

Depth-Attention: Cross-Layer Value Mixing for Language Models

Boyi Zeng^{1,*}, Yiqin Hao^{1,*}, Zitong Wang^{1,4}, Shixiang Song^{1,5}, He Li¹,
Feichen Song¹, Yifan Liu¹, Ziwei He⁵, Xinbing Wang³, Zhouhan Lin^{1,2†}

¹ LUMIA Lab, School of Artificial Intelligence, Shanghai Jiao Tong University

² Shanghai AI Laboratory ³ Shanghai Jiao Tong University

⁴ Sun Yat-sen University ⁵ Shanghai Innovation Institute

✉ boyizeng@sjtu.edu.cn, hyq718@sjtu.edu.cn, lin.zhouhan@gmail.com

* Equal Contribution † Corresponding Author.

Abstract

Self-attention selects information freely across the sequence, but across depth, Transformers merely add each layer’s output to the residual stream, so later layers cannot selectively reuse earlier-layer representations. Recent cross-layer methods improve this flow but operate on hidden states outside attention, adding state beyond the key-value cache at inference—a cost that becomes increasingly salient as modern LLMs compress the cache with grouped-query and multi-head latent attention. We introduce **Depth-Attention**, which performs this selection inside the attention module itself: before a layer attends over the sequence, its query attends *along depth* over the keys of earlier layers at the same position and mixes their values into the value that self-attention then reads. Because Depth-Attention reuses the standard attention queries, keys, and value-cache slots, storing depth-mixed values in place of the original values, it adds no parameters and introduces no persistent inference state beyond the standard key-value cache—the same cache size as a vanilla decoder and less than hidden-state-based cross-layer methods. On Qwen3-style decoders at 1.5B and 3B parameters, Depth-Attention attains the lowest perplexity and the highest average downstream accuracy, improving over the vanilla Transformer by up to 2.3 accuracy points and surpassing strong cross-layer baselines (mHC, Attention Residuals, DenseFormer) in perplexity and average accuracy, while adding under 0.01% extra arithmetic FLOPs and no additional persistent inference state. The gains hold from 360M to 3B parameters, persist at a 60B-token training budget, and extend to LLaMA-style decoders and looped Transformers.¹

¹The code is available at <https://github.com/LUMIA-Group/Depth-Attention>.

1 Introduction

Transformer language models rely on self-attention to dynamically select information across the sequence dimension (Vaswani et al., 2017). Across depth, however, information flows far less selectively: standard decoders propagate earlier-layer information only through residual connections (He et al., 2016), collapsing all previous layers into a single hidden state, so deeper layers cannot selectively retrieve the representation of any individual earlier layer.

A growing line of work makes this depth-wise flow more explicit. DenseFormer (Pagliardini et al., 2024) aggregates the outputs of preceding layers through learned depth-wise averaging, and Hyper-Connections (Zhu et al., 2025) replace the single residual path with multiple interacting streams that exchange information across depths; most recently, Attention Residuals (Kimi Team et al., 2026) make depth-wise selection adaptive by replacing fixed residual accumulation with a softmax attention over preceding layer outputs. These mechanisms all operate on hidden states, however: they require access to earlier layers’ hidden representations, which lie outside the key-value cache. At inference, a model must therefore maintain or recompute this extra state on top of its key-value cache—an overhead that grows in relative terms as modern LLMs aggressively shrink the cache with grouped-query attention (GQA) (Ainslie et al., 2023) or multi-head latent attention (MLA) (DeepSeek-AI, 2024). It raises a natural question: can a layer reuse the keys and values it already stores to carry information across depth, without adding any memory at inference?

We propose **Depth-Attention**, which performs this depth-wise selection inside the attention module itself. Before a layer mixes information across

the sequence with self-attention, it uses the current-layer query to attend, along depth, over the keys of earlier layers at the same token position, and mixes their values into an updated value state—the same attention operation as self-attention, but applied across layers instead of across the sequence. The layer then runs standard causal self-attention on this depth-mixed value, leaving its queries, keys, and the causal mask unchanged. Because Depth Attention reuses the queries and keys of self-attention and stores its depth-mixed values in place of the originals in the V cache, it adds no parameters and its only persistent inference state is the standard key-value cache, the same size as a vanilla decoder’s—in contrast to hidden-state methods, which carry additional state on top of the cache in the form of widened residual streams or retained earlier-layer representations.

Across Qwen3 (Yang et al., 2025) architecture at 1.5B and 3B parameters, Depth-Attention attains the best average downstream accuracy and the lowest perplexity against the vanilla Transformer and recent cross-layer methods such as manifold-constrained hyper-connections (mHC) (Xie et al., 2025) and Attention Residuals, while incurring lower compute and memory overhead than these baselines.

2 Related Work

Residual connections (He et al., 2016) are the standard mechanism for passing information across layers in deep networks. In Transformer decoders (Vaswani et al., 2017), each block updates the hidden state by adding its newly computed transformation to the representation inherited from the previous block. This design greatly improves the optimization of deep models, and a large body of work has further studied how normalization placement (Xiong et al., 2020), residual scaling (Wang et al., 2024), gating (Bachlechner et al., 2021), and skip-connection variants affect training stability (Xie et al., 2023). However, the standard residual pathway also compresses the computation history into a single hidden state. After the information produced by earlier layers has been merged into this state, later layers have no direct way to revisit a specific earlier representation.

A straightforward solution is to expose intermediate representations more explicitly. DenseNet (Huang et al., 2017) connects each layer to all preceding feature maps, allowing later layers

to reuse earlier computations directly, while Peters et al. (2018) and Zhou et al. (2025) learn scalar combinations of representations from different layers. Transformer variants follow a similar intuition. DenseFormer (Pagliardini et al., 2024) and ANCRE (Zhang et al., 2026) introduce learned layer-wise coefficients to combine historical representations, and RealFormer (He et al., 2021) carries attention scores across consecutive layers. These methods suggest that intermediate-layer representations provide useful information beyond the final hidden state. Nevertheless, many of them either use input-independent layer weights or propagate attention scores, rather than enabling each token to selectively choose the value information it reads from previous layers.

More recent architectures introduce more flexible cross-layer communication. HyperConnections and its variants (Zhu et al., 2025; Xie et al., 2025) replace the single residual state with multiple interacting streams. MUDDFormer (Xiao et al., 2025) predicts position-dependent connection weights for separated query, key, value, and residual streams. Other methods, including LAuReL (Menghani et al., 2025), Dreamer (Knupp et al., 2026), and Attention Residuals (Kimi Team et al., 2026), design more specialized mechanisms for reusing earlier activations or layer outputs. These approaches improve access to intermediate representations from different perspectives, but their cross-layer reuse is still mainly built on hidden representations. When applied to deep autoregressive models, retaining or communicating such intermediate hidden states can introduce additional memory or communication overhead.

Most closely related, the contemporaneous Mixture-of-Depths Attention (MoDA) (Zhu et al., 2026) jointly attends over sequence-level and depth-level key-value states within a single unified attention operator, whereas Depth-Attention leaves causal self-attention unchanged and reuses the existing attention states; a detailed mechanism, cost, and matched empirical comparison is provided in Appendix J.

Depth-Attention follows the same general direction of making earlier-layer information more accessible, but places the cross-layer interaction at the value-state level. Instead of storing an additional history of full hidden states, it reuses value states in a form compatible with the existing KV-cache structure. This allows the model to mix information across layers while avoiding the extra memory

cost associated with hidden-state-based reuse.

3 Methodology

3.1 Overview

Self-attention lets each token mix information across the sequence, but offers no comparable mechanism across depth. **Depth-Attention** fills this gap with an analogous operation along the depth dimension (Figure 1): at a fixed token position, the current-layer query attends over the keys of the current and earlier layers and mixes their values into a single depth-mixed value. A layer first applies Depth-Attention to update its value, then runs standard causal self-attention that reads from this depth-mixed value, leaving the queries, keys, and causal mask unchanged. Because Depth-Attention reuses the query and key already computed by self-attention, and stores the depth-mixed value in the same V-cache slot as the original, it adds no parameters and its only persistent inference state is the standard key-value cache, unchanged in size from a vanilla decoder’s.

3.2 Depth-Attention

Consider a decoder layer ℓ operating on a sequence of length T , with query, key, and value states

$$Q_\ell, K_\ell, V_\ell \in \mathbb{R}^{T \times d}, \quad (1)$$

where d is the head dimension; q_ℓ^t , k_ℓ^t , and v_ℓ^t denote the states of the token at position t .

Depth-Attention is a standard attention operation applied along the *depth* dimension at a fixed token position. For the token at position t in layer ℓ , the current query q_ℓ^t attends over the keys from the current and earlier layers at the same position:

$$\alpha_{\ell,j}^t = \text{softmax}_{j \leq \ell} \left(\frac{q_\ell^t \cdot k_j^t}{\sqrt{d}} \right), \quad j \leq \ell. \quad (2)$$

Using these depth-attention weights, Depth-Attention forms a depth-mixed value by reading from the current-layer value and the depth-mixed values of earlier layers:

$$\tilde{v}_\ell^t = \alpha_{\ell,\ell}^t v_\ell^t + \sum_{j < \ell} \alpha_{\ell,j}^t \tilde{v}_j^t. \quad (3)$$

The current layer contributes its own value v_ℓ^t , while each earlier layer $j < \ell$ contributes its depth-mixed value $\tilde{v}_j^t$, so that information accumulated at shallower depths is propagated upward. Applying

this to every position gives the depth-mixed value state $\tilde{V}_\ell \in \mathbb{R}^{T \times d}$.

