State Space Duality (Mamba-2)

Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality

Tri Dao, Albert Gu

Department of Computer Science, Princeton University
Machine Learning Department, Carnegie Mellon University

Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering & Artificial Intelligence

References

Mamba-2

- Github : https://github.com/state-spaces/mamba
- Hugging Face : https://huggingface.co/docs/transformers/main/model_doc/mamba2
- Paper : https://arxiv.org/abs/2405.21060

  ICML'24: Proceedings of the 41st International Conference on Machine Learning, Article No.: 399, Pages 10041 - 10071

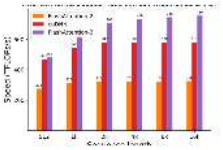
Tri Dao

- https://tridao.me/blog/2024/mamba2-part1-model/
- https://goombalab.github.io/blog/2024/mamba2-part1-model/
- https://pli.princeton.edu/blog/2024/mamba-2-algorithms-and-systems

FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision

12 min read · July 11, 2024

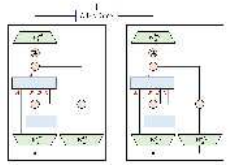
2024



State Space Duality (Mamba-2) Part IV - The Systems

6 min read · May 31, 2024

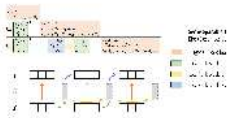
2024



State Space Duality (Mamba-2) Part III - The Algorithm

16 min read · May 31, 2024

2024



State Space Duality (Mamba-2) Part II - The Theory

19 min read · May 31, 2024

2024

State Space Duality (Mamba-2) Part I - The Model

18 min read · May 31, 2024

2024

State Space Duality (Mamba-2) Part I - The Model

1. Mamba-1

- **Problem (Understanding)**

- From a ***conceptual*** standpoint, one of the reasons we found **SSMs** so fascinating is how they **just feel fundamental.**
  - ✓ Example; **structured SSMs** seem to capture the essence of *continuous*, *convolutional*, and *recurrent* sequence models
- **SSMs** always felt somewhat *disjoint* from **attention**, and we've tried for a while to understand their relationship better.

Question 1: ***What are the conceptual connections between state space models and attention?*** *Can we combine them?*

- **Problem 2 (Efficiency)**

- From a ***computational*** standpoint, despite its *hardware-aware selective scan* implementation, it's still ***much less hardware-efficient than mechanisms such as attention.***
- ***Modern accelerators*** such as GPUs and TPUs are highly specialized for ***matrix multiplications***.

Question 2: ***Can we speed up the training of Mamba models by recasting them as matrix multiplications?***

State Space Duality (Mamba-2) Part I - The Model

2. The SSD Model

- ***Main point of the Mamba-2*** is what we call ***structured state space duality (SSD)***, which refers to several things:

- **SSD model** : *A specific standalone layer*, like attention or an SSM, that can be incorporated into deep neural networks
- **SSD framework** : *A general framework for reasoning* about this model (and many more theoretical connections)
- **SSD algorithm** : An algorithm *for computing SSD layers* much more efficiently than previous SSMs

❖ The main SSD model or “state space dual model” itself really isn’t so complicated!

State Space Duality (Mamba-2) Part I - The Model

➢ The Linear (SSM) Mode

- Structured state space model (SSM) defines a map from $x_t \in \mathbb{R}^T \rightarrow y_t \in \mathbb{R}^T$. x_t, y_t as being scalars, the hidden state h_t as an N-dimensional vector, where *N* is an independent hyperparameter called the *state size*, *state dimension*, or *state expansion factor*.

$$h_t = A_t h_{t-1} + B_t x_t$$
$$y_t = C_t^\top h_t \qquad (1)$$

- A selective state space model allows the (A, B, C) SSM parameters to vary across time; $A \in \mathbb{R}^{(T,N,N)}$, $B \in \mathbb{R}^{(T,N)}$, $C \in \mathbb{R}^{(T,N)}$
  - ✓ Structured SSMs require $\boldsymbol{A}$ to have structure to be efficiently computable, such as the most commonly used ***diagonal structure***.
  - ✓ In this case, A has shape (T, N) where only the diagonal elements of the $N \times N$ matrices are stored.
- The original **Mamba** (or more precisely its core "S6" layer) is exactly **a selective SSM** with ***diagonal structure***.

- **SSD: Scalar Structured SSM**
- **The SSD layer of Mamba-2 makes only one small modification**
- It restricts **the diagonal $\boldsymbol{A}$** even further to a ***scalar times identity structure***; The diagonal elements of A must all be the same value.

State Space Duality (Mamba-2) Part I - The Model

➢ The Linear (SSM) Mode

- **Multihead SSMs**

$$h_t = A_t h_{t-1} + B_t x_t$$
$$y_t = C_t^\top h_t$$

- If $X \in \mathbb{R}^{(T,P)}$ has *P* separate channels, we can use the same dynamics (i.e. the same SSM (A,B,C)) independently for each channel. This can be interpreted as *a single head* of the SSM model.

✓ X shape (T,P); Think of ***T*** is the ***sequence (time) dimension*** and ***P*** is the "***head dimension***".

- *Multiple heads* can be constructed completely *independently*; we assume that we're working with a single head.

- [Note that] these heads are exactly analogous to how heads in multi-head attention models work, and in Mamba-2 we choose similar dimensions as modern Transformers, e.g. **P=64** or **P=128.**

- Notate the general (selective) state space model as

$$Y^{(\mathtt{T},\mathtt{P})} = \mathsf{SSM}(A^{(\mathtt{T},\ldots)}, B^{(\mathtt{T},\mathtt{N})}, C^{(\mathtt{T},\mathtt{N})})(X^{(\mathtt{T},\mathtt{P})}) \qquad (2)$$

1. The structure on A, which affects its parameter shape:
   - `... = (N,N)` for general (unstructured) SSMs
   - `... = (N)` for diagonal SSMs (or other structures, such as diagonal-plus-low-rank [17])
   - `... = ()` for scalar SSMs (i.e. SSD)
2. The state dimension $\mathtt{N}$ (i.e. `d_state`)
3. The head dimension $\mathtt{P}$ (i.e. `d_head`)

State Space Duality (Mamba-2) Part I - The Model

➢ The Quadratic (Attention) Mode

- Given the same tensors above with the same shapes; $(A^{(T)}, B^{(T,N)}, C^{(T,N)})$

$$L = \begin{bmatrix} 1 & & & \\ a_1 & 1 & & \\ a_2 a_1 & a_2 & 1 & \\ \vdots & \vdots & \ddots & \ddots \\ a_{T-1} \dots a_1 & a_{T-1} \dots a_2 & \dots & a_{T-1} \quad 1 \end{bmatrix}.$$

- Let's define the following matrix

$$M = L \circ CB^{\top} \in \mathbb{R}^{(T,T)} \qquad (3)$$

- M encodes ***a sequence transformation*** $x_t \in \mathbb{R}^T \to y_t \in \mathbb{R}^T$ mapping a 1D input to a 1D output - just as in equation (1) through basic matrix multiplication; $y = Mx$

❖ Looks very similar to an *attention computation*

- If all a_t=1, then $\boldsymbol{L}$ is simply *the lower-triangular causal mask* and Eq (3) is equivalent to ***causal linear attention***

$$Y = (L \circ QK^{\top})V$$

- Exactly the same as equation (3) if we rename

$$(C, B, X) \mapsto (Q, K, V)!$$

State Space Duality (Mamba-2) Part I - The Model

3. State Space Duality

- "***Duality***" : Two models defined in equations (1) (for the scalar-identity structured A_t case) and (3) are actually exactly the same model

$$(A^{(T)}, B^{(T,N)}, C^{(T,N)}, X^{(T,P)}) \mapsto Y^{(T,P)}$$

- In the general SSD Framework (Part II of this series), we'll show this equivalence in two completely different ways, both of which are actually much more general and each quite illuminating.

❖ Main motivation of restriction is **efficiency**: these changes are necessary to be able to view the model in its [***dual attention form***], which allows matrix multiplications to be used.

➢ SSD vs. State Space Models

- Compared to previous SSMs, **SSD** is much the same as the core layer of Mamba but with even *more structure on the recurrent A matrices.*

Mamba-1 (S6, SSM)	Mamba-2 (SSD)
Diagonal structure on A	*Scalar-times-identity structure* on A
A head dimension of **P=1** (i.e. all channels are completely independently controlled by separate SSMs)	A head dimension of **P>1** (something like P=64 by default).

- By **restricting *the diagonal structure*** of A to ***scalar-times-identity***, the ***recurrence dynamics*** are ***shared*** across **all N elements of the state space** and also *shared* across **all P channels of a given head**.

→ ***A single SSM head*** has **total state size P×N**, which are each *governed by separate scalar recurrences in Mamba-1* but are *controlled by a single shared recurrence in Mamba-2*.

State Space Duality (Mamba-2) Part I - The Model

➢ SSD vs. State Space Models

> THE BOTTOM LINE: MAMBA-1 VS. MAMBA-2
>
> *Compared to Mamba-1, Mamba-2 allows **much larger state dimensions** (from `N=16` in Mamba-1 to `N=64` to `N=256` or even higher in Mamba-2) while simultaneously being **much faster during training**.*

- Compared to previous SSMs, **SSD** is much the same as the core layer of Mamba but with even *more structure on the recurrent A matrices.*
- One of the main reasons for **the selectivity** (e.g. A that depends on the input X) in Mamba is to control whether to remember or ignore particular pieces of information.
- If such information should be ignored, then the entire state can ignore it together, and so it should be okay if the state's dynamics are shared across all features.

❖ From one perspective, Mamba-2 isn't strictly better than Mamba-1: while it's a dramatic improvement from a training perspective, Mamba-1 might be better from a pure inference perspective.

- Since inference speed of SSMs is entirely governed by the state dimension, if one wants to maximize performance for a target inference efficiency (i.e. for a particular state size N), then the increased expressivity of Mamba-1 might be better.

State Space Duality (Mamba-2) Part I - The Model

➢ SSD vs. Attention

- Compared, to **standard (self-)attention**, **SSD** also only has **two differences:**

1. The softmax normalization is dropped.

- Interpreted as what reduces the effective state size of the model from linear to constant, and improves its efficiency from quadratic to linear.

2. A separate elementwise mask matrix is applied multiplicatively.

- It is what distinguishes SSD from standard linear attention.
- One way to think of the **mask** is as ***input-dependent relative positional encodings***.
- Because of the mask L in (3), the ***standard attention score*** $< Q_j, K_j >$ is attenuated by a weight

$$a_{i:j}^{\times} = a_i \cdots a_{j+1}$$

  interpreted as a "*discount factor*" based on how far apart the positions *i* and *j* are.

- In its attention form, this ***input-dependent positional mask*** can be interpreted as **the key factor** that **encodes the "selectivity" of Mamba**.

State Space Duality (Mamba-2) Part I - The Model

4. Best of Both Worlds

➢ Efficiency: the SSM and Attention Modes

- The SSM (1) and attention (3) modes represent two different ways of computing the same function

- SSM Mode
- One of the most SSM interesting things; Computing (1) as a recurrence requires *maintaining a constant-size state* (size N per channel) and *scales linearly in the sequence length* T.
- Downside : the raw FLOPs don't reflect actual speed in practice because of hardware considerations

- Attention Mode
- Computing the sequence transformation $y = Mx$ through equation takes quadratic time in the sequence length, because we're materializing this T × T matrix.
- But it can be fast in practice because it only uses matrix multiplications, which are optimized on **GPUs** and **TPUs**.

