DUAL ATTENTION RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work extends Transformer residual pathways along two complementary axes: historical retrieval provides direct access to earlier computation, whereas multi-stream methods maintain multiple evolving representations. These directions have largely developed independently, leaving unexplored how depth-wise retrieval and multi-stream representation can complement each other within a unified residual architecture. We propose *Dual Attention Residuals* (DAR), which combines these capabilities by allowing the two trajectories to participate reciprocally in depth retrieval. For each target stream, DAR computes depth weights from normalized states in the opposite stream and applies them to values from the target stream's own history. The retrieved states are combined as input to a Transformer branch. To control overhead, DAR retrieves over completed block histories while maintaining two temporary stream states within each block. Under matched training-token budgets, DAR achieves lower validation loss than the evaluated residual baselines across dense models from 0.1B to 1B parameters and a 7B sparse-MoE model. Parameter-matched ablations show that DAR benefits from both its dual-stream representation and reciprocal cross-stream retrieval. Representation and intervention analyses further show that reciprocal cross-stream selection preserves depth-wise diversity and avoids the redundancy or functional imbalance observed in alternative two-stream designs.

1 INTRODUCTION

Residual connections form the main pathway for transporting representations across Transformer depth (He et al., 2016; Vaswani et al., 2017). In a standard Transformer, successive computational branches are accumulated into a single residual stream and propagated sequentially through the network. Recent work extends this pathway along two complementary directions. Historical retrieval methods expand access across depth: Transparent Attention and DenseFormer expose earlier representations to later computation (Bapna et al., 2018; Pagliardini et al., 2024), while Attention Residuals makes this access selective through learned depth-wise retrieval (Kimi Team et al., 2026). Multi-stream residual methods instead expand the pathway into multiple learned trajectories, beginning with Hyper-Connections (Zhu et al., 2025a) and extending to Frac-Connections, MUDDFormer, and mHC (Zhu et al., 2025b; Xiao et al., 2025; Xie et al., 2026). Historical retrieval introduces freedom over which depths to access, whereas multi-stream residuals introduce freedom over which residual trajectories to maintain and combine. These two degrees of freedom have mostly been studied separately, leaving open whether they can interact usefully in historical retrieval.

A straightforward combination is to equip each residual stream with an independent historical retriever. This, however, produces only parallel instances of single-stream retrieval: each trajectory determines which depths to access and supplies the content returned from those depths, without allowing the other trajectory to influence its selection. We call this use of the same trajectory both to score depths and to supply the retrieved content *address-content coupling*. To bring multi-stream flexibility into depth-wise retrieval, the trajectories must instead interact during selection. This makes trajectory diversity part of the retrieval rule: historical content in one stream can receive a high depth weight because the other stream supplies a matching address, even when the content representation would not itself strongly match the target query. The trajectories therefore contribute not only different stored content but also complementary signals for selecting it.

To make multi-stream interaction part of depth-wise retrieval, we introduce Dual Attention Residuals (DAR) with reciprocal cross-stream addressing. For each target stream, DAR computes depth

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.

weights from normalized states in the opposite stream and applies them to values from the target stream's history. The resulting stream-specific states are mixed as input to an unchanged Transformer branch. DAR then updates both residual trajectories through gated branch writes, with constrained mixing transporting the existing partial state within each block. For efficient scaling, Block DAR retrieves from completed block histories at a coarser depth granularity.

Our contributions are summarized as follows:

- We show that independently evolving residual trajectories can serve not only as parallel information pathways, but also as complementary signals for selecting historical depths.
- We propose DAR, which realizes this interaction through reciprocal cross-stream keys and self-stream values while leaving the Transformer branches unchanged.
- We validate DAR through scaling experiments on dense models from 0.1B to 1B parameters and a 7B sparse-MoE model, retrieval ablations, and representation analyses, demonstrating consistent improvements over standard residual connections and Attention Residuals.

2 Related Work

2.1 Residual Transport and Multi-Stream Topologies

Standard residual connections transport a single representation through network depth (He et al., 2016; Vaswani et al., 2017). Transformer normalization improves optimization while preserving this single residual trajectory (Ba et al., 2016; Xiong et al., 2020; Zhang & Sennrich, 2019). Hyper-Connections instead expand the residual pathway into multiple streams and learn how Transformer branches read from and write to them (Zhu et al., 2025a). Frac-Connections and MUDDFormer further enrich cross-layer connectivity, while mHC constrains multi-stream mixing to preserve stable transport at scale (Zhu et al., 2025b; Xiao et al., 2025; Xie et al., 2026). These methods show how multiple residual streams can transport and mix information within a layer. DAR extends this idea across depth by storing both trajectories in the history bank and retrieving over them.