The layer then performs standard causal self-attention, reading from the depth-mixed value while keeping the original queries and keys:

$$O_\ell = \text{CausalAttn}(Q_\ell, K_\ell, \tilde{V}_\ell). \quad (4)$$

Depth-Attention thus replaces only the value consumed by self-attention: the queries, keys, and causal mask remain those of a vanilla decoder. At inference, layer ℓ stores its depth-mixed value $\tilde{V}_\ell$ in the standard V-cache slot *in place of* V_ℓ ; because the two have identical shape, this reuses the existing key-value cache without enlarging it. Subsequent tokens read $\tilde{V}_\ell$ from that slot both as the value of layer ℓ ’s causal self-attention and, when needed, as the depth source for any later layer that mixes through it—so Depth-Attention introduces no persistent inference state beyond what a vanilla decoder already maintains. Because the attention runs over the depth dimension, whose size (the number of layers) is far smaller than the sequence length T , its overhead is negligible compared with sequence-wise self-attention. In practice, the index j in the equations above ranges over a sparse strided subset of earlier layers rather than the full $j \leq \ell$; the form of the operation is unchanged, only the number of source layers summed over, as detailed in the next subsection.

3.3 Efficient Implementation

Sparse depth sources via stride. Although the per-layer overhead of Depth-Attention is asymptotically negligible, allowing layer ℓ to attend to *every* earlier layer makes the depth source set grow with ℓ . This inflates activation memory and—under pipeline parallelism, where consecutive layers live on different devices—the cross-stage communication required to fetch earlier-layer keys and values. To bound both costs, we restrict the depth sources to a fixed strided set of layers shared across the network. Specifically, given a stride s , layer ℓ attends to its own value together with the values of earlier layers whose indices are multiples of s , i.e., layers $0, s, 2s, \dots$ below ℓ . This keeps the number of candidate depth sources small while preserving access to representations spanning different depths. We find that this sparsification does not hurt model quality compared with denser source sets: our ablations (Figure 4) show that a stride of $s = L/2$, where L is the total number of layers, gives the best

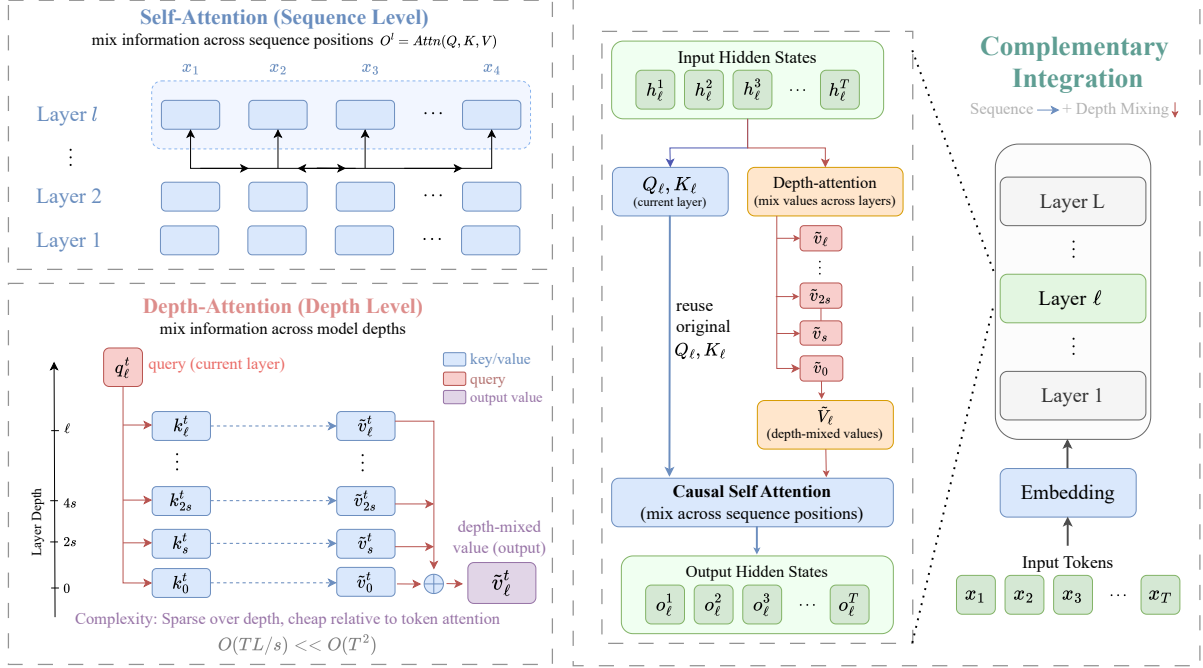


Figure 1: Overview of **Depth-Attention**. Standard self-attention mixes information across sequence positions within each layer, whereas Depth-Attention performs a complementary operation across model depths at the same token position. For token position t in layer ℓ , the current-layer query q_t^ℓ attends to keys from the current layer together with a fixed strided set of shallower layers whose indices are multiples of s , i.e., $\{k_t^\ell\} \cup \{k_t^j \mid j < \ell, j \bmod s = 0\}$, and uses the resulting depth-wise attention weights to combine the current value with previously depth-mixed values into $\tilde{v}_t^\ell$. Because the depth source set is sparse, Depth-Attention adds only $O(TL/s)$ operations per layer, which is negligible compared with the $O(T^2)$ cost of token-level self-attention. Applying this operation to all token positions produces the depth-mixed value state $\tilde{V}_\ell$, which replaces the original value state V_ℓ in the subsequent causal self-attention while keeping the queries, keys, and causal mask unchanged. Depth-Attention can be seamlessly integrated into Transformer layers with negligible overhead.

results, and we use this setting as the default for all subsequent experiments unless otherwise specified.

Grouped-query attention. Modern LLMs widely adopt grouped-query attention (GQA), which shrinks the key-value cache by letting g query heads share a single key-value head. Depth-Attention adapts naturally to this setting: we average each group of g queries into a single query so that the depth-wise attention runs at the key-value head resolution. Because the key-value dimension is several times smaller than the hidden size, performing Depth-Attention at this resolution further shrinks both its extra compute and its extra memory, making our method especially efficient in this widely-used regime.

4 Experiments

Our experiments are organized into five parts. First, we conduct large-scale pretraining experiments on Qwen3-style architectures at 1.5B and 3B scales, comparing Depth-Attention with vanilla Trans-

former decoders and representative cross-layer baselines. Second, we analyze the efficiency of Depth-Attention from both theoretical and empirical perspectives, covering computation, training wall-clock overhead, inference throughput, and memory usage. Third, we study the scaling behavior of Depth-Attention across model sizes, evaluating whether its gains remain consistent on different model scales. Fourth, we visualize the learned depth-attention weights to examine how the model uses shallower value sources when constructing depth-mixed value states. Finally, we evaluate Depth-Attention in a looped Transformer setting, testing whether the same depth-wise value mixing mechanism can extend to recurrent depth induced by parameter sharing.

4.1 Main Results

We first evaluate Depth-Attention on Qwen3-style decoder architectures (Yang et al., 2025) at 1.5B and 3B scales. All models are trained from scratch for 32B tokens on the Pile (Gao

et al., 2020) under the same pretraining protocol, including the same training data, optimization schedule, and hyperparameters. We compare Depth-Attention with the vanilla Transformer decoder and representative cross-layer baselines, including manifold-constrained hyper-connections (mHC) (Xie et al., 2025), Attention Residuals (Kimi Team et al., 2026), and DenseFormer-style dense layer reuse (Pagliardini et al., 2024).

We report Pile validation perplexity to measure language modeling quality, and evaluate downstream performance under both zero-shot and five-shot settings, including LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), SciQ (Welbl et al., 2017), HellaSwag (Zellers et al., 2019), ARC-Easy, ARC-Challenge (Clark et al., 2018), and RACE (Lai et al., 2017). Detailed model configurations and training hyperparameters are provided in Appendix A.

As shown in Table 1, Depth-Attention attains the best perplexity and the best average downstream accuracy against the vanilla Transformer and all cross-layer baselines on both 1.5B and 3B Qwen3-style models, under both zero-shot and five-shot evaluation; per-task trends are mixed, but no individual baseline matches Depth-Attention’s average at either scale or evaluation setting.

Paired statistical analyses on the 3B per-example predictions (task-stratified paired bootstrap and paired permutation tests, Appendix D) confirm that the improvements over the vanilla Transformer are significant in both shot settings, and the improvement over a strong hidden-state baseline, DenseFormer, is significant in the zero-shot setting. These results demonstrate that Depth-Attention effectively improves model performance by enriching current-layer value states through depth-wise value mixing. The advantages also carry over to supervised fine-tuning: after identical Alpaca SFT and GSM8K fine-tuning, the 3B Depth-Attention model achieves higher MT-Bench and GSM8K scores than its vanilla counterpart (Appendices F and G).

4.2 Efficiency Analysis

Theoretical overhead. We compare three sources of overhead that cross-layer mechanisms add to a vanilla decoder: extra computation (FLOPs), extra activation I/O, and extra memory for long contexts. Table 2 reports the per-token extra FLOPs and I/O at a target layer ℓ , instantiates

their totals for one token under our 3B Qwen3-style configuration, and lists the theoretical extra memory required to prefill a 128K-token context. Depth-Attention adds far less computation and activation I/O than representative cross-layer baselines. Moreover, because it reuses the standard key-value cache, Depth-Attention adds no persistent state at inference: its extra prefill memory is zero, whereas the hidden-state baselines require gigabytes of additional memory at long context (up to 24 GiB for DenseFormer at 128K tokens). Detailed derivations are provided in Appendix B.

