ENA: Efficient N-dimensional Attention

Anonymous authors
Paper under double-blind review

Abstract

Efficient modeling of long sequences of high-order data requires a more efficient architecture than Transformers. In this paper, we investigate two key aspects of extending linear recurrent models, especially those originally designed for long-context language modeling, to high-order data: scanning strategies and attention-hybrid architectures. Empirical results suggest that scanning provides limited benefits, while attention-hybrid models yield promising results. Focusing on the latter, we further evaluate types of attention that are worth integrating and find that tiled high-order sliding window attention (SWA) is efficient in both theory and practice. We term the resulting hybrid architecture of linear recurrence and high-order SWA Efficient N-dimensional Attention (ENA), and conduct several experiments to evaluate its effectiveness. With comparable performance to Transformers and higher efficiency, ENA offers a promising and practical solution for ultra-long high-order data modeling.

1 Introduction

Softmax attention in LLMs has quadratic time complexity, making it inefficient for long sequences. To address this, linear recurrent models[1] have emerged as efficient alternatives. Representative variants include RetNet (Sun et al., 2023), HGRN (Qin et al., 2024), GLA (Yang et al., 2024a), GSA (Zhang et al., 2024), Mamba (Gu & Dao, 2023) and RWKV (Peng et al., 2023). Subsequent advancements such as DeltaNet (Yang et al., 2024b), Gated DeltaNet (Yang et al., 2025), DeltaProduct (Siems et al., 2025), LaCT (Zhang et al., 2025b) and MesaNet (von Oswald et al., 2025) further enhance expressiveness while preserving linear-time complexity and are optimized for parallel training.

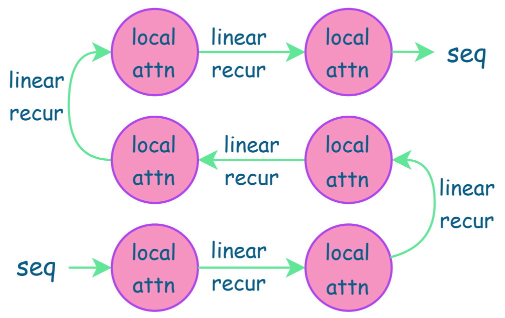

Figure 1: An overview of the architecture of ENA with alternating layers of linear recurrence and local attention. Note that although we ultimately perform no sequence permutation, the framework remains compatible with any scanning.

Bridging the dimensional gap with scanning

Drawing inspiration from their efficacy in language modeling, these linear models have been extended to high-order data modeling [2]. To reconcile the inherent 1-dimensional nature of linear models with the N-dimensional structure of high-order data (e.g., images, videos), prior works typically employ various "scanning" methods. These methods transform an input into single (single-pass scan) or multiple (multi-pass scan) derived sequences using predefined patterns, subsequently processed by a shared linear model, as illustrated in Fig. 4.

[1]In this paper, "linear recurrent models", "linear recurrence" and "linear models" are used interchangeably to denote models that perform sequence modeling via state updates with linear time complexity. The same applies to "softmax attention" and "full attention".

[2]For convenience, we refer to all data with more than one dimension as higher-order data, including images, which have only two dimensions.

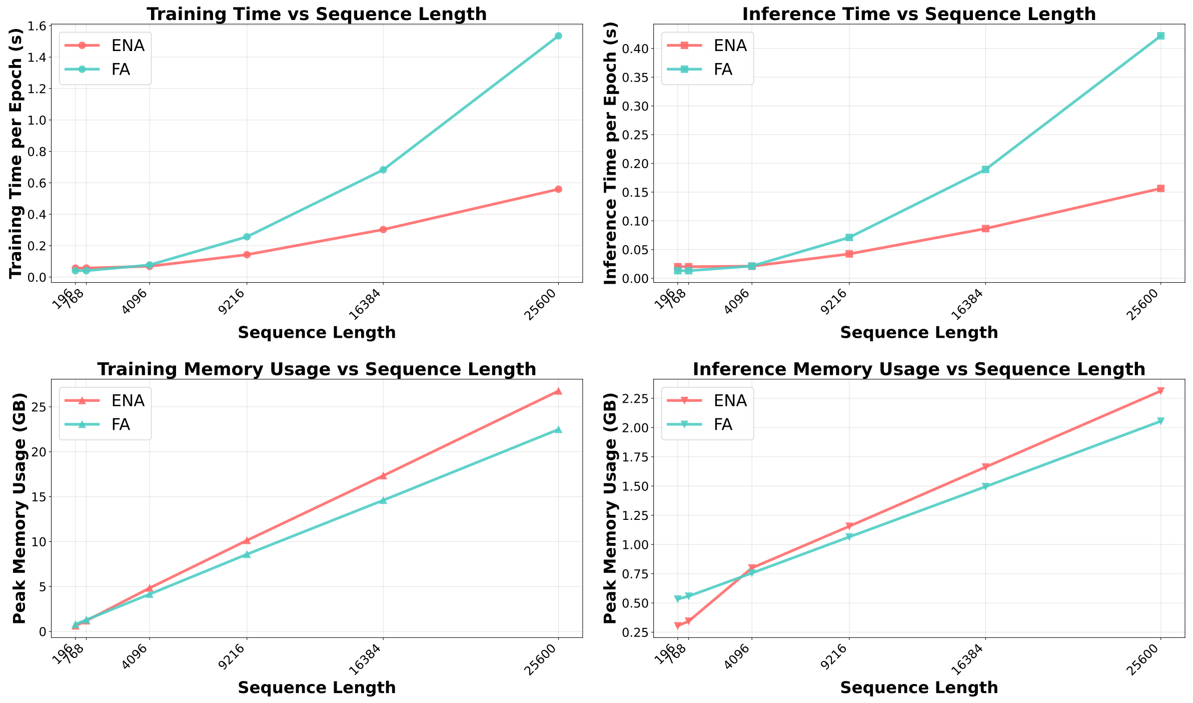

Figure 2: Performance comparison between ENA and Flash Attention (FA)-based Transformer vision encoders across different sequence lengths. Training and inference times (top row), as well as peak memory usage (bottom row), are measured with a batch size of 4 over 50 epochs using mixed precision on a single GPU. For sequence lengths ≥ 1152, ENA employs 2D STA (tile size: 16×8) with a window size covering approximately 30% of the tokens, resulting in 70% sparsity.

However, multi-pass scanning methods process multiple sequences at once (serially or in parallel), causing significant speed and memory overhead. Consequently, in a simple stacked-layer architecture, multi-pass scanning often renders linear models even slower than a Transformer using FlashAttention (Dao, 2023). To address this inefficiency, many works introduce complex architectures such as downsampling to reduce computational cost. Recent works (Wang et al., 2024; Zhu et al., 2024) propose single-pass approaches that permute the sequence between blocks while ensuring only a single sequence is fed to the linear model. However, the effectiveness of these single-pass methods has not yet been comprehensively and fairly evaluated.

Bridging the dimensional gap with attention-hybrid architectures instead

Alternatively, hybrid architectures bridge the dimensional gap by combining linear recurrence with attention. While linear recurrence compresses information into a fixed-size state, it may overlook important local patterns. Introducing attention helps compensate for this limitation. For long sequences, we primarily consider local attention, where attention is computed within a fixed-size neighborhood. This design leverages the inherent locality in data while maintaining computational efficiency. For short sequences, we adopt full attention in the hybrid architecture.

We categorize local attention for high-order data into block attention and sliding window attention (SWA). Block attention divides the sequence into non-overlapping blocks and computes attention within each block. In contrast, SWA defines local regions centered around each token (or token tile in sliding tile attention), enabling finer-grained locality modeling. Our experiments show that a hybrid architecture using high-order SWA outperforms one using block attention.