State Space Duality (Mamba-2) Part I - The Model

➢ Efficiency: SSD Mode

- During inference; the SSM mode is designed for fast autoregressive inference.
- There's a tension between **FLOPs and hardware efficiency** where the attention mode uses more FLOPs, but uses them more efficiently through matrix multiplications.

- Two equivalent interpretations of this "**state space dual**" algorithm, either as

1) A block decomposition of a particular structured matrix that defines the *SSD "token-mixing" sequence transformation*.
2) A "***chunkwise***" algorithm that *splits the sequence into segments*, *computes the quadratic attention form on each segment*, and *adjusts the result by passing the SSM states between segments*

☞ Explain detail the algorithm to [Part 3]

- Benefits of the SSD algorithm
- Preserves the same efficient FLOP counts as SSMs (compared to quadratic attention)
- Dramatically speeds up training compared to general state space models by utilizing matmuls.

	Attention	**SSM**	**SSD**
State size	$\mathbf{T}$	$\mathbf{N}$	$\mathbf{N}$
Training FLOPs	$\mathbf{T^2N}$	$\mathbf{TN^2}$	$\mathbf{TN^2}$
Inference FLOPs	$\mathbf{TN}$	$\mathbf{N^2}$	$\mathbf{N^2}$
(Naive) memory	$\mathbf{T^2}$	$\mathbf{TN^2}$	$\mathbf{TN}$
Matrix multiplications?	✔	✘	✔

State Space Duality (Mamba-2) Part I - The Model

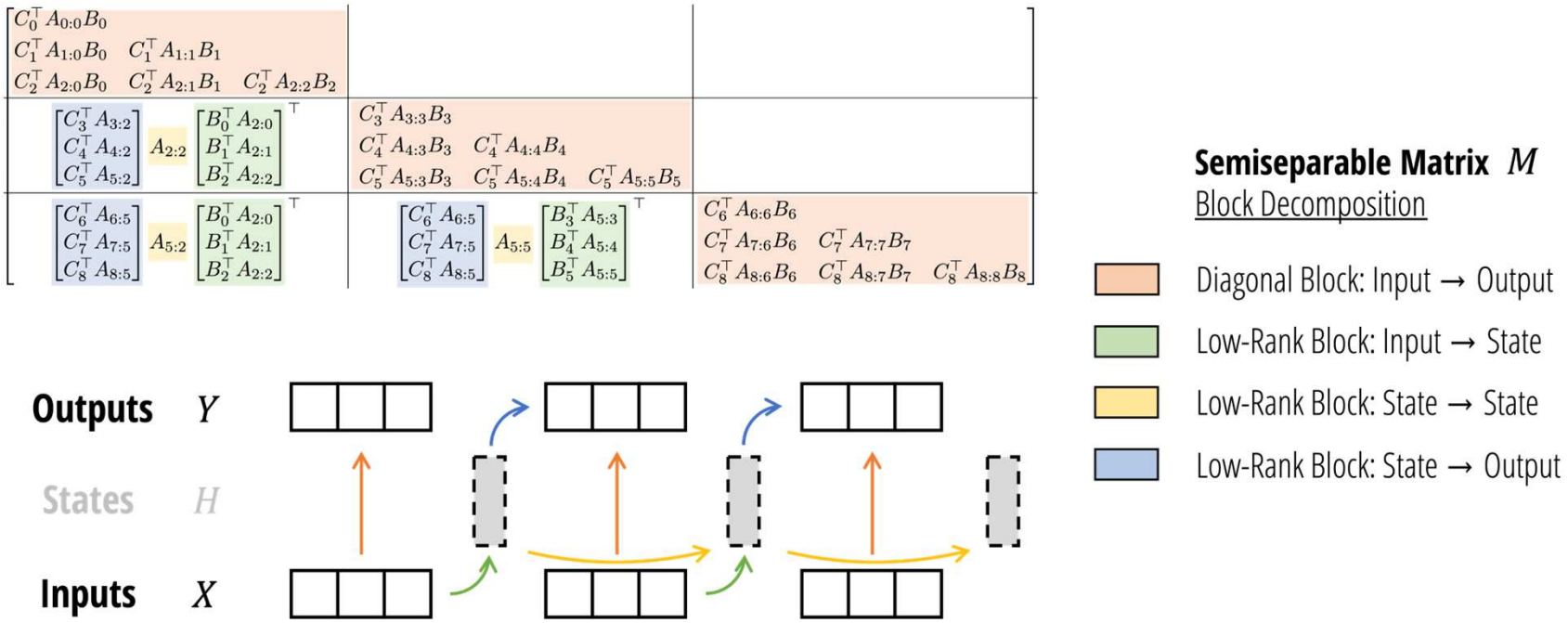


Figure 5. SSD Algorithm; By using the matrix transformation viewpoint of state space models to write them as semiseparable matrices (Section 3), we develop a more hardware-efficient computation of the SSD model through a block decomposition matrix multiplication algorithm. The matrix multiplication also has an interpretation as a state space model, where blocks represent chunking the input and output sequence. Diagonal blocks represent intra-chunk computations and the off-diagonal blocks represent inter-chunk computations, factored through the SSM's hidden state.

State Space Duality (Mamba-2) Part I - The Model

5. Mamba-2 Architecture

- The core contribution of Mamba-2 is the **new SSD layer and theory**. Also, we make some small changes to Mamba's neural network architecture.
- The main change is producing the (A,B,C) SSM parameters in parallel with the input X, instead of sequentially.

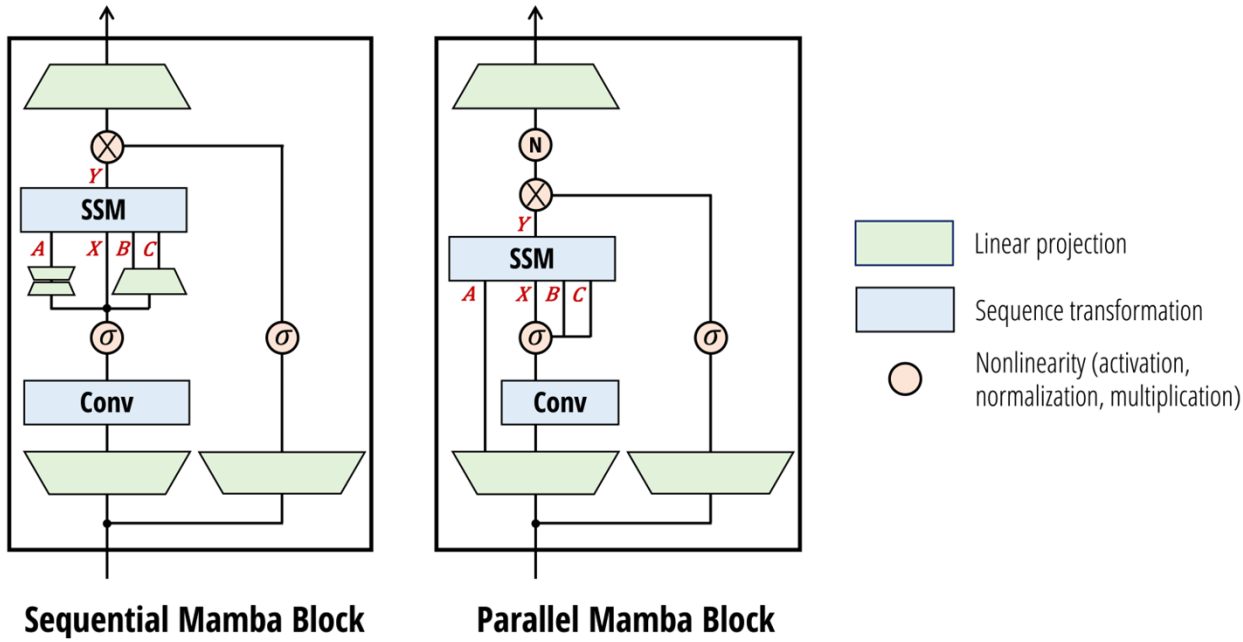


Figure 6: (Mamba-2 Architecture.) The Mamba-2 block **simplifies the Mamba block** by **removing sequential linear projections**; the SSM parameters A,B,C are produced at the beginning of the block instead of as a function of the SSM input X. An **additional normalization layer** is added as in **NormFormer** (Shleifer, Weston, and Ott 2021), improving stability. The B and$C$ projections only have a single head shared across the X heads, analogous to multi-value attention (MVA)

State Space Duality (Mamba-2) Part I - The Model

➢ Language Modeling

- In terms of empirical results, we didn't test Mamba-2 as extensively as Mamba-1 but believe it should generally be on par or better across the board.
- Our full language model results use the same protocol as Mamba and found slightly better scaling at Chinchilla laws.
- Fully trained models on the **Pile dataset** [26] and the **standard zero-shot downstream evaluations** show similar trends. Even when the performance is comparable, Mamba-2 is much faster to train than Mamba-1!

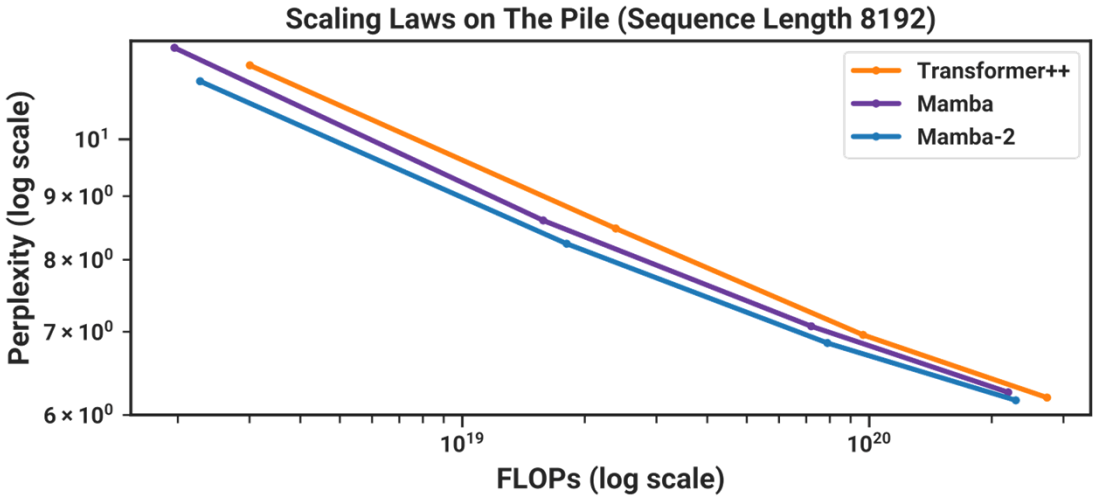


Figure 9: (**Scaling Laws**.) Models of size ≈125M to ≈1.3B parameters, trained on the Pile. Mamba-2 matches or exceeds the performance of Mamba as well as a strong "Transformer++" recipe. Compared to our Transformer baseline, Mamba-2 is Pareto dominant on performance (perplexity), theoretical FLOPs, and actual wall-clock time.

State Space Duality (Mamba-2) Part I - The Model

➢ Synthetic Language Modeling: MQAR

- Highlight the one synthetic task; The **multi-query associative recall** (MQAR) task introduced by the Zoology and Based line of work has become a de facto standard.
- Mamba-2 is substantially better than Mamba-1.
- One reason for the improved performance is **the much larger state size (up to 16× larger than Mamba-1 here)**, which was one of the primary motivations of Mamba-2 in the first place.