2.2 Depth-Wise Historical Retrieval

Another line of work makes representations from earlier depths directly accessible to later computation. Transparent attention and DenseFormer aggregate intermediate layer states, while Attention Residuals (AttnRes) uses learned pseudo-queries to perform softmax retrieval over previous residual states (Bapna et al., 2018; Pagliardini et al., 2024; Kimi Team et al., 2026). DeepCrossAttention also learns input-dependent combinations of earlier layer outputs through depth-wise cross-attention (Heddes et al., 2025). Delta Attention Residuals replaces cumulative hidden-state candidates with individual branch outputs (Luo et al., 2026). These methods vary the candidate states and aggregation rules. In single-stream historical retrieval, however, the representations that determine depth weights and the values aggregated under them still come from the same residual trajectory. DAR instead uses the opposite trajectory to set depth weights, keeping values in their own stream.

3 Preliminaries

We treat self-attention and feed-forward modules in a Pre-Norm decoder-only Transformer as separate residual branches. Branches start at $\ell = 1$; h_1 is the token embedding. Let $h_\ell \in \mathbb{R}^{T\times d}$ be the input to branch ℓ, and let f_ℓ denote the complete branch transformation, including its input normalization. A standard residual branch updates $h_{\ell+1} = h_\ell + f_\ell(h_\ell)$, where T and d are the sequence length and hidden size. We omit token indices because depth retrieval is token-wise.

Attention Residuals. AttnRes (Kimi Team et al., 2026) replaces cumulative addition with softmax retrieval over the token embedding and preceding branch outputs. Following its original formulation, we use the positive kernel

$$\phi(q,k) = \exp\big(q^\top \mathrm{Norm}_h(k)\big) \tag{1}$$

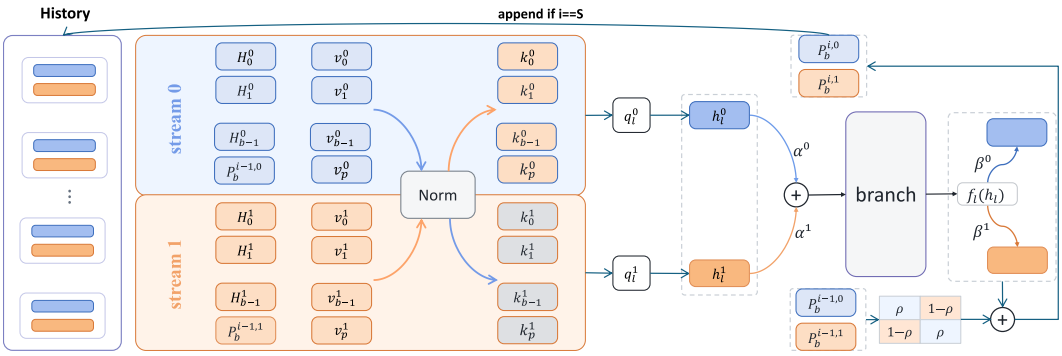


Figure 1: Block DAR, shown for branch $i > 1$ in block b. Opposite-stream keys retrieve self-stream values, which are mixed for an unchanged Transformer branch; its gated output combines with constrained transport of P_b^{i-1} to form P_b^i. The partial path is omitted for $i = 1$, and P_b^S is appended as H_b at the block boundary.

and write the depth-attention weights as

$$\pi_{r\to\ell} = \frac{\phi(q_\ell, k_r)}{\sum_{u=0}^{\ell-1} \phi(q_\ell, k_u)}. \tag{2}$$

For every residual branch ℓ, the query, keys, and values are

$$q_\ell = w_\ell, \qquad k_r = v_r = \begin{cases} h_1, & r = 0, \\ f_r(h_r), & 1 \le r \le \ell - 1, \end{cases} \tag{3}$$

where $w_\ell \in \mathbb{R}^d$ is a branch-specific learnable vector and Norm_h denotes parameter-free RMSNorm along the hidden dimension. Thus $v_0 = h_1$ is the embedding candidate, while $v_r = f_r(h_r)$ for $r \ge 1$ is the output of an earlier residual branch. The input to branch ℓ is

$$h_\ell = \sum_{r=0}^{\ell-1} \pi_{r\to\ell} v_r. \tag{4}$$

This use of v_r for both depth scoring and retrieved content is *address-content coupling*.

4 Methods

DAR combines depth-wise historical retrieval with multi-stream residual pathways, allowing the two trajectories to interact reciprocally during depth selection rather than operating as independent retrievers. As shown in Figure 1, each stream uses the other to weight its own history. The retrieved states are mixed for an attention or feed-forward branch, whose output is written to both streams.