Based on our findings, we propose **Efficient N-dimensional Attention (ENA)**, a simple hybrid architecture that combines linear recurrence and high-order SWA, as illustrated in Fig. 1. ENA maintains a

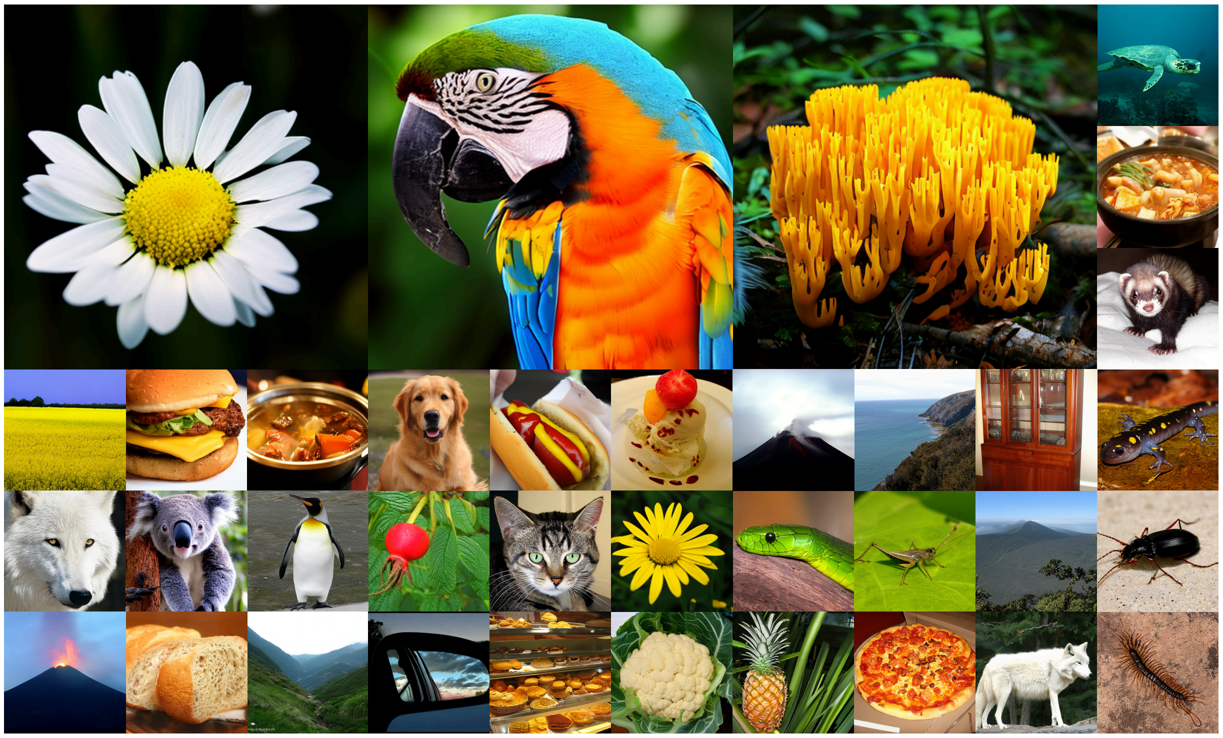

Figure 3: Selected image generation results from *ena-deltanet-sta-w24x24-t8x8-xl-gen2d* on ImageNet with a resolution of 512×512.

straightforward architecture of stacked layers and employs a simple block-wise hybrid strategy for ease of implementation, thus avoiding unnecessary complexity. For ENA, we primarily use DeltaNet (Yang et al., 2024b) with no sequence permutation (i.e., scanning) as the linear model. For hardware-efficient high-order SWA, we use Sliding Tile Attention (STA Zhang et al. (2025a)).

This paper is organized as follows: we first introduce two ways for bridging the dimensional gap: scanning and integrating attentions, accompanied by a small-scale pilot experiment. We then evaluate various design choices via ImageNet classification. After establishing ENA's architecture, we conduct several experiments to validate its effectiveness. Finally, we provide additional discussions, which can be skipped if desired.

2 From 1D to ND

This section introduces methods for adapting the 1D nature of linear models to high-order data. We discuss two main approaches: scanning methods for pure linear models and attention-based hybrid architectures.

2.1 Scanning for Linear Recurrence

The diverse scanning methods outlined in Table 1 can be described with a general formulation. Let $\mathbf{X} \in \mathbb{R}^{B \times L \times D}$ be the input tensor, where B is the batch size, L is the sequence length, and D is the feature dimension. The core of each method involves a token mixer operation, denoted as $\mathcal{TM}(\cdot)$, which maps an input tensor to an output tensor of the same shape.

For single-pass methods, the transformation can be expressed generally. For example, the **uni-scan** performs a direct token mixing:

$$\mathbf{Y} = \mathcal{TM}(\mathbf{X}) \tag{1}$$

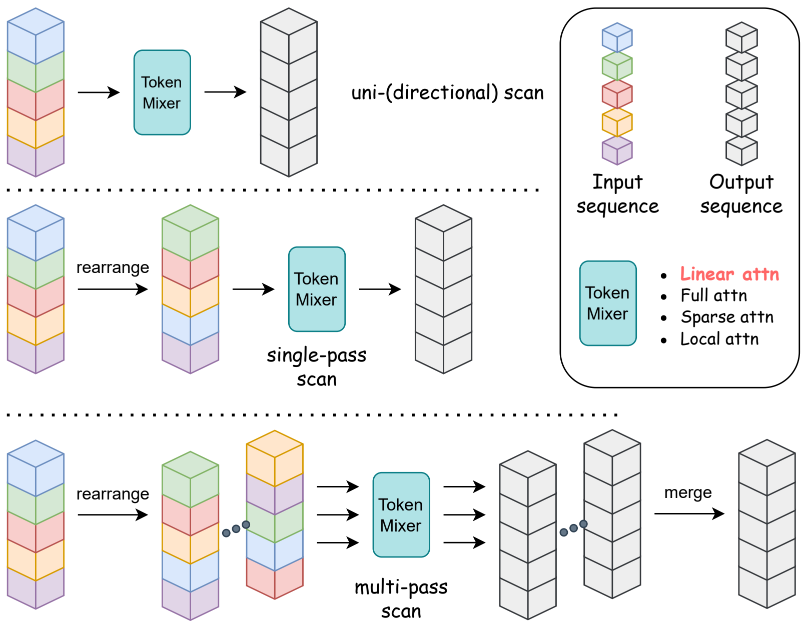

Figure 4: A simple illustration of the operations performed by different scanning methods. Single-pass scan rearranges the input into one sequence, while multi-pass scanning generates multiple input sequences. For multi-pass scanning, the output sequences are merged into a single output sequence. Uni-directional scan is a special case of single-pass scan where no rearrangement is performed. The rules for rearrangement and merging are mostly predefined and fixed.

Table 1: Overview of different scanning methods considered. B represents batch size, L represents sequence length, and D represents the feature dimension. In multi-pass methods, we arrange multiple sequences along the batch dimension to enable parallel processing, even though prior works might perform these operations sequentially. Since single-pass scanning is underexplored, we introduce several new scan types for comparison, marked in light blue and light red. Methods highlighted in light red leverage the multi-head design in token mixers to enable multi-directional scan within a single pass.

Scan Type	**Pass**	**Operation Flow**
uni-scan	Single	`[B,L,D] → token mixer → [B,L,D]`
switch-scan	Single	`[B,L,D] → token mixer → flip/transpose → [B,L,D]`
flip-scan	Single	`[B,L,D] → flip → token mixer → [B,L,D]`
1d-shift-scan	Single	`[B,L,D] → token mixer → shift → [B,L,D]`
2d-shift-scan	Single	`[B,L,D] → token mixer → reshape → 2D shift → [B,L,D]`
random-scan	Single	`[B,L,D] → random shuffle → token mixer → [B,L,D]`
learnable-scan	Single	`[B,L,D] → learnable module → token mixer → [B,L,D]`
multi-head-bi-scan	Single	`[B,L,D] → head-wise flip → token mixer → head-wise reverse flip → [B,L,D]`
multi-head-2d-scan	Single	`[B,L,D] → head-wise rearrange → token mixer → head-wise reverse rearrange → [B,L,D]`
bi-scan	Multi	`[B,L,D] → flip → [2B,L,D] → token mixer → [2B,L,D] → merge → [B,L,D]`
cross-scan	Multi	`[B,L,D] → rearrange → [4B,L,D] → token mixer → [4B,L,D] → merge → [B,L,D]`

Methods that permute the sequence such as **flip-scan** and **switch-scan** can be unified as applying a pre-processing mapping $OP_{pre}(\cdot)$ and a post-processing mapping $OP_{post}(\cdot)$ around the token mixer:

$$\mathbf{Y} = OP_{post}(\mathcal{TM}(OP_{pre}(\mathbf{X}))) \tag{2}$$

Here, $OP_{pre}(\cdot)$ and $OP_{post}(\cdot)$ may be identity mappings or concrete operations such as $flip(\cdot)$ or $shift(\cdot)$, depending on the specific method.

For multi-pass methods like **bi-scan**, the input $\mathbf{X}$ is first transformed into multiple (N) views (e.g., $\mathbf{X}_1 = \mathbf{X}$, $\mathbf{X}_2 = \text{flip}(\mathbf{X})$ for $N = 2$). These views are processed in parallel using a shared token mixer $\mathcal{TM}(\cdot)$, and the outputs are merged into a single sequence:

$$\mathbf{Y} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{TM}(\mathbf{X}_i) \tag{3}$$

The **cross-scan** follows a similar pattern with $N = 4$ views. Note that the merge operation can be more sophisticated than simple summation; however, we adopt summation for simplicity.

While we include two common multi-pass methods, our primary focus is on the less explored single-pass approaches. To facilitate a comprehensive comparison, we introduce several single-pass variants: *1D scan* and *2D shift-scan*, which shift tokens along spatial or flattened dimensions, respectively; *random-scan*, which applies a random token permutation prior to the token mixer; and *learnable-scan*, which learns token re-ordering through a trainable permutation matrix, employing Gumbel-Softmax for differentiable selection. *Learnable-scan* is initialized near identity and trained end-to-end. However, it is inefficient and, in our current implementation, can lead to the omission of tokens due to repeated selections. Finally, we also introduce multi-head scanning methods, including *multi-head-bi-scan* and *multi-head-2d-scan*, which divide the attention heads into groups and assign each group a dedicated scanning direction. These methods aim to achieve multi-directional scanning within a single pass; however, **in our current experiments, they yield no observable performance gains over a simple uni-scan in our experiments.**.