Training wall-clock. Since none of the cross-layer baselines provides an official kernel-fused implementation, we benchmark every method—including Depth-Attention—under a common reference PyTorch implementation, with no custom kernels. This setting penalizes all methods relative to a fully optimized stack, but it isolates the *methodological* overhead and puts every baseline on the same footing as our method; the numbers below should therefore be read as controlled reference-implementation comparisons rather than production-optimized latency. To rule out confounds from memory-induced batch differences, all methods are measured on the 3B Qwen3-style configuration under a fully matched setting: larger-memory GPUs, identical microbatch size, gradient accumulation, global batch size, sequence length, training data, optimizer settings, hardware, and software stack, with each method timed for 200 training steps after warm-up. As shown in Table 3, Depth-Attention adds only +9.12% wall-clock overhead—the smallest of any cross-layer mechanism by a wide margin: DenseFormer and Attention Residuals are roughly $5\times$ – $6\times$ further from vanilla than Depth-Attention, and mHC is roughly $35\times$ further. A dedicated fused kernel, which we leave to future work, would likely close even this small gap. The measurement protocol is described in Appendix L, and a systems-level analysis of the interaction with activation checkpointing, pipeline and tensor parallelism, and fused attention kernels is given in Appendix K.

Inference efficiency. We further measure the empirical inference cost of Depth-Attention on the trained 3B Qwen3-style model. With batch size 64, prefill length 2048, and decode length 2048, total generation time rises from 526.6 s to 532.8 s—an overhead of only +1.18%—while peak GPU memory during prefill is unchanged at 30.6 GB.

Size	Method	PPL ↓	LAMBADA	PIQA	Wino Grande	SciQ	Hella Swag	ARC- Easy	ARC- Challenge	RACE	Avg.
Zero-shot evaluation											
1.5B	Vanilla	8.17	55.02	68.61	54.85	84.10	36.57	54.08	23.72	33.11	51.26
1.5B	mHC	7.88	57.77	69.10	56.75	86.60	37.88	56.52	24.83	33.59	52.88
1.5B	Attention Residuals	7.78	58.49	69.86	56.04	85.90	38.80	57.07	24.15	32.06	52.80
1.5B	DenseFormer	7.88	56.34	69.26	56.27	85.40	38.47	56.36	25.43	33.78	52.66
1.5B	Depth-Attention	7.77	59.89	70.24	55.72	88.20	38.86	58.50	24.49	32.92	53.60
3B	Vanilla	7.55	58.86	70.57	55.96	86.60	39.84	59.13	25.34	33.78	53.76
3B	mHC	7.30	61.73	70.89	56.75	85.60	41.11	60.06	27.30	33.97	54.68
3B	Attention Residuals	7.27	61.60	71.82	56.99	86.70	41.32	60.52	27.65	33.49	55.01
3B	DenseFormer	7.35	59.91	71.82	58.17	86.90	40.95	59.43	26.11	34.35	54.70
3B	Depth-Attention	7.25	62.25	72.36	60.38	88.00	41.88	60.14	25.51	35.02	55.69
Five-shot evaluation											
1.5B	Vanilla	–	48.71	69.48	53.28	90.10	36.60	57.95	25.09	31.96	51.65
1.5B	mHC	–	51.06	70.08	56.67	91.80	38.24	59.30	27.30	33.01	53.43
1.5B	Attention Residuals	–	53.13	70.08	56.43	91.90	38.55	60.19	26.02	33.01	53.66
1.5B	DenseFormer	–	52.71	71.16	56.67	91.00	38.37	58.46	26.45	34.45	53.66
1.5B	Depth-Attention	–	54.05	70.51	55.56	91.50	38.59	59.55	25.94	34.07	53.72
3B	Vanilla	–	53.23	71.16	55.01	91.20	39.79	61.53	27.90	33.49	54.16
3B	mHC	–	56.76	71.87	58.64	93.50	40.86	62.58	27.22	33.78	55.65
3B	Attention Residuals	–	55.73	72.63	57.46	92.70	41.07	64.31	28.67	34.83	55.92
3B	DenseFormer	–	54.12	72.47	59.27	91.90	40.71	63.80	27.99	34.45	55.59
3B	Depth-Attention	–	57.40	72.91	59.19	92.80	41.51	63.22	26.71	35.60	56.17

Table 1: Main results on Qwen3-style architectures. We report validation perplexity and downstream accuracy under zero-shot and five-shot evaluation settings. Since perplexity is independent of the shot setting, we report it only in the zero-shot block and use “–” in the five-shot block. All downstream numbers are accuracies in percentage. Avg. is averaged over the eight downstream tasks shown in the table.

Combined with the negligible extra arithmetic cost (+0.004% FLOPs, normalized by the standard dense one-token decoding estimate $2N$ (Kaplan et al., 2020)), Depth-Attention adds essentially no latency and no additional persistent inference state, consistent with the structural fact that it stores its depth-mixed values in the standard V-cache slot rather than in any new buffer or wider residual stream.

4.3 Scaling Behavior

We further evaluate whether the benefit of Depth-Attention persists across model scales. To this end, we train Qwen-style decoder architectures at multiple model sizes under the same pretraining setting, and compare Depth-Attention with the corresponding vanilla Transformer baseline. Detailed model configurations and optimization hyperparameters are provided in Appendix C.

As shown in Figure 2, Depth-Attention consistently achieves lower validation loss than vanilla Transformers across all tested model sizes. This improvement remains stable from 360M to 3B parameters, indicating that the benefit of depth-

wise value mixing persists as the model scales up. The advantage also does not vanish when the training-data budget grows: pretraining the 3B models for 60B tokens—the compute-optimal budget suggested by Hoffmann et al. (2022)—Depth-Attention still lowers perplexity from 7.08 to 6.90 and improves the eight-task average by 1.16 points (Appendix E). Moreover, the gains are not specific to the Qwen-style architecture: Depth-Attention yields comparable validation-loss reductions on LLaMA-style decoders under matched settings (Appendix H). These results suggest that Depth-Attention improves the scaling behavior of Transformer decoders by enriching current-layer value states with information from earlier depths.

4.4 Analysis of Depth-Attention Weights

To examine whether Depth-Attention actually uses shallower value sources, we visualize the learned depth-attention weights of the trained 3B model in Figure 3. To quantify how each target layer uses sources in its depth source set, we calculate the average depth-attention weight for every target-source layer pair over tokens, heads, and evaluation

Method	Layer- ℓ Extra FLOPs	3B Extra FLOPs / Token	Layer- ℓ Extra I/O	3B Extra I/O / Token	128K Prefill Extra Memory
Depth-Attention	$O((\ell/s + 1)d_{kv})$	2.40×10^5	$O((\ell/s + 1)d_{kv})$	282 KiB	0 GiB
DenseFormer	$O(\ell d_{model})$	5.01×10^6	$O(\ell d_{model})$	4.97 MiB	24.0 GiB
Attention Residuals	$O(R_\ell d_{model})$	4.26×10^6	$O(R_\ell d_{model})$	4.44 MiB	4.0 GiB
mHC	$O(N^3 d_{model})$	4.72×10^7	$O(N d_{model})$	8.63 MiB	1.5 GiB

Table 2: Theoretical comparison of additional computation, activation I/O, and long-context memory beyond the vanilla Transformer decoder. The FLOPs and I/O formulas report the extra per-token overhead at target layer ℓ , while the 3B columns instantiate the total overhead for one token over the full model. Extra I/O denotes additional activation read/write over the vanilla decoder; for methods that replace the standard residual connection, we subtract the vanilla residual I/O. The 128K-prefill memory column reports the extra persistent memory required to prefill a 128K-token context. The estimates use our 3B Qwen3-style configuration with $L = 48$, $d_{model} = 2048$, $d_{kv} = 512$, and Depth-Attention stride $s = 24$. Here R_ℓ is the number of residual blocks used by Attention Residuals at layer ℓ , and N is the number of mHC streams. Detailed derivations are provided in Appendix B.

Method	Seconds /step	Time overhead	Tokens/s /GPU
Vanilla	2.335 ± 0.382	—	14,032
Depth-Attention	2.548 ± 0.383	+9.12%	12,860
DenseFormer	3.449 ± 0.559	+47.71%	9,501
Attention Residuals	3.684 ± 1.317	+57.77%	8,895
mHC	9.774 ± 1.190	+318.59%	3,353

Table 3: Per-step training wall-clock and per-GPU token throughput on the 3B Qwen3-style configuration under a common reference PyTorch implementation (no custom kernels), measured in a fully matched setting: identical microbatch size, gradient accumulation, global batch size, sequence length, data, optimizer, hardware, and software stack for every method; we report mean and step-level standard deviation over 200 steps after warm-up. Among cross-layer mechanisms, Depth-Attention is the only one whose training overhead stays below 10%; all others incur 48%–319%.

samples.

As shown in Figure 3, the learned weights are not concentrated solely on the current-layer source. Besides the diagonal pattern, deeper target layers assign clear weight to shallower source layers, with a particularly visible band around the intermediate layers. This shows that Depth-Attention does not collapse to the vanilla current-layer computation; instead, it learns to enrich the current-layer value state with information from earlier depths.

4.5 Depth-Attention in Looped Transformers

Looped, or recurrent, Transformers have recently attracted renewed interest as a way to increase effective depth through parameter sharing rather than by adding new layers (Giannou et al., 2023; Zeng et al., 2026; Bae et al., 2025). Depth-Attention applies directly to this setting: when a layer is ex-

Method	Extra FLOPs	Inference Time (s)	Prefill Memory (GB)
Vanilla	0	526.6	30.6
Depth-Attention	2.40×10^5	532.8	30.6
Overhead	+0.004%	+1.18%	+0.000

Table 4: Inference efficiency on the 3B Qwen3-style model. Inference time is measured with batch size 64, prefill length 2048, and decode length 2048, including both prefill and decoding. Prefill memory reports peak GPU memory during prefill. Extra FLOPs denotes the additional depth-wise attention FLOPs per token; the overhead is normalized by the standard dense one-token decoding estimate $2N$.

ecuted repeatedly, the value states it produced at earlier loop steps serve as additional depth sources, which the current step mixes into its value before standard causal self-attention. On a 500M looped Transformer with three loops, adding Depth-Attention lowers the validation loss from 2.208 to 2.194, showing that depth-wise value mixing remains effective when depth is induced by recurrence rather than by stacking layers.