4.1 Overview and Notation

DAR maintains an ordered history bank $\mathcal{H}$. Its r-th history state is

$$H_r = [v_r^0, v_r^1], \qquad v_r^j \in \mathbb{R}^{T\times d}, \quad j \in \{0, 1\}, \tag{5}$$

We initialize $H_0 = [v_0^0, v_0^1] = [h_1, h_1]$ from the token embedding.

Block DAR partitions the residual branches into consecutive blocks. Let $b \ge 1$ index blocks and let the *block size* K denote the number of Transformer layers per block. Because each layer contains an attention branch and a feed-forward branch, a block contains $S = 2K$ residual branches; we use $i \in \{1, \dots, S\}$ to index their positions within the block. With $K = 1$ ($S = 2$), each block contains one Transformer layer. We call this Full DAR. Splitting the layer into separate attention and feed-forward blocks would make the stored streams collinear (Appendix A).

4.2 Block Dual-Stream Retrieval

Before block b, the history bank contains

$$\mathcal{H}_b = (H_0, H_1, \ldots, H_{b-1}). \tag{6}$$

Within block b, DAR maintains two temporary stream states across its residual branches. We denote their values after the first i branches by $P_b^i = [P_b^{i,0}, P_b^{i,1}]$. P_b^1 is constructed from the first branch output, whereas P_b^i for $i > 1$ follows the update rule defined in the next subsection. These states remain local to block b and enter the history bank only at the block boundary. For branch ℓ at position i in block b, we define its retrieval set as

$$\mathcal{C}_\ell = \begin{cases} \mathcal{H}_b, & i = 1, \\ (\mathcal{H}_b, P_b^{i-1}), & i > 1. \end{cases} \tag{7}$$

Only branches with $i > 1$ include the current partial state in retrieval.

Each candidate $C_r = [C_r^0, C_r^1] \in \mathcal{C}_\ell$ contains one state from each stream. For output stream j, DAR uses the opposite-stream state C_r^{1-j} as the key and the same-stream state C_r^j as the value:

$$k_r^j = C_r^{1-j}, \qquad v_r^j = C_r^j, \qquad j \in \{0, 1\}. \tag{8}$$

Every residual branch and stream has an independent query $q_\ell^j \in \mathbb{R}^d$. The resulting depth weights and retrieved stream states are

$$\pi_{r\to\ell}^j = \frac{\phi(q_\ell^j, k_r^j)}{\sum_{C_u \in \mathcal{C}_\ell} \phi(q_\ell^j, k_u^j)}, \tag{9}$$

$$h_\ell^j = \sum_{C_r \in \mathcal{C}_\ell} \pi_{r\to\ell}^j v_r^j. \tag{10}$$

The opposite stream therefore controls where stream j retrieves from, while the aggregated content remains in stream j's own residual space.

4.3 Block-Local Mixing and Gated Writes

Following mHC (Xie et al., 2026), DAR predicts token-wise mixing and write coefficients from the two retrieved states:

$$\bar{h}_\ell = \mathrm{Norm}([h_\ell^0; h_\ell^1]), \tag{11}$$

$$\alpha_\ell = \sigma(\bar{h}_\ell W_{\alpha,\ell} + b_{\alpha,\ell}), \qquad \beta_\ell = 2\sigma(\bar{h}_\ell W_{\beta,\ell} + b_{\beta,\ell}), \tag{12}$$

where $\alpha_\ell \in (0,1)^2$ and $\beta_\ell \in (0,2)^2$. Here Norm denotes parameter-free RMSNorm applied at each token across the concatenated $2d$ hidden features; it has no learned scale or bias. The unchanged Transformer branch receives

$$h_\ell = \alpha_\ell^0 h_\ell^0 + \alpha_\ell^1 h_\ell^1 \tag{13}$$

and produces $u_\ell = f_\ell(h_\ell)$.