2.2 Integrating Local Attention

An alternative approach incorporates auxiliary token mixers to compensate for inherent limitations of linear models, such as their lack of fine-grained local information. The linear model compresses global context, while the auxiliary mixer captures local patterns, creating a complementary architecture. A natural consideration for such token mixers is local attention, in which attention is restricted to a fixed window, leveraging the inherent locality of the data.

The block-wise hybrid approach used in 1 can be formulated as follows. Let $\mathbf{X}^{(0)}$ denote the initial embeddings. For the i-th block in a model, the output $\mathbf{X}^{(i)}$ is computed as:

$$\begin{aligned}\mathbf{X}'^{(i)} &= \mathbf{X}^{(i-1)} + \mathcal{TM}^{(i)}(\mathcal{N}(\mathbf{X}^{(i-1)}))\\ \mathbf{X}^{(i)} &= \mathbf{X}'^{(i)} + \mathcal{CM}(\mathcal{N}(\mathbf{X}'^{(i)}))\end{aligned} \tag{4}$$

where $\mathcal{N}$ denotes normalization and $\mathcal{CM}$ is the channel mixer. The token mixer $\mathcal{TM}^{(i)}$ alternates according to the block index:

$$\mathcal{TM}^{(i)} = \begin{cases}\mathcal{TM}_{\text{linear}} & \text{if } i \bmod 2 = 1\\ \mathcal{TM}_{\text{local}} & \text{if } i \bmod 2 = 0\end{cases} \tag{5}$$

Here, $\mathcal{TM}_{\text{linear}}$ denotes the linear model, and $\mathcal{TM}_{\text{local}}$ denotes local attention. This alternating strategy aims to balance global context aggregation with local feature refinement.

We categorize local attention into two main types:

1. Local attention with non-overlapping blocks (Block attention). This type of local attention is easy to implement with Flash Attention and achieves high hardware utilization. However, as we will demonstrate in our experiments, this simple partitioning impose restrictions on queries near the block borders, resulting in sub-optimal performance. Swin (Liu et al., 2021) addresses this by shifting blocks across layers, but at the cost of increased design complexity.

2. High-order sliding window attention (SWA). These types of local attention are natural extensions of SWA from language modeling to higher-order domains, imposing no restrictions on border queries.

However, as illustrated in Zhang et al. (2025a), naive implementations (Hassani et al., 2023; Liu et al., 2024) suffer from poor hardware utilization and thus offer little to no speedup over full attention, despite having significantly fewer FLOPs due to high attention sparsity.

As prior works point out, the primary reason for slow high-order SWA is the generation of mixed blocks in the attention map, where only some elements require computation. Therefore, Sliding Tile Attention (STA Zhang et al. (2025a)) addresses this by sliding the window tile by tile, which eliminates mixed blocks and achieves true hardware speedup over full attention. At a high level, STA is a coarser-grained variant of high-order SWA. This design choice enables it to leverage the underlying hardware efficiently and achieve tangible speedups.

2.3 Pilot Experiments

We conduct pilot experiments using tiny models on small-scale datasets (CIFAR-10/100 and Tiny ImageNet) to evaluate the effectiveness of multi-pass scanning methods (bi-scan and cross-scan), which are otherwise too slow to test at larger scales. While such methods aim to bridge the dimensional gap, they introduce additional computational and memory overhead. Our central question is: *Is the extra cost of multi-pass scanning worth it?*

We compare bi-scan and cross-scan against uni-scan (baseline), random-scan (which helps prevent overfitting), and hybrids that insert a full attention layer at the second block. Linear models include DeltaNet (Yang et al., 2024b), Gated DeltaNet (Yang et al., 2025), GSA (Zhang et al., 2024), HGRN, HGRN2 (Qin et al., 2024), and RetNet (Sun et al., 2023). All models use 6 layers and under 10M parameters, trained for 50/25/30 epochs on CIFAR-100/10 and Tiny ImageNet, respectively, with a learning rate of 1×10^{-4} and a 0.2 warm-up ratio.

As shown in Table 2, multi-pass scans achieve the best results in only 5 out of 18 settings, despite increased latency and memory usage. In contrast, inserting a single full attention layer yields consistent gains. These results demonstrate that the cost of multi-pass scanning is unjustified even in small-scale settings.

3 Evaluate Design Choices

This section evaluates two primary strategies for extending 1D linear models to ND data: scanning and attention-based hybrid architectures on ImageNet classification to guide our architecture design.

3.1 Scanning or Hybrid?

We evaluate a range of scanning methods using DeltaNet (Yang et al., 2024b) on ImageNet with a short sequence length, with the baseline being Transformer. For these short sequences, full attention is computationally feasible and is thus used in the hybrid model. In Table 3, we compare several scanning methods (primarily single-pass) along with full attention hybrid models. We observe that uni-scan, as a simple baseline, already yields good performance, whereas many single-pass scanning variants degrade performance compared to the baseline. Bi-scan and multi-head-scan fail to show notable improvement over the uni-scan baseline. In contrast, integrating full attention into several layers yields a notable performance boost. Regarding the position and number of attention layers, we find that:

1. Integrating attention into half of the layers yields performance comparable to Transformer.
2. Interleaving attention layers across the model achieves better performance than stacking attention only in the deeper layers, as discussed in (Hatamizadeh & Kautz, 2024).
3. In our implementation, scanning alone fails to improve performance and can even be detrimental.

To corroborate this finding, we evaluate single-pass scanning using more linear model types in Table 5. The results consistently show that uni-scan outperforms other single-pass variants, reaffirming that these scanning methods offer no advantages over a simple uni-scan baseline..

Table 2: Pilot experiments conducted with tiny models on several small datasets: CIFAR-100 (C100), CIFAR-10 (C10), and TinyImageNet (TIN). The terms *rand* and *attn1* denote random-scan and the addition of a full attention layer at the second block, respectively.

(a) DeltaNet

Method	C100	C10	TIN
uni	16.66	60.30	19.66
rand	27.22	62.93	28.62
bi	20.91	59.82	20.94
cross	22.58	61.50	22.20
uni + attn1	20.98	61.87	20.35
rand + attn1	29.17	63.90	28.82

(b) GDN

Method	C100	C10	TIN
uni	24.12	68.47	20.35
rand	35.00	70.42	28.22
bi	28.81	66.75	21.61
cross	31.95	69.14	26.16
uni + attn1	28.59	69.56	21.67
rand + attn1	35.22	70.85	28.11

(c) GSA

Method	C100	C10	TIN
uni	21.96	67.18	25.01
rand	24.29	58.02	27.71
bi	26.31	63.02	26.29
cross	27.43	62.20	24.92
uni + attn1	22.44	66.88	26.98
rand + attn1	23.35	58.22	26.39

(d) HGRN

Method	C100	C10	TIN
uni	23.72	74.60	26.83
rand	31.16	66.30	27.95
bi	30.99	71.84	27.54
cross	31.53	73.78	29.72
uni + attn1	37.32	75.35	29.01
rand + attn1	34.15	67.54	29.05

(e) HGRN2

Method	C100	C10	TIN
uni	24.42	74.03	27.46
rand	32.08	67.43	26.33
bi	31.94	70.20	26.89
cross	34.22	73.69	28.07
uni + attn1	34.28	75.34	27.73
rand + attn1	31.07	67.79	26.55

(f) RetNet

Method	C100	C10	TIN
uni	26.60	70.65	25.53
rand	33.91	69.56	28.82
bi	35.26	69.05	26.10
cross	37.72	70.34	29.25
uni + attn1	33.60	70.79	26.08
rand + attn1	33.66	69.09	27.46

Meanwhile, we evaluate cross-scan using a smaller batch size due to its significant memory overhead. As shown in Table 4, we observe that:

5. Although cross-scan outperforms uni-scan, it incurs significantly higher latency and memory overhead. Moreover, its performance still lags behind that of the Transformer.
6. Integrating just two full attention layers achieves even higher accuracy, making the cost of cross-scan unjustified.

In summary, scanning methods offer little to no performance benefit compared to forming a hybrid model with attention. Therefore, hybrid attention is a more promising choice. However, when processing longer sequences, even several full attention layers can impose significant latency, making local attention a necessary component for processing longer sequences efficiently.

