

DepthSSD: Rethinking Residual Connections via State Space Models on the Depth Axis

Zhengqing Yuan¹, Xiaoyu Ma^{*}, Yunhong He[†], Zhengtao Yao², Lichao Sun³, Yanfang Ye^{1,‡}

¹University of Notre Dame

²University of Southern California

³Lehigh University

Abstract

Residual connections are fundamental to deep transformer training, yet their theoretical relationship to more expressive alternatives, such as attention-based depth mixing (AttnRes), remains poorly understood. We show that the depth-axis mixing matrix in transformers is *semiseparable*, and its rank precisely controls the expressiveness of inter-layer information flow. Standard residual connections correspond to rank-1 semiseparable matrices; attention residuals to full rank. We introduce DEPTHSSD (Depth State Space Duality), which applies the structured state space duality framework from Mamba-2 to the *depth axis*, yielding a depth mixing matrix of controllable rank N with $O(L \cdot N)$ parameters per sub-layer.

We evaluate DEPTHSSD at three model scales (125M, 455M, and 1B parameters) against four depth-mixing baselines: standard residuals, BLOCKATTNRES, DENSEFORMER, and Hyper-Connections. At 1B, DEPTHSSD- $N=8$ achieves the best validation loss (3.124, perplexity 22.7), a 2.4-point perplexity reduction over standard residuals (PPL 25.1) and outperforming DENSEFORMER (PPL 24.1), while running $1.72\times$ faster (16.5 vs. 9.6 TFLOPS). At 455M, DEPTHSSD- $N=8$ again leads (PPL 29.4 vs. 30.1 for DENSEFORMER and 31.2 for standard) at $1.88\times$ the throughput of DENSEFORMER. At 125M, DEPTHSSD matches the validation quality of all baselines while maintaining $1.3\text{--}1.6\times$ higher throughput than the other depth-mixing methods.

1 Introduction

Deep transformers (Vaswani et al., 2017; Brown et al., 2020; Chowdhery et al., 2022; Dosovitskiy et al., 2021; Radford et al., 2021; Ye et al., 2025) have become the dominant architecture across language, vision, and multimodal learning. At the heart of their trainability lies the residual connection (He et al., 2016; Xiong et al., 2020; Wang et al., 2022; Bachlechner et al., 2021): by computing $\mathbf{y}_l = \mathbf{x}_l + f_l(\mathbf{x}_l)$ at every layer, it provides an additive shortcut that stabilizes optimization and supports training at depth. As models grow deeper, modern large language models routinely exceed 80 layers (Brown et al., 2020; Chowdhery et al., 2022), how information is propagated along the depth axis becomes increasingly important. In standard transformers, the residual stream is the primary pathway through which early-layer representations influence later layers and, ultimately, the output.

Standard residuals, however, provide only a restricted form of cross-layer communication. At layer l , the model operates on the current representation $\mathbf{x}_l$ without a mechanism for selectively retrieving representations from layers $0, 1, \dots, l-2$. Information from early layers is therefore propagated only implicitly through the accumulated residual stream, where it is repeatedly mixed and transformed by intermediate layers and may gradually attenuate or obscure useful early-layer features as depth increases (Wang et al., 2022; Xiong et al.,

^{*}Yunhong He is an independent researcher working remotely with Lichao Sun.

[†]Xiaoyu Ma is an independent researcher working remotely with Yanfang Ye.

[‡]Corresponding Author: yye7@nd.edu

2020). Residual connections therefore provide an excellent optimization pathway, but a rigid interface for depth-wise information routing. The question, then, is whether this routing can be made more expressive *without* incurring proportional computational cost, so that a fixed model depth can support richer representations and make better use of the parameter budget.

Building on this need, several recent methods revisit the fixed additive residual through learned depth-axis mixing. Some methods increase expressivity by preserving access to the full depth history. Kimi Team et al. (2026) introduce Attention Residuals (ATTNRES), which apply softmax attention across the depth axis so that every layer can aggregate information from all preceding layers, achieving consistent quality gains from 70M to 1B parameters. Pagliardini et al. (2024) propose DenseFormer, which maintains depth-weighted averages over all preceding sub-layer outputs. Zhu et al. (2025) introduce Hyper-Connections, maintaining multiple parallel residual streams with learned linear mixing. While each method demonstrates measurable improvements over standard residuals, they also share a common limitation: their computational or memory overhead grows rapidly with model scale. ATTNRES incurs $O(L^2 \cdot d)$ complexity for L layers with hidden dimension d , and both BLOCKATTNRES and DENSEFORMER require progressively smaller batch sizes as model size increases due to memory pressure. These developments suggest that the central issue is not residual addition per se, but how information should be mixed along the depth axis. Concurrently, Zhang (2026) observe a duality between the depth axis and the sequence axis in transformers, drawing connections to state space models, but a complete algorithmic unification has remained elusive.

These observations reveal three interrelated gaps. **1) Lack of theoretical unification:** existing residual and depth-mixing methods, including standard residuals, ATTNRES, DENSEFORMER, and Hyper-Connections, are motivated by separate intuitions and analyzed with distinct tools, with no shared framework that clarifies their relative expressiveness or reveals intermediate design points. **2) Lack of scalable depth mixing:** methods that improve quality at small scale often introduce overhead that scales poorly with model size, making them increasingly impractical at moderate and larger scales under fixed training budgets. **3) Lack of a principled efficiency–expressiveness control:** existing methods do not expose a single, interpretable parameter that smoothly trades depth-mixing capacity against cost, making it difficult to navigate the space between restricted residual mixing and full-rank attention. Without addressing these gaps, transformer depth remains an under-exploited axis of model design, potentially leaving significant representational capacity on the table.

