RECURRENT VISION IN PARALLEL: SPECTRAL CONVOLUTIONAL STATE SPACE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Brains and deep neural networks share the goal of building rich and robust visual representations but pursue it through different means. Cortical circuits route sensory signals through re-entrant feedback, refining their representations iteratively over time. Deep neural networks (DNNs), by contrast, refine theirs across layers of distinct parameters, in a single feedforward pass. This distinction becomes especially pronounced for high-dimensional signals like video, where capturing rich spatiotemporal structure either demands many sequential refinement steps or a rapid growth in model size to encode it all at once. Brains take the first route via recurrence, but its sequential nature does not translate easily to GPUs. Here, we close this gap with a parallelizable state space model (SSM) formulation that lifts convolutional recurrence into the spectral domain, where composing convolutions reduces to pointwise multiplication and yields linear-time scaling. We instantiate this idea as spectral convolutional state space models (sCSSMs): a drop-in primitive that composes with other SSM circuits including Mamba and Gated Delta Networks. A single sCSSM layer approaches published human accuracy on visual reasoning challenges where transformers fail, and parallels the time–accuracy tradeoff that humans display on these tasks. Replacing self-attention with sCSSMs on ImageNet-1K and as a LoRA adapter on a video generation model beats Standard and Transformer baselines with fewer parameters on VideoPhy-2.

1 INTRODUCTION

Modern deep neural networks (DNNs) rival the human visual system on a wide variety of tasks, ranging from object classification [29] to 3D perception [4]. They do so by stacking feedforward layers and scaling parameters, exploiting the parallelism of GPUs. Brains take a different route. Cortical circuits are organized around *recurrent connectivity* [8; 5; 33], which refines representations over time by repeatedly routing signals back through the same neurons [28; 14] (for review see [17; 16]). Recurrent connectivity in vision carries a strong inductive bias: signals spread locally across visual space, building long-range structure over time.

Take Figure 1: as a bluebird flies across the frames of a video, the pixels that matter for tracking it cluster in space and time. Transformer-based DNNs ignore this structure, letting every pixel interact with every other from the outset (Fig. 1 red). Cortical recurrence is built around this structure: interactions begin local and grow over time through repeated feedback. A computational analog is *convolutional recurrence*: a dynamic mode of processing in which interactions between units spread over time through the repeated application of convolutional kernels. Models built on this principle solve long-range dependency problems that challenge conventional DNNs [34]. But after a decade of effort to build and scale them [30; 20], they face a fundamental challenge: their sequential nature is a poor fit for the massively parallel operations of modern GPUs.

A natural way to recover GPU parallelism in convolutional recurrence is through state space models (SSMs) [12], which model linear recurrences that admit efficient parallel scans. ConvS5 [31] took this route, replacing S5's [32] dense transitions with spatial convolutions. But efficient ConvS5 implementations must compose the kernel $\boldsymbol{K}$ with itself, causing polynomial memory growth over T steps and training instability (Fig. 1 green). ConvS5 sidesteps this issue by restricting itself to 1×1 kernels—equivalent to separating space and time dynamics, and eschewing any representational benefits that might come from a spatiotemporally local bias. This reflects a broader tradeoff in vision

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

models: jointly model space and time, or scale computation. We discuss prior approaches to this tradeoff in detail in Appendix B.

We resolve this tension with a single modification: lifting convolutional recurrence into the spectral domain. By the convolution theorem, convolving kernels in the spatial domain corresponds to *pointwise multiplication* in the frequency domain, which eliminates the memory blow-up and training instability of ConvS5 at any kernel size. We call the resulting primitive a *spectral convolutional state space model* (sCSSM). sCSSMs are replacements for self-attention that integrate naturally with other SSM circuits like Mamba [9] and Gated Delta Networks [38]—realizing the brain's recurrent connectivity strategy on hardware designed for parallelism.

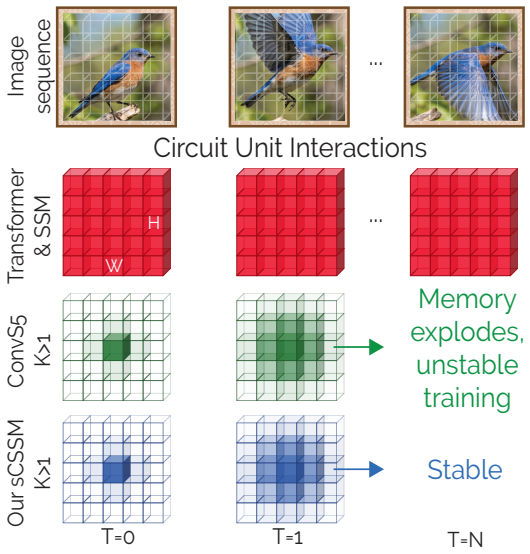


Figure 1: **sCSSMs enable parallelized convolutional recurrence.** Transformers and SSMs couple every unit to every other immediately (red), discarding spatiotemporal locality. Convolutional recurrence preserves locality by spreading interactions through composed kernels—mirroring cortical feedback—but in the spatial domain those kernels grow with depth, blowing up memory and compute (ConvS5, green) [31]. sCSSMs lift composition into the spectral domain, where convolution becomes pointwise multiplication and kernels stay fixed-size (blue). The result is a compute- and data-efficient inductive bias for images and video, e.g., the bluebird in the top row.

The sCSSM unlocks three results:

- **A parallelizable model of convolutional recurrence.** sCSSMs lift iterated convolution into the spectral domain, where it reduces to pointwise multiplication — a stable, memory-efficient drop-in for self-attention that composes with modern SSM circuits like Mamba and Gated Delta Networks.
- **Human-level visual reasoning.** A single sCSSM layer approaches published human accuracy on PathFinder [20; 15] and Path-Tracker [22] — challenges where transformers fail [34] — and parallels the human time–accuracy tradeoff.
- **Real-world vision at scale.** On ImageNet-1K, replacing self-attention with sCSSMs in an edge-class DNN matches a larger baseline at half the parameter count. As a LoRA adapter on a video generator, sCSSMs improve VideoPhy-2 scores where Standard and Transformer LoRAs degrade them.

2 Background

Our convention for the state space model follows S5's MIMO models [32]. SSM layers are simply recurrent layers without activation function. We write $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ the input and output, $\boldsymbol{h} \in \mathbb{C}^N$ the hidden state, $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ the transition matrix, $\boldsymbol{B} \in \mathbb{C}^{N\times d}$ and $\boldsymbol{C} \in \mathbb{C}^{d\times N}$ the input and output matrices, respectively. This yields the model of Eq. (1), which can be discretized as in Eq. (2).

$$\boldsymbol{h}'(t) = \boldsymbol{A}\boldsymbol{h}(t) + \boldsymbol{B}\boldsymbol{x}(t) \quad (1a) \qquad \boldsymbol{h}_t = \overline{\boldsymbol{A}}\boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}}\boldsymbol{x}_t \quad (2a)$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{h}(t) + \boldsymbol{D}\boldsymbol{x}(t) \quad (1b) \qquad \boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{h}_t + \boldsymbol{D}\boldsymbol{x}_t \quad (2b)$$

Analytic solutions exist for the discretization step, using for example the zero order hold (Eq. (3)). Recent work on the SSM showed that linearity in the recurrence allowed for efficient parallelization, achieving logarithmic depth and linear work in asymptotic time complexity (see e.g. [12; 32]), as well as the (almost) equivalence between general and diagonal SSMs, recalled in Eq. (4). Appendix D provides more detail on the notations and on the SSM.

$$\overline{\boldsymbol{A}} = e^{\Delta\boldsymbol{A}} \qquad \overline{\boldsymbol{B}} = (\Delta\boldsymbol{A})^{-1}(e^{\Delta\boldsymbol{A}} - \boldsymbol{I}) \cdot \Delta\boldsymbol{B} \quad (3)$$

$$\widetilde{\boldsymbol{A}} = \boldsymbol{\Lambda} \qquad \widetilde{\boldsymbol{B}} = \boldsymbol{P}^{-1}\boldsymbol{B} \qquad \widetilde{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{P} \quad (4)$$

3 Convolutional State Space Model

Our goal is to extend state space models to high-dimensional spatiotemporal data such as images (of size hwc) and videos (of length T). This requires combining *local interactions in space* with *efficient long-range integrations in time*. Convolutions provide a natural inductive bias for spatial locality, while SSMs offer scalable temporal dynamics through linear recurrences. Smith et al. [31] introduced ConvSSMs as a principled translation of spatiotemporal recurrent connectivity into the SSM framework, unifying spatial coupling and temporal dynamics in a single, scalable model. Rewriting the SSM (Eq. (1)) for kernel convolutions, we get the following spatial ConvSSM, where each hidden state $\boldsymbol{h}$ at position (x, y) corresponds to a MIMO SSM that is coupled to the neighboring SSMs through the convolution operator:

$$\frac{\partial \boldsymbol{h}}{\partial t}(x,y,t) = \boldsymbol{K}_h * \boldsymbol{h}(x,y,t) + \boldsymbol{K}_x * \boldsymbol{x}(x,y,t) \quad \boldsymbol{h}_t(x,y) = \overline{\boldsymbol{K}}_h * \boldsymbol{h}_{t-1}(x,y) + \overline{\boldsymbol{K}}_x * \boldsymbol{x}_t(x,y) \tag{5a}$$

$$\boldsymbol{y}(x,y,t) = \boldsymbol{K}_y * \boldsymbol{h}(x,y,t) \qquad \boldsymbol{y}_t(x,y) = \boldsymbol{K}_y * \boldsymbol{h}_t(x,y) \tag{5b}$$

Here, x and y index spatial position, $\boldsymbol{h}$ (resp. $\boldsymbol{x}$) : $x, y, t \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{R}_+ \mapsto \boldsymbol{h}(x, y, t) \in \mathbb{R}^c$ (resp. $\in \mathbb{R}^{c_{\text{in}}}$) and $\boldsymbol{K} \in \mathbb{R}^{k_1 \times k_2 \times c \times c}$ is a tensor corresponding to a convolution kernel acting across spatial position, and similarly for $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$. The spatial ConvSSM faces two key challenges, both of which are consequences of its spatiotemporal coupling.

(*Challenge A:*) It is unclear how to compute discrete-time kernels $\overline{\boldsymbol{K}}_h$ and $\overline{\boldsymbol{K}}_x$ from their continuous-time counterparts, or what such kernels would even look like. Equation (3) requires matrices and cannot be applied directly to kernels.

(*Challenge B:*) Parallel algorithms break down when adapted to the spatial ConvSSM [31] (Prop. 1).

Proposition 1 (Spatial ConvSSM Complexity)**.** *For a spatial ConvSSM with kernel size $k > 1$ and image size hwc, the parallel scan applied over T time steps has work (W) and depth (T_∞)*

$$W = T_\infty = O(T^4)$$

with a $O(k^4c^3)$ constant factor (reducible by space or channel separation in the kernel). Proof in Appendix G.3.

However, in practice, convolutional networks interleave convolution operations by projections p to ensure a bounded spatial support of the feature map. These projections allow us to bound the spatial support of the representation of the convolution kernel, allowing to retrieve an asymptotic depth complexity in $\log T$, though with an impractically large constant factor (Proposition 2).

Proposition 2 (Reduced Spatial ConvSSM Complexity)**.** *For a spatial ConvSSM with kernel size k and image size hwc, and with spatial projections keeping the feature map size constant, the parallel scan applied over T time steps has work (W) and depth (T_∞)*

$$W = O(T), \qquad T_\infty = O(\log T)$$

with a $O(hwc)$ constant factor. Proof in Appendix G.4.