Later branches in a block transport the existing partial state with

$$\begin{aligned} \bar{p}_{b,i} &= \mathrm{Norm}([P_b^{i-1,0}; P_b^{i-1,1}]), \\ \rho_{b,i} &= \sigma(\bar{p}_{b,i} W_{\rho,\ell} + b_{\rho,\ell}), \\ M(\rho_{b,i}) &= \begin{bmatrix} \rho_{b,i} & 1-\rho_{b,i} \\ 1-\rho_{b,i} & \rho_{b,i} \end{bmatrix}. \end{aligned} \tag{14}$$

For two streams, this parameterization directly guarantees a doubly stochastic matrix: each row and column sums to one, so no iterative normalization is required. The complete partial-state update is

$$P_b^i = \begin{cases} [\beta_\ell^0 u_\ell, \beta_\ell^1 u_\ell], & i = 1, \\ M(\rho_{b,i}) P_b^{i-1} + [\beta_\ell^0 u_\ell, \beta_\ell^1 u_\ell], & i > 1. \end{cases} \tag{15}$$

Thus the first branch initializes the partial state, while subsequent branches mix its two streams before adding their gated writes. At the block boundary, P_b^S is stored as the new history state

$$H_b = P_b^S. \tag{16}$$

At the network output, learned output queries apply the same cross-key/self-value retrieval to the history bank. The sum $h_{\mathrm{out}}^0 + h_{\mathrm{out}}^1$ is passed to the original final normalization and vocabulary projection. Appendix Figure 6 gives framework-level pseudocode for Block DAR.

4.4 Design Comparisons

AttnRes + VProj. This single-stream control assigns each stored state an independent value projection.

$$\tilde{v}_r = W_{v,r} v_r. \tag{17}$$

VProj therefore adds one $d \times d$ map to each branch's value path, totaling $O(Rd^2)$ parameters for R sources. A source-specific key projection is algebraically equivalent to replacing q_ℓ with $q_{\ell,r} = W_{k,r}^\top q_\ell$, because the key map can be moved to the query side of the dot product. It also introduces the same number of additional parameters as VProj.

Retrieval variants. We compare three parameter-matched retrieval rules that differ only in the key and value sources for output stream j:

$$\begin{aligned} \text{DAR-SelfKV:} \quad & (k_r^j, v_r^j) = (C_r^j, C_r^j), \\ \text{DAR-FixedKV:} \quad & (k_r^j, v_r^j) = (C_r^0, C_r^1), \\ \text{DAR:} \quad & (k_r^j, v_r^j) = (C_r^{1-j}, C_r^j). \end{aligned} \tag{18}$$

DAR-SelfKV. Each stream retrieves independently from its own history, using each state both to score its depth and to supply the retrieved content; DAR-SelfKV thus retains AttnRes's within-trajectory address-content coupling. Beyond this coupling, the streams interact only after retrieval through branch-input mixing and subsequent writes, so neither trajectory can help the other decide which historical depths are relevant. This limited interaction can underuse complementary information carried by the other trajectory and encourage similar retrieval behavior across streams.

DAR-FixedKV. Both outputs use stream 0 as the key source and stream 1 as the value source, with their independent queries producing different depth weights over the same two banks. This fixed assignment makes the interaction one-sided: stream 0 supplies the key features used to determine the depth weights for both outputs, whereas stream 1 supplies all retrieved values. The two trajectories never exchange these roles.

DAR. For output stream j, the opposite trajectory supplies the keys while stream j retains its own values. The retrieved content therefore remains in its original residual trajectory, while the other trajectory contributes a complementary signal for selecting its relevant depths. Applying the rule symmetrically in both directions lets each stream serve as both a content trajectory and an address for the other, enabling reciprocal interaction during historical retrieval. One could instead let each stream score its own history and retrieve values from the other stream; we call this CrossV. With independent queries and learned mixing and write heads, CrossV is an equivalent parameterization of DAR, so we do not treat it as a separate ablation. A proof is given in Appendix A.

5 Experiments

We evaluate DAR under matched training-token budgets across dense models from 0.1B to 1B parameters and a 7B sparse-MoE model, comparing against standard residual connections, Attention Residuals, and mHC-4. Beyond validation loss, we report training efficiency under matched fused implementations, conduct parameter-matched ablations of retrieval rules and value transformations, and analyze the learned streams through historical retrieval patterns, cross-depth representation similarity, and intervention-based output recovery.

5.1 Experimental Setup

Models and data. We evaluate three smaller dense models with 0.1B, 0.3B, and 0.5B non-vocabulary parameters, a main 1B dense model, and a sparse-MoE model. The smaller models are trained on FineWeb-Edu-10B (Penedo et al., 2024) with a token budget equal to $20\times$ their parameter count and evaluated on 10M held-out tokens. The 1B model is trained on 50B FineWeb-Edu-100B tokens. The sparse model has 7.22B total parameters and activates 1.40B per token, including vocabulary parameters; it is trained for 60K steps, corresponding to 31.46B FineWeb-Edu-100B tokens. All runs use 4096-token contexts and global batch sizes of 256 (dense) or 128 (sparse MoE).