To address these gaps, we introduce DEPTHSSD (Depth State Space Duality), which applies the structured state space duality framework of Mamba-2 (Dao & Gu, 2024) to the *depth axis* of transformers, yielding a depth mixing mechanism of controllable semiseparable rank N with $O(L \cdot N \cdot d)$ parameters per sub-layer. Our contributions are as follows: **1) We prove that the depth-axis mixing matrix $M \in \mathbb{R}^{L \times L}$ is semiseparable of rank N when parameterized as a state space model along the depth axis** (Section 3). This single framework subsumes standard residuals ($N=1$), Hyper-Connections ($N=m$), and ATTNRES ($N=L$) as special cases, resolving the lack of theoretical unification (gap 1). **2) We design the DEPTHSSD algorithm, which achieves $O(L \cdot N \cdot d)$ parameter cost per sub-layer with N typically 4–16, and maintains constant batch size across all tested scales** (Section 4), directly addressing the scalability bottleneck of existing methods (gap 2). **3) We validate DEPTHSSD at 125M, 455M, and 1B parameters against four baselines.** At 1B, DEPTHSSD- $N=16$ is the only depth-mixing method to outperform the standard baseline (4.086 vs. 4.272), while all competitors degrade. The semiseparable rank N exhibits a log-linear relationship with quality for $N \leq 16$, providing practitioners with an interpretable, single-parameter knob for the efficiency–expressiveness tradeoff (gap 3) (Section 5).

2 Background

2.1 Residual Connections and Depth Mixing

A standard transformer with L layers computes:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + f_l(\mathbf{x}_l), \quad l = 0, 1, \dots, L-1 \quad (1)$$

where f_l denotes the l -th sub-layer (self-attention or MLP). This can be written in matrix form as $\mathbf{x}_L = \mathbf{M} \cdot [\mathbf{x}_0, f_0(\mathbf{x}_0), \dots, f_{L-1}(\mathbf{x}_{L-1})]^\top$, where the mixing matrix $\mathbf{M}$ is lower-triangular with all ones, a rank-1 semiseparable matrix.

Kimi Team et al. (2026) replace this fixed mixing with learned attention:

$$\mathbf{h}_l = \sum_{j \leq l} \alpha_{l,j} \mathbf{b}_j, \quad \alpha_{l,j} = \text{softmax}_j(\mathbf{q}_l^\top \mathbf{k}_j / \sqrt{d}) \quad (2)$$

where $\{\mathbf{b}_j\}$ are block representations and $\mathbf{q}_l, \mathbf{k}_j$ are learned projections of the depth representations. The block variant groups sub-layers into blocks of size B and applies attention only across block boundaries, reducing complexity from $O(L^2)$ to $O(L^2/B^2)$.

2.2 State Space Models and Semiseparable Matrices

A discrete-time state space model (SSM) with state dimension N computes (Gu et al., 2022; Gu & Dao, 2023; Dao & Gu, 2024):

$$\mathbf{s}_l = \mathbf{A}_l \mathbf{s}_{l-1} + \mathbf{b}_l x_l \quad (3)$$

$$y_l = \mathbf{c}_l^\top \mathbf{s}_l + D_l x_l \quad (4)$$

where $\mathbf{A}_l \in \mathbb{R}^{N \times N}$ (diagonal), $\mathbf{b}_l, \mathbf{c}_l \in \mathbb{R}^N$, and $D_l \in \mathbb{R}$.

Dao & Gu (2024) show that the input-output map of this recurrence is a *semiseparable matrix*:

$$M_{ij} = \begin{cases} \mathbf{c}_i^\top \left(\prod_{k=j+1}^i \mathbf{A}_k \right) \mathbf{b}_j + D_i \delta_{ij} & i \geq j \\ 0 & i < j \end{cases} \quad (5)$$

The key property is that every submatrix of the strictly lower-triangular part of $\mathbf{M}$ has rank at most N . This is precisely the *semiseparable rank*.

3 Depth State Space Duality

3.1 The Depth Mixing Matrix is Semiseparable

We now formalize the connection between depth-axis mixing and semiseparable matrices. Consider a transformer with L sub-layers. At each sub-layer l , the input representation $\mathbf{h}_l \in \mathbb{R}^d$ is computed as a function of all previous representations $\{\mathbf{x}_0, \dots, \mathbf{x}_l\}$ through a mixing mechanism, followed by the sub-layer function.

Theorem 1 (Depth Mixing as Semiseparable Matrix). *Let the depth mixing at layer l be defined by a state space recurrence (3)–(4) with state dimension N and diagonal state transition $\mathbf{A}_l = \text{diag}(a_{l,1}, \dots, a_{l,N})$. Then for each hidden dimension d , the mixing matrix $\mathbf{M}^{(d)} \in \mathbb{R}^{L \times L}$ defined by $h_l^{(d)} = \sum_{j=0}^l M_{lj}^{(d)} x_j^{(d)}$ is lower-triangular semiseparable of rank at most N .*

For $i \geq j$, the (i, j) entry is:

$$M_{ij}^{(d)} = \sum_{n=1}^N c_i^{(n,d)} \left(\prod_{k=j+1}^i a_k^{(n)} \right) b_j^{(n,d)} + D_i \delta_{ij}$$

The strictly lower-triangular part (excluding the diagonal $D_i \delta_{ij}$ term) can be written as $\mathbf{L} = \mathbf{C} \mathbf{P} \mathbf{B}^\top$, where $\mathbf{C} \in \mathbb{R}^{L \times N}$, $\mathbf{P}$ encodes the cumulative products, and $\mathbf{B} \in \mathbb{R}^{L \times N}$. Any submatrix of $\mathbf{L}_{i_1:i_2, j_1:j_2}$ (with $i_1 > j_2$) has rank at most N since it factors through an N -dimensional space.

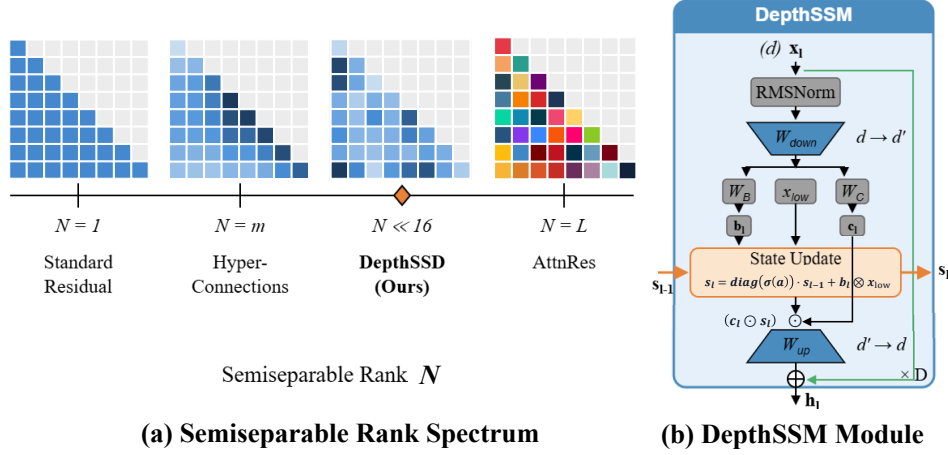


Figure 1: **(a)** The semiseparable rank N provides a single parameter that continuously interpolates between standard residuals ($N = 1$) and full-rank depth attention ($N = L$). **(b)** The DepthSSM module projects to a low-dimensional state space, applies a depth-axis SSM recurrence, and adds a learnable skip connection.