These challenges of ConvSSMs are structural, not implementation artifacts. They are a direct consequence of manipulating infinite-dimensional linear operators. Projecting the state to a bounded spatial domain in between each convolution fundamentally changes the structure of the problem where the effective space is finite dimensional. Indeed, let $T_{\boldsymbol{K}}$ be the linear operator corresponding to the convolution, and p the projection to the bounded spatial support. Then, $\text{supp}(p \circ T_{\boldsymbol{K}}) < +\infty$, i.e., the recurrence can be represented with linear transitions over a finite dimensional space, and Proposition 4 applies. However, the dimensionality of this space is $(3h)(3w)c$, which means that even using a diagonal SSM, $\boldsymbol{B} \in \mathbb{R}^{(9hwc)^2}$ would be **impossible to store on a GPU in practice**. Even worse: we need the result to still have some semblance of spatial organization, we need to explicitly parameterize $\boldsymbol{P}$, which means the computation of the corresponding $\boldsymbol{C}$ requires inverting this arbitrary matrix, which costs $O((hwc)^3)$.

4 SPECTRAL CONVSSM

The next step is therefore to find another structure of convolution that is *(i)* diagonalizable using a practically computable change of basis $\boldsymbol{P}$, while *(ii)* retaining finite effective dimensionality. The natural structure we turn to is thus that of circulant convolution, diagonalized by Fourier transforms.

Start by flattening the $h \times w \times c_{\text{in}}$ input tensor into a $d = hwc_{\text{in}}$ dimensional vector. The linear operators corresponding to circulant convolutions have a matrix representation whose structure is block-circulant with circulant blocks (BCCB) (Eq. (6)), yielding an SSM with $d = hwc_{\text{in}}$-dimensional inputs and outputs and $N = hwc$-dimensional hidden states.

$$\boldsymbol{A} = \text{Circ}(\boldsymbol{K}_h \in \mathbb{R}^{k\times k\times c\times c}) \in \mathbb{R}^{N\times N} \qquad \boldsymbol{B} = \text{Circ}(\boldsymbol{K}_x) \in \mathbb{R}^{N\times d} \qquad \boldsymbol{C} = \text{Circ}(\boldsymbol{K}_y)\mathbb{R}^{d\times N} \tag{6}$$

These transition matrices are diagonalized by the Discrete Fourier Transforms (DFT) as described below. Diagonalization decouples recurrence across spatial frequency. This allows each spatial frequency to evolve independently in time while preserving spatiotemporal coupling through the (now pointwise) frequency-domain transition. This motivates the *Spectral SSM*.

Spectral State Space Models. Consider the case where $\boldsymbol{A}$ is block-circulant (not BCCB yet) with blocks of size $c\times c$. This can be viewed as a 1D convolution over $\frac{N}{c}$ positions and c channels. Let $\boldsymbol{F}_{\frac{N}{c}} \in \mathbb{C}^{\frac{N}{c}\times\frac{N}{c}}$ be the DFT matrix. Then, $\boldsymbol{A}$ is diagonalized by $\boldsymbol{P} = \boldsymbol{F} \otimes \boldsymbol{I}_c$, where $\otimes$ is the Kronecker product. It can be written as $\boldsymbol{A} = (\boldsymbol{F} \otimes \boldsymbol{I}_c)\boldsymbol{\Lambda}_c(\boldsymbol{F}^{-1} \otimes \boldsymbol{I}_c) \in \mathbb{R}^{N\times N}$, where $\boldsymbol{\Lambda}_c \in \mathbb{C}^{N\times N}$ is a block-diagonal matrix with blocks of size $c\times c$. We call the resulting SSM a Spectral State Space Model (SSSM).

The core idea of SSSMs is to constrain $\boldsymbol{A}$ not only so that it is structured after some change of basis $\boldsymbol{P}$, but also so that $\boldsymbol{P}$ itself is structured, allowing for efficient matrix-matrix and matrix-vector products. This generalizes beyond 1D Fourier transforms; for example, it still works when $\boldsymbol{F}_{\frac{N}{c}}$ is replaced by $\boldsymbol{F}_h \otimes \boldsymbol{F}_w$ for 2D convolutions. These are efficiently computable and diagonalize BCCB matrices.

$$\widetilde{\boldsymbol{A}} = \boldsymbol{\Lambda}_c \qquad \widetilde{\boldsymbol{B}} = \boldsymbol{P}^{-1}\boldsymbol{B} \qquad \widetilde{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{P} \tag{7}$$

Diagonal SSSMs. Equation (7) structures the change of basis, but powering $\widetilde{\boldsymbol{A}}$ costs $O(\frac{N}{c}c^3) = O(Nc^2)$. We recover the diagonal cost $O(N)$ (the bottleneck in parallel algorithms [13; 32]) by writing $\widetilde{\boldsymbol{A}} = \boldsymbol{Q}_c\boldsymbol{\Lambda}\boldsymbol{Q}_c^{-1}$, where $\boldsymbol{\Lambda}$ is diagonal and $\boldsymbol{Q}_c$ is block-diagonal with blocks of size $c\times c$. $\boldsymbol{Q}_c$ captures channel mixing within each frequency block in $\boldsymbol{\Lambda}_c$, leaving a pure diagonal $\boldsymbol{\Lambda}$:

$$\widehat{\boldsymbol{A}} = \boldsymbol{\Lambda} \qquad \widehat{\boldsymbol{B}} = \boldsymbol{Q}_c^{-1}\boldsymbol{P}^{-1}\boldsymbol{B} \qquad \widehat{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{P}\boldsymbol{Q}_c \tag{8}$$

Our Spectral Convolutional SSM (sCSSM). Combining Equations (6) and (8) takes three steps. **(I)** Move from convolution operators to their BCCB matrix counterpart. **(II)** Diagonalize these BCCB through the Fourier transform. We are left with block-diagonals instead of pure diagonals due to channel mixing. **(III)** Absorb the channel mixing of the recurrent state into the input and output transforms $\boldsymbol{B}$ and $\boldsymbol{C}$:

$$\boldsymbol{A} = \text{Circ}(\boldsymbol{K}_h) = \boldsymbol{P}\boldsymbol{\Lambda}_c^A\boldsymbol{P}^{-1} = \boldsymbol{P}\boldsymbol{Q}_c\boldsymbol{\Lambda}\boldsymbol{Q}_c^{-1}\boldsymbol{P}^{-1} \tag{9}$$

$$\boldsymbol{B} = \text{Circ}(\boldsymbol{K}_x) = \boldsymbol{P}\boldsymbol{\Lambda}_c^B\boldsymbol{P}_{\text{in}}^{-1} \tag{10}$$

$$\boldsymbol{C} = \text{Circ}(\boldsymbol{K}_y) = \boldsymbol{P}_{\text{in}}\boldsymbol{\Lambda}_c^C\boldsymbol{P}^{-1} \tag{11}$$

with $\boldsymbol{P} = (\boldsymbol{F}_h \otimes \boldsymbol{F}_w) \otimes \boldsymbol{I}_c$, $\boldsymbol{P}_{\text{in}} = (\boldsymbol{F}_h \otimes \boldsymbol{F}_w) \otimes \boldsymbol{I}_{c_{\text{in}}}$. $\text{Circ}(\boldsymbol{K})$ is the convolution operator with kernel $\boldsymbol{K}$, and the $\boldsymbol{\Lambda}_c^{\cdot}$ are block-diagonal matrices with blocks of size $c\times c$, $c\times c_{\text{in}}$ and $c_{\text{in}}\times c$, respectively, corresponding to the Fourier transform of the convolution kernels. This decomposition yields a diagonal SSSM, where all the computational load is absorbed by $\boldsymbol{B}$ and $\boldsymbol{C}$:

$$\boxed{\widehat{\boldsymbol{A}} = \boldsymbol{\Lambda}} \tag{12}$$

$$\widehat{\boldsymbol{B}} = \boldsymbol{Q}_c^{-1}\boldsymbol{P}^{-1}\boldsymbol{P}\boldsymbol{\Lambda}_c^B\boldsymbol{P}_{\text{in}}^{-1} = \left(\boldsymbol{Q}_c^{-1}\boldsymbol{\Lambda}_c^B\right)\boldsymbol{P}_{\text{in}}^{-1}\boxed{= \boldsymbol{\Lambda}_c^{B'}\boldsymbol{P}_{\text{in}}^{-1}} \tag{13}$$

$$\widehat{\boldsymbol{C}} = \boldsymbol{P}_{\text{in}}\boldsymbol{\Lambda}_c^C\boldsymbol{P}^{-1}\boldsymbol{P}\boldsymbol{Q}_c = \boldsymbol{P}_{\text{in}}\left(\boldsymbol{\Lambda}_c^C\boldsymbol{Q}_c\right)\boxed{= \boldsymbol{P}_{\text{in}}\boldsymbol{\Lambda}_c^{C'}} \tag{14}$$

This formulation resolves both challenges mentioned above. *Challenge A* arose because discretized kernels have full spatial support, breaking the kernel locality that makes spatial convolutions tractable and efficient. The sCSSM trivializes the discretization process through diagonalization, and replaces expensive convolutions with pointwise multiplications in the spectral domain. *Challenge B* also boils down to kernel size explosion, and is bypassed by the spectral formulation for the same reason: composition becomes pointwise multiplication, and kernel representations stay fixed-size.

Proposition 3 (sCSSM Complexity)**.** *For a spectral ConvSSM with kernel size k and image size hwc, the parallel scan applied over T time steps has work and span*

$$W(\text{Scan}) = O(T), \qquad T_\infty(\text{Scan}) = O(\log T)$$

with a $O(hwc)$ constant factor. Proof in Appendix G.5.

And this time, contrary to Proposition 2, the asymptotic complexity does not come at the cost of a huge constant precomputing step.

Summary. The sCSSM computes arbitrary circulant convolutions in $\mathcal{O}(\log T)$ by leveraging the special structure afforded by circularity, combined with careful changes of basis and state equivalences. This turns the convolutional transitions (responsible for the $\mathcal{O}(T^4)$ cost) into tractable linear ones (responsible for the $\mathcal{O}(\log T)$ cost).

4.1 Relation to previous work

The restricted structure of convolution chosen by ConvS5 [31] to achieve that same computational complexity is to simply not model spatiotemporal interactions between units: they use 1×1 convolutions in Equation (5) to restrict the transition operator to act independently at each spatial location. This means that the operator $T_{\boldsymbol{K}}$ is already diagonal, and therefore, the dimensionality of the state is limited to the support of the input, sidestepping the need for all the derivation of the sCSSM. But it comes at the cost of removing the very interactions that convolutional recurrences were built to model.

In contrast, our approach makes no compromises on what actually matters. It explicitly models spatiotemporal coupling, disentangling it through the spectral domain while retaining the computational benefits of structured SSMs. Instead, the compromise is to impose a circulant structure, whose boundary condition may not be entirely realistic. A comparison with the broader, overall landscape of architectures for video modeling is available in Appendix B.

4.2 Composition with modern SSM design

The sCSSM replaces the standard linear recurrence with its convolutional spectral counterpart while leaving the surrounding architectural design unchanged. As a result, any method that modifies the SSM transition, input, or output operators can be directly applied on top of our formulation. This includes diagonal parameterizations [11; 13], input-dependent dynamics [9], and attentional mechanisms [7; 38].

Modularity is a key strength of our approach: improvements in SSM design immediately transfer to the spatiotemporal setting without additional constraints through our sCSSM. We investigate three representative circuits in our experiments: (*i*) the minimal baseline sCSSM core; (*ii*) a Mamba-version [9]; and (*iii*) a Gated Delta Network (GDN)-version [38].