5 Ablation Studies

We conduct controlled ablation studies using the same 500M Qwen-style configuration and training setup as in subsection 4.3. All other settings are kept unchanged, allowing us to isolate which design choices of Depth-Attention contribute to the observed improvement. Specifically, we study three factors: the construction of the depth source list, the depth-mixing rule, and the attention states updated by depth mixing. To verify that the conclusions drawn at this scale remain stable at the main experimental scales, we additionally repeat the

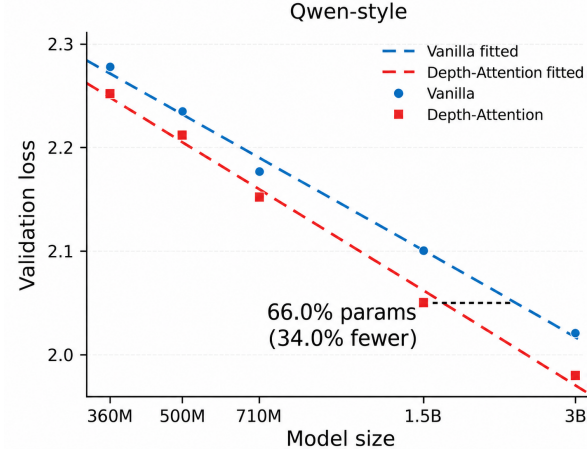


Figure 2: Scaling behavior on Qwen-style architectures. Depth-Attention consistently shifts the validation-loss scaling curve downward compared with vanilla Transformers. Dashed lines denote fitted scaling trends, and the black dashed segment indicates the estimated parameter reduction required for vanilla baseline to match the validation loss of the 1.5B Depth-Attention.

depth-mixing-object ablation on the 1.5B model; the conclusion is unchanged, with value-only mixing again performing best (Appendix I).

Stride. We first study how the density of the depth source list affects performance. Specifically, we compare four Depth-Attention variants: using only the first-layer source, the default half-depth stride, a denser quarter-depth stride, and the full-source setting with stride 1. This ablation tests whether exposing each layer to more historical value states leads to consistent gains, and whether a sparse source list is sufficient to capture most of the benefit. As shown in Figure 4, the half-depth stride achieves the lowest validation loss. Using only the first-layer source performs slightly worse, suggesting that an overly sparse source list does not provide enough useful intermediate-depth information. However, making the source list denser does not further improve performance: both the quarter-depth and full-source settings yield slightly higher validation loss than the half-depth setting. This suggests that overly dense source lists may introduce redundant or less useful historical value states. Overall, the results indicate that both too few and too many depth sources are suboptimal, while a moderate source density works best. For this reason, we use the half-depth stride as the default setting. All four depth-mixing variants still outperform vanilla attention (validation loss 2.2348), indicating that the benefit of depth-wise value reuse

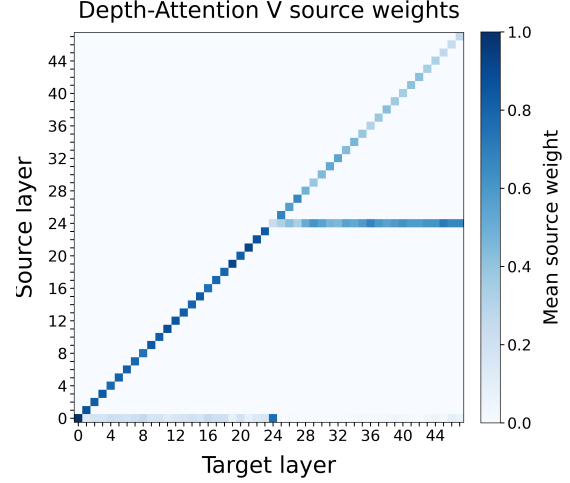


Figure 3: Visualization of learned depth-attention weights in the trained 3B Depth-Attention model. Each entry denotes the average depth-attention weight from a target layer to a source layer. Off-diagonal weights indicate that the model uses shallower value sources when constructing depth-mixed value states.

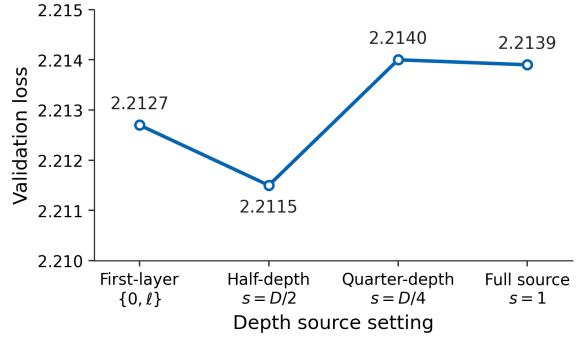


Figure 4: Ablation on the stride of the depth source list using the 500M model. The half-depth stride achieves the lowest validation loss, while both overly sparse and overly dense source lists perform slightly worse.

is robust across different source-list designs.

Depth-mixing rule. We next ablate the rule used to mix value states across the depth source list. Keeping the same source-list configuration, we compare vanilla attention, a uniform-mixing variant that simply averages all source values, and Depth-Attention, which computes per-head softmax weights over the depth source list. This ablation examines whether adaptive depth-wise value mixing is more effective than merely exposing previous value states with fixed uniform weights. As shown in Table 5, uniform mixing already improves over the vanilla baseline, indicating that reusing previous value states is beneficial. Depth-Attention further reduces the validation loss, showing that

Method	Mixing rule	Val. loss
Vanilla	–	2.2348
Uniform Mix	Uniform weights	2.2153
Depth-Attention	Per-head softmax mixing	2.2115

Table 5: Ablation on the depth-mixing rule using the 500M model. Uniform Mix and Depth-Attention use the same depth source list, while differing only in how source values are mixed.

Method	Updated state	Val. loss
Vanilla	–	2.2348
Key-only	K_{mix}	2.2354
Key-value	$K_{\text{mix}}, V_{\text{mix}}$	2.2198
Value-only	V_{mix}	2.2115

Table 6: Ablation on the attention state updated by depth mixing using the 500M model. All variants use the same depth-routing weights, but apply them to different attention states.

per-head softmax depth mixing is more effective than a fixed average over the same source list.

Depth-mixing object. Finally, we ablate which attention states should be updated by depth mixing. Besides the proposed value-only update, we compare key-only and key-value variants, where the same depth-routing weights are used to construct mixed key states. This ablation tests whether Depth-Attention should modify the keys used for temporal attention, or instead preserve the current-layer keys while reusing historical value states. As shown in Table 6, value-only mixing achieves the best validation loss, showing that using values for depth mixing is beneficial. In contrast, mixing keys either hurts performance or weakens the gain, suggesting that Depth-Attention should preserve current-layer keys while updating only values.

6 Conclusion

We introduced Depth-Attention, a simple way to bring adaptive, content-based selection to the depth dimension of Transformer decoders. Instead of modifying the residual stream, each layer’s query attends along depth over the keys of earlier layers and mixes their values into the value state that self-attention reads, reusing the queries, keys, and values the model already computes. This makes cross-layer reuse attention-native: it adds no parameters, keeps standard key-value cache as its only persistent inference state (the same size as a vanilla decoder’s, in contrast to methods that widen the residual stream or retain earlier-layer hidden

states), and leaves the causal attention pattern unchanged. On Qwen3-style decoders at 1.5B and 3B parameters, Depth-Attention attains the best results over the vanilla Transformer and strong cross-layer baselines—at a small fraction of their compute and memory cost; the benefits hold across model scales, persist under a larger 60B-token training budget, transfer to LLaMA-style decoders, extend to looped Transformers, and carry over to instruction following and mathematical reasoning after supervised fine-tuning.

Acknowledgement

This work is sponsored by the National Natural Science Foundation of China (NSFC) grant (No. 62576211) and the National Key Research and Development Program of China (No. 2023ZD0121402). This work is also supported by the Specialized Program on Fundamental Research from Science and Technology Commission of Shanghai Municipality (No. 2025SHZDZX025G09).

Limitations

Our study has several limitations. First, although Depth-Attention adds negligible FLOPs and no persistent state beyond the key-value cache in theory, our current implementation is not fused into the attention kernel, so the measured training and inference overhead is larger than the asymptotic analysis suggests; a dedicated kernel would likely recover most of this gap. Second, our evaluation mainly covers dense Qwen-style decoders up to 3B parameters trained for up to 60B tokens, together with smaller LLaMA-style models (360M and 500M); whether the gains persist on substantially larger models and sparse mixture-of-experts architectures remains to be confirmed. These restrictions are primarily due to our limited computational resources rather than any inherent constraint of the method. Finally, we mix only value states and select depth sources with a fixed stride; learned source selection and richer cross-layer interactions remain open, although our ablations indicate that value-only mixing is the most effective variant we examined.

Ethical Considerations

This work proposes an architectural modification to improve the performance and efficiency of Transformer language models. The method does not involve human-subject studies, private user data, or

the collection of new datasets, and it is not designed for sensitive applications such as surveillance, profiling, or decision making about individuals. Since Depth-Attention only changes how intermediate value states are reused across layers, it does not introduce additional ethical risks beyond those already associated with standard language models. Nevertheless, systems using it should be evaluated for bias, harmful outputs, and misuse before deployment.