3.2 Choosing the Right Local Attention

As noted in Section 2.2, there are two types of local attention: Block Attention, which operates in non-overlapping blocks, and high-order SWA with a flexible local window. We evaluate their effectiveness using a longer sequence length (784) with DeltaNet on ImageNet. The results are presented in Table 6. The baselines are pure DeltaNet with no scanning, pure 2D SWA, and DeltaNet + full attention. For all the

Table 3: Top-1 accuracy of DeltaNet (30M parameters, 12 layers) on ImageNet-1K validation set. The input resolution is set to 224 with a patch size of 16, resulting in a sequence length of 196. The model is trained from scratch under various settings for 100 epochs, with a batch size of 2048 (unless otherwise specified, 2048 is used as the default batch size for ImageNet training). For comparison, a baseline transformer trained using the same codebase is also included.

Variant	Top-1 Acc (%)
Baseline	
Transformers	78.51
Single-pass Scan Variants	
uni-scan	75.75
random-scan	68.91
flip-scan	72.78
switch-scan	72.63
2dshift-scan	72.16
learnable-scan	72.64
Multi-pass Scan Variants	
bi-scan	75.45
Multi-Head Scan Variants	
multi-head-cross-scan	75.34
multi-head-bi-scan	75.04
Uni-Scan with Full Attention Layers	
Layers: *5,11*	76.77
Layers: *0,6*	77.20
Layers: *3,7,11*	77.38
Layers: *0,2,4,6,8,10*	78.28
Layers: *6,7,8,9,10,11*	77.75
Uni-Scan with 1D Sliding Window Attention	
Layers: *0,2,4,6,8,10*; Window Size: 32	77.16

Table 4: Validating the effectiveness of cross-scan. We report Top-1 accuracy (%) on ImageNet-1K validation set. All models are trained from scratch for 100 epochs at a resolution of 224 with a patch size of 16, resulting in a sequence length of 196. Since cross-scan requires more GPU memory, we use a batch size of 1024. Uni-scan + full attn {0, 6} denotes hybrid architecture where layer 0 and 6 are replaced with full attention layers. We can observe that simply integrating two full attention layer could yield better result than cross-scan.

Model / Configuration	Top-1 Acc (%)
uni-scan	76.42
cross-scan	77.30
uni-scan + full attn {0, 6}	77.71

local attentions, we consider both 1D and 2D variants, and we use NATTEN (Hassani et al., 2023) as the implementation for 2D SWA.

From the results, we can conclude that:

Table 5: Top-1 accuracy (%) of Single-pass Scan Variant using various linear model on ImageNet-1K validation set. We evaluate various scan methods (uni-scan, flip-scan, switch-scan, and 2D-shift-scan with a shift size of 7) across different linear model. All models are trained from scratch for 100 epochs at a resolution of 224 with a patch size of 16, resulting in a sequence length of 196.

	Top-1 Acc (%)			
Model	uni-scan	flip-scan	switch-scan	2d-shift-scan
DeltaNet	75.75	71.22	72.63	72.16
HGRN	74.11	71.22	70.80	69.45
RetNet	64.03	58.06	54.48	55.57

Table 6: Top-1 accuracy (%) on the ImageNet-1K validation set for hybrid DeltaNet models with 30M parameters and 12 layers. For DeltaNet hybrid architectures (denoted as "DeltaNet + ..."), odd-numbered layers utilize DeltaNet, while even-numbered layers employ the specified attention mechanism as token mixers. We use an input resolution of 448 with a patch size of 16, resulting in a sequence length of 784. For 2D sliding window attention, we adopt the implementation from Hassani et al. (2023). All models are trained from scratch for 100 epochs.

Model / Configuration	Top-1 Acc (%)
Baselines	
Pure DeltaNet	64.17
Pure 2D SWA (window size=14x14)	66.83
DeltaNet + Full Attention	68.72
Hybrid with 1D Attention	
DeltaNet + 1D BA (block size=128)	66.58
DeltaNet + 1D BA (block size=196)	66.47
DeltaNet + 1D SWA (window size=128)	67.19
DeltaNet + 1D SWA (window size=256)	67.91
Hybrid with 2D Attention	
DeltaNet + 2D BA (block size=14x14)	63.66
DeltaNet + 2D SWA (window size=14x14)	68.17

Table 7: Top-1 accuracy (%) and training time for hybrid DeltaNet models with sparse sequences on ImageNet-1K. For DeltaNet hybrid architectures (denoted as "DeltaNet + ..."), odd-numbered layers utilize DeltaNet, while even-numbered layers employ the specified attention mechanism. Input resolution is 448, patch size is 7 (sequence length 4096). All models are trained from scratch for 100 epochs.

Model / Configuration	Training Time	Top-1 Acc (%)
Baselines		
DeltaNet + Full Attention	14h 42m	66.11
Hybrid with 1D Attention		
DeltaNet + 1D SWA (window size=256)	–	62.35
Hybrid with 2D Attention		
DeltaNet + 2D SWA (window size=16x16)	13h 03m	66.34
DeltaNet + 2D STA (window size=32x16, tile size=16x8)	–	65.28
DeltaNet + 2D STA (window size=48x24, tile size=16x8)	10h 55m	66.41

1. The choice of local attention must account for data dimensionality, as 2D local attention consistently outperforms its 1D counterpart.

2. High-order SWA outperforms Block Attention in both 1D and 2D scenarios. This is likely due to the restrictive boundary effects imposed by Block Attention's non-overlapping partitions.

3. A 2D SWA hybrid achieves performance comparable to a full attention hybrid.

4. Both pure DeltaNet and pure 2D SWA underperform compared to their hybrid counterpart.

While vanilla high-order SWA performs well, it offers no significant speedup over full attention, despite the high sparsity. This limitation motivates our adoption of Sliding Tile Attention (STA Zhang et al. (2025a)), a hardware-efficient variant of high-order SWA.

3.3 Sliding the Window by Tiles Instead of Tokens

STA is a hardware-efficient variant of high-order SWA. Rather than shifting the window token-by-token, STA shifts the window tile-by-tile, with tokens within a tile sharing the same window. This way, by ensuring that the number of tokens within a tile equals the block size used in Flex Attention (Dong et al., 2024), no mixed blocks are generated, which enables tangible hardware speedups over full attention. [3].

We evaluate the effectiveness of STA in Table 7. Since the block size used in Flex Attention is required to be at least 128, we use a longer sequence length of 4096. The baseline is the full attention hybrid. From the table, we can conclude that:

1. As previously observed, dimensionality is critical, with 2D SWA variants significantly outperforming their 1D counterparts.

2. The STA hybrid achieves performance comparable to the full attention hybrid with faster speed. While 2D SWA ensures performance, it provides no notable speedup over full attention.

Therefore, for the integration of local attention in processing long sequences, STA stands out as both a fast and expressive local attention mechanism compared to vanilla high-order SWA and Block Attention. We note that our experiments are limited to moderately long sequences due to the computational cost of including a full attention baseline.

4 Efficient N-Dimensional Attention

4.1 Architecture Overview

We name our proposed architecture, a hybrid of linear recurrence and efficient high-order SWA, Efficient N-dimensional Attention (ENA). In ENA, the window size of SWA is configurable, with full attention being a special case corresponding to a large window. A simple illustration is shown in Fig. 1. We highlight several key aspects of ENA below:

1. Unlike many prior works that focus on domain-specific enhancements to linear models, such as introducing multi-scale processing, modifying gating mechanisms, or altering update rules, this work directly adopts the linear model implementations from FLA (Yang & Zhang, 2024), originally designed for language tasks. These domain-specific enhancements are orthogonal to ENA's architecture. Notably, these linear models, originally designed for language, already yield competitive results on vision tasks without domain-specific modifications.

[3]The official STA implementation primarily uses kernels written in ThunderKittens. For simplicity, we implement STA using Flex Attention while preserving the same high-level design.

2. We directly adopt this interleaved hybrid pattern without performing any search for optimal layer placements. The primary reason is that we treat the combination of linear and local attention as a single functional module. In general, two adjacent layers in ENA can be viewed as an unified big layer. In cases where a transformer requires both self-attention and cross-attention in a single layer for an application, for example text-to-video generation, ENA must use linear attention for the self-attention part and local attention for the cross-attention part. Thus, the configuration is fixed by design, and there is also no need to perform a search over layer placements.
3. We do not focus on scanning, as our experiments show its benefits are marginal compared to those of a hybrid architecture. Unless otherwise specified, we use uni-scan without any permutation to the sequence, even when using a causal linear model.
4. We treat full attention as a special case of SWA (with a large window) and use it for short sequences (<1K tokens) where local attention offers no efficiency benefits. As shown in Section 4.5.4, ENA with full attention (i.e., half linear recurrence and half attention) sometimes even outperforms the Transformer (which uses full attention throughout all the layers).
5. We primarily use DeltaNet as the linear recurrence module in ENA due to its strong empirical performance. Other types of linear model are discussed in previous section (e.g., HGRN, RetNet) and we aim to include more comprehensive discussion in future.
6. Throughout this paper, `ENA` denotes our proposed hybrid architecture. We specify the attention type, e.g., `ENA-DeltaNet-Full-Attention`, to distinguish between full attention and STA variants.