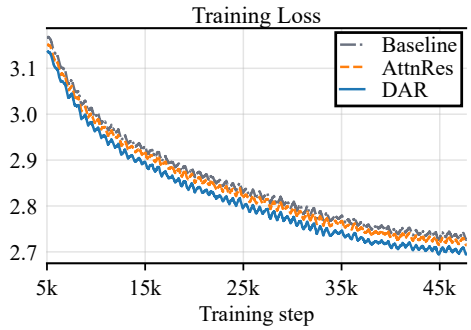


Figure 2: Training loss of the 1B dense models (300-step moving average).

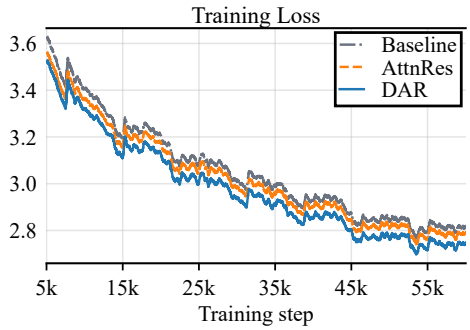


Figure 3: Training language-modeling loss of the 7B sparse-MoE models (300-step moving average).

Table 1: Language-model fit for the 1B models. Paloma reports token perplexity (PPL), byte PPL, and bits per byte (BPB); aggregate PPL/BPB values are micro-averaged over all tokens or bytes from the 16 sources. Lower is better for all columns.

Method	Val. loss	Token PPL	Byte PPL	BPB
Baseline	2.716	21.575	2.040	1.029
AttnRes	2.704	20.707	2.021	1.015
DAR	**2.684**	**20.264**	**2.011**	**1.008**

Training and implementation. All methods within each setting share the same data, backbone, training configuration, and token budget. Because DAR leaves the Transformer branches unchanged, the dominant component of total training FLOPs remains comparable across methods. The mHC-4 baseline uses four streams; to facilitate analysis, we additionally train mHC-2 under the same 0.5B configuration. Models are trained on a single node with eight H800 GPUs using AdamW (Loshchilov & Hutter, 2019). Dense experiments use PyTorch with DeepSpeed (Rasley et al., 2020) and torch.compile (Ansel et al., 2024), whereas sparse-MoE experiments use Megatron-Core (Shoeybi et al., 2019) and fused AttnRes/DAR operator implementations. For the parameter-matched 0.5B ablation, we set the FFN width of AttnRes + VProj to 4992. We use Block DAR in the main dense and sparse-MoE experiments and evaluate both Block and Full DAR in the smaller models. Detailed configurations and parameter counts are provided in Appendix Tables 6–11.

5.2 Main Results

1B-parameter setting. We first assess the 1B models using FineWeb-Edu validation loss and Paloma token PPL, byte PPL, and BPB (Magnusson et al., 2024). Table 1 compares the baseline, block-form Attention Residuals, and block-form DAR.

DAR performs best on all four reported measures and obtains the lowest perplexity on 15 of the 16 Paloma domains; Attention Residuals is best only on 4chan. The complete domain results are provided in Appendix D. Figure 2 further shows that DAR maintains lower smoothed training loss than both baselines for most of training.

To test whether these gains extend beyond language modeling to downstream task performance, we evaluate the same checkpoints zero-shot with the default lm-evaluation-harness task formulations (Biderman et al., 2024). The eight tasks are HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ARC-Easy (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2020), BoolQ (Clark et al., 2019), OpenBookQA (Mihaylov et al., 2018), SocialIQA (Sap et al., 2019), and SciQ (Welbl et al., 2017). Table 2 shows that DAR improves over the baseline on all eight tasks and achieves the best score on five; Attention Residuals is best on PIQA, ARC-Easy, and SciQ.

Table 2: Zero-shot performance (%) of the 1B models. Higher is better; Avg. is the unweighted mean across tasks.

Method	HellaSwag	PIQA	ARC-E	WinoGrande	BoolQ	OpenBookQA	SocialIQA	SciQ	Avg.
Baseline	53.96	72.14	49.92	53.67	61.16	33.00	40.02	72.40	54.53
AttnRes	54.65	**73.94**	**52.10**	53.83	58.84	31.40	40.84	**76.00**	55.20
DAR	**55.35**	73.01	51.30	**54.22**	**62.45**	**33.20**	**40.94**	74.60	**55.63**

Table 3: Validation loss in the small-scale dense suite. Model size excludes vocabulary parameters.

Method	0.1B	0.3B	0.5B
Baseline	3.577	2.866	2.796
DAR-Block	**3.504**	**2.803**	2.674
DAR-Full	3.515	2.804	**2.653**
AttnRes-Block	3.572	2.823	2.754
AttnRes-Full	3.541	2.827	2.708
mHC-4	3.531	2.844	2.764

Other model sizes. Table 3 compares the residual baseline, mHC-4, and the block and full forms of DAR and Attention Residuals.