5 Experiments

We evaluate sCSSMs along four axes: their computational properties as a primitive (§5.1), their ability to learn human-level visual reasoning on controlled stimuli (§5.2), their behavior as a self-attention replacement on natural images (§5.3), and their utility as a parameter-efficient adapter for video generation (§5.4).

Setup. All experiments ran on NVIDIA B200 GPUs (single-GPU except ImageNet, which used 2×B200 with `pmap`). Step-time benchmarks, visual reasoning, and ImageNet were implemented

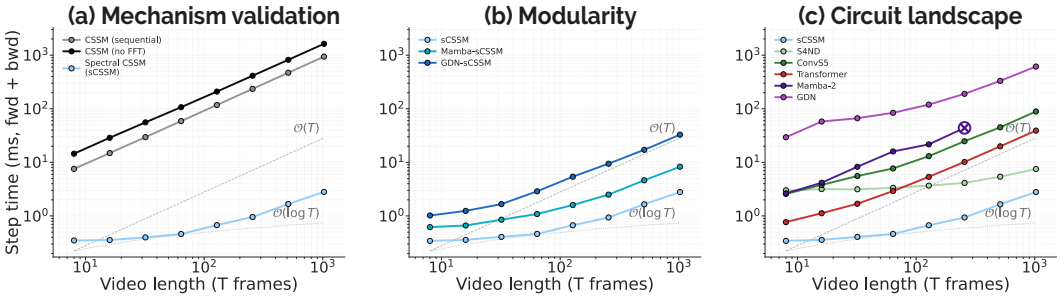


Figure 2: **sCSSMs are efficient architectures for visual data.** Step-time of sCSSMs, SSMs, and Transformers on fp32 tensors of shape (C, T, H, W) with $C, H, W = 32$ and T swept from 10^1 to 10^3. **(a)** The Sequential CSSM uses a sequential (not associative) scan; the Spatial CSSM uses a parallel scan in the spatial domain. These are controls for the sCSSM. **(b)** sCSSMs compose seamlessly with modern SSM circuits (Mamba, GDN). **(c)** Against existing visual sequence models, **sCSSMs achieve state-of-the-art scaling on long-range video**. ConvS5 and all other s/CSSMs use 11×11 kernels; $\otimes$ denotes GPU OOM. All measurements are the median over 100 timings.

in JAX/Flax; VideoPhy-2 used PyTorch on CogVideoX-5B [39] with the official VideoPhy-2 [2] mPLUG-Owl autograder. Total compute was ∼810 B200 GPU-hours; per-phase breakdown in Appendix E.1. Code: `https://anonymous.4open.science/r/CSSM-2669`.

Step-time benchmarks. We measure forward+backward step time of single-layer implementations after JIT compilation, median over 100 timings on random data with $C = 32$ embedding dim and 11×11 kernels for all CSSMs. SSMs use the same JAX associative scan (no custom CUDA kernels); Transformers use Flash Attention.

5.1 Validating the Spectral Mechanism

Spatial ConvSSMs scaled linearly in T, but had an intractable constant—$\approx 10\times$ slower than the sequential algorithm for all $T \in [\![10^1, 10^3]\!]$. Though not quartic, this behavior is as expected since we are in the circular convolution case. Kernels have a big maximum size. In contrast, sCSSMs exhibited logarithmic-time behavior until reaching hardware parallelization limits at $T = 128$ and converged towards linear scaling. At $T = 1024$, we reach $\approx 100\times$ the computational efficiency of the sequential algorithm. This matches our theoretical analysis and confirms that our spectral formulation removes the computational bottleneck (Fig. 2a).

Compatibility with Modern SSM Circuits We merged sCSSMs with representative SSM circuit types, Mamba and Gated Delta Networks (GDN). See Appendix E for more details. The scaling behavior and performance trends of the sCSSM are preserved across the Mamba and GDN variants—logarithmic span until reaching hardware limits, then linear work. Mamba-sCSSM and GDN-sCSSM are slower than the plain sCSSM by constant step-time factors of $\approx 3\times$ and $\approx 12\times$ across all T, with the same scaling curve (Fig. 2b). In other words, **the spectral ConvSSM composes seamlessly with existing designs**, combining their benefits in expressivity with the capability to learn spatiotemporal recurrent connections.

Benchmarking Against Standard Architectures We compared the step time scaling of the sCSSM with a variety of architectures which are typically used for modeling vision data: Transformers, a variety of SSMs (which flatten the input signal into a sequence), and the ConvS5 [31]. sCSSMs outperform any of these other architectures in our experiments. Indeed, our naive JAX implementation is $\approx 14\times$ faster than the transformer at $T = 1024$ (Fig. 2c).

5.2 Visual Reasoning

Recurrent connections have long been associated with perceptual grouping [27]. For example, visually tracing an unfamiliar path in the woods, or tracking a single bird in its flock by following its motion. Both cases require a visual system that can bind parts of space together into whole objects based on features like oriented lines, textures, and motion. The PathFinder [20] and PathTracker [22]



Figure 3: **Visual reasoning challenges for evaluating neural circuits. (a)** The PathFinder challenge [20; 15] evaluates long-range spatial reasoning capabilities. **(b)** The PathTracker challenge [22] is its spatiotemporal extension (white circles added to PathTracker for clarity). Both tasks are easy for humans but hard for transformers.

challenges distill these capabilities into controlled stimuli, and remain easy for humans but extremely challenging for dominant machine vision architectures like Transformers [34]. We asked whether the sCSSM, which is designed to learn recurrent connections like those thought to underlie perceptual grouping, could close this gap.

PathFinder (Fig. 3a) asks whether two endpoints lie on the same long contour amid distractor segments. We use the 128×128 px PathFinder-14 variant (paths of 14 dashes), with 160K/20K/20K training/validation/test images. Spatiotemporal models like the sCSSMs, ConvS5, and S4ND were presented a sequence of each image repeated $T=8$ times to drive the temporal recurrence; static image models like the Transformer, Mamba-2, and GDN viewed the original images.

PathTracker (Fig. 3b) is a spatiotemporal variant of PathFinder, in which the viewer must track a single dot across $T=32$ frames as it passes through a swarm of 15 distractors, and say whether it lands in the blue square or not. We used 20K training/validation/test videos with 32×32 px frames.

Experimental design. To fairly compare different circuits, we placed them in a standardized model architecture. This architecture had three parts: a *convolutional stem*, the *circuit*, and a *readout*. The stem had convolutional kernels of size 3×3 for PathFinder and 1×1 for PathTracker, followed by a softplus pointwise nonlinearity, and a LayerNorm (computed across timesteps for spatiotemporal models). The stem for PathFinder additionally had a spatial max pool with kernel and stride size of 2×2. Finally, we introduced a RoPE position encoding before passing activities to the circuit. The spatial map on the final (or only) timestep of circuit activity was then passed to a readout, consisting of a LayerNorm, SoftPlus, spatial max pool, LayerNorm, and a linear mapping to a binary classification decision.

Within this fixed scaffold, we swapped in representative architectures from four families: (*i*) standard transformers and spatiotemporal transformers (Transformer-ST), (*ii*) 1D SSM circuits (Mamba-2 [7], GDN [38]), (*iii*) spatiotemporal SSMs (S4ND [25], ConvS5 [31]), and (*iv*) our sCSSM, including its Mamba- and GDN-composed variants and ablations that remove the spatial or temporal aspects of the recurrence. Each architecture was swept over a grid of factors to map the accuracy/efficiency tradeoff (embed dim, state dim, depth; additional training details in Appendix E).

Models were trained with AdamW (peak LR 10^{-3}, batch 128) for up to 120 epochs with early stopping on validation loss. Early stopping used only the validation split. The selected checkpoint was evaluated once on the held-out test split. We report the resulting test accuracy, training-step time, and circuit parameter count. Human-level performance was extracted from the original papers [15; 22].

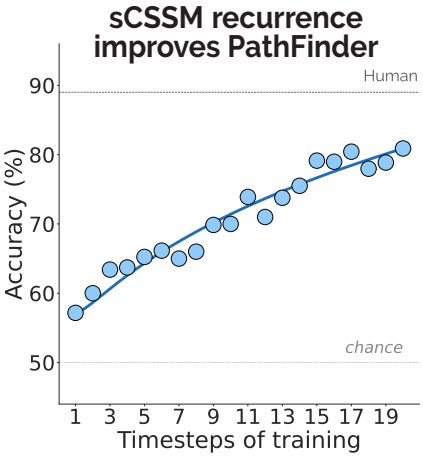


Figure 4: **sCSSMs have a time-performance tradeoff on PathFinder, much like humans do.**

sCSSMs show a human-like time-accuracy trade-off. A classic result in perceptual grouping is that the time it takes for human observers to accurately answer if two dots lie on the same line or not correlates with

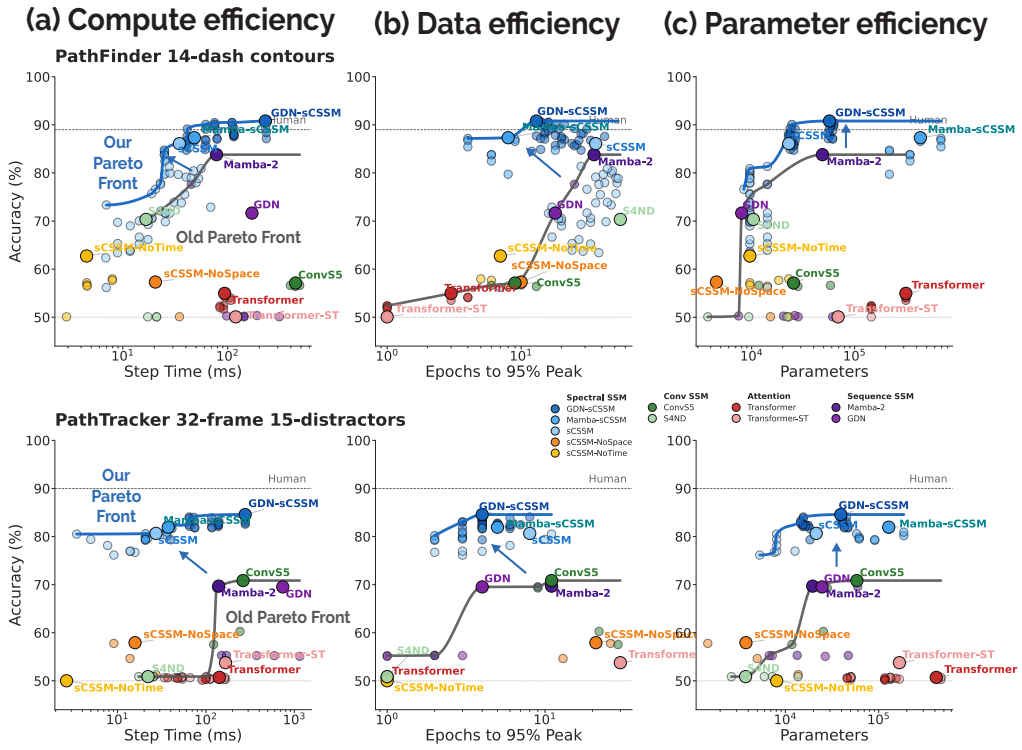


Figure 5: **Our sCSSMs establish a new Pareto front on both tasks across all three measures of efficiency.** We plot model accuracy against three complementary aspects of efficiency. Pareto fronts are drawn as the upper envelope across the depth/embed/state-dim sweep for cSSMs and non-cSSMs.

the length of the line. Human psychophysics experiments with PathFinder have exhibited a similar trade-off. We began by investigating if the sCSSM also shows such a trade-off. If it does, that would imply that its recurrent connections are capable of learning an incremental and human-like perceptual grouping strategy for solving PathFinder. We investigated this question by training versions of the sCSSM for all $T \in [\![1, 20]\!]$ and recording the accuracy of each (Fig. 4). Indeed, we found that the model's accuracy grew as it was given more time to respond, approaching published human accuracy by $T = 20$ (timesteps *vs.* accuracy, $\rho = 0.96$, $p < 0.001$).

sCSSMs establish a new Pareto front in accuracy and efficiency. We next set out to determine whether the sCSSM's performance came at a cost in efficiency, or whether the sCSSM could match or beat existing architectures across the board. Across both tasks, sCSSMs and their Mamba- and GDN-composed variants traced a new Pareto front in accuracy vs. task step time (Fig. 5). We found similar improvements in the trade-off between model accuracy, sample efficiency, and parameters: once again, the sCSSM family of models pushed the frontier. We use the GDN variant as the default circuit for downstream experiments because it achieved the strongest performance on these tasks.

