * x(t)
State update
How the current state evolves over time
How the input influences the state
State
y(t) = C * h(t) + D * x(t)
Output
How the current state translates to the output
How the input directly influences the output

Mamba - SSM


D
multiply
State Representation
B
multiply
sum
updates during training
C
multiply
sum
Input
x(t)
A
multiply
Output
y(t)
SSM

从SSM理论到S_4模型

SSM常用来观测连续信号的输入输出，但是在针对离散数据的训练上SSM也能学习到底层蕴含的连续信息

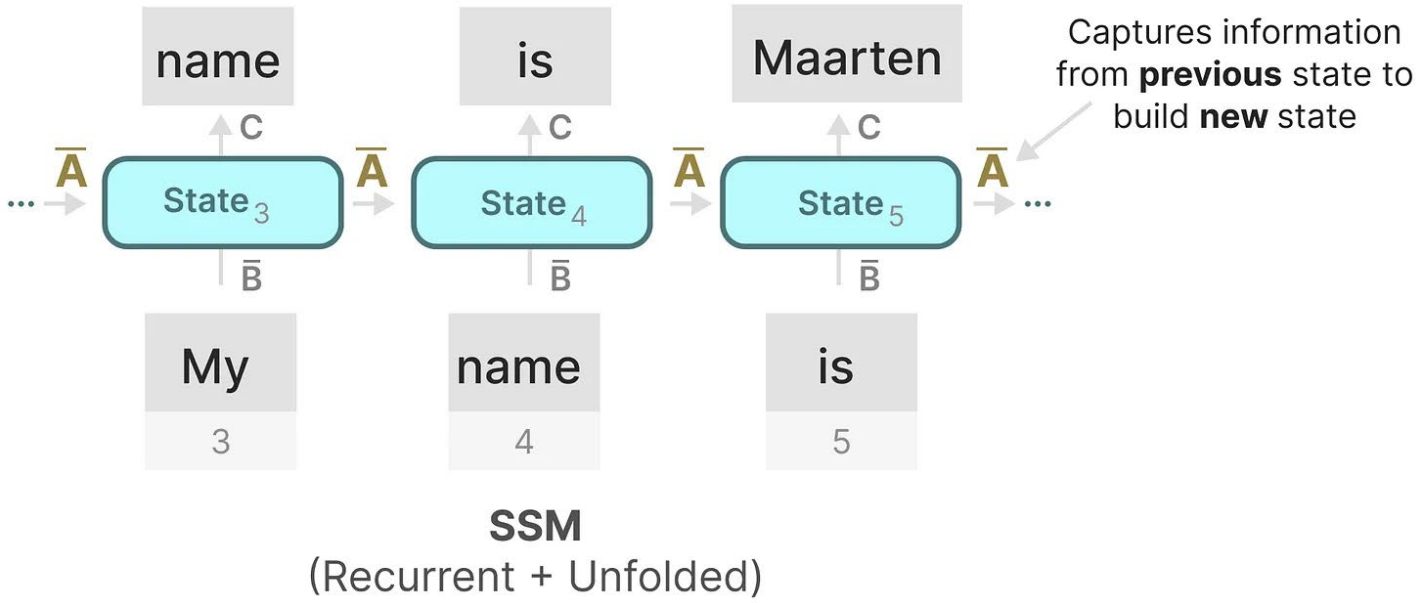
S_4模型（Structured State Space Sequence Models）是第一个基于状态空间表示的模型，其本身是对SSM做了结构化处理,使得参数矩阵具有特定的结构，从而在初始化层面上能简化计算、提高参数学习效率

从SSM到S_4大致包含三部分内容

1. **基于Hippo处理长序列** --- 长距离依赖问题
2. **离散化SSM** --- 离散数据的处理 和 卷积训练的实现
3. **循环/卷积表示** --- 快速训练和推理的转换

Mamba – S4

HiPPO: Recurrent Memory with Optimal Polynomial Projections -- *NeurIPS*



Produces hidden state

$$h_k = \bar{A}h_{k-1} + \bar{B}x_k$$

$$y_k = Ch_k$$

由于矩阵A只记住之前的几个token和捕获迄今为止看到的每个token之间的区别，特别是在循环表示的上下文中，因为它只回顾以前的状态 --- 如何保留以保留比较长Memory的方式来创建矩阵？ --- Hippo

Mamba

HiPPO: Recurrent Memory with Optimal Polynomial Projections -- *NeurIPS*

HiPPO Matrix A_{nk}

$$\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{everything below the diagonal} \\ n+1 & \text{the diagonal} \\ 0 & \text{everything above the diagonal} \end{cases}$$

HiPPO Matrix

1	0	0	0
1	2	0	0
1	3	3	0
1	3	5	4

k (rows), n (columns)

它使用矩阵构建一个“可以很好地捕获最近的token并衰减旧的token”状态表示(to build a state representation that captures recent tokens well and decays older tokens)， 通过函数逼近产生状态矩阵 A 的最优解

正由于HiPPO 矩阵可以产生一个隐藏状态来记住其历史(从数学上讲，它是通过跟踪Legendre polynomial的系数来实现的，这使得它能够逼近所有以前的历史)，使得在被应用于循环表示和卷积表示中时，可以处理远程依赖性