Table 8: Top-1 accuracy (%) on the K400 test set for hybrid DeltaNet models with sparse sequences. For hybrid architectures (denoted as "DeltaNet + ..."), odd-numbered layers use DeltaNet, while even-numbered layers employ the specified attention mechanism as token mixers. We evaluate models at input resolution 224 with patch sizes of 16 and 14, using 32 frames. Since the sequence length is long enough, ENA uses 3D STA, and the corresponding window size and tile size are provided in the table. The resulting sequence lengths are 6272 and 8192, respectively. All models are initialized from distilled image models and trained for 25 and 20 epochs.

Model / Configuration	Top-1 Acc (%)
Sequence Length = 6272	
Pure DeltaNet	69.74
ENA-DeltaNet-Full Attention	75.14
ENA-DeltaNet-STA (window size=32×6×6, tile size=32×2×2)	73.79
Sequence Length = 8192	
Pure STA (window size=24×12×12, tile size=8×4×4)	65.90
ENA-DeltaNet-Full Attention	72.28
ENA-DeltaNet-STA (window size=24×12×12, tile size=8×4×4)	72.63

While combining linear models with SWA is not new, as explored in Arora et al. (2024), ENA can be viewed as a natural extension of this idea, with the key distinction being the N-dimensional data. We additionally conduct several experiments to validate its performance.

4.2 ENA in 3D Understanding

To validate the architectural principles established in 2D image classification, we now evaluate ENA's performance on 3D video classification. For datasets, we use K400 (Kay et al., 2017). The video model is initialized using image classification models distilled from a ImageNet-1K finetuned version of SigLIP2. We evaluate two settings: sequence lengths of 6K and 8K, trained for 25 and 20 epochs, respectively. Given the long sequence length, ENA employs 3D STA in these experiments. The results are shown in Table 8, from which we can conclude that:

1. STA steadily improves pure linear recurrence even in 3D scenarios. As shown in the 6K and 8K settings, adding STA improves the performance notably.
2. For high-order data, STA must leverage locality across all dimensions to maintain high performance. In the 6K setting, we do not utilize the locality of the time dimension, thus resulting in worse performance than full attention hybrids. However, in the 8K setting, we utilize all three dimensions' locality, resulting in even better performance than the full attention hybrid.
3. The pure STA model in the 8K setting serves as an ablation for the effectiveness of linear recurrence. The inferior performance of the pure STA model confirms the necessity of the linear recurrence component in our hybrid architecture.

Table 9: Quantitative generative performance metrics of *ena-deltanet-xl-gen2d* on ImageNet 512 × 512 over 50k samples.

Iter	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
400k	4.7	4.7	170.1	0.83	0.58
600k	4.4	4.6	174.1	0.83	0.60
600k (mh2d-scan)	4.6	4.8	170.7	0.82	0.59

Table 10: Quantitative generative performance metrics for different models on ImageNet 512 × 512 over 50k samples. All models were trained for 400k iterations with a learning rate decaying from 1×10^{-3} to 1×10^{-4} after a 10k warmup period.

Models	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
SiT	5.88	4.98	150.51	0.84	0.56
ENA-Deltanet-Full Attention	5.02	5.16	165.71	0.82	0.61
ENA-DeltaNet-STA (window size=24×24, tile size=8×8)	4.87	4.81	165.21	0.83	0.58

4.3 ENA in 2D Generation

We train an image generation model using REPA (Yu et al., 2024). The model builds on the SiT (Ma et al., 2024) architecture, using DeltaNet as the linear model and STA as the local attention. We use half the batch size of the original REPA setting and train for 400K and 600K steps (equivalent to 200K and 300K steps in REPA's setting, respectively). We first train the model for 400K steps, then fine-tune it for an additional 200K steps using both uni-scan and multi-head-2d-scan to compare their effects. We use Muon as the optimizer and adopt a higher overall learning rate for faster convergence.

Since the maximum sequence length of REPA is 1024 when using a resolution of 512 × 512 (due to feature alignment constraints from pretrained encoders), we apply a window size of 24×24 in STA only, as a proof-of-concept. This leads to only 44% sparsity and is not optimal in terms of efficiency, as the minimum block size of Flex Attention should be 128, while we use 64 here. We train a model with roughly 366M parameters, referred to as *ena-deltanet-sta-w24x24-t8x8-xl-gen2d*.

Selected qualitative generation results are shown in Fig. 3, and quantitative metrics are provided in Table 9. We report Frechet Inception Distance (FID↓), sFID↓, Inception Score (IS↑), precision (Pre.↑), and recall (Rec.↑) using 50K generated samples. Lower values indicate better performance for FID and sFID, while higher values are preferred for IS, precision, and recall. We observe fast convergence at 400K steps (equivalent to 200K steps in REPA), reaching a FID of 4.7. To further evaluate multi-head scanning, we finetune the

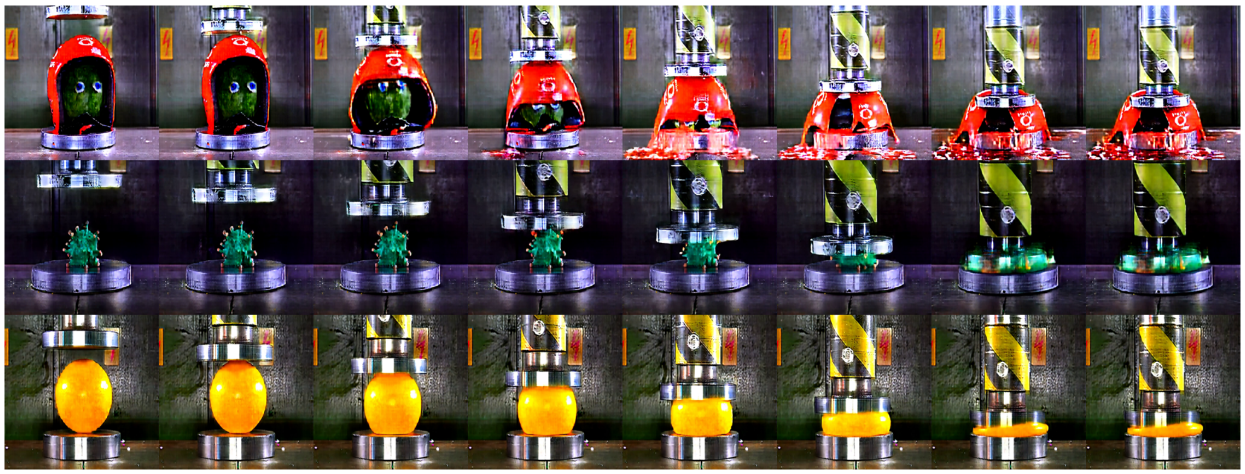

Figure 5: Video generation results on a toy dataset from *ena-deltanet-sta-w3x24x48-t1x8x16-cogvideox-2b*, demonstrating relatively consistent temporal behavior. Due to the limited training steps and simplified training settings, some artifacts and blurriness are observed in the spatial dimensions.

400K checkpoint (which uses uni-scan, i.e., no permutation) using both uni-scan and multi-head-2d-scan (denoted as mh2d-scan) for an additional 200K steps. For the mh2d-scan variant, we use a learning rate twice as high as the uni-scan counterpart. This result reinforces our earlier finding that scanning methods have a negligible impact on performance.

To compare against the Transformer and ENA with full attention, we conduct an additional experiment for 400K steps, using a learning rate that decays from 1×10^{-3} to 1×10^{-4} after a 10K-step warmup period, with the results shown in Table 10. From the table, we observe that ENA with full attention generally achieves comparable or even better performance than the Transformer. Meanwhile, ENA with 50%-sparsity STA also performs on par with ENA with full attention. These observations confirm our central finding: combining linear recurrence and attention is beneficial, and that further speedups can be obtained by controlling the attention sparsity level.

4.4 ENA in 3D Generation

We also briefly explore video generation using ENA. Specifically, we adapt the architecture introduced in CogVideo (Hong et al., 2022; Yang et al., 2024c), which consists of two attention modules within a single layer: a self-attention and a cross-attention. We replace the self-attention with the linear model used in ENA, and substitute the cross-attention with STA, which supports cross-modality attention mechanisms. Most weights are initialized from the pretrained CogVideoX-2B model, except for newly introduced components such as the short convolution used in the linear model. During training, we freeze all parameters in the channel mixers, updating only the token mixer, thereby minimizing training cost.

As a proof of concept, we train the model only on a toy dataset of 47 videos. We first adapt the pretrained transformer model to our target resolution of 1024×768. We then use this adapted transformer as a teacher model. Its noise predictions provide direct supervision for fine-tuning the ENA model. We employ DeltaNet as the linear model and STA with a window size of $3 \times 24 \times 48$ and a tile size of $1 \times 8 \times 16$. Throughout training, we use Muon as the optimizer.

The generation results are shown in Fig. 5, exhibiting relatively good temporal consistency. Due to limited computational resources and the high training cost associated with video generation, the model is trained for only around 20K steps, resulting in visible artifacts in individual frames. We leave further scaling and optimization of training settings as future work.