Corollary 2 (Rank Continuum). *The semiseparable rank N provides a continuous interpolation:*

- $N=1$: The mixing matrix has rank-1 off-diagonal blocks, equivalent to a scalar-weighted cumulative sum (standard residual with learned weighting).
- $N=L$: The mixing matrix can represent any lower-triangular matrix, achieving full-rank capacity equivalent to ATTNRES.
- $1 < N < L$: The mixing matrix has bounded rank N , providing a controlled tradeoff between expressiveness and efficiency.

3.2 Relationship to Existing Methods

Standard residuals ($N=1, A=1, B=1, C=1, D=0$). Setting $N=1$ with $A=1, B=1, C=1, D=0$ yields $M_{ij} = 1$ for all $i \geq j$, recovering the cumulative residual stream $\mathbf{h}_l = \sum_{j=0}^l \mathbf{x}_j$. Setting $D=1$ and $B=0$ gives $M_{ij} = \delta_{ij}$, the pure skip connection.

Hyper-Connections ($N=m$). Zhu et al. (2025) maintain m parallel residual streams, mixing them through learned linear transformations at each layer. This is equivalent to a depth SSM with $N=m$ and specific parameterization of $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

AttnRes (unconstrained rank). ATTNRES uses softmax attention to compute arbitrary mixing weights α_{ij} with no rank constraint on the resulting matrix. It achieves full expressiveness but at $O(L^2)$ cost in the number of layers.

DEPTHSSD occupies the sweet spot: it achieves near-full-rank expressiveness with $N \ll L$ while adding only $O(L \cdot N \cdot d)$ parameters and $O(L \cdot N)$ computation per forward pass.

4 Method: DEPTHSSD

4.1 Architecture

DEPTHSSD generalizes the fixed additive residual pathway before each transformer sub-layer using a learned depth-axis SSM. We apply two SSM instances per layer: one before self-attention and one before the MLP. For clarity, we omit batch and sequence dimensions in the equations below; all operations are applied independently per token position.

For a single token representation $\mathbf{x}_l \in \mathbb{R}^d$, the DEPTHSSM module first applies RMSNorm and projects to a lower dimension $d' \ll d$ via $\mathbf{x}_{\text{low}} = W_{\text{down}} \text{RMSNorm}(\mathbf{x}_l) \in \mathbb{R}^{d'}$. Input-dependent SSM parameters $\mathbf{b}_l = W_B \mathbf{x}_{\text{low}}$ and $\mathbf{c}_l = W_C \mathbf{x}_{\text{low}}$ (both in $\mathbb{R}^N$) are then computed via learned linear maps. The module maintains a recurrent state $\mathbf{s}_l \in \mathbb{R}^{N \times d'}$ and computes:

$$\mathbf{s}_l = \text{diag}(\sigma(\mathbf{a})) \cdot \mathbf{s}_{l-1} + \mathbf{b}_l \otimes \mathbf{x}_{\text{low}} \quad (6)$$

$$\mathbf{h}_l = W_{\text{up}} \left(\sum_{n=1}^N \mathbf{c}_{l,n} \odot \mathbf{s}_{l,n} \right) + D \cdot \mathbf{x}_l \quad (7)$$

where σ is the sigmoid function, $\mathbf{a} \in \mathbb{R}^N$ are learnable decay logits, $W_{\text{up}} \in \mathbb{R}^{d \times d'}$ projects the SSM output back to the full hidden dimension, and $D \in \mathbb{R}$ is a learnable skip weight. We initialize the decay, skip, and B/C projection parameters so that $\mathbf{h}_l \approx \mathbf{x}_l$ at the start of training, recovering the standard residual; full details are given in Appendix A.1.

The full transformer block applies DepthSSM symmetrically before both sub-layers:

$$\begin{aligned} \mathbf{h}_{\text{attn}}, \mathbf{s}_{\text{attn}} &= \text{DepthSSM}_{\text{attn}}(\mathbf{x}_l, \mathbf{s}_{\text{attn}}) \\ \mathbf{x}'_l &= \mathbf{h}_{\text{attn}} + \text{Attn}(\text{RMSNorm}(\mathbf{h}_{\text{attn}})) \\ \mathbf{h}_{\text{mlp}}, \mathbf{s}_{\text{mlp}} &= \text{DepthSSM}_{\text{mlp}}(\mathbf{x}'_l, \mathbf{s}_{\text{mlp}}) \\ \mathbf{x}_{l+1} &= \mathbf{h}_{\text{mlp}} + \text{MLP}(\text{RMSNorm}(\mathbf{h}_{\text{mlp}})) \end{aligned} \quad (8)$$

4.2 State Dimension Projection

A naive implementation stores the SSM state in $\mathbb{R}^{B \times T \times N \times d}$ (restoring batch B and sequence T dimensions). For $N=16, d=768, T=1024, B=32$, this amounts to 768 MB in BF16 per SSM, substantial overhead. The down/up projections ($W_{\text{down}}, W_{\text{up}}$) confine the state to $\mathbb{R}^{B \times T \times N \times d'}$ with $d'=64$, reducing memory by $12\times$ while ensuring the resulting depth mixing still acts on the full d -dimensional representation through the down/up projection pair.

This projection is not merely an efficiency optimization; it is also important for stable training in practice. Without it ($d'=d$), training fails to converge in our ablations (Section 5.4), suggesting that the unprojected state introduces excessive flexibility.