Mamba – S4

HiPPO: Recurrent Memory with Optimal Polynomial Projections -- *NeurIPS*

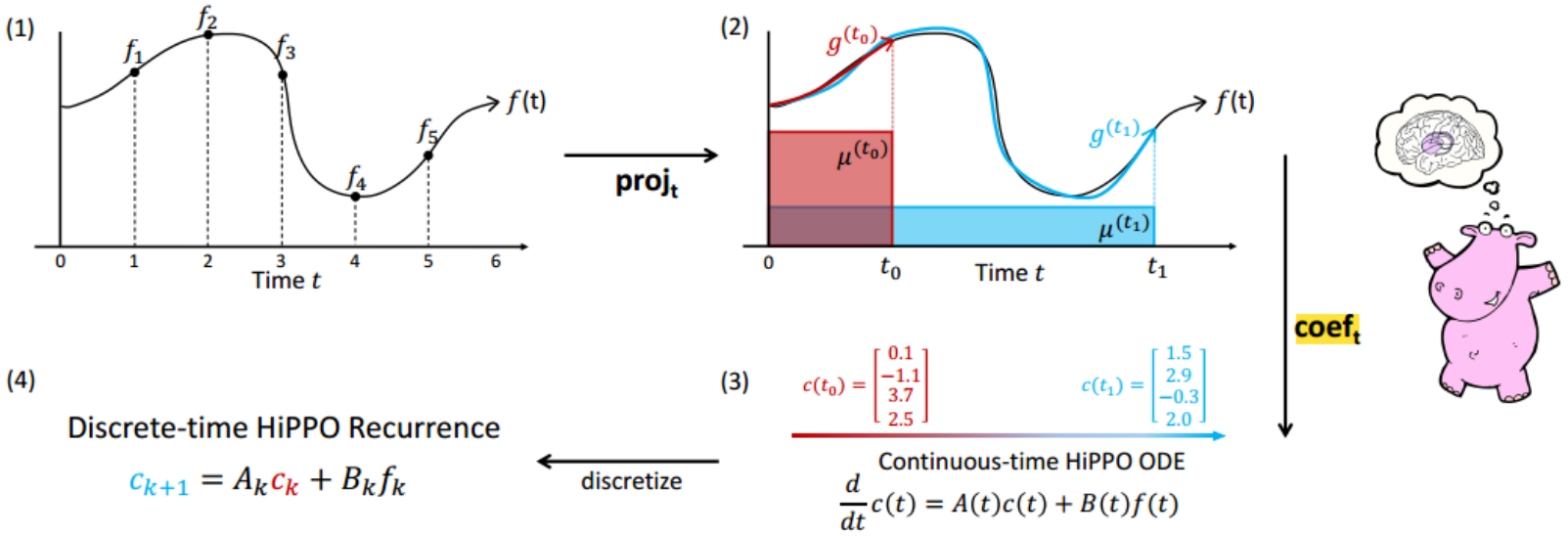


Figure 1: **Illustration of the HiPPO framework.** (1) For any function f, (2) at every time t there is an optimal projection $g^{(t)}$ of f onto the space of polynomials, with respect to a measure $\mu^{(t)}$ weighing the past. (3) For an appropriately chosen basis, the corresponding coefficients $c(t) \in \mathbb{R}^N$ representing a compression of the history of f satisfy linear dynamics. (4) Discretizing the dynamics yields an efficient closed-form recurrence for online compression of time series $(f_k)_{k \in \mathbb{N}}$.

Mamba – S4

Discrete-time SSM: The Recurrent Representation

有4个参数(Δ, A, B, C)

$$h'(t) = Ah(t) + Bx(t) \quad (1a)$$
$$y(t) = Ch(t) \quad (1b)$$

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t \quad (2a)$$
$$y_t = Ch_t \quad (2b)$$

$$\overline{K} = (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^k\overline{B}, \dots) \quad (3a)$$
$$y = x * \overline{K} \quad (3b)$$

针对离散化的输入（如文本序列）代替连续函数的情况，我们引入输入分辨率的步长Δ，来对SSM连续模型进行离散化处理

$$\overline{A} = \exp(\Delta A) \qquad \overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$$

"During training, the continuous representation is discretized"
只是在训练过程中A才被离散化，保存A时还是采用连续形式

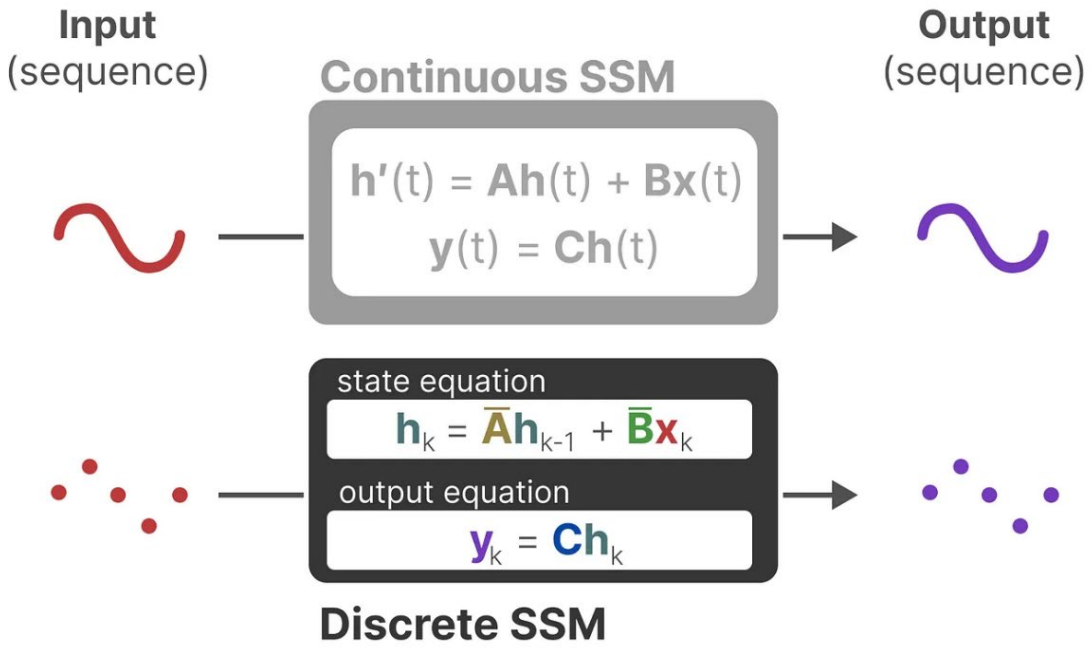
Mamba – S4

Discrete-time SSM: The Recurrent Representation

$$h'(t) = Ah(t) + Bx(t) \quad (1a)$$
$$y(t) = Ch(t) \quad (1b)$$

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t \quad (2a)$$
$$y_t = Ch_t \quad (2b)$$

$$\overline{K} = (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^k\overline{B}, \dots) \quad (3a)$$
$$y = x * \overline{K} \quad (3b)$$