4.5 Discussions

4.5.1 Impact of Learning Rate

While our main experiments use a learning rate of 2e-3, some gated linear models like Gated DeltaNet (GDN) and Gated DeltaProduct (GDP) are known to prefer smaller learning rates. We conducted experiments in Table 11 to verify that ENA's performance gains hold across different learning rates. Since training with a short sequence length, ENA here employs full attention. From the table, we can observe that:

1. Smaller learning rates indeed improve the performance of gated linear models.
2. ENA consistently outperforms pure linear models regardless of the learning rate.
3. ENA performs better using a larger learning rate than the optimal learning rate for gated linear models, consistent with our default settings.

4.5.2 Impact of Optimizer

We use AdamW as the default optimizer. To demonstrate that ENA's benefits are not optimizer-specific, we replaced AdamW with Muon (Jordan et al., 2024) as the optimizer and test the performance in 4K sequence length training with STA. We use a larger learning rate of 4e-3 for Muon. Since training with a long sequence length, ENA here uses 2D STA. From the results shown in Table 12, we can observe that ENA (the hybrid) performs notably better than the pure DeltaNet model, thus demonstrating that the benefits of ENA are not specific to a single optimizer.

Table 11: The influence of learning rate. Gated linear models perform better in a small learning rate. However, hybrid model (ENA) consistently improve the performance. We use a 12-layer model with a hidden size of 448, resulting in a 30M parameters model. Since training with a sequence length of 196, ENA here uses full attention.

Model / Configuration	Learning Rate	Top-1 Acc (%)
Sequence Length = 196, 20 Epochs, Warmup Epochs = 20		
Pure GDP	2e-3	36.34
ENA-GDP-Full Attention	2e-3	55.25
Pure GDP	5e-4	47.29
ENA-GDP-Full Attention	5e-4	49.91
Pure GDN	2e-3	34.92
ENA-GDN-Full Attention	2e-3	47.48
Pure GDN	2e-4	34.62
ENA-GDN-Full Attention	2e-4	39.07
Sequence Length = 4096, 10 Epochs, Warmup Epochs = 2		
Pure GDP	5e-4	38.06
ENA-GDP-STA (window size=48x24, tile size=16x8)	5e-4	39.76

Table 12: Performance using Muon as the optimizer. It can be shown that hybrid model (ENA) consistently improves the performance. We use a 12-layer model with a hidden size of 448, resulting in a 30M parameters model. Since training with a sequence length of 4096, ENA here uses STA.

Model / Configuration	Top-1 Acc (%)
Pure DeltaNet	68.95
ENA-DeltaNet-STA (window size=48x24, tile size=16×8)	74.45

Table 13: Faster training results on ImageNet using Muon. The resolution is 224 with a patch size of 16, resulting in a sequence length of 196. Since training with a short sequence length, ENA here uses full attention.

Model / Configuration	Top-1 Acc (%)	Epoch	Warmup
ENA-DeltaNet-Full Attention	74.27	20	4
ENA-DeltaNet-Full Attention	77.08	30	5

4.5.3 Training ENA Faster with Muon

We also find that using Muon results in faster convergence compared to AdamW. We train the tiny-size model (30M parameters) using Muon and a larger learning rate, with a cosine learning rate scheduler. The results are shown in Table 13, from which we can observe that using Muon can accelerate the convergence speed. This approach achieves performance comparable to 100-epoch training with AdamW in a fraction of the time, making it resource-efficient for future architectural evaluations. Given the observation, we also use Muon as the optimizer for image generation and video generation.

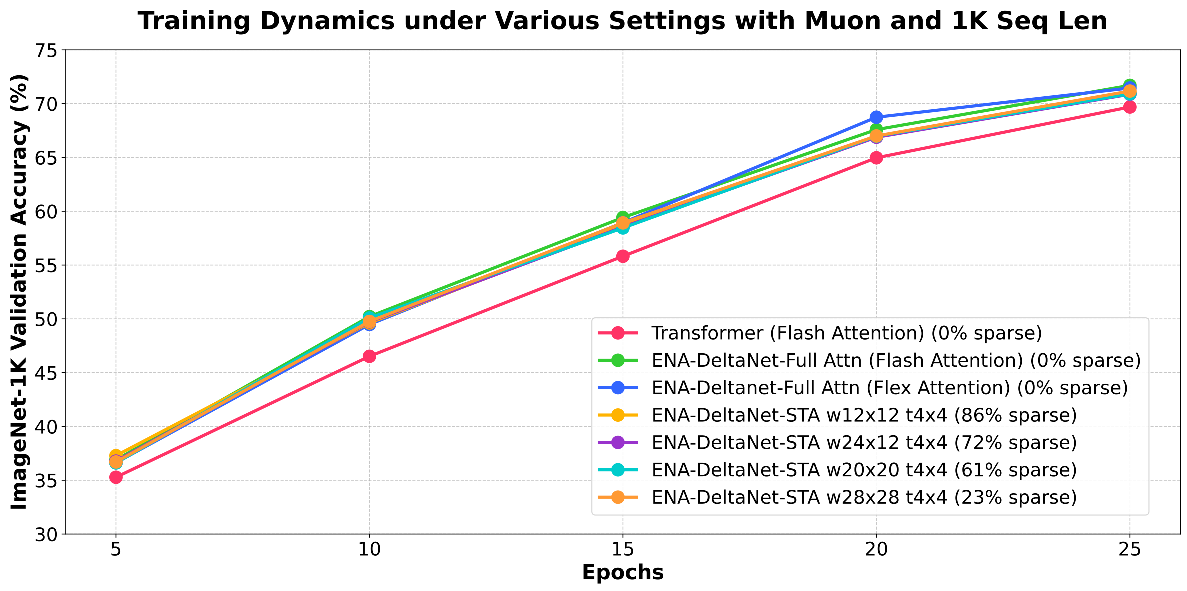

Figure 6: Training dynamics under various settings on ImageNet-1K with Muon and a sequence length of 1024. The hybrid architecture of ENA outperforms the Transformer, and the performance gain from decreasing sparsity levels (i.e., increasing window size) gradually diminishes, indicating diminishing returns. For STA, we use the notation `w12×12` and `t4×4` to denote a window size of 12×12 and a tile size of 4×4.

4.5.4 Impact of Sparsity Levels

The attention in ENA can have varying sparsity levels, determined by the window size used in local attention. Different levels of sparsity affect both hardware efficiency and model performance. To investigate this, we first conduct a set of experiments on ImageNet-1K pretraining with a sequence length of 1K as a proof of concept. The image size is set to 128 and the patch size to 4. We use both Muon and AdamW optimizers and present the results in Fig. 6 and Fig. 7, respectively. The batch size is 128×8, with learning rates of 1e$-$2 for Muon and 2e$-$3 for AdamW. All models are trained for 25 epochs with 5 epochs of linear warmup. The sparsity level in STA is controlled by the window size with larger windows correspond to lower sparsity. The tile size is fixed at 4×4.

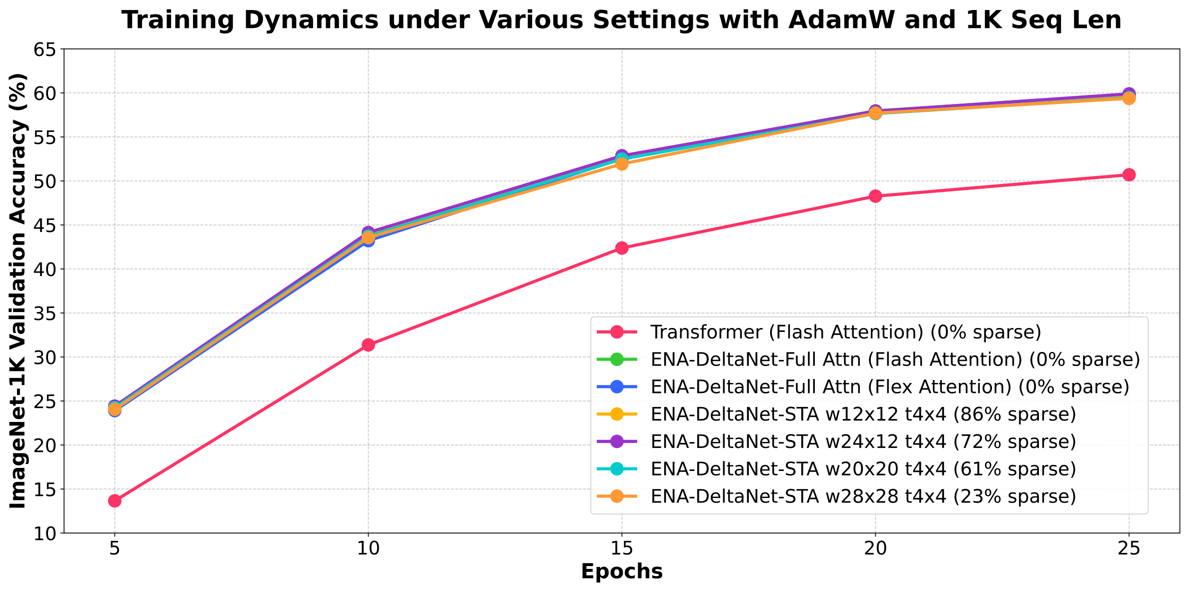

Figure 7: Training dynamics under various settings on ImageNet-1K with AdamW and a sequence length of 1024. Similar conclusions can be drawn from Fig. 6. Notably, the performance gap between the Transformer and hybrid models is even larger in this case. For STA, we use the notation `w12×12` and `t4×4` to denote a window size of 12×12 and a tile size of 4×4.

