Native Multi-Dimensional Subquadratic Operators via Input Dependent Long Convolutions

David R. Wessels[*,‡,1] **Farhad Ramezanghorbani**[*,‡,2] **Alireza Moradzadeh**[‡,2] **David W. Romero**[†,‡,3]
Olivia Viessmann[2] **Maksim Zhdanov**[1] **John St. John**[2] **Ken Janik**[2]
David M. Knigge[4] **Yucheng Tang**[2] **Erik J. Bekkers**[1] **Saee Gopal Paliwal**[2]
[1]AMLab, University of Amsterdam [2]NVIDIA [3]Cartesia AI [4]New Theory AI

Abstract

Subquadratic alternatives to attention require compromises when applied to multi-dimensional data: standard convolutions lack global receptive fields and input dependency, while recurrent models require rasterizing data such as images, volumes, and partial differential equation (PDE) into an ad-hoc 1D scan order that violates their spatial structure. We introduce *HyenaND*, a subquadratic, global, input-dependent operator that acts directly on the native geometry of multidimensional data through convolutions with implicitly parametrized global, input-dependent multi-dimensional convolutional kernels. Our CUDA implementation, `nvSubquadratic-ops`, fuses the FFT-convolution path to turn HyenaND's $\mathcal{O}(L \log L)$ scaling into wall-clock speedups. Across long-context genomics, computer vision, medical imaging, and PDE modeling, pure HyenaND stacks match the accuracy of strong attention baselines, while hybrid configurations that interleave HyenaND and attention layers outperform both pure attention and strong recurrence-based hybrids.

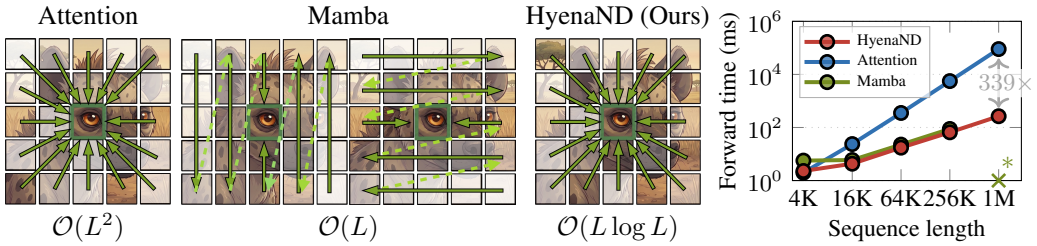


Figure 1: *(Left):* Complexity of global multi-dimensional operators by token count L. Attention is natively multi-dimensional but scales quadratically. Mamba is subquadratic but inherently 1D, requiring ad-hoc 1D scan orders to process multi-dimensional data. HyenaND (ours) is global, natively multi-dimensional and subquadratic. *(Right):* Forward-pass time vs. sequence length using `flash-attention`, the official `mamba_chunk_scan_combined` Mamba2 kernel and `nvSubquadratic-ops`; HyenaND scales gracefully to million-token sequences while attention collapses at long contexts.
*The Mamba2 kernels cannot reach 1M tokens: they fail with a CUDA illegal memory access, a known open issue with the kernels' 32-bit indexing at long sequence lengths (`state-spaces/mamba` #686,#1015).

*Equal contribution. †Technical lead. ‡Core contributor.
Correspondence: `davidwessels15@gmail.com`, `farhadr@nvidia.com`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Deep learning models increasingly operate on large, multi-dimensional data: high-resolution images, 3D medical scans, high-resolution spatio-temporal fields, long genomic sequences. On such inputs, the quadratic cost of Transformers [44] hinders its applicability to even moderate sizes. The standard workaround is patchification [7], but it trades performance for tractability: patchification scaling laws [45] show that accuracy improves monotonically as patch size decreases across architectures, tasks and input scales. However, shrinking the patch size to harvest these gains drives the token count back up, and attention's quadratic cost again becomes binding. We need a subquadratic operator that can follow the patchification scaling curve past attention's quadratic wall.

State-space models (SSMs) such as S4 [11] and their input-dependent descendants Mamba [10, 3] have become the prevailing subquadratic approach, with $\mathcal{O}(L)$ training and $\mathcal{O}(1)$ per-step autoregressive inference. As recurrences, they are fundamentally causal and 1D: bidirectional contexts must run the recurrence twice –on the input and its reversal– doubling compute and memory. Worse, recurrences admit no natural extension to multi-dimensional data. As such, existing adaptations must rasterize ND inputs into ad-hoc 1D scan orders, destroying their geometric structure (§2.2; Fig. 1).

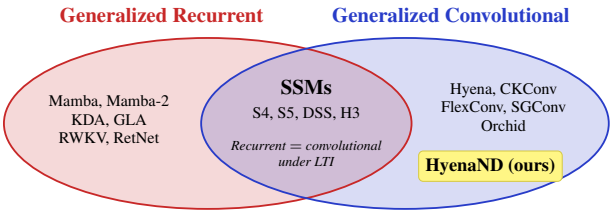
Mamba's selectivity is one specific way to induce input dependency: per-token dependency at the cost of linear time invariance (LTI). This is a well-motivated trade for 1D autoregressive language modeling, where the data is non-LTI and causality precludes conditioning on the whole input. But for natural and multi-dimensional signals –audio, images, volumes, physical fields– that's the wrong trade, as these signals are themselves approximately LTI and often available at once. In fact, the original Mamba paper reports that Mamba underperforms the LTI S4 on audio generation [10, Appx. E.4].

A separate, non-recurrent family of subquadratic operators acts on the input in its native geometry: *long convolutions* [36, 35, 34]. This family has been overlooked for two reasons: First, the dominant SSM formulation views the convolutional form as a trick to speed up training of recurrent operators rather than a distinct architectural family, a conflation we untangle in §2. Second, long convolutions lack the input dependency that underlies Mamba's selectivity, and this absence is widely regarded as a fundamental expressiveness limitation. This second objection, however, is specifically tied to autoregressive 1D processing, where causality forces input-dependence to be expressed per token. In non-autoregressive settings (diffusion, multi-dimensional data) the input is available at once and the operator can be conditioned on the input as a whole: input dependency at the *sample* level rather than at the *token* level (§2.3). Combined with the LTI inductive bias argued for above, this positions long convolutions as the matched subquadratic primitive for natural and multi-dimensional data.

Generalized convolutional models also carry no recurring hidden state: each output is a direct function of the input weighted by the kernel, so propagating information across long distances is as simple as placing kernel mass at the corresponding offset. Recurrent operators, by contrast, must compress relevant history into a fixed-size state, a bottleneck that becomes especially severe in multi-dimensional settings. In these settings the spatial extent of dependencies grows multiplicatively.

Together, subquadratic scaling, native multi-dimensional structure, sample-level input-dependence, and preserved LTI define a target operator that no existing primitive simultaneously satisfies. We realize it with *HyenaND* (§3): an ND convolution acts directly on the input geometry (no rasterization), with a global, freely learned kernel conditioned on the full input through FiLM [33] with globally available register tokens [5] (sample-level input-dependence, LTI) evaluated via ND FFTs in $\mathcal{O}(L \log L)$ (subquadratic). We pair it with `nvSubquadratic-ops` (§4), an IO-aware CUDA library that turns this asymptotic edge into wall-clock speedups via fused ND FFTs, spectral modulations, and mixed precision. Across long-context genomics, computer vision, medical imaging, and PDE modeling, pure HyenaND stacks match or outperform strong attention and modality-specialized baselines, while hybrid HyenaND-attention stacks outperform both pure attention and recurrence-based hybrids. Our primary contributions are:

- A reframing of the subquadratic landscape (§2) that deconflates the prevalent SSM-centric framing into *SSMs*: S4 [11], H3 [8]; *generalized recurrent*: Mamba [10], Linear Attention [15], GDA [48]; and *generalized convolutional models*: CKConv [36], Hyena [34], HyenaND (ours).
- HyenaND (§3): a strong native ND, LTI, sample-level input-dependent subquadratic operator.
- `nvSubquadratic-ops` (§4): An IO-aware CUDA implementation that translates the subquadratic theoretical gains of ND convolutional operators into wall-clock speedups.



	Gen. Recurrent	Gen. Conv.	SSMs	Transformer
Input-dependent	✓†	✓†	×	✓
AR inference cost (1D)	$\mathcal{O}(1)$	$\mathcal{O}(L)$	$\mathcal{O}(1)$	$\mathcal{O}(L)$
Fixed-size state	✓	×	✓	×
LTI	×	✓	✓	×
Natively ND	×	✓	×	×‡
Bidirectional	w/ caveats	✓	w/ caveats	✓

† Kernel-level input-dependence via registers/FiLM (HyenaND). Input-dependence only through gating in vanilla Hyena.
‡ Possible via flattened tokenization + ND positional encodings. Risks lacking a proper multi-dimensional inductive bias [46].

Figure 2: The subquadratic landscape. SSMs are simultaneously recurrent and convolutional. Relaxing the equivalence in either direction yields two distinct families, *generalized recurrent* and *generalized convolutional (left)*. Each family has different computational forms and trade-offs *(right)*.

- A thorough empirical validation of HyenaND across modalities in 1D, 2D and 3D, showing the strength of both pure and hybrid HyenaND backbones (§5, Appendix D).

2 From SSMs to Generalized Recurrent and Convolutional Models

The success of State-Space Models (SSMs) has reshaped how the broader landscape of subquadratic architectures is read to the point where ***subquadratic operators are routinely labeled as SSMs, regardless of whether they admit a state-space formulation***. This conflation has two consequences. First, purely recurrent models like Mamba are incorrectly labeled as SSMs despite admitting no convolutional form. Second, purely convolutional models are wrongly assumed to share the inherent recurrence and reduced expressivity of pure SSMs [11]. We defy both readings and trace them to a single source.

2.1 State-Space Models: The Source of the Confusion

An SSM maps a sequence $\mathbf{x}{=}\{x_t\}_{t=1}^{L}$ to an output $\mathbf{y}{=}\{y_t\}_{t=1}^{L}$ via a hidden state $\mathbf{h}_t \in \mathbb{R}^d$ as:

$$y_t = \mathbf{c}\,\mathbf{h}_t; \quad \mathbf{h}_t = \mathbf{A}\,\mathbf{h}_{t-1} + \mathbf{b}\,x_t \quad \text{with parameters } \mathbf{A} \in \mathbb{R}^{d\times d}, \mathbf{b} \in \mathbb{R}^d, \mathbf{c} \in \mathbb{R}^{1\times d}. \tag{1}$$

For multi-channel inputs $x_t \in \mathbb{R}^H$, an SSM treats each channel independently, corresponding to a depthwise operation. Unrolling the recurrence in (1) yields an *equivalent* convolution $\mathbf{y}{=}K * \mathbf{x}$, with a long kernel $K{=}(\mathbf{cb}, \mathbf{cAb}, ..., \mathbf{cA}^{L-1}\mathbf{b})$, evaluable in $\mathcal{O}(L \log L)$ time via FFT. *This equivalence is specific to SSMs, and relaxing it in either direction –recurrent or convolutional– yields two distinct architectural families that share neither computational form nor the same trade-offs.*

2.2 Generalized Recurrent Models

Generalized recurrent models –with Mamba [10] as its archetype– sacrifice LTI and the convolutional form of SSMs in exchange for input dependency. A linear projection f_θ makes its parameters depend on the input: $(\mathbf{A}_t, \mathbf{b}_t, \mathbf{c}_t){=}f_\theta(x_t)$. The upside is content-based selectivity, recovering much of attention's behavior on associative-recall and induction-head tasks at $\mathcal{O}(1)$-per-step inference.

The cost is structural and shows up in three places. *(i) No fast convolutional path.* Without a single global kernel, training instead requires *parallel scans* [38, 1] to recover parallelism during training, which require hardware-aware kernels to stay competitive with the FFT path. *(ii) Causal in* 1D. Any non-autoregressive use (1D vision, audio, BERT-style encoders, diffusion) must run the recurrence twice –on the input and its reversal– doubling compute and memory. *(iii) No natural generalization to* ND. Multi-dimensional adaptations such as Vision Mamba [52], VMamba [19] and Mamba-ND [17] must rasterize the input into one or several ad-hoc 1D scan orders, which destroy the spatial structure of the input and incur costs proportional to the number of scan orders used.

These costs make the recurrent branch a specialist for 1D non-LTI data (e.g., text) rather than a universal subquadratic primitive. Other recurrent models such as Mamba-2 [3], RetNet [40], GDA [48], KDA [42], and linear attention [15, 32] fall in this family and share the trade-offs outlined here.

2.3 Generalized Convolutional Models

Generalized convolutional models such as CKConv [36] and Hyena [34] take the converse direction: they sacrifice the recurrence in exchange for a freely learned kernel. They retain the convolutional form $\mathbf{y}{=}K * \mathbf{x}$ but replace the constraining SSM kernel $(\mathbf{cb}, \mathbf{cAb}, ..., \mathbf{cA}^{L-1}\mathbf{b})$ with an arbitrary function $K_i{=}f_\theta(\mathbf{c}_i)$ learned over coordinates $\mathbf{c} \in \mathbb{R}^N$.

The benefits are structural and show up in three places. *(i) Fast convolutional path.* The single global kernel keeps the fast FFT path intact. *(ii) Natural ND generalization.* Convolutions extend naturally to multi-dimensional signals: an ND input is convolved with a freely learnable kernel parameterized in ND –no ad-hoc 1D rasterization. *(iii) LTI inductive bias.* The operator is shift-equivariant by construction, a useful prior for the approximately-LTI signals that dominate natural and multi-dimensional data such as audio, vision, PDEs, medical volumes, etc.

Beyond the fast path, the absence of a recurrent state has a second structural consequence: long-range routing is direct. An output at position j is a linear combination of inputs weighted by the kernel, so propagating a value across distance δ is just kernel mass at offset δ – with no fixed-size hidden state to compress through. Recurrent operators must instead carry information through state updates, a bottleneck that becomes especially acute in multi-dimensional settings where dependencies span the full grid. The simple_copy results in §5.1 make this concrete: Mamba (bidir) degrades to random-baseline territory on 3D copy, while HyenaND maintains 4-orders-of-magnitude lower error.

The standard concern with this branch is input independency: the same K is used for *all* inputs. This is not an inherent property of the family, but rather an artifact of current instantiations, which inherit the autoregressive 1D assumption that conditioning must happen at the token level. In non-autoregressive settings (diffusion, multi-dimensional data) the input is available at once and the kernel can be conditioned on it: $K_i(\mathbf{x}){=}f_\theta\big(\mathbf{c};\, z(\mathbf{x})\big)$, to obtain sample-level input dependency while preserving LTI and the FFT path.

Together, the properties outlined in this section make the convolutional branch the natural subquadratic primitive for approximately LTI and multi-dimensional data, as well as bidirectional tasks.

2.4 The Subquadratic Landscape at a Glance

In conclusion, the SSM equivalence breaks into two specialist branches: recurrent for 1D non-LTI sequences (text); convolutional for approximately-LTI, multi-dimensional signals and bidirectional tasks. Figure 2 consolidates the design space.

3 HyenaND

3.1 Background: Hyena and CKConv

Hyena. Hyena [34] replaces attention with a data-controlled linear operator built from long implicit convolutions interleaved with elementwise gating. For a 1D input $\mathbf{x} \in \mathbb{R}^{L\times H}$, an order-2 Hyena operator computes $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^{L\times H}$ from $\mathbf{x}$ as the composition of a linear projection and a short causal depthwise convolution, then combines them through gating and a causal long FFT convolution:

$$\mathbf{y} = \sigma(\mathbf{v}) \;\odot\; \big(K * (\mathbf{q} \odot \sigma(\mathbf{k}))\big). \tag{2}$$

σ is a pointwise non-linearity –set to the identity in Hyena–, $\odot$ is elementwise multiplication, and $K \in \mathbb{R}^{L\times H}$ is a global convolutional kernel evaluated via FFT convolution in $\mathcal{O}(L \log L)$ time.

Continuous convolutional kernels. A key design choice in Hyena is the parameterization of K. Following CKConv [36, 35], Hyena defines the kernel as the pointwise product $K_i{=}w(c_i) \odot f_\theta(c_i)$ of a small MLP $f_\theta : \mathbb{R} \to \mathbb{R}^H$ and a non-learnable exponential decay window $w(c_i){=}\exp(-\boldsymbol{\alpha}\, c_i)$ with evenly spaced per-channel decay rates $\boldsymbol{\alpha} \in \mathbb{R}^H_{>0}$ evaluated on a grid of coordinates $\{c_i\}_{i=1}^L \subset \mathbb{R}_{\geq 0}$. f_θ learns the kernel's functional form, while w promotes learning features at multiple scales.

3.2 The HyenaND Operator

For an N-dimensional input $\mathbf{x} \in \mathbb{R}^{L_1\times\cdots\times L_N\times H}$, our HyenaND operator (Eq. 3) computes $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^{L_1\times\cdots\times L_N\times H}$ from $\mathbf{x}$ as the composition of a linear projection and a short depthwise convolution. Before the inner gate, $\mathbf{q}$ is ℓ_2-normalized along the channel axis to bound the magnitude that the linear projection leaves unconstrained; $\mathbf{k}$ is left unnormalized as SiLU already plays the magnitude-control role on its side of the gate. The gated signal $\ell_2(\mathbf{q}) \odot \mathrm{SiLU}(\mathbf{k})$ is then RMSNormed [50] before the long convolution to stabilize the per-token product magnitudes that enter the FFT –the same reasoning that led HyenaPixel [39] to place a LayerNorm at this position. The outer gate uses $\mathrm{Sigmoid}$ for a strictly bounded $[0, 1]$ selection on each channel of the convolution output, so that the gate softly attenuates without further amplifying.[1] Finally, as customary in many subquadratic operators [10, 48],

[1] Gating is more than a stabilizer: the gated Hyena mixer computes exactly the class of *separable* attention matrices, connecting gated Hyena to the literature on low-displacement-rank operators. See Appendix B.1.