References

- Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. [GQA: Training generalized multi-query transformer models from multi-head checkpoints](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.
- Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. 2021. [ReZero is all you need: Fast convergence at large depth](#). In *Uncertainty in artificial intelligence*, pages 1352–1361. PMLR.
- Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyouon Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. 2025. [Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation](#). *Advances in Neural Information Processing Systems*, 38.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. [PIQA: Reasoning about physical commonsense in natural language](#). In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think you have solved question answering? try ARC, the AI2 reasoning challenge](#). *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- DeepSeek-AI. 2024. [DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model](#). *arXiv preprint arXiv:2405.04434*.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. [The Pile: An 800GB dataset of diverse text for language modeling](#). *arXiv preprint arXiv:2101.00027*.
- Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. 2023. [Looped transformers as programmable computers](#). In *International Conference on Machine Learning*, pages 11398–11442. PMLR.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. [Deep residual learning for image recognition](#). In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2021. [RealFormer: Transformer likes residual attention](#). In *Findings of the association for computational linguistics: ACL-IJCNLP 2021*, pages 929–943.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. [Training compute-optimal large language models](#). *arXiv preprint arXiv:2203.15556*.
- Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. [Densely connected convolutional networks](#). In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *arXiv preprint arXiv:2001.08361*.
- Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, and 1 others. 2026. [Attention residuals](#). *arXiv preprint arXiv:2603.15031*.
- Jonas Knupp, Jan Hendrik Metzen, Jeremias Bohn, Georg Groh, and Kristian Kersting. 2026. [Depth-recurrent attention mixtures: Giving latent reasoning the attention it deserves](#). *arXiv preprint arXiv:2601.21582*.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. [RACE: Large-scale reading comprehension dataset from examinations](#). *arXiv preprint arXiv:1704.04683*.
- Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. 2025. [LAuReL: Learned augmented residual layer](#). In *Proceedings of the 42nd International Conference on Machine Learning*, pages 43826–43836. PMLR.
- Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. 2024. [DenseFormer: Enhancing information flow in transformers via depth weighted averaging](#). *Advances in neural information processing systems*, 37:136479–136508.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. [The LAMBADA dataset: Word prediction requiring a broad discourse context](#). In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1525–1534.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for*

- Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. [WinoGrande: An adversarial winograd schema challenge at scale](#). *Communications of the ACM*, 64(9):99–106.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [LLaMA: Open and efficient foundation language models](#). *arXiv preprint arXiv:2302.13971*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *Advances in neural information processing systems*, 30.
- Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. 2024. [DeepNet: Scaling transformers to 1,000 layers](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774.
- Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. [Crowdsourcing multiple choice science questions](#). *arXiv preprint arXiv:1707.06209*.
- Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. 2025. [MUDDFormer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections](#). In *Proceedings of the 42nd International Conference on Machine Learning*, pages 68440–68458. PMLR.
- Shufang Xie, Huishuai Zhang, Junliang Guo, Xu Tan, Jiang Bian, Hany Hassan Awadalla, Arul Menezes, Tao Qin, and Rui Yan. 2023. [ResiDual: Transformer with dual residual connections](#). *arXiv preprint arXiv:2304.14802*.
- Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, and 1 others. 2025. [mHC: Manifold-constrained hyper-connections](#). *arXiv preprint arXiv:2512.24880*.
- Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. [On layer normalization in the transformer architecture](#). In *International conference on machine learning*, pages 10524–10533. PMLR.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) *arXiv preprint arXiv:1905.07830*.
- Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhiyu Li, and Zhouhan Lin. 2026. [PonderLM: Pretraining language models to ponder in continuous space](#). In *The Fourteenth International Conference on Learning Representations*.
- Yilang Zhang, Bingcong Li, Niao He, and Georgios B Giannakis. 2026. [ANCR: Adaptive neural connection reassignment for efficient depth scaling](#). *arXiv preprint arXiv:2602.09009*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-Bench and chatbot arena](#). In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*.
- Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, Fares Obeid, and Zhenzhong Lan. 2025. [Value residual learning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28341–28356.
- Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. 2025. [Hyper-connections](#). In *The Thirteenth International Conference on Learning Representations*.
- Lianghui Zhu, Yuxin Fang, Bencheng Liao, Shijie Wang, Tianheng Cheng, Zilong Huang, Chen Chen, Lai Wei, Yutao Zeng, Ya Wang, Yi Lin, Yu Li, and Xinggang Wang. 2026. [Mixture-of-depths attention](#). *arXiv preprint arXiv:2603.15619*.

A Main Experiment Details

For the main experiments in Section 4, we provide the model configurations and training hyperparameters used for the 1.5B and 3B Qwen3-style models in Table 7. The baseline-specific settings are summarized in Table 8. For each model scale, all methods use the same base architecture and training setup, differing only in the corresponding cross-layer mechanism.

We choose the baseline-specific hyperparameters by following the recommended or commonly used practical configurations in the corresponding papers, while keeping the base architecture, data, optimizer, training budget, sequence length, and precision fixed across methods. For DenseFormer (Pagliardini et al., 2024), we use the full 1×1 setting, where a DWA module is applied after every Transformer block and aggregates the current block output, all previous block outputs, and the input embedding. For Attention Residuals (Kimi Team et al., 2026), we use the practical Block AttnRes variant with eight stored block representations. In the Attention Residuals implementation, the block size counts attention and MLP residual sublayers; since each Transformer layer contains two such residual sublayers, our 48-layer model with block size 12 gives $2L/12 = 8$ blocks, matching the recommended practical setting. For mHC (Xie et al., 2025), we use the expansion rate $n = 4$, following the standard mHC configuration. These choices give each baseline a strong practical configuration from its original design, rather than weakening the baselines through additional sparsification or smaller cross-layer state.

B Overhead Derivations and Benchmark Details

This appendix provides the derivations for the theoretical overhead estimates in Table 2 and the empirical efficiency measurements in Table 4. All theoretical estimates report the additional cost beyond a vanilla Transformer decoder.

B.1 Common Counting Assumptions

We use the following notation. Let L be the number of Transformer layers, d_{model} be the hidden dimension, and d_{kv} be the total key-value dimension used by the grouped-query attention cache. For our 3B Qwen3-style configuration, we use

$$L = 48, \quad d_{\text{model}} = 2048, \quad d_{\text{kv}} = 512.$$

For long-context memory estimates, we use a pre-fill length of

$$T = 128K = 131072$$

tokens. All memory and activation-I/O estimates assume 16-bit activations, so each scalar takes

$$b = 2$$

bytes. This is consistent with the FP16 precision used in our experiments.

We count one multiply-add as two FLOPs. We focus on the dominant vector dot products and weighted sums, and omit lower-order scalar operations such as the softmax normalization over a small number of depth sources. For the normalized FLOPs overhead in Table 4, we follow the standard scaling-law compute approximation that dense language-model training costs about $6ND$ FLOPs for N parameters and D tokens (Kaplan et al., 2020). Since training consists of a forward pass and a backward pass, this corresponds to a dense forward cost of about $2N$ FLOPs per token. Therefore, for a 3B model, the standard dense one-token forward/decode estimate is

$$2N = 2 \times 3 \times 10^9 = 6 \times 10^9.$$

B.2 Depth-Attention

For a target layer ℓ , Depth-Attention mixes value states from a depth source set $\mathcal{D}_\ell$. Let

$$M_\ell = |\mathcal{D}_\ell|$$

denote the number of depth sources at layer ℓ . In our implementation, the depth source set is maintained with stride $s = 24$, giving the layer-wise asymptotic cost

$$O((\ell/s + 1)d_{\text{kv}}).$$

At each target layer, Depth-Attention performs two dominant operations along the depth dimension. First, it computes depth-wise scores between the current query and the source keys:

$$q_\ell K_{\mathcal{D}_\ell}^\top,$$

which costs approximately

$$2M_\ell d_{\text{kv}}$$

FLOPs. Second, it computes the weighted sum over source values:

$$\alpha_\ell V_{\mathcal{D}_\ell},$$

Configuration / Hyperparameter	1.5B	3B
<i>Model configuration</i>		
Architecture	Qwen3-style decoder	Qwen3-style decoder
Hidden size	1536	2048
Intermediate size	4096	6912
Number of hidden layers	48	48
Number of attention heads	24	32
Number of key-value heads	6	8
GQA group size	4	4
Activation function	SiLU	SiLU
Vocabulary size	50304	50304
Maximum sequence length	2048	2048
Maximum position embeddings	2048	2048
RMSNorm epsilon	1.0×10^{-5}	1.0×10^{-5}
Initializer range	0.01	0.01
Tie word embeddings	False	False
BOS / EOS token id	0 / 0	0 / 0
Padding token id	1	1
Model dtype	FP16	FP16
Depth-Attention stride	24	24
<i>Training hyperparameters</i>		
Training data	The Pile	The Pile
Training tokens	32B	32B
Training steps	15625	15625
Global batch size	1024	1024
Sequence length	2048	2048
Optimizer	AdamW	AdamW
Learning rate	1.0×10^{-3}	1.0×10^{-3}
Learning rate schedule	cosine decay	cosine decay
Warmup ratio	0.02	0.02
Adam β_1	0.9	0.9
Adam β_2	0.95	0.95
Weight decay	0.1	0.1
Gradient clipping	1.0	1.0
Precision	FP16	FP16

Table 7: Model configurations and training hyperparameters for the main 1.5B and 3B Qwen3-style experiments. Depth-Attention-specific entries apply only to our method; the vanilla Transformer and cross-layer baselines use the same base architecture with their corresponding mechanism enabled.

which also costs approximately

$$2M_\ell d_{\text{kv}}$$

FLOPs. Therefore, the extra FLOPs at layer ℓ are

$$C_\ell^{\text{Depth}} \approx 4M_\ell d_{\text{kv}}.$$

Using the exact depth source schedule of our 3B model, we have

$$\sum_{\ell=1}^L M_\ell = 117.$$

Thus, the total extra FLOPs are

$$\begin{aligned}
C^{\text{Depth}} &= 4d_{\text{kv}} \sum_{\ell=1}^L M_\ell \\
&= 4 \times 512 \times 117 \\
&= 2.40 \times 10^5,
\end{aligned} \tag{5}$$

which gives 2.40×10^5 extra FLOPs per token.