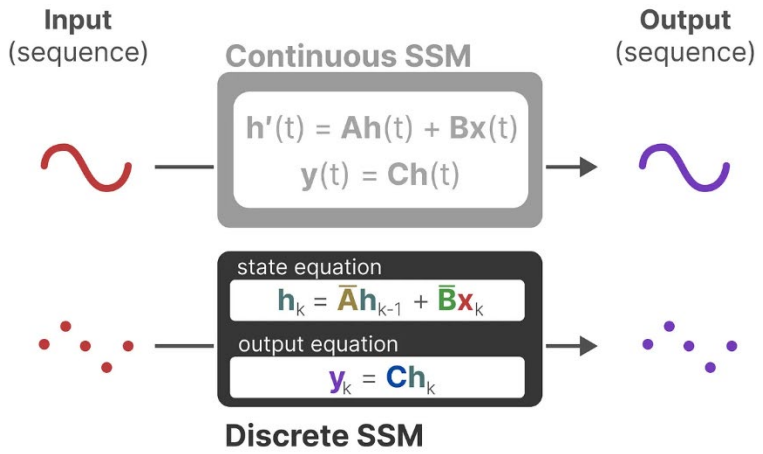


经过离散化处理后，我们将原来函数到函数的映射关系转换为序列到序列的映射关系，由此可以对离散化的数据进行训练与推理

此外，经过离散化的状态方程变为关于Xk的递归式，从而允许了离散SSM像RNN一样递归计算

Mamba – S4

Discrete-time SSM: The Recurrent Representation



离散的SSM允许我们利用时间步长重新描述问题

Timestep 0

$$h_0 = \bar{B}x_0$$

$$y_0 = Ch_0$$

Timestep -1 does not exist so Ah_{-1} can be ignored

Timestep 1

$$h_1 = \bar{A}h_0 + \bar{B}x_1$$

$$y_1 = Ch_1$$

State of **previous** timestep

State of **current** timestep

Timestep 2

$$h_2 = \bar{A}h_1 + \bar{B}x_2$$

$$y_2 = Ch_2$$

State of **previous** timestep

State of **current** timestep

Mamba – S4

Timestep 0

$h_0 = \bar{B}x_0$

$y_0 = Ch_0$

Timestep -1 does not exist so $\bar{A}h_{-1}$ can be ignored

Timestep 1

$h_1 = \bar{A}h_0 + \bar{B}x_1$

$y_1 = Ch_1$

State of **previous** timestep

State of **current** timestep

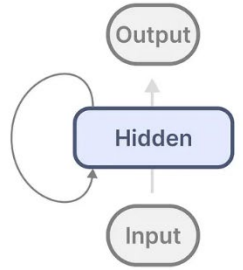
Timestep 2

$h_2 = \bar{A}h_1 + \bar{B}x_2$

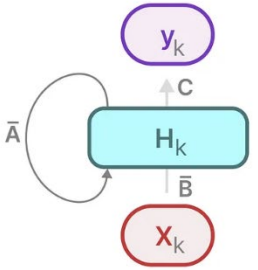
$y_2 = Ch_2$

State of **previous** timestep

State of **current** timestep

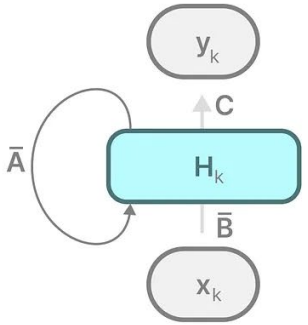


RNN

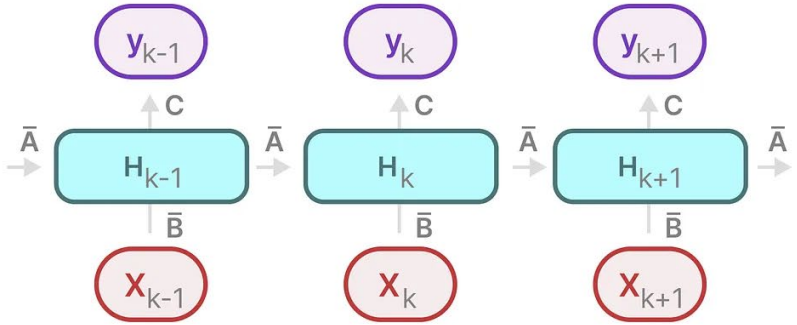


SSM (Recurrent)

$$
\begin{aligned}
y_2 &= Ch_2 \\
&= C\left(\bar{A}h_1 + \bar{B}x_2\right) \\
&= C\left(\bar{A}\left(\bar{A}h_0 + \bar{B}x_1\right) + \bar{B}x_2\right) \\
&= C\left(\bar{A}\left(\bar{A}\cdot\bar{B}x_0 + \bar{B}x_1\right) + \bar{B}x_2\right) \\
&= C\left(\bar{A}\cdot\bar{A}\cdot\bar{B}x_0 + \bar{A}\cdot\bar{B}x_1 + \bar{B}x_2\right) \\
&= C\cdot\bar{A}^2\cdot\bar{B}x_0 + C\cdot\bar{A}\cdot\bar{B}\cdot x_1 + C\cdot\bar{B}x_2
\end{aligned}
$$



SSM (Recurrent)



SSM (Recurrent + Unfolded)

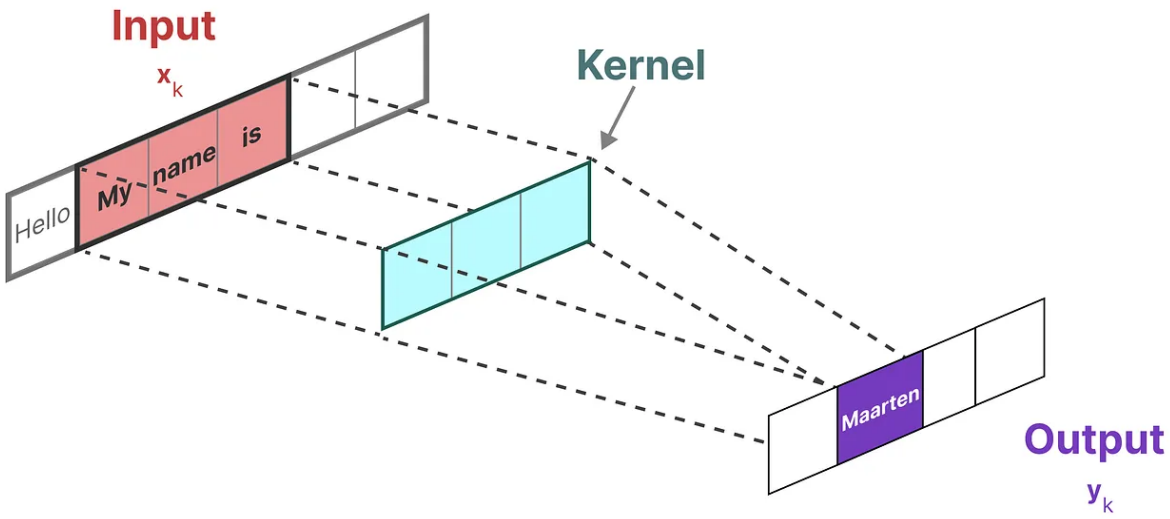
Mamba

Discrete-time SSM: The Recurrent Representation

$$h'(t) = Ah(t) + Bx(t) \quad (1a)$$
$$y(t) = Ch(t) \quad (1b)$$

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t \quad (2a)$$
$$y_t = Ch_t \quad (2b)$$

$$\overline{K} = (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^k\overline{B}, \dots) \quad (3a)$$
$$y = x * \overline{K} \quad (3b)$$



$$\text{kernel} \longrightarrow \overline{K} = (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^k\overline{B}, \dots)$$