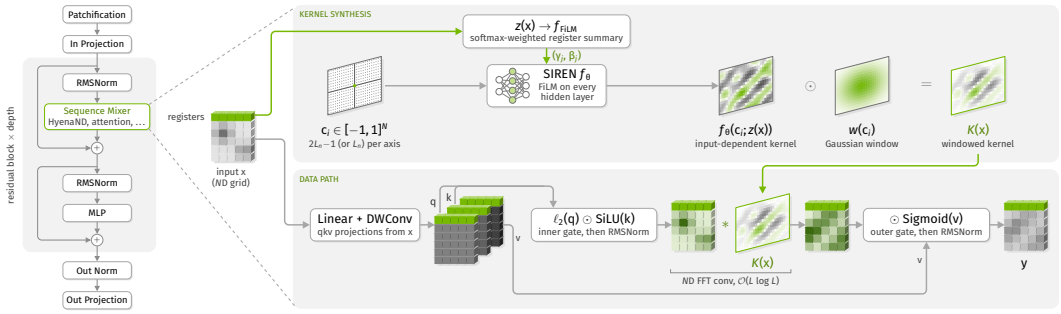


Figure 3: The HyenaND operator. The input is an ND token grid with R register tokens prepended along the first axis. *Top (kernel synthesis):* kernel-grid coordinates $\mathbf{c}_i$ feed a SIREN f_θ whose hidden layers are FiLM-conditioned on $z(\mathbf{x})$; a learned Gaussian window w masks its output, yielding the kernel $K(\mathbf{x})$ (§3.2.1–3.2.2). *Bottom (data path):* the input is projected into $\mathbf{q}, \mathbf{k}, \mathbf{v}$; the inner gate $\ell_2(\mathbf{q}) \odot \mathrm{SiLU}(\mathbf{k})$ is normalized and convolved with $K(\mathbf{x})$ in $\mathcal{O}\big(\prod_n L_n \log \prod_n L_n\big)$, then gated by $\mathrm{Sigmoid}(\mathbf{v})$ and RMSNormed to yield $\mathbf{y}$. *Left:* the operator is the sequence mixer of each pre-norm residual block, interchangeable with attention or Mamba.

RMSNorm is used before the residual addition to keep the operator's contribution bounded. Without it, the long convolution could amplify its input by up to $\|K\|_2$ and the residual stream's magnitude would grow with depth. The HyenaND operator (Fig. 3) is given as:

$$\mathbf{y} = \mathrm{RMSNorm}\big(\mathrm{Sigmoid}(\mathbf{v}) \odot (K * \mathrm{RMSNorm}(\ell_2(\mathbf{q}) \odot \mathrm{SiLU}(\mathbf{k})))\big). \tag{3}$$

3.2.1 Multi-Dimensional Continuous Convolutional Kernels

Just as in Hyena, we parameterize the convolutional kernel K as the pointwise product $K_i{=}w(\mathbf{c}_i) \odot f_\theta(\mathbf{c}_i)$ of a coordinate-based MLP f_θ and a window function w, but add several modifications to improve learning and adapt to multi-dimensional signals.

Joint N-dimensional parameterization. We define f_θ is a joint function in $\mathbb{R}^N$ rather than a separable axis-product $f_\theta(\mathbf{c}_i){=}\prod_{n=1}^{N} f_{\theta,n}(c_{i,n})$ of S4ND [22] and the product variant of Zimerman and Wolf [53], which can only express rank-1 spatial structure. This lets the kernel learn multi-dimensional patterns such as oriented edges, isotropic blurs, or anisotropic PDE propagation kernels.

Coordinate-based MLP f_θ. HyenaND adapts the 1D Hyena kernel of §3.1 to ND by replacing the 1D MLP $f_\theta : \mathbb{R} \to \mathbb{R}^H$ with a joint N-dimensional SIREN [37] $f_\theta : \mathbb{R}^N \to \mathbb{R}^H$ evaluated on the kernel grid. Let $\mathbf{c}_i \in [-1, 1]^N$ be a normalized coordinate vector indexing a position on the kernel grid. A SIREN with J layers maps grid coordinates $\mathbf{c}_i$ to a continuous convolutional kernel as:

$$f_\theta(\mathbf{c}_i) = \mathbf{W}_{\text{out}}\, \sin\big(\mathbf{W}_J \sin(\cdots \sin(\boldsymbol{\omega_0}\mathbf{W}_1\mathbf{c}_i))\big), \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{N \times H_{f_\theta}}, \mathbf{W}_2, ..., \mathbf{W}_J \in \mathbb{R}^{H_{f_\theta} \times H_{f_\theta}}, \mathbf{W}_{\text{out}} \in \mathbb{R}^{H_{f_\theta} \times H}$ are learned weight matrices and $\boldsymbol{\omega_0} \in \mathbb{R}^{H_{f_\theta}}$ is a frequency scaling factor. In contrast to CKConv and SIREN, which use a single non-learnable scalar ω_0 shared across all channels, we parameterize $\boldsymbol{\omega_0}$ as a learnable frequency scaling vector to allow different channels to specialize in different frequency bands.

Block-diagonal multi-frequency initialization. A scalar ω_0 shared across all rows confines SIREN's spectrum to a narrow band at initialization, limiting the kernel's ability to simultaneously resolve low- and high-frequency content. HyenaND broadens this coverage by structuring $\boldsymbol{\omega_0}$ into B equal blocks initialized to base frequencies on a linear $[\omega_0^{\min}, \omega_0^{\max}]$ schedule. Hidden and output layers are initialized near-block-diagonal so each block approximately behaves like an independent SIREN tuned to its own band at initialization. Off-block entries are scaled by a small value $\rho{=}0.1$ but remain free to be updated via gradient flow. Each row of $\boldsymbol{\omega_0}$ is further allowed to scale its block's base frequency by a factor $s \in [s_{\min}, s_{\max}]$, where $s_{\min}{>}0$ prevents the first-layer sine from collapsing to zero, and $s_{\max}$ bounds the highest effective frequency $s_{\max}\,\omega_0^{\max}$ to stay below the grid Nyquist limit to avoid aliasing. A full analysis and parameter choices are provided in Appx. B.3.

Window function w. The window w controls the length at which different channels of the kernel operate. We replace Hyena's static exponential decay $w(c_i){=}\exp(-\boldsymbol{\alpha} c_i)$ with a zero-centered Gaussian window $w(c_i){=}\exp(-\frac{1}{2}c_i^2/\boldsymbol{\sigma}^2)$, parameterized by a learnable per-channel variance $\boldsymbol{\sigma}^2 \in \mathbb{R}^H_{>0}$.

Building on the $\boldsymbol{\omega}_0$ initialization above, we initialize $\boldsymbol{\sigma}^2$ to be block-aligned with the $\boldsymbol{\omega}_0$ schedule such that rows in the lowest-ω_0 block receive the widest initial bandwidths, with bandwidths shrinking as the block frequency increases. We choose Gaussian over exponential decay for two reasons: Gaussian windows are standard in signal processing for their isotropic decay and minimal time-frequency uncertainty product, and they avoid the harsher decay that the exponential axis-product exhibits in multiple dimensions. For ND signals, we give the variance vector an N-dimensional form $\boldsymbol{\sigma}^2 \in \mathbb{R}^{N\times H} > 0$, such that each spatial axis can adapt its receptive field independently during training. Ablations in Appx. B.3 empirically validate our choice of Gaussian over exponential windows.

Support for circular and non-circular convolutions. FFT-based convolution is naturally circular: it implicitly treats the grid as a torus, so a kernel positioned near the right edge of an image pulls in values from the left edge of the same row, as if they were stitched together. Many physical signals – fluid simulations, magnetohydrodynamics, reaction-diffusion systems – exhibit periodic boundary conditions, for which circular convolutions naturally match their geometry. In such cases, global context on a signal $\mathbf{x} \in \mathbb{R}^{L_1\times\cdots\times L_N\times H}$ is obtained by evaluating the kernel K on a grid $L_1\times\ldots\times L_N$ (equal to the input). For data with no adjacency between opposite edges – natural images, 3D medical scans, physical systems with closed boundaries – circular warping contradicts the data's geometry: the operator would treat opposite edges as neighbors, leaking content across an artificial seam. In such cases, we evaluate the kernel K on a double grid $(2L_1-1)\times\ldots\times(2L_N-1)$ and zero-pad the input to the same shape before the FFT to recover a non-circular convolution at constant overhead. `nvSubquadratic-ops` (§4) fuses padding, FFT, kernel evaluation and spectral modulation into a single CUDA kernel that keeps intermediates in on-chip shared memory at the SM level, making this overhead negligible.

Resolution independence. Since f_θ and w are both continuous, the same convolutional kernel can be evaluated on grids of any size or aspect ratio without retraining [36]. We prove a $\mathcal{O}(\mathcal{L}_K\, b\, l\, |\Delta l|/L_0)$ bound on the output shift when the input length L deviates from the training length L_0 by $|\Delta l|$. $\mathcal{L}_K$ denotes the Lipschitz constant of the SIREN kernel and b bounds the input norms (Appx. B.2).

Scaling $\boldsymbol{\omega}_0$ across resolutions and dimensionalities. While a trained kernel can generalize to new resolutions seamlessly, initializing a network for a new target grid resolution L' or input dimensionality N' requires care. The per-block schedule on ω_0 must be adapted at initialization to preserve the same fractional spectral coverage relative to the grid's Nyquist limit. We derive a uniform schedule $\boldsymbol{\omega}_0' = \boldsymbol{\omega}_0 (L'/L)\sqrt{N'/N}$ (full derivation in Appx. B.3): the L'/L factor preserves the Nyquist-relative coverage as the grid density changes, and the $\sqrt{N'/N}$ factor compensates for the $1/\sqrt{N}$ shrinkage of the spectrum that arises from the SIREN initialization $\mathbf{W}_1 \sim \mathcal{U}(-1/N, +1/N)^N$.

3.2.2 Input-Dependent Kernels via Registers and FiLM

Hyena uses the *same* convolutional kernel K for every input $\mathbf{x}$, limiting input adaptation. HyenaND removes this restriction extending the static kernel of §3.2.1 to an input-dependent kernel $K_i(\mathbf{x}) = w(\mathbf{c}_i) \odot f_\theta\big(\mathbf{c}_i;\, z(\mathbf{x})\big)$, where $z(\mathbf{x}) \in \mathbb{R}^H$ is a *control* variable extracted from $\mathbf{x}$. Since $z(\mathbf{x})$ does not depend on specific coordinates i, $K(\mathbf{x})$ is computed once per input and applied at every position.

FiLM-conditioned kernel. We use Feature-wise Linear Modulation (FiLM) [33] to condition the kernel f_θ on the control variable $z(\mathbf{x})$. A 2-layer MLP $f_{\mathrm{FiLM}} : \mathbb{R}^H \to \mathbb{R}^{2JH_{f_\theta}}$ uses $z(\mathbf{x})$ to produce per-layer scales and shifts $(\boldsymbol{\gamma}_j, \boldsymbol{\beta}_j)_{j=1}^J$, that modulate the SIREN hidden activations $\{\mathbf{h}_j\}_{j=1}^J$ as:

$$\mathbf{h}_j \;\leftarrow\; \boldsymbol{\gamma}_j \odot \mathbf{h}_j + \boldsymbol{\beta}_j, \qquad (\boldsymbol{\gamma}_j, \boldsymbol{\beta}_j)_{j=1}^J = f_{\mathrm{FiLM}}\big(z(\mathbf{x})\big). \tag{5}$$

Following Perez et al. [33], we initialize the FiLM scales and shifts as $(\boldsymbol{\gamma}_j, \boldsymbol{\beta}_j) = (\mathbf{1}, \mathbf{0})$ to reduce to an unmodulated kernel at initialization.

The control variable $z(\mathbf{x})$. Squeeze-and-Excitation networks [13, 51] modulate activations via global pooling. We find such summaries to be too crude for kernel conditioning, as pooling discards the spatial structure that the kernel is meant to specialize on. Instead, we concatenate R learnable *register* tokens [5] $\{\mathbf{e}_r\}_{r=1}^R \in \mathbb{R}^H$ to the input and use them for construction of $z(\mathbf{x})$. Being processed jointly at every layer, registers integrate information from the entire input, and training shapes them into the summaries that are the most useful for kernel conditioning. Let $\{\mathbf{e}'_r(\mathbf{x})\}_{r=1}^R$ denote the activations of the registers the present layer. The control variable $z(\mathbf{x})$ is computed as a softmax-weighted average over the registers $z(\mathbf{x}) = \sum_{r=1}^R \mathrm{softmax}(\boldsymbol{\lambda})_r\, \mathbf{e}'_r(\mathbf{x})$, with $\boldsymbol{\lambda} \in \mathbb{R}^R$ learnable.

To preserve the ND structure of the input, registers are prepended as additional rows above the input grid. For a 2D input of shape $L_1 \times L_2$, the R registers (with minimal zero-padding to align boundaries) fill $\lceil R/L_2 \rceil$ additional rows above the original $L_1 \times L_2$ patch grid.

FFT path preservation. Since $z(\mathbf{x})$ is computed once per input instance, the modulated kernel $K(\mathbf{x})$ is used across all input positions. Consequently, the FFT-convolution in (3) proceeds identically to the unmodulated case with the only difference that it now uses B different convolutional kernels, one for each of the inputs in a batch of size B. Concretely, $K(\mathbf{x})$ is evaluated on the kernel grid (sized per §3.2.1) and applied via a single ND FFT call in $\mathcal{O}\left(\prod_n L_n \log \prod_n L_n\right)$. This is in contrast to per-position input dependency, which would be incompatible with the FFT path (§2.1).

4 CUDA Implementation: `nvSubquadratic-ops`

Asymptotic complexity is necessary but not sufficient for practical scaling of an ML architecture: a theoretically cheaper operator is useful only when its implementation maps efficiently onto modern accelerator hardware. This distinction is especially sharp for FFT-based operators. GPU matrix-multiply throughput has generally grown faster than off-chip memory bandwidth [24–27], which strongly benefits standard attention: the dominant SDPA operations ($\mathbf{QK}^\top$ and $\mathbf{PV}$) are GEMM-like, and FlashAttention-style fused kernels avoid materializing the full $L \times L$ attention matrix in HBM, making quadratic attention far more competitive than its $O(L^2)$ complexity alone would suggest [4, 49]. FFT-based operators face a different performance profile: their fast path depends less on GEMM throughput and more on batched FFT efficiency, memory layout, and bandwidth. Attempts to use tensor-cores for FFT generally lag behind cuFFT and cuFFTDx, especially for real FFTs [9, 28, 29], making the practical crossover point over fused SDPA hardware- and implementation-dependent.

To close this asymptotic-vs-wall-clock gap, `nvSubquadratic-ops` implements a suite of CUDA kernels for FFT-convolution that fuse forward FFT, spectral modulation, inverse FFT, and output segmentation into a single operation. The fused kernels keep intermediate representations within fast but capacity-limited on-chip memory (see Appx. C for the full kernel description). `nvSubquadratic-ops` provides a unified API for fast long convolutions in 1D, 2D and 3D. Furthermore, we fuse a critical path often overlooked in Hyena operators: the generation of the kernel K. In our Evo2 ablations (Appx. C.2), our fused generation pipeline reduces runtime by up to $40\times$ and HBM traffic by up to $32\times$.

5 Experiments

5.1 Controlled Spatial-Recall Experiments

Before turning to downstream tasks, we use a controlled family of probes to read off the ND spatial abilities of Attention, HyenaND and Mamba at parameter parity ($\sim$1.9M, 4 blocks, 50k iters). Two tasks, defined identically in 1D/2D/3D, stress complementary demands: `simple_copy` isolates long-range copy capabilities, whereas `color_cond` evaluates spatial associative recall (Fig. 11 in Appendix D.1).

As shown in Table 1, HyenaND outperforms both Attention and Mamba across all tasks. The advantage over Mamba is one of geometry: Mamba must rasterize ND data into an ad-hoc 1D scan order, and this reduces accuracy when the task structure lives in the native geometry. On 3D `simple_copy` it barely improves on the mean-predictor baseline (0.451 vs. 0.482), while HyenaND, operating directly on the native geometry, solves the task ($1.2\text{e}{-4}$). Attention fares worst: at native resolution it is only marginally better than random (e.g., 0.253 vs. 0.284 on 2D `color_cond`) and cannot run at all in 3D. We attribute this to attention's database-inspired design: retrieval works by comparing queries against keys, which requires individual tokens to be information-dense. Yet, on a raw canvas each token is a single pixel, so the information-to-token ratio is far too low. Consistent with this reading, non-overlapping patchification, which packs many pixels into each token, makes attention competitive (though the optimal patch size remains a function of the data and the task), and the same mechanism explains why patchification is a key component on ViTs in practice (see Appendix D.1 for additional experiments and details about spatial recall tasks).

5.2 Long-Context Genomics (1D)

We train three replicates each of 1B-parameter Evo2-style striped-Hyena language models with $k \in \{0, \ldots, 4\}$ attention layers interleaved among Hyena layers (H_k), and a full-attention baseline

Table 1: Spatial-recall summary at native resolution (no patchification). MSE (↓)

	1D (L=4096)		2D (64^2)		3D motion (32^3)	
	simple	color	simple	color	simple	color
Random (mean pred.)	0.549	0.284	0.549	0.284	0.482	0.257
Attention (RoPE)	0.150	0.274	0.092	0.253	OOM	OOM
Mamba (bidir)	0.085	6.5×10^{-3}	0.349	8.3×10^{-3}	0.451	1.0×10^{-4}
HyenaND (ours)	5.2×10^{-5}	6.2×10^{-3}	1.3×10^{-4}	2.5×10^{-3}	1.2×10^{-4}	5.7×10^{-5}

(T), on the OpenGenome2 metagenomics subset [2] at 8K context (Appendix D.2.1). Every Hyena configuration significantly outperforms the transformer in final validation perplexity after Holm–Bonferroni correction (Table 2), consistent with the Evo scaling law [23]; H_2 is best (2.77 ± 0.0006, $p = 4.3 \times 10^{-4}$), and the ranking already holds at step 9,899, before the transformer runs required an LR reduction (Appendix D.2.2).