We also validated the role of spatial kernels and temporal recurrence in these experiments with ablation experiments of the sCSSM architecture. Specifically, we verified that the sCSSM's performance depends on having an intact capacity for spatial and temporal processing: giving it 1×1 spatial kernels (sCSSM-NoSpace) or 1-timestep of recurrent processing (sCSSM-NoTime) dropped its performance to near-chance on both challenges.

5.3 ImageNet Classification

We next tested whether sCSSMs extend to natural vision settings at scale on ImageNet-1K object classification. ImageNet-1K contains 1.28M training images and 50K validation images across 1000 classes at 224×224 px. Although ImageNet does not exercise the temporal dimension of the sCSSM, it remains the standard test of how a vision architecture scales to natural images.

Experimental design. We started from the edge-scale Single-Head Vision Transformer (SHViT [40]), a hierarchical ViT available in multiple sizes (we focus on the 6M and 11M parameter variants). SHViT was chosen for two reasons: (*i*) its single-head attention design isolates

the block we are replacing, and (*ii*) at this scale a full 300-epoch run completes relatively quickly on 2× B200 GPUs (≈40 h per model).

We replaced the self-attention layers in the SHViT with GDN-sCSSM layers. We chose the GDN version because it outperformed the other sCSSM variants on PathFinder and PathTracker. We matched the input/output embedding dimension at each layer and left the surrounding architecture, augmentation, and optimizer untouched. Both models were trained with the TIMM [37] recipe (AdamW, peak LR 10^{-3}, weight decay 5×10^{-2}, RandAugment, Mixup, CutMix, RandomErasing, EMA 0.9998, label smoothing 0.1, drop-path 0.1) for 300 epochs at batch size 512; full details in Appendix E. To test whether the sCSSM benefits from temporal recurrence even on static inputs, we swept $T \in \{1, 4, 8\}$. Images were repeated to drive the temporal recurrence. As a control, we additionally train a version of the SHViT with ConvNeXt-style convolution kernels (Hybrid SHViT-S1; Appendix E).

sCSSMs match attention with fewer parameters. Despite the static image processing setting, the sCSSM variant of the SHViT-S1 outperformed the published version with fewer parameters, reaching the level of SHViT-S2 with just over *half* of its total parameters (Fig. A1a). Performance also improved monotonically with T, suggesting that even on static images, the sCSSM's recurrence iteratively refines spatial representations in a way that single-pass self-attention cannot. Or put differently, recurrent timesteps in the sCSSM substitute for processing depth in feedforward models.

5.4 Video Generation on VideoPhy-2

sCSSMs are naturally suited to video, where their recurrent connections propagate over both space and time without needing to repeat the input as we did for ImageNet and PathFinder. We therefore turned to the VideoPhy-2 benchmark [2], which scores generated clips on Semantic Adherence (SA: does the video match the prompt?) and Physical Commonsense (PC: does the video obey physics?) using a trained mPLUG-Owl autograder. A joint metric counts a clip as a success only if both SA and PC scores are ≥ 4 out of 5.

Experimental design. We fine-tuned the 5B text-to-video CogVideoX-5B [39] using low-rank adaptation (LoRA) on a curated 14,857-clip subset of OpenVid-1M [24]. We compared the unadapted base CogVideoX-5B to three rank-16 LoRA variants — Standard, Transformer (multi-head temporal self-attention), and GDN-sCSSM — injected into self-attention $Q/K/V/O$ projections (cross-attention untouched). All shared an identical training and inference protocol; see Appendix E.4.

sCSSM LoRAs correct physical-reasoning failures where Transformer LoRAs do not. Both the standard LoRA and the Transformer LoRA degraded the joint VideoPhy-2 score relative to the unadapted CogVideoX-5B base, indicating that naively augmenting self-attention disrupts the model's learned dynamics. In contrast, the GDN-sCSSM LoRA improved on the base model with 12–13% fewer trainable parameters than the other LoRAs (Fig. A1b), correcting errors in physical consistency and object dynamics that are visible in side-by-side generation comparisons (Fig. A1c, purple boxes highlight the baseline model's errors in physics and appearance).

6 Discussion

We introduced the sCSSM: an architectural primitive that lifts convolutional recurrence into the frequency domain so that composed kernels collapse to pointwise multiplication. This single change resolves two structural obstacles that had confined prior convolutional SSMs to spatially decoupled ($k = 1 \times 1$) recurrence: discretization no longer requires evaluating powers of a spatially expanding kernel, and parallel scans no longer suffer from a blow-up in kernel-size. The result is a replacement for self-attention, designed from the bottom up for high-dimensional data like video, which can be composed with any other SSM circuit (like Mamba or GDNs) for additional performance improvements.

Beyond machine learning, these results have direct implications for computational neuroscience. Visual cortex is fundamentally organized around recurrent, locally structured interactions evolving over time. By making such dynamics tractable at scale, our framework provides a concrete path toward building and evaluating models of cortical computation at scale. More broadly, these results suggest that aligning architectural principles with the structure of natural signals need not come at the expense of scalability.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript for readability, including correcting typos and grammar. All authors reviewed the final text, and we take responsibility for the full content of this work, including any text produced with the aid of generative AI.

REFERENCES

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.

[2] Hritik Bansal, Clark Peng, Yonatan Bitton, Roman Goldenberg, Aditya Grover, and Kai-Wei Chang. VideoPhy-2: A challenging action-centric physical commonsense evaluation in video generation. In *International Conference on Learning Representations*, 2026.

[3] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning*, 2021.

[4] Tyler Bonnen, Jitendra Malik, and Angjoo Kanazawa. Human-level 3D shape perception emerges from multi-view learning. *arXiv preprint arXiv:2602.17650*, 2026.

[5] William H. Bosking, Ying Zhang, Brett Schofield, and David Fitzpatrick. Orientation selectivity and the arrangement of horizontal connections in tree shrew striate cortex. *Journal of Neuroscience*, 17(6):2112–2127, 1997.

[6] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. OpenAI research blog post, 2024. URL https://openai.com/index/video-generation-models-as-world-simulators/. Companion technical report accompanies the Sora announcement.

[7] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the International Conference on Machine Learning*, 2024.

[8] Charles D. Gilbert and Torsten N. Wiesel. Columnar specificity of intrinsic horizontal and corticocortical connections in cat visual cortex. *Journal of Neuroscience*, 9(7):2432–2442, 1989.

[9] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Proceedings of the First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2. Also arXiv:2312.00752.

[10] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[11] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[12] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

[13] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[14] Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B. Issa, and James J. DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature Neuroscience*, 22(6):974–983, 2019.

[15] Junkyung Kim, Drew Linsley, Kalpit Thakkar, and Thomas Serre. Disentangling neural mechanisms for perceptual grouping. In *International Conference on Learning Representations*, 2020.

[16] Gabriel Kreiman and Thomas Serre. Beyond the feedforward sweep: Feedback computations in the visual cortex. *Annals of the New York Academy of Sciences*, 1464(1):222–241, 2020.

[17] Victor A. F. Lamme and Pieter R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, 23(11):571–579, 2000.

[18] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. VideoMamba: State space model for efficient video understanding. In *European Conference on Computer Vision*, 2024.

[19] Shufan Li, Harkanwar Singh, and Aditya Grover. Mamba-ND: Selective state space modeling for multi-dimensional data. In *European Conference on Computer Vision*, 2024.

[20] Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[21] Drew Linsley, Alekh Ashok, Lakshmi Narasimhan Govindarajan, Rex Liu, and Thomas Serre. Stable and expressive recurrent vision models. In *Advances in Neural Information Processing Systems*, volume 33, pages 10456–10467, 2020.

[22] Drew Linsley, Girik Malik, Junkyung Kim, Lakshmi Narasimhan Govindarajan, Ennio Mingolla, and Thomas Serre. Tracking without re-recognition in humans and machines. In *Advances in Neural Information Processing Systems*, volume 34, pages 19473–19486, 2021.

[23] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. In *International Conference on Learning Representations*, 2023.

[24] Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. OpenVid-1M: A large-scale high-quality dataset for text-to-video generation. In *International Conference on Learning Representations*, 2025.

[25] Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. S4ND: Modeling images and videos as multidimensional signals with state spaces. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[26] Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the International Conference on Machine Learning*, 2023.

[27] Pieter R. Roelfsema. Cortical algorithms for perceptual grouping. *Annual Review of Neuroscience*, 29:203–227, 2006.

[28] Pieter R. Roelfsema, Victor A. F. Lamme, and Henk Spekreijse. Object-based attention in the primary visual cortex of the macaque monkey. *Nature*, 395(6700):376–381, 1998.

[29] Vaishaal Shankar, Rebecca Roelofs, Horia Mania, Alex Fang, Benjamin Recht, and Ludwig Schmidt. Evaluating machine accuracy on ImageNet. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8634–8644. PMLR, 2020.

[30] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, volume 28, pages 802–810, 2015.

[31] Jimmy T. H. Smith, Shalini De Mello, Jan Kautz, Scott Linderman, and Wonmin Byeon. Convolutional state space models for long-range spatiotemporal modeling. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[32] Jimmy T. H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

[33] Dan D. Stettler, Aniruddha Das, Jean Bennett, and Charles D. Gilbert. Lateral connectivity and contextual interactions in macaque primary visual cortex. *Neuron*, 36(4):739–750, 2002.

[34] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. arXiv:2011.04006.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[36] Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[37] Ross Wightman. PyTorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[38] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

[39] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, 2025.

[40] Seokju Yun and Youngmin Ro. SHViT: Single-head vision transformer with memory efficient macro design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5756–5767, 2024.

[41] Xiangdong Zhang, Jiaqi Liao, Shaofeng Zhang, Fanqing Meng, Xiangpeng Wan, Junchi Yan, and Yu Cheng. VideoREPA: Learning physics for video generation through relational alignment with foundation models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=oHjLfABsK4.

[42] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision Mamba: Efficient visual representation learning with bidirectional state space model. In *Proceedings of the International Conference on Machine Learning*, 2024.