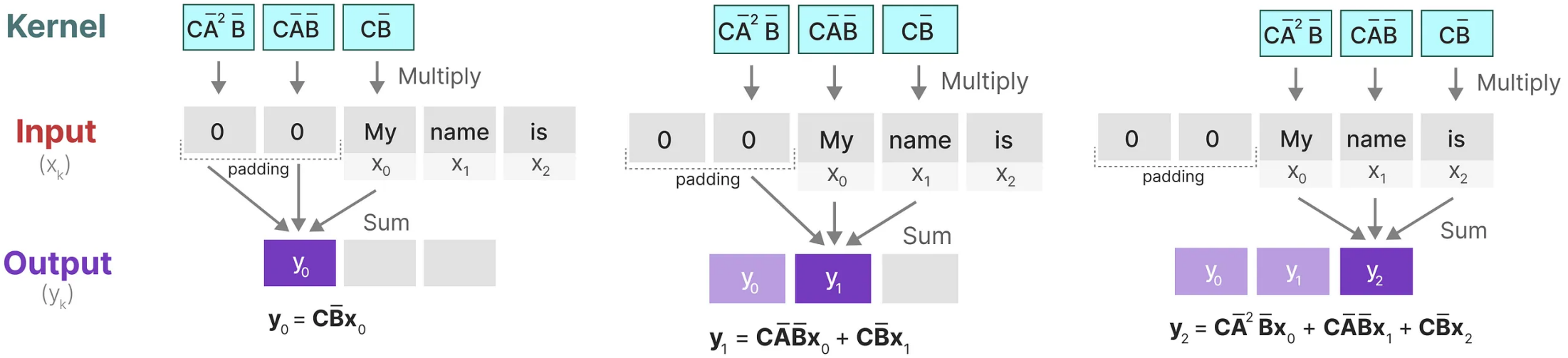
$$y = x * \overline{K}$$

output input kernel

Discrete-time SSM: The Recurrent Representation

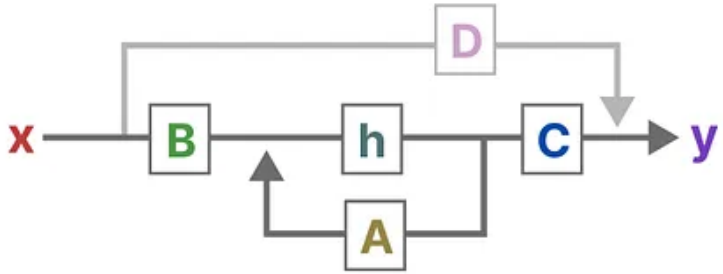
由此我们可以设计卷积核，超高速的并行的去计算y值，从而实现训练的快速性

$$
\begin{aligned}
y_2 &= Ch_2 \\
&= C\left(\bar{A}h_1 + \bar{B}x_2\right) \\
&= C\left(\bar{A}\left(\bar{A}h_0 + \bar{B}x_1\right) + \bar{B}x_2\right) \\
&= C\left(\bar{A}\left(\bar{A}\cdot\bar{B}x_0 + \bar{B}x_1\right) + \bar{B}x_2\right) \\
&= C\left(\bar{A}\cdot\bar{A}\cdot\bar{B}x_0 + \bar{A}\cdot\bar{B}x_1 + \bar{B}x_2\right) \\
&= C\cdot\bar{A}^2\cdot\bar{B}x_0 + C\cdot\bar{A}\cdot\bar{B}\cdot x_1 + C\cdot\bar{B}x_2
\end{aligned}
$$



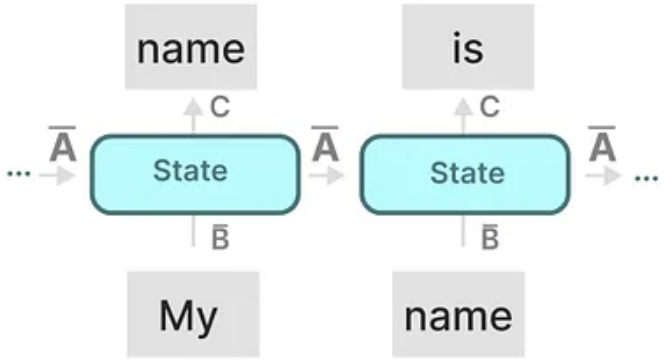
Mamba

Continuous-time



Discretize

Recurrent

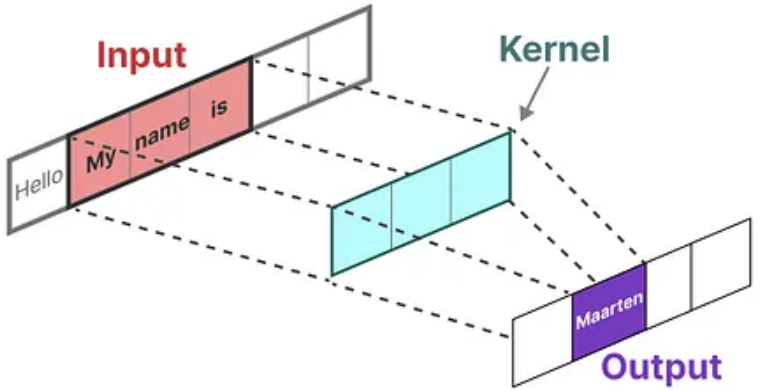


✔ *efficient inference*

✘ *parallelizable training*

or

Convolutional



✘ *unbounded context*

✔ *parallelizable training*

SSM模型（S_4）的问题

1. 系数矩阵A B C 是线性时不变（LTI）的，不随输入变化而变化，这意味着针对每一个token，矩阵A B C都是相同的，无法做到针对性推理
2. 缺乏选择性，无法根据各个token重要性的不同而选择性聚焦

这两个问题时常是不兼容的，如果想要实现有选择性，就要用RNN模式进行训练；然而RNN训练速度又非常慢，使用CNN快速训练又必须对A B C 矩阵计算卷积核加入训练，无法聚焦

需要找到一种**无需卷积的并行训练方式**

Mamba

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu*[1] and Tri Dao*[2]

[1]Machine Learning Department, Carnegie Mellon University
[2]Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

Abstract

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several improvements. First, simply letting the SSM parameters be functions of the input addresses their weakness with discrete modalities, allowing the model to selectively propagate or forget information along the sequence length dimension depending on the current token. Second, even though this change prevents the use of efficient convolutions, we design a hardware-aware parallel algorithm in recurrent mode. We integrate these selective SSMs into a simplified end-to-end neural network architecture without attention or even MLP blocks (Mamba). Mamba enjoys fast inference (5× higher throughput than Transformers) and linear scaling in sequence length, and its performance improves on real data up to million-length sequences. As a general sequence model backbone, Mamba achieves state-of-the-art performance across several modalities such as language, audio, and genomics. On language modeling, our Mamba-3B model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream evaluation.

S6设计

硬件感知和并行

深度框架

Mamba – S6架构

3.1 Motivation: Selection as a Means of Compression

We argue that a fundamental problem of sequence modeling is compressing context into a smaller state. In fact, we can view the tradeoffs of popular sequence models from this point of view. For example, attention is both effective and inefficient because it explicitly does not compress context at all. This can be seen from the fact that autoregressive inference requires explicitly storing the entire context (i.e. the KV cache), which directly causes the slow linear-time inference and quadratic-time training of Transformers. On the other hand, recurrent models are efficient because they have a finite state, implying constant-time inference and linear-time training. However, their effectiveness is limited by how well this state has compressed the context.