For activation I/O, Depth-Attention reads the source keys and source values and writes the mixed value state. Thus the extra I/O at layer ℓ is counted as

$$B_\ell^{\text{Depth}} \approx b d_{\text{kv}} (2M_\ell + 1).$$

Setting	Vanilla	Depth-Attention	DenseFormer	Attention Residuals	mHC
Cross-layer mechanism	–	Depth-wise value mixing	Dense hidden-state reuse	Block-wise residual aggregation	Residual-stream mixing
Main hyperparameter	–	Stride $s = 24$	Depth-weighted averaging	Block size 12	4 streams
Base architecture			Same as Table 7		
Training setup		Same data, token budget, optimizer, schedule, and hyperparameters			

Table 8: Baseline-specific settings used in the main experiments. All methods share the same base model configurations and training hyperparameters in Table 7. mHC denotes manifold-constrained hyper-connections.

Summing over all layers gives

$$\begin{aligned}
B^{\text{Depth}} &= b d_{\text{kv}} \left(2 \sum_{\ell=1}^L M_{\ell} + L \right) \\
&= 2 \times 512 \times (2 \times 117 + 48) \\
&= 288768 \text{ bytes} \\
&\approx 282 \text{ KiB}.
\end{aligned} \tag{6}$$

Depth-Attention does not require storing additional hidden states during inference. The mixed value state replaces the standard value state in the ordinary key-value cache, and the current-layer key is unchanged. Therefore, the extra persistent memory for a 128K-token prefill is

$$0 \text{ GiB}.$$

B.3 DenseFormer

DenseFormer-like dense cross-layer mechanisms aggregate hidden states from many previous layers. For a target layer ℓ , we count the number of accessible hidden states as approximately $\ell + 1$, giving the layer-wise cost

$$O(\ell d_{\text{model}}).$$

The main computation is the weighted aggregation of d_{model} -dimensional hidden states. The per-layer FLOPs are

$$C_{\ell}^{\text{Dense}} \approx 2(\ell + 1)d_{\text{model}}.$$

We define

$$S_{\text{Dense}} = \sum_{\ell=1}^L (\ell + 1) = 1224.$$

The total extra FLOPs are then

$$\begin{aligned}
C^{\text{Dense}} &= 2d_{\text{model}}S_{\text{Dense}} \\
&= 2 \times 2048 \times 1224 \\
&= 5.01 \times 10^6,
\end{aligned} \tag{7}$$

which gives 5.01×10^6 extra FLOPs per token.

For activation I/O, DenseFormer reads the previous hidden states and writes the aggregated hidden state. Thus,

$$\begin{aligned}
B^{\text{Dense}} &= b d_{\text{model}}(S_{\text{Dense}} + L) \\
&= 2 \times 2048 \times (1224 + 48) \\
&= 5.21 \times 10^6 \text{ bytes} \\
&\approx 4.97 \text{ MiB}.
\end{aligned} \tag{8}$$

For long-context prefill, DenseFormer must keep the hidden states of all layers. The extra persistent memory is therefore

$$M^{\text{Dense}} = T L d_{\text{model}} b.$$

Plugging in $T = 131072$, $L = 48$, $d_{\text{model}} = 2048$, and $b = 2$, we obtain

$$\begin{aligned}
M^{\text{Dense}} &= 131072 \times 48 \times 2048 \times 2 \text{ bytes} \\
&= 24.0 \text{ GiB}.
\end{aligned} \tag{9}$$

B.4 Attention Residuals

Attention Residuals attends over a set of residual blocks at each target layer. Let R_{ℓ} denote the number of residual blocks used at layer ℓ . Its layer-wise cost is therefore

$$O(R_{\ell} d_{\text{model}}).$$

The dominant operations are residual-block scoring and the weighted sum over residual representations. We count these as

$$C_{\ell}^{\text{ResAttn}} \approx 4R_{\ell} d_{\text{model}}.$$

We denote the total number of residual blocks used across all target layers by

$$S_{\text{Res}} = \sum_{\ell=1}^L R_{\ell} = 520.$$

The total extra FLOPs are

$$\begin{aligned} C^{\text{ResAttn}} &= 4d_{\text{model}}S_{\text{Res}} \\ &= 4 \times 2048 \times 520 \\ &= 4.26 \times 10^6, \end{aligned} \quad (10)$$

which gives 4.26×10^6 extra FLOPs per token.

The extra activation I/O is computed by counting the residual-block states read by the attention module and the additional residual-attention outputs written by the module. After subtracting the standard vanilla residual I/O, we obtain

$$\begin{aligned} B^{\text{ResAttn}} &\approx bd_{\text{model}}(2S_{\text{Res}} + 2L) \\ &= 2 \times 2048 \times (2 \times 520 + 2 \times 48) \\ &= 4.65 \times 10^6 \text{ bytes} \\ &\approx 4.44 \text{ MiB}. \end{aligned} \quad (11)$$

For the 128K-prefill memory estimate, Attention Residuals stores a fixed set of residual-block hidden states. In our comparison, the number of stored residual blocks is $R = 8$, so the extra persistent memory is

$$M^{\text{ResAttn}} = TRd_{\text{model}}b.$$

Plugging in $T = 131072$, $R = 8$, $d_{\text{model}} = 2048$, and $b = 2$, we obtain

$$\begin{aligned} M^{\text{ResAttn}} &= 131072 \times 8 \times 2048 \times 2 \text{ bytes} \\ &= 4.0 \text{ GiB}. \end{aligned} \quad (12)$$

B.5 mHC

For mHC, let n denote the total number of residual streams. Following the standard mHC configuration, we use an expansion rate of $n = 4$, i.e., the model maintains four residual streams in total. The method introduces interactions among these streams, giving a layer-wise stream-mixing computation cost of

$$O(n^3d_{\text{model}})$$

and an activation I/O cost of

$$O(nd_{\text{model}}).$$

Using the operation count of the mHC stream-mixing module under this configuration, the extra FLOPs over the full model are

$$C^{\text{mHC}} \approx 4.72 \times 10^7$$

extra FLOPs per token. The corresponding activation I/O count gives

$$B^{\text{mHC}} \approx 8.63 \text{ MiB}$$

extra I/O per token.

For long-context prefill, mHC needs to keep multiple residual streams. Since a vanilla Transformer already has one residual stream, the extra persistent memory over vanilla comes from the additional $n - 1$ streams:

$$M^{\text{mHC}} = T(n - 1)d_{\text{model}}b.$$

With $T = 131072$, $n = 4$, $d_{\text{model}} = 2048$, and $b = 2$,

$$M^{\text{mHC}} = 131072 \times (4 - 1) \times 2048 \times 2 = 1.5 \text{ GiB}.$$

Thus, although mHC maintains four residual streams in total, its reported extra memory over a vanilla Transformer corresponds to the three additional streams.

B.6 Empirical Benchmark Details

We further measure the practical efficiency of the trained 3B Qwen3-style model. All empirical numbers in Table 4 are measured on trained checkpoints under the same hardware and software environment.

For inference, we use batch size 64, prefill length 2048, and decode length 2048. We use one warmup run and two repeated measurement runs. The table reports the total wall-clock generation time, including both prefill and decoding. The vanilla model takes

$$526.6 \text{ s},$$

while Depth-Attention takes

$$532.8 \text{ s}.$$

The relative inference overhead is therefore

$$\frac{532.8 - 526.6}{526.6} \times 100\% = 1.18\%.$$

The prefill peak memory is

$$30.6 \text{ GB}$$

for both vanilla and Depth-Attention, so the reported prefill-memory overhead is

$$0.0 \text{ GB}.$$

Finally, the extra FLOPs entry in Table 4 uses the theoretical Depth-Attention estimate derived above:

$$2.40 \times 10^5$$

extra FLOPs per token. The overhead row normalizes this value by the standard dense one-token forward estimate $2N$ for a 3B model:

$$\frac{2.40 \times 10^5}{2 \times 3 \times 10^9} = 4.0 \times 10^{-5} = 0.004\%.$$

This explains why the theoretical arithmetic overhead is negligible even though the measured training wall-clock overhead is larger: the practical overhead is mainly caused by additional activation reads/writes, kernel launches, and non-fused depth-wise operations rather than by dense matrix-multiplication FLOPs.

C Scaling Experiment Details

For the scaling experiments in [subsection 4.3](#), we provide the model configurations and training hyperparameters used for the Qwen-style scaling analysis in [Table 9](#). For each model size, Depth-Attention and the vanilla Transformer use the same base configuration and optimization setup, differing only in whether depth-wise value mixing is enabled. The Qwen-style scaling experiments include 360M, 500M, and 710M models. The earlier 160M setting is not included in the final scaling analysis. The larger 1.5B and 3B Qwen3-style models follow the configurations and training setup in [Table 7](#).

D Statistical Significance of the Main Results

To assess the robustness of the improvements reported in [Table 1](#), we conduct paired statistical analyses using the per-example predictions from the 3B evaluations. Specifically, we use a task-stratified paired bootstrap with 10,000 resamples to construct 95% confidence intervals, together with a paired permutation test using 10,000 permutations. All reported differences are computed from the average score across the evaluated tasks. The results are summarized in [Table 10](#).

Depth-Attention significantly outperforms the vanilla Transformer in both the zero-shot and five-shot settings. Compared with DenseFormer, the improvement is also statistically significant in the zero-shot setting. In the five-shot setting, Depth-Attention maintains a positive gain of +0.58 pp, with the confidence interval only narrowly crossing zero, providing suggestive evidence that the improvement remains positive.

