MULTI-STREAM ATTENTION RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

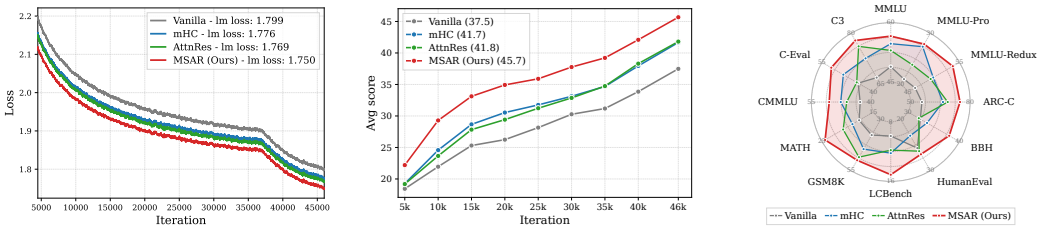
Attention Residuals (AttnRes) enables content-dependent retrieval over network depth, which can be viewed as single-stream softmax attention along the depth axis. Inspired by the success of multi-head attention along the sequence axis, we introduce **Multi-Stream Attention Residuals (MSAR)**, an extension of AttnRes that generalizes this mechanism to multi-head depth attention, yielding substantial performance gains with minimal additional overhead. MSAR draws stream diversity from sequence context through multi-scale causal convolutions, allowing different heads to retrieve historical representations through distinct contextual views while preserving the integrity of the hidden representation. The retrieved streams are combined through lightweight adaptive mixing to generate the single-stream layer input. By retaining single-stream sources and expanding them on demand, MSAR supports multi-stream computation without requiring persistent multi-stream storage or communication. Across all 12 downstream benchmarks at the 18B and 28B MoE scales, MSAR consistently outperforms prior residual variants, including standard residual connections, manifold-constrained Hyper-Connections (mHC), and AttnRes. Scaling-law experiments yield an iso-loss compute speedup of $\mathbf{1.58}\times$ over the standard residual baseline, compared with $\mathbf{1.35}\times$ for AttnRes and $\mathbf{1.30}\times$ for mHC. These results make MSAR a promising residual architecture for large-scale pretraining.

1 INTRODUCTION

Scaling depth has been a primary lever for improving Transformer-based large language models (LLMs) [21, 33, 40]. In modern PreNorm architectures [39], however, adding layers does not always yield the expected gains [12, 24]. The residual stream [13] aggregates preceding computations through a single hidden-state trajectory with fixed additive updates; as the stack grows deeper, features formed in early layers can be progressively diluted and become difficult for later layers to retrieve. This depth-wise information dilution limits the effective use of additional layers, making depth scaling less efficient than its representational capacity would suggest [6, 31, 34]. Motivated by this, recent work has begun to challenge the standard residual connection design itself. Representatively, Hyper-Connections (HC), together with its manifold-constrained variant mHC, expands the single residual stream into multiple parallel streams with learned stream mixing [38, 49], while Attention Residuals (AttnRes) replaces fixed residual accumulation with learned depth-wise softmax retrieval over earlier representations [34]. From the time-depth duality perspective of AttnRes [34, 44], these methods can be viewed as attention-like mechanisms along the depth axis: HC-family methods maintain a mixed multi-stream state, resembling recurrent, linear-attention-style depth memory, whereas AttnRes instantiates single-head depth-wise softmax attention.

Inspired by the success of multi-head attention along the sequence axis [35], a natural question emerges: can we realize *multi-head depth attention*, bringing greater expressiveness to residual retrieval over depth? However, previous attempts indicate that achieving this goal is non-trivial. AttnRes already tried channel-splitting the hidden state to form multi-head depth aggregation, which instead degraded performance [34]: in depth-wise attention the retrieved value is essentially the hidden state itself and should be treated as a coherent whole, so slicing it into low-dimensional fragments breaks this integrity. Another intuitive route is to derive several candidate streams from a single layer output using the $\mathcal{H}^{\text{post}}$-style write-back of (m)HC, and then treat them as depth-attention heads. This keeps each candidate full-dimensional, but the candidates are effectively scalar reweightings of the same vector; normalization largely removes these magnitude differences, yield-

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



(a) Training loss. (b) Average downstream score. (c) Per-benchmark performance.

Figure 1: **MSAR improves optimization and downstream performance at 18B scale.** MSAR achieves lower training loss and higher average downstream score than baseline residual variants.

ing nearly identical keys and limiting the diversity of source representations across heads. Taken together, these observations point to a central design challenge: how to construct diverse depth-attention heads from a single layer output without compromising its integrity.

We therefore propose **M**ulti-**S**tream **A**ttention **R**esiduals (**MSAR**), a low-cost extension of AttnRes that couples contextual stream expansion with depth-wise retrieval to address this challenge. Starting from a single residual source, MSAR draws diversity from the sequence axis: it applies a small set of multi-scale causal depthwise convolutions to the layer output, producing m full-dimensional streams whose receptive fields encode different local-context scales. These streams serve as distinct source views for depth attention, so each stream can perform its own softmax retrieval over depth instead of sharing one depth distribution. The m retrieved states are then fused by a lightweight $\mathcal{H}^{\text{pre}}$-style mixer into the input of the target layer. In this way, MSAR turns residual retrieval into a multi-head depth-attention primitive: different heads can select different depth histories through distinct sequence-context views while preserving the integrity of the hidden representation.

At the system level, MSAR is fully compatible with AttnRes block mode, cache-based pipeline communication, and the two-phase computation strategy. In our 28B MoE setting, MSAR increases training FLOPs by only 0.5% and activated parameters by 0.3%. Notably, during the forward pass, all streams are deterministically generated from the main residual state through causal depthwise convolutions, allowing MSAR to keep only the main stream in memory and expand it on demand when computing depth attention. The convolution-expanded source streams are intermediate values used only by depth attention and are immediately consumed by the following attention computation; in practice, this allows stream expansion and consumption to be handled by a fused kernel, avoiding write-back of these m intermediate streams. The same principle applies in block mode: each block accumulates layer outputs into a single-stream summary, and the m streams are recomputed from these summaries when they are consumed by depth attention. With this design, for N stored sources, MSAR reduces pipeline communication and prefill activation storage from $O(mNd)$ to $O(Nd)$, matching the storage cost of AttnRes. Compared with AttnRes, the remaining overhead mainly comes from the memory access needed to write the retrieved multi-stream outputs and aggregate these streams.

Empirically, MSAR consistently improves both training loss and downstream performance in large-scale pre-training. As summarized in Figure 1, in our 18B MoE setting, MSAR achieves the lowest final training loss, reaching 1.750 compared with Vanilla (1.799), mHC (1.776), and AttnRes (1.769). This training-loss gain consistently translates into higher average downstream scores throughout training across representative benchmarks covering knowledge, reasoning, math, and code. Scaling-law experiments further confirm that MSAR reaches the same loss with substantially less compute: relative to the Vanilla baseline, MSAR achieves an iso-loss compute speedup of $1.58\times$, compared with $1.35\times$ for AttnRes and $1.30\times$ for mHC. These results suggest that enriching depth-wise residual retrieval with multi-stream sequence-context views is an effective route for improving depth scaling. Our contributions can be summarized as follows:

1). Multi-Stream Attention Residuals. We introduce MSAR, which generalizes AttnRes from single-stream residual retrieval to multi-head depth attention. MSAR uses low-cost multi-scale causal depthwise convolutions to construct multiple streams, performs per-stream depth attention, and aggregates the retrieved streams with lightweight scalar mixing. This yields stream diversity without breaking the full-dimensional coherence of the hidden representation.

2). Efficient design. MSAR stores only single-stream summaries and expands multi-stream representations on demand, reducing pipeline communication and prefill activation storage from $O(mNd)$ to $O(Nd)$ and matching the storage cost of AttnRes. It remains fully compatible with AttnRes's cross-stage caching and two-phase inference strategy, with only minimal training FLOPs and parameter overhead.

3). Large-scale validation. We validate MSAR in large-scale pre-training, where it delivers significant and consistent downstream gains in both 18B and 28B MoE settings. Scaling-law experiments and ablation studies further verify the effectiveness of the design, showing that the gains come from the proposed inductive bias rather than from a model-size-specific artifact.

2 Related Work

Multi-State Residual Recurrence. One approach improves depth-wise information flow by maintaining multiple residual states [20, 22, 41, 45, 47]. Hyper-Connections (HC) [49] implement this through learned stream aggregation, output distribution, and inter-stream mixing. Its manifold-constrained variant, mHC [38], stabilizes multi-stream recurrence through doubly stochastic residual mixing, while its Single-Pass variant [8] removes the dependency on current-state read coefficients to enable single-pass fused execution. xHC [43] enriches the write-back signal and uses sparse stream operations to support larger residual expansion. Gated Residual [27] uses elementwise gated reads and input-dependent scalar writes over multiple residual streams, without an explicit inter-stream residual-mixing operator.