The $\mathcal{O}(L \log L)$ FFT path then takes the same architecture to context lengths at which attention's quadratic cost is prohibitive. Starting from an 8K-pretrained 1B striped-Hyena model, we double the context stage by stage to 16M tokens on up to 128 GB200 GPUs, sharding the sequence with context parallelism (Appendix D.2.3, Table 10). Per-GPU throughput is flat over a $128\times$ range of context (442–461 TFLOP/s/GPU from 128K to 16M); since the recompute-free peak is $\sim$600 TFLOP/s/GPU up to 8M, the as-run figure is set by memory, not by the operator. Every stage trains to convergence (validation NLL $1.075 \to 1.058$ at 16M; Fig. 13), and the converged 16M model *uses* its context: per-token NLL keeps decreasing with position out to millions of tokens (Fig. 14). At this length an entire bacterial genome, or a chromosome-scale eukaryotic fragment, fits inside a single training window, bringing genome-scale dependencies within reach of end-to-end training.

Table 2: Parametric statistical comparison of model Architectures using one-sided Welch's t-test (H_1: mean(model) $<$ mean(reference)), with Holm–Bonferroni correction applied per reference column (5 tests each; self-comparisons excluded). While we lack the power to confirm the nominal improvement of 2 MHA layers over any of the other striped hyena configurations, every hyena configuration shows superior accuracy to the full transformer on this task, even after multiple hypothesis correction.

Model Architecture	**Val. PPL (Mean ± SD)**	**p vs.** T	**p vs.** H_0	**Holm p vs.** T	**Holm p vs.** H_0
T (Full Transformer, 25 MHA)	2.9235 ± 0.0039	5.000×10^{-1}	9.873×10^{-1}	—	1.000
H_4 (Mixed, 4 MHA layers)	2.8312 ± 0.0088	3.800×10^{-4}	5.650×10^{-1}	1.520×10^{-3}	1.000
H_3 (Mixed, 3 MHA layers)	2.8214 ± 0.0313	1.417×10^{-2}	3.974×10^{-1}	2.546×10^{-2}	1.000
H_2 **(Mixed, 2 MHA layers)**	2.7729 ± 0.0006	8.663×10^{-5}	3.760×10^{-2}	4.332×10^{-4}	1.880×10^{-1}
H_1 (Mixed, 1 MHA layer)	2.8308 ± 0.0108	9.468×10^{-4}	5.558×10^{-1}	2.840×10^{-3}	1.000
H_0 (Pure Hyena, 0 MHA)	2.8282 ± 0.0279	1.273×10^{-2}	5.000×10^{-1}	2.546×10^{-2}	—

5.3 Computer Vision: ImageNet (2D)

We evaluate HyenaND on ImageNet-1K [6] classification using the ViT-5-Small backbone [46] (22M parameters). We compare pure HyenaND, two hybrids ($(\mathrm{H\,A})^{\times 6}$ and $(\mathrm{H\,H\,H\,A})^{\times 3}$) and the attention baseline, none with patch-merge blocks (details in Appendix D.3.1).

Table 3 reports patch 16×16 results to match the standard training regime of external baselines; the primary result is the patch-size ablation in Table 11 (Appendix D.3.2), sweeping $\{16, 8, 4, 2\}$ and reporting top-1, GFLOPs, and throughput. Because attention converges slowly, we trained all our models for 800 epochs to ensure full convergence, while external baselines (DeiT, Mamba-ViT, etc.) report 300-epoch results. At patch 16×16, pure HyenaND achieves 81.5% top-1, matching ViT-5-Small (81.8%), the first native-2D subquadratic operator to match ViT on ImageNet without rasterizing the spatial grid or patch-merging blocks. In the patch-size ablation, both our attention baseline and HyenaND clearly outperform all external baselines at comparable compute; HyenaND additionally offers FLOP savings compared to attention that grow with sequence length, 14% at patch 8×8 (39.0 vs. 45.5 GFLOPs), 51% at patch 4×4 (155 vs. 317 GFLOPs), and 82% at patch 2×2 (623 vs. 3,444 GFLOPs), while remaining tractable at very small patch sizes enabled by the fused 2D FFT kernels of Appendix C.4. The hybrid $(\mathrm{H\,A})^{\times 6}$ makes this trade-off concrete: it matches the attention baseline to within 0.1% top-1 while using $\sim$40% fewer FLOPs (e.g. 85.2% at 2,033 vs. 85.3% at $3{,}444$ GFLOPs for p=2), inheriting attention's selectivity at a fraction of its cost. Beyond this FLOP trade-off, at matched hybrid ratios the HyenaND-attention hybrids also outperform recurrence-based Mamba-attention hybrids (Table 3; full patch sweep in Table 11): 82.1% vs. 81.8% at the 1:1 ratio and 82.0% vs. 81.0% at the 3:1 ratio (patch 16×16), even though the bidirectional Mamba hybrids carry markedly more parameters (30M/34M vs. 22M).

Table 3: ImageNet-1K top-1 accuracy at 224×224. For our models, results are at the best-performing patch size; see Table 11 for the full ablation (incl. GFLOPs and img/s). All HyenaND models use the ViT-5-Small backbone. †Same backbone and training recipe as HyenaND.

Model	Type	Patch	Merge	Params	Top-1 (%)
DeiT-S [43]	Attention	16×16	False	22M	79.8
Swin-T [20]	Attention	4×4	True	29M	81.3
ConvNeXt-T [21]	Convolution	4×4	True	29M	82.1
Vim-S [52]	GRM (1D scan)	16×16	False	26M	80.3
VMamba-T [19]	GRM (1D scan)	4×4	True	30M	82.6
ViT-5-Small (Attention)†	Attention	16×16	False	22M	81.80
Mamba-S (MA)$^{\times 6}$	Hybrid	16×16	False	30M	81.80
Mamba-S (MMMA)$^{\times 3}$	Hybrid	16×16	False	34M	81.00
HyenaND-S (pure)	GCM (2D)	16×16	False	22M	81.50
HyenaND-S (HA)$^{\times 6}$	Hybrid	16×16	False	22M	82.10
HyenaND-S (HHHA)$^{\times 3}$	Hybrid	16×16	False	22M	82.00

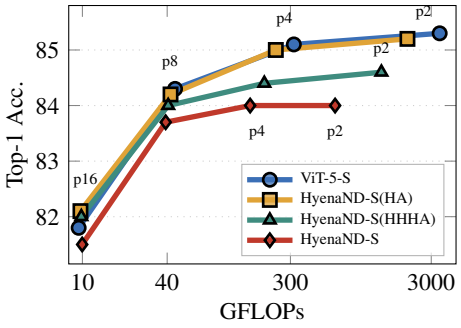


Figure 4: **ImageNet Top-1 vs. compute.** Each curve sweeps patch size $p \in \{16, 8, 4, 2\}$; smaller patches yield more tokens and more FLOPs.

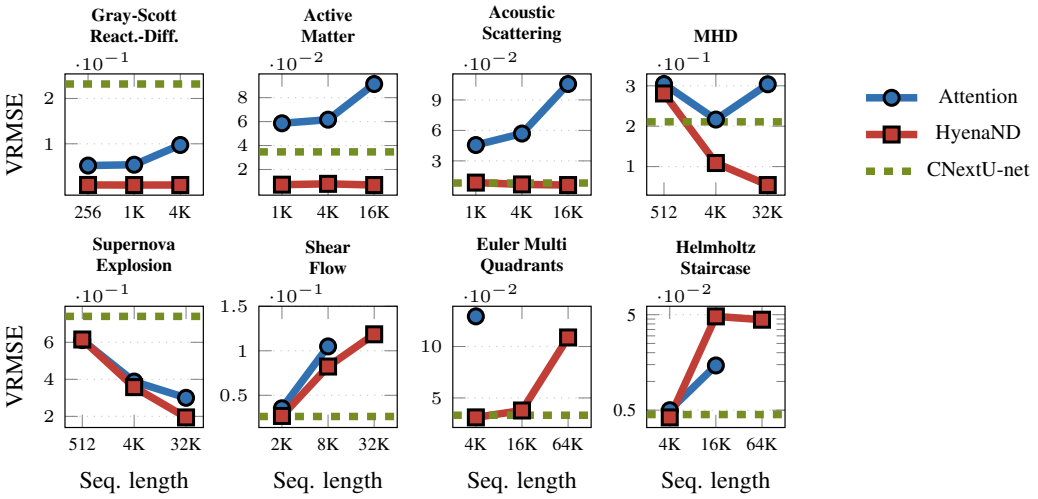


Figure 5: **VRMSE vs. sequence length on The Well.** Sequence length is controlled by patch size: smaller patches p yield more tokens. Each panel shows one dataset (lower is better). HyenaND (red) improves as sequence length grows on most datasets. CNextU-net (green dashed) is a fixed full-resolution baseline with no patch-size axis. For Shear Flow, the Attention p=2 result is omitted (undertrained run).

5.4 PDE Surrogate Modeling on The Well (2D/3D)

We assess HyenaND on eight PDE datasets from The Well [31], spanning fluid dynamics, MHD, astrophysics, acoustics, and biological pattern formation. The datasets were chosen to cover resolutions of 64–1024-pixel on 2D and 3D Cartesian grids. We compare HyenaND against a full-resolution ConvNeXt U-Net baseline and a matched Attention model across patch sizes $p \in \{2, 4, 8\}$ for total run times of 24 hours per experiment. Full results are compiled in Table 12 (Appendix D.4), together with architecture and training details. Following the benchmark protocol, we report variance-normalized RMSE (VRMSE), scaled so that the mean-field predictor scores 1.

HyenaND achieves the lowest VRMSE on all eight datasets, as detailed in Table 12 (Appendix D.4). With increasing input length HyenaND improves over the best Attention configuration by $5.8\times$ on `gray_scott_reaction_diffusion`, $8.4\times$ on `active_matter`, $7.4\times$ on `acoustic_scattering`, $4.0\times$ on `MHD`, and $1.5\times$ on `supernova_explosion`, and beats the full-resolution CNextU-net baseline on every dataset. For `shear_flow`, `euler_multi_quadrants` and `helmholtz_staircase`, best performance was achieved with larger patch sizes but still outperforms attention and CNextU-net. We attribute this to the fixed frequency parameter, which was kept constant across all settings without individual tuning. Figure 5 shows VRMSE per dataset as a function of sequence length. The scaling superiority of HyenaND in throughput becomes evident with longer sequences. For example, at L>16k, HyenaND achieves more samples/sec than Attention by factors ranging from $2\times$–$10\times$.

Against CNextU-net, HyenaND's lead depends on sequence length: CNextU-net is comparable on `acoustic_scattering` and `MHD` at p=8, with the gap opening up as patch-size shrinks.

5.5 3D Medical Image Segmentation

We retrofit HyenaND into a SwinUNETR-style four-stage volumetric encoder, replacing per-stage windowed attention with D=3 HyenaND according to four attention/Hyena stage patterns $s \in \{A, H\}^4$. We evaluate on the PanTS dataset [18], a 28-class abdominal segmentation benchmark using 901 held-out volumes. Training details and dataset statistics are in Appendix D.5.

All three Hyena-containing variants peak at $\sim\mathbf{58.7}$ GB of allocated memory per GPU during training, $\mathbf{10.8}\%$ below the 65.79 GB attention baseline, whichever stages carry Hyena. The dominant memory cost of windowed attention at moderate ROI is the materialized attention matrix; an FFT-based global mixer eliminates that term without changing activation width. The decoder is identical across variants and accounts for $\sim$590 of the $\sim$625 GFLOPs, so total savings look modest (2–3%); encoder-only savings are $\sim$30%.

Table 4: PanTS 3D segmentation (ROI= 96^3, 100 epochs). Mean Dice across 28 classes and 901 held-out test volumes; peak per-GPU allocated memory during training (batch= 32, 4×GB200); forward-pass GFLOPs at batch 1, including an analytical FFT-convolution correction for Hyena.

Variant	Mean Dice ↑	± SEM	Peak alloc. mem. (GB) ↓	Fwd. GFLOPs ↓
AAAA (SwinUNETR baseline)	0.7496	0.0053	65.79	638
HHHH (all-Hyena)	0.7510	0.0110	**58.69**	**621**
HAHA (striped hybrid)	0.7535	0.0054	58.70	627
HHAA (hierarchical hybrid)	**0.7559**	**0.0052**	58.67	623

The all-Hyena variant matches the attention baseline (0.7510 vs. 0.7496), sufficient to establish Hyena as a drop-in replacement, and both hybrids strictly outperform both pure variants: **HAHA** reaches 0.7535 and **HHAA** reaches 0.7559: +0.63 pp and +0.49 pp over the pure attention and Hyena baselines. Paired Wilcoxon signed-rank tests on per-subject Dice differences confirm that all three Hyena-containing variants significantly outperform the attention baseline: **HHHH** vs. **AAAA**: p=2.4×10^{-6}; **HAHA** vs. **AAAA**: p=9.1×10^{-16}; **HHAA** vs. **AAAA**: p=5.7×10^{-24}.

6 Conclusion

The prevailing view of the subquadratic landscape conflates an operator's class with its computational form. We argued this equivalence holds for one class only, the LTI SSM, and that relaxing it yields two complementary branches. The recurrent branch buys per-token selectivity at the price of LTI, causality, a 1D scan order, and a fixed-size state through which all long-range information must pass. The convolutional branch retains LTI, generalizes natively to ND, and routes information across any distance as kernel mass at the corresponding offset. Two objections have kept it from serving as the subquadratic primitive for approximately-LTI ND data, and we remove both. Conceptually, long convolutions were regarded as input-independent, an objection inherited from autoregressive 1D processing [34]; in non-autoregressive settings the input is available at once, and HyenaND conditions the kernel on it through register-based FiLM while preserving the single-FFT path. Practically, the $\mathcal{O}(L \log L)$ cost of FFT convolutions is routinely defeated by HBM round-trips; `nvSubquadratic-ops` fuses this path under a unified 1D/2D/3D API, turning the asymptotic advantage into wall-clock speedups.

With these gaps closed, pure HyenaND matches strong attention and specialized baselines on genomics, ImageNet, 3D medical segmentation, and PDE modeling on The Well, and HyenaND–attention hybrids outperform both pure attention and recurrence-based hybrids. Finally, our results reframe patchification: it is a workaround for attention's quadratic cost that discards fine-grained structure before learning begins, not a necessary preprocessing step. HyenaND removes the need for it: a native ND, input-dependent, subquadratic operator can ingest data in its intrinsic geometry, as our 3D experiments show by mixing directly over $\sim$110k-token volumes. Looking ahead, this suggests a learned alternative to patchification itself, where a few HyenaND layers aggregate raw pixels or voxels into information-dense tokens for any downstream backbone. Limitations are discussed in Appx. A.

Acknowledgments and Disclosure of Funding

Funding. David R. Wessels is funded by a public grant of the Dutch Cancer Society (KWF) under subsidy 15059/2022-PPS2. This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-17835. All other compute was provided by NVIDIA.

Competing interests. F. Ramezanghorbani, A. Moradzadeh, O. Viessmann, J. St. John, K. Janik, Y. Tang and S. G. Paliwal are employees of NVIDIA, which develops the GPUs that `nvSubquadratic-ops` targets and on which all benchmarks in this paper were run. D. W. Romero is employed by Cartesia AI and D. M. Knigge by New Theory AI. The remaining authors declare no competing interests.

References

[1] Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990.

[2] Garyk Brixi, Matthew G. Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A. Gonzalez, Samuel H. King, David B. Li, Aditi T. Merchant, Mohsen Naghipourfar, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Gwanggyu Sun, Ali Taghibakshi, Anton Vorontsov, Brandon Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K. Wang, Etowah Adams, Stephen A. Baccus, Steven Dillmann, Stefano Ermon, Daniel Guo, Rajesh Ilango, Ken Janik, Amy X. Lu, Reshma Mehta, Mohammad R.K. Mofrad, Madelena Y. Ng, Jaspreet Pannu, Christopher Ré, Jonathan C. Schmok, John St. John, Jeremy Sullivan, Kevin Zhu, Greg Zynda, Daniel Balsam, Patrick Collison, Anthony B. Costa, Tina Hernandez-Boussard, Eric Ho, Ming-Yu Liu, Thomas McGrath, Kimberly Powell, Dave P. Burke, Hani Goodarzi, Patrick D. Hsu, and Brian L. Hie. Genome modeling and design across all domains of life with evo 2. *bioRxiv*, 2025. doi: 10.1101/2025.02.18.638918. URL `https://www.biorxiv.org/content/early/2025/02/21/2025.02.18.638918`.

[3] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024.

[4] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf`.

[5] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2023.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[8] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*, 2023.

[9] Daniel Y. Fu, Hermann Kumbong, Eric Nguyen, and Christopher Ré. FlashFFTConv: Efficient convolutions for long sequences with tensor cores. In *International Conference on Learning Representations (ICLR)*, 2024.

[10] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

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

[12] Ali Hatamizadeh, Vishwesh Nath, Yucheng Tang, Holger R Roth, and Daguang Xu. Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images. In *International MICCAI Brainlesion Workshop*, pages 272–284. Springer, 2022.