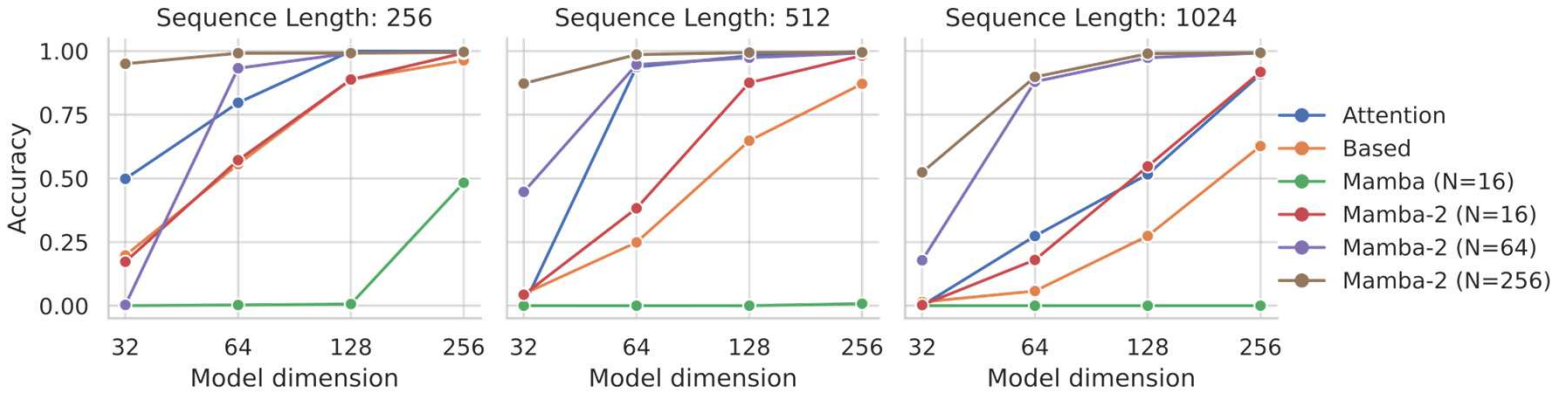


Figure 8: (Multi-Query Associative Recall (MQAR)). Associative recall tasks are challenging for SSMs, which must memorize all relevant information into their recurrent state. The SSD layer combined with improved architecture allows for much larger state sizes in Mamba-2, which performs significantly better than Mamba-1 and even vanilla attention.

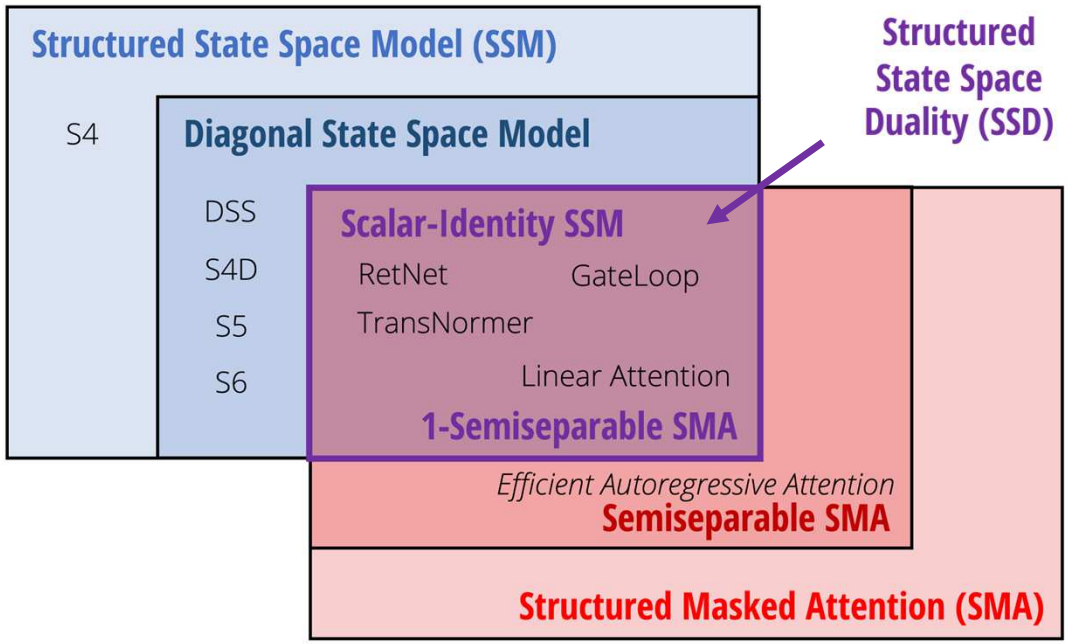
State Space Duality (Mamba-2) Part II - The Theory

❖ Dive into the theory behind SSD model.

- We'll derive the **SSD** "***duality***" in two completely separate way; **SSM perspective** and **attention perspective.**
- The union of these two strong generalizations is what we call the ***SSD framework***.
  - ✓ Provides a rich body of connections between state space models, attention, and structured matrices

❖ *Note that this theory is not necessary to use the SSD model itself; this part of the series can be safely skipped for the practitioner that just wants to use SSD (Mamba-2).*

➢ **The State Space Duality framework**



SSD Framework (red, blue): State space models (i.e. semiseparable matrices) and structured masked attention encapsulate large classes of efficient sequence models. Their intersection is the SSD model (purple).

State Space Duality (Mamba-2) Part II - The Theory

1. Recap: The SSD Model

Part I of this series introduced the SSD layer, which is defined as a selective SSM

$$
\begin{aligned}
h_t &= A_t h_{t-1} + B_t x_t \\
y_t &= C_t^\top y_t
\end{aligned}
$$

$$
\text{(Selective state space model (SSM))} \tag{1}
$$

with scalar-identity structure on A.

More formally, we view it as a *sequence transformation* $X \mapsto Y$

$$
Y^{(\mathtt{T},\mathtt{P})} = \mathsf{SSM}(A^{(\mathtt{T})}, B^{(\mathtt{T},\mathtt{N})}, C^{(\mathtt{T},\mathtt{N})})(X^{(\mathtt{T},\mathtt{P})}) \tag{2}
$$

The dual attention-like form of the SSD layer is

$$
M = L \circ CB^\top \in \mathbb{R}^{(\mathtt{T},\mathtt{T})} \tag{3}
$$

Now let's see how to prove this!

State Space Duality (Mamba-2) Part II - The Theory

2. SSD Framework 1: Structured Matrix Transformations

❖ **The first framing of the *duality*** will be from **an SSM-centric perspective**, where we'll prove the duality through the framework ***of matrix sequence transformations*** or "***matrix mixers***".

➢ Matrix Transformations

- Many sequence models, i.e. *sequence transformations* $X \in \mathbb{R}^{(T,P)} \to Y \in \mathbb{R}^{(T,P)}$, can be written in the form of a single matrix multiplication

$$\boldsymbol{Y = M(X) \cdot X}$$

where M is a matrix which can itself depend on X.

- Many examples distinguished by the structure of the M matrix.
  - ✓ The de facto example is ***self-attention*** itself, where $M = softmax(QK^T)$ is the attention matrix.
  - ✓ Other examples : MLP-Mixer, FNet, and Monarch Mixer.

❖ Call this a ***matrix sequence transformation***, or **matrix transformation** for short. In the literature *sequence transformations* have also been referred to as "**sequence mixers**" or "**token mixers**", and *matrix sequence transformations* as "***matrix mixers***".

Writing a sequence model as a matrix transformation provides a powerful tool to understand the structure and characteristics of the model.

State Space Duality (Mamba-2) Part II - The Theory

➢ Matrix Transformations

❖ Although general non-linear RNNs such as LSTMs cannot be written as matrix mixers, state space models can! (by unrolling the SSM recurrence)

$$Y = \text{SSM}(A, B, C)(X) = MX$$

where $M_{ij} = 0$ for $i < j$ (i.e. it's lower triangular)

$$M_{ij} = C_i^\top A_{i:j}^\times B_j := C_i^\top A_i \dots A_{j+1} B_j \qquad (4)$$

$$\begin{bmatrix} C_0^\top B_0 & & & & \\ C_1^\top A_1 B_0 & C_1^\top B_1 & & & \\ C_2^\top A_2 A_1 B_0 & C_2^\top A_2 B_1 & C_2^\top B_2 & & \\ \vdots & \vdots & \ddots & \ddots & \\ C_{\mathrm{T}}^\top A_{\mathrm{T}-1} \dots A_1 B_0 & C_{\mathrm{T}}^\top A_{\mathrm{T}-1} \dots A_2 B_1 & \dots & C_{\mathrm{T}}^\top A_{\mathrm{T}-1} B_{\mathrm{T}-2} & C_{\mathrm{T}}^\top B_{\mathrm{T}-1} \end{bmatrix}$$

Matrix Transformation Representation of State Space Model (5)

This type of matrix is called a (triangular) ***semiseparable matrix*** *and has been studied in other fields of engineering and computational linear algebra.*

an alternative characterization of semiseparable matrices is their structured rank property, which says that every submatrix contained in the lower-triangular portion is low rank.

State Space Duality (Mamba-2) Part II - The Theory

➢ Semiseparable Matrices

- All submatrices contained on-and-below the diagonal of a semiseparable matrix are low-rank.

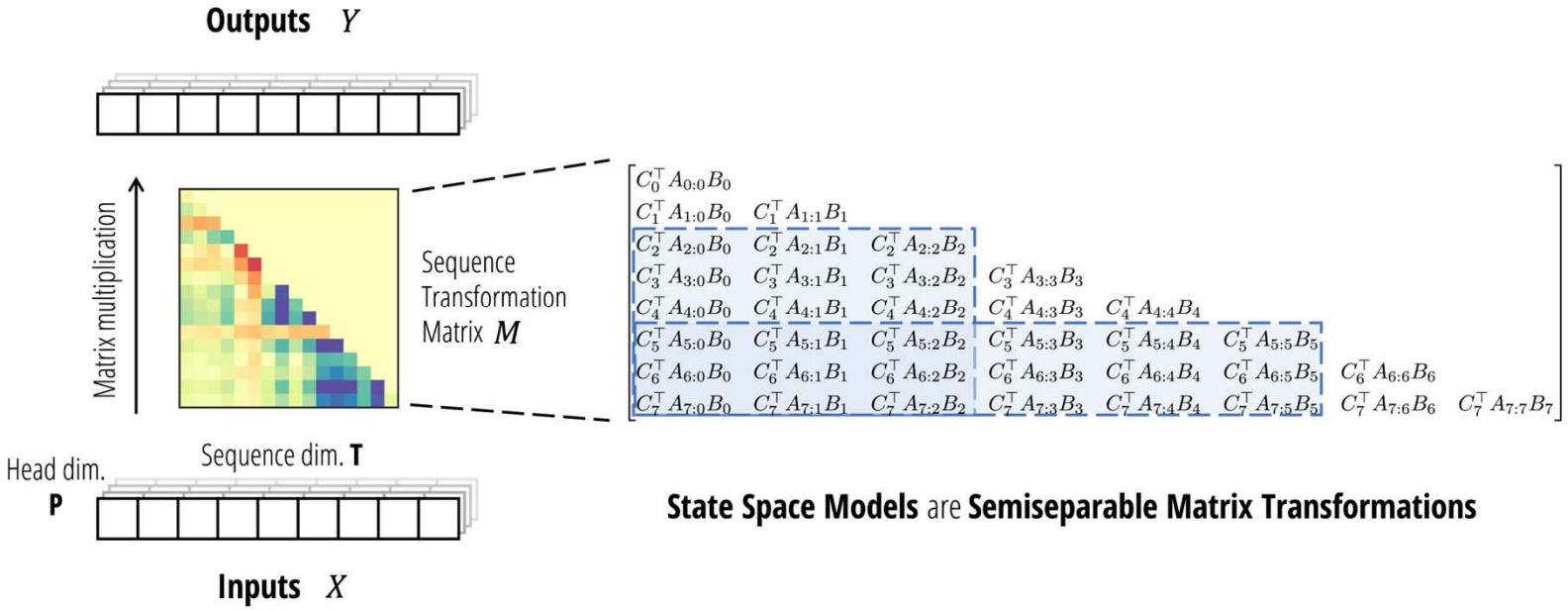


Figure 2: (**State Space Models are Semiseparable Matrices**.)