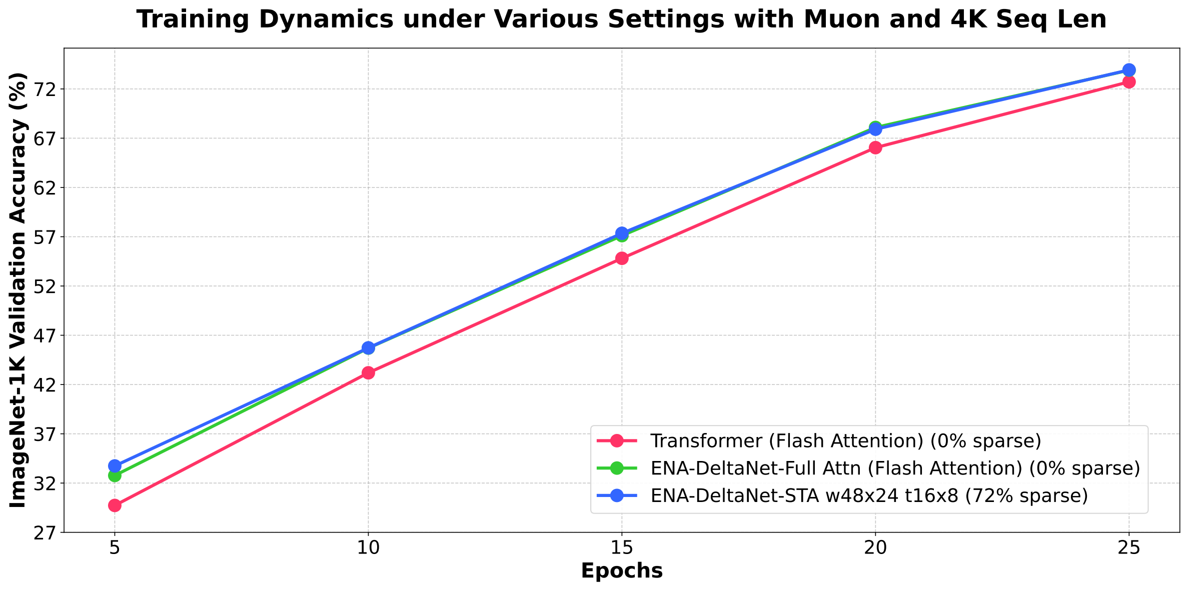

Figure 8: Training dynamics under various settings on ImageNet-1K with Muon and a sequence length of 4096. Similar conclusions can be drawn from Fig. 6 and Fig. 7. Notably, with approximately 70% sparsity, ENA with STA even outperforms ENA using full attention. For STA, we use the notation `w48×24` and `t16×8` to denote a window size of 48×24 and a tile size of 16×8.

From the results, we observe that increasing the window size (i.e., reducing sparsity) yields diminishing performance improvements. For example, under the AdamW setting, a window size of 24×12 (approximately 72% sparsity) achieves accuracy comparable to full attention (0% sparsity). Further increasing the window size to 28×28 does not lead to noticeable performance gains. A similar trend is observed when using Muon,

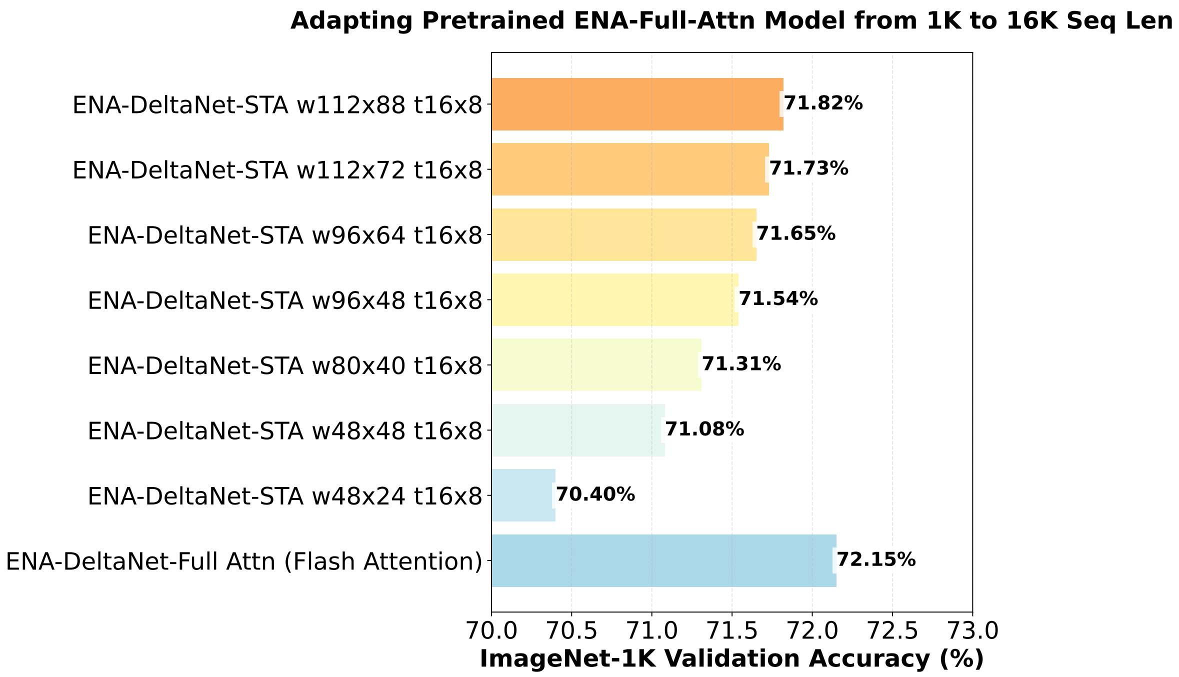

Figure 9: ImageNet-1K validation accuracy at a sequence length of 16,384, obtained by finetuning ENA models pretrained with a sequence length of 1K using full attention under various settings. Reducing sparsity by increasing the window size leads to diminishing performance gains, with a sparsity level of approximately 70% achieving performance comparable to full attention.

where a window size of 12×12 already matches the performance of full attention, and larger windows incur additional computation without meaningful improvements.

We present the 4K sequence length training results using AdamW in Table 7, where ENA with a window size of 48×24 for STA (corresponding to 72% sparsity) matches the performance of ENA with full attention. To support a more robust conclusion, we also report 4K training results using Muon in Fig. 8, which confirm the same observation.

To evaluate the generalization of this trend in longer sequences, we finetune the ENA model pretrained with full attention (from Fig. 6) on ImageNet-1K at a sequence length of 16,384. The image size is set to 512 and the patch size remains 4. All other parameters are kept the same, except for a randomly initialized learnable positional embedding. We use a constant learning rate of 1e$-$4 and train for 2 epochs. As shown in Fig. 9, the same observation holds: increasing the window size leads to diminishing gains, with a sparsity level around 70% once again achieves performance comparable to full attention. It is important to note that this observation can also be found in Table 7, where ENA with 70% sparse STA yields performance comparable to ENA with full attention.

Overall, our findings on the impact of sparsity levels in STA can be summarized as follows:

1. The strong locality inherent in many datasets means that full attention often performs redundant computations on distant, irrelevant tokens. Local attention leverages this redundancy to improve efficiency while maintaining comparable performance.

2. Reducing sparsity by increasing the window size yields diminishing, and even negligible, performance gains.

3. To maintain a consistent trade-off between performance and efficiency across different sequence lengths, fixing the sparsity level is more effective than fixing the window size. As shown in Fig. 9,

overly high sparsity levels can degrade performance. Based on our experiments, a sparsity level of around 70% (i.e., each token attends to only 30% of the sequence) offers a good trade-off.

4.6 Why a Hybrid Model over Transformer?

This paper mainly introduces a hybrid architecture that combines linear recurrence with high-order SWA, where we count full attention as a special case (in which the window size is large enough to cover all tokens). A natural question is: why prefer a hybrid model over a standard Transformer? We offer several reasons, as partially demonstrated by previous experiments.