A Additional Figure

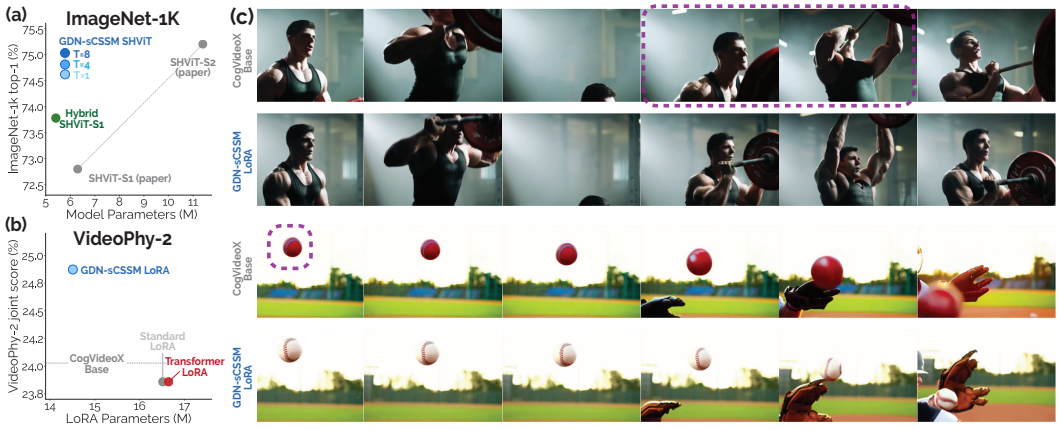


Figure A1: **sCSSMs scale to image and video processing benchmarks. (a)** ImageNet-1K top-1 accuracy versus model parameters. Replacing self-attention in SHViT [40] with GDN-sCSSM blocks (swept over $T \in \{1, 4, 8\}$) matches SHViT-S2 at roughly half its parameter count; the Hybrid SHViT-S1 control isolates the contribution of recurrence from convolutional bias alone. **(b)** VideoPhy-2 [2] joint score versus trainable parameter count for LoRA adapters on CogVideoX-5B [39]. The GDN-sCSSM LoRA improves on the base model with 12–13% fewer parameters than the Transformer and Standard LoRAs, both of which degrade it. **(c)** Side-by-side generations: base CogVideoX-5B (top) and GDN-sCSSM LoRA (bottom). Dashed purple outlines mark errors in generating the physical interactions and appearances of objects in the baseline that the LoRA corrects.

B Extended Related Work

Recurrent architectures based on local interactions, such as the ConvLSTM [30] or the horizontal Gated Recurrent Unit (hGRU) [20], demonstrated that these connections could be incorporated into a deep learning architecture and trained end-to-end to solve multiple perceptual challenges [21]. But like all nonlinear RNNs, they remain inherently sequential, rendering training slow and poorly suited for the capabilities of GPUs and TPUs.

Transformers [35] offer a parallel alternative, but implement coupling through dense attention, which lacks appropriate inductive bias for structured interactions and incurs quadratic complexity in the number of tokens. As a result, models that preserve dense spatiotemporal coupling remain fundamentally constrained in their scalability [6; 39; 36]. More scalable alternatives such as TimeSformer [3] and ViViT [1] factorize attention across space and time.

State space models (SSMs) have recently emerged as a powerful alternative to attention-based architectures for long-range sequence modeling, combining linear recurrent dynamics with efficient parallel algorithms. Early work on structured SSMs builds on the HiPPO framework [10], which introduced principled continuous-time representations of long-range dependencies, and culminates in S4 [12], where carefully parameterized transition matrices together with a convolutional representation and parallel algorithms enable efficient modeling of long sequences. Subsequent work has further simplified and structured SSMs along multiple axes to improve efficiency and scalability. Diagonal parameterizations [11; 13] reduce computational cost while retaining performance, while extensions from single-input single-output to multi-input multi-output [32] formulations were enabled by efficient parallelization through scan algorithms.

Additional refinements such as linear recurrent units (LRUs) [26] and gating mechanisms [23] further enhance expressivity and stability. Further circuit derivations and refinements around the core of SSMs include input-dependent selection of transition matrices in Mamba [9], and the SSM–attention duality in Mamba-2 [7] through the lens of semiseparable matrices. However, these advances remain fundamentally limited to one-dimensional sequence modeling and do not natively capture spatial structure.

Extending SSMs to high-dimensional visual data such as images and videos requires incorporating dynamics over multiple axes. S4ND [25] introduces precisely this extension, formalising SSMs evolving over multiple dimensions through separable dynamics. Though successful as a replacement for state-of-the-art visual processing layers, S4ND's fundamental separation between space and time limits its ability to capture spatiotemporal coupling. Other SSM-based approaches to vision and video modeling have emerged [42; 18; 19].

True spatiotemporal coupling was formulated in Smith et al. [31]. Their ConvS5 beats ConvLSTMs and Transformers on long-horizon video modeling. However, ConvS5's convolutional recurrence is limited to 1×1 kernels, thereby forfeiting spatiotemporal coupling of the ConvSSM. This is not a mere design choice, but a fundamental consequence of parallelising over coupled spatiotemporal dynamics. In contrast, our Spectral ConvSSM formulation explicitly models spatiotemporal coupling through the spectral domain while retaining the computational benefits of structured state space models.

C Scope of empirical evaluation

Our from-scratch evaluations use controlled visual-reasoning datasets and ImageNet-1K, while our video-generation experiments adapt a pretrained model using LoRA. These experiments do not establish whether the observed accuracy–efficiency advantages persist when training large models from scratch on frontier-scale datasets. Our timing results are also specific to the evaluated hardware, precision, and implementations. Favorable asymptotic complexity does not by itself establish competitive end-to-end throughput against highly optimized alternatives; evaluating this tradeoff at larger scales remains future work.

D Extended background

D.1 Notations

Linear Algebra. Let $\mathbb{K}$ be a field. We identify $\mathcal{M}_{m,n}(\mathbb{K})$ to $\mathbb{K}^{m \times n}$, the set of $m \times n$ matrices with coefficients in $\mathbb{K}$. We denote vectors in bold lowercase $\boldsymbol{x} \in \mathbb{K}^n$ and linear operators in bold uppercase $\boldsymbol{A} \in \mathbb{K}^{m \times n}$. Scalars are denoted with non bold letters. We write $e^{\boldsymbol{A}} = \sum_{k=0}^{\infty} \frac{\boldsymbol{A}^k}{k!}$ the matrix exponential, and $\sigma(\boldsymbol{A})$ the spectrum of $\boldsymbol{A}$.

Special Matrices. $\boldsymbol{I}_n$ is the identity matrix of size n. $\mathrm{Circ}(\boldsymbol{K}) \in \mathbb{R}^{n \times n}$ denotes the circulant matrix generated by the vector $\boldsymbol{K} \in \mathbb{R}^n$, possibly zero padded. $\mathrm{diag}(\boldsymbol{\Lambda})$ is a diagonal matrix with diagonal $\boldsymbol{\Lambda} \in \mathbb{K}^n$. For ease of notation, we often identify $\boldsymbol{\Lambda}$ and $\mathrm{diag}(\boldsymbol{\Lambda})$. $\boldsymbol{\Lambda}_c$ is a block-diagonal matrix with blocks of size $c \times c$. Any matrix $\cdot_c$ notation is block diagonal with blocks of size $c \times c$. $\boldsymbol{F}_n \in \mathbb{C}^{n \times n}$ is the Discrete Fourier Transform (DFT) matrix of size n.

State Space Models. For the SSMs, there are two main conventions in the literature, mostly separated by lineage from S4 [12] or S5 [32]. The convention we adopt is the latter, where SSMs are MIMO (multiple-input multiple-output) rather than SISO (single-input single-output), essentially meaning that our SSM is a standard RNN with a linear activation function.

We write $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ the input and output, $\boldsymbol{h} \in \mathbb{C}^N$ the hidden state, $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ the transition matrix, $\boldsymbol{B} \in \mathbb{C}^{N \times d}$ and $\boldsymbol{C} \in \mathbb{C}^{d \times N}$ the input and output matrices, respectively. Note that for generality, and given the changes of basis involved in this work and throughout the literature, we assume $\boldsymbol{h}$ to be complex valued in general, with the corresponding transformations consequentially also complex valued.

These shapes do not apply to Convolutional SSMs by default, as we often go back and forth between *(i)* multidimensional tensors and convolution kernels, and *(ii)* flattened vectors and the linear operator corresponding to the convolution. Whenever we use convolution kernels, the shape convention is *(i)*, and whenever we use $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ matrices, the shape convention is *(ii)*. To avoid ambiguity, we annotate the shapes on the fly.

The $\widetilde{\cdot}$ and $\widehat{\cdot}$ notations are used for equivalent formulations of the SSM, while the $\overline{\cdot}$ notation is used for discrete-time versions of continuous-time SSMs. Two SSMs are equivalent if they produce the same output and their hidden states are linearly related. For instance: $\boldsymbol{A}$, $\widetilde{\boldsymbol{A}}$ and $\widehat{\boldsymbol{A}}$ denote equivalent formulations of a continuous-time SSM while $\overline{\boldsymbol{A}}$ denotes the discrete-time version of $\boldsymbol{A}$. T denotes the sequence length in the discrete-time setting, and Δ the time step for the discretizations.

Operators. As for operators, we use $\otimes$ the Kronecker product, $\circledast$ the block convolution operator, which takes two convolution kernels and produces a new kernel corresponding to the composition of the two convolutions. The size of the resulting kernel is the sum of the sizes of the input kernels minus one. $*$ is the convolution operator, which takes a convolution kernel and an input and produces the convolution of the input with the kernel.

D.2 State Space Model

A state space model (SSM) is a class of recurrent models with linear dynamics. The continuous-time formulation (Eq. (1)) discretizes into a linear recurrence (Eq. (2)), accommodating arbitrary sampling rates. We omit the residual connection $\boldsymbol{D}$ throughout for simplicity.

$$\boldsymbol{h}'(t) = \boldsymbol{A}\boldsymbol{h}(t) + \boldsymbol{B}\boldsymbol{x}(t) \quad (15a) \qquad \boldsymbol{h}_t = \overline{\boldsymbol{A}}\boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}}\boldsymbol{x}_t \quad (16a)$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{h}(t) + \boldsymbol{D}\boldsymbol{x}(t) \quad (15b) \qquad \boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{h}_t + \boldsymbol{D}\boldsymbol{x}_t \quad (16b)$$

Discretization. Various rules can be used for discretization, such as the zero-order hold (ZOH), which yields the discrete-time transition matrices below (Δ is the time step):

$$\overline{\boldsymbol{A}} = e^{\Delta \boldsymbol{A}} \qquad \overline{\boldsymbol{B}} = (\Delta \boldsymbol{A})^{-1}(e^{\Delta \boldsymbol{A}} - \boldsymbol{I}) \cdot \Delta \boldsymbol{B} \tag{17}$$

Parallelization. The crucial property of linear recurrences is that they allow for efficient parallel algorithms such as the parallel scan. This particular algorithm relies on associativity to achieve linear work and logarithmic span [32] (Prop. 4).

Proposition 4 (SSM Complexity)**.** *For a linear SSM with hidden dimension N, the parallel scan applied over T time steps has work (W) and span (T_∞)*

$$W(\text{Scan}) = O(T), \qquad T_\infty(\text{Scan}) = O(\log T),$$

with a $O(N^3)$ constant factor (reducible to $O(N)$ for diagonal SSMs, see Equation (4)).

Proof. See Appendix G.2. Key observation: scan steps have constant cost. □

Diagonal SSMs. In the case of scalar inputs and the convolutional representation, Gu et al. [11] show that if the transition matrix $\boldsymbol{A}$ is well conditioned, computing the convolution kernel only requires knowing its spectrum $\sigma(\boldsymbol{A})$. More generally, for any vector-valued input SSM, with $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{-1}$ diagonalizable, we can rewrite the SSM as follows:

$$\widetilde{\boldsymbol{A}} = \boldsymbol{\Lambda} \qquad \widetilde{\boldsymbol{B}} = \boldsymbol{P}^{-1}\boldsymbol{B} \qquad \widetilde{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{P} \tag{18}$$

These two formulations are equivalent in the sense that they produce the same output $\widetilde{\boldsymbol{y}}(t) = \boldsymbol{y}(t)$, with hidden states linearly related by $\widetilde{\boldsymbol{h}}(t) = \boldsymbol{P}^{-1}\boldsymbol{h}(t)$. The diagonal formulation allows to reduce the constant factors from $O(N^3)$ to $O(N)$, as shown in [13].