As sequence transformations, state space models can be represented as a matrix transformation $M \in \mathbb{R}^{(T,T)}$ acting on the sequence dimension T, sharing the same matrix for each channel in a head (Left).

This matrix is a semiseparable matrix (Right), which is a rank-structured matrix where every submatrix contained on-and-below the diagonal (Blue) has rank at most N, equal to the SSM's state dimension.

TAKEAWAY: COMPUTING SSMS THROUGH MATRIX MULTIPLICATION

All algorithms for computing state space models can be viewed as structured matrix multiplication algorithms on semiseparable matrices.

State Space Duality (Mamba-2) Part II - The Theory

➢ Deriving the Duality: SSM to Attention

- To show that equation (3) follows from equation (1) (in the case of the SSD model, i.e. **scalar SSM**), we directly use the matrix form of the state space model (4).
- Because A_t are all scalars in this case, they can be factored out of the entries which directly implies equation.

$$C_i^\top A_{i:j}^\times B_j = A_{i:j}^\times \cdot (C_i^\top B_j)$$

❖ In summary

> DUALITY REPRESENTATION 1 (SSM)
>
> *The duality for the SSD model can be seen as two **different matrix multiplication algorithms** on the semiseparable matrix.*

- The **linear form** is ***a structured matrix multiplication algorithm*** that *computes the outputs sequentially*, *leveraging the structure of the semiseparable matrix*.
- The **quadratic form** is the ***naive matrix multiplication algorithm*** that *materializes the full matrix*.

State Space Duality (Mamba-2) Part II - The Theory

➢ Going Beyond the SSD Layer 1

- The **power of the *semiseparable matrix representation*** applies to all state space models, with various downstream implications.

● **Algorithms (Algorithmic ideas)**

- Algorithmically, the Mamba-2 paper explores several consequences, such as:

1) The above **duality result for SSD model**, i.e. *a scalar-identity structured SSM*.

2) **New asymptotic efficiency** results for SSM (Theorem 3.7), which follow from applying known results from the semiseparable matrix literature.

3) A **more general hybrid algorithm** that can be viewed as combining both the linear and quadratic forms to get the best of both worlds. This can be derived as a new matrix multiplication algorithm utilizing block decompositions of the semiseparable matrix. (☞ Part III of this blog series!)

● **Understanding (Conceptual ideas)**

- Conceptually, the **matrix transformation viewpoint** helps *provide a unifying view of sequence models*.

- Some example downstream ideas;

► ***New sequence models*** : Restricting to matrix transformations reduces the problem of developing *new sequence models* to that of *finding structured matrix classes with target properties*. (☞ Derive the most natural bidirectional extension of Mamba (coming very soon!))

► ***Expressivity*** : Help us understand what different models can represent from a linear algebraic perspective. (☞ Study which subquadratic models are the most amenable to being distilled from Transformers.)

► ***Interpretability*** : Cocurrent work derived the matrix formulation of SSMs and use it to probe the internal representations of Mamba models.

State Space Duality (Mamba-2) Part II - The Theory

3. SSD Framework 2: Structured Attention

- **The second framing of the *duality*** is from **an attention-centric perspective**, where we'll prove the duality through **the framework of tensor contractions**.

➢ Warm-up: Kernel Attention

- Define attention as a function

$$(Q^{(\mathrm{T},\mathrm{N})}, K^{(\mathrm{S},\mathrm{N})}, V^{(\mathrm{S},\mathrm{P})}) \mapsto Y^{(\mathrm{T},\mathrm{P})}$$

given by the pairwise matrix multiplications

$$Y = (QK^\top) \cdot V$$

- We'll restrict to the case when ψ is finite, called ***kernel attention***.

- **On Dimensions**
- Think of **P=N** as the head dimension;
  - ✓ In attention the **V** head dimension **P** can differ from the **QK** head dimension **N**.
  - ✓ Think of **T** as the *target* sequence dimension and **S** as the *source* sequence dimension.
- Covers more general forms of attention such as *cross-attention*, where the *source* and *target* are separate sequences with *different lengths*.
- However, we'll assume *the self-attention* setting where **S=T**.

- **Why can we assume this form?**
- The usual form of attention $Y = f(QK^T) \cdot V$ (e.g. where f is the softmax function) can be written as $Y = \psi(Q)\psi(K)^T \cdot V$ for some appropriate feature map ψ (which may be infinite dimensional).
- In this case, we can simply redefine $Q \leftarrow \psi(Q)$ and define N to be the feature dimension of the attention kernel to begin with. Softmax attention, for example, can be represented with a particular infinite-dimensional feature map ($N = \infty$) which represents the exponential kernel.

State Space Duality (Mamba-2) Part II - The Theory

➢ Warm-up: Kernel Attention

$$(Q^{(\mathtt{T},\mathtt{N})}, K^{(\mathtt{S},\mathtt{N})}, V^{(\mathtt{S},\mathtt{P})}) \mapsto Y^{(\mathtt{T},\mathtt{P})} \qquad Y = (QK^\top) \cdot V$$

- When the sequence length **T** grows and the feature dimension **N** is small (commonly, in the regime when ψ is simple such as an elementwise transform and so **N** is constant), then the cost of attention can be reduced from quadratic in **T** to linear.
- This follows from simply computing the matrix multiplications in a different order.

$$Y = Q \cdot (K^\top V)$$

> *The most common way of linearizing attention is usually viewed as a consequence of the* ***associativity of matrix multiplication***

State Space Duality (Mamba-2) Part II - The Theory

➢ (Causal) Linear Attention

- Once the basic kernel attention is slightly modified, we can no longer use *the associativity of matrix multiplication* directly.

❖ The seminal **Linear Attention (LA)** framework of Katharopoulos et al. can be extended to the case of incorporating **causality** into **attention**, for autoregressive settings such as language modeling.

- The quadratic form of **causal linear attention** is defined with the **causal mask matrix *L***;

$$Y = (L \circ QK^\top) \cdot V \qquad (6)$$

where $L = \begin{bmatrix} 1 & & \\ \vdots & \ddots & \\ 1 & \dots & 1 \end{bmatrix}$

- The issue is : once the ***L*** mask is incorporated into (6), we can no longer directly apply ***matrix associativity***!

$$Y = (QK^\top) \cdot V$$
$$Y = Q \cdot (K^\top V)$$
$$Y = (L \circ QK^\top) \cdot V$$

- Eq (6) is equivalent to a different form which avoids materializing the quadratic QK^T attention matrix and has linear time complexity.

$$Y = Q \cdot \mathbf{cumsum}(K^\top V)$$

WHERE DOES THE CUMSUM IN LINEAR ATTENTION COME FROM?

The appearance of the cumulative sum *in linear attention is exactly equivalent to the fact that the causal mask* L*, as a matrix multiplication, encodes cumulative sums:*

$$y = L \cdot x \iff y = \mathbf{cumsum}(x)$$

State Space Duality (Mamba-2) Part II - The Theory

➢ A Tensor Contraction Proof of Linear Attention

- Let's write out the quadratic form of linear attention (6) very explicitly in **tensor contraction** or ***einsum*** notation, with shape annotations:

$$\begin{aligned}
&\text{(Structured Masked Attention - Quadratic Form)} \\
&G = \texttt{contract}(\texttt{TN, SN} \to \texttt{TS})(Q, K) \\
&M = \texttt{contract}(\texttt{TS, TS} \to \texttt{TS})(G, L) \\
&Y = \texttt{contract}(\texttt{TS, SP} \to \texttt{TP})(M, V)
\end{aligned} \tag{7}$$

☞ numpy.einsum : https://numpy.org/doc/stable/reference/generated/numpy.einsum.html

- We can notice that this sequence of contractions can be written as *a single four-way contraction*

$$y = \texttt{contract}(\texttt{TN, SN, SP, TS} \to \texttt{TP})(Q, K, V, L). \tag{8}$$

- Finally, it can be computed with any other contraction ordering. In particular, we can perform pairwise reductions on the order **V,K,L,Q** instead of **Q,K,L,V.**

$$\begin{aligned}
&\text{(Structured Masked Attention - Linear Form)} \\
&Z = \texttt{contract}(\texttt{SP, SN} \to \texttt{SPN})(V, K) \\
&H = \texttt{contract}(\texttt{TS, SPN} \to \texttt{TPN})(L, Z) \\
&Y = \texttt{contract}(\texttt{TN, TPN} \to \texttt{TP})(Q, H)
\end{aligned} \tag{9}$$

- **The key observation** is that the second line of (9) is simply a matrix multiplication by L, which can be computed with a ***cumulative sum***.
- We didn't have to write out a single summation, which was abstracted out into a tensor contraction combined with the structure of L.
- This immediately proves our claim about ***the cumsum in linear attention***. Moreover, this immediately reveals that the efficiency of linear attention can be made much more general…

State Space Duality (Mamba-2) Part II - The Theory

➢ Structured Masked Attention

- In order for (9) to be fast, all that is necessary is for **L** to be any **structured matrix** – any matrix that has *subquadratic matrix-vector multiplication*.

DEFINITION: STRUCTURED MASKED ATTENTION

Structured masked attention (SMA) *is defined as the* four-way tensor contraction (8) *using an attention mask* L *that is a structured matrix.*

DUALITY REPRESENTATION 2 (SMA)

SMA has ***dual quadratic and linear*** [3] ***modes*** *which are simply* two different pairwise reduction orders (7) *and* (9).

- Finally, let's just **connect** this back to the commonly view of *linear attention* as *matrix multiplication associativity*.

> *Although it is commonly believed that incorporating attention masks* L *prevents matrix multiplication reordering, it turns out to still be compatible. In particular,* ***associativity of matrix multiplication*** *is a special case of* ***tensor contraction reduction orders****; although the former no longer applies, the latter can integrate the attention mask* L*.*

State Space Duality (Mamba-2) Part II - The Theory

➢ Deriving the Duality: Attention to SSM

- Recall that the **SSD model** is defined as either **a scalar-identity SSM** in equation (1), or through **the attention-like form** in equation (3).
- To show the ***equivalence*** of these forms, we recognize that (3) is a special case of ***structured masked attention*** where the mask matrix is

$$L = \underbrace{\begin{bmatrix} 1 & & & & \\ a_1 & 1 & & & \\ a_2 a_1 & a_2 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ a_{T-1} \dots a_1 & a_{T-1} \dots a_2 & \dots & a_{T-1} & 1 \end{bmatrix}}_{\text{(1-semiseparable (1-SS) matrix)}} \quad (10)$$

- The SSD model is **1-semiseparable masked attention** or **1-SS SMA**.

- To prove that this can be written as an SSM, we appeal to **the SMA framework**, which the dual form of this model can be computed through matrix multiplication by L.

→ So how fast is that? : Multiplication $\boldsymbol{y = Lx}$ can be computed in *linear time* through ***a scalar recurrence***:

$$\begin{aligned} y_0 &= x_0 \\ y_1 &= a_1 x_0 + a_1 \\ y_2 &= a_2 a_1 x_0 + a_2 x_1 + x_2 = a_2 y_1 + x_2 \\ &\vdots \quad \vdots \end{aligned}$$

- This corresponds exactly to the original SSM recurrence!

State Space Duality (Mamba-2) Part II - The Theory

➢ Going Beyond the SSD Layer 2

- **Structured masked attention** not only helps **define the SSD model and prove its duality**, but it is **a much broader framework of efficient attention models**.