作者认为序列建模的一个基本问题就是将上下文压缩成更小的状态

从这个角度来看：

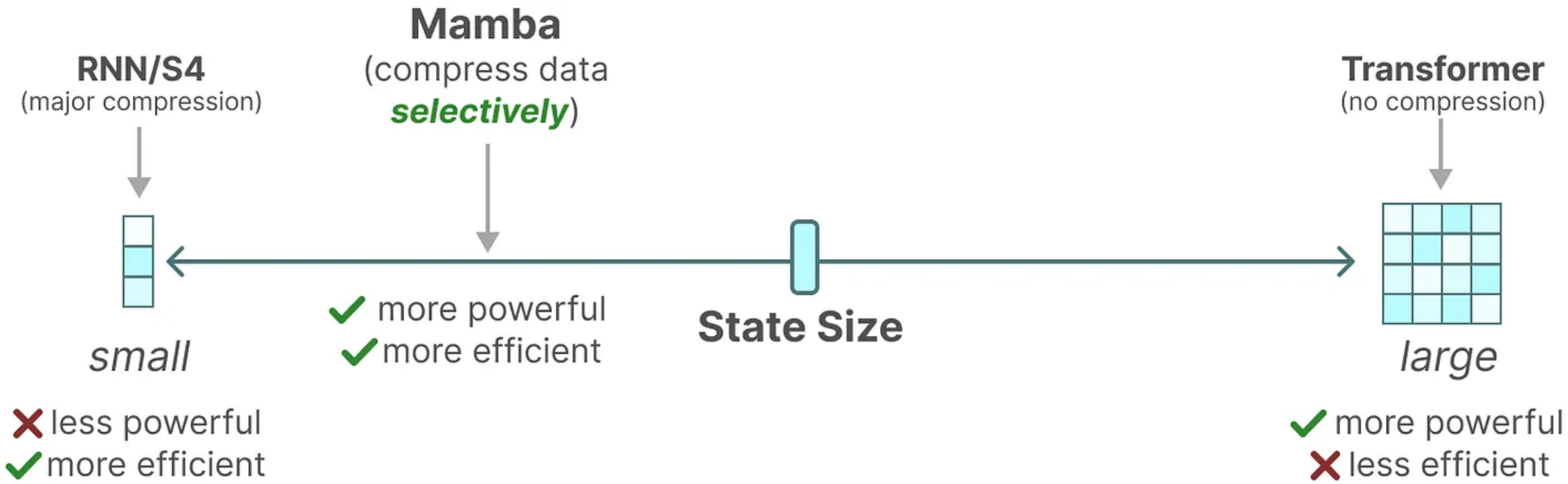
Attention机制有效果但是不高效，因为它不压缩上下文（显式存储整个上下文，KV catch），这直接导致了transformer的线性时间推理和二次时间训练缓慢

RNN是有效的，有一个有限状态，可以实现常数时间推理和线性时间训练，但是这种有效性受上下文压缩程度的限制，序列过长则可能导致遗忘

Mamba – S6架构

总之，序列模型的效率与效果的权衡点在于它们对状态的压缩程度：

- **高效的模型必须有一个小的状态(比如RNN或S_4)**
- **而有效的模型必须有一个包含来自上下文的所有必要信息的状态(比如transformer)**



Mamba – S6 架构

在Mamaba中，作者让 B 矩阵、C 矩阵、Δ 成为输入的函数，让模型能够根据输入内容自适应地调整其行为

较小的步长Δ会忽略特定单词，而更多地使用先前的上文，而较大的步长Δ会更多地关注输入单词而不是上文

B 和 C 矩阵则分别作用于X(t)输入 和 Y(t)输出，设计B、C可以控制x进入h的程度，以及从h输出到y的程度

Algorithm 1 SSM (S4)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
1: $\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter
3: $\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter
4: $\Delta : (\mathtt{D}) \leftarrow \tau_\Delta$(Parameter)
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{D}, \mathtt{N}) \leftarrow$ discretize$(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \mathrm{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
7: **return** y

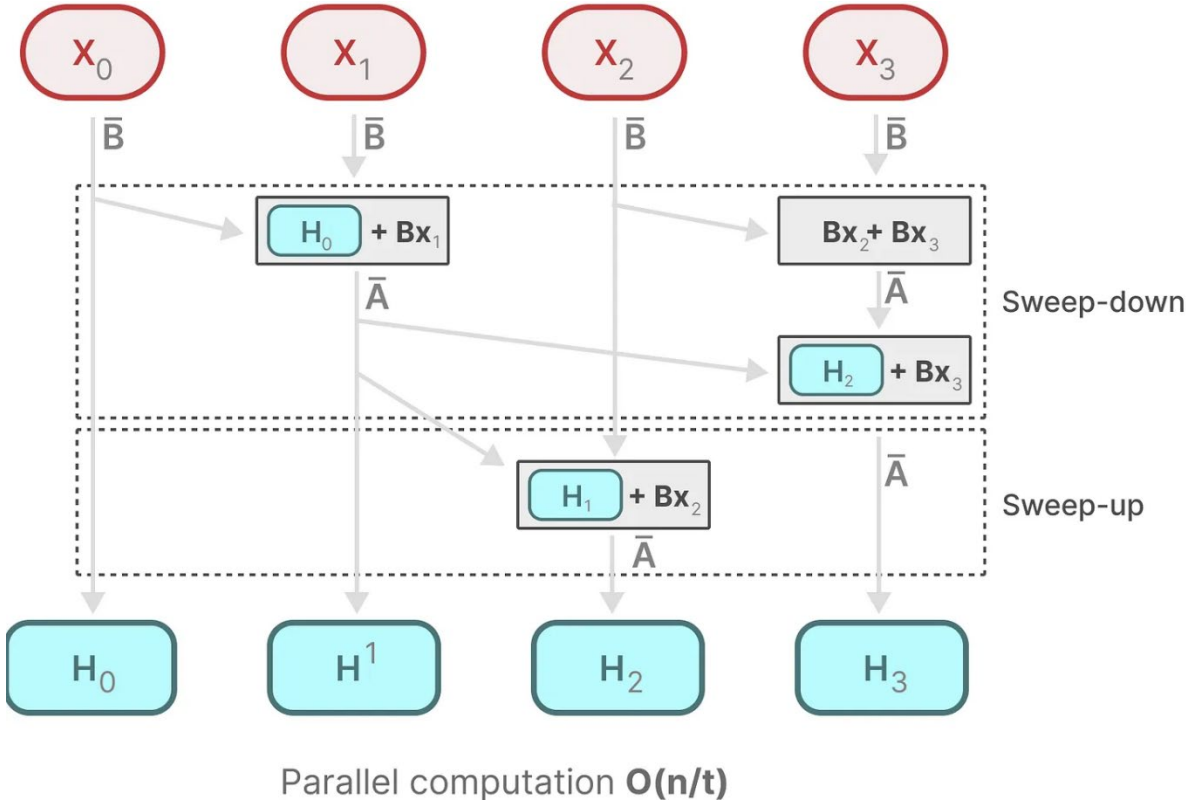
Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
1: $\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_B(x)$
3: $\boldsymbol{C} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_C(x)$
4: $\Delta : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \tau_\Delta$(Parameter$+s_\Delta(x)$)
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N}) \leftarrow$ discretize$(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \mathrm{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-varying: recurrence (*scan*) only
7: **return** y