4.3 Parameter and Computation Costs

Each DepthSSM module adds the down/up projections, the B/C projections, the decay logits, the skip weight, and an RMSNorm layer, for a total of $2dd' + 2d'N + N + d + 1$ parameters, where all projections are bias-free. With two modules per layer and L layers, the total overhead for our 125M configuration ($L=12, d=768, d'=64, N=16$) is 2.4M parameters, or 1.56% of the base model’s 152.7M parameters (Appendix A.2). Across a full sequence, the state update costs $O(T \cdot N \cdot d')$ per layer, negligible relative to self-attention’s $O(T^2 \cdot d)$.

5 Experiments

5.1 Setup

Model architecture. We use GPT-style decoder-only transformers (Radford et al., 2019; Brown et al., 2020) with RMSNorm (Zhang & Sennrich, 2019), no bias terms, SwiGLU

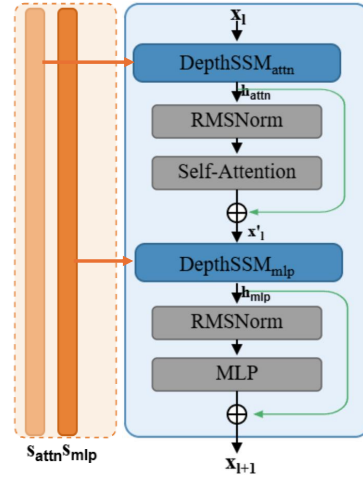


Figure 2: DepthSSD block with DepthSSM modules applied before attention and MLP, each combining normalization, depth-axis SSM transformation, and skip connection.

Table 1: Results across two model scales. All methods share the same architecture, data, and hyperparameters per scale; only the depth-mixing mechanism differs. Effective batch size is 32 for all runs. Best per scale in **bold**, second best underlined.

	Method	Rank	Train Loss ↓	Val. Loss ↓	Per. ↓	Tok/s	TFLOPS	Mem
125M	Standard	1 (fixed)	3.464	3.774	43.5	24,125	19.3	41.0
	BLOCKATTNRES	$\leq L^2/B^2$	3.394	3.770	43.4	9,987	8.0	63.4
	DENSEFORMER	L (full)	3.401	3.714	41.0	12,198	9.7	57.4
	HYPERCONN ($m=4$)	4	3.247	3.760	42.9	10,903	8.7	53.3
	DEPTHSSD- $N=4$	4	3.374	<u>3.759</u>	42.9	15,914	12.7	48.2
	DEPTHSSD- $N=8$	8	3.361	3.760	42.9	15,194	12.1	49.1
	DEPTHSSD- $N=16$	16	<u>3.277</u>	3.761	43.0	13,890	11.1	50.8
455M	Standard	1 (fixed)	3.343	3.440	31.2	9,234	21.0	45.1
	BLOCKATTNRES	$\leq L^2/B^2$	3.427	3.440	31.2	2,853	6.5	49.7
	DENSEFORMER	L (full)	3.295	<u>3.405</u>	<u>30.1</u>	3,087	7.0	48.2
	HYPERCONN ($m=4$)	4	3.346	<u>3.423</u>	30.7	3,891	8.8	35.7
	DEPTHSSD- $N=4$	4	3.372	3.437	31.1	6,017	13.7	54.4
	DEPTHSSD- $N=8$	8	3.345	3.381	29.4	5,808	13.2	53.4
	DEPTHSSD- $N=16$	16	3.426	3.421	30.6	5,422	12.3	56.9
1B	Standard	1 (fixed)	3.158	3.221	25.1	5,200	24.8	41.6
	BLOCKATTNRES	$\leq L^2/B^2$	3.213	3.178	<u>24.0</u>	1,849	8.8	74.7
	DENSEFORMER	L (full)	<u>3.112</u>	3.184	24.1	2,016	9.6	72.9
	HYPERCONN ($m=4$)	4	3.137	3.192	24.3	2,485	11.8	53.8
	DEPTHSSD- $N=4$	4	3.159	3.206	24.7	3,563	17.0	48.4
	DEPTHSSD- $N=8$	8	3.091	3.124	22.7	3,464	16.5	48.8
	DEPTHSSD- $N=16$	16	3.146	3.157	23.5	3,312	15.8	49.7
	DEPTHSSD- $N=24$	24	3.083	3.138	23.1	3,193	15.2	50.5

MLP (Shazeer, 2020), and learned positional embeddings with $T=1024$. We evaluate at three scales: **1) 152M**: 12 layers, 12 heads, $d=768$, $d_{\text{ff}}=3072$, **2) 455M**: 24 layers, 16 heads, $d=1024$, $d_{\text{ff}}=4096$, **3) 1B**: 24 layers, 16 heads, $d=1536$, $d_{\text{ff}}=6144$.

Baselines. We compare against four depth-mixing baselines: **1) Standard**: standard additive residual connections, **2) BLOCKATTNRES**: block attention residuals with block size 8 (Kimi Team et al., 2026), **3) DENSEFORMER**: depth-weighted average (DWA) with learned scalar weights across all preceding sub-layer outputs (Pagliardini et al., 2024), **4) HyperConnections ($m=4$)**: four parallel residual streams with learned linear mixing (Zhu et al., 2025)

5.2 Main Results

Table 1 presents the main comparison across all three scales (125M, 455M, and 1B). We report cross-entropy loss on the held-out validation set, perplexity, and throughput on a single NVIDIA H100 80GB GPU. All methods across all scales use an identical effective batch size of 32, ensuring that throughput differences reflect the pure architectural overhead of each depth-mixing mechanism.

125M scale. At the 125M scale, DENSEFORMER achieves the best validation loss (3.714), followed by DEPTHSSD- $N=4$ (3.759). HYPERCONN achieves the lowest training loss (3.247) but does not generalize as effectively to the validation set. Notably, all DEPTHSSD configurations maintain significantly higher throughput than other depth-mixing baselines. For example, DEPTHSSD- $N=4$ processes 15,914 tokens/s, which is $1.3\times$ faster than DENSEFORMER (12,198 tokens/s) and $1.59\times$ faster than BLOCKATTNRES (9,987 tokens/s).

455M scale. At 455M, DEPTHSSD- $N=8$ emerges as the best-performing method, achieving the lowest validation loss (3.381) and perplexity (29.4), successfully overtaking DENSE-

Table 2: Ablation study: effect of state dimension N on DEPTHSSD performance at 125M scale. All models use $d'=64$ state projection and identical training setup.