E Architecture and Training Details

E.1 Compute budget

All training, inference, and scoring runs on NVIDIA B200 GPUs (180 GB HBM3e). Sweeps are sequential (at most one run per GPU at a time) so the OOM-retry harness can attempt halved batch sizes on an empty GPU. The full experimental campaign reported in this paper consumed approximately 810 B200 GPU-hours in total: $\approx$ 360 for the step-time benchmarks and visual

reasoning sweeps (PathTracker, PathFinder-14, PathFinder-25, timestep ablations, S4ND and ConvS5 controls; $\approx$ 297 runs across 9 sweep scripts), $\approx$ 240 for ImageNet-1K experiments (300-epoch sweep at $\sim$ 40 h per model on 2$\times$ B200 with `pmap`), and $\approx$ 210 for VideoPhy-2 experiments (latent precompute $\sim$ 22 h, training $\sim$ 31 h across 6 LoRA variants, generation $\sim$ 154 h for 7×591 clips, autograding $\sim$ 3 h). Counting earlier failed iterations on the wrong training set and two re-scoring rounds before settling on the upstream autograder, the full project consumed $\approx$ 1.5–2$\times$ the reported budget.

E.2 sCSSM BLOCK: PARALLEL SCAN AND STABILITY

We describe the per-block forward pass at the level of detail required to reproduce the implementation. The construction is the diagonal SSSM of Section 4 specialised to per-channel (depthwise) spatial recurrence: each $c \times c$ block of $\mathbf{\Lambda}_c$ collapses to 1×1 by construction, so $\mathbf{\Lambda}$ is already diagonal in the rfft2 basis without invoking the change-of-basis $\boldsymbol{Q}_c$. We work with the spectral tensors $\widehat{\boldsymbol{U}}_t = \mathrm{rfft2}(\boldsymbol{x}_t)$, $\widehat{\boldsymbol{S}}_t = \mathrm{rfft2}(\boldsymbol{h}_t)$, and $\widehat{\boldsymbol{K}} = \mathrm{rfft2}(\mathrm{pad}_{H,W}(\boldsymbol{K}))$, all in $\mathbb{C}^{C\times H\times(W/2+1)}$, where $\boldsymbol{K} \in \mathbb{R}^{C\times k\times k}$ is a learned depthwise real-space kernel of side k (default $k = 11$, corresponding to $\boldsymbol{K}_h$ from Section 4 under the depthwise restriction). $\widehat{\boldsymbol{K}}$ collects the per-frequency entries that populate $\mathbf{\Lambda}$ in Eq. (12); $\widehat{\boldsymbol{U}}_t$ is the spectral image of the input that $\widehat{\boldsymbol{B}}$ acts on; and $\widehat{\boldsymbol{S}}_t$ is the corresponding spectral hidden state.

All reported sCSSM accuracy and timing numbers use the default precision configuration: float32 parameters and activations, complex64 spectral scan. We do not use mixed-precision paths (bf16 matmul, complex32 split-real/imag scans) or fused-spectral-state optimizations, though in pilot tests we find those achieve similar accuracies.

Per-frequency recurrence. Because $\boldsymbol{K}$ is depthwise, the recurrence decouples elementwise across channels and spatial frequencies:

$$\widehat{\boldsymbol{S}}_t = \widehat{\boldsymbol{K}} \odot \widehat{\boldsymbol{S}}_{t-1} + \widehat{\boldsymbol{U}}_t, \tag{19}$$

where $\odot$ is the Hadamard product. At each component (c, p, q) this is a complex scalar linear recurrence $h_t = a\, h_{t-1} + u_t$ with $a = \widehat{K}^{(c,p,q)}$ and $u_t = \widehat{U}_t^{(c,p,q)}$.

Associativity. Treating each step as the affine map $h \mapsto a_t h + u_t$, the composition of two such maps is itself affine, $(a_i, u_i) \circ (a_j, u_j) = (a_j a_i,\ a_j u_i + u_j)$, and associativity follows from associativity of complex multiplication and addition: both bracketings of three steps yield $(a_k a_j a_i,\ a_k a_j u_i + a_k u_j + u_k)$. We evaluate all T prefix states using `jax.lax.associative_scan` with this combiner in $\mathcal{O}(\log T)$ depth and $\mathcal{O}(T)$ work; the per-frequency carry is a single complex scalar at every level of the scan. The same combiner applied to spatial-domain kernels would require $\boldsymbol{K}_j * \boldsymbol{K}_i$, whose support grows with each composition: composing T kernels of side k yields a kernel of side $T(k-1)+1$ and area $\mathcal{O}(T^2k^2)$. This is the parallel-scan blow-up that confines ConvS5 to 1×1 kernels, where $k - 1 = 0$ and composition is degenerate. For the basic sCSSM, $a_t = \widehat{K}^{(c,p,q)}$ is constant in t; the same combiner applies unchanged when transitions are input-dependent.

Stability squash. A scalar linear recurrence $h_t = a h_{t-1} + u_t$ is geometrically stable iff $|a| < 1$. We enforce this on $\widehat{\boldsymbol{K}}$ before the recurrence with a smooth squash that preserves phase:

$$\widehat{\boldsymbol{K}} \leftarrow \widehat{\boldsymbol{K}} \cdot \frac{\rho}{1 + |\widehat{\boldsymbol{K}}|}, \qquad \rho = 0.999, \tag{20}$$

where $|\cdot|$ is taken elementwise. The map $z \mapsto z \cdot \rho/(1+|z|)$ contracts $\mathbb{C}$ into the open disk of radius ρ: for any $z \in \mathbb{C}$, $|z \cdot \rho/(1+|z|)| = \rho \cdot |z|/(1+|z|) < \rho$, with equality only in the limit $|z| \to \infty$. The transformation rescales magnitude only, leaving $\arg(\widehat{\boldsymbol{K}})$ untouched, so the spatiotemporal phase structure that the kernel learns is preserved. Combined with the per-frequency-scalar recurrence (Eq. (19)), this guarantees that every spatial frequency mode decays geometrically with rate at most ρ, ruling out the divergence modes that destabilise unconstrained convolutional recurrences [21]. We use $\rho = 0.999$ in all experiments.

Forward pass. The full per-block computation is: (i) compute $\widehat{\boldsymbol{U}}_t = \mathrm{rfft2}(\boldsymbol{x}_t)$ for each frame; (ii) compute $\widehat{\boldsymbol{K}}$ from the learned kernel $\boldsymbol{K}$ via zero-padding to (H, W) and rfft2, and apply Eq. (20); (iii) execute the associative scan of Eq. (19) along the temporal axis using the combiner above; (iv) recover $\boldsymbol{h}_t = \mathrm{irfft2}(\widehat{\boldsymbol{S}}_t)$; (v) apply a single $\mathrm{Dense}(C)$ out-projection added residually to the block input. Total work per block is $\mathcal{O}(T \cdot C \cdot H \cdot (W/2+1))$ in `complex64`, dominated by the FFTs and the associative scan.

E.3 ImageNet-1K

We start from the edge-class Single-Head Vision Transformer (SHViT) [40] at the 6M and 11M parameter scales. The GDN-sCSSM variant replaces every self-attention layer with a GDN-sCSSM block of matched input/output dimension; surrounding architecture, augmentation, and optimizer are unchanged. The *Hybrid SHViT-S1* control replaces self-attention with ConvNeXt-style depthwise convolutions of matched receptive field, isolating the contribution of recurrence from the convolutional inductive bias alone. All variants are trained for 300 epochs at batch size 512 (auto-halved on OOM) using the TIMM [37] recipe: AdamW with peak LR 10^{-3} and weight decay 5×10^{-2}, RandAugment-3, Mixup ($\alpha = 0.8$), CutMix ($\alpha = 1.0$), RandomErasing ($p = 0.25$), label smoothing 0.1, drop-path 0.1, EMA 0.9998, and global gradient norm clipping at 1.0. We sweep $T \in \{1, 4, 8\}$ for the sCSSM variants. ImageNet runs use `pmap` across two B200 GPUs; all other experiments are single-GPU.

E.4 LoRA fine-tuning and VideoPhy-2 evaluation

Backbone. We fine-tune CogVideoX-5B T2V (`THUDM/CogVideoX-5b`) [39]; the diffusion transformer (5.57B parameters), T5-XXL text encoder, and 3D VAE remain frozen. Mixed-precision `bf16` throughout. CogVideoX uses the `CogVideoXDPMScheduler` with `timestep_spacing='trailing'`. Output latents at training resolution are $T = 13, H = 30, W = 45$, produced by the VAE at $4\times$ temporal and $8\times$ spatial stride with the transformer's spatial patch size 2.

LoRA variants. All variants use rank $r = 16$, $\alpha = 16$, injected only into the self-attention $Q/K/V/O$ projections (`attn1.to_q/k/v/out.0`); cross-attention is left untouched, since rank-16 LoRAs on cross-attention reproducibly degrade prompt following. We compare four variants: (i) *Standard* LoRA (vanilla MLP, 16,515,072 trainable parameters); (ii) *Transformer* LoRA (multi-head temporal self-attention over T pooled tokens, 16,644,096 parameters); (iii) *GDN-sCSSM* LoRA with 1×1 spectral kernel (14,508,816 parameters); (iv) *GDN-sCSSM* LoRA with 5×5 spectral kernel (14,508,816 parameters). The GDN-sCSSM LoRA pipeline is: down-project to rank 16, per-head $Q/K/V$ at $d_k = 4$, FFT2 over (H, W), L2-normalised keys, factored α/β decay gates, learned spectral kernel, delta-rule scan over T, iFFT2, RMSNorm, SiLU output gate, up-project.

Training data. A curated subset of OpenVid-1M [24] following [41]: filter `motion_score` ≥ 1, `aesthetic_score` ≥ 5, $6.5 \leq$ `seconds` ≤ 30 (the lower bound matters because CogVideoX trains at 49 frames / 8 fps = 6.125 s, so shorter clips would be temporally oversampled). Retaining only successfully downloaded clips yields 14,857 train and 742 val (5% random val split, seed 0). Each clip is pre-encoded once: 49 uniformly-sampled frames at 720×480, VAE-encoded with tiling and slicing to a $[13, 16, 60, 90]$ `bf16` latent, plus a $[226, 4096]$ `bf16` T5 caption embedding. The training loop never instantiates T5 or the VAE. **Training loop.** AdamW ($\beta = (0.9, 0.999)$, weight decay 0.01), peak LR 10^{-4}, linear warmup of 200 steps to cosine decay to 0, batch size 1, max 5000 steps, flow-matching loss with target = noise − latents, evaluation every 250 steps (averaged over 10 batches), early stopping with patience 5 on validation loss, gradient checkpointing enabled, gradient clipping at max-norm 1.0, seed 42. GDN-sCSSM is $\sim 2.4\times$ slower per step than Standard or Transformer due to two FFT2/iFFT2 calls and the sequential delta-rule scan over $T = 13$.

Inference. For each variant we generate 591 videos, one per unique caption from the official VideoPhy-2 [2] test set, joined to the published Mistral-NeMo-12B upsampled prompts. Following the VideoPhy-2 protocol, we generate with the upsampled long caption and score with the raw short caption. Generation settings are CogVideoX-5B's native release values: 49 frames, 720×480, 8 fps, 50 DPM steps, CFG 6.0, seed 42. Our base CogVideoX-5B Joint of 24.0% matches the published human-rated CogVideoX-5B Joint of 25.0% within 1pp, validating the generation pipeline.