Cross-Layer Access. Another line of work gives later layers explicit access to earlier representations [10, 25, 46, 48]. DenseNet [16] connects layers by concatenating preceding feature maps. DenseFormer [26] aggregates earlier hidden states using learned, input-independent depth weights. MUDDFormer [37] introduces input-dependent, multiway aggregation of historical layer representations. Mixture-of-Depths Attention (MoDA) [50] jointly attends to sequence key-value pairs and same-position key-value pairs from preceding layers. Attention Residuals (AttnRes) [34] replaces residual accumulation with softmax-weighted retrieval over earlier layer outputs, using block-level sources and specialized execution strategies to reduce storage and communication costs. Representative methods from both lines, mHC and AttnRes, have also been validated in trillion-parameter-scale pretraining through DeepSeek-V4-Pro [9] and Kimi K3 [18], respectively.

Position of MSAR. MSAR instantiates the intersection of the two residual-design axes above. It preserves the explicit cross-layer access of AttnRes, while introducing a multi-stream retrieval structure inspired by the capacity expansion of HC-family methods. Rather than widening the persistent residual recurrence, MSAR widens the depth-attention primitive itself by exposing historical hidden states through multiple sequence-context views.

3 Method

3.1 Preliminaries: Residuals as Depth Mixing

Notation. We index depth at the granularity of residual branches: each self-attention or MLP branch in a Transformer stack is counted as one layer. Unless otherwise stated, equations are written for a single sequence position, with batch and token indices suppressed. Let $\boldsymbol{h}_l \in \mathbb{R}^d$ denote the representation provided as input to the l-th residual branch, for $l \in \{1, \dots, L\}$, and let $\boldsymbol{h}_1$ be the token embedding. We write $f_l(\cdot)$ for the transformation performed by branch l, absorbing the local normalization, attention or MLP computation, and output projection into this operator.

Standard Residual Accumulation. A standard residual stack propagates representations by adding each branch output to the running hidden state:

$$\boldsymbol{h}_{l+1} = \boldsymbol{h}_l + f_l(\boldsymbol{h}_l). \tag{1}$$

We treat the embedding as the initial depth source and define

$$\boldsymbol{v}_0 = \boldsymbol{h}_1, \qquad \boldsymbol{v}_i = f_i(\boldsymbol{h}_i), \quad i \geq 1. \tag{2}$$

Unrolling the recurrence gives $\boldsymbol{h}_l = \boldsymbol{h}_1 + \sum_{i=1}^{l-1} f_i(\boldsymbol{h}_i) = \sum_{i=0}^{l-1} \boldsymbol{v}_i$. Thus, standard residual connections perform depth mixing with fixed unit coefficients: every previous source is accumulated into

a single residual stream. It preserves a direct identity path across layers, allowing information and gradients to pass through the stack without being forced through every nonlinear transformation.

The same formulation also prescribes how information is aggregated across depth. All previous updates are compressed into one running state, so later layers read the accumulated representation rather than individual depth sources. Moreover, as more updates are added, the residual state grows with depth, making each new update progressively smaller relative to the accumulated stream.

(Manifold-Constrained) Hyper-Connections. A direct way to relax this single-state constraint is to expand the residual state into multiple persistent streams. This increases the capacity of the residual path to carry distinct depth histories, yielding significant pre-training improvements with modest additional FLOPs [38, 49]. Let

$$\mathbf{X}_l = \left(\boldsymbol{x}_l^{(1)}, \ldots, \boldsymbol{x}_l^{(m)}\right)^\top \in \mathbb{R}^{m \times d} \tag{3}$$

denote the multi-stream state before layer l, where m is the number of streams. Hyper-Connections first read a layer input from this state and then write the layer output back to all streams:

$$\boldsymbol{h}_l = \left(\mathcal{H}_l^{\text{pre}} \mathbf{X}_l\right)^\top, \qquad \boldsymbol{v}_l = f_l(\boldsymbol{h}_l), \tag{4}$$

$$\mathbf{X}_{l+1} = \mathcal{H}_l^{\text{res}} \mathbf{X}_l + \mathcal{H}_l^{\text{post}} f_l(\boldsymbol{h}_l)^\top. \tag{5}$$

Here $\mathcal{H}_l^{\text{pre}} \in \mathbb{R}^{1 \times m}$ aggregates streams into the layer input, $\mathcal{H}_l^{\text{post}} \in \mathbb{R}^{m \times 1}$ distributes the new residual update across streams, and $\mathcal{H}_l^{\text{res}} \in \mathbb{R}^{m \times m}$ mixes the persistent state [49]. The manifold-constrained variant mHC further projects $\mathcal{H}_l^{\text{res}}$ onto the Birkhoff polytope of doubly stochastic matrices via the Sinkhorn–Knopp algorithm, preserving stable signal propagation while retaining the multi-state residual form [38].

Attention Residuals. A complementary route is to expose individual sources instead of only the accumulated state. For layer l, AttnRes replaces the residual sum with a normalized mixture over the embedding source and all preceding branch outputs:

$$\boldsymbol{h}_l = \alpha_{0 \to l} \boldsymbol{h}_1 + \sum_{i=1}^{l-1} \alpha_{i \to l} f_i(\boldsymbol{h}_i). \tag{6}$$

The coefficients $\alpha_{i \to l}$ are layer-specific attention weights satisfying $\sum_{i=0}^{l-1} \alpha_{i \to l} = 1$; they are obtained by applying a softmax over all available sources:

$$\alpha_{i \to l} = \frac{\phi(\boldsymbol{q}_l, \boldsymbol{k}_i)}{\sum_{j=0}^{l-1} \phi(\boldsymbol{q}_l, \boldsymbol{k}_j)}. \tag{7}$$

AttnRes instantiates this attention with a layer-specific learned pseudo-query and uses each residual source as its key:

$$\boldsymbol{q}_l = \boldsymbol{w}_l, \qquad \boldsymbol{k}_i = \boldsymbol{v}_i, \qquad \phi(\boldsymbol{q}, \boldsymbol{k}) = \exp\left(\boldsymbol{q}^\top \operatorname{RMSNorm}(\boldsymbol{k})\right). \tag{8}$$

This turns residual aggregation into content-dependent depth attention by explicitly preserving previous hidden states as retrievable sources [34].

3.2 MSAR: Multi-Stream Attention Residuals

Multi-Stream Source Expansion. As discussed in Section 1, multi-head depth attention requires a non-trivial expansion of a single layer output into multiple streams with sufficient head diversity. A natural approach is to generate these streams through separate dense projections, but this substantially increases the parameter count and FLOPs. Instead, we borrow additional contextual information from neighboring tokens to construct multiple views of each residual source at low cost.

Under autoregressive modeling with a causal mask, earlier hidden states in the same layer encode the context available at their respective positions and can therefore supplement the current source without exposing future information. Aggregating these states over different local windows provides source views with different contextual coverage while preserving the integrity of the hidden representation. Specifically, we use lightweight causal depthwise 1D convolutions to expand each residual source into m streams:

$$\boldsymbol{v}_i^{(r)} = \mathcal{E}_{i,r}[\boldsymbol{v}_i], \qquad r = 0, \ldots, m-1, \tag{9}$$

where $\mathcal{E}_{i,0} = \mathrm{Id}$ and $\mathcal{E}_{i,r}$ for $r \geq 1$ is a layer-specific causal depthwise convolution with kernel size κ_r. Restoring the token index t gives

$$\boldsymbol{v}_{i,t}^{(0)} = \boldsymbol{v}_{i,t}, \qquad \boldsymbol{v}_{i,t}^{(r)} = \sum_{\delta=0}^{\kappa_r - 1} \boldsymbol{c}_{i,r,\delta} \odot \boldsymbol{v}_{i,t-\delta}, \quad r = 1, \ldots, m-1, \tag{10}$$

where $\boldsymbol{c}_{i,r,\delta} \in \mathbb{R}^d$ contains the channel-wise convolution weights, and positions before the sequence start are zero-padded. The kernel sizes assign different local receptive fields to the streams: the original stream preserves the current source, while the additional streams incorporate preceding representations over different ranges. In our main setting, we retain the original source and apply three convolutions with kernel sizes $\{4, 8, 12\}$, yielding $m = 4$ streams.

Per-Stream Depth Attention. The expanded streams serve as distinct depth-attention heads, each associated with a different contextual scale. For each target layer, we perform depth attention independently within each stream, aggregating the corresponding source representations across depth:

$$\boldsymbol{z}_l^{(r)} = \alpha_{0\to l}^{(r)} \mathcal{E}_{0,r}[\boldsymbol{h}_1] + \sum_{i=1}^{l-1} \alpha_{i\to l}^{(r)} \mathcal{E}_{i,r}[f_i(\boldsymbol{h}_i)] = \sum_{i=0}^{l-1} \alpha_{i\to l}^{(r)} \boldsymbol{v}_i^{(r)}, \qquad r = 0, \ldots, m-1. \tag{11}$$

Each target layer now has m independently learned pseudo-queries, one per stream. The attention weights are defined as

$$\boldsymbol{q}_l^{(r)} = \boldsymbol{w}_l^{(r)}, \qquad \alpha_{i\to l}^{(r)} = \frac{\phi(\boldsymbol{q}_l^{(r)}, \boldsymbol{k}_i^{(r)})}{\sum_{j=0}^{l-1} \phi(\boldsymbol{q}_l^{(r)}, \boldsymbol{k}_j^{(r)})}, \tag{12}$$

where $\boldsymbol{k}_i^{(r)} = \boldsymbol{v}_i^{(r)}$, and ϕ is the same RMS-normalized exponential kernel defined in Section 3.1. The weights are normalized over historical sources within each stream, so $\sum_{i=0}^{l-1} \alpha_{i\to l}^{(r)} = 1$ for every r. This produces m retrieved states $\mathbf{Z}_l = [\boldsymbol{z}_l^{(0)}, \ldots, \boldsymbol{z}_l^{(m-1)}]^\top \in \mathbb{R}^{m\times d}$.