State Dim N	Rank	Train Loss ↓	Val. Loss ↓	Per. ↓	Tok/s	Mem
$N=1$	1	3.431	3.760	43.0	16,592	47.5
$N=2$	2	3.375	3.761	43.0	16,322	47.7
$N=4$	4	3.374	3.759	42.9	15,912	48.2
$N=8$	8	3.361	3.760	42.9	15,241	49.1
$N=16$	16	3.277	3.761	43.0	13,862	50.8
$N=32$	32	3.301	3.763	43.1	11,861	54.4
Standard	1 (fixed)	3.464	3.774	43.5	24,105	41.0
BLOCKATTNRES	$\leq L^2/B^2$	3.394	3.770	43.4	9,975	63.4

FORMER (3.405 validation loss). The efficiency advantage of DEPTHSSD also becomes more pronounced at this scale: DEPTHSSD- $N=8$ runs at 5,808 tokens/s, making it $1.88\times$ faster than DENSEFORMER (3,087 tokens/s) and $2.03\times$ faster than BLOCKATTNRES (2,853 tokens/s), all while delivering superior predictive performance.

1B scale. At the 1B scale, the scaling advantages of DEPTHSSD are clearly demonstrated. DEPTHSSD- $N=8$ achieves the best validation loss (3.124) and perplexity (22.7), substantially outperforming the Standard baseline (3.221) and DENSEFORMER (3.184). Furthermore, deeper integration configurations like DEPTHSSD- $N=24$ achieve the lowest training loss (3.083), indicating strong model capacity. In terms of hardware efficiency, DEPTHSSD- $N=8$ maintains a robust throughput of 3,464 tokens/s, running $1.71\times$ faster than DENSEFORMER (2,016 tokens/s) and nearly $1.87\times$ faster than BLOCKATTNRES (1,849 tokens/s).

Scaling trends. A consistent scaling pattern emerges across the three model sizes: while attention-based methods like DENSEFORMER are highly competitive at the smaller 125M scale, their relative performance and hardware efficiency degrade as model size increases. In contrast, DEPTHSSD maintains highly competitive or superior quality, widening the performance gap at 1B. This suggests that the depth-mixing mechanism of DEPTHSSD scales far more effectively than dense or hyper-connected alternatives, enabling a significantly better trade-off between language modeling perplexity and computational throughput at billion-parameter scales.

5.3 Ablation: Effect of State Dimension N

We conduct a comprehensive ablation over $N \in \{1, 2, 4, 8, 16, 32\}$ to understand how semiseparable rank affects performance. Results are shown in Table 2.

The complete ablation reveals several findings (Table 2): From $N=1$ to $N=16$, loss decreases approximately log-linearly with state dimension: a linear fit gives loss $\approx -0.034 \cdot \log_2(N) + 3.43$, or approximately 0.034 loss reduction per doubling of N . Even $N=1$ (loss 3.431) improves over standard residuals (3.464), demonstrating that learned scalar depth mixing provides value. The jump from $N=1$ to $N=2$ is substantial (3.431 $\rightarrow$ 3.375), while $N=2$ to $N=4$ shows diminishing returns (3.375 $\rightarrow$ 3.374). $N=32$ (loss 3.301) performs *worse* than $N=16$ (loss 3.277), despite having strictly greater capacity. This suggests overfitting of the depth mixing parameters: with 32 state dimensions and only 12 layers, the SSM is over-parameterized relative to the depth of the network. The optimal N appears to be $N^* \approx L$ (the number of layers), consistent with the theoretical capacity of full-rank mixing. Throughput decreases from 16,592 tokens/s at $N=1$ to 11,861 at $N=32$, a 29% reduction. Memory increases from 47.5 GB to 54.4 GB. The cost is dominated by the SSM state tensor ($B \times T \times N \times d'$), which scales linearly with N . DEPTHSSD- $N=4$ offers an attractive operating point: 3.374 loss at 15,912 tokens/s, $1.60\times$ faster than BLOCKATTNRES. Every DEPTHSSD variant with $N \geq 2$ achieves lower loss than BLOCKATTNRES (3.394) while running $1.19\text{--}1.64\times$ faster and using 14–25% less memory.

Table 3: Ablation of projected state dimension d' for DEPTHSSD- $N=16$ at 125M scale. $d'=0$ denotes no state projection (SSM operates in full $d=768$ space); training fails to converge. Best result in **bold**.

State Dim d'	Train Loss ↓	Val. Loss ↓	Per. ↓	Tok/s	Converged?
$d'=0$ (no projection)	10.81	,	,	4,851	No
$d'=32$	3.361	3.758	42.9	15,038	Yes
$d'=64$ (default)	3.277	3.761	43.0	13,862	Yes
$d'=128$	3.475	3.758	42.9	12,100	Yes

Table 4: Ablation of decay initialization A_{init} for DEPTHSSD- $N=16$ at 125M scale. A_{init} is the logit value such that $\sigma(A_{\text{init}})$ gives the initial diagonal decay. Best result in **bold**.

A_{init}	Initial $\sigma(A_{\text{init}})$	Train Loss ↓	Val. Loss ↓	Per. ↓	Tok/s
1.0	0.731	3.370	3.764	43.1	13,936
2.2 (default)	0.900	3.277	3.761	43.0	13,862
3.0	0.953	3.554	3.759	42.9	13,938

5.4 Component Ablations

We ablate two key design choices of DEPTHSSD: the projected state dimension d' and the decay initialization A_{init} . All experiments use the 125M model with $N=16$.

State projection dimension (d'). Table 3 shows the effect of projecting the hidden dimension from $d=768$ to a lower-dimensional space before the SSM. Setting $d'=0$ (no projection, operating directly in the full d -dimensional space without learned down/up projections) causes training to fail entirely, the loss plateaus near 10.81, never decreasing below initialization. This demonstrates that the state projection is not merely an efficiency optimization but is *essential* for stable training: without it, the SSM state update $\mathbf{b}_l \otimes \mathbf{x}_l \in \mathbb{R}^{N \times d}$ introduces too many degrees of freedom, preventing the model from learning meaningful depth mixing.