Autograder. We use the official VideoPhy-2 mPLUG-Owl autograder. Scoring uses `--task sa` and `--task pc` with deterministic decoding (`do_sample=False, top_k=1, temperature=0.001`), 32 frames per clip. Joint = (SA ≥ 4) AND (PC ≥ 4) per clip.

Significance. Joint differences between variants are tested with paired McNemar over the 591 matched prompts; SA and PC means are tested with paired bootstrap CIs (5000 resamples).

F Limitations

While our formulation enables scalable spatiotemporal coupling, it remains grounded in linear state space dynamics, which may limit its ability to capture strongly nonlinear or multistable behaviors characteristic of biological circuits. The spectral representation also relies on structured assumptions such as translation invariance and circulant boundary conditions. The implementation we evaluate uses depthwise (per-channel) spatial kernels, which collapse $\mathbf{\Lambda}_c$'s $c \times c$ blocks to 1×1 and bypass the $\boldsymbol{Q}_c$ change-of-basis derived in Section 4; channel mixing within the spectral recurrence remains untested. The improvement on VideoPhy-2 over the unadapted base model is modest (1–2 percentage points on Joint), and our GDN-sCSSM LoRA is $\sim$2.4$\times$ slower per training step than Standard or Transformer LoRAs due to the FFT2/iFFT2 calls and sequential delta-rule scan over $T = 13$. Finally, all reported numbers come from single-seed runs, with variability captured only across the embed-/state-dim sweep rather than seed replicates.

A key limitation of moving from the real to spectral domain is that the latter operates on complex tensors, doubling per-block memory. On B200-class hardware this overhead is modest in practice—the FFT/iFFT and elementwise spectral steps are bandwidth-limited and pay a roughly proportional cost, while the surrounding tensor-core matmuls remain compute-bound and largely unaffected. Lower-precision split-real/imag variants (complex32, complex16) are possible and work well in pilot experiments. We leave data type and kernel optimizations for future work.

G Complexity Proofs

We analyse the complexity of the parallel scan algorithm to show why convolutions in spatial domain make the complexity explode, and how doing convolutions in spectral domain preserves $O(\log T)$ complexity. We consider both the work complexity W, measuring the total amount of atomic operations to perform, and the span, or time complexity under unlimited parallelization T_∞, measuring the longest sequence of atomic operations remaining after optimal parallelization.

We begin by addressing the main issue of the complexity explosion of convolutional SSMs. Our main result is Proposition 3. We show in Appendix G.5 that we can achieve the same work and span as a regular linear SSM, with no hidden costs, and by preserving the connectivity we cared about.

G.1 The Parallel Scan

We first describe the parallel scan operation in the most general way. This operation is the core of the SSM parallelization. Let $(a_i)_{i\in I} \in \mathcal{M}^I$ be elements of a monoid $(\mathcal{M}, \oplus, e)$, with $I = [\![0, T-1]\!]$. The parallel scan is an algorithm that computes $(s_i = \sum_{j=0}^{i} a_j)_{i\in I}$. This algorithm is parallelisable and requires associativity of the $\oplus$ operator and a neutral element e, hence the minimal requirement of a monoid structure. It proceeds in two sweeps mirroring each other. The first sweep proceeds in a tree like structure, with level l of the tree computing $a_k^{l+1} := a_{2k}^l \oplus a_{2k+1}^l$, and initialised with $a_k^0 := a_k$. Once this upward sweep is over, the downward sweep computes mirror operations, with $b_1^0 := e$ and $(b_{2k}^{l+1}, b_{2k+1}^{l+1}) = (b_k^l, b_k^l \oplus a_{2k}^{\log_2(T)-(l+1)})$. The order is crucial here as $\oplus$ is not required to be commutative. It is only required to be associative. In our application for SSM, it is not commutative. We denote the parallel scan procedure as $\mathrm{Scan}((a_i)_{i\in I})$ or Scan for short. Figure A2 provides a visualisation of the entire procedure.

The procedure described above yields $(\sum_{j=0}^{i-1} a_j)_{i\in I}$, with the cumulative sum being $a_0^{\log_2(T)}$ the intermediate result of the upward sweep. We analyse the work W and span T_∞ of Scan in terms of

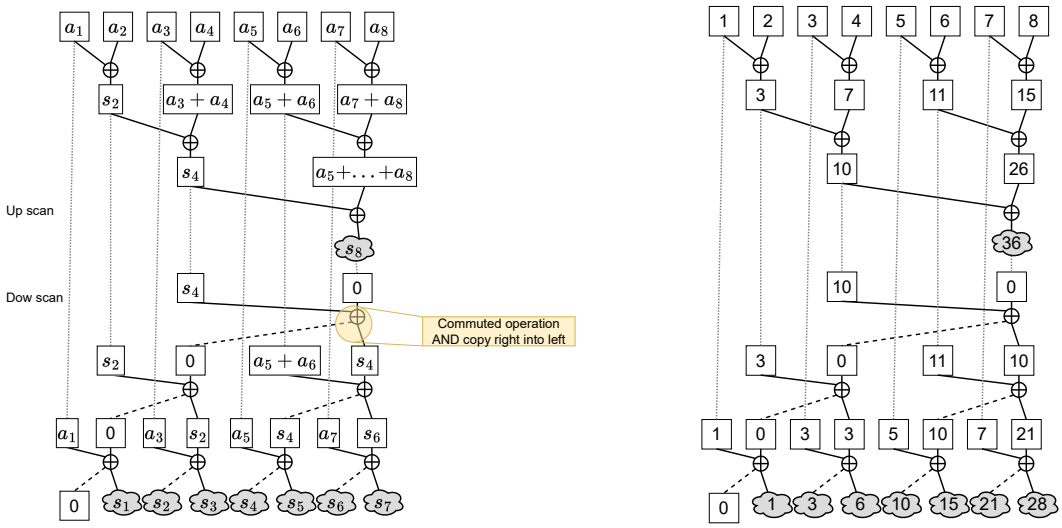


Figure A2: **Left:** parallel scan with a generic monoid. **Right:** illustration on the special case of natural integers.

floating point operations, where W counts the total number of operations needed and T_∞ counts the time complexity under unlimited parallelism.

Proposition 5 (Parallel Scan Complexity for Integers)**.** *Let $(\mathcal{M}, \oplus, e) \coloneqq (\mathbb{N}, +, 0)$ be the monoid of natural integers equipped with addition. Let $(a_i)_{i \in I} \in \mathbb{N}^I$ be a sequence of integers.*

$$W(\text{Scan}) = 2(T-1) \tag{21}$$
$$T_\infty(\text{Scan}) = 2\log_2 T \tag{22}$$

Proof. At level l of the upward scan, $U_l = T/(2^{l+1})$ $\oplus$ operations are performed, each of them having a complexity of 1. There are $\log_2 T$ levels in total. The same goes on for the downward scan, with $D_l = T/(2^{\log_2(T)-l}) = 2^l$ operations at level l. Therefore, the work complexity of level l is U_l (resp. D_l) while its span is 1 with unlimited parallelism. It follows that

$$W = 2 \sum_{l=0}^{\log_2 T - 1} 2^l = 2(T-1) \tag{23}$$
$$T_\infty = 2 \sum_{l=0}^{\log_2 T - 1} 1 = 2\log_2 T \tag{24}$$

□

The key point of this analysis is that the $\oplus$ operation has constant time complexity.

G.2 SSM Complexity

The case of SSMs is a little more complicated than that of real numbers. In SSMs, the recurrence is defined by Equation (2). We take $B = I$ and $d = N$ for simplicity. Thus, the goal is to compute in parallel the hidden states $(h_i = \sum_{j=0}^{i} \overline{A}^{i-j} x_j)_{i \in I}$, where x_i are vectors of dimension d and $\overline{A}$ is a square matrix. This is almost but not quite what we need for the parallel scan. We define the

following monoid:

$$\mathcal{M}^{\mathrm{SSM}} \coloneqq \left\{a = (A, x) \mid A \in \mathcal{M}_d(\mathbb{R}),\ x \in \mathbb{R}^d\right\} \tag{25}$$

$$(A_a, x_a) \oplus^{\mathrm{SSM}} (A_b, x_b) \coloneqq (A_b \odot A_a, A_b \cdot x_a + x_b) \tag{26}$$

$$e^{\mathrm{SSM}} \coloneqq (I_d, 0) \tag{27}$$

With this monoid structure, and the sequence $(a_t)_{t\in I} \coloneqq ((\overline{A}_t, x_t))_{t\in I} \in \mathcal{M}^I$, the result of the parallel scan is $((A^t, \sum_{j=0}^t A^{t-j} x_j))_{t\in I}$. The complexity of the scan is less straightforward, but the complexity of each $\oplus^{\mathrm{SSM}}$ operation depends only on d. It is a constant of T. Therefore, the overall span of the scan is $T_\infty = O(\log T)$ and $W = O(T)$. Recall Proposition 4:

Proposition (SSM Complexity)**.** *Let $(\mathcal{M}, \oplus) \coloneqq (\mathcal{M}^{\mathrm{SSM}}, \oplus^{\mathrm{SSM}})$ be the monoid defined by Equations (25) to (27). Let $(a_i)_{i\in I} \in (\mathcal{M}^{\mathrm{SSM}})^I$.*

$$W(\mathrm{Scan}) = 2(T-1)(d^3 + d^2 + d) \tag{28}$$

$$T_\infty(\mathrm{Scan}) = 2\log_2(T)(d^3 + d^2 + d) \tag{29}$$

Proof. The proof of Proposition 5 applies with the only difference that now, each $\oplus^{\mathrm{SSM}}$ operation has complexity d^3 for the matrix multiplication on the left, d^2 for the matrix-vector multiplication and d for the vector addition on the right. □

What about the time constants? As mentioned in Proposition 4, as is, the work complexity is $O(T)$ with a constant that is $O(d^3)$ due to matrix multiplications. Note that it retains the constant term in $O(d^2)$ of applying the input and output transformations $\boldsymbol{B}$ and $\boldsymbol{C}$. This is largely amortized in standard SSM, but in the case of our sCSSM, d will be so large that even this becomes intractable (see Appendix G.5).

The diagonal SSM (Equation (4)) reduces this constant from $O(d^3)$ to $O(d)$. The complexity gain comes from the change of basis being absorbed outside of the recurrence such that the recurrence can operate on diagonalized matrices.