Stream Fusion. The retrieved streams $\mathbf{Z}_l$ must be fused into a single input for the target layer. Following the output projection of sequence-level multi-head attention would require an $md \times d$ matrix. Instead, we use a lightweight mHC-style $\mathcal{H}^{\mathrm{pre}}$ mixer [38] to generate input-dependent scalar weights:

$$\mathcal{H}_l^{\mathrm{pre}} = \sigma(\alpha_l^{\mathrm{pre}}\, \mathrm{RMSNorm}(\mathrm{Flatten}(\mathbf{Z}_l))\, \mathbf{W}_l^{\mathrm{pre}} + \boldsymbol{b}_l^{\mathrm{pre}}) \in \mathbb{R}^{1\times m}, \tag{13}$$

Here $\mathbf{W}_l^{\mathrm{pre}} \in \mathbb{R}^{md\times m}$ is a lightweight projection, α_l^{pre} is a learnable scaling factor, $\boldsymbol{b}_l^{\mathrm{pre}} \in \mathbb{R}^{1\times m}$ is a per-stream bias, and σ denotes the sigmoid function. The target layer input is a weighted sum of the retrieved streams:

$$\boldsymbol{h}_l = (\mathcal{H}_l^{\mathrm{pre}} \mathbf{Z}_l)^\top = \sum_{r=0}^{m-1} [\mathcal{H}_l^{\mathrm{pre}}]_r\, \boldsymbol{z}_l^{(r)} \in \mathbb{R}^d. \tag{14}$$

For the last layer output, we omit further expansion for depth retrieval, directly perform a residual merge and pass the result to the LM head.

3.3 Block MSAR

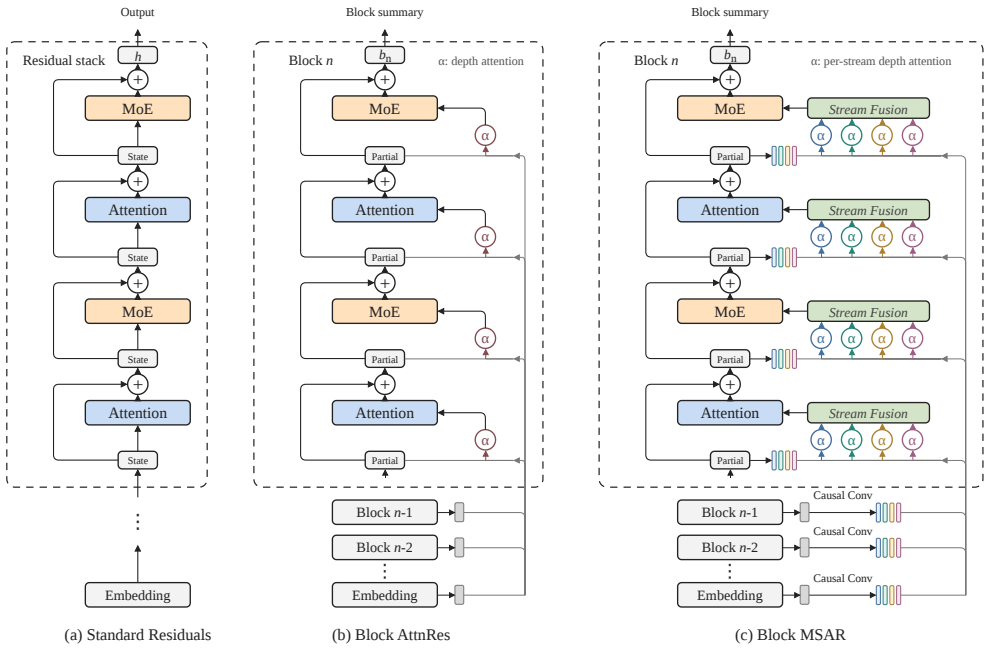
Directly applying MSAR requires retaining the hidden state produced by every layer, incurring considerable memory and pipeline communication overhead in large-scale training. Following AttnRes [34], we introduce Block MSAR, which replaces individual layer outputs with block-level summaries as depth-attention sources. We partition the L residual branches into N contiguous blocks $\mathcal{B}_1, \ldots, \mathcal{B}_N$. Within each block, outputs accumulate in a single stream; the resulting summaries are then expanded for multi-stream depth attention.

We retain the embedding as the initial source $\boldsymbol{b}_0 = \boldsymbol{h}_1$ and define the completed summary of block n as

$$\boldsymbol{b}_n = \sum_{j\in\mathcal{B}_n} f_j(\boldsymbol{h}_j) \in \mathbb{R}^d. \tag{15}$$

For a target layer $l \in \mathcal{B}_n$, the current block contributes the partial sum of its preceding outputs,

$$\boldsymbol{b}_n^{<l} = \sum_{j\in\mathcal{B}_n, j<l} f_j(\boldsymbol{h}_j), \qquad \boldsymbol{b}_n^{<l+1} = \boldsymbol{b}_n^{<l} + f_l(\boldsymbol{h}_l). \tag{16}$$



(a) Standard Residuals (b) Block AttnRes (c) Block MSAR

Figure 2: **Comparison of residual architectures.** (a) Standard Residuals; (b) Block AttnRes; (c) Block MSAR. Block MSAR expands single-stream sources into contextual streams for per-stream depth attention, then fuses the retrieved streams into a single input for each residual branch.

The partial sum starts at zero and becomes $\boldsymbol{b}_n$ when the block completes. Stream expansion is applied to the accumulated single-stream representation. Below, $\mathcal{E}_r$ omits bank indices.

For layer $l \in \mathcal{B}_n$, each head retrieves from the embedding, completed block summaries, and the current partial after source expansion:

$$\boldsymbol{z}_l^{(r)} = \sum_{j=0}^{n-1} \alpha_{j\to l}^{(r)} \mathcal{E}_r[\boldsymbol{b}_j] + \alpha_{n\to l}^{(r)} \mathcal{E}_r[\boldsymbol{b}_n^{<l}], \qquad r = 0, \ldots, m-1. \tag{17}$$

Here the source index in $\alpha_{j\to l}^{(r)}$ refers to a block summary, with index n denoting the current partial. The layer-specific pseudo-queries and key construction follow Section 3.2, with softmax normalization over all available sources within each stream. At the first layer of a block, the partial source is absent and is omitted from both the retrieval sum and the softmax denominator. The retrieved states form $\mathbf{Z}_l$ and are fused into $\boldsymbol{h}_l$ using Eq. 14. Figure 2 contrasts this design with standard residual connections and Block AttnRes.

3.4 Efficiency Analysis

Single-Stream Storage and Communication. Since the expanded streams are deterministic functions of the single-stream block summaries defined in Section 3.3, MSAR only needs to retain and communicate single-stream sources. These sources are expanded through lightweight causal convolutions only when depth attention is computed. To preserve this property in Block MSAR, we deliberately accumulate layer outputs before stream expansion, rather than expand each output and accumulate streams separately. For N summaries, this reduces persistent source storage and the corresponding pipeline communication payload per token from $O(mNd)$ to $O(Nd)$, matching Block AttnRes under the same pipeline schedule. The single-stream representation also supports AttnRes's cross-stage caching [34], allowing previously received summaries to be reused across virtual pipeline stages without retransmission.

Two-Phase Computation. All target layers within a block access the same completed sources, and their learned pseudo-queries are available before the block executes. This enables a two-phase computation strategy for MSAR, analogous to AttnRes, with each stream processed independently.

In the first phase, queries for all target layers in the block are batched against the completed sources to compute per-target, per-stream retrieval results and softmax normalization statistics. Batching allows the targets to share source reads, reducing repeated memory access across layers. In the second phase, each layer computes the contribution of its current partial and merges it with its precomputed result through online softmax. This recovers the attention in Eq. 17; the first layer of a block uses only the completed-source result. Stream fusion is applied after the merge, so the first phase retains m retrieved streams per target, separately from the single-stream source storage.

Fused Stream Expansion. On-demand expansion can introduce additional memory traffic if its outputs are written out and subsequently read by attention. The expanded streams are consumed immediately by the subsequent depth attention and have no other consumers in the forward pass. This allows causal convolutional expansion and per-stream attention to be combined in a fused kernel, avoiding writes of these intermediate source streams to memory. This complements two-phase computation: batching reduces repeated source reads across targets, while fusion removes intermediate writes and reads during stream expansion. The final retrieved multi-stream outputs are still written and subsequently read for $\mathcal{H}^{\text{pre}}$ aggregation, contributing additional memory traffic relative to AttnRes. Appendix C compares the per-token, per-layer memory access costs of the residual mechanisms across methods.