DAR consistently outperforms both the baseline and mHC-4. Compared with the baseline, its best variant reduces validation loss by 2.03%, 2.22%, and 5.12% at 0.1B, 0.3B, and 0.5B, respectively. Block DAR performs best at 0.1B and 0.3B, while Full DAR performs best at 0.5B.

Sparse-MoE setting. To test whether DAR remains effective as the model scales up, we further evaluate it on a 7B sparse-MoE model. DAR achieves the lowest validation loss (2.739), compared with 2.785 for Attention Residuals and 2.814 for the baseline. Its training curve remained below both baselines for most of training (Figure 3). The reported values and curves include only the language-modeling loss and exclude the auxiliary router load-balancing loss. Appendix Figure 7 shows similar router balance across the three models.

Appendix Figure 8 compares the historical retrieval distributions learned by the two sparse-MoE DAR streams before the attention and MLP branches. This complementary retrieval indicates a division of labor across depth: the streams expose different historical contexts to the same Transformer branch rather than reproducing a single retrieval pattern.

5.3 Ablation Study

Table 4 compares the retrieval designs under parameter-matched training. VProj performs worse than AttnRes, indicating that address–content coupling cannot be effectively addressed by value projection alone. All multi-stream variants outperform AttnRes, showing that multi-stream residual pathways and historical retrieval are complementary. DAR has the lowest loss among the distinct two-stream rules, supporting reciprocal cross-stream retrieval over self-only and fixed-role retrieval in this setting.

5.4 Stream Analysis

We then use cross-stream CKA and single-stream interventions to examine whether the two streams learn distinct representations and both contribute to predictions.

Representation similarity between streams. We use cross-stream CKA (Figure 4) to test whether the two streams remain distinct rather than duplicate one another across depth. DAR shows low similarity between the two streams at different depths, while alignment is confined near corresponding late layers, indicating distinct depth-wise trajectories without widespread redundancy. Using four FineWeb-Edu-10B validation sequences of length 4096, we collect post-MLP stream states over the final 128 tokens and compute cross-stream, cross-depth linear CKA (Kornblith et al., 2019). CKA

Table 4: Key–value retrieval ablations in the 0.5B Full setting under parameter-matched training. The key and value columns correspond to the right-hand side of each retrieval definition in the Design Comparisons subsection.

Method	Key source	Value source	Val. loss
AttnRes	v_r	v_r	2.708
AttnRes + VProj	v_r	$\tilde{v}_r$	2.757
DAR-SelfKV	C_r^j	C_r^j	2.678
DAR-FixedKV	C_r^0	C_r^1	2.690
DAR	C_r^{1-j}	C_r^j	2.653

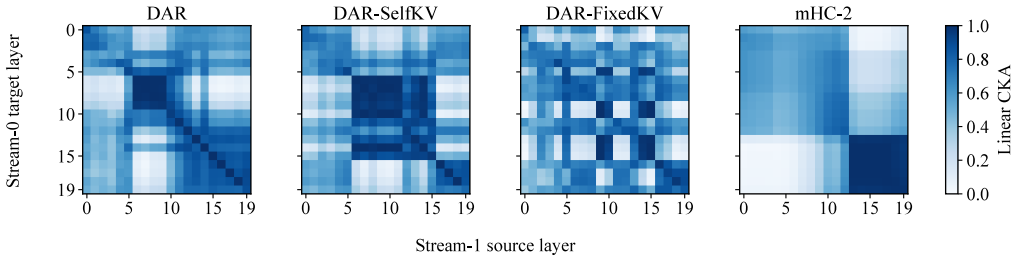


Figure 4: Cross-stream linear CKA for post-MLP residual-stream states in the 0.5B models. Rows are stream-0 layers and columns are stream-1 layers; mHC-2 is the two-stream configuration used for this analysis.

compares pairwise similarities among token representations; a high value means that token pairs represented similarly at one depth are also represented similarly at the other.

By contrast, DAR-SelfKV's early alignment expands into a broad block across the middle layers, indicating that many different depths in the two streams learn similar representations and thus provide less depth-specific diversity for retrieval. DAR-FixedKV produces repeated vertical bands, meaning that a few stream-1 depths are similar to many stream-0 depths. This many-to-one pattern suggests that fixed key–value roles make multiple depths reuse similar representations, reducing the depth-wise diversity available for historical retrieval. For mHC-2, many pairs of late layers at different depths have high CKA, forming a broad cross-depth region. This pattern is consistent with a lower effective depth, as representations from distinct late layers become less distinguishable.