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[14] Peter St. John, Dejun Lin, Polina Binder, Malcolm Greaves, Vega Shah, John St. John, Adrian Lange, Patrick Hsu, Rajesh Illango, Arvind Ramanathan, Anima Anandkumar, David H Brookes, Akosua Busia, Abhishaike Mahajan, Stephen Malina, Neha Prasad, Sam Sinai, Lindsay Edwards, Thomas Gaudelet, Cristian Regep, Martin Steinegger, Burkhard Rost, Alexander Brace, Kyle Hippe, Luca Naef, Keisuke Kamata, George Armstrong, Kevin Boyd, Zhonglin Cao, Han-Yi Chou, Simon Chu, Allan dos Santos Costa, Sajad Darabi, Eric Dawson, Kieran Didi, Cong Fu, Mario Geiger, Michelle Gill, Darren J Hsu, Gagan Kaushik, Maria Korshunova, Steven Kothen-Hill, Youhan Lee, Meng Liu, Micha Livne, Zachary McClure, Jonathan Mitchell, Alireza Moradzadeh, Ohad Mosafi, Youssef Nashed, Saee Paliwal, Yuxing Peng, Sara Rabhi, Farhad Ramezanghorbani, Danny Reidenbach, Camir Ricketts, Brian C Roland, Kushal Shah, Tyler Shimko, Hassan Sirelkhatim, Savitha Srinivasan, Abraham C Stern, Dorota Toczydlowska, Srimukh Prasad Veccham, Niccolò Alberto Elia Venanzi, Anton Vorontsov, Jared Wilber, Isabel Wilkinson, Wei Jing Wong, Eva Xue, Cory Ye, Xin Yu, Yang Zhang, Guoqing Zhou, Becca Zandstein, Alejandro Chacon, Prashant Sohani, Maximilian Stadler, Christian Hundt, Feiwen Zhu, Christian Dallago, Bruno Trentini, Emine Kucukbenli, Saee Paliwal, Timur Rvachov, Eddie Calleja, Johnny Israeli, Harry Clifford, Risto Haukioja, Nicholas Haemel, Kyle Tretina, Neha Tadimeti, and Anthony B Costa. Bionemo framework: a modular, high-performance library for ai model development in drug discovery, 2025. URL `https://arxiv.org/abs/2411.10548`.

[15] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[16] Jerome Ku, Eric Nguyen, David W. Romero, Garyk Brixi, Brandon Yang, Anton Vorontsov, Ali Taghibakhshi, Amy X. Lu, Dave P. Burke, Greg Brockman, Stefano Massaroli, Christopher Ré, Patrick D. Hsu, Brian L. Hie, Stefano Ermon, and Michael Poli. Systems and algorithms for convolutional multi-hybrid language models at scale, 2025. URL `https://arxiv.org/abs/2503.01868`.

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

[18] Wenxuan Li, Xinze Zhou, Qi Chen, Tianyu Lin, Pedro RAS Bassi, Xiaoxi Chen, Chen Ye, Zheren Zhu, Kai Ding, Heng Li, et al. Pants: The pancreatic tumor segmentation dataset. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[19] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. VMamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[21] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

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

[23] Eric Nguyen, Michael Poli, Matthew G. Durrant, Brian Kang, Dhruva Katrekar, David B. Li, Liam J. Bartie, Armin W. Thomas, Samuel H. King, Garyk Brixi, Jeremy Sullivan, Madelena Y. Ng, Ashley Lewis, Aaron Lou, Stefano Ermon, Stephen A. Baccus, Tina Hernandez-Boussard, Christopher Ré, Patrick D. Hsu, and Brian L. Hie. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024. doi: 10.1126/science.ado9336. URL https://www.science.org/doi/abs/10.1126/science.ado9336.

[24] NVIDIA Corporation. NVIDIA Tesla V100 GPU Architecture. Technical Report WP-08608-001_v1.1, NVIDIA Corporation, August 2017. URL https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf. Whitepaper.

[25] NVIDIA Corporation. NVIDIA A100 Tensor Core GPU Architecture. Technical Report V1.0, NVIDIA Corporation, 2020. URL https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf. Whitepaper.

[26] NVIDIA Corporation. NVIDIA H100 Tensor Core GPU Architecture. Technical Report V1.03, NVIDIA Corporation, 2022. URL https://nvdam.widen.net/s/95bdhpsgrs/nvidia_h100_tensor_core_gpu_architecture_whitepaper_v1.03. Whitepaper.

[27] NVIDIA Corporation. NVIDIA Blackwell Architecture Technical Overview. Technical report, NVIDIA Corporation, 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture. Technical overview; covers GB200 Grace Blackwell Superchip, HGX B200, and HGX B100.

[28] NVIDIA Corporation. *cuFFT API Reference*. NVIDIA Corporation, cuda toolkit 13.2; cufft 12.2.0.46 edition, 2026. URL https://docs.nvidia.com/cuda/cufft/index.html. Last updated 2026-04-02; accessed 2026-04-30.

[29] NVIDIA Corporation. *cuFFTDx Documentation*. NVIDIA Corporation, mathdx 26.03.0 (cufftdx 1.7.0) edition, 2026. URL https://docs.nvidia.com/cuda/cufftdx/index.html. Accessed: 2026-04-30.

[30] NVIDIA Corporation. CUTLASS: CUDA Templates for Linear Algebra Subroutines. https://github.com/NVIDIA/cutlass, 2026. Version 4.3.5; released 2026-01-09; accessed 2026-04-30.

[31] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina J. Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B. Dalziel, Drummond B. Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich R. Kerswell, Suryanarayana Maddu, Jonah Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam H. Parker, Miles Cranmer, and Shirley Ho. The Well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024.

[32] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Cao, Xuan Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. In *Empirical Methods in Natural Language Processing*, 2023.

[33] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

[34] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

[35] David W. Romero, Robert-Jan Bruintjes, Jakub M. Tomczak, Erik J. Bekkers, Mark Hoogendoorn, and Jan C. van Gemert. FlexConv: Continuous kernel convolutions with differentiable kernel sizes. In *International Conference on Learning Representations (ICLR)*, 2022.

[36] David W. Romero, Anna Kuzina, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations (ICLR)*, 2022.

[37] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473, 2020.

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

[39] Julian Spravil, Sebastian Houben, and Sven Behnke. HyenaPixel: Global image context with convolutions, 2024. URL `https://arxiv.org/abs/2402.19305`.

[40] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Tengchao Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[41] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models, 2025. URL `https://arxiv.org/abs/2312.16903`.

[42] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[44] 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.

[45] Feng Wang, Yaodong Yu, Guoyizhe Wei, Wei Shao, Yuyin Zhou, Alan Yuille, and Cihang Xie. Scaling laws in patchification: An image is worth 50,176 tokens and more. *arXiv preprint arXiv:2502.03738*, 2025.

[46] Feng Wang, Sucheng Ren, Tiezheng Zhang, Predrag Neskovic, Anand Bhattad, Cihang Xie, and Alan Yuille. Vit-5: Vision transformers for the mid-2020s. *arXiv preprint arXiv:2602.08071*, 2026.

[47] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved baseline in 2.5 lines of code, 2021.

[48] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pages 29687–29707, 2025.

[49] Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling, 2026. URL `https://arxiv.org/abs/2603.05451`.

[50] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[51] Zilong Zhong, Zhong Qiu Lin, Rene Bidart, Xiaodan Hu, Ibrahim Ben Daya, Zhifeng Li, Wei-Shi Zheng, Jonathan Li, and Alexander Wong. Squeeze-and-attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[52] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision Mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

[53] Itamar Zimerman and Lior Wolf. Multi-dimensional Hyena for spatial inductive bias. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

A Limitations

Per-sample rather than per-position input dependence. The kernel $K(\mathbf{x})$ is conditioned on the register summary $z(\mathbf{x})$ once per input (§3.2.2). This is what preserves the single-FFT path, but it rules out the token-level selectivity that recurrent operators provide; where such selectivity is needed, our hybrids with attention layers supply it.

Initialization hyperparameters. The SIREN base frequencies $\boldsymbol{\omega_0}$ with their scale range $[s_{\min}, s_{\max}]$ (§3.2.1) and the Gaussian-window bandwidths $\sigma_{d,c}$ (Appx. B.3) are initialization hyperparameters to which performance is sensitive. We kept them fixed across The Well datasets and attribute the larger optimal patch sizes on `shear_flow`, `euler_multi_quadrants` and `helmholtz_staircase` to this choice.

Hardware. `nvSubquadratic-ops` targets NVIDIA GPUs (CuTe and cuFFTDx kernels, Appx. C). On other hardware HyenaND falls back to library FFT convolutions, which keeps its $\mathcal{O}(L \log L)$ complexity but forfeits the wall-clock speedups of §4.

B Appendix: Theory

B.1 Gated Hyena Computes Separable Attention

We show that a gated HyenaND layer computes exactly the class of *separable attention matrices*—attention patterns that factor as a Toeplitz matrix sandwiched between two diagonal matrices. This gives a precise characterization of the inductive bias of gated Hyena in terms of structured linear algebra, connecting it to the literature on low-displacement-rank operators.

We work with a single-channel, 1D formulation; the multi-channel and multi-dimensional cases follow by applying the result independently per channel.

B.1.1 Formal Setup

Definition B.1 (SIREN kernel)**.** An L-layer SIREN kernel with hidden width m, output dimension N, first-layer frequency $\omega_0 > 0$, and hidden frequency $\omega_h > 0$ is a function $K_\theta : \mathbb{R} \to \mathbb{R}^d$ computed as

$$\begin{aligned} h_0(\tau) &= \sin(\omega_0 W_0 \tau + b_0), \\ h_\ell(\tau) &= \sin(\omega_h W_\ell \, h_{\ell-1}(\tau)), \quad \ell = 1, \dots, L-2, \\ K_\theta(\tau) &= W_{\text{out}} \, h_{L-2}(\tau), \end{aligned}$$

with parameters $W_0 \in \mathbb{R}^{m \times 1}$, $b_0 \in \mathbb{R}^m$, $W_\ell \in \mathbb{R}^{m \times m}$, and $W_{\text{out}} \in \mathbb{R}^{d \times m}$. The input τ is a normalized displacement $(i-j)/n \in [-1, 1]$.

Definition B.2 (Hyena mixer)**.** Given an input sequence $x_1, \dots, x_L \in \mathbb{R}^d$ and a SIREN kernel K_θ, the Hyena mixer with shortcut vector $s \in \mathbb{R}^d$ produces outputs

$$y_i^{(L)} = \sum_{j=1}^{L} K_\theta\!\left(\frac{i-j}{L}\right) x_j + s \odot x_i, \qquad i = 1, \dots, L.$$

Normalizing by L maps all displacements into $[-1, 1]$ regardless of sequence length.

Definition B.3 (Gated Hyena mixer)**.** A gated Hyena mixer extends Definition B.2 with projection matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ and a pointwise nonlinearity σ. The convolution input is gated element-wise,

$$z_j = (W_Q x_j) \odot \sigma(W_K x_j),$$

and an output gate is applied after the convolution:

$$y_i = \left(\sum_{j=1}^{L} K_\theta\!\left(\frac{i-j}{L}\right) z_j \right) \odot \sigma(W_V x_i).$$

B.1.2 Separable Attention Matrices

Definition B.4 (Separable attention matrix)**.** An $L \times L$ matrix $A \in \mathbb{R}^{L \times L}$ is *separable* if

$$A = D_l \, T \, D_R,$$

where $T \in \mathbb{R}^{L \times L}$ is Toeplitz ($T_{ij} = t_{i-j}$ for some sequence t) and $D_l = \mathrm{diag}(r_1, \ldots, r_n)$, $D_R = \mathrm{diag}(c_1, \ldots, c_L)$ are diagonal. Equivalently, $A_{ij} = T_{i-j} \cdot r_i \cdot c_j$.

B.1.3 Main Result

Theorem B.5 (Gated Hyena computes separable attention)**.** *Fix a channel $h \in [d]$ and let $\sigma : \mathbb{R} \to \mathbb{R}$ be any pointwise nonlinearity. The effective attention weight that a gated Hyena mixer places from token j to position i at channel h is*

$$A_{ij}^{(h)} = \underbrace{K_\theta^{(h)}\!\left(\frac{i-j}{n}\right)}_{T_{i-j}} \cdot \underbrace{\sigma(W_V x_i)^{(h)}}_{r_i} \cdot \underbrace{(W_Q x_j)^{(h)} \cdot \sigma(W_K x_j)^{(h)}}_{c_j}.$$

That is, $A^{(h)}$ is a separable attention matrix (Definition B.4) with Toeplitz component $T_{i-j} = K_\theta^{(h)}((i-j)/L)$, row weights $r_i = \sigma(W_V x_i)^{(h)}$, and column weights $c_j = (W_Q x_j)^{(h)} \cdot \sigma(W_K x_j)^{(h)}$. Separability holds for any pointwise σ; no linearity assumption is required.

Proof. Expanding the gated Hyena mixer output (Definition B.3) at channel h, the Hadamard products reduce to scalar multiplication:

$$y_i^{(h)} = \sigma(W_V x_i)^{(h)} \cdot \sum_{j=1}^{n} K_\theta^{(h)}\!\left(\frac{i-j}{n}\right) (W_Q x_j)^{(h)} \cdot \sigma(W_K x_j)^{(h)}.$$

Set $r_i = \sigma(W_V x_i)^{(h)}$, $c_j = (W_Q x_j)^{(h)} \cdot \sigma(W_K x_j)^{(h)}$, and $T_{i-j} = K_\theta^{(h)}((i-j)/L)$. Then $y_i^{(h)} = r_i \sum_j T_{i-j} \, c_j$, so $A_{ij}^{(h)} = T_{i-j} \cdot r_i \cdot c_j$. Since σ is applied point-wise, r_i depends only on x_i and c_j depends only on x_j, so the dependencies are multiplicatively separable. The matrix T is Toeplitz because T_{i-j} depends only on the displacement $i - j$. Hence $A^{(h)} = D_l \, T \, D_R$ is separable by Definition B.4. □

B.2 Length Generalization of HyenaND

The Hyena mixer normalizes the displacement between positions i and j by the sequence length L, mapping all offsets to $[-1, 1]$ regardless of L (Definition B.2). This makes the same SIREN network meaningful across sequence lengths. The theorem below quantifies how much the output changes when a model is evaluated at a length L different from its training length L_0.

B.2.1 Lipschitz Constant of a SIREN

Lemma B.6 (SIREN Lipschitz bound)**.** *For any SIREN kernel K_θ (Definition B.1) and any $\tau, \tau' \in \mathbb{R}$,*

$$\|K_\theta(\tau) - K_\theta(\tau')\|_2 \leq \mathcal{L}_K \cdot |\tau - \tau'|,$$

where $\mathcal{L}_K = \omega_0 \, \omega_h^{L-2} \, \|W_{\mathrm{out}}\|_2 \prod_{\ell=0}^{L-2} \|W_\ell\|_2$.

Proof. We bound the operator norm of the Jacobian $dK_\theta/d\tau$ and apply the mean value theorem. By the chain rule:

$$\frac{dK_\theta}{d\tau} = W_{\mathrm{out}} \left[\prod_{\ell=L-2}^{1} \omega_h \, \mathrm{diag}\big(\cos(\omega_h W_\ell h_{\ell-1}(\tau))\big) \, W_\ell \right] \omega_0 \, \mathrm{diag}\big(\cos(\omega_0 W_0 \tau + b_0)\big) \, W_0.$$

Since $|\cos(\cdot)| \le 1$ point-wise, each diagonal factor has operator norm at most 1. Sub-multiplicativity of the operator norm then gives

$$\left\| \tfrac{dK_\theta}{d\tau} \right\|_2 \le \|W_{\text{out}}\|_2 \prod_{\ell=1}^{L-2} \left(\omega_h \|W_\ell\|_2\right) \cdot \omega_0 \|W_0\|_2 = \mathcal{L}_K,$$

and the mean value theorem yields $\|K_\theta(\tau) - K_\theta(\tau')\|_2 \le L_K |\tau - \tau'|$. □

B.2.2 Main Bound

Theorem B.7 (Length generalization)**.** *Let $x_1, \dots, x_L$ be an input sequence with $\|x_j\|_2 \le B$ for all j, and let H be a Hyena mixer (Definition B.2) with SIREN Lipschitz constant $\mathcal{L}_K$. Evaluating H with length normalizations L and L_0 respectively, the outputs at the last position satisfy*

$$\left\| y_L^{(nL)} - y_n^{(L_0)} \right\|_2 \le \frac{\mathcal{L}_K \, B \, (L-1) \, |L - L_0|}{2 \, L_0}. \tag{6}$$

In particular, for $|\Delta L| = |L - L_0| \ll L_0$:

$$\left\| y_L^{(L)} - y_L^{(L_0)} \right\|_2 \le \frac{\mathcal{L}_K \, B \, L \, |\Delta L|}{2 \, L_0} + O\left(\frac{\Delta L^2}{L_0^2}\right).$$

Proof. Since both evaluations share the same input sequence $x_1, \dots, x_L$, the shortcut terms $s \odot x_L$ are identical and cancel in the difference:

$$\begin{aligned} y_n^{(n)} - y_n^{(n_0)} &= \left(\sum_{j=1}^{L} K_\theta\!\left(\frac{L-j}{n}\right) x_j + s \odot x_n \right) - \left(\sum_{j=1}^{L} K_\theta\!\left(\frac{L-j}{n_0}\right) x_j + s \odot x_n \right) \\ &= \sum_{j=1}^{L} \left[K_\theta\!\left(\frac{L-j}{L}\right) - K_\theta\!\left(\frac{L-j}{L_0}\right) \right] x_j. \end{aligned}$$

The triangle inequality and Lemma B.6 give

$$\left\| y_n^{(L)} - y_L^{(L_0)} \right\|_2 \le \mathcal{L}_K \, B \sum_{j=1}^{L} (L-j) \left| \frac{1}{L} - \frac{1}{L_0} \right| = \mathcal{L}_K \, B \left| \frac{1}{L} - \frac{1}{L_0} \right| \cdot \frac{L(L-1)}{2},$$

using $|(L-j)/L - (L-j)/L_0| = (L-j)|1/L - 1/L_0|$ and $\sum_{j=1}^{n}(L-j) = L(L-1)/2$. Substituting $|1/L - 1/L_0| = |L - L_0|/(L\,L_0)$ yields (6). □

Remark B.8. The bound grows linearly in L_K, which increases with the first-layer frequency ω_0. High-frequency SIREN kernels are therefore less stable under length shifts than smoother, lower-frequency ones. This is consistent with the masking and initialization strategy of §B.3, which controls the spectral behavior of the kernel at initialization.