4 Experiments

4.1 Experimental Setup

Training Settings. We conduct language model pretraining experiments with DeepSeekMoE-style Transformers [7] using SwiGLU feed-forward networks [28]. Our main results cover two MoE scales with 18B and 28B total parameters, and we use a 10B MoE model for ablation studies. Comprehensive training details are provided in Appendix A. Unless otherwise specified, we use Block MSAR with $m = 4$ streams and $N \approx 8$ blocks. We compare MSAR with two representative residual architectures, mHC [38] with $m = 4$ streams and AttnRes [34] with $N \approx 8$ blocks. We also report a Transformer with standard residual connections as the Vanilla baseline.

Evaluation. We evaluate downstream performance on 12 benchmarks grouped into three categories: general capabilities (MMLU [14], MMLU-Pro [36], MMLU-Redux [11], ARC-C [3], and BBH [32]); code and mathematics (HumanEval [2], LCBench [5], GSM8K [4], and MATH [15]); and Chinese language understanding (CMMLU [19], C-Eval [17], and C3 [30]). Evaluation protocols and few-shot settings are provided in Appendix A.

4.2 Main Results

We compare MSAR against mHC, AttnRes, and the Vanilla residual baseline on 18B and 28B MoE models. Table 1 summarizes downstream results under comparable parameter counts and training FLOPs. In our 28B MoE setting, MSAR with $m = 4$ streams and $N \approx 8$ blocks adds only 0.3% activated parameters and 0.5% training FLOPs relative to Vanilla. Detailed parameterization and overhead comparisons across methods are provided in Appendix C.

As shown in Table 1, MSAR outperforms mHC, AttnRes, and the Vanilla baseline on all 12 evaluated benchmarks at both model scales. In the 18B setting, MSAR achieves an average score of 45.7, compared with 37.5 for Vanilla (+8.2), 41.7 for mHC (+4.0), and 41.8 for AttnRes (+3.9). While both mHC and AttnRes improve substantially over standard residual connections, MSAR provides a further gain over both approaches. Representative gains over AttnRes include MMLU (+3.5), ARC-C (+6.2), and BBH (+5.9). The gains persist in the larger 28B setting, where MSAR achieves an average score of 52.4, compared with 44.6 for Vanilla (+7.8), 47.4 for mHC (+5.0), and 49.4 for AttnRes (+3.0). This consistent downstream advantage across both model scales further supports the effectiveness of MSAR in large-scale pretraining.

4.3 Scaling Laws

To quantify the compute efficiency of MSAR across model scales, we compare its training-loss scaling with Vanilla, mHC, and AttnRes. We evaluate four model scales for each method, spanning

Table 1: Downstream performance comparison of MSAR with Vanilla, mHC, and AttnRes on 18B and 28B MoE models. Scores are reported in percent (higher is better), with the best result for each benchmark at each scale in **bold**.

	18B				28B			
Benchmark	Vanilla	mHC	AttnRes	MSAR	Vanilla	mHC	AttnRes	MSAR
General								
MMLU	48.9	54.7	53.0	**56.6**	54.6	56.8	59.9	**62.5**
MMLU-Pro	21.1	27.4	23.9	**27.9**	30.1	34.9	36.0	**38.7**
MMLU-Redux	46.4	49.9	49.4	**53.7**	50.6	53.9	55.7	**58.7**
ARC-C	55.7	66.3	68.7	**74.9**	70.8	74.9	75.6	**78.0**
BBH	32.4	33.7	32.1	**38.0**	41.7	43.6	40.7	**46.1**
Code & Math								
HumanEval	25.6	23.2	26.2	**26.8**	27.4	26.8	29.3	**31.1**
LCBench	9.9	12.2	11.8	**15.1**	15.1	14.8	16.5	**18.4**
GSM8K	37.7	44.5	48.4	**49.9**	50.3	56.3	57.2	**65.7**
MATH	17.8	19.1	20.9	**24.4**	21.9	26.5	28.0	**28.8**
Chinese								
CMMLU	42.7	47.6	46.1	**50.5**	47.6	50.1	51.3	**56.4**
C-Eval	44.5	48.8	44.9	**52.3**	50.2	51.2	59.1	**59.9**
C3	67.1	72.7	76.2	**77.9**	75.2	78.7	83.5	**84.9**
Average	37.5	41.7	41.8	**45.7**	44.6	47.4	49.4	**52.4**

approximately 3×10^{19} to 1×10^{21} training FLOPs. Our FLOPs accounting excludes the LM head and includes method-specific residual overhead. We fit a shifted power law to the final training loss:

$$\mathcal{L}(C) = AC^{-\beta} + E, \tag{18}$$

where C denotes training FLOPs, A and β are fitted separately for each method, and $E = 0.72$ is the estimated irreducible loss, which represents the asymptotic loss floor as training compute increases. We estimate the parameters by linear regression in log space. Detailed model configurations and fitting procedures are provided in Appendix B.

Figure 3 shows that MSAR achieves the lowest training loss at all four scales, and its fitted curve remains below all three baselines over their shared compute range. At the largest model scale, MSAR reduces training loss by 2.7% relative to Vanilla, compared with reductions of 1.6% for mHC and 1.8% for AttnRes. Averaged across the four model scales, MSAR achieves an iso-loss compute speedup of $1.58\times$ relative to Vanilla, compared with $1.35\times$ for AttnRes and $1.30\times$ for mHC. These results support a consistent improvement in the loss–compute trade-off across the evaluated scales.

4.4 Ablation Studies

We evaluate the key design choices of MSAR on a 10B MoE model with 15 Transformer layers, comprising 30 residual sublayers. Table 2 reports Pile loss, with S denoting the number of residual sublayers per block. Detailed model configurations are provided in Appendix A.

Comparison with Prior Methods. Table 2, rows (1)–(5), compares MSAR with standard residual connections and prior residual architectures. mHC and AttnRes reduce loss from the Vanilla baseline's 1.898 to 1.871 and 1.867, respectively, while MSAR achieves a substantially lower loss of 1.847. We also evaluate a hybrid residual architecture, AttnRes + mHC, that simultaneously maintains AttnRes's historical sources for cross-layer access and mHC's multi-stream recurrent state, summing their readouts to form the layer input. This direct combination reaches 1.861, an additional reduction of only 0.006 over AttnRes. With the same $m = 4$ and $S = 4$, MSAR lowers loss by a further 0.014. These results suggest that MSAR's advantage stems from its specific design, rather than simply adding the states produced by cross-layer access and multi-stream recurrence.

Component Design. Rows (5)–(7) isolate the contributions of contextual source expansion and adaptive stream fusion. Removing the convolutions replicates each source into four identical streams while retaining four independently learned pseudo-queries per target layer, yielding a multi-query

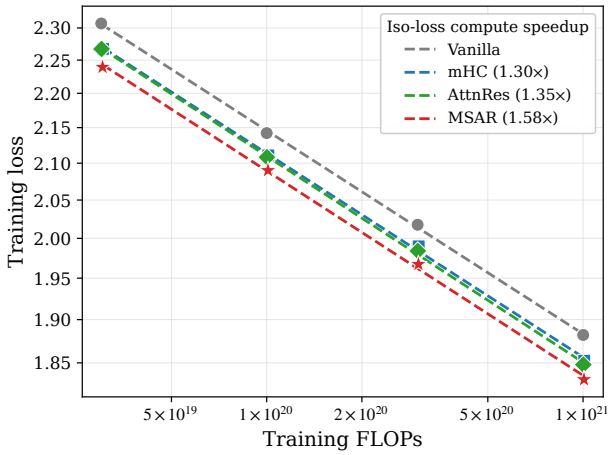


Figure 3: Scaling-law comparison of MSAR, AttnRes, mHC, and Vanilla. MSAR substantially outperforms all three baselines in the loss–compute trade-off.

Table 2: Ablations on a 10B MoE model. m: number of streams; S: residual sublayers per block.

Variant	m	S	Pile loss ↓
(1) Vanilla	–	–	1.898
(2) mHC	4	–	1.871
(3) AttnRes	1	4	1.867
(4) AttnRes + mHC	4	4	1.861
(5) **MSAR (default)**	4	4	1.847
(6) w/o Conv	4	4	1.862
(7) w/o $\mathcal{H}^{\text{pre}}$	4	4	1.851
(8) $m = 2$	2	4	1.850
(9) $m = 8$	8	4	1.846
(10) Full MSAR	4	1	1.843
(11) $S = 2$	4	2	1.845
(12) $S = 8$	4	8	1.852

depth-attention variant over shared source representations. The independent queries can still produce distinct depth-attention distributions, allowing this ablation to distinguish the benefit of contextual source expansion from that of multiple retrieval heads alone. This increases loss from 1.847 to 1.862, supporting the importance of source diversity beyond query diversity alone. Replacing $\mathcal{H}^{\text{pre}}$ with mean aggregation produces a relatively small loss degradation to 1.851, indicating that adaptive fusion provides an additional benefit while uniform averaging retains much of the gain. This modest contribution is consistent with the HC component ablations reported in mHC [38], and our experiments also indicate that $\mathcal{H}^{\text{pre}}$ contributes to training stability.