- Prior examples include *the original linear attention* as well as *the recent Retentive Network (RetNet) model*.

✓ These can be viewed as direct special cases of SSD.

- We can define **classes of efficient attention** by replacing the mask L with any structured matrix.

- *Toeplitz* or *Fourier* structured attention may be interesting to consider because they might encode different forms of positional information.

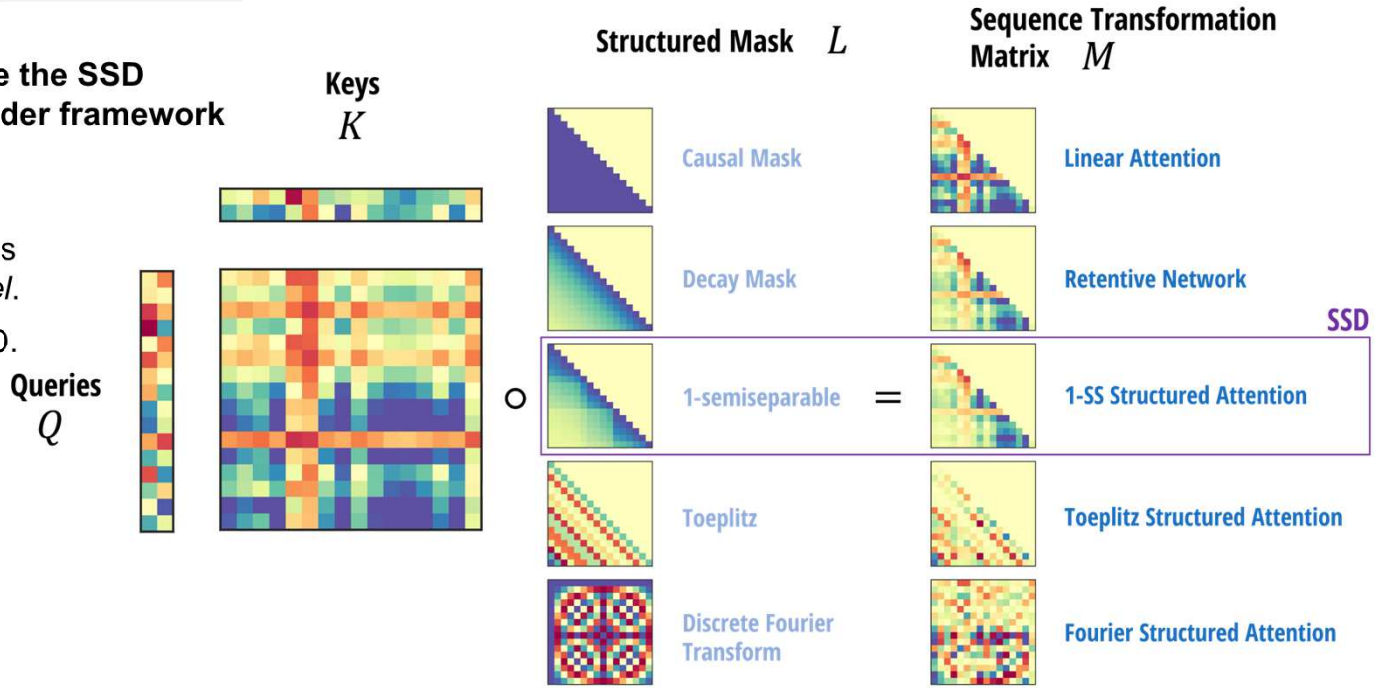


Figure 3: (Structured Masked Attention.) SMA constructs a masked attention matrix $M = QK^\top \circ L$ for any structured matrix L, which defines a matrix sequence transformation $Y = MV$. All instances of SMA have a dual subquadratic form induced by a different contraction ordering, combined with the efficient structured matrix multiplication by L.

Previous examples include Linear Attention (Katharopoulos et al.2020) and RetNet (Y. Sun et al.2023). Beyond SSD (1-semiseparable SMA), the focus of this paper, many other potential instantiations of structured attention are possible.

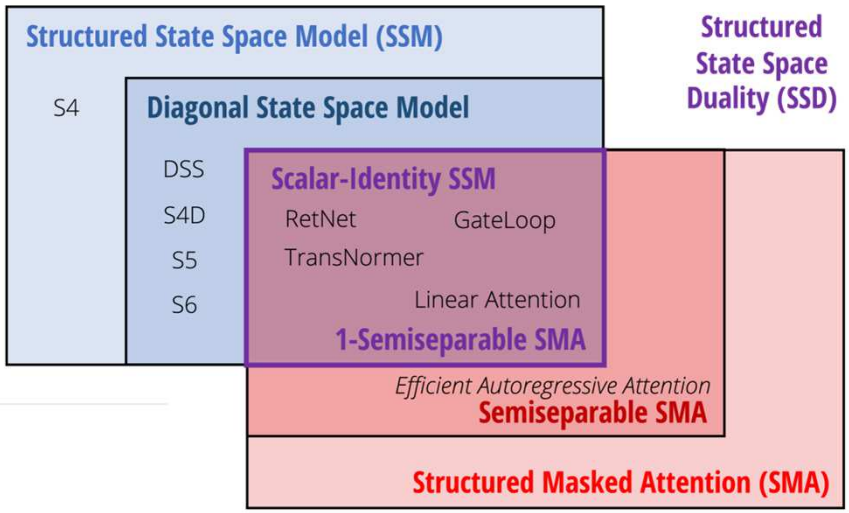
State Space Duality (Mamba-2) Part II - The Theory

4. State Space Duality

- **SSD framework** consists of the two broad approaches covered in this post, which is summarized by the two areas of the [Venn diagram]:

  1) Viewing state space models through [structured matrix transformations]

  2) Generalizing linear attention through [tensor contractions]



SSD Framework	**Structured SSMs**	**Structured Attention**
The main representation is...	Structured matrix (5) sequence transformations	The 4-way (8) tensor contraction
This generalizes...	State space models	Linear attention
The SSD model is an instantiation as...	Scalar state space model (A_t is a scalar-identity matrix)	1-semiseparable masked attention (L mask is a 1-SS matrix)
The linear-quadratic duality is revealed through...	Structured matrix multiplication algorithms	Tensor contraction reduction orderings

State Space Duality (Mamba-2) Part III - The Algorithm

1. The SSD Algorithm

- The theoretical **framework** of **structured state space duality** (see Part I and Part II of this series) **connects SSMs** and **(linear) attention** through **structured matrices**.
- This connection allows us to **derive new algorithms for selective SSMs** that are faster than the parallel associative scan in Mamba-1 by **leveraging matrix multiplication as a primitive**. (in Part I)
- Moreover, the connection can bring **system optimizations** (e.g. *tensor parallelism*, *sequence parallelism*, *variable sequence length*) originally developed for Transformer to SSM-land.

- Developed optimized **scans** implementations for **Mamba-1**
- **Limited to small state expansion** (typically **N=16**) as the algorithm and **did not use tensor cores** (specialized hardware units that perform matrix multiplication).

- Typically matrix multiplication (**matmul**) **FLOPs** are much faster (up to **16x**) than **non-matmul FLOPs**:
- A100 : 312 TFLOPS of BF16 matmul, but only 19 TFLOPS of FP32 arithmetics
- H100 : 989 TFLOPS of BF16 matmul, but only 67 TFLOPS of FP32 arithmetics

➢ One of our **primary goals with Mamba-2** is **to leverage tensor cores to speed up the SSM.**

State Space Duality (Mamba-2) Part III - The Algorithm

1. The SSD Algorithm

- To Recap

- After tying parameters and introducing the head structure, the **SSM** in Mamba-1 turns into **SSD**, a more restrictive form that has an attention-like formulation.
- As SSD connects SSMs and structured matrices, we saw in Part II that efficient algorithms to compute SSMs correspond directly to different decompositions of the "***token-mixing***" or "***sequence-mixing***" matrix.

- We can therefore **create new algorithms to compute SSMs simply by looking for alternative ways to multiply this matrix**.
- A simple **block decomposition of this matrix**, with carefully chosen ***block sizes***, turns out to get all the advantages of both the *linear-recurrent* and *quadratic-attention dual forms* of SSD.

→ This leads to the SSD algorithm, which has 4 steps.

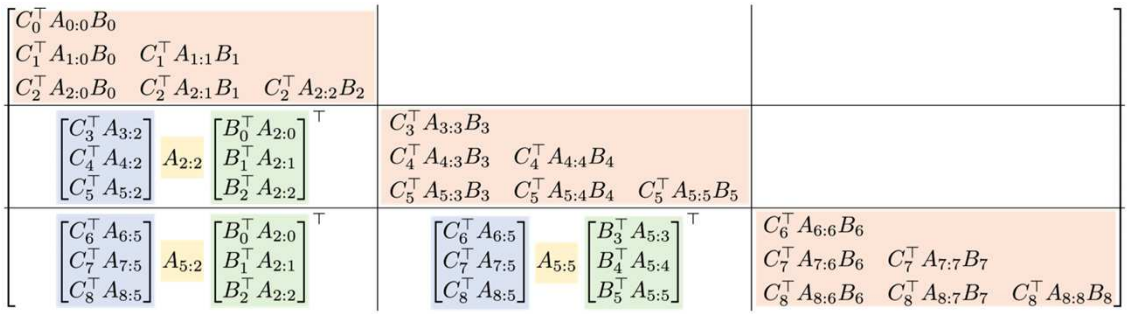
- Two completely different interpretations of this algorithm!
  1) Block Matrix Decomposition
  2) Chunking and State Passing

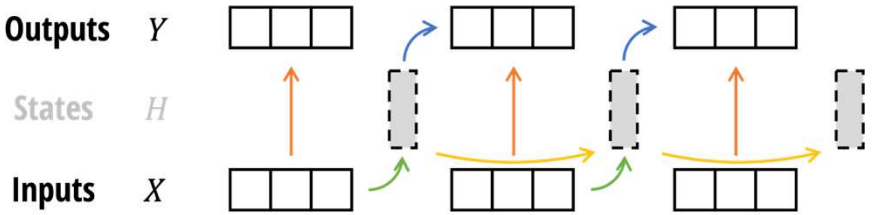
State Space Duality (Mamba-2) Part III - The Algorithm

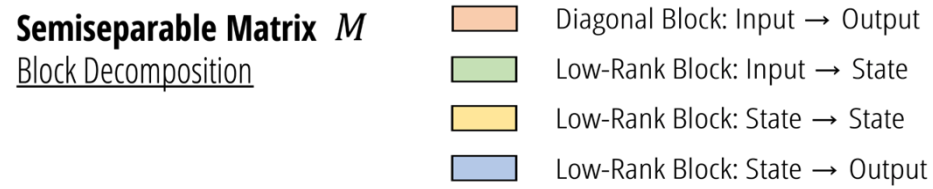
1) SSD Algorithm: Block Matrix Decomposition

- We first partition the SSM (semiseparable) matrix into blocks of size **Q**×**Q**. Then, we use the properties of semiseparable matrices to factorize each off-diagonal block, which is low rank.

1) (Orange) Each diagonal block is ***a smaller semiseparable matrix***; we can compute this multiplication however we like; in particular, using *the quadratic (attention-like)* form of SSD.
2) (Green) There are only **T/Q** total different green blocks because many of them are **shared**. These can be computed with **a batched matmul**.
3) (Yellow) Notice that the yellow terms themselves form **a 1-semiseparable matrix**; this step is equivalently to **an SSM scan** (on some modified A factors)!
4) (Blue) Similar to green, these can be computed with **a batched matmul**.