Remark B.9 (Practical scope of the bound)**.** When $L > L_0$, the displacement $(L-j)/L_0$ exceeds 1 for tokens near $j = 1$, placing SIREN queries outside the $[-1, 1]$ domain on which the kernel was trained. The bound in (6) remains mathematically valid, Lemma B.6 holds for all $\tau \in \mathbb{R}$, but its practical interpretation relies on the kernel behaving predictably outside the training domain. The periodicity of the sine activations provides some continuity beyond $[-1, 1]$, yet does not eliminate the concern. Theorem B.7 is therefore most meaningful in the regime $|L - L_0| \ll L_0$, where the displacements remain close to the trained range.

B.3 HyenaND Design Details: Masking and Initialization

Naively applying the 1D Hyena masking and SIREN frequency schedule to $N{>}1$ data introduces two failure modes: the effective receptive field collapses near the origin as N grows, and the spectrum of the kernel-MLP's embedding layer shrinks with N, so a frequency schedule tuned on one grid does not transfer to another. We detail the three design choices that resolve both: a learnable Gaussian mask for spatial data, a dimensionality-aware $\boldsymbol{\omega_0}$ schedule for the SIREN first layer, and zero-start initialization for the FiLM conditioning heads.

Masking strategy. For spatial tasks ($N \geq 2$), the primary setting of this work, we use a learnable *Gaussian mask* akin to Romero et al. [35], but where channel c learns its own per-axis standard deviation $\sigma_{d,c}>0$:

$$M_c(\mathbf{r}) = \prod_{d=1}^{N} \exp\left(-\frac{r_d^2}{2\sigma_{d,c}^2}\right) = \exp\left(-\tfrac{1}{2}\sum_d \left(\tfrac{r_d}{\sigma_{d,c}}\right)^2\right). \tag{7}$$

When all axes share the same σ_c this is a radially symmetric Gaussian; anisotropic grids (e.g. a volume with $N_z \ll H, W$) are handled through per-axis initialization of the $\sigma_{d,c}$. The bandwidths are initialized on a log-spaced ramp across channels, so at the start of training the operator spans from narrow, local filters to near-global ones; all $\sigma_{d,c}$ are learned during training. Grid boundaries are handled by zero-padding each spatial dimension by $N_d - 1$ before the FFT and cropping the output, following standard linear-convolution practice.

The Gaussian mask has two structural advantages over the exponential in the multi-dimensional setting. First, the decay rate is governed by Euclidean distance from the origin when $\sigma_{d,c}$ is isotropic, so the effective support scales with σ_c independent of N. The naive axis-product exponential $M(\mathbf{r}) = \prod_d e^{-\alpha_d |r_d|}$, by contrast, has magnitude $e^{-\alpha N}$ at unit distance from the origin, collapsing the receptive field by a factor of N. Second, $\sigma_{d,c}$ is a directly interpretable bandwidth parameter per channel and axis, making the effective receptive field explicitly learnable.

For causal 1D data (e.g. autoregressive genomic sequence modeling), we instead retain the original Hyena exponential mask $K(L) \leftarrow K(L) \cdot e^{-\alpha|L|}$, additionally enforcing causality by zeroing negative offsets: $M(r) \leftarrow M(r) \cdot \mathbf{1}[r \geq 0]$. The exponential form is the natural choice here: it provides a monotone decay matched to the intuition of recency bias in causal sequence modeling.

Kernel-MLP (SIREN) initialization. The SIREN initialization scheme [37] calibrates weight variances so that the pre-activations at every hidden layer have unit variance at initialization, which is essential for gradient propagation through sinusoidal activations.

Review of the 1D *case.* For a scalar coordinate $r \in [-1, 1]$ and first-layer weight matrix $W_0 \in \mathbb{R}^{d_{\text{emb}} \times 1}$ with entries drawn i.i.d. from $\mathcal{U}(-a, a)$, the pre-activation is $w_i r$ for each row w_i. Since $\mathbb{E}[w_i^2] = a^2/3$ and $\mathbb{E}[r^2] = 1/3$ for $r \sim \mathcal{U}(-1, 1)$,

$$\mathrm{Var}(w_i r) = \mathbb{E}[w_i^2]\,\mathbb{E}[r^2] = \frac{a^2}{9}. \tag{8}$$

SIREN's fan-in rule $a = 1/N$ gives $a = 1$ here (i.e., $\mathcal{U}(-1, 1)$), so $\mathrm{Var}(w_i r) = 1/9$, consistent with well-behaved activations for 1D coordinate inputs.

Extension to N dimensions. Our embedding layer maps a N-dimensional coordinate $\mathbf{r} \in [-1, 1]^N$ to $\mathbb{R}^{d_{\text{emb}}}$ via

$$\phi = \sin(\omega_0 W_0 \mathbf{r}), \qquad W_0 \in \mathbb{R}^{d_{\text{emb}} \times N}. \tag{9}$$

For row $w_0^\top \in \mathbb{R}^N$ of W_0 with entries drawn i.i.d. from $\mathcal{U}(-a, a)$ and coordinate components $r_d \sim \mathcal{U}(-1, 1)$ independently,

$$\mathrm{Var}(w_0^\top \mathbf{r}) = \sum_{d=1}^{N} \mathrm{Var}(w_{0,d}\, r_d) = N \cdot \frac{a^2}{9}. \tag{10}$$

We keep SIREN's fan-in rule $a = 1/N$, i.e. $W_0 \sim \mathcal{U}(-1/N, 1/N)$ as in §3.2.1 (where this matrix is written $\mathbf{W}_1$), which gives

$$\mathrm{Var}(w_0^\top \mathbf{r}) = \frac{1}{9N}. \tag{11}$$

Relative to the 1D case, the standard deviation of the first-layer pre-activations, and with it the spectrum of the embedding, shrinks by a factor of $\sqrt{N}$. Rather than changing the bound, we absorb this into the frequency schedule. In normalized coordinates a grid with L points per axis has spacing $\approx 2/L$ and Nyquist frequency $\approx \pi L/2$, so the fraction of the band that a base frequency ω_0 covers scales as ω_0/L in the grid size and as $1/\sqrt{N}$ in the dimensionality. Preserving this fractional coverage when a kernel is initialized for a new resolution L' or dimensionality N' therefore requires $\boldsymbol{\omega_0'} = \boldsymbol{\omega_0}\,(L'/L)\sqrt{N'/N}$, the schedule of §3.2.1. The subsequent hidden layers receive a d_{emb}-dimensional input and follow the standard SIREN prescription unchanged.

FiLM parameter initialization. At initialization, the modulated kernel should equal the base kernel so that training begins from the same well-characterized regime as the unmodulated model. We achieve this by initializing the final linear layers of γ_θ and β_θ to zero with a constant identity residual, giving $\gamma_\theta(c)\big|_{t=0} \equiv \mathbf{1}$ and $\beta_\theta(c)\big|_{t=0} \equiv \mathbf{0}$ for all c. This ensures $K(\mathbf{r};\, \mathbf{x})\big|_{t=0} = K_0(\mathbf{r})$ for every input, so the FiLM heads start from the same fixed point as the unmodulated baseline rather than a random perturbation.

C CUDA Implementation and Benchmarking

Although our developmental framework is broadly applicable, we focus on Evo2 and 2D ImageNet classification to demonstrate our contributions to the Hyena architecture and other machine learning models relying on similar primitive operations. All our custom kernels are implemented using CuTe and cuFFTDx [30, 29], and the benchmarking is performed on an NVIDIA RTX PRO 6000 Blackwell Server Edition. Prior work, such as "Systems and Algorithms for Convolutional Multi-Hybrid Language Models at Scale," introduced similar operators based on hardware-algorithm co-design principles to maximize hardware utilization [16]. Building upon this foundation, we introduce novel kernels for the 1D operators utilized in Evo2, and extend them to 2D cases, closing the gap between the theoretical performance of Hyena Layers and their real runtime.

C.1 Hardware Context and IO Profile

To fully contextualize our methodology, it is essential to consider the strict memory hierarchy of modern GPUs. This hierarchy spans from massive but relatively slow global High Bandwidth Memory (HBM), up through progressively smaller and faster layers—L2 cache, L1 cache/Shared Memory (SRAM)—and finally, ultra-fast thread-level registers. Standard PyTorch implementations are frequently bottlenecked by this hierarchy, forced to materialize intermediate tensors and perform costly HBM read/write round-trips between sequential operations. Our approach is therefore primarily oriented toward minimizing HBM accesses through kernel fusion, keeping intermediate computations localized within SRAM and registers.

Two hardware trends shape the practical performance comparison between attention and FFT-based operators. First, GPU matrix-multiply throughput has generally grown faster than off-chip memory bandwidth [24–27]. This strongly benefits standard attention, because the dominant operations in SDPA, $QK^\top$ and PV, are GEMM-like. Modern fused SDPA kernels further improve efficiency by tiling the computation and avoiding materialization of the full $N \times N$ attention matrix in HBM. FlashAttention-style kernels, therefore, make quadratic attention far more competitive in practice than its asymptotic complexity alone would suggest [4, 49].

FFT-based Hyena operators face a different performance profile. Their fast path depends less directly on tensor-core GEMM throughput and more on batched FFT efficiency, memory layout, fusion, and bandwidth. Although there have been attempts to enable the usage of tensor-cores for FFT, they usually lag behind the routines provided by cuFFT and cuFFTDx, especially for real FFT [9, 28, 29]. FFTs introduce nontrivial constant factors, often require complex-valued intermediate representations, and may involve transposes or layout conversions that increase HBM traffic. As a result, an $\mathcal{O}(L \log L)$ sequence mixer may only outperform fused quadratic attention beyond a hardware- and implementation-dependent sequence length.

A second relevant hardware trend is the increasing support for explicit cooperation across thread blocks. Hopper-class NVIDIA GPUs [26] introduced thread-block clusters and distributed shared memory, allowing CTAs in the same cluster to be co-scheduled, synchronize at cluster scope, and access one another's shared-memory segments. Blackwell [27] separately introduces Cluster Launch Control, which provides more dynamic control over CTA work assignment through cluster-level cancellation and work-stealing mechanisms. These features should not be conflated: distributed shared memory is a Hopper-era cluster cooperation mechanism, while Cluster Launch Control is a Blackwell-era scheduling mechanism.

For dense attention, these cluster-level mechanisms are less central than the highly optimized GEMM and fused-SDPA paths. For FFT-based subquadratic sequence models, they may become more relevant because the dominant bottleneck is often not arithmetic, but HBM traffic induced by padding, complex-valued intermediate tensors, precision conversion, and kernel-boundary materialization.

In particular, FlashAttention-style SDPA keeps the $L \times L$ score matrix off HBM and exposes bf16 input/output interfaces to the surrounding model. A naive FFT-convolution path has a less favorable IO profile. Even when the model activations are stored in bf16, practical library FFT paths often require either fp32/complex-fp32 intermediates or constrained bf16 transform layouts; in the unrestricted layouts used by HyenaND, this precision widening alone can double the traffic per real-valued element relative to bf16 attention activations.

The effect is amplified by the real-to-complex FFT layout used for the long convolution. A 1D sequence of logical length l requires padding to a transform length $l \geq 2L - 1$; in practice we use $l \approx 2L$. The forward real-to-complex transform therefore materializes

$$\widehat{x} = \mathcal{F}_l(x) \in \mathbb{C}^{l/2+1} \approx \mathbb{C}^{L+1}, \tag{12}$$

followed by complex multiplication with the spectral filter,

$$\widehat{y} = \widehat{x} \odot \widehat{h}. \tag{13}$$

The inverse complex-to-real transform then produces an $l \approx 2L$ real-valued buffer, from which the valid L outputs must be cropped or chunked. A naive implementation writes each of these objects to HBM: the padded real input, the activation spectrum, the filter spectrum, the inverse-FFT buffer, and the final segmented output. The nominal $\mathcal{O}(L \log L)$ arithmetic advantage can therefore be hidden by a large constant factor in memory traffic and by multiple small kernel launches around the FFT calls.

The multi-dimensional case worsens this IO pressure. For a 2D input of size $H \times W$, linear convolution pads each axis to $L_H \approx 2H$ and $L_W \approx 2W$, and the real-to-complex spectrum has shape

$$\widehat{x} \in \mathbb{C}^{L_H \times (L_W/2+1)} \approx \mathbb{C}^{2H \times (W+1)}. \tag{14}$$

The padded real-space footprint grows by roughly $4\times$, and the complex spectrum carries another factor from storing real and imaginary components. In D dimensions, padding alone contributes an approximate 2^D expansion before accounting for complex storage, temporary workspaces, or layout conversions. This is the practical source of the HBM explosion in naive FFT-based Hyena implementations, especially for 2D images and 3D volumes. In this setting, Hopper and Blackwell cluster-cooperation features are useful not because they accelerate dense GEMM, but because they provide additional mechanisms for staging non-GEMM work and coordinating CTAs that cooperatively perform FFT.

C.2 Filter Generation

Through Nsight Systems (`nsys`) profiling, we observed that 30% of the computational cost in the LI layer stems from filter generation. To remedy this, we developed a dedicated kernel specifically for this operation.[2] As illustrated in Figure 6, this dedicated kernel accelerates implicit filter runtime by over $40\times$ at the module level compared to a vanilla PyTorch implementation. Concurrently, it reduces memory overhead by a factor proportional to the order (16) of the implicit modal filter ($32\times$), thereby freeing up crucial memory to accommodate larger batch sizes and scale to significantly longer context lengths. Additionally, compared to the first release of Evo2, we shard the parameters of implicit modal filter instead of generated filter, saving $CP\times$ memory for context extension. Through both optimizations, we achieve $16M$ context length for $1B$ Evo2 Model.

C.3 1D FFT Convolution

For the LI layer of Evo2, and long implicit layers in general, we developed a highly optimized, fused causal FFTConv1D kernel. For sequence lengths up to 16K, this single kernel fuses the Real Fast Fourier Transform (RFFT) of the input and filter, element-wise complex multiplication, the Inverse RFFT (IRFFT), and chunking. This extensive fusion yields a $6\times$ runtime speedup and a $4\times$ reduction in memory consumption compared to standard implementations. For longer sequence lengths, we employ a three-kernel approach based on Cooley-Tukey (CT) decomposition, effectively casting the 1D FFT into a 2D FFT. In this regime, we fuse the Complex-to-Complex (C2C) FFT, the Inverse C2C (IC2C) FFT, and the complex multiplication. The performance improvements and scaling behavior of this architecture are detailed in Figure 7. While the three-stage CT approach achieves $2 - 4\times$

[2] Source code and documentation omitted for double-blind review.

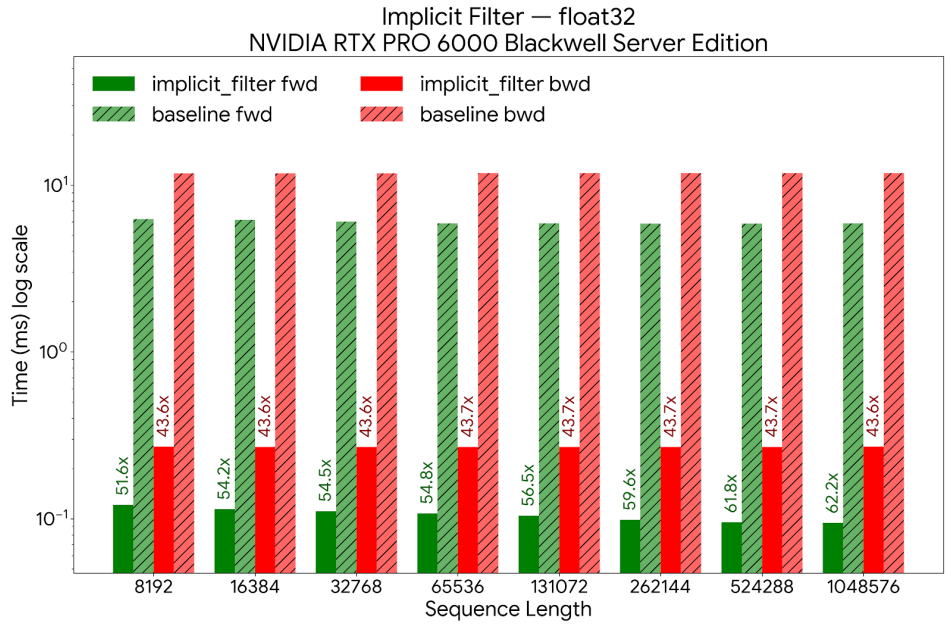


Figure 6: Performance comparison of the implicit filter generation. Profiling results demonstrate that our dedicated, fused kernel accelerates filter generation runtime by over $40\times$ compared to the baseline PyTorch implementation. Furthermore, the kernel reduces High Bandwidth Memory (HBM) overhead by a factor of $32\times$, a reduction proportional to the order of the modal filter.

speedup and memory reduction compared to native baselines, it never approaches the performance of fully-fused single kernel.

Fortunately, a sequence length of 16K marks the critical point where Hyena's asymptotic scaling dominates quadratic SDPA. However, to extend the superior performance of the fully-fused single kernel to longer sequences, our future work will leverage distributed shared memory (DSM). Specifically, the DSM capabilities of the Hopper and Blackwell architectures provide vital staging mechanisms for this non-GEMM workload, enabling us to circumvent the massive memory traffic and multiple HBM round-trips inherent to naive FFT-based convolutions. This will fix the speedup drops for the sequence lengths of $[16, 64]$K. Note: the drop in speedup is hardware dependent.

We also developed `causal-conv1d` kernels that support filter sizes up to 256 or 128, depending on whether the input layout is channel-first or channel-last, respectively. As shown in Figure 8, these kernels provide a substantially more efficient implementation than native PyTorch code and support a wider range of filter lengths compared to the Dao-AILab `causal-conv1d` implementation.

For the SE and MR layers, both of which utilize projection and mixer convolutions with filter lengths of ≤ 128, we developed a highly fused `b2b causal-conv1d` kernel. This custom kernel seamlessly integrates the projection convolution, pre-gating, mixer convolution, and post-gating steps into a single operation. As shown in Figure 9, this fusion improves runtime by more than $7.5\times$ and reduces the overall memory footprint and HBM traffic. Additionally, in context-parallel setups, this fused approach effectively reduces the number of required communication points by half, substantially mitigating communication overhead across devices.