Hyperparameter Sensitivity. Rows (5) and (8)–(12) examine the number of streams and block granularity. Two streams already capture most of the improvement over Vanilla, achieving a loss of 1.850. Increasing m to four lowers loss to 1.847, while eight streams yield only a marginal further improvement to 1.846. We therefore use $m = 4$ by default. For block granularity, $S = 2, 4, 8$ correspond to $N = 15, 8, 4$ blocks, respectively, excluding the embedding source. Their losses are 1.845, 1.847, and 1.852, compared with 1.843 for Full MSAR. Thus, retaining approximately eight blocks recovers most of the benefit of full depth access, with diminishing gains from finer partitioning. Consistent with the block-size ablations in AttnRes, we use $N \approx 8$ as the default.

5 Conclusion and Outlook

We introduce Multi-Stream Attention Residuals (MSAR), which extends single-stream residual retrieval to multi-head depth attention. MSAR draws source diversity from the sequence axis through lightweight multi-scale causal convolutions, retrieves independently over depth for each stream, and aggregates the retrieved states with adaptive scalar mixing. By storing single-stream summaries and expanding them on demand, Block MSAR matches Block AttnRes's persistent source storage and pipeline communication costs while supporting cross-stage caching and two-phase computation. Experiments on 18B and 28B MoE models show consistent downstream improvements over prior residual variants, while scaling-law experiments yield an iso-loss compute speedup of $1.58\times$ relative to the standard residual baseline. These results establish MSAR as an effective residual architecture for improving the quality and compute efficiency of language model pretraining. Appendix C shows that MSAR has theoretical costs comparable to those of mHC and AttnRes. Further infrastructure optimization and kernel fusion tailored to MSAR remain important directions for improving its end-to-end training efficiency. More broadly, our results suggest that the benefits of multi-state representations and cross-layer access are not fully overlapping, motivating more elegant residual designs that combine their complementary strengths with efficient execution across diverse model architectures and large-scale pretraining settings.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript language refinement and assistance with code review and revision. The core research ideas originated with the human authors, who designed the training pipeline and conducted the experiments. AI-assisted text and code are subject to author review and verification. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We pursue this research with a commitment to responsible AI development. MSAR aims to improve the performance and compute efficiency of large language model pretraining through residual architecture design. We recognize that advances in language modeling can support beneficial applications while also carrying risks of misuse, biased outputs, and harmful content. We encourage researchers and practitioners to apply MSAR responsibly, evaluate safety in the context of intended applications, and adopt appropriate safeguards consistent with established ethical guidelines.

REPRODUCIBILITY STATEMENT

All compared methods use the same training configurations and evaluation protocols within each experimental setting to ensure a fair comparison. Our pretraining corpus is primarily proprietary and comprises web text, code, mathematical content, and multilingual sources, processed through exact and fuzzy deduplication, rule-based filtering, and LLM-based quality filtering. Due to licensing and privacy constraints, we are unable to release the full raw corpus. The same data recipe is used for all compared methods, so this restriction does not affect the fairness of comparisons within this study. Model configurations, training hyperparameters, and evaluation protocols are provided in Appendix A, with scaling-law settings in Appendix B. We will release the complete implementation of MSAR upon acceptance.

REFERENCES

[1] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth, 2020. URL https://arxiv.org/abs/2003.04887.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[3] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[5] OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

[6] Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*, 2025.

[7] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, 2024.

[8] DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026.

[9] DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

[10] Yanwen Fang, Yuxi Cai, Jintai Chen, Jingyu Zhao, Guangjian Tian, and Guodong Li. Cross-layer retrospective retrieving via layer attention, 2023. URL https://arxiv.org/abs/2302.03985.

[11] Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. Are we done with mmlu? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096, 2025.

[12] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[15] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[17] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010, 2023.

[18] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

[19] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11260–11285, 2024.

[20] Tianyu Li, Dongchen Han, Zixuan Cao, Haofeng Huang, Mengyu Zhou, Ming Chen, Erchao Zhao, Xiaoxi Jiang, Guanjun Jiang, and Gao Huang. Siamesenorm: Breaking the barrier to reconciling pre/post-norm, 2026. URL https://arxiv.org/abs/2602.08064.

[21] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

[22] Zhaoyi Liu, Haichuan Zhang, and Ang Li. Beyond the birkhoff polytope: Spectral-sphere-constrained hyper-connections. *arXiv preprint arXiv:2603.20896*, 2026.

[23] Cheng Luo, Zefan Cai, and Junjie Hu. Multi-head attention residuals. *arXiv preprint arXiv:2607.27230*, 2026.

[24] Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

[25] Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. Laurel: Learned augmented residual layer, 2025. URL https://arxiv.org/abs/2411.07501.

[26] Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

[27] Zihan Qiu, Zekun Wang, Xiao Li, Yanpeng Li, Yang Xu, Yixuan Wang, Huaqing Zhang, Rui Men, Bochao Mao, Chengruidong Zhang, et al. On the design of qwen3. 8-next architecture: Evaluation, efficiency, and training stability. *arXiv preprint arXiv:2608.30320*, 2026.

[28] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[29] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks, 2015. URL https://arxiv.org/abs/1505.00387.

[30] Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155, 2020.

[31] Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025.

[32] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

[33] Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[34] Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

[35] 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*, 30, 2017.

[36] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[37] Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Muddformer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. *arXiv preprint arXiv:2502.12170*, 2025.

[38] Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

[39] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.

[40] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[41] Yongyi Yang and Jianyang Gao. mhc-lite: You don't need 20 sinkhorn-knopp iterations. *arXiv preprint arXiv:2601.05732*, 2026.

[42] Xingda Yu, Yining Li, Xinzhang Liu, Zhihao Yang, Haowei He, Chao Wang, Yongxiang Li, and Shuangyong Song. Dual attention residuals, 2026. URL https://arxiv.org/abs/2607.18730.

[43] Xiangdong Zhang, Xiaohan Qin, Sunan Zou, Tuo Dai, Xiaoming Shi, Huaijin Wu, Yebin Yang, Zhuo Xia, Shaofeng Zhang, Lin Yao, et al. xhc: Expanded hyper-connections. *arXiv preprint arXiv:2607.14530*, 2026.

[44] Yifan Zhang. Residual stream duality in modern transformer architectures. *arXiv preprint arXiv:2603.16039*, 2026.

[45] Yifan Zhang, Yifeng Liu, Mengdi Wang, and Quanquan Gu. Deep delta learning, 2026. URL https://arxiv.org/abs/2601.00417.

[46] Yilang Zhang, Bingcong Li, Niao He, and Georgios B. Giannakis. Ancre: Adaptive neural connection reassignment for efficient depth scaling, 2026. URL https://arxiv.org/abs/2602.09009.

[47] Zhizhan Zheng, Feiyun Zhang, Shuchun Liu, Tian Xia, Xi Liu, Dasheng Hu, and Hongquan Zhou. Multi-gate residuals. *arXiv preprint arXiv:2605.23259*, 2026.

[48] Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, Fares Obeid, and Zhenzhong Lan. Value residual learning, 2025. URL https://arxiv.org/abs/2410.17897.

[49] Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2024.

[50] Lianghui Zhu, Yuxin Fang, Bencheng Liao, Shijie Wang, Tianheng Cheng, Zilong Huang, Chen Chen, Lai Wei, Yutao Zeng, Ya Wang, et al. Mixture-of-depths attention. *arXiv preprint arXiv:2603.15619*, 2026.

A EXPERIMENTAL DETAILS

Model Architectures and Training Settings. We use a 10B MoE model for ablation studies and 18B and 28B MoE models for the main comparisons. All backbones follow the DeepSeekMoE-style architecture [7], with one leading dense Transformer layer followed by MoE layers, grouped-query attention, and SwiGLU feed-forward networks. Table 3 summarizes the model configurations and training hyperparameters; layer counts refer to Transformer layers, each containing an attention and an MLP residual sublayer.

Table 3: Backbone configurations and training hyperparameters. The 10B model is used for ablation studies, and the 18B and 28B models are used for the main results. Activated parameter counts are approximate and exclude input embeddings.