Among working configurations, $d'=64$ achieves the best loss (3.277), outperforming both $d'=32$ (3.361) and $d'=128$ (3.475). The degradation at $d'=128$ suggests overfitting of the projection parameters, consistent with the relatively small training set (104M tokens). The $d'=64$ setting provides a $12\times$ reduction in SSM state memory compared to operating in the full hidden dimension.

Decay initialization (A_{init}). Table 4 ablates the initialization of the diagonal decay logits $\mathbf{a}$, which control the initial value of $\sigma(\mathbf{a}) = A$. The default $A_{\text{init}}=2.2$ gives $\sigma(2.2) \approx 0.90$, meaning each state dimension retains 90% of its value per layer.

$A_{\text{init}}=1.0$ ($\sigma(1.0) \approx 0.73$) achieves loss 3.370, slightly worse than the default. This faster initial decay reduces the effective memory span of the SSM, limiting cross-layer information flow during early training. $A_{\text{init}}=3.0$ ($\sigma(3.0) \approx 0.95$) performs substantially worse (3.554), suggesting that very slow initial decay creates near-degenerate mixing matrices where all layers receive nearly equal weight, hindering specialization.

The optimal initialization at $A_{\text{init}}=2.2$ balances two competing objectives: providing enough memory span for meaningful cross-layer mixing while allowing sufficient differentiation between layers' contributions.

5.5 Training Dynamics

Figure 3(c,d) compares the per-block output and gradient magnitudes for Standard and DEPTHSSD- $N=24$ at 455M, measured on a random validation batch after training.

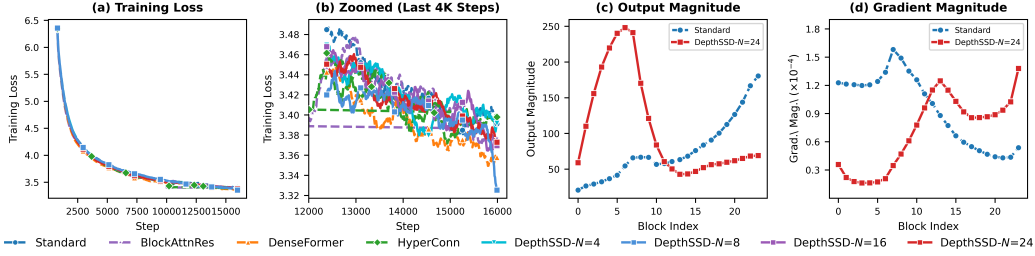


Figure 3: Training analysis at 455M scale. (a) Training loss curves (smoothed, 80-step window); baselines dashed, DEPTHSSD solid. (b) Zoomed view of the last 4,000 steps where method differences become visible. (c) Per-block output ℓ_2 norm: Standard shows monotonic growth (PreNorm dilution); DEPTHSSD peaks early then decays. (d) Per-block gradient ℓ_2 norm: DEPTHSSD redistributes gradients toward deeper layers.

Output magnitude. The Standard model exhibits monotonically growing output norms ($\approx 20 \rightarrow 180$ across 24 blocks), the well-known PreNorm dilution effect (Xiong et al., 2020; Wang et al., 2022): as hidden states accumulate through the residual stream, deeper layers must produce increasingly large outputs from fixed-scale normalized inputs to remain influential. DEPTHSSD- $N=24$ breaks this pattern: output norms peak at ≈ 247 around block 6 and then *decrease* to ≈ 70 by block 23. The SSM’s learned decay ($\sigma(\mathbf{a}) \approx 0.84$) actively dampens the cumulative state, preventing late-layer norm explosion.

Gradient magnitude. Standard shows the highest gradients in the early-to-middle layers (peak $\approx 1.6 \times 10^{-4}$ at block 7), with gradients decaying toward both ends. DEPTHSSD exhibits a complementary pattern: near-zero gradients in the earliest blocks ($< 0.2 \times 10^{-4}$ for blocks 0–4), rising to $\approx 1.4 \times 10^{-4}$ at block 23. This redistribution concentrates learning signal in the deeper layers where DEPTHSSD has already built a rich cross-layer summary in its state, while the early layers, whose output is dominated by the amplified skip connection ($D \approx 2.4$), require less gradient adjustment.

5.6 Analyses of Learned Parameters and Efficiency

The learned depth-SSM parameters exhibit structured layer-wise patterns, and DEPTHSSD provides a favorable efficiency–quality tradeoff relative to BLOCKATTNRES, while the Standard baseline remains strongest on raw efficiency. Full analyses and supporting tables are deferred to Sec. C.

6 Related Work

Our work is related to several lines of research on inter-layer information flow and efficient sequence modeling, including residual connections and skip connections (He et al., 2016; Huang et al., 2017; Srivastava et al., 2015; Xiong et al., 2020; Wang et al., 2022), depth-axis attention and related methods (Kimi Team et al., 2026; Zhang, 2026; Zhu et al., 2025; Pagliardini et al., 2024), state space models (Gu et al., 2022; Gu & Dao, 2023; Dao & Gu, 2024), and depth-wise recurrence (Dehghani et al., 2019; Tay et al., 2021). Recent work has also explored richer residual pathways within transformers, for example through residual attention (He et al., 2021) and dual residual formulations (Xie et al., 2023). A fuller discussion is deferred to Sec. D.

7 Conclusion

We showed that depth-axis mixing in transformers admits a semiseparable structure, yielding a unified view of standard residual connections and recent learned depth-mixing mechanisms. Building on this perspective, DEPTHSSD applies the Structured State Space Duality framework to the depth axis, enabling controllable-rank depth mixing with modest overhead.

Across all tested scales, DEPTHSSD provides a stronger quality–efficiency tradeoff than prior depth-mixing baselines and scales more favorably to larger models. These results suggest that the transformer depth axis is a meaningful target for learned information routing, and that structured state space models offer an efficient and principled way to exploit it.

Acknowledgments

This work was partially supported by the NSF under grants IIS-2533550, IIS-2321504, IIS-2217239, CNS-2426514, and CMMI-2146076, Notre Dame Strategic Framework Research Grant (2025), and Notre Dame Poverty Research Package (2025). Any expressed opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

References

- Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2021. URL <https://arxiv.org/abs/2003.04887>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL <https://arxiv.org/abs/2005.14165>.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022. URL <https://arxiv.org/abs/2204.02311>.
- Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *International Conference on Learning Representations (ICLR)*, 2019.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>.
- Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *International Conference on Learning Representations (ICLR)*, 2022.