E Pretraining with a 60B-Token Budget

The main experiments train all models for 32B tokens. To examine whether the advantage of Depth-Attention persists under a substantially larger training-data budget, we additionally train the 3B vanilla Transformer and 3B Depth-Attention models from scratch for 60B tokens—approximately the compute-optimal budget for a 3B model suggested by the scaling analysis of [Hoffmann et al. \(2022\)](#)—under identical data, optimization, and model settings. We evaluate the resulting checkpoints using both language-modeling perplexity and the same downstream benchmark suite as in the main paper. The results are shown in [Table 11](#).

After 60B tokens of pretraining, Depth-Attention continues to outperform the vanilla Transformer. It reduces perplexity by 0.18 and improves the eight-task downstream average by 1.16 points, improving six tasks, tying on PIQA, and trailing only on WinoGrande. These results demonstrate that the advantage of Depth-Attention is not restricted to the early stage of pretraining and remains observable at the 60B-token training budget; together with the stable gains across model sizes in [Figure 2](#), they indicate that the benefit has not saturated when scaling either the model size or the amount of training data.

F Instruction Tuning and MT-Bench Evaluation

To examine whether the pretraining advantage of Depth-Attention is retained after instruction tuning, we perform controlled full-parameter supervised fine-tuning (SFT) of the 3B vanilla Transformer and Depth-Attention models pretrained for 60B tokens ([Appendix E](#)). Both models are fine-tuned on the same 51,699-example Alpaca dataset ([Taori et al., 2023](#)) for three epochs, with identical optimization, batch, prompt-template, seed, and generation settings. We then evaluate both models on the official 80-question MT-Bench set ([Zheng et al., 2023](#)), scoring all responses with the same GPT-5.4 judge under identical judging prompts. [Figure 5](#) visualizes the per-category scores, and [Table 12](#) lists the exact numbers.

Depth-Attention improves the overall MT-Bench average from 2.18 to 2.66 (+0.48), with gains in seven of the eight categories and a tie on Writing. The largest improvement occurs in Reasoning (+2.05), followed by Humanities (+0.60), Math (+0.40), and Roleplay (+0.35). At the same time,

Configuration / Hyperparameter	360M	500M	710M
<i>Model configuration</i>			
Architecture	Qwen-style decoder	Qwen-style decoder	Qwen-style decoder
Hidden size	960	1152	1280
Intermediate size	2540	3072	3072
Number of hidden layers	24	24	32
Number of attention heads	15	18	20
Number of key-value heads	15	18	20
GQA group size	1	1	1
Activation function	SiLU	SiLU	SiLU
Vocabulary size	50304	50304	50304
Maximum sequence length	2048	2048	2048
Maximum position embeddings	2048	2048	2048
RMSNorm epsilon	1.0×10^{-5}	1.0×10^{-5}	1.0×10^{-5}
Initializer range	0.02	0.02	0.02
Tie word embeddings	False	False	False
BOS / EOS token id	0 / 0	0 / 0	0 / 0
Padding token id	1	1	1
Model dtype	FP16	FP16	FP16
Depth-Attention stride	12	12	16
<i>Training hyperparameters</i>			
Training data	The Pile	The Pile	The Pile
Training tokens	32B	32B	32B
Training steps	15625	15625	15625
Global batch size	1024	1024	1024
Sequence length	2048	2048	2048
Optimizer	AdamW	AdamW	AdamW
Learning rate	1.5×10^{-3}	1.0×10^{-3}	1.0×10^{-3}
Learning rate schedule	cosine decay	cosine decay	cosine decay
Warmup ratio	0.02	0.02	0.02
Adam β_1	0.9	0.9	0.9
Adam β_2	0.95	0.95	0.95
Weight decay	0.1	0.1	0.1
Gradient clipping	1.0	1.0	1.0
Precision	FP16	FP16	FP16

Table 9: Model configurations and training hyperparameters for the Qwen-style scaling experiments. Depth-Attention-specific entries apply only to our method; vanilla baselines use the same base configurations with depth-wise value mixing disabled. The 1.5B and 3B scaling points use the main-experiment setup in Table 7.

the results identify settings in which the benefit is limited: Writing remains unchanged, while STEM and Extraction show only modest gains, so the improvement is not uniform across generation categories. Together with the GSM8K results in Appendix G, these findings suggest that Depth-Attention provides particularly pronounced gains on reasoning-oriented tasks, and that its pretraining advantage is retained after instruction tuning rather than being restricted to pretrained-model evaluation.

G Mathematical Reasoning on GSM8K

To evaluate mathematical reasoning, we fine-tune the 3B vanilla Transformer and Depth-Attention checkpoints pretrained for 60B tokens

(Appendix E) on the official GSM8K training set (Cobbe et al., 2021) for three epochs using identical SFT hyperparameters, and evaluate the resulting models under both 0-shot and 8-shot settings. The results are shown in Table 13.

Depth-Attention achieves clearly higher GSM8K accuracy under both evaluation settings after the same task-specific fine-tuning (+3.41 points 0-shot and +2.96 points 8-shot), consistent with the pronounced Reasoning gains observed on MT-Bench (Appendix F).

H Transfer to LLaMA-Style Architectures

Depth-Attention does not rely on any Qwen-specific component: it operates only on the stan-

Setting	Comparison	Difference (pp)	95% bootstrap CI (pp)	Permutation (p)
3B zero-shot	Depth-Attention – Vanilla	+1.93	[+1.31, +2.57]	< 0.001
3B zero-shot	Depth-Attention – DenseFormer	+0.99	[+0.35, +1.64]	0.0021
3B five-shot	Depth-Attention – Vanilla	+2.01	[+1.40, +2.61]	< 0.001
3B five-shot	Depth-Attention – DenseFormer	+0.58	[−0.04, +1.20]	0.0643

Table 10: Paired statistical analyses of the 3B main results, based on per-example predictions. Confidence intervals use a task-stratified paired bootstrap with 10,000 resamples; p -values use a paired permutation test with 10,000 permutations. Differences are computed on the average score across the evaluated downstream tasks.

Metric / Task	Vanilla	Depth-Attention
Perplexity ↓	7.08	6.90
LAMBADA	62.72	65.28
PIQA	72.80	72.80
WinoGrande	60.70	60.22
SciQ	88.40	90.20
HellaSwag	43.53	44.20
ARC-Easy	62.25	63.34
ARC-Challenge	27.05	28.84
RACE	34.07	35.89
8-task average	56.44	57.60

Table 11: Results of pretraining the 3B models for 60B tokens under identical settings. Downstream numbers are zero-shot accuracies in percentage.

Category	Vanilla	Depth-Attn.	Difference
Writing	3.50	3.50	+0.00
Roleplay	2.75	3.10	+0.35
Reasoning	1.95	4.00	+2.05
Math	1.15	1.55	+0.40
Coding	1.45	1.70	+0.25
Extraction	1.80	1.95	+0.15
STEM	2.50	2.60	+0.10
Humanities	2.30	2.90	+0.60
Average	2.18	2.66	+0.48

Table 12: MT-Bench scores of the 3B models after identical Alpaca SFT, judged by the same GPT-5.4 judge on the official 80-question set.

dard query, key, and value states available in decoder-only Transformer layers. To examine whether the improvement is specific to the Qwen-style architecture, we additionally train Depth-Attention on LLaMA-style (Touvron et al., 2023) models at the 360M and 500M scales under matched training settings. The results are shown in Table 14.

Depth-Attention consistently improves the LLaMA-style models, with validation-loss reductions comparable to those observed on the Qwen-style models. These results provide direct evidence that its benefit is not specific to Qwen3 and trans-

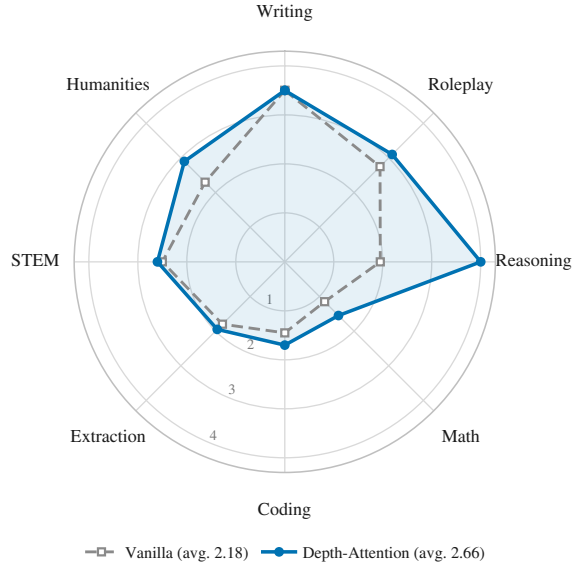


Figure 5: MT-Bench scores by category after identical Alpaca SFT of the 3B models pretrained for 60B tokens. Depth-Attention improves seven of the eight categories and ties on Writing, with the largest gain on Reasoning (+2.05).

Setting	Vanilla	Depth-Attn.	Improvement
0-shot	13.57	16.98	+3.41
8-shot	10.61	13.57	+2.96

Table 13: GSM8K accuracy (%) of the 3B models after identical fine-tuning on the official GSM8K training set for three epochs.

fers across different dense decoder architectures. Comprehensive validation on sparse mixture-of-experts architectures remains an important direction for future work.

I Depth-Mixing-Object Ablation at 1.5B

The controlled ablations in the main paper are conducted on the 500M model due to the substantial cost of fully pretraining every ablation variant at larger scales; this practice is also common in large-scale architecture studies. To verify that the design

Architecture	Size	Vanilla	Depth-Attn.	Diff.
Qwen-style	360M	2.278	2.252	-0.026
Qwen-style	500M	2.236	2.212	-0.024
LLaMA-style	360M	2.286	2.262	-0.024
LLaMA-style	500M	2.244	2.224	-0.020

Table 14: Validation language-modeling loss of vanilla and Depth-Attention models on Qwen-style and LLaMA-style architectures under matched training settings.