State Space Duality (Mamba-2) Part III - The Algorithm

1) SSD Algorithm: Block Matrix Decomposition

[☞ Paper, 6 A Hardware-Efficient Algorithm for SSD Models]

- We partition the matrix M into a T/Q × T/Q grid of submatrices of size Q × Q, for some block size Q.
- Note that the off-diagonal blocks are low-rank by the defining property of semiseparable matrices (Definition 3.1).

$$\text{(Block Decomposition)} \quad M = \begin{bmatrix} M^{(0,0)} & & & \\ M^{(1,0)} & M^{(1,1)} & & \\ \vdots & \vdots & \ddots & \\ M^{(\mathtt{T}/\mathtt{Q}-1,0)} & M^{(\mathtt{T}/\mathtt{Q}-1,1)} & \cdots & M^{(\mathtt{T}/\mathtt{Q}-1,\mathtt{T}/\mathtt{Q}-1)} \end{bmatrix}$$

$$\text{(Diagonal Block)} \quad M^{(j,j)} = \mathrm{SSM}(A_{j\mathtt{Q}:(j+1)\mathtt{Q}}, B_{j\mathtt{Q}:(j+1)\mathtt{Q}}, C_{j\mathtt{Q}:(j+1)\mathtt{Q}})$$

$$\text{(Low-Rank Block)} \quad M^{(j,i)} = \begin{bmatrix} C_{j\mathtt{Q}}^\top A_{j\mathtt{Q}:j\mathtt{Q}-1} \\ \vdots \\ C_{(j+1)\mathtt{Q}-1}^\top A_{(j+1)\mathtt{Q}-1:j\mathtt{Q}-1} \end{bmatrix} A_{j\mathtt{Q}-1:(i+1)\mathtt{Q}-1} \begin{bmatrix} B_{i\mathtt{Q}}^\top A_{(i+1)\mathtt{Q}-1:i\mathtt{Q}} \\ \vdots \\ B_{(i+1)\mathtt{Q}-1}^\top A_{(i+1)\mathtt{Q}-1:(i+1)\mathtt{Q}-1} \end{bmatrix}^\top$$

State Space Duality (Mamba-2) Part III - The Algorithm

1) SSD Algorithm: Block Matrix Decomposition

- An example, e.g. for **T=9** and decomposing into chunks of length **Q=3.**
- The shaded cells are low-rank factorizations of the off-diagonal blocks of the semiseparable matrix.
- From here we can reduce the problem into these two parts. These can also be interpreted as dividing the output of a "chunk" $y_{jQ:(j+1)Q}$ into two components: the effect of inputs within the chunk $x_{jQ:(j+1)Q}$, and the effect of inputs before the chunk $x_{0:jQ}$.

$$
M = \left[\begin{array}{c|cc|ccc}
C_0^\top A_{0:0} B_0 & & & & & \\
C_1^\top A_{1:0} B_0 & C_1^\top A_{1:1} B_1 & & & & \\
C_2^\top A_{2:0} B_0 & C_2^\top A_{2:1} B_1 & C_2^\top A_{2:2} B_2 & & & \\
\hline
C_3^\top A_{3:0} B_0 & C_3^\top A_{3:1} B_1 & C_3^\top A_{3:2} B_2 & C_3^\top A_{3:3} B_3 & & \\
C_4^\top A_{4:0} B_0 & C_4^\top A_{4:1} B_1 & C_4^\top A_{4:2} B_2 & C_4^\top A_{4:3} B_3 & C_4^\top A_{4:4} B_4 & \\
C_5^\top A_{5:0} B_0 & C_5^\top A_{5:1} B_1 & C_5^\top A_{5:2} B_2 & C_5^\top A_{5:3} B_3 & C_5^\top A_{5:4} B_4 & C_5^\top A_{5:5} B_5 \\
\hline
C_6^\top A_{6:0} B_0 & C_6^\top A_{6:1} B_1 & C_6^\top A_{6:2} B_2 & C_6^\top A_{6:3} B_3 & C_6^\top A_{6:4} B_4 & C_6^\top A_{6:5} B_5 \\
C_7^\top A_{7:0} B_0 & C_7^\top A_{7:1} B_1 & C_7^\top A_{7:2} B_2 & C_7^\top A_{7:3} B_3 & C_7^\top A_{7:4} B_4 & C_7^\top A_{7:5} B_5 \\
C_8^\top A_{8:0} B_0 & C_8^\top A_{8:1} B_1 & C_8^\top A_{8:2} B_2 & C_8^\top A_{8:3} B_3 & C_8^\top A_{8:4} B_4 & C_8^\top A_{8:5} B_5
\end{array}\right.
\left.\begin{array}{ccc}
 & & \\
 & & \\
 & & \\
\hline
 & & \\
 & & \\
 & & \\
\hline
C_6^\top A_{6:6} B_6 & & \\
C_7^\top A_{7:6} B_6 & C_7^\top A_{7:7} B_7 & \\
C_8^\top A_{8:6} B_6 & C_8^\top A_{8:7} B_7 & C_8^\top A_{8:8} B_8
\end{array}\right]
$$

$$
= \left[\begin{array}{c|cc|ccc}
C_0^\top A_{0:0} B_0 & & & & & \\
C_1^\top A_{1:0} B_0 & C_1^\top A_{1:1} B_1 & & & & \\
C_2^\top A_{2:0} B_0 & C_2^\top A_{2:1} B_1 & C_2^\top A_{2:2} B_2 & & & \\
\hline
\begin{bmatrix} C_3^\top A_{3:2} \\ C_4^\top A_{4:2} \\ C_5^\top A_{5:2} \end{bmatrix} A_{2:2} \begin{bmatrix} B_0^\top A_{2:0} \\ B_1^\top A_{2:1} \\ B_2^\top A_{2:2} \end{bmatrix}^\top & & & C_3^\top A_{3:3} B_3 & & \\
 & & & C_4^\top A_{4:3} B_3 & C_4^\top A_{4:4} B_4 & \\
 & & & C_5^\top A_{5:3} B_3 & C_5^\top A_{5:4} B_4 & C_5^\top A_{5:5} B_5 \\
\hline
\begin{bmatrix} C_6^\top A_{6:5} \\ C_7^\top A_{7:5} \\ C_8^\top A_{8:5} \end{bmatrix} A_{5:2} \begin{bmatrix} B_0^\top A_{2:0} \\ B_1^\top A_{2:1} \\ B_2^\top A_{2:2} \end{bmatrix}^\top & & & \begin{bmatrix} C_6^\top A_{6:5} \\ C_7^\top A_{7:5} \\ C_8^\top A_{8:5} \end{bmatrix} A_{5:5} \begin{bmatrix} B_3^\top A_{5:3} \\ B_4^\top A_{5:4} \\ B_5^\top A_{5:5} \end{bmatrix}^\top & &
\end{array}\right.
\left.\begin{array}{ccc}
 & & \\
 & & \\
 & & \\
\hline
 & & \\
 & & \\
 & & \\
\hline
C_6^\top A_{6:6} B_6 & & \\
C_7^\top A_{7:6} B_6 & C_7^\top A_{7:7} B_7 & \\
C_8^\top A_{8:6} B_6 & C_8^\top A_{8:7} B_7 & C_8^\top A_{8:8} B_8
\end{array}\right]
$$

State Space Duality (Mamba-2) Part III - The Algorithm

2) SSD Algorithm: Chunking and State Passing

- An alternative interpretation of the algorithm **involves reasoning about how the SSM operates on the actual sequence.**

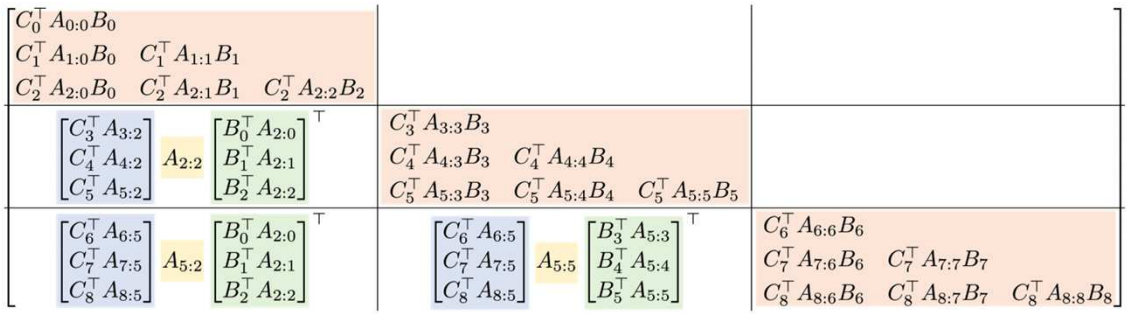
- We first split the sequence of input into blocks (or chunks) of size Q. The steps then have the interpretation

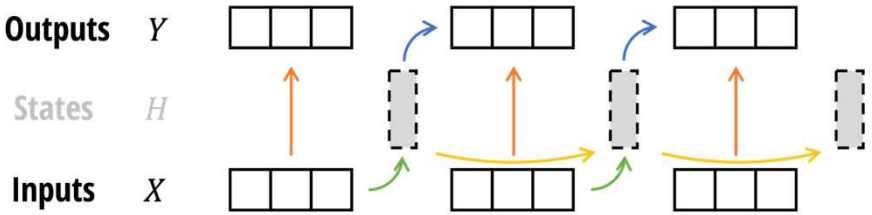
1) **Intra-chunk outputs**: compute the local output of each chunk *(what is the output per chunk supposing that the initial state (to the chunk) is 0?*)

2) **Chunk states**: compute the final state of each chunk (*what is the final state per chunk supposing that the initial state (to the chunk) is 0?*)

3) **Pass states**: compute a recurrence on all of the chunks' final states – using any desired algorithm, e.g. parallel or sequential scan (*what is the actual final state per chunk taking into account all previous inputs?*)

4) **Output states**: for each chunk, given its true initial state (computed in Step 3), compute the contribution to the output just from the initial state





Semiseparable Matrix M

Block Decomposition

Diagonal Block: Input → Output

Low-Rank Block: Input → State

Low-Rank Block: State → State

Low-Rank Block: State → Output

State Space Duality (Mamba-2) Part III - The Algorithm

2) SSD Algorithm: Chunking and State Passing

- An alternative interpretation of the algorithm **involves reasoning about how the SSM operates on the actual sequence.**

- We first split the sequence of input into blocks (or chunks) of size Q. The steps then have the interpretation

1) **Intra-chunk outputs**: compute the local output of each chunk *(what is the output per chunk supposing that the initial state (to the chunk) is 0?*)
2) **Chunk states**: compute the final state of each chunk (*what is the final state per chunk supposing that the initial state (to the chunk) is 0?*)
3) **Pass states**: compute a recurrence on all of the chunks' final states – using any desired algorithm, e.g. parallel or sequential scan (*what is the actual final state per chunk taking into account all previous inputs?*)
4) **Output states**: for each chunk, given its true initial state (computed in Step 3), compute the contribution to the output just from the initial state

- Most of the algorithm (**Step 1, 2, and 4**) leverages **matmuls** (and hence tensor cores), and also can be computed completely in **parallel**!
- Only **Step 3 requires a scan**, but it **operates on a much shorter sequence** and usually only takes a small fraction of the time of the full algorithm

State Space Duality (Mamba-2) Part III - The Algorithm

2. Code

- "Minimal SSD" code (☞ https://github.com/state-spaces/mamba/blob/main/mamba_ssm/modules/ssd_minimal.py)
- This algorithm is not only faster but also much easier to implement than the original selective scan of Mamba, coming in at just around 25 lines of code!