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
- Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. Realformer: Transformer likes residual attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 929–943, 2021.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. URL <https://arxiv.org/abs/2203.15556>.
- Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708, 2017.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL <https://arxiv.org/abs/2001.08361>.
- Kimi Team et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026. URL <https://arxiv.org/abs/2603.15031>.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bkg6RiCqY7>.
- Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=kMnoh7CXrq>.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024. URL <https://arxiv.org/abs/2406.17557>.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Technical Report, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. URL <https://arxiv.org/abs/2103.00020>.
- Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL <https://arxiv.org/abs/2002.05202>.
- Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.
- Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 10183–10192, 2021.

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.
- Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.
- Shufang Xie, Huishuai Zhang, Junliang Guo, Xu Tan, Jiang Bian, Hany Hassan Awadalla, Arul Menezes, Tao Qin, and Rui Yan. Residual: Transformer with dual residual connections. *arXiv preprint arXiv:2304.14802*, 2023. URL <https://arxiv.org/abs/2304.14802>.
- Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 10524–10533, 2020.
- Yanfang Ye, Zheyuan Zhang, Tianyi Ma, Zehong Wang, Yiyang Li, Shifu Hou, Weixiang Sun, Kaiwen Shi, Yijun Ma, Wei Song, et al. Llm4all: A review of large language models across academic disciplines. *arXiv preprint arXiv:2509.19580*, 2025.
- Biao Zhang and Rico Sennrich. Root mean square layer normalization. *arXiv preprint arXiv:1910.07467*, 2019. URL <https://arxiv.org/abs/1910.07467>.
- Yifan Zhang. Residual stream duality in modern transformer architectures. *arXiv preprint arXiv:2603.16039*, 2026.
- Defa Zhu et al. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2025.

A Method Details

A.1 Initialization Details

Proper initialization is critical for depth-axis SSMs, as the model must behave approximately as a standard residual network at the start of training. We initialize:

- $\mathbf{a}$: set to $\sigma^{-1}(0.9) \approx 2.2$, so the diagonal decay starts near identity.
- D : set to 1.0, giving a strong skip connection.
- W_B, W_C : small normal initialization ($\sigma = 0.01$).
- $W_{\text{down}}, W_{\text{up}}$: scaled normal initialization.

With these values, the initial output satisfies $\mathbf{h}_l \approx D \cdot \mathbf{x}_l \approx \mathbf{x}_l$, since the B/C projections start near zero.

A.2 Parameter and Computation Cost Breakdown

We use bias-free linear projections throughout DepthSSD. Each DepthSSM module adds:

- Down/up projections: $2 \cdot d \cdot d'$ parameters.
- B and C projections: $2 \cdot d' \cdot N$ parameters.
- Decay logits $\mathbf{a}$: N parameters.
- Skip weight D : 1 parameter.
- RMSNorm: d parameters.

Total per module: $2d \cdot d' + 2d' \cdot N + N + d + 1$. With two modules per layer (pre-attention and pre-MLP) and L layers:

$$\text{Total DepthSSD params} = 2L \cdot (2dd' + 2d'N + N + d + 1) \quad (9)$$

For our 125M model ($L=12$, $d=768$, $d'=64$, $N=16$): 2,427,288 parameters, or 1.56% of the base model’s 152.7M parameters. This overhead is independent of the sequence length T .

Across a full sequence, the state update costs $O(T \cdot N \cdot d')$ per layer (dominated by the outer product $\mathbf{b}_l \otimes \mathbf{x}_{\text{low}}$ at each of T positions), which is negligible relative to self-attention’s $O(T^2 \cdot d)$.

B Discussion and Limitations

The rank–quality tradeoff. Our experiments establish a consistent relationship between semiseparable rank N and language modeling quality at the 125M scale, with a log-linear regime for $N \leq 16$ and saturation at $N=32$. This suggests that depth-axis information routing in transformers benefits from richer mixing than standard residuals provide, but full-rank mixing (ATTNRES) may be unnecessarily expensive. The optimal N likely depends on model depth and may follow a scaling law analogous to those for model width and depth (Kaplan et al., 2020; Hoffmann et al., 2022).

Scaling behavior. A striking finding is that depth-mixing baselines (BLOCKATTNRES, DENSEFORMER, HYPERCONN) degrade relative to Standard as model size increases from 125M to 1B, while DEPTHSSD maintains competitive or superior quality. At 1B, only DEPTHSSD- $N=16$ outperforms Standard among all tested methods. We attribute this to DEPTHSSD’s lower per-layer overhead: with the training compute budget held constant (same batch size and step count), methods with higher overhead effectively see fewer gradient updates per wall-clock second, which becomes increasingly detrimental at larger scales.

Parameter overhead. The 1.5% parameter overhead of DEPTHSSD is larger than BLOCKATTNRES’s 0.02% but is dominated by the down/up projections ($d \times d'$) rather than the SSM parameters ($d' \times N$). Future work could explore shared projections across layers or alternative dimensionality reduction strategies to reduce this overhead.

Limitations.

1. **Scale:** While we evaluate at three scales (125M, 455M, 1B), training beyond 1B parameters and on larger datasets remains an open question. The relationship between optimal N and model depth may change at larger scales.
2. **Data:** The 125M models train on 104M tokens (3.4 epochs) and the 455M/1B models on 517M tokens.
3. **Downstream evaluation:** We report only cross-entropy loss on a held-out validation set. Downstream task performance (e.g., on benchmarks) may not correlate perfectly with perplexity.
4. **Sequential depth recurrence:** The SSM recurrence along the depth axis is inherently sequential (layer l depends on layer $l-1$), which prevents depth-wise parallelism. However, this is the same constraint as standard transformers, the key overhead is the state maintenance, not the recurrence itself.
5. **Chunk-wise algorithm:** For very deep models ($L > 100$), a chunk-wise SSD algorithm (analogous to Dao & Gu 2024’s sequence-axis chunking) could reduce memory by operating on chunks of layers. We leave this optimization for future work.