Variant	Validation loss
Vanilla	2.1005
Key-only	2.1162
Key-value	2.0804
Value-only	2.0503

Table 15: Depth-mixing-object ablation repeated on the 1.5B model. Value-only mixing remains the best variant, matching the conclusion obtained at 500M.

conclusion transfers to a larger scale, we repeat the depth-mixing-object ablation of Table 6 on the 1.5B model. All variants use identical training data, token budgets, hyperparameters, and evaluation settings; the only difference is the state updated by depth mixing. The results are shown in Table 15.

The conclusion remains unchanged at the 1.5B scale: value-only mixing achieves the lowest validation loss, outperforming vanilla, key-only, and key-value mixing. The advantage of updating only the value states is therefore preserved, rather than diluted, as the model scale increases.

J Comparison with MoDA

Mixture-of-Depths Attention (MoDA) (Zhu et al., 2026) is contemporaneous work that was developed independently of ours. Although both approaches enable adaptive cross-layer retrieval, their mechanisms differ substantially:

- Depth-Attention first mixes value states across depth and then applies unchanged standard causal self-attention, whereas MoDA jointly attends to sequence-level key-value states and historical depth-level key-value states within a single unified attention operator.
- Depth-Attention reuses the existing attention states and writes the mixed value into the ordinary V-cache slot, whereas default MoDA additionally projects FFN states into separate depth key-value representations.

Architectural cost comparison. Table 16 summarizes the resulting architectural costs. Here $m_\ell = \lceil \ell/s \rceil + 1$ denotes the number of depth sources at layer ℓ under this uniform approximation, and the numerical estimates use our 3B configuration with $L = 48$, $d_{\text{model}} = 2048$, $d_{\text{kv}} = 512$, stride $s = 24$, and 16-bit states; we count one multiply-add as two FLOPs, consistently with Table 2.² Under this configuration, Depth-Attention introduces no additional parameters or persistent inference state, requires approximately three orders of magnitude fewer extra FLOPs, and retains the standard causal-attention interface.

Matched empirical comparison. We further train 3B Depth-Attention and default MoDA models from scratch for 60B tokens using matched data, optimization, and evaluation settings. The results are shown in Table 17.

Under the same 60B-token pretraining budget, Depth-Attention achieves lower perplexity and outperforms MoDA on all eight downstream tasks, improving the macro-average accuracy by 1.05 points. Together with its substantially lower parameter, computation, activation-I/O, and persistent-memory overhead, these results indicate that Depth-Attention and MoDA have materially different mechanisms and that Depth-Attention provides a more favorable accuracy-efficiency trade-off under our matched evaluation.

K Systems-Level Analysis

This appendix analyzes how Depth-Attention interacts with large-scale training infrastructure, covering pipeline parallelism, tensor parallelism, activation checkpointing, and fused causal-attention kernels.

Pipeline parallelism. The main systems advantage of Depth-Attention is that it communicates only a sparse set of key-value states at the GQA KV-head width, rather than full-width hidden states or multiple residual streams. Let d denote the hidden size, d_{kv} the total KV-head dimension, and A_b the number of historical Depth-Attention sources crossing pipeline boundary b . Table 18 compares the maximum additional forward payload beyond a vanilla pipeline.

²The Depth-Attention totals here (2.42×10^5 FLOPs, 284 KiB) use the uniform source-count approximation $m_\ell = \lceil \ell/s \rceil + 1$, whereas Table 2 uses the exact per-layer source schedule ($\sum_\ell M_\ell = 117$), yielding the marginally smaller totals 2.40×10^5 and 282 KiB.

Aspect	Depth-Attention	Default MoDA
Extra parameters	None	$2Ld_{\text{model}}d_{\text{kv}} = 100.7\text{M}$
Specialized fused sequence–depth kernel	No; standard causal attention remains unchanged	Yes; its reported efficient implementation uses a specialized joint sequence–depth kernel
Layer- ℓ extra FLOPs	$4m_{\ell}d_{\text{kv}}$	$8\ell d_{\text{model}} + 4d_{\text{model}}d_{\text{kv}}$
3B extra FLOPs / token	2.42×10^5	2.20×10^8
Layer- ℓ extra activation I/O	$b(2m_{\ell} + 1)d_{\text{kv}}$	$b(4\ell d_{\text{kv}} + d_{\text{model}} + 2d_{\text{kv}})$
3B extra activation I/O / token	284 KiB	4.69 MiB
128K-prefill extra persistent memory	0 GiB	12.0 GiB

Table 16: Architectural cost comparison between Depth-Attention and default MoDA under our 3B configuration ($L = 48$, $d_{\text{model}} = 2048$, $d_{\text{kv}} = 512$, stride $s = 24$, 16-bit states), counting one multiply-add as two FLOPs.

Metric / Task	Depth-Attn.	MoDA	Diff.
Perplexity ↓	6.90	7.02	−0.12
LAMBADA	65.28	63.75	+1.53
PIQA	72.80	72.09	+0.71
WinoGrande	60.22	58.33	+1.89
SciQ	90.20	89.60	+0.60
HellaSwag	44.20	43.63	+0.57
ARC-Easy	63.34	62.25	+1.09
ARC-Challenge	28.84	27.47	+1.37
RACE	35.89	35.31	+0.58
8-task average	57.60	56.55	+1.05

Table 17: Matched 60B-token pretraining comparison between 3B Depth-Attention and default MoDA under identical data, optimization, and evaluation settings. Downstream numbers are zero-shot accuracies in percentage.

The maximum additional pipeline payload of Depth-Attention is approximately $47\times$ smaller than DenseFormer, $8\times$ smaller than Block AttnRes, and $3\times$ smaller than mHC. As in the cross-stage caching strategy used by Block AttnRes, each selected key-value source can be transmitted once and cached by downstream stages, avoiding repeated transfers to multiple target layers.

Tensor parallelism. Under standard head-wise tensor parallelism, each rank owns the query heads and corresponding GQA KV-head shards. Depth scoring and value mixing are therefore rank-local and introduce no additional tensor-parallel collective communication. The resulting mixed value then follows the unchanged causal-attention and output-projection tensor-parallel paths.

Activation checkpointing. Depth-Attention is compatible with standard activation checkpointing. The depth logits, softmax weights, and mixed-value intermediates can be discarded during the forward pass and recomputed during backward, while only

the sparse source key-value dependencies must remain accessible.

Fused causal attention. Depth-Attention leaves the causal-attention query, key, mask, and kernel interface unchanged. Existing fused causal-attention kernels can therefore be used without modification, with depth mixing implemented as a lightweight preceding operation.

Empirical profiling. To verify these properties empirically, we profiled the 3B vanilla Transformer and Depth-Attention models under matched training settings; the results are shown in Table 19.

Activation checkpointing remains fully effective: when checkpointing is enabled, Depth-Attention adds only 0.26% peak memory relative to vanilla, while maintaining a throughput overhead comparable to that in the non-checkpointed setting. The fused causal-attention kernel runtime also remains comparable, confirming that Depth-Attention does not cause a fallback from the existing optimized causal-attention kernel. Overall, Depth-Attention is directly compatible with activation checkpointing, head-wise tensor parallelism, and existing fused causal-attention implementations; pipeline parallelism introduces sparse cross-stage key-value dependencies, but their communication volume is tightly bounded by the small KV width and sparse source schedule and can be further reduced through incremental cross-stage caching.

L Wall-Clock Measurement Protocol

The training wall-clock comparison in Table 3 uses the following protocol. To eliminate differences in batch hyperparameters, all methods are rerun on larger-memory GPUs using exactly the same microbatch size, gradient accumulation, global batch size, sequence length, training data, optimizer settings, hardware, and software stack, on

Method	Additional cross-stage payload	Maximum in our 3B configuration	2K-token sequence
Depth-Attention	$2A_b d_{kv}$, with $A_b \leq 2$	$4d_{kv} = d$ (4 KiB/token)	8 MiB
DenseFormer	$(L - 1)d$	$47d$ (188 KiB/token)	376 MiB
Block AttnRes ($N = 8$)	Nd	$8d$ (32 KiB/token)	64 MiB
mHC ($n = 4$)	$(n - 1)d$	$3d$ (12 KiB/token)	24 MiB

Table 18: Maximum additional forward pipeline payload beyond a vanilla pipeline, using our 3B configuration with $L = 48$, $d = 2048$, $d_{kv} = 512$, stride $s = 24$, and FP16 communication.

System setting	Metric	Vanilla	Depth-Attention	Difference
Checkpointing off	Peak memory (GiB/GPU)	122.60	127.53	+4.02%
Checkpointing off	Throughput (tokens/s/GPU)	17,161.97	15,719.50	−8.41%
Checkpointing on	Peak memory (GiB/GPU)	36.07	36.16	+0.26%
Checkpointing on	Throughput (tokens/s/GPU)	13,218.84	12,244.10	−7.37%
Fused causal attention	Kernel runtime (ms/step)	55.50	54.10	−2.52%

Table 19: Systems profiling of the 3B models under matched training settings.

the 3B Qwen3-style configuration. After warm-up, each method is measured for 200 training steps, and we report the mean and step-level standard deviation. We do not interpret these measurements as production-optimized latency: specialized fused CUDA kernels may improve the absolute runtime of all methods. This setting nevertheless remains practically relevant because developing and maintaining method-specific fused kernels is non-trivial, and such kernels may not be readily available across training frameworks and hardware platforms. More importantly, kernel optimization does not eliminate the underlying architectural costs reported in Table 2—additional arithmetic, activation I/O, and persistent inference state—and Depth-Attention has substantially lower costs along all three dimensions. Its low overhead without a custom depth-mixing kernel reflects the simplicity of the design and leaves additional room for production optimization.