Listing 1 Full PyTorch example of the state space dual (SSD) model.

```python
def segsum(x):
    """Naive segment sum calculation. exp(segsum(A)) produces a 1-SS matrix,
       which is equivalent to a scalar SSM."""
    T = x.size(-1)
    x_cumsum = torch.cumsum(x, dim=-1)
    x_segsum = x_cumsum[..., :, None] - x_cumsum[..., None, :]
    mask = torch.tril(torch.ones(T, T, device=x.device, dtype=bool), diagonal=0)
    x_segsum = x_segsum.masked_fill(~mask, -torch.inf)
    return x_segsum


def ssd(X, A, B, C, block_len=64, initial_states=None):
    """
    Arguments:
        X: (batch, length, n_heads, d_head)
        A: (batch, length, n_heads)
        B: (batch, length, n_heads, d_state)
        C: (batch, length, n_heads, d_state)
    Return:
        Y: (batch, length, n_heads, d_head)
    """
    assert X.dtype == A.dtype == B.dtype == C.dtype
    assert X.shape[1] % block_len == 0

    # Rearrange into blocks/chunks
    X, A, B, C = [rearrange(x, "b (c l) ... -> b c l ...", l=block_len) for x in (X, A, B, C)]

    A = rearrange(A, "b c l h -> b h c l")
    A_cumsum = torch.cumsum(A, dim=-1)
```

State Space Duality (Mamba-2) Part III - The Algorithm

2. Code

```
# 1. Compute the output for each intra-chunk (diagonal blocks)
L = torch.exp(segsum(A))
Y_diag  = torch.einsum("bclhn,bcshn,bhcls,bcshp->bclhp", C, B, L, X)

# 2. Compute the state for each intra-chunk
# (right term of low-rank factorization of off-diagonal blocks; B terms)
decay_states = torch.exp((A_cumsum[:, :, :, -1:] - A_cumsum))
states = torch.einsum("bclhn,bhcl,bclhp->bchpn", B, decay_states, X)

# 3. Compute the inter-chunk SSM recurrence; produces correct SSM states at chunk boundaries
# (middle term of factorization of off-diag blocks; A terms)
if initial_states is None:
    initial_states = torch.zeros_like(states[:, :1])
states = torch.cat([initial_states, states], dim=1)
decay_chunk = torch.exp(segsum(F.pad(A_cumsum[:, :, :, -1], (1, 0))))
new_states = torch.einsum("bhzc,bchpn->bzhpn", decay_chunk, states)
states, final_state = new_states[:, :-1], new_states[:, -1]

# 4. Compute state -> output conversion per chunk
# (left term of low-rank factorization of off-diagonal blocks; C terms)
state_decay_out = torch.exp(A_cumsum)
Y_off = torch.einsum('bclhn,bchpn,bhcl->bclhp', C, states, state_decay_out)

# Add output of intra-chunk and inter-chunk terms (diagonal and off-diagonal blocks)
Y = rearrange(Y_diag+Y_off, "b c l h p -> b (c l) h p")
return Y, final_state
```

State Space Duality (Mamba-2) Part III - The Algorithm

3. The Details

➢ The SSM Scan

- In the above code, we utilized the connection between scalar SSM recurrences $h_{t+1} = A_t h_t + b_t x_t$ and matrix multiplication by 1-semiseparable matrices

$$L = \begin{bmatrix} 1 & & & \\ a_1 & 1 & & \\ a_2 a_1 & a_2 & 1 & \\ \vdots & \vdots & \ddots & \ddots \\ a_{T-1} \dots a_1 & a_{T-1} \dots a_2 & \dots & a_{T-1} & 1 \end{bmatrix}$$

- We compute Step 3 of the algorithm, which is computing a scalar SSM by any algorithm, by explicitly materializing a 1-SS matrix and doing dense matrix multiplication.

- Some reasons for using this version
- Code-wise, it's **simpler** to materialize and multiply by this matrix than to actually implement a parallel associative scan.
- Because of the **block decomposition of the SSM matrix**, the **sequence length T is reduced by a factor of ≈100** – so doing the scan in time $O(T^2)$ instead of $O(T)$ isn't too bad.
- We have to materialize a **1-SS matrix** for Step 1 of the algorithm (**the diagonal blocks**), so might as well **reuse the code**

- Note that
- While this example code is *simpler* and *reasonably* efficient on GPU (and probably TPU as well!), it's **no longer truly linear at long sequences**.
- Our more **optimized Triton implementation** does replace **the 1-SS multiplication in Step 3** with **an actual associative scan**.

State Space Duality (Mamba-2) Part III - The Algorithm

3. The Details

➢ Stability

- How should we do the **1-semiseparable matrix** in a simple and fast way?

Attempt 1: Ratios of cumprods

- The first naive attempt : notice that **the entries of this matrix are cumulative products**

$$a^{\times}_{i:j} = a_i \times \cdots \times a_{j-1} = \frac{a^{\times}_{i:\mathrm{T}}}{a^{\times}_{j:\mathrm{T}}}$$

- This runs into severe numerical issues because these products can get really tiny

Fix 1: The Segment Sum (segsum) Operation

- The second attempt : Do all of this in log-space, because all the a_t are positive; so the products become additions, and instead of cumprods to deal with we have cumsums instead.
- Then in order to compute the 1-SS matrix, we just have to compute the sums $\log a_i + \cdots + \log a_{j-1}$ for every *segment* [i:j].

☞ We call this the **segment sum** (**segsum**) primitive, analogous to cumulative sum (cumsum).

State Space Duality (Mamba-2) Part III - The Algorithm

3. The Details

➢ **Stability**

Attempt 2: Differences of cumsums

- The obvious way to do this again is using the same idea as above, but in log space

$$a^{\times}_{i:j} = \exp(\log a_i + \cdots + \log a_{j-1}) = (\log a)^{+}_{i:\mathrm{T}} - (\log a)^{+}_{j:\mathrm{T}}$$

where we compute a single cumulative sum of a along the time axis, and then compute all pairwise differences

```
def segsum_unstable(x):
    """Naive segment sum calculation."""
    T = x.size(-1)
    x_cumsum = torch.cumsum(x, dim=-1)
    x_segsum = x_cumsum[..., :, None] - x_cumsum[..., None, :]
    mask = torch.tril(torch.ones(T, T, device=x.device, dtype=bool), diagonal=0)
    x_segsum = x_segsum.masked_fill(~mask, -torch.inf)
    return x_segsum
```

- The 1-semiseparable matrix is just the exponential of this output
- Sums/differences are a lot more stable than products/quotients, so this should work – right?

State Space Duality (Mamba-2) Part III - The Algorithm

3. The Details

➢ **Stability**

Fix 2: Remove All Subtractions

- Unfortunately, this still doesn't work.
- **The values of this 1-SS matrix** roughly represent the SSM dynamics, which are very sensitive to these values of at, so we **have to be very precise**.
- Even in log space, these cumsums can be fairly large, which runs into **catastrophic cancellation** when *subtracted*.
- So we have to find **a way to compute this matrix with only additions**, while still vectorizing everything…

Attempt 3: Stable Segsum

- This leads to the helper function in the reference SSD code.
- Instead of computing a single cumsum and then subtracting, we find a way to use **a batch of independent cumsums** that immediately produces the right answer without subtraction.
- Without the right implementation of these primitives, the basic SSD algorithm produces NaNs immediately during training (even with FP32).

State Space Duality (Mamba-2) Part III - The Algorithm

3. The Details

➢ Discretization

- ► **S4** is a ***continuous-time model*** that excels at modeling continuous data, e.g. perceptual signals such as audio waveforms and pixel-level vision.
- ► **Mamba** is a ***discrete-time model*** that excels at modeling discrete data, e.g. tokenized data such as language.

- However, the parameterization of Mamba still used the same discretization step as in prior structured SSMs, where there is another parameter **Δ** being modeled.
- We're pretty sure that **the discretization step isn't necessary for Mamba**.
- In the Mamba-2 paper, we chose to work directly with **the "discrete parameters" A and B**, which in all previous structured SSM papers (including Mamba-1) were denoted $(\overline{\mathbf{A}}, \overline{\mathbf{B}})$

$$\bar{A} = \exp(e^{\Delta A})$$
$$\bar{B} = (\exp(e^{\Delta A}) - I)A^{-1}B$$

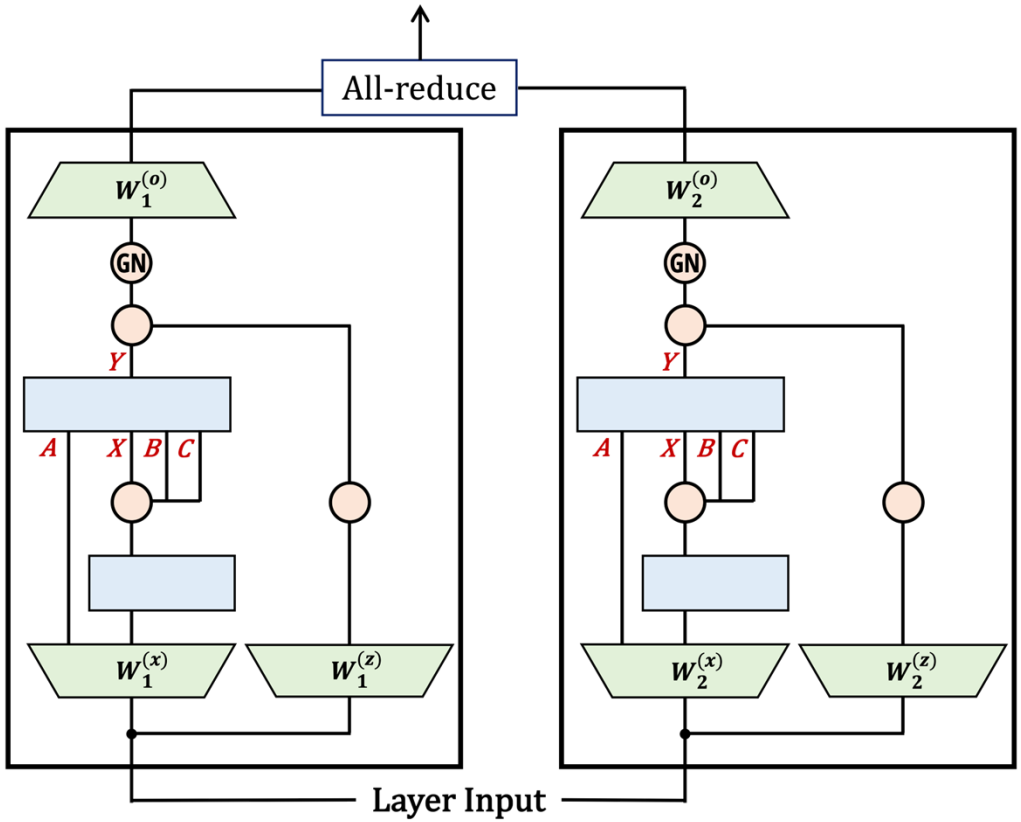
IS DISCRETIZATION NECESSARY?

It's useful for other structured SSMs, but perhaps not needed for Mamba. But it's just a simple invertible transformation, so use either discrete or continuous parameterizations as you like!

State Space Duality (Mamba-2) Part IV - The Systems

1. Systems and Scaling Optimizations

➢ **Tensor Parallelism**



- One difficulty with large-scaling training of Mamba-1 using ***tensor parallelism (TP)*** is that it **requires 2 all-reduces per layer**, compared to 1 all-reduce per attention or MLP layer in Transformer.
  - ✓ Because some SSM parameters are functions of the inner activations, not of the input to the layer.
- In **Mamba-2**, with the "***parallel projection***" structure, **all SSM parameters are functions of the input to the layer**, and we can **easily apply TP to the input projection**:
  1) Split the input projection and output projection matrices into 2, 4, 8 shards, depending on the TP degree.
  2) Use a grouped norm with number of groups divisible by the TP degree, so that normalization is done separately per GPU.
- These changes result in 1 all-reduce per layer, instead of 2.