Attribute	**10B**	**18B**	**28B**
Total Parameters	10B	18B	28B
Activated Parameters	1.1B	1.7B	2.7B
Transformer Layers	15	28	32
Leading Dense Layers	1	1	1
MoE Layers	14	27	31
Hidden Dimension	2080	2112	2560
FFN Dimension (dense layer)	6144	6144	6144
Attention Heads	16	16	20
KV Groups (GQA)	8	8	10
Head Dimension	128	128	128
Routed Experts	144	144	144
Shared Experts	1	1	1
Active Experts (Top-k)	8	8	8
Expert FFN Dimension	704	672	768
Shared Expert Dimension	704	672	768
Normalization	RMSNorm ($\epsilon = 10^{-5}$)		
Position Embedding	RoPE ($\theta = 50000$)		
Activation	SwiGLU		
Sequence Length	8192		
Vocabulary Size	152064		
Tokenizer	Qwen2		
Global Batch Size	288	512	768
Training Tokens	48.3B	200B	250B
Warmup Steps	500	500	500
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-15}$)		
Base Learning Rate	4.82×10^{-4}	3.97×10^{-4}	3.5×10^{-4}
LR Schedule	Warmup–stable–decay (exponential decay)		
Min LR Ratio	0.1		
Weight Decay	0.1		
Gradient Clipping	1.0		

Residual Configurations. mHC uses $m = 4$ residual streams, 20 Sinkhorn–Knopp iterations, and gating scales initialized to 0.01. For the 18B and 28B main experiments, both AttnRes and MSAR group four Transformer layers into each block, corresponding to $S = 8$ residual sublayers and yielding $N = 7$ and $N = 8$ blocks, respectively. The embedding is included as an additional, separate source and is not counted in N. Unless otherwise specified, MSAR uses $m = 4$ streams and initializes its gating scale α_l^{pre} to 0.01, as in mHC. The three causal depthwise convolution branches have kernel sizes $\{4, 8, 12\}$, with convolution weights initialized to zero. The learned pseudo-queries in both MSAR and AttnRes are also initialized to zero.

Evaluation Protocols. We use a lightweight evaluation suite based on OpenCompass [5]. Likelihood-based evaluation selects the candidate answer with the lowest conditional perplexity for MMLU [14] (5-shot), MMLU-Redux [11] (5-shot), CMMLU [19] (5-shot), C-Eval [17] (5-shot), ARC-C [3] (25-shot), and C3 [30] (3-shot). Generation-based evaluation is used for MMLU-Pro [36]

(5-shot), BBH [32] (3-shot), GSM8K [4] (4-shot), MATH [15] (4-shot), HumanEval [2] (0-shot), and LCBench [5] (5-shot), with task-specific answer scoring for reasoning and mathematics and execution-based pass@1 for code.

B Scaling-Law Experiment Details

We evaluate Vanilla, mHC, AttnRes, and MSAR at four model scales with matched backbones and training-token budgets at each scale. Table 4 lists the configurations used in Figure 3. All models use one leading dense Transformer layer, 144 routed experts with top-8 routing, one shared expert, and a sequence length of 8192; common optimizer settings follow Appendix A. mHC and MSAR use $m = 4$ streams, while AttnRes and MSAR share the block sizes listed in the table, yielding 8, 8, 8, and 7 blocks across the four scales. The last block may be shorter, and the embedding is an additional source excluded from these block counts.

Table 4: Scaling-law configurations. All methods use matched backbones and training-token budgets at each scale. The block size is shared by AttnRes and MSAR.

Total Parameters†	Activated Parameters†	Layers	Hidden Dimension	Dense FFN	Expert FFN	Q/KV Heads	Training Tokens (B)	Global Batch Size	Learning Rate	Layers per Block (AttnRes/MSAR)
2.21B	0.34B	15	1024	3104	320	8/4	16.32	160	6.95×10^{-4}	2
3.75B	0.46B	23	1040	3072	352	8/4	33.77	208	5.74×10^{-4}	3
9.47B	1.10B	15	2080	6144	704	16/8	48.30	288	4.82×10^{-4}	2
17.40B	1.77B	28	2112	6144	672	16/8	87.50	384	3.97×10^{-4}	4

†Backbone parameter counts exclude input embeddings.

We fit Eq. 18 separately for each method using its final training losses and compute budgets. With the estimated irreducible loss fixed at $E = 0.72$, ordinary least squares is applied to $\log_2(\mathcal{L} - E)$ against $\log_{10} C$. For fitted slope a and intercept b, the power-law parameters are $A = 2^b$ and $\beta = -a \log_{10} 2$. Table 5 reports the resulting parameters, with C expressed in FLOPs. The iso-loss compute speedup is the ratio of the fitted Vanilla compute at each observed loss to the corresponding method's training FLOPs, averaged across the four model scales.

Table 5: Fitted parameters of $\mathcal{L}(C) = AC^{-\beta} + E$, with $E = 0.72$ fixed for all methods.

Method	A	β
Vanilla	83.594860	0.088406
mHC	82.416109	0.088593
AttnRes	86.376620	0.089672
MSAR	82.217339	0.088921

C Parameter, Compute, and Memory Costs

Parameter Overhead. We count additional parameters relative to the standard residual backbone, including biases and learned scales. The residual modules use RMS normalization without separate learnable affine parameters. Here L counts residual sublayers (twice the number of Transformer layers), and d is the hidden dimension. For mHC, each sublayer has a projection with $md(m^2+2m)$ parameters, m^2+2m biases, and three scalar scales. AttnRes adds one d-dimensional pseudo-query per sublayer and one for the final output read:

$$\Delta P_{\mathrm{mHC}} = L\big[md(m^2+2m) + m^2 + 2m + 3\big], \qquad \Delta P_{\mathrm{AttnRes}} = (L+1)d. \tag{19}$$

MSAR adds md query parameters and m^2d+m+1 mixer parameters per sublayer. Each convolution bank contributes $d\sum_{r=1}^{m-1}\kappa_r$ parameters. Our configuration uses independent banks for the first $L-1$ sublayer outputs and one for the embedding source; the last output is merged without expansion. The embedding uses $\mathcal{E}_{0,r}$. In Block MSAR, completed summary $\boldsymbol{b}_j$ uses $\mathcal{E}_{e_j,r}$, where $e_j = \max \mathcal{B}_j$

is the last sublayer of block j; a nonempty partial $\boldsymbol{b}_n^{<l}$ uses $\mathcal{E}_{l-1,r}$. Completed-source banks remain fixed across subsequent target layers. Thus,

$$\Delta P_{\mathrm{MSAR}} = L\left[\left(m^2 + m + \sum_{r=1}^{m-1} \kappa_r\right) d + m + 1\right]. \tag{20}$$

With $m = 4$ and kernel sizes $\{4, 8, 12\}$, this becomes $L(44d + 5)$, compared with $L(96d + 27)$ for mHC. The resulting parameter increments at the 18B and 28B scales are 5.204M and 7.209M for MSAR, 11.356M and 15.730M for mHC, and 0.120M and 0.166M for AttnRes, respectively.

Training FLOPs. We count a multiply–add as two FLOPs and estimate training FLOPs as three times the forward cost, excluding activation recomputation and optimizer updates. For the main experiments, the backbone cost includes attention and active FFN projections, causal attention, and the LM-head projection. For residual overhead, we account for stream-mapping projections and mixing in mHC using the training implementation's estimator. For AttnRes, we include key normalization, query–key scoring, softmax, value aggregation, and block accumulation. For MSAR, the overhead comprises convolutional stream expansion, per-stream depth attention, and lightweight stream fusion. Table 6 summarizes the estimated parameter and training FLOPs overheads relative to the Vanilla backbone.

Table 6: Residual-module parameter and training FLOPs overheads. $\Delta P/P_{\mathrm{act}}$ is relative to Vanilla's activated parameters (approximately 1.7B and 2.7B at the two scales); $\Delta F/F_{\mathrm{Vanilla}}$ is relative to its total training FLOPs, including the LM head.

	18B		**28B**	
Method	$\Delta P/P_{\mathrm{act}}$	$\Delta F/F_{\mathrm{Vanilla}}$	$\Delta P/P_{\mathrm{act}}$	$\Delta F/F_{\mathrm{Vanilla}}$
mHC	0.67%	0.66%	0.58%	0.60%
AttnRes	0.01%	0.10%	0.01%	0.10%
MSAR	0.31%	0.51%	0.27%	0.48%

Storage, Communication, and Memory Access. As discussed in Section 3.4, Block MSAR matches Block AttnRes in persistent source storage and pipeline communication under the same block partition and pipeline schedule. Table 7 compares activation I/O costs, using optimized execution with kernel fusion for Block MSAR. The costs for the other methods are taken from AttnRes [34]. For Block MSAR, source expansion is fused with depth attention: the expanded source streams are immediately consumed by online softmax without being materialized in memory, and only the resulting m retrieved streams are written back. $\mathcal{H}^{\mathrm{pre}}$ gate computation is fused with the online merge. The retrieved streams are then read for final aggregation, giving a leading-order cost of $(N/S + 4m + 5)d$ per token per residual sublayer.

D Learned Depth-Attention Patterns

We visualize the learned depth-attention weights of the 10B Block MSAR model in Figure 4. The model has 15 Transformer layers, four streams, and eight blocks, with the embedding retained as an additional source. We average the pre-fusion attention weights over tokens from several 2,048-token Pile sequences and show Pre-Attn and Pre-MLP reads separately.

Sublayer-Specific Access Modes. Pre-Attn reads distribute weight more broadly across depth, retaining substantial access to the embedding and earlier blocks. Pre-MLP reads instead tend to concentrate on fewer sources, producing sharper near-diagonal and vertical structures. The near-diagonal band corresponds to the current partial or most recent completed block, while vertical bands indicate repeated access to a particular source across target layers.