因此，通过在S_4模型上加入Select设计，原有的时不变矩阵B，C 和 离散步长Δ 可以随输入的变化而自行调整，从而可以对输入的token做自适应，从而实现针对性推理和选择性聚焦

Mamba – 硬件感知和并行

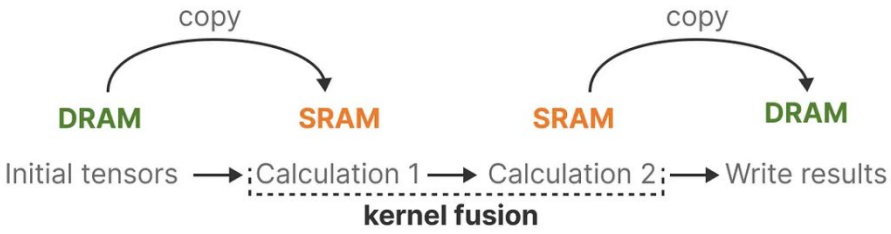
并行扫描(parallel scan)且借鉴Flash Attention



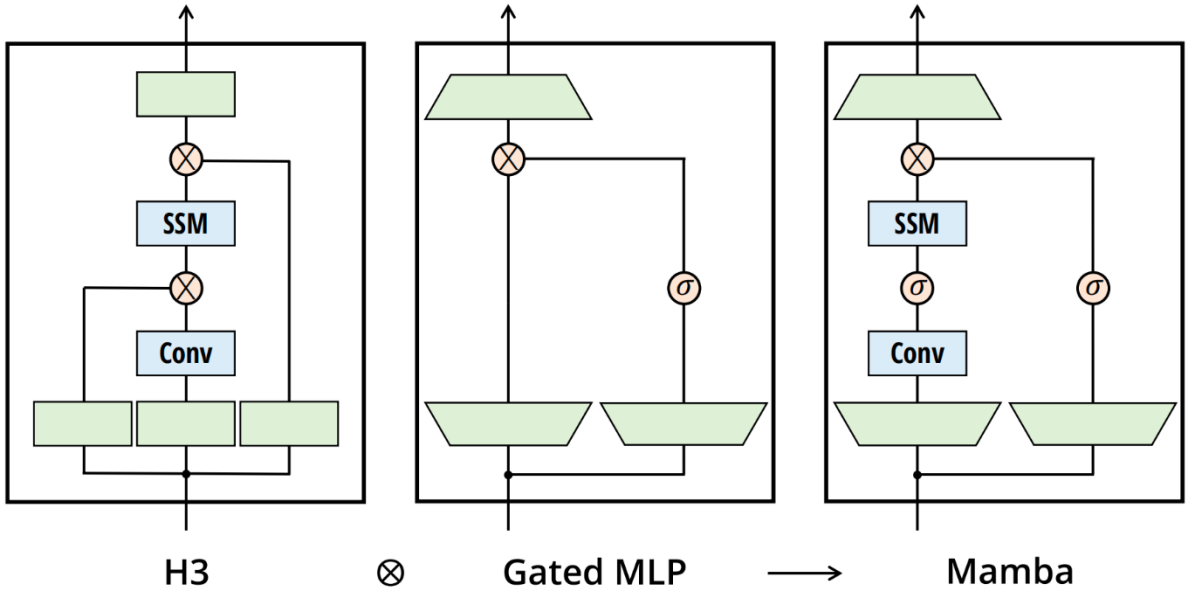
我们可以分段计算序列并迭代地组合它们,即动态矩阵B和C以及并行扫描算法,一起创建选择性扫描算法

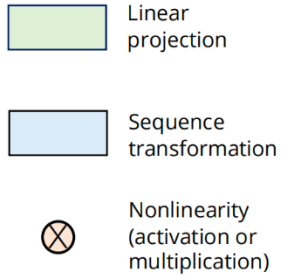
此外，为了让传统的SSM在现代GPU上也能高效计算，Mamba中也使用了Flash Attention技术

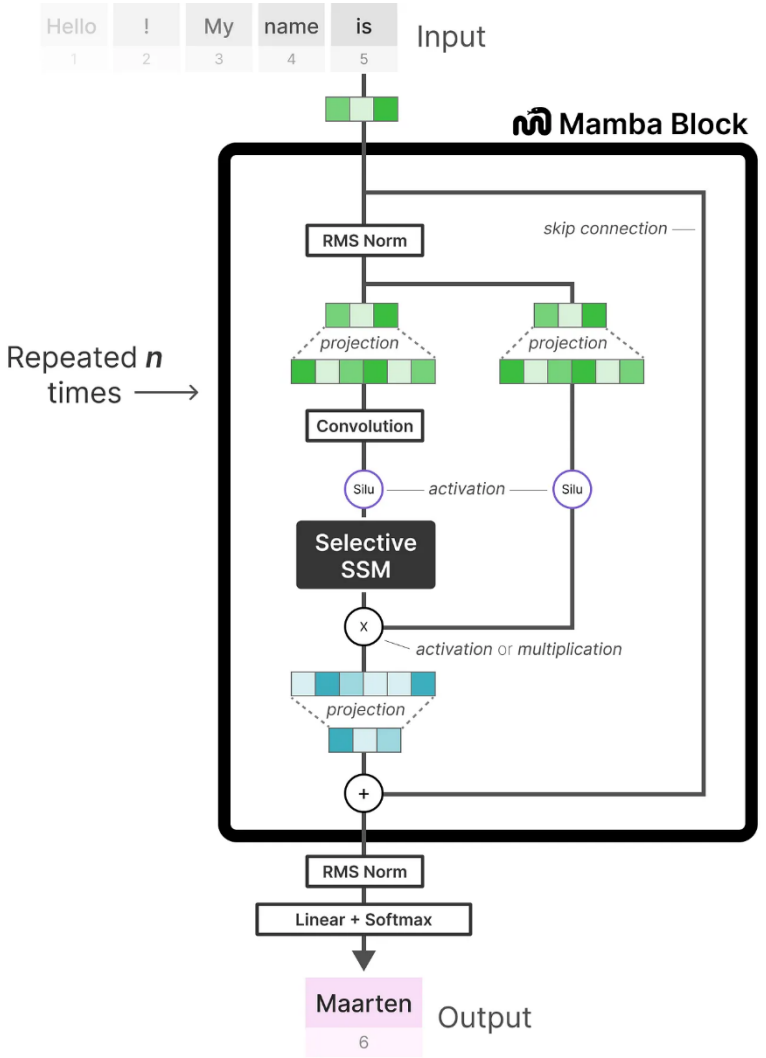
简而言之，利用内存的不同层级结构处理SSM的状态，减少高带宽但慢速的HBM内存反复读写这个瓶颈具体而言，就是限制需要从 DRAM 到 SRAM 的次数(通过内核融合kernel fusion来实现)，避免一有结果便从SRAM写入到DRAM，而是待SRAM中有一批结果再集中写入DRAM中，从而降低来回读写的次数