1. An ENA model with full attention already replaces half of the layers in Transformer with linear recurrence, which has linear time complexity and is much faster when modeling long sequences. This setup already performs comparably to or sometimes better than Transformer, as shown in Table 3, Table 10, Fig. 6, and Fig. 7. A more comprehensive evaluation and explanation remains as future work.
2. Furthermore, the full attention component can be replaced with high-order SWA to achieve greater speedups with minimal performance loss, as shown in previous sections. By carefully controlling the sparsity level in SWA, we can achieve similar performance than using full attention.

To conclude, simply combining half linear recurrence and half full attention yields comparable performance to Transformer, and the full attention part can be further accelerated using high-order SWA with a high sparsity level, without notable performance degradation.

4.7 Distillation from Pretrained Models

Distilling from pretrained models is a resource-efficient alternative to training from scratch. To evaluate ENA's effectiveness as a student model, we distill knowledge from SigLIP2-Base (Tschannen et al., 2025). **For our base-sized model, we initialize the majority of weights directly from the teacher model.** For smaller models, we randomly initialize the weights. We train on ImageNet-1K using temperature-scaled KL divergence between student and teacher outputs for 200 epochs, followed by 10 epochs of supervised finetuning. The results of several models on ImageNet-1K are shown in Table. 14. For naming consistency, we adopt the template `ena-{linear model type}-{attention type}-{size}-{task}-{teacher info}` throughout this paper.

Table 14: Performance of distilled model on ImageNet-1K validation set.

Models	Top-1 Acc (%)
ena-deltanet-fullattn-base-imgc-siglip2-base-p16-224	85.15
ena-lact-fullattn-base-imgc-siglip2-base-p16-224	84.95
ena-hgrn-fullattn-base-imgc-siglip2-base-p16-224	84.16

4.8 Hardware Efficiency

We compare the hardware efficiency of ENA against a Flash Attention (FA) based Transformer. Both ENA and FA models use a 12-layer encoder with a hidden size of 224, and we report the results in Fig. 2. For linear recurrence, we use DeltaNet implemented in FLA (Yang & Zhang, 2024) with uni-directional scan. We use full attention for short sequences and switch to Flex Attention implementation of STA when the sequence length exceeds 1152. During training, we apply an MSE loss on the hidden states and use a window size that covers 30% of the tokens for STA. From the figure, we observe that:

1. ENA's training and inference times scale more favorably than FA's, offering notable speedups for sequence lengths in the thousands.

2. Although ENA's memory consumption is slightly higher than FA's, the difference is minor and can potentially be reduced through kernel fusion. It is also worth noting that both the linear model and STA in our implementation are built using Triton, while FA (specifically, FA2) is implemented directly in CUDA with extensive optimizations for A100 GPUs. As demonstrated in the original STA paper, further efficiency gains may be achieved by developing specialized kernels using CUDA or ThunderKittens (Spector et al., 2024). This remains a promising direction for future work.

Table 15: Arithmetic Intensity comparison across models and sequence lengths. The tests are performed on an NVIDIA A100 40GB SXM GPU with bf16 precision. The window sizes for STA are 48×24, 64×64, and 80×80, with the tile size being 16×8.

Model / SeqLen	**256**	**1024**	**4096**	**16384**	**25600**
Transformer	122.67	146.98	307.93	825.51	1240.40
DeltaNet-uni-scan	91.14	85.46	96.32	88.12	82.31
DeltaNet-bi-scan	70.92	70.02	79.08	70.69	65.06
DeltaNet-cross-scan	58.20	55.60	63.49	54.15	48.24
ENA-DeltaNet-Full Attention	105.33	110.62	186.93	284.30	506.65
ENA-DeltaNet-STA	–	–	127.79	166.99	193.12

In addition to direct speed and GPU memory measurement, we also provide arithmetic intensity (AI) of our studied models in Table 15. **As a reference, the boundary arithmetic intensity (AI) between memory-bound and computation-bound on an A100 is roughly 200.** From the results, we can observe that:

1. Pure linear recurrent models have low AI across various sequence lengths, which indicates their low utilization of the hardware. Multi-pass scanning not only slows down speed and increases GPU memory usage, but also has a lower AI, which further proves their hardware inefficiency.
2. Transformers using flash attention exhibit a sharp increase in AI with longer sequences, which indicates the compute-heavy nature of full attention due to the large number of FLOPs required.
3. ENA models strike a balance by significantly improving AI compared to pure linear recurrent models while keeping the overall FLOPs small, thus achieving both fast speed and high hardware utilization.

In conclusion, ENA provides a compelling alternative to Transformers for long-sequence modeling, achieving notable speedups with comparable memory usage and a high level of hardware utilization.

5 Related Works

Linear recurrent models are token mixers with states and linear time complexity. Representative ones include RetNet (Sun et al., 2023), HGRN (Qin et al., 2024), GLA (Yang et al., 2024a), GSA (Zhang et al., 2024), RWKV (Peng et al., 2023), DeltaNet (Yang et al., 2024b), Gated DeltaNet (Yang et al., 2025), RWKV-7 (Peng et al., 2025), LaCT (Zhang et al., 2025b), and MesaNet (von Oswald et al., 2025). Other sub-quadratic token mixers, such as Log-Linear Attention (Guo et al., 2025) have also been proposed to balance efficiency and expressiveness. Although ENA is built on linear recurrence, its hybrid framework is potentially compatible with any sub-quadratic token mixer.

Prior works have also utilized local attention with linear recurrence, including MambaVision (Hatamizadeh & Kautz, 2024), TTT-MLP (Dalal et al., 2025), and LaCT (Zhang et al., 2025b). The local attention in these works typically consists of non-overlapping blocks, denoted in this paper as *Block Attention*. Block Attention is simple to implement using Flash Attention but imposes strong restrictions on border tokens.

High-order SWA, introduced in Neighborhood Attention (Hassani et al., 2023) extends SWA from language modeling and, unlike Block Attention, imposes no border restrictions. However, naive implementation of high-order SWA generates many mixed blocks, which prevents actual speedup despite sparsity and reduced FLOPs.

Sliding Tile Attention (STA Zhang et al. (2025a)) addresses this by sliding the window by tiles, where a number of tokens share the same window. In this way, by ensuring that the number of tokens within a tile equals the block size used in Flex Attention (Dong et al., 2024), no mixed blocks are generated, thus achieving tangible hardware speedups.

6 Conclusion

In this paper, we evaluated two primary strategies for adapting linear recurrent models to high-order data: scanning and attention integration. Our findings show that while scanning offers negligible benefits, integrating efficient high-order SWA yields notable performance gains. Finally, we propose ENA, a simple hybrid of linear recurrent models and high-order SWA, achieving performance comparable to Transformers. ENA is a general architecture applicable to any linear model, and in this work, we primarily demonstrate its effectiveness using DeltaNet.

References

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

Karan Dalal, Daniel Koceja, Gashon Hussein, Jiarui Xu, Yue Zhao, Youjin Song, Shihao Han, Ka Chun Cheung, Jan Kautz, Carlos Guestrin, et al. One-minute video generation with test-time training. *arXiv preprint arXiv:2504.05298*, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2024.

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

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention, 2025. URL `https://arxiv.org/abs/2506.04761`.

Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6185–6194, 2023.

Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv preprint arXiv:2407.08083*, 2024.

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017. URL `https://arxiv.org/abs/1705.06950`.

Songhua Liu, Zhenxiong Tan, and Xinchao Wang. Clear: Conv-like linearization revs pre-trained diffusion transformers up. *arXiv preprint arXiv:2412.16112*, 2024.

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*, pp. 10012–10022, 2021.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. In *Proceedings of COLM*, 2024.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear rnns via householder products, 2025. URL `https://arxiv.org/abs/2502.10297`.

Benjamin F. Spector, Simran Arora, Aaryan Singhal, Daniel Y. Fu, and Christopher Ré. Thunderkittens: Simple, fast, and adorable ai kernels, 2024. URL `https://arxiv.org/abs/2410.20399`.

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

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. Mesanet: Sequence modeling by locally optimal test-time training, 2025. URL `https://arxiv.org/abs/2506.05233`.

Feng Wang, Timing Yang, Yaodong Yu, Sucheng Ren, Guoyizhe Wei, Angtian Wang, Wei Shao, Yuyin Zhou, Alan Yuille, and Cihang Xie. Causal image modeling for efficient visual understanding. *arXiv preprint arXiv:2410.07599*, 2024.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL `https://github.com/fla-org/flash-linear-attention`.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of ICML*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proceedings of NeurIPS*, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *Proceedings of ICLR*, 2025.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024c.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhenghong Liu, and Hao Zhang. Fast video generation with sliding tile attention. *arXiv preprint arXiv:2502.04507*, 2025a.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025b.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. In *Proceedings of NeurIPS*, 2024.

Lianghui Zhu, Zilong Huang, Bencheng Liao, Jun Hao Liew, Hanshu Yan, Jiashi Feng, and Xinggang Wang. Dig: Scalable and efficient diffusion models with gated linear attention. *arXiv preprint arXiv:2405.18428*, 2024.