Diverse Patterns across Streams. Within the same sublayer type, different streams develop distinct source preferences. In Pre-MLP reads beyond the first block, prominent attention to the embedding is concentrated in Stream 2 (Conv-4) and Stream 3 (Conv-8), while Stream 1 (identity) and Stream 4 (Conv-12) assign relatively little weight to this source. The streams also differ in which blocks they repeatedly access: Stream 2 (Conv-4) and Stream 4 (Conv-12) exhibit strong vertical bands

Table 7: Memory access cost per token per residual sublayer incurred by each residual mechanism during the forward pass. We report leading terms proportional to d, excluding parameter traffic and the internal I/O of f_l. Phase 1 costs are averaged across depth and amortized over the S targets in each block. The Typical column uses $N = 8$ blocks, $S = 16$ residual sublayers per block (128 residual sublayers in total, corresponding to 64 Transformer layers), and $m = 4$ streams.

Method	Operation	Read	Write	Total I/O	
				Symbolic	Typical
Vanilla	Residual merge	$2d$	d	$3d$	$3d$
mHC	Compute mixing coefficients	md	–	$(8m+2)d$	$34d$
	Apply $\mathcal{H}^{\text{pre}}$	md	d		
	Apply $\mathcal{H}^{\text{post}}$	d	md		
	Apply $\mathcal{H}^{\text{res}}$	md	md		
	Residual merge	$2md$	md		
Block AttnRes	Phase 1 (amortized)	$\frac{N}{S}d$	d	$(N/S+5)d$	$5.5d$
	Phase 2	$3d$	d		
Block MSAR	Phase 1: logits (amortized)	$\frac{N}{2S}d$	–	$(N/S+4m+5)d$	$21.5d$
	Phase 1: values (amortized)	$\frac{N}{2S}d$	md		
	Phase 2: partial update and scoring	$2d$	d		
	Phase 2: merge and HPre gate	$(m+1)d$	md		
	Stream aggregation	md	d		

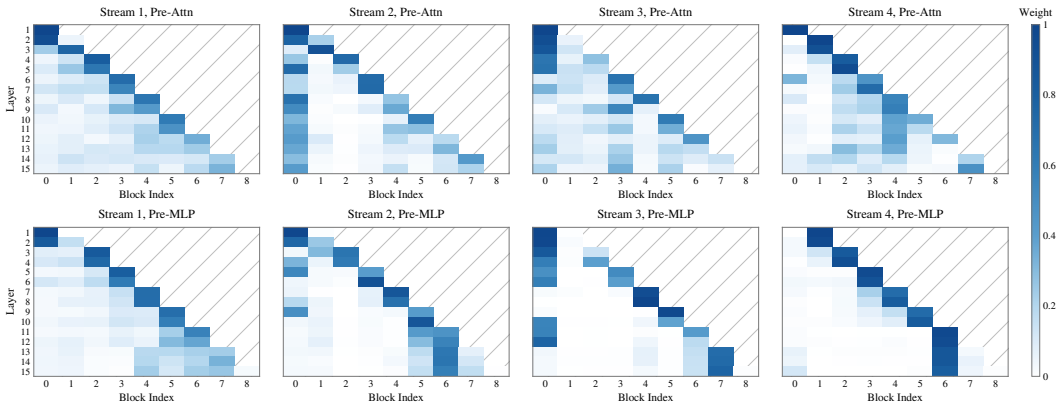


Figure 4: Token-averaged depth-attention weights of the 10B Block MSAR model. Rows show Pre-Attn (top) and Pre-MLP (bottom); columns show the four streams, numbered 1–4. Source 0 is the embedding, and sources 1–8 are blocks, with the current block represented by its nonempty partial sum. Hatched regions mark unavailable sources.

at block 6, whereas Stream 3 (Conv-8) shows a pronounced preference for block 7. These differences produce distinct vertical attention patterns across streams, showing that MSAR learns diverse source-selection distributions rather than repeating a single depth mixture.

E Discussion of Residual Mixing

E.1 Comparison of Residual Mechanisms

Table 8: Comparison of residual update mechanisms. Weight distinguishes architecture-fixed, learned-static (input-independent after training), and input-dependent (dynamic) mixing coefficients. Source indicates the representations directly accessible to the residual mechanism. Branch normalization is absorbed into f_l; depth-attention weights are normalized over available sources.

Method	Update rule	Weight	Source
Single-state recurrence			
Vanilla	$\boldsymbol{h}_l = \boldsymbol{h}_{l-1} + f_{l-1}(\boldsymbol{h}_{l-1})$	Fixed	$\boldsymbol{h}_{l-1}$
ReZero [1]	$\boldsymbol{h}_l = \boldsymbol{h}_{l-1} + \alpha_l f_{l-1}(\boldsymbol{h}_{l-1})$	Static	$\boldsymbol{h}_{l-1}$
Highway [29]	$\boldsymbol{h}_l = (1-\boldsymbol{g}_l)\odot \boldsymbol{h}_{l-1} + \boldsymbol{g}_l \odot f_{l-1}(\boldsymbol{h}_{l-1})$	Dynamic	$\boldsymbol{h}_{l-1}$
Multi-state recurrence			
SiameseNorm [20]	$\boldsymbol{h}_l^1 = \mathrm{Norm}(\boldsymbol{h}_{l-1}^1 + \boldsymbol{y}_{l-1});\quad \boldsymbol{h}_l^2 = \boldsymbol{h}_{l-1}^2 + \boldsymbol{y}_{l-1}$	Fixed	$[\boldsymbol{h}_{l-1}^1, \boldsymbol{h}_{l-1}^2]$
HC/mHC [38, 49]	$\mathbf{H}_l = \mathbf{H}_{l-1}\mathbf{A}_l + f_{l-1}(\mathbf{H}_{l-1}\boldsymbol{\alpha}_{l-1})\boldsymbol{\beta}_{l-1}^\top$	Dynamic	$\mathbf{H}_{l-1}$
DDL [45]	$\mathbf{H}_l = (\mathbf{I} - \beta_l \boldsymbol{k}_l\boldsymbol{k}_l^\top)\mathbf{H}_{l-1} + \beta_l \boldsymbol{k}_l\boldsymbol{v}_l^\top$	Dynamic	$\mathbf{H}_{l-1}$
Cross-layer access			
DenseNet [16]	$\boldsymbol{h}_l = \mathrm{ConvPool}([\boldsymbol{h}_1; f_1(\boldsymbol{h}_1); \ldots; f_{l-1}(\boldsymbol{h}_{l-1})])$	Static	$[\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{l-1}]$
DenseFormer [26]	$\boldsymbol{h}_l = \alpha_{0\to l}\boldsymbol{h}_1 + \sum_{i=1}^{l-1} \alpha_{i\to l} f_i(\boldsymbol{h}_i)$	Static	$[\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{l-1}]$
Block AttnRes [34]	$\boldsymbol{h}_l = \sum_{i=0}^{n-1} \alpha_{i\to l}\boldsymbol{b}_i + \alpha_{n\to l}\boldsymbol{b}_n^j$	Dynamic	$[\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{n-1}, \boldsymbol{b}_n^j]$
MHAR [23]	$\boldsymbol{h}_l = \mathrm{Concat}_{r=0}^{H-1}\left(\sum_{i=0}^{l-1} \alpha_{i\to l}^{(r)} \boldsymbol{v}_i^{[r]}\right)$	Dynamic	$\{[\boldsymbol{v}_0^{[r]}, \ldots, \boldsymbol{v}_{l-1}^{[r]}]\}_{r=0}^{H-1}$
Multi-stream cross-layer access			
DAR [42]	$\boldsymbol{h}_l = g_l^{(0)} \sum_{i\in\mathcal{I}_l} \alpha_{i\to l}^{(0)} \boldsymbol{c}_i^0 + g_l^{(1)} \sum_{i\in\mathcal{I}_l} \alpha_{i\to l}^{(1)} \boldsymbol{c}_i^1$	Dynamic	$[\mathbf{H}_0, \ldots, \mathbf{H}_{n-1}, \mathbf{P}_n^j]$
Block MSAR	$\boldsymbol{h}_l = \sum_{r=0}^{m-1} g_l^{(r)} \sum_{\boldsymbol{b}\in\mathcal{S}_l} \alpha_l^{(r)}(\boldsymbol{b})\mathcal{E}_r[\boldsymbol{b}]$	Dynamic	$\{\mathcal{E}_r[\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{n-1}, \boldsymbol{b}_n^j]\}_{r=0}^{m-1}$

Table 8 compares representative residual mechanisms by their update rules, mixing weights, and accessible sources. The formulations of several prior methods are adapted from AttnRes [34].