C.4 2D FFT Convolution

Beyond our optimizations for 1D sequence modeling, we extend our hardware-algorithm co-design framework to 2D spatial data to accelerate vision-based architectures, specifically targeting ImageNet classification workloads. To this end, we developed highly optimized 2D FFT-based convolution kernels (`fft-conv2d`) to efficiently handle large-scale spatial filters.

In standard implementations, computing a 2D FFT convolution requires materializing multiple intermediate frequency-domain tensors, resulting in severe HBM bottlenecks. To circumvent this, we

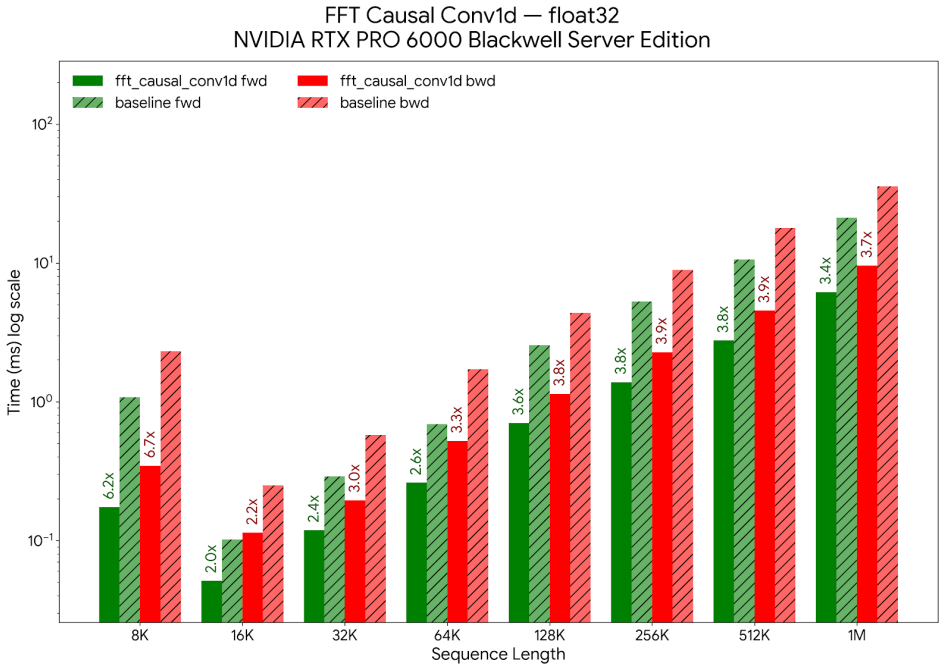


Figure 7: Performance evaluation of the three-kernel causal FFTConv1D approach based on Cooley-Tukey (CT) decomposition. The plot illustrates the scaling behavior and efficiency gains for sequence lengths exceeding 8K. Except for 8K sequence length that uses fully-fused kernels, the rest of points use three different stages.

designed a pipeline that executes an initial Real-to-Complex FFT (RFFT) along the first dimension, followed by a fused core kernel for Complex-to-Complex (C2C) transformations, and concludes with an Inverse Real-to-Complex FFT (IRFFT). Our custom core kernel fuses the C2C FFT along the second dimension, the element-wise complex multiplication with the filter, and the Inverse C2C FFT.

Our benchmarking focuses specifically on forward-pass execution. As demonstrated in Figure 10, comparing this approach against an unfused baseline reveals that our memory-aware fusion delivers an over $5\times$ increase in runtime throughput and an over $2\times$ reduction in overall memory footprint. Similar to the 1D FFT case, our plan is to extend this workflow to use a single-stage kernel for the RFFT, C2C FFT, complex multiplication, and the Inverse C2C FFT and IRFFT.

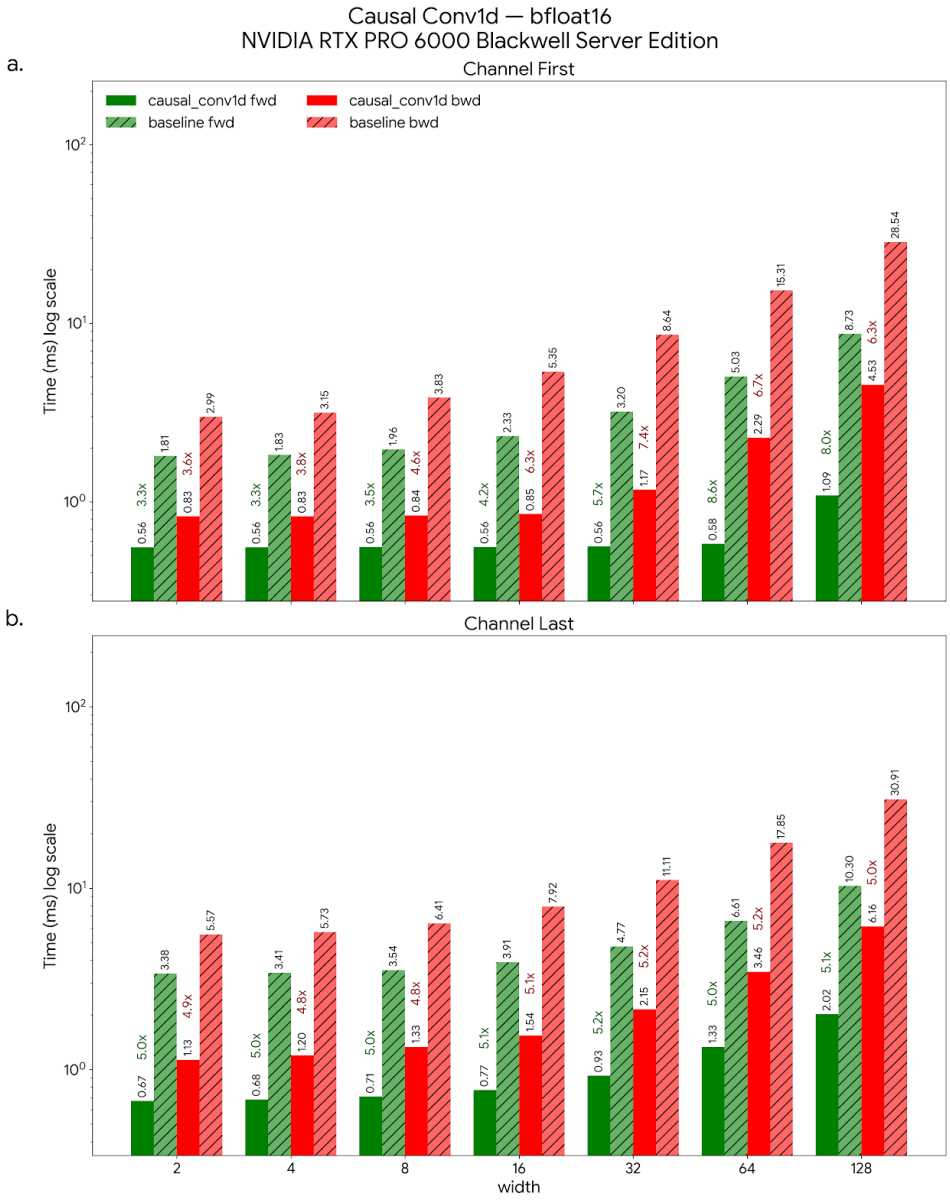


Figure 8: Performance comparison of our custom `causal-conv1d` kernels against native PyTorch implementations. Our kernels deliver substantial efficiency gains while extending support for filter sizes up to 256 (for channel-first layouts) and 128 (for channel-last layouts), surpassing the filter length constraints of existing optimized packages.

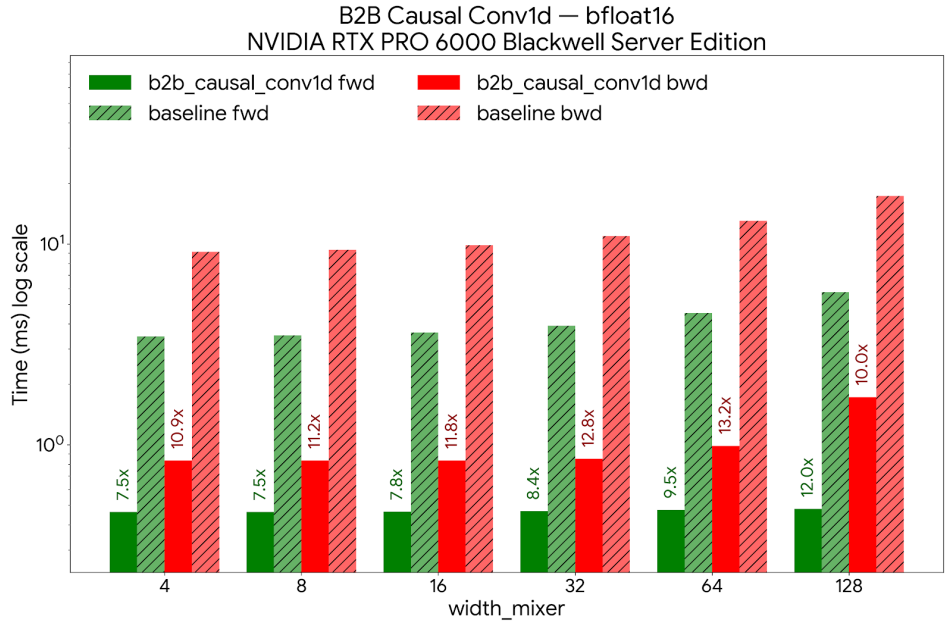


Figure 9: Performance evaluation of the fused `b2b causal-conv1d` kernel for the SE and MR layers. By integrating the projection convolution, pre-gating, mixer convolution, and post-gating into a single operation, the fused kernel achieves more than a $7.5\times$ runtime speedup and lowers the memory footprint. It also halves the necessary communication points in context-parallel configurations for the hyena operator.

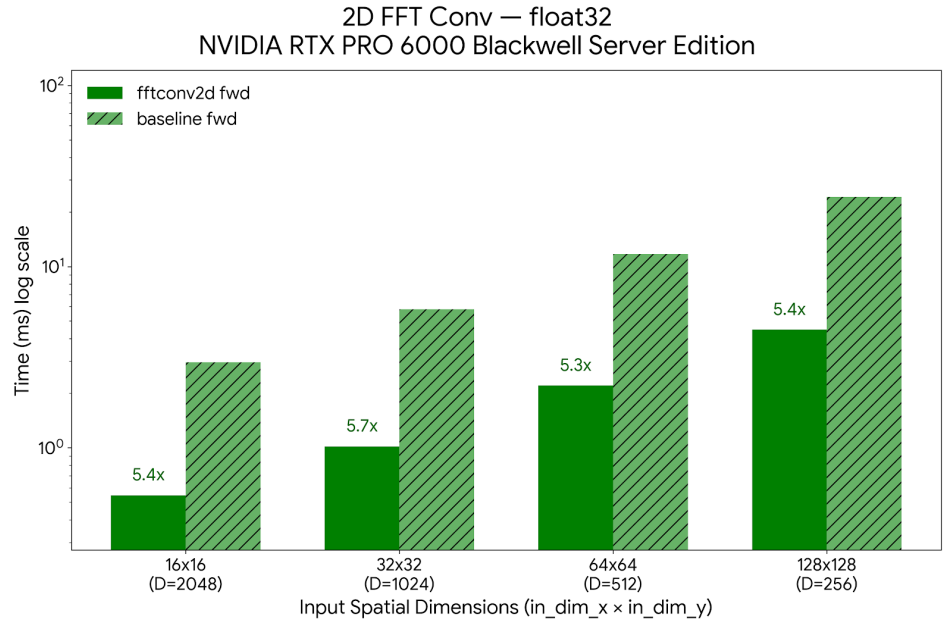


Figure 10: Forward-pass performance evaluation of our 2D FFT convolution kernels for image classification. We compare our approach against an unfused baseline. By successfully fusing the C2C FFT, complex multiplication, and IC2C FFT into a single core operation, intermediate computations are kept within fast Shared Memory and registers, achieving over $5\times$ runtime speedup and over $2\times$ reduction in memory footprint.

D Appendix: Experiments

D.1 Controlled Spatial-Recall Experiments

We introduce a family of controlled *spatial-recall* tasks that isolate the two primitive abilities any spatial token mixer needs: (i) *transport* — moving content unchanged across a long canvas (`simple_copy`) — and (ii) *content-keyed retrieval* — selecting the right item among distractors via an associative key and routing it to the readout (`color_cond`). The tasks span 1D sequences, 2D images, and, in the hardest setting, 3D spatio-temporal volumes in which the target itself moves, so recall must additionally capture continuous latent factors (trajectory, spin) rather than a static pattern; a patch-size sweep in 1D/2D further probes how each operator trades token count against per-token detail. Together the probes read off *operator inductive bias* under controlled conditions: HyenaND's global convolution encodes translation equivariance architecturally; attention must recover positional structure from its encodings (RoPE in our setup) and, as retrieval over per-token summaries, depends critically on the information each token carries; and scan-based SSMs must rasterize the canvas into a 1D order — even bidirectional variants impose an arbitrary serialization of space. Parameter count and training budget are fixed across operators, and sensitivity to the optimization regime is measured explicitly (the learning-rate $\times$ grad-clip factorial below), so remaining gaps are attributable to the operator itself rather than to capacity or tuning.

Task family and dataset construction. All canvases are built from EMNIST digits on an empty background; the model reads the full canvas and regresses the target content at a *designated readout region* in a fixed corner (MSE is computed on the readout region only). `simple_copy` places a single item and isolates *long-range positional recall*; `color_cond` places four items with distinct color markers and isolates *spatial binding with distractors*. Construction per dimensionality:

- **1D (L=4,096).** A 16×16 digit is flattened to a 256-token segment and placed at a uniformly random offset (the readout span excluded). In `color_cond`, four such segments are placed without overlap, each flanked by 2-token markers in a distinct color drawn without replacement from a fixed 8-color palette; the readout — the last 256 tokens — is flanked by markers in the *target's* color. Labels are the target segment, recolored for `color_cond`. For `simple_copy` we report both a causal (autoregressive) and a non-causal variant.
- **2D (64^2).** 16×16 digits are placed at uniformly random, non-overlapping positions anywhere outside the bottom-right quadrant, which is reserved for the readout. In `simple_copy`, a single digit is placed and must be reproduced at the fixed 16×16 readout square in the bottom-right corner. In `color_cond`, four digits are placed, each wrapped in a one-pixel outline in a distinct color from the same 8-color palette. The readout square is outlined in the color of one of them, and the model must identify the digit whose outline matches and reproduce it recolored to that color. Labels are the 16×16 digit – grayscale for `simple_copy`, recolored for `color_cond`.
- **3D motion (32^3, depth = time).** The recall target is a *spatio-temporal tube*: a 4×4 digit is stamped on every depth slice of a 6^3 block while translating along a random continuous path and turning in-plane in crisp 90° steps. Movement paths are designed such that the motion range is bounded by the digit's ink rather than its stamp box and follow a monotone full-span sweep with a per-slice step of at most 2 voxels to ensure the digit is present in every slice. The tube interior is filled with the digit's own background value, leaving no visible stamp-box boundary. In `simple_copy`, a single tube is placed at a fixed position and must be reproduced at the 6^3 readout corner. In `color_cond`, four tubes are each wrapped in a one-voxel colored ring (in-plane on every slice, depth faces open; 8^3 framed) and scattered at random non-overlapping positions. The 8^3 readout corner carries a ring in the target's color, and the label is the target's framed block with its content recolored.

Dataset-mean predictors give the random performance floors reported in Table 1: 0.549/0.284 (`simple_copy`/`color_cond`) in 1D/2D, and 0.482/0.257 on the 3D motion volumes. Figure 11 shows one exemplar per task and dimensionality; Figure 12 shows full slice sequences for the motion tasks.

Models, capacity, and training. We compare three token mixers at parameter parity ($\sim$ 1.9M, 4 residual blocks): (i) multi-dimensional Hyena (hidden dim 256), (ii) multi-head self-attention with

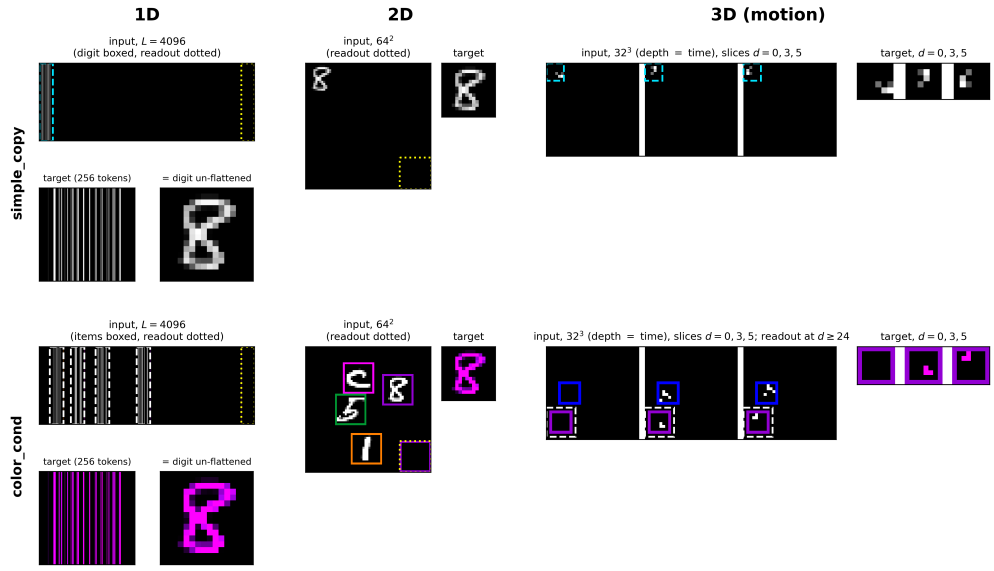


Figure 11: Spatial-recall tasks across dimensionalities (real training samples). Every example is a mostlyempty canvas – a 1D sequence (L=4096), a 2D image (64^2), or a 3D volume (32^3, where depth describes time) – containing one or more small EMNIST digits plus a designated readout region (dotted). The model processes the whole canvas and is trained with MSE to reproduce the correct content at the readout region. *Top* (`simple_copy`): a single digit (boxed) must be *transported*, unchanged, to the readout — pure long-range routing, with no ambiguity about what to copy; in 3D the digit additionally drifts and spins through a 6^3 spatio-temporal tube (time slices d=$0, 3, 5$ shown for both input and target). *Bottom* (`color_cond`): four items carry distinct color markers; the readout displays one of those colors, and the model must select the item whose marker matches and reproduce it recolored: associative recall (retrieve-by-key) laid out in space, with color as the key and content as the value. The two tasks thus separate routing capacity from input-dependent selection.