Mamba – 深度框架







将大多数SSM架构比如H_3的基础块，与现代神经网络比如transformer中普遍存在的门控MLP相结合，组成新的Mamba块，重复这个块，与归一化和残差连接结合，便构成了Mamba架构

Mamba

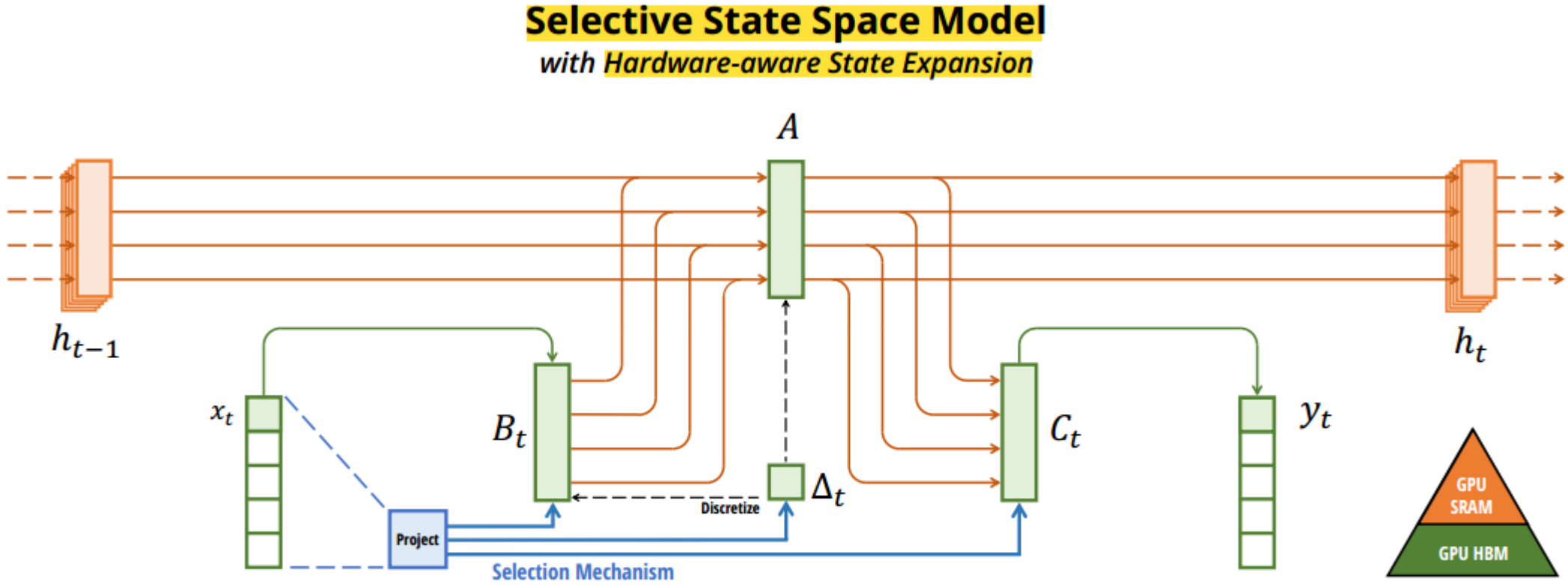


Figure 1: (**Overview.**) Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN, times batch size B and sequence length L) through clever alternate computation paths requiring time-invariance: the $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ parameters are constant across time. Our selection mechanism adds back input-dependent dynamics, which also requires a careful hardware-aware algorithm to only materialize the expanded states in more efficient levels of the GPU memory hierarchy.

Mamba

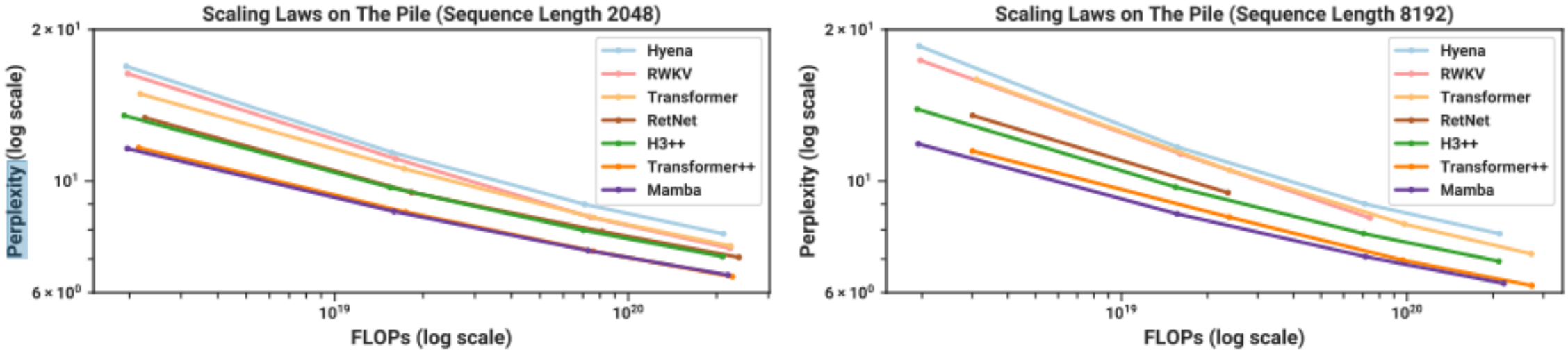


Figure 4: (**Scaling Laws.**) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence length grows.

Mamba

Model	Arch.	Layer	Acc.
S4	No gate	S4	18.3
-	No gate	S6	**97.0**
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	**99.7**
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	**99.8**

Table 1: (**Selective Copying.**) Accuracy for combinations of architectures and inner sequence layers.