G.3 Spatial ConvSSM Complexity

Rewriting the SSM (Equation (2)) to the ConvSSM formulation of Equation (5), we can similarly adapt the monoid structure as:

$$\mathcal{M}^{\mathrm{Conv}_1} \coloneqq \left\{a = (\boldsymbol{K}, x) \mid \exists k_1, k_2,\quad \boldsymbol{K} \in \mathbb{R}^{k_1\times k_2\times c\times c},\ x \in \mathbb{R}^{h\times w\times c}\right\} \tag{30}$$

$$(\boldsymbol{K}_a, x_a) \oplus^{\mathrm{Conv}_1} (\boldsymbol{K}_b, x_b) \coloneqq (\boldsymbol{K}_b \circledast \boldsymbol{K}_a, \boldsymbol{K}_b * x_a + x_b) \tag{31}$$

$$e^{\mathrm{Conv}_1} \coloneqq (I, 0) \tag{32}$$

where $\circledast$ is the block convolution between two kernels and $*$ is the convolution between a kernel and an image:

$$\forall \begin{cases} i \in [\![0, h-1]\!] \\ j \in [\![0, w-1]\!] \\ c_{\mathrm{out}} \in [\![1, c]\!] \end{cases}, (\boldsymbol{K} * x)_{i,j,c_{\mathrm{out}}} = \sum_{u=0}^{k_1-1} \sum_{v=0}^{k_2-1} \sum_{c_{\mathrm{in}}=1}^{c} \boldsymbol{K}_{u,v,c_{\mathrm{in}},c_{\mathrm{out}}} x_{(i-u,j-v)\%(h,w),c_{\mathrm{in}}}$$

$$\forall \begin{cases} u \in [\![0, k_1+l_1-2]\!] \\ v \in [\![0, k_2+l_2-2]\!] \\ c_{\mathrm{in}}, c_{\mathrm{out}} \in [\![1, c]\!] \end{cases}, (L \circledast \boldsymbol{K})_{u,v,c_{\mathrm{in}},c_{\mathrm{out}}} = \sum_{p=0}^{l_1-1} \sum_{q=0}^{l_2-1} \sum_{c_k=1}^{c} L_{p,q,c_k,c_{\mathrm{out}}} \boldsymbol{K}_{u-p,v-q,c_{\mathrm{in}},c_k}$$

$\boldsymbol{K}$ being zero-padded, and with $L \in \mathbb{R}^{l_1\times l_2\times c\times c}$, $\boldsymbol{K} \in \mathbb{R}^{k_1\times k_2\times c\times c}$ and $x \in \mathbb{R}^{h\times w\times c}$.

With this structure, the key aspect is that the spatial size of the kernel resulting from $\circledast$ is the sum of the sizes of the input kernels. Moreover, the support of the image grows in the same way. Therefore,

the parallel scan algorithm operates on larger and larger kernel sizes, with sizes growing exponentially with the depth in the tree structure. The complexity of the $\oplus^{\text{Conv}_1}$ operator is thus no more constant as it depends on the depth. The bottleneck is at the very last layer of the upward scan, where kernel size is $O(Tk)$, with k the size of the original kernels $\overline{\boldsymbol{K}}_t$, resulting in a single $*$ operation whose complexity is in $O(T^4)$. This bottleneck is illustrated in Figure A3. Recall Proposition 1:

Proposition (Spatial ConvSSM Complexity)**.** *Let* $(\mathcal{M}, \oplus) := (\mathcal{M}^{\text{Conv}_1}, \oplus^{\text{Conv}_1})$ *be the monoid defined by Equations (30) to (32). Let* $(a_i)_{i\in I} \in (\mathcal{M}^{\text{Conv}_1})^I$.

$$W = O(T^4) \tag{33}$$
$$T_\infty = O(W) \tag{34}$$

Proof. The proof of Proposition 5 applies with the major difference that in the upward scan, at level l, each $\oplus^{\text{Conv}_1}$ operation has complexity $C_1 = (k + (2^l - 1)(k-1))^4 \times c^3 = O((2^l k)^4 \times c^3)$ for the kernel-kernel convolution on the left, $C_2 = O((h + 2^l k)(w + 2^l k) \times (2^l k)^2 \times c^2) = O((2^l k)^4 c^2)$ for the kernel-image convolution and $C_3 = O((2^l k)^2 c)$ for the image addition on the right. The same goes on for the downward scan.

Thus, Equations (23) and (24) become:

$$W = O(2 \sum_{l=0}^{\log_2(T)-1} 2^{\log_2(T)-(l+1)} \times \left[(2^l k)^4 \times c^3 + (2^l k)^4 \times c^2\right]) \tag{35}$$
$$= O(T^4 \times k^4 \times c^3) \tag{36}$$
$$T_\infty = O(2 \sum_{l=0}^{\log_2(T)-1} \left[(2^l k)^4 \times c^3 + (2^l k)^4 \times c^2\right]) \tag{37}$$
$$= O(W) \tag{38}$$

□

What about the time constants? The standard equivalent of diagonalization in convolutional neural network is depthwise convolution, or the absence of channel mixing. This reduces the time constant from $O(c^3)$ to $O(c)$.

Another standard restriction of convolution is to have kernels separable in space, i.e., the spatial rank of $\boldsymbol{K}$ is 1, which means it can be written as $\boldsymbol{K} \in \mathbb{R}^{k\times k} = \boldsymbol{K}_1 \boldsymbol{K}_2^\top$, where both $\boldsymbol{K}_1 \in \mathbb{R}^k$ and $\boldsymbol{K}_2 \in \mathbb{R}^k$. By treating the horizontal and vertical kernels separately, we get $C_1 = O((2^l k)^2 c^3), C_2 = O((2^l k)^3 c^2), C_3 = O((2^l k)^2 c)$. This gives:

$$W = T_\infty = O(T^3 k^3 c^2) \tag{39}$$

with the c^2 again reducible to c using depthwise convolution.

Note that with circular padding constraint, the kernel size can be bounded to $h \times w$. With this trick, the complexity is polynomial in time until $T = hw/k^2$ where it becomes linear with an absurd constant factor. Moreover, as mentioned in the main text, the discretization step of the SSM leads to kernels with full spatial support, meaning that the initialization of the scan is $(a_t)_{t\in I} = ((\overline{\boldsymbol{K}}_t, x_t))_{t\in I}$ with $\overline{\boldsymbol{K}}_t$ of size $h \times w$. Thus, in practice, with circular padding, the scan is linear in time and even the first step is intractable.

G.4 "SAME" SPATIAL CONVOLUTION

In practice, convolutional recurrent models always interleave convolutions with projections to maintain the same state size across time. This means that, as mentioned in Section 3, the resulting linear operators have column support restricted to the maximum image size given by the projection (call it H and W), and row support restricted to the max between the input image size (h, w) and the projection size (H, W). Assuming $h = H, w = W$, this means the recurrence can be expressed as a linear recurrence with dimension hwc, and Proposition 4 applies, which proves Proposition 2.

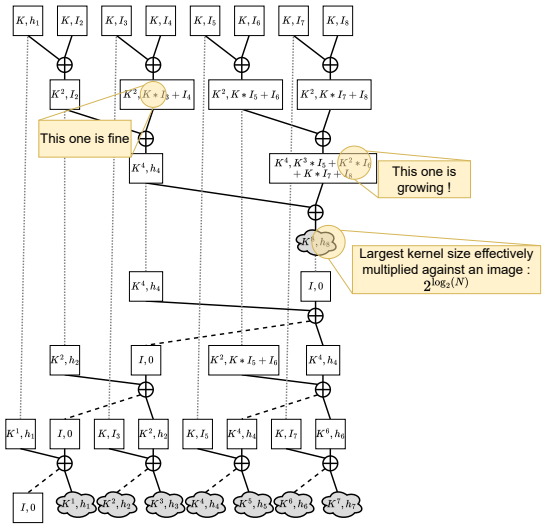


Figure A3: Parallel scan applied to the monoid structure defined for spatial convolution SSM. Highlighted is the core of the issue: growing kernels lead to a bottleneck where a single operation has $O(T^4)$ complexity.

G.5 Spectral ConvSSM Complexity

As mentioned in Section 4, to dispel the curse of growing kernels we move to the spectral domain and the ConvSSM formulation of Equation (7). In this domain, our monoid becomes:

$$\mathcal{M}^{\text{Conv}_2} \coloneqq \left\{ a = (\mathbf{\Lambda}_c, x) \;\middle|\; \exists (\boldsymbol{K}, x') \in \mathcal{M}^{\text{Conv}_1}, \begin{cases} \mathbf{\Lambda}_c = P^{-1}\text{Circ}(\boldsymbol{K}_{\text{padded}})P \in \mathbb{C}^{hwc \times hwc} \\ x = P^{-1}x' \in \mathbb{C}^{hwc} \end{cases} \right\} \tag{40}$$

$$(\mathbf{\Lambda}_c^a, x_a) \oplus^{\text{Conv}_2} (\mathbf{\Lambda}_c^b, x_b) \coloneqq (\mathbf{\Lambda}_c^b \odot \mathbf{\Lambda}_c^a, \mathbf{\Lambda}_c^b \cdot x_a + x_b) \tag{41}$$

$$e \coloneqq (I, 0) \tag{42}$$

with $\odot$ the element-wise matrix multiplication and $\cdot$ the element-wise matrix-vector multiplication. With this operator, the kernel grows no more, the cost is again a constant of time and depth, depending only on h, w and c. Again, we can directly apply Proposition 4 to get the asymptotic complexity. To get the constant factors, note that $\mathbf{\Lambda}_c$ is not exactly diagonal but block diagonal, with blocks of size c by c. This gives the constant factor of $O(hwc^3)$.

We can reduce the constant to $O(hwc)$ through the equivalent SSM of Eq. (8). However, this still leaves the hidden constant term in $O((hwc)^2)$ due to the now dense and arbitrary B. This term is prohibitive. This is where the spectral domain and the algorithmic efficiency of the DFT comes in. By once again reformulating to an equivalent SSM, we arrive at our final formulation of the sCSSM in Eq. (12). This formulation reduces the constant term from $O((hwc)^2)$ to a more tractable $O(hw\log(hw)c)$.

G.6 PathFinder Hyperparameter Sweeps

Hyperparameter search and Pareto reporting. Our objective is to characterize the best-observed accuracy–efficiency tradeoffs across the evaluated hyperparameter configurations and training runs, including retries, rather than average performance across runs. For each efficiency metric, we constructed an empirical Pareto front as the upper envelope of the observed accuracy–cost pairs. These fronts therefore describe the best tradeoffs found within our search, not the expected performance of an individual training run. Hyperparameter axes and configuration counts are reported in Table A1.

Table A1: **Hyperparameter sweeps for visual reasoning.** Numbers in parentheses indicate the number of settings tested along each axis; *Configs.* gives the number of configurations in the resulting grid. All circuits use a shared stem and readout.

Circuit	T	Depth	Configs.	Hyperparameter axes
Transformer	8	$\{1, 3\}$	32	depth (2) $\times$ embed (2) $\times$ lr (2) $\times$ heads (4)
GDN-sCSSM	8	$\{1, 3\}$	24	depth (2) $\times$ embed (2) $\times$ QKV conv (2) $\times$ delta-key (3)
Transformer-ST	8	$\{1, 3, 6\}$	24	depth (3) $\times$ embed (2) $\times$ heads (2) $\times$ drop path (2)
Mamba-sCSSM	8	$\{1, 3\}$	12	depth (2) $\times$ embed (2) $\times$ kernel (3)
ConvS5	8	$\{1, 3\}$	8	depth (2) $\times$ embed (2) $\times$ kernel (2)
ConvSSM	8	$\{1, 3\}$	8	depth (2) $\times$ embed (2) $\times$ kernel (2)
GDN	1	$\{1, 3\}$	8	depth (2) $\times$ embed (2) $\times$ delta-key (2)
Mamba-2	1	$\{1, 3\}$	8	depth (2) $\times$ embed (2) $\times$ state dim (2)
sCSSM	8	$\{1, 3\}$	8	depth (2) $\times$ embed (2) $\times$ kernel (2)
S4ND	1	$\{1, 3\}$	4	depth (2) $\times$ embed (2)
S4ND-S4D (abl.)	1	$\{1, 3\}$	4	depth (2) $\times$ embed (2)
no-FFT (abl.)	8	$\{1, 3\}$	4	depth (2) $\times$ embed (2)
sCSSM-NoSpace (abl.)	8	$\{1, 3\}$	4	depth (2) $\times$ embed (2)
sCSSM-NoTime (abl.)	1	$\{1, 3\}$	4	depth (2) $\times$ kernel (2)