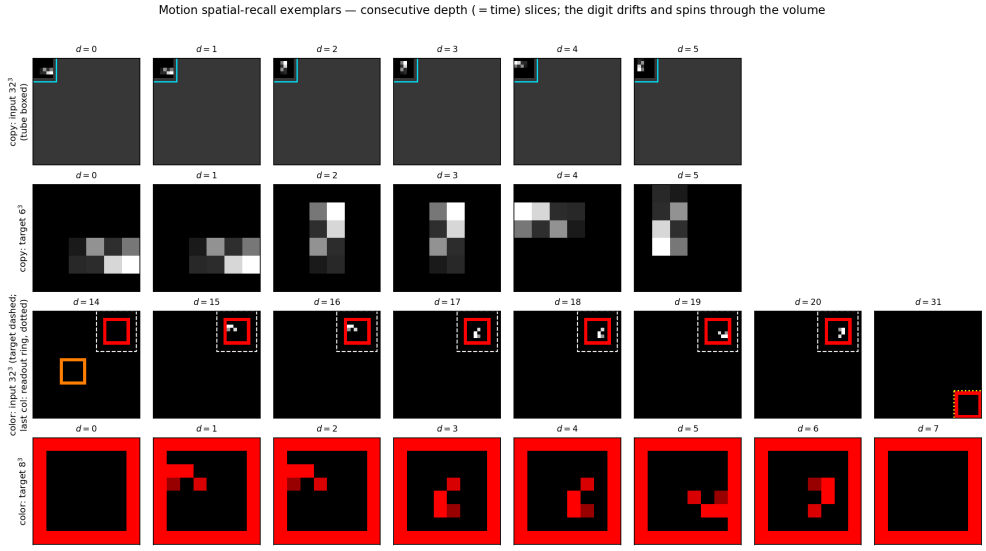


Figure 12: Motion dataset construction (consecutive depth (=time) slices from real training samples): *Rows 1–2* (`simple_copy`): input canvas with the 6^3 tube (cyan box) and the corresponding target tube: the digit drifts along its sweep path (such that they never leave the tube) and turns in 90° steps. *Rows 3–4* (`color_cond`): the target tube's ring (dashed white) matches the color of the readout ring (last column, d=31, dotted yellow); a distractor's ring (Three distractors in total) is visible at d=14 (orange). The label (row 4) is the target's framed block with its content recolored.

RoPE (256 in 1D/2D; 240 in 3D, where head_dim must be divisible by 6 for axial RoPE), and (iii)

Mamba-2 in both unidirectional (causal, dim 208) and bidirectional (dim 160) variants. All models are trained for 50k iterations with AdamW, cosine schedule and 5% warmup, `bf16`-mixed precision, and RMSNorm. Hyena and Attention run with `torch.compile` (`max-autotune-no-cudagraphs`); Mamba is run eagerly. For 2D experiments, we additionally test a HyenaND variant with a learnable Gaussian mask on the convolutional kernel (replacing the default identity mask).

Patchification as a probe. Because the three operators respond very differently to sequence length, we additionally sweep a `Patchify`/`Unpatchify` in/out projection over patch sizes p, reducing the number of tokens by a factor of p^N while packing the discarded spatial detail into the channel dimension. This exposes the *token-count vs. spatial-detail* trade-off: large p shortens the sequence (helping attention's cost and tightening the effective context) at the cost of per-token spatial resolution.

1D results. Table 5 reports non-patched results on the L=4,096 canvas and the best patch size per model/task; the full patch-size sweep is in 7. Three patterns emerge:

(i) Hyena dominates the long-context regime. On the raw L=4,096 canvas Hyena is 3–4 orders of magnitude below attention on `simple_copy` and attention never leaves the random floor on `color_cond` (0.274 vs. baseline 0.284). *(ii) Attention requires heavy patchification to become competitive* on `simple_copy` (a 1250× improvement from p=1 → p=128) and remains essentially stuck at the random floor on `color_cond` at every patch size. *(iii) Causal Mamba cannot solve* `simple_copy` (test loss 0.739 > random 0.549), but *bidirectional Mamba is strong*: it matches Hyena on non-patched `color_cond` (0.0065 vs. 0.0062) and, once patched, matches or beats Hyena on `simple_copy` and improves `color_cond` to 2.4e−4.

Table 5: **1D spatial recall.** MSE (lower is better) on a $L = 4{,}096$ canvas, 50k iterations, ∼1.9M parameters. "Best p" reports the best patch size per model/task; the non-patched regime exposes long-range behavior. Random baselines: 0.549 (`simple_copy`), 0.284 (`color_cond`).

Task	Causality	Hyena	Attention (RoPE)	Mamba (causal)	Mamba (bidir)
No patchification ($L = 4{,}096$ tokens)					
`simple_copy`	causal	1.32×10^{-5}	0.128	0.739	—
`simple_copy`	non-causal	5.20×10^{-5}	0.150	—	0.0854
`color_cond`	causal	0.0629	0.270	0.205	—
`color_cond`	non-causal	0.0062	0.274	—	0.006 49
Best patch size per model (non-causal)					
`simple_copy`	—	3.34×10^{-5} (p=32)	1.20×10^{-4} (p=128)	—	4.66×10^{-5} (p=64)
`color_cond`	—	5.0×10^{-3} (p=16)	0.256 (p=128)	—	2.38×10^{-4} (p=8)

2D results. Extending the tasks to a genuine 2D canvas with 2D Hyena and axial-RoPE Attention yields the results in Table 6. The qualitative picture mirrors 1D but is sharper: on the full 4,096-token grid, Hyena is ∼400× better than attention on `simple_copy` and ∼100× better on `color_cond`, while attention again saturates at the `color_cond` random floor regardless of patch size. Bidirectional Mamba with p=4 now establishes a new best on `color_cond` (4.06×10^{-4}), beating Hyena's best patched result by 5×; on `simple_copy`, Mamba at p=8 matches Hyena. Adding a Gaussian mask slightly improves Hyena on `simple_copy` no-patch (1.31×10^{-4} vs 2.08×10^{-4}) but offers no advantage on `color_cond` in 2D; the full sweep is in Table 8.

Table 6: **2D spatial recall.** MSE on a 64×64 canvas, 50k iterations, ∼1.9M parameters. "Best p" for Hyena (Gauss) `color_cond`: no-patch is already optimal.

Task	Variant	Hyena	Hyena (Gauss)	Attention (RoPE)	Mamba (bidir)
`simple_copy`	No patch (L=4,096)	2.08×10^{-4}	1.31×10^{-4}	0.0923	0.349
`simple_copy`	Best patched	6.53×10^{-5} (p=8)	1.01×10^{-4} (p=4)	1.96×10^{-4} (p=16)	5.70×10^{-5} (p=8)
`color_cond`	No patch (N=4,096)	2.54×10^{-3}	3.18×10^{-3}	0.253	8.34×10^{-3}
`color_cond`	Best patched	2.09×10^{-3} (p=2)	3.18×10^{-3} (no patch)	0.127 (p=8)	4.06×10^{-4} (p=4)

3D motion results. The 3D suite runs at native $32^3 = 32{,}768$ voxels, where dense self-attention is computationally inadmissible: `torch.compile` attempts a >500 GiB allocation during inductor lowering, so no practical non-patched 3D Attention baseline exists (and patched attention sits at

Table 7: **Full 1D patch-size sweep** (non-causal, 50k iterations, $\sim$1.9M parameters). MSE lower is better; bold = best per task/column. Random baselines: 0.549 (`simple_copy`), 0.284 (`color_cond`).

Task	p (L tokens)	Hyena	Attention (RoPE)	Mamba (bidir)
simple_copy				
	1 (4096)	5.20×10^{-5}	0.150	0.0854
	2 (2048)	1.12×10^{-4}	0.197	0.0153
	4 (1024)	1.20×10^{-4}	0.041	2.57×10^{-4}
	8 (512)	9.56×10^{-5}	9.00×10^{-3}	1.40×10^{-3}
	16 (256)	4.47×10^{-5}	5.38×10^{-3}	1.27×10^{-4}
	32 (128)	3.34×10^{-5}	9.96×10^{-4}	5.27×10^{-5}
	64 (64)	3.42×10^{-5}	1.51×10^{-4}	4.66×10^{-5}
	128 (32)	3.73×10^{-5}	1.20×10^{-4}	7.45×10^{-5}
	256 (16)	5.70×10^{-5}	1.76×10^{-4}	1.86×10^{-3}
color_cond				
	1 (4096)	6.20×10^{-3}	0.274	6.49×10^{-3}
	2 (2048)	0.0445	0.275	0.0121
	4 (1024)	0.0278	0.274	7.88×10^{-3}
	8 (512)	7.80×10^{-3}	0.269	2.38×10^{-4}
	16 (256)	5.00×10^{-3}	0.266	8.51×10^{-4}
	32 (128)	7.90×10^{-3}	0.264	2.48×10^{-3}
	64 (64)	0.0235	0.261	7.57×10^{-3}
	128 (32)	0.142	0.256	0.0348
	256 (16)	0.246	0.256	0.113

Table 8: **Full 2D patch-size sweep** (64×64 canvas, 50k iterations, $\sim$1.9M parameters). MSE lower is better; bold = best per task/column. Token counts: no patch = 4096, p=2 = 1024, p=4 = 256, p=8 = 64, p=16 = 16.

Task	p	Hyena	Hyena (Gauss)	Attention (RoPE)	Mamba (bidir)
`simple_copy`					
	1	2.08×10^{-4}	1.31×10^{-4}	0.0923	0.349
	2	3.27×10^{-4}	2.98×10^{-4}	0.0632	5.25×10^{-3}
	4	1.33×10^{-4}	1.01×10^{-4}	4.30×10^{-3}	2.92×10^{-4}
	8	6.53×10^{-5}	1.03×10^{-4}	2.56×10^{-4}	5.70×10^{-5}
	16	8.52×10^{-5}	1.44×10^{-4}	1.96×10^{-4}	1.87×10^{-3}
`color_cond`					
	1	2.54×10^{-3}	3.18×10^{-3}	0.253	8.34×10^{-3}
	2	2.09×10^{-3}	4.07×10^{-3}	0.271	8.61×10^{-4}
	4	5.61×10^{-3}	7.81×10^{-3}	0.216	4.06×10^{-4}
	8	1.69×10^{-2}	2.27×10^{-2}	0.127	1.06×10^{-2}
	16	6.38×10^{-2}	6.16×10^{-2}	0.131	5.50×10^{-2}

the random floor on every `color` task in 1D–2D). HyenaND and bidirectional Mamba are thus the only operators evaluated at native 3D resolution; Table 1 reports the outcome. On `simple_copy`, HyenaND solves the task (1.2×10^{-4}) while Mamba barely improves on the mean-predictor floor throughout training (0.451; floor 0.482): rasterizing a moving 3D target into a 1D scan breaks positional recall. On `color_cond`, both operators solve the task, and the ordering is governed by the optimization regime rather than by architecture, as the factorial below shows.

Discussion: what the two probes tell us. On raw canvases each token is a single pixel, and the three operator families respond to this regime in characteristically different ways. Attention — a retrieval operation over per-token summaries — collapses: with one pixel per query and key there is nothing to compare, and it recovers only when patchification packs many pixels into each token. This explains its monotone improvement with patch size on `simple_copy`, its failure on `color_cond` at every patch size, and, we argue, why ViTs require patchification in practice. Scan-based recurrence survives the low-information regime on selection but not on transport: serializing a (moving) N-D target into

a 1-D scan destroys the positional structure that copying requires. The global convolution is the only operator *at home* at native resolution: it transports across $32{,}768$ tokens with pixel precision, binds as well as recurrence once the optimization regime is matched, and needs no resolution-dependent preprocessing to do either.

The practical reading is that patchification is not a modeling choice but a workaround for attention's information-to-token requirement, and an operator that does not need the workaround changes what the front of a network can look like. The downstream experiments already point this way: in the 3D motion tasks, HyenaND operates directly on raw $32{,}768$-voxel volumes with no patchification at all, and in 3D medical segmentation (§5.5) it drops into a SwinUNETR-style encoder as a global mixer at $\sim$110k-token stages (a scale where dense attention is infeasible and windowed attention is the standard workaround) matching accuracy at $\sim 11\%$ lower peak training memory. This suggests a role for HyenaND beyond a drop-in mixer: *as a learned, native-resolution replacement for the tokenizer*. A few HyenaND layers operating directly on raw pixels or voxels could aggregate fine-grained structure into information-dense tokens for a backbone (including the attention hybrids of §5.3) turning patch size from a hand-tuned hyperparameter into something the network learns.

D.2 Long-Context Genomics (1D)

D.2.1 Genomics Modeling: Experimental Details

We trained three replicates each of 1B-parameter LLMs of two varieties: full transformer LLMs, and Evo2-inspired mixed transformer/Hyena LLMs [2]. For both architectures we applied the following modifications: embedding weights initialized to stdev$= 1.0$ and not tied to the output projection [41]; activations included in all MLP layers (standard practice); bias excluded from projection layers (Evo2 only); bias added to short convolution layers (Evo2 only). The Evo2 variant is implemented as `striped_hyena_1b_nv` in BioNeMo [14]. For the full-transformer runs we used the same MHA layer configuration as those in the striped Hyena architecture; the full transformer is composed entirely of such MHA layers. The striped or full Hyena models had between 0 and 4 transformer layers mixed in. All models were trained for 72,926 steps (no early stopping) with Adam, global batch size 960, sequences padded to 8,192 bp, sampled from the OpenGenome2 metagenomics subset [2]. Full-transformer runs at learning rate 3×10^{-4} diverged consistently around step 10,000; we reduced their learning rate to 3×10^{-5} for the final reported results.

D.2.2 Genomics Modeling: Pre-divergence comparison

Section 5.2 reports a $10\times$ LR reduction for the full-transformer runs after they diverged around step 10,000. Table 9 reports the same statistical comparison at step 9,899, before any divergence, with the transformer evaluated at both the original (3×10^{-4}) and reduced (3×10^{-5}) learning rates. The hyena ranking matches Table 2: every striped configuration outperforms the transformer at the higher LR, and H_2 remains the top performer. This rules out the reduced LR as a confound for the headline result.

Table 9: Parametric Statistical Comparison at Step 9,899 (before transformer loss explosion) using One-Sided Welch's T-Test (H_1: mean(model) $<$ mean(reference)). Holm–Bonferroni correction is applied per reference column (6 tests each; self-comparisons excluded). The transformer is shown at both the high LR (3×10^{-4}, which later diverged) and the low LR (3×10^{-5}, used for the final results). We use T to denote the transformer with the lower LR to align with that model's name in table 2.

Model Architecture	Mean $\pm$ SD	p vs. $T_{\text{normal-lr}}$	p vs. H_0	Holm p vs. $T_{\text{normal-lr}}$	Holm p vs. H_0
$T_{\text{normal-lr}}$ (Full Transformer, LR=3×10^{-4})	3.0717 ± 0.0049	5.000×10^{-1}	9.833×10^{-1}	—	1.000
T (Full Transformer, LR=3×10^{-5})	3.2682 ± 0.0034	1.000	9.994×10^{-1}	1.000	1.000
H_4 (Mixed, 4 MHA layers)	3.0464 ± 0.0086	9.530×10^{-3}	9.286×10^{-1}	3.812×10^{-2}	1.000
H_3 (Mixed, 3 MHA layers)	3.0449 ± 0.0227	8.617×10^{-2}	8.591×10^{-1}	1.723×10^{-1}	1.000
H_2 **(Mixed, 2 MHA layers)**	3.0031 ± 0.0014	4.004×10^{-4}	8.366×10^{-2}	2.402×10^{-3}	5.020×10^{-1}
H_1 (Mixed, 1 MHA layer)	3.0381 ± 0.0052	6.053×10^{-4}	8.511×10^{-1}	3.027×10^{-3}	1.000
H_0 (Full Hyena, 0 MHA)	3.0242 ± 0.0173	1.670×10^{-2}	5.000×10^{-1}	5.009×10^{-2}	—

D.2.3 Context Length Scalability

The configurations in Table 10 are steady-state throughput measurements from *real, end-to-end training runs* of a 1B striped-hyena model on GB200 GPUs, not fixed-step smoke tests. Each row is the median per-GPU throughput over the hundreds-to-thousands of optimizer steps of one stage of a staged context-extension schedule, run with the full data pipeline (FP8 current-scaling precision, `--use-subquadratic-ops`). Starting from a separate 8K-pretrained 1B model (checkpoint at step 89,000), we progressively extend context and re-interpolate the rotary embeddings, following the gradual context-extension policy of Evo2 [2]. Pretraining and extension use a broader genomic blend dominated by NCBI eukaryotic sequences and stitched GTDB v220, distinct from the OpenGenome2 subset used in §5.2.