Contribution of each stream to predictions. To test whether each stream contributes to the model's prediction, we perform layer-wise interventions following Peng et al. (2026). Figure 5 shows the desired result for DAR: retaining either stream improves over removing both, but neither stream alone consistently recovers the clean prediction, indicating complementary rather than redundant contributions. Immediately after the MLP at layer n, we separately keep only stream 0, keep only stream 1, or remove both, and then continue the forward pass. For stream j, the rescue gain measures how much keeping it reduces the KL divergence from the clean output relative to removing both:

$$R_{n,j} = 1 - \frac{\mathrm{KL}(p_{\text{clean}} \| p_{\text{keep-}j})}{\max(\mathrm{KL}(p_{\text{clean}} \| p_{\text{zero-}both}), 10^{-12})}.$$

Thus, a score near 1 means that keeping stream j nearly reproduces the clean output, whereas a score near 0 means that it helps little beyond removing both streams.

In contrast, despite high cross-stream CKA, DAR-SelfKV's KL rescue consistently favors stream 1. This strong asymmetry indicates that independent self-retrieval does not let the two trajectories cooperate effectively: without DAR's reciprocal cross-stream selection, one stream becomes functionally dominant while the other remains underused. To explain how this imbalance arises, we examine β_n^j, which directly controls how much branch output enters stream j. Appendix Figure 9 shows that SelfKV has a much larger write imbalance than DAR, consistent with its asymmetric rescue. DAR-FixedKV's one-sided recovery follows directly from its fixed roles: stream 1 stores all retrievable values, whereas stream 0 supplies only their keys and therefore cannot recover content once stream 1

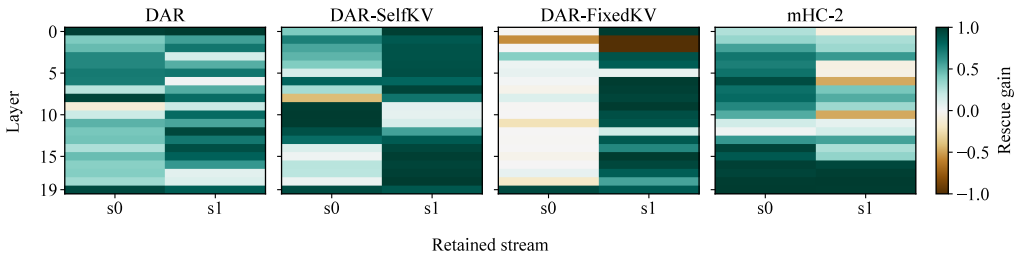


Figure 5: Post-MLP residual-stream rescue gain by layer in the 0.5B models. Columns indicate the retained stream; values are normalized against zeroing both streams.

Table 5: Training efficiency for the 7B sparse-MoE models on one node with eight H800 GPUs. Throughput excludes the first 300 steps to reduce sensitivity to early MoE-routing transients; peak memory is reported per GPU.

Method	Avg. tok/s (K)	Peak GiB
Baseline	114	59.8
AttnRes	104	67.4
DAR	94	70.5

is removed. Because stream 0 is never used as retrievable content, this fixed assignment wastes part of the two-stream capacity, consistent with FixedKV's higher validation loss than DAR in Table 4. For mHC-2, both streams individually show high late-layer rescue, matching the broad high-CKA region across late depths: because the streams encode similar information, either can replace much of what is lost when the other is removed. Together, the two analyses indicate redundancy rather than complementary specialization, consistent with a lower effective depth. Negative stream-1 rescue at several middle layers further shows that this redundancy does not ensure useful cooperation throughout the network.

5.5 Training Efficiency

For the 7B sparse-MoE efficiency comparison, Attention Residuals and DAR use matched fused operators implemented in Triton (Tillet et al., 2019; Kimi Team et al., 2026). DAR retrieves from both streams jointly, fuses residual reads with gate and branch-input computation, and uses a fused epilogue for branch-output writes. The Transformer branches remain unchanged. More details are provided in Appendix B.

Relative to the residual baseline, Attention Residuals and DAR reduce throughput by approximately 8.8% and 17.5%, while increasing per-GPU peak memory by 7.6 and 10.7 GiB, respectively. Under the matched fused implementations, DAR adds approximately 0.363M total and active parameters over Attention Residuals, reduces throughput by approximately 9.6%, and uses 3.1 GiB (4.6%) more peak memory per GPU. Most of the memory increase already appears with single-stream historical retrieval; DAR's cross-stream retrieval adds a smaller but measurable cost.