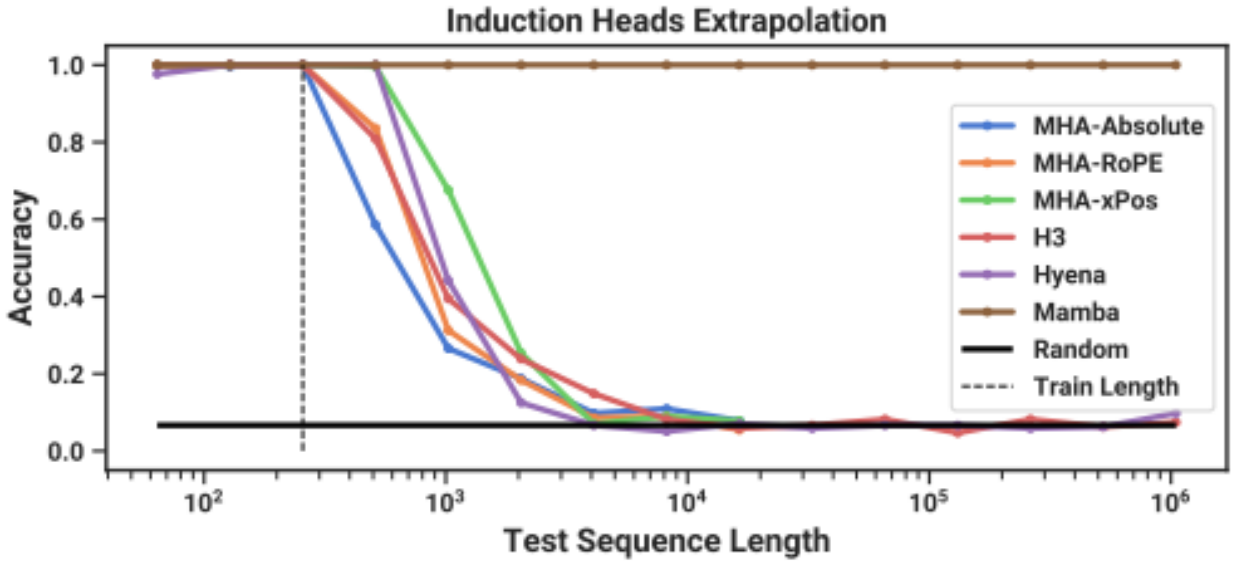


Table 2: (**Induction Heads.**) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Mamba

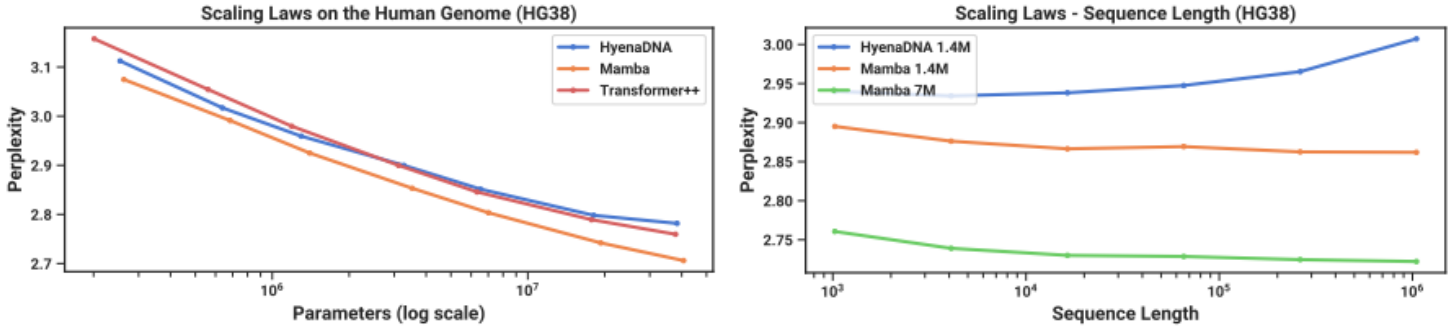


Figure 5: (**DNA Scaling Laws.**) Pretraining on the HG38 (human genome) dataset. (*Left*) Fixing short context length $2^{10} = 1024$ and increasing size from $\approx 200K$ to $\approx 40M$ parameters, Mamba scales better than baselines. (*Right*) Fixing model size and increasing sequence lengths while keeping tokens/batch and total training tokens fixed. Unlike baselines, the selection mechanism of Mamba facilitates better performance with increasing context length.

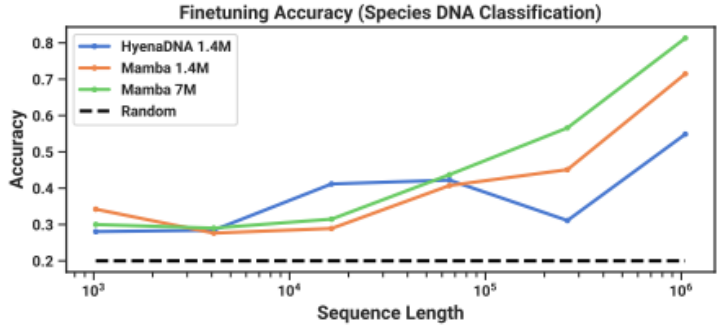


Figure 6: (**Great Apes DNA Classification.**) Accuracy after fine-tuning on sequences of length $2^{10} = 1024$ up to $2^{20} = 1048576$ using pretrained models of the same context length. Numerical results in Table 13.

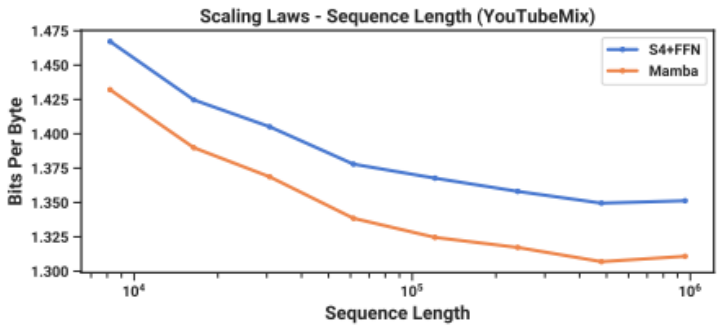


Figure 7: (**Audio Pretraining.**) Mamba improves performance over prior state-of-the-art (Sashimi) in autoregressive audio modeling, while improving up to minute-long context or million-length sequences (controlling for computation).

Mamba 被ICLR拒收

1. 缺少 LRA（Long Range Arena）的结果：审稿人强调缺少 LRA 的结果，而 LRA 是公认的长序列建模基准。在之前的状态空间模型研究中，LRA 已成为惯例，因此必须对其进行全面评估。
2. 使用困惑度进行评估：审稿人质疑将困惑度作为主要评价指标的做法。论文引用了 Sun et al. (2021)（《Do Long-Range Language Models Actually Use Long-Range Context?》）的观点，他们认为较低的困惑度并不一定意味着最终 NLP 应用的建模能力有所提高。Zhang et al. (2023)（《Efficient Long-Range Transformers: You Need to Attend More, but Not Necessarily at Every Layer》）进一步加强了他们的观点，他们强调了一些 transformer 模型的局限性，这些模型虽然实现了较低的困惑度，但在生成任务（如摘要和问题解答）中却举步维艰。

Q&A

Thanks!