Throughput and capability. The *Training* column of Table 10 is the per-GPU throughput of the actual context-extension runs, which use a deliberately memory-conservative configuration (activation recompute enabled) so the full model state fits; it is essentially flat across context length ($\sim$450 TFLOP/s/GPU from 128K to 16M, higher at the TP=1 stages that incur no tensor-parallel all-reduce). This as-run figure is *not* the operator's ceiling: disabling activation recompute (a mock-data throughput benchmark on the same compute path; *Peak* column) raises throughput by $\sim$**1.3**$\times$**, to $\sim$585–620 TFLOP/s/GPU** at 128K–8M, and around 800 at the short TP=1 stages. **16M is the exception**: without recompute the activation footprint no longer fits and the run OOMs, so 16M requires recomputation at $\sim$449 TFLOP/s/GPU (a TP/CP rebalance does not change this). In short, the $\sim$450 TFLOP/s/GPU of the training runs reflects a configuration chosen so the model fits, not a hardware or operator throughput ceiling; 8M is the largest tested length that fits recompute-free, and the two columns are close at 16M because both use recomputation.

Convergence at length. Following the fixed-budget extension stages, we perform additional training at 1M and 16M. Post-training validation PPL improves from 2.836 to 2.757 at 1M; at 16M, exponentiating the mean NLL of the in-training validation plateaus gives PPL 2.927 to 2.880 (Figure 13). Position-binned PPL decreases at large position depths (Figure 14), suggesting context use, although changing sequence support with depth prevents a definitive attribution to distant context.

Table 10: Context-length scalability and throughput capability of the 1B striped-hyena model on GB200 (FP8 current-scaling, `--use-subquadratic-ops`). *Training* = per-GPU throughput of the context-extension runs (activation recompute *on* — the memory-conservative config used so the full state fits). *Peak* = throughput with recompute *off* (a mock-data benchmark on the same compute path), i.e. the operator's capability when not memory-constrained. Recompute-off OOMs at 16M, so 16M requires recomputation (†: value is with recompute on).

Model	Seq Len	TP	CP	Nodes (GPUs)	Training (recompute on)	Peak (recompute off)
1B	16K	1	1	1 (4)	584	796
1B	32K	1	1	1 (4)	606	823
1B	64K	1	1	1 (4)	586	778
1B	128K	2	2	1 (4)	458	605
1B	256K	2	4	2 (8)	458	605
1B	512K	2	8	4 (16)	459	604
1B	1M	2	8	4 (16)	461	616
1B	2M	2	16	8 (32)	457	614
1B	4M	2	32	16 (64)	454	608
1B	8M	2	64	32 (128)	442	584
1B	16M	2	64	32 (128)	443	449†

D.3 ImageNet (Computer Vision, 2D)

D.3.1 Training Setup

Backbone. All HyenaND models use the ViT-5-Small backbone [46]: 12 blocks, hidden dimension 384, patch size 16×16, 224×224 input, 22M parameters.

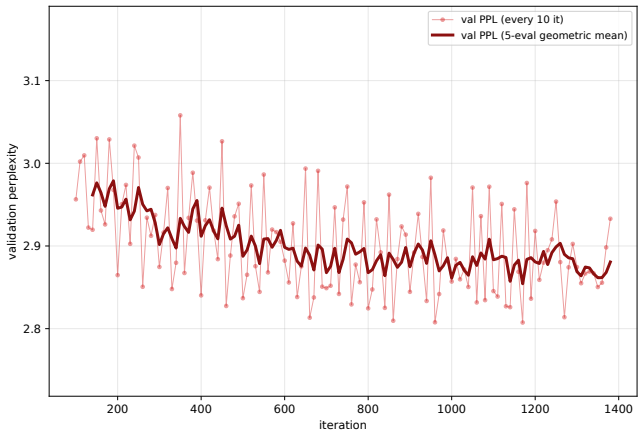


Figure 13: 16M long-convergence validation PPL. The dark curve is the exponential of a five-evaluation running mean NLL (geometric-mean PPL). Iterations 100–1380 are shown, excluding the initial restart transient.

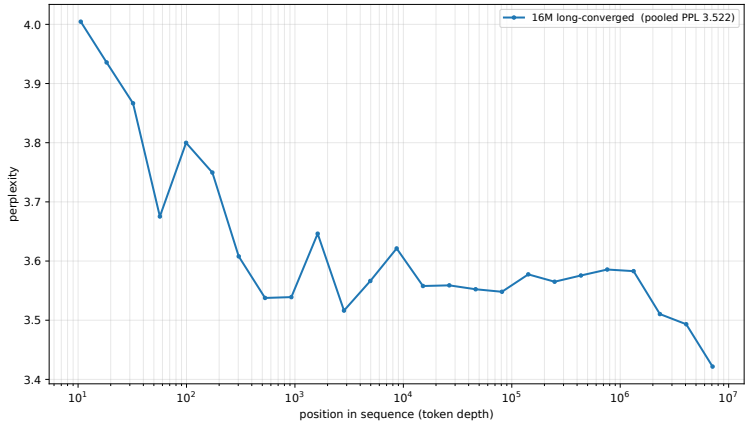


Figure 14: Position-binned PPL of the 16M model after continued training, evaluated on nine stitched-genome sequences held out from training, with lengths up to 9.40M tokens. Each bin reports the exponential of its token-weighted mean NLL. The final bin contains two genomes; changing sequence support limits interpretation of the deep-position decline as evidence of distant-context use.

Training recipe. All models are trained for 800 epochs with LAMB (lr = 4e-3, wd = 0.05), cosine learning-rate schedule, 3-Augment [47], Mixup/CutMix, LayerScale, and EMA on 8×H100 SXM (batch size 2048, bf16-mixed precision).

Model variants. The attention baseline (†, same backbone and recipe as HyenaND) uses 6-head self-attention with RoPE and L2 QK-normalization. Pure HyenaND replaces every attention block with a 2D HyenaND layer: a SIREN-parameterized [37] long convolution evaluated via 2D FFT, operating directly on the 14×14 patch grid without rasterization. The hybrid variants interleave HyenaND and attention at 1:1 $(\mathrm{H\,A})^{\times6}$ and 3:1 $(\mathrm{H\,H\,H\,A})^{\times3}$ ratios within the same backbone.

D.3.2 Patch-Size Ablation

Patchification is the dominant image tokenization strategy for Vision Transformers: by grouping pixels into non-overlapping $p\times p$ patches before feeding them to the model, the token count is reduced by a factor of p^2, keeping the $\mathcal{O}(L^2)$ cost of self-attention tractable. This compression, however, discards spatial detail. Wang et al. [45] recently demonstrated a consistent patchification scaling law: classification accuracy improves monotonically as patch size decreases toward 1×1 (pixel

tokenization), across architectures, tasks, and input resolutions. For standard ViT, this law hits a hard wall at around p=4 (≈3,000 tokens): reducing p further causes the $\mathcal{O}(L^2)$ attention cost to exceed GPU memory, making smaller patches practically infeasible, though FlashAttention [4] kernels push this boundary lower by substantially reducing the memory footprint of attention, enabling p=4 and even p=2 for carefully tuned batch sizes. Wang et al. circumvent this with a Mamba-based $\mathcal{O}(L)$ architecture, reaching p=1 (50,176 tokens, 84.6% top-1 with a base-sized model) at the cost of rasterizing the spatial grid into a 1D scan.

This experiment tests whether HyenaND can exploit the same patchification scaling law without rasterization. Because HyenaND operates via 2D FFT in $\mathcal{O}(L \log L)$, it should degrade far more gracefully than attention as p decreases, while preserving the native 2D spatial structure of the image at every scale. We ablate $p \in \{16, 8, 4, 2\}$ on the ViT-5-Small backbone (ImageNet-1K, 224×224) for the following mixer configurations: pure Attention, pure HyenaND, and the two hybrid ratios (H A)$^{\times 6}$ and (H H H A)$^{\times 3}$. At p=16 the token count is 196; at p=2 it reaches 12,544. All settings follow the training recipe in §D.3.1. Table 11 additionally reports matched recurrence-based Mamba-attention hybrids at the same 1:1 (M A)$^{\times 6}$ and 3:1 (M M M A)$^{\times 3}$ ratios, isolating the token mixer. The HyenaND hybrids outperform their Mamba counterparts at every measured patch size, despite the bidirectional Mamba blocks costing both more parameters (30M/34M vs. 22M) and more FLOPs (e.g. 12.5/14.0 vs. 9.7/9.8 GFLOPs at patch 16×16). Table 11 reports top-1 accuracy and GFLOPs at each patch size, notably demonstrating that at the finest resolution (p=2), pure Attention requires around 6× more FLOPs compared to pure HyenaND.

Table 11: Patch-size ablation on ImageNet-1K. Top-1 accuracy (%) and GFLOPs on a single H100 SXM. Mamba hybrids (MA, MMMA) share the ViT-5-Small backbone with attention layers replaced by bidirectional Mamba blocks; their GFLOPs are analytic estimates (standard selective-scan accounting) and patch 2 was not run (–) for compute reasons. At equal hybrid ratios the Mamba variants use more parameters than the HyenaND hybrids (30M/34M vs. 22M).

		Attention		**HyenaND** (pure)		**Hybrid** (HA)$^{\times 6}$		**Hybrid** (HHHA)$^{\times 3}$		**Mamba** (MA)$^{\times 6}$		**Mamba** (MMMA)$^{\times 3}$	
Patch	**Tokens**	**Top-1**	**GFLOPs**	**Top-1**	**GFLOPs**	**Top-1**	**GFLOPs**	**Top-1**	**GFLOPs**	**Top-1**	**GFLOPs**	**Top-1**	**GFLOPs**
16	196	81.8	9.41	81.5	9.97	82.1	9.69	82.0	9.83	81.8	12.46	81.0	13.99
8	784	84.3	45.51	83.7	39.04	84.2	42.27	84.0	40.66	83.9	52.85	82.9	56.70
4	3,136	85.1	317.34	84.0	155.32	85.0	236.33	84.4	195.83	84.4	278.14	83.9	259.40
2	12,544	85.3	3,443.93	84.0	623.02	85.2	2,033.48	84.6	1,328.25	–	–	–	–

D.4 The Well: Results and Experimental Details

We compare three architectures on five selected datasets from The Well. The CNextU-net baseline reproduces the strongest published model from The Well [31]: a 4-stage U-Net encoder–decoder with ConvNext blocks (42 initial features, 2 blocks per stage, ∼18.6M parameters) that operates at full pixel resolution. Against this, we evaluate two patch-based models that share a common ResidualNetwork backbone (12 blocks, hidden dimension 384, RMSNorm, GLU-activated MLPs) and differ only in their sequence mixer: Attention, using 6-head self-attention with QKV projections and axial RoPE (non-causal); and Hyena, a HyenaND operator whose implicit kernel is a 3-layer SIREN MLP modulated by a learned Gaussian spatial envelope. Input fields are tokenized into non-overlapping p^d patches with $p \in \{2, 4, 8\}$. All models are trained for 24 hours or up to 110,000 iterations with AdamW (cosine schedule, 5% warmup, gradient clipping at 1.0, `bf16`-mixed precision) on a single H100 GPU. This budget exceeds the 12-hour time-box of the published The Well baselines; all three architectures are trained under the same extended protocol. The Attention and HyenaND models have parameter counts (12–19M) comparable to the ConvNext baseline.

Table 12 reports validation VRMSE and training throughput for all models across the five The Well datasets and three patch sizes. HyenaND attains the lowest VRMSE on every dataset; the gap over Attention widens as the patch size decreases and sequence length grows (see also Figure 5 in the main text).

D.5 3D Medical Imaging: Experimental Details

SwinUNETR [12] encodes volumetric inputs through four hierarchical stages of windowed self-attention, with a patch-merging factor of 2 applied after each stage. With ROI= 96^3 inputs, the per-stage token counts span nearly three orders of magnitude: $48^3 \approx 1.1 \times 10^5$, $24^3 \approx 1.4 \times 10^4$,

Table 12: **Patch-size ablation on The Well.** Validation VRMSE (lower is better) and training throughput (samples/sec) on a single H100 GPU. CNextU-net operates at full pixel/voxel resolution (no patch size). All runs target 24 hours or 110,000 iterations. Patch-based models use batch size 64 for $p \in \{4, 8\}$ and batch size 16 for p=2 due to memory constraints at long sequences; samples/sec is reported to normalize across batch sizes.

			CNextU-net		**Attention**		**Hyena**	
Dataset	**Patch**	**Tokens**	**VRMSE**	**samples/s**	**VRMSE**	**samples/s**	**VRMSE**	**samples/s**
gray_scott_reaction	8	256	—	—	0.0520	89.6	0.0092	87.0
_diffusion	4	1,024	—	—	0.0538	92.8	**0.0090**	87.0
$[128 \times 128]$	2	4,096	—	—	0.0974	6.9	0.0091	32.0
	—	16,384	0.2319	384.0	—	—	—	—
active_matter	8	1,024	—	—	0.0586	91.5	0.0073	89.0
$[256 \times 256]$	4	4,096	—	—	0.0616	85.1	0.0080	87.0
	2	16,384	—	—	0.0914	6.9	**0.0070**	42.1
	—	65,536	0.0347	89.0	—	—	—	—
acoustic_scattering	8	1,024	—	—	0.0456	369.3	0.0086	271.4
$[256 \times 256]$	4	4,096	—	—	0.0569	151.0	0.0068	105.6
	2	16,384	—	—	0.1057	13.9	**0.0062**	25.3
	—	65,536	0.0082	344.3	—	—	—	—
shear_flow	8	2,048	—	—	0.0354	130.4	0.0268	101.8
	4	8,192	—	—	0.1049	32.2	0.0823	41.6
$[256 \times 512]$	2	32,768	—	—	0.4418	2.5	0.1185	11.2
	—	131,072	**0.0262**	124.2	—	—	—	—
MHD	8	512	—	—	0.3044	48.0	0.2810	47.4
$[64 \times 64 \times 64]$	4	4,096	—	—	0.2164	23.7	0.1088	33.9
	2	32,768	—	—	0.3037	1.3	**0.0543**	13.0
	—	262,144	0.2108	7.0	—	—	—	—
supernova_explosion	8	512	—	—	0.6117	217.0	0.6151	271.4
$[64 \times 64 \times 64]$	4	4,096	—	—	0.3879	217.0	0.3578	144.6
	2	32,768	—	—	0.3000	2.6	**0.1943**	5.0
	—	262,144	0.7400	21.1	—	—	—	—
euler_multi	8	4,096	—	—	0.1293	58.0	**0.0311**	55.9
_quadrants	4	16,384	—	—	OOM	—	0.0378	24.8
$[512 \times 512]$	2	65,536	—	—	OOM	—	0.1088	5.7
	—	262,144	0.0332	63.8	—	—	—	—
helmholtz_	8	4,096	—	—	0.0050	40.1	**0.0042**	30.2
staircase	4	16,384	—	—	0.0147	7.6	0.0480	6.3
$[1024 \times 256]$	2	65,536	—	—	OOM	—	0.0443	1.9
	—	262,144	0.0045	35.1	—	—	—	—

$12^3 = 1728$, $6^3 = 216$. We replace the per-stage mixer with our multi-dimensional Hyena operator lifted to D=3 according to a per-stage pattern $s \in \{A, H\}^4$, yielding four variants: **AAAA** (SwinUNETR baseline), **HHHH** (all-Hyena), **HAHA** (striped hybrid), and **HHAA** (hierarchical hybrid). All variants share the decoder, feature widths (C_0=48), and optimization recipe, and are trained from scratch for 100 epochs on 4×NVIDIA GB200 GPUs. The training corpus is a pre-extracted patch collection derived from PanTS [18] (252 376 patches of size 96^3); evaluation uses 901 held-out volumes against the 28-class abdominal segmentation target with mean Dice as the primary metric.

Stage placement aligns with the resolution hierarchy. The ordering **HHAA** > **HAHA** > **HHHH** > **AAAA** has a simple explanation in the per-stage token counts. Hyena's advantage over windowed attention is greatest at the high-resolution early stages, where its global FFT receptive field captures anatomical extents that windowed attention can only approximate by stacking shifted windows; at the deep stages the contracted grid fits inside a single attention window, so Hyena's

global advantage collapses while attention becomes essentially free. The hierarchical placement **HHAA** aligns this gradient with the encoder's resolution hierarchy. The striped placement **HAHA** uses the same Hyena budget but misallocates it: a Hyena block at stage 2 (where the contracted grid no longer rewards a global filter) and an attention block at stage 1 (where windowed attention is most constrained). This is a 3D analogue of the striped-Hyena ordering Evo2-style language models exploit along depth (Section 5.2): the best configuration is a hybrid in which Hyena is deployed where its global-context bias is most useful, namely the early, high-resolution end of the encoder.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are directly supported by the theoretical results (§3–4) and experiments across four domains (§5); no aspirational goals are presented as attained.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix A (referenced from the conclusion, §6): the per-input (rather than per-position) granularity of input-dependence, the sensitivity to the SIREN frequency and Gaussian-window initialization hyperparameters, and the current restriction of `nvSubquadratic-ops` to NVIDIA GPUs.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All theorems (gated Hyena computes separable attention, SIREN length generalization, ND initialization) state assumptions explicitly and are proved in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Full architecture descriptions, training recipes, and hyperparameters are provided in the appendix. The `nvSubquadratic-ops` library is released publicly.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is released as the `nvSubquadratic-ops` library (anonymized URL in supplemental material). All datasets used (ImageNet, The Well, genomics benchmarks) are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Optimizer, learning-rate schedule, augmentation, hardware, and batch size are specified per domain in the appendix; key settings are summarised in the main text.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: No error bars are reported. Experiments such as ImageNet training (800 epochs on $8\times$H100) make multiple independent runs computationally prohibitive.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware type (H100 SXM), number of GPUs, training duration, and batch size are specified per domain in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work presents foundational sequence-mixing architecture research with no foreseeable ethical violations.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This is foundational ML infrastructure research; we do not foresee a direct path to harmful applications and have omitted a broader impacts section.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: The paper releases an architecture library (`nvSubquadratic-ops`), not pre-trained generative models or scraped datasets; no special safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenges, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and code dependencies (FlashAttention, mamba-ssm, ImageNet, The Well, genomics benchmarks) are cited with their original papers and used in accordance with their respective licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The `nvSubquadratic-ops` CUDA library is documented in Appendix C and released alongside the submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: LLMs are not a component of the core method. Any LLM use was limited to writing assistance and does not impact the scientific methodology or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.