We use $\boldsymbol{v}_0$ for the embedding and $\boldsymbol{v}_i = f_i(\boldsymbol{h}_i)$ for branch outputs. ReZero uses a learned scalar α_l, initialized to zero; Highway uses an input-dependent gate $\boldsymbol{g}_l$; and DenseFormer uses learned, input-independent coefficients $\alpha_{i\to l}$. For DenseNet, $\mathrm{ConvPool}$ schematically denotes the convolutional readout of concatenated features. SiameseNorm adds the same branch output $\boldsymbol{y}_{l-1}$ to two states, with normalization applied to one state. In the DDL row, $\mathbf{H}_l$ denotes its matrix-valued state, $\boldsymbol{k}_l$ and $\boldsymbol{v}_l$ denote its update key and value, and β_l controls the delta update; these symbols are local to that row. For the block variants, layer l belongs to block n and accesses completed summaries together with the current partial. The partial term is omitted at the first sublayer of a block. We write $\boldsymbol{b}_n^j = \boldsymbol{b}_n^{<l}$ for the partial after the j preceding sublayers of the current block. For compactness, let $\mathcal{S}_l = [\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{n-1}, \boldsymbol{b}_n^j]$ denote these available sources, with the empty partial excluded. For MSAR, $\mathcal{E}_r[\mathcal{S}_l]$ denotes the list obtained by expanding each source in stream r, $\alpha_l^{(r)}(\boldsymbol{b})$ is its normalized depth-attention weight, and $g_l^{(r)} = [\mathcal{H}_l^{\mathrm{pre}}]_r$ is the MSAR fusion weight. For HC/mHC, the column-stream notation is $\mathbf{H}_l = \mathbf{X}_l^\top \in \mathbb{R}^{d\times m}$, $\boldsymbol{\alpha}_l = (\mathcal{H}_l^{\mathrm{pre}})^\top$, $\boldsymbol{\beta}_l = \mathcal{H}_l^{\mathrm{post}}$, and $\mathbf{A}_l = (\mathcal{H}_{l-1}^{\mathrm{res}})^\top$. Here $\boldsymbol{\alpha}_l$ is a stream-read vector, distinct from the scalar coefficients used by ReZero, DenseFormer, and AttnRes. MHAR and DAR are closely related to MSAR in extending depth retrieval beyond a single attention head or residual stream. For MHAR [23], H is the number of routing heads,

$\boldsymbol{v}_i^{[r]}$ is the r-th feature slice of source $\boldsymbol{v}_i$, and $\alpha_{i\to l}^{(r)}$ is its per-head depth-softmax weight; Concat_r concatenates the retrieved slices. For DAR [42], $\mathbf{H}_0, \ldots, \mathbf{H}_{n-1}$ are completed two-stream history states and $\mathbf{P}_n^j$ is the current partial, omitted when empty. In this row, each available candidate is written as $\mathbf{C}_i = [\boldsymbol{c}_i^0, \boldsymbol{c}_i^1]$, with indices $i \in \mathcal{I}_l$. The cross-stream weights satisfy $\alpha_{i\to l}^{(0)} \propto \phi(\boldsymbol{q}_l^0, \boldsymbol{c}_i^1)$ and $\alpha_{i\to l}^{(1)} \propto \phi(\boldsymbol{q}_l^1, \boldsymbol{c}_i^0)$, each normalized over $\mathcal{I}_l$; $\boldsymbol{q}_l^0, \boldsymbol{q}_l^1$ are learned pseudo-queries, and the sigmoid gates $g_l^{(0)}, g_l^{(1)}$ combine the two retrieved states. The DAR row summarizes this retrieval; its stored states are updated through gated branch writes and block-local stream mixing. Both methods diversify depth retrieval by routing feature subspaces or maintaining distinct residual trajectories at the same token position. MSAR instead constructs retrieval sources from neighboring positions through multi-scale causal convolutions, introducing head diversity through distinct sequence-context views. Each head can therefore select depth histories using a different contextual scale.

E.2 Residual Connections as Structured Matrices

Following AttnRes [34], Figure 5 visualizes depth mixing with $L = 4$ residual sublayers and block size $S = 2$. Rows correspond to target inputs $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_4$ and columns to sources $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_3$. Vanilla assigns unit weights to every preceding source. For mHC, unrolling the recurrence gives

$$M_{i\to l} = \boldsymbol{\beta}_i^\top \mathbf{A}_{i+1\to l}^{\times} \boldsymbol{\alpha}_l, \qquad \mathbf{A}_{a\to b}^{\times} = \mathbf{A}_{a+1}\mathbf{A}_{a+2}\cdots\mathbf{A}_b. \tag{21}$$

The product is ordered in increasing depth, with an empty product equal to the identity. For the embedding, $\boldsymbol{\beta}_0 = \mathbf{1}_m$ and the transition product starts at $\mathbf{A}_2$. Each entry therefore contains the source write vector, intervening state transitions, and target read vector. In mHC, the transition matrices are doubly stochastic.

For Block AttnRes, we use $\boldsymbol{k}_i = \boldsymbol{v}_i$ and the learned pseudo-query $\boldsymbol{w}_l$. As in AttnRes, the figure displays unnormalized scores $\phi(\boldsymbol{w}_l, \boldsymbol{k}) = \exp(\boldsymbol{w}_l^\top \mathrm{RMSNorm}(\boldsymbol{k}))$. The first completed block contributes the single source $\boldsymbol{k}_1 + \boldsymbol{k}_2$; its score spans both constituent columns. Normalization counts this grouped source once, rather than once per column. Dashed cells mark the evolving partial source.

Block MSAR retains this grouping and constructs one score matrix per stream, using $\boldsymbol{w}_l^{(r)}$ and $\mathcal{E}_r[\boldsymbol{b}]$ in place of $\boldsymbol{w}_l$ and $\boldsymbol{b}$. Each stream is normalized independently:

$$\alpha_l^{(r)}(\boldsymbol{b}) = \frac{\phi(\boldsymbol{w}_l^{(r)}, \mathcal{E}_r[\boldsymbol{b}])}{\sum_{\boldsymbol{b}' \in \mathcal{S}_l} \phi(\boldsymbol{w}_l^{(r)}, \mathcal{E}_r[\boldsymbol{b}'])}. \tag{22}$$

The expanded sources also serve as values, and the retrieved streams are combined using $g_l^{(r)}$. The notation $\mathcal{E}_r$ suppresses the convolution-bank index, whose assignment is defined in Appendix C. The MSAR panel thus shows a family of per-stream score matrices, not a scalar mixing matrix acting on unexpanded sources.

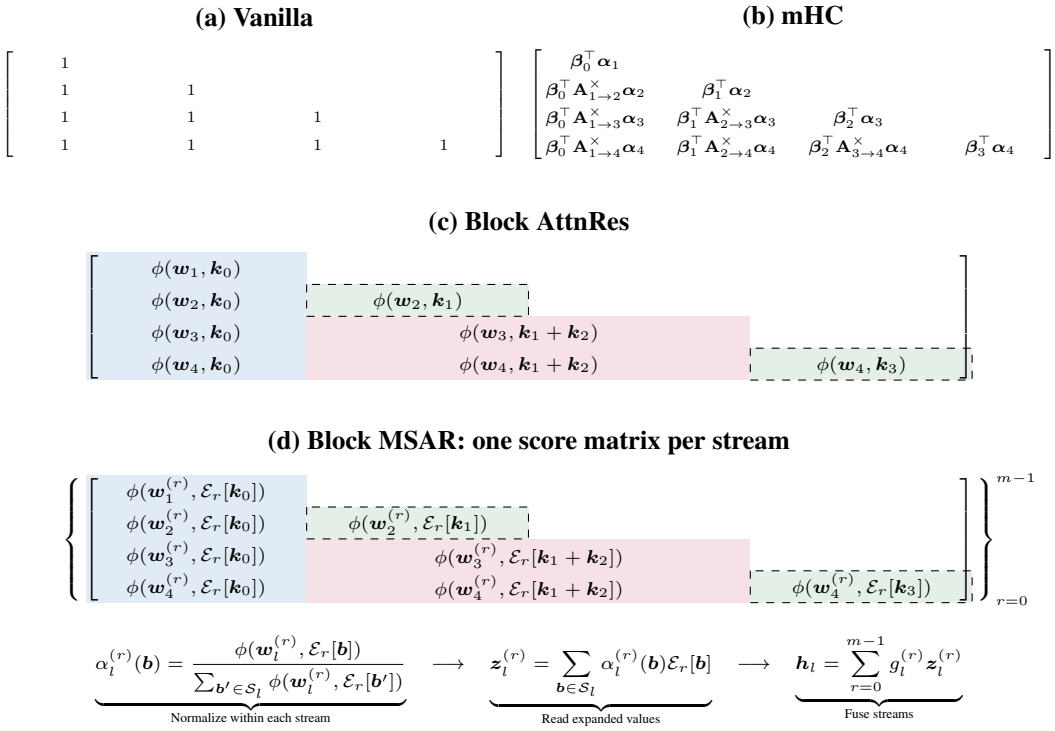


Figure 5: Depth mixing for four residual mechanisms with $L = 4$ and block size $S = 2$. Blank entries are zero. Panels (a)–(b) show effective mixing coefficients; panels (c)–(d) show unnormalized ϕ scores. Blue denotes the embedding source, pink a completed block, and dashed green cells a current partial. A score spanning two columns corresponds to one block source. Block MSAR expands each source before scoring and value aggregation, normalizes separately within each stream, and fuses the retrieved states.