C Additional Experimental Analyses

C.1 Experimental Details

Training. All models are trained for 10,000 steps using AdamW (Loshchilov & Hutter, 2019) ($\beta_1=0.9$, $\beta_2=0.95$, weight decay 0.1, $\epsilon=10^{-8}$) with cosine learning rate decay and 500-step linear warmup. We use BF16 mixed precision on NVIDIA H100 80GB GPUs with

Table 5: Learned skip connection weights D by layer for DEPTHSSD- $N=16$ at training completion (step 9,990). Layer 0 shows the strongest skip connection ($D=2.55$); middle layers (5, 8, 9) have the weakest ($D \approx 0.69$).

Layer	0	1	2	3	4	5	6	7	8	9	10	11
D	2.55	1.31	1.06	0.95	0.82	0.69	0.74	0.74	0.69	0.70	1.00	0.97

Table 6: Efficiency comparison. Throughput relative to Standard baseline and memory relative to BLOCKATTNRES.

Method	Tokens/s	vs. Standard	Memory	vs. BlockAttnRes	Loss
Standard	24,105	1.00×	41.0 GB	0.65×	3.464
BLOCKATTNRES	9,975	0.41×	63.4 GB	1.00×	3.394
DEPTHSSD- $N=4$	15,912	0.66×	48.2 GB	0.76×	3.374
DEPTHSSD- $N=8$	15,241	0.63×	49.1 GB	0.77×	3.361
DEPTHSSD- $N=16$	13,862	0.57×	50.8 GB	0.80×	3.277

gradient clipping of 1.0. Scale-specific settings: 125M uses batch size 32, gradient accumulation 1 and peak learning rate 6×10^{-4} ; 455M uses batch size 16, gradient accumulation 2 and peak 3×10^{-4} ; 1B uses batch size 8, gradient accumulation 4 and peak 2×10^{-4} .

Data. The 125M experiments train on a 104M-token subset of FineWeb-Edu (Penedo et al., 2024). The 455M and 1B experiments use a larger 517M-token subset. All data is tokenized with the GPT-2 tokenizer (Radford et al., 2019) (vocabulary size 50,304).

C.2 Analysis of Learned Parameters

Decay values (A). The learned decay values $A_l = \sigma(\mathbf{a}_l)$ across layers for DEPTHSSD- $N=16$ show a characteristic pattern:

- Early layers (0–2) maintain relatively high decay ($A \approx 0.83$), preserving more information from previous layers.
- Middle layers (4–8) develop lower decay values ($A \approx 0.77$ – 0.80), indicating faster information turnover.
- The final layers show a slight recovery in decay values.

This “U-shaped” pattern across depth mirrors findings in sequence-axis SSMs, where boundary positions (beginning/end) tend to maintain longer memory spans.

Skip connection weights (D). The learned D values show a striking pattern (Table 5): layer 0 learns $D \approx 2.55$ (strong skip connection), while middle layers decrease to $D \approx 0.69$ – 0.74 , and the final layers recover to $D \approx 0.97$ – 1.00 . This suggests that:

- The first layer primarily passes through the embedding with amplification, allowing the SSM state to build up a running summary.
- Middle layers rely more heavily on the SSM-mixed representation, using richer cross-layer information.
- Final layers return to stronger skip connections, perhaps to preserve task-specific features for the output head.

C.3 Efficiency Analysis

Table 6 summarizes the efficiency–quality tradeoffs. DEPTHSSD achieves a Pareto improvement over BLOCKATTNRES: better loss, higher throughput, and lower memory at all

tested values of N . The standard baseline remains Pareto-optimal on the efficiency axis but sacrifices substantial quality.

The throughput reduction from Standard to DEPTHSSD is primarily due to the SSM state maintenance (storing and updating $\mathbf{s} \in \mathbb{R}^{B \times T \times N \times d'}$ at each layer) rather than the SSM computation itself, which is $O(N \cdot d')$ per layer. The memory overhead is modest: DEPTHSSD- $N=16$ uses only 50.8 GB vs. 41.0 GB for Standard (a 24% increase), compared to BLOCKATTNRES’s 63.4 GB (a 55% increase over Standard).

D Extended Related Work

Residual connections and skip connections. The residual connection (He et al., 2016) enables training of very deep networks by providing identity shortcuts. DenseNet (Huang et al., 2017) concatenates all previous layer outputs, while Highway Networks (Srivastava et al., 2015) use gating mechanisms. In transformers, Pre-LN (Xiong et al., 2020) and DeepNorm (Wang et al., 2022) improve training stability through normalization placement and scaling.

Depth-axis attention. Kimi Team et al. (2026) introduce Attention Residuals, applying softmax attention across the depth axis to replace the fixed residual stream. Their work demonstrates consistent gains at scales from 70M to 1B but incurs quadratic cost in the number of layers. We observe a duality between depth and sequence axes, connecting residual connections to linear attention.

Hyper-Connections. Zhu et al. (2025) propose maintaining multiple parallel residual streams that interact through learned linear transformations at each layer. This can be viewed as a specific instantiation of our framework with $N=m$ (number of streams) and particular constraints on the SSM parameters.

Depth-weighted averaging. Pagliardini et al. (2024) introduce DenseFormer, which computes a depth-weighted average (DWA) of all preceding sub-layer outputs using learned scalar weights. While DWA provides full connectivity across layers, its $O(L^2)$ scalar parameters and storage of all intermediate activations lead to substantial memory overhead at larger model scales.

State space models. S4 (Gu et al., 2022), Mamba (Gu & Dao, 2023), and Mamba-2 (Dao & Gu, 2024) have established SSMs as efficient alternatives to attention for *sequence*-axis modeling. Our work applies the same mathematical framework to the orthogonal *depth* axis, exploiting the semiseparable structure for efficient inter-layer information flow.

Depth-wise recurrence. Universal Transformers (Dehghani et al., 2019) share parameters across layers with a halting mechanism. Tay et al. (2021) explore learned attention patterns. Our approach differs by maintaining an explicit recurrent state across depth while keeping layer-specific parameters.

E LLM Usage

A large language model (LLM) is used exclusively for editorial assistance during manuscript preparation. The LLM is applied to improve writing quality, including grammar correction, phrasing, and overall readability.

The LLM is *not* used for generating research ideas, designing methods, conducting experiments, or interpreting results. All technical contributions and experimental findings are produced entirely by the authors.

Thus, the use of LLMs is limited to language polishing and does not influence the scientific content of this work.