Figure 7: (**Parallelism with the Mamba-2 Block**.) (**Tensor Parallelism**) We split the input projection matrices $W^{(x)}$, $W^{(z)}$ and the output projection matrix $W^{(o)}$. Each SSM head $(A, B, C, X) \mapsto Y$ lives on a single device. Choosing *GroupNorm* for the final normalization layer avoids extra communication. We need one all-reduce per layer, just like the MLP or attention blocks in a Transformer.

State Space Duality (Mamba-2) Part IV - The Systems

1. Systems and Scaling Optimizations

➢ **Sequence Parallelism**

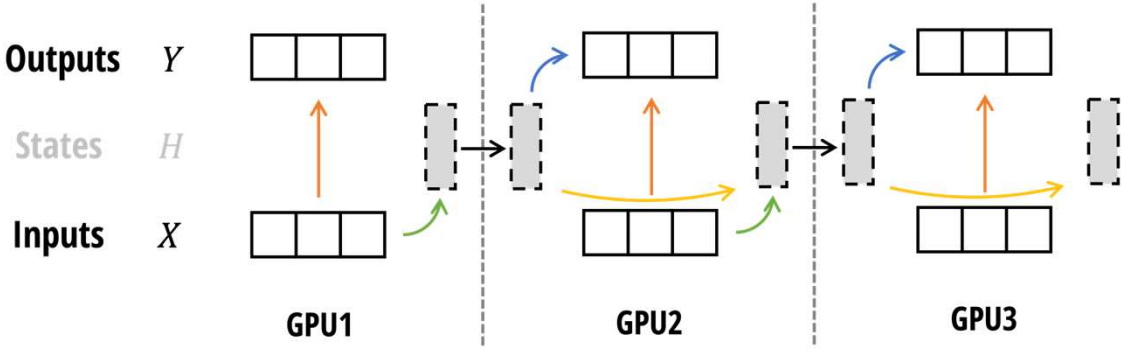


Figure 7: (**Parallelism with the Mamba-2 Block**.)

(**Sequence/Context Parallelism**) Analogous to the SSD algorithm, with multiple devices, we can split along the sequence dimension. Each device computes the state of its sequence, then pass that state to the next GPU.

- When training on very long sequence length, we might **need to split along the sequence length and assign different parts to different devices.**

- Two main forms of **sequence parallelism (SP)**:

1) ***For the residual and normalization operation***:
   - This replaces *the all-reduce in TP* with a **reduce-scatter**, **residual + normalization**, then **all-gather.**
   - Since Mamba-2 uses the same residual and normalization structure as Transformer, this form of SP applies directly with no modification.

2) ***For the attention or SSM operation*** (aka ***Context parallelism***):
   - For ***attention***; one could use ***Ring attention*** to split it up along the sequence dimension.
   - For ***Mamba-2***; the ***SSD framework*** : Using *the same block decomposition*, we can have each GPU computing its local output and its final states, then pass the states between GPUs (using send/receive communication primitives), before updating the final output of each GPU.

State Space Duality (Mamba-2) Part IV - The Systems

1. Systems and Scaling Optimizations

➢ **Variable Length**

- For **finetuning** and **inference**
- We often have sequences of different lengths in the same batch.

- For **Transformer**,
- One would usually pad so all sequences have the same length (wasting computation), or implement attention specifically for variable length sequences with careful *load-balancing*.

- **With SSM**,
- We can simply treat the whole batch as a long "sequence" and avoid passing the states between different sequences in the batch by setting the state transition A_t to 0 for tokens at the end of each sequence.

2. Results

- The **faster SSD algorithm** allows us to **increase the state dimension** (**N=64 or 128** compared to **N=16 in Mamba-1**).
- Even though technically Mamba-2 is more restricted than Mamba-1 for the same N, **the larger state dimensions generally improve model quality.**
- Results for models trained on **300B tokens** on the Pile, with Mamba-2 outperforming Mamba-1 and Pythia.

Table 1: (**Zero-shot Evaluations**.) Best results for each size in bold, second best unlined. We compare against open source LMs with various tokenizers, trained for up to **300B** tokens. Pile refers to the validation split, comparing only against models trained on the same dataset and tokenizer (GPT-NeoX-20B). For each model size, Mamba-2 outperforms Mamba, and generally matches Pythia at twice the model size.

Model	Token.	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	OpenbookQA acc ↑	Average acc ↑
Pythia-1B	NeoX	7.82	7.92	56.1	47.2	70.7	57.0	27.1	53.5	31.4	49.0
Mamba-790M	NeoX	7.33	6.02	**62.7**	**55.1**	**72.1**	**61.2**	**29.5**	56.1	34.2	53.0
Mamba-2-780M	NeoX	**7.26**	**5.86**	61.7	54.9	72.0	61.0	28.5	**60.2**	**36.2**	**53.5**
Hybrid H3-1.3B	GPT2	—	11.25	49.6	52.6	71.3	59.2	28.1	56.9	34.4	50.3
Pythia-1.4B	NeoX	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	30.8	51.7
RWKV4-1.5B	NeoX	7.70	7.04	56.4	52.5	72.4	60.5	29.4	54.6	34.0	51.4
Mamba-1.4B	NeoX	6.80	5.04	65.0	59.1	**74.2**	**65.5**	32.8	**61.5**	36.4	**56.4**
Mamba-2-1.3B	NeoX	**6.66**	**5.02**	**65.7**	**59.9**	73.2	64.3	**33.3**	60.9	**37.8**	**56.4**
Hybrid H3-2.7B	GPT2	—	7.92	55.7	59.7	73.3	65.6	32.3	61.4	33.6	54.5
Pythia-2.8B	NeoX	6.73	5.04	64.7	59.3	74.0	64.1	32.9	59.7	35.2	55.7
RWKV4-3B	NeoX	7.00	5.24	63.9	59.6	73.7	67.8	33.1	59.6	37.0	56.4
Mamba-2.8B	NeoX	6.22	4.23	69.2	66.1	75.2	**69.7**	36.3	63.5	**39.6**	59.9
Mamba-2-2.7B	NeoX	**6.09**	**4.10**	**69.7**	**66.6**	**76.4**	69.6	**36.4**	**64.0**	38.8	**60.2**

State Space Duality (Mamba-2) Part IV - The Systems

2. Results

- Hybrid models

- **Jamba** (A Hybrid Transformer-Mamba Language Model, https://arxiv.org/abs/2403.19887), **Zamba** (A Compact 7B SSM Hybrid Model, https://arxiv.org/abs/2405.16712) : Combining Mamba layers with attention layers can improve over pure Transformer or Mamba

- We validate at **2.7B parameters** and **300B tokens scale** that hybrid model with just **6 attention layers** (and **58 SSD layers**) outperforms 64 SSD layers, as well as Transformer++ (32 gated MLP and 32 attention layers).

Table 1: (**Zero-shot Evaluations**.) Best results for each size in bold. We compare different ways SSD, MLP, and attention layers can be combined, evaluated at 2.7B scale trained to 300B tokens on the Pile.

Model	Token.	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	OpenbookQA acc ↑	Average acc ↑
Transformer++	NeoX	6.13	3.99	70.3	66.4	75.2	67.7	37.8	63.9	**40.4**	60.2
Mamba-2	NeoX	6.09	4.10	69.7	66.6	**76.4**	69.6	36.4	64.0	38.8	60.2
Mamba-2-MLP	NeoX	6.13	4.18	69.3	65.0	**76.4**	68.1	37.0	63.1	38.2	59.6
Mamba-2-Attention	NeoX	**5.95**	**3.85**	**71.1**	**67.8**	75.8	69.9	37.8	**65.3**	39.0	**61.0**
Mamba-2-MLP-Attention	NeoX	6.00	3.95	70.0	66.6	75.4	**70.6**	**38.6**	64.6	39.2	60.7

2. Results

- We validated that the SSD algorithm is significantly faster than the parallel associative scan from Mamba-1 for the same state-dimension (see figure below, with sequence length 2k).
- Getting those tensor cores to go brrr is the key!

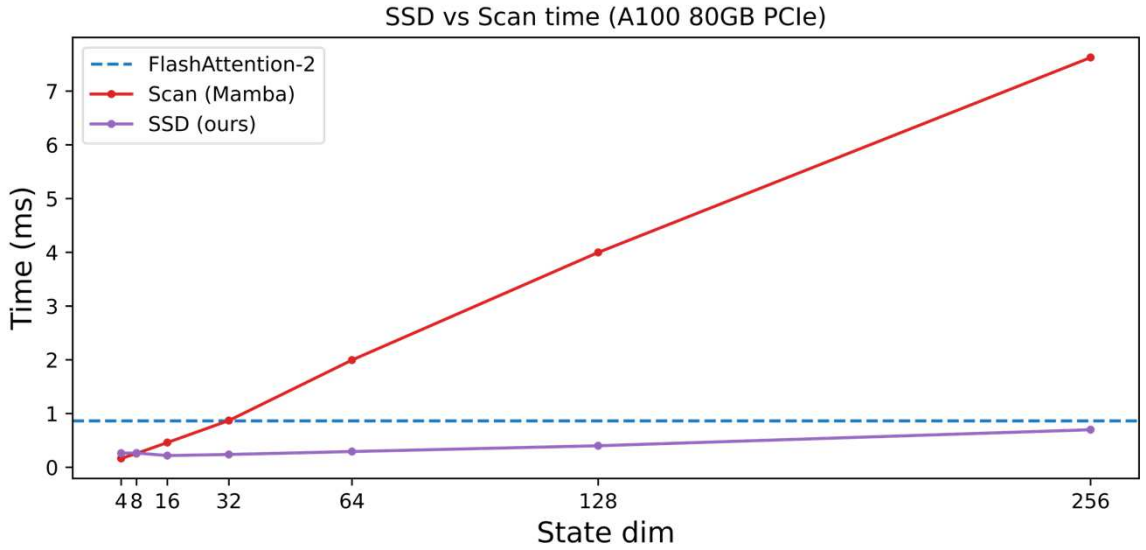


Figure 10: (**Efficiency Benchmarks**.) (Right) Sequence length 4K: Increasing state expansion slows down the Mamba optimized scan implementation linearly. SSD can handle much larger state expansion factors without much slowdown.

State Space Duality (Mamba-2) Part IV - The Systems

3. Future Directions

- With **SSD**, we have **connected (linear) attention and SSMs**, allowing us to **design faster algorithms and implement system optimizations for SSMs**.

❖ **Tons of exciting directions**

● **Understanding**:

- Hybrid models with a few (4-6) attention layers perform very well, even better than pure Mamba(-2) or Transformer++.
- What are these attention layers doing? Can they be replaced with another mechanism?

● **Inference optimizations**:

- There's a whole suite of optimizations tailored to Transformers, in particular handling the **KV cache** (quantization, speculative decoding).
- How would the inference landscape change if model states (e.g. SSM states) no longer scale with context length, and KV cache is no longer the bottleneck?

● **Training optimizations**:

- Though SSD might be faster than attention, Mamba-2 as a whole might still be slower than Transformers at short (e.g. 2K) sequence length, since the MLP layers in Transformers are very hardware-friendly.
- Our implementation of SSD does not specifically take advantage of new features on H100 GPUs, and we look forward to future optimizations that could make SSMs faster to train than Transformers for large-scale pretraining at 2-4K sequence length.