6 Conclusion

Dual Attention Residuals formulates historical residual retrieval as a joint choice over depth and stream roles. It uses cross-stream keys to compute historical depth weights and applies them to self-stream values, without changing the Transformer branches. Across dense and sparse models, DAR improves validation loss. Together, the ablations and stream analyses show that DAR benefits from dual-stream representation and reciprocal retrieval while using the two streams more complementarily than the tested alternatives.

AI USE STATEMENT

Generative AI tools were used to polish the language of the manuscript and to implement code for this work. The authors take responsibility for the final text, code, and research claims.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Jason Ansel, Edward Z. Yang, Horace He, et al. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 929–947. ACM, 2024. doi: 10.1145/3620665.3640366. URL https://doi.org/10.1145/3620665.3640366.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. Training deeper neural machine translation models with transparent attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3028–3033. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1338. URL https://aclanthology.org/D18-1338/.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models, 2024. URL https://arxiv.org/abs/2405.14782.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(5):7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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, 2018. URL https://arxiv.org/abs/1803.05457.

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. doi: 10.1109/CVPR.2016.90. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Mike Heddes, Adel Javanmard, Kyriakos Axiotis, Gang Fu, Mohammadhossein Bateni, and Vahab Mirrokni. DeepCrossAttention: Supercharging transformer residual connections. In *Proceedings*

of the 42nd International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, pp. 22881–22903. PMLR, 2025. URL https://proceedings.mlr.press/v267/heddes25a.html.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals, 2026. URL https://arxiv.org/abs/2603.15031.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://iclr.cc/virtual/2019/poster/935.

Cheng Luo, Zefan Cai, and Junjie Hu. Delta attention residuals, 2026. URL https://arxiv.org/abs/2605.18855.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Pete Walsh, Yanai Elazar, Kyle Lo, Dirk Groeneveld, Iz Beltagy, Hannaneh Hajishirzi, Noah A. Smith, Kyle Richardson, and Jesse Dodge. Paloma: A benchmark for evaluating language model fit. In *Advances in Neural Information Processing Systems*, volume 37, pp. 64338–64376, 2024. doi: 10.52202/079017-2052. URL https://papers.nips.cc/paper_files/paper/2024/hash/760b2d94398aa61468aa3bc11506d9ea-Abstract-Datasets_and_Benchmarks_Track.html.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. DenseFormer: Enhancing information flow in transformers via depth weighted averaging. In *Advances in Neural Information Processing Systems 37*, pp. 136479–136508, 2024. doi: 10.52202/079017-4336. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f67449c7ab72f441d3a713b046c6818c-Abstract-Conference.html.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems 37*, pp. 30811–30849, 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

William Peng, Josheev Rai, Kevin Tseng, Siwei Wang, and Sean Wu. Ablate and rescue: A causal analysis of residual stream hyper-connections. In *ICLR 2026 Workshop on Scientific Methods for Understanding Deep Learning*, 2026. URL https://openreview.net/forum?id=Zu3B2TKOPV.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 3505–3506. ACM, 2020. doi: 10.1145/3394486.3406703. URL https://doi.org/10.1145/3394486.3406703.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(5):8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 4463–4473. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Noam Shazeer. GLU variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism, 2019. URL https://arxiv.org/abs/1909.08053.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19. ACM, 2019. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106. Association for Computational Linguistics, 2017. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. MUDDFormer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68440–68458. PMLR, 2025. URL https://proceedings.mlr.press/v267/xiao25d.html.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Mingyu Xu, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=mDhyxu8WRb.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 2020. URL https://proceedings.mlr.press/v119/xiong20b.html.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems 32*, pp. 12381–12392, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=9FqARW7dwB.

Defa Zhu, Hongzhi Huang, Jundong Zhou, Zihao Huang, Yutao Zeng, Banggu Wu, Qiyang Min, and Xun Zhou. Frac-connections: Fractional extension of hyper-connections, 2025b. URL https://arxiv.org/abs/2503.14125.

A Additional Method Properties

CrossV equivalence. For output stream j, DAR uses $(k_r^j, v_r^j) = (C_r^{1-j}, C_r^j)$, while CrossV uses $(k_r^j, v_r^j) = (C_r^j, C_r^{1-j})$. Let the CrossV query for stream j equal the DAR query for stream $1-j$ at every branch. The two rules then use the same keys and values with exchanged output labels, giving

$$\pi^j_{\mathrm{CrossV},r\to\ell} = \pi^{1-j}_{\mathrm{DAR},r\to\ell}, \qquad h^j_{\mathrm{CrossV},\ell} = h^{1-j}_{\mathrm{DAR},\ell}. \tag{19}$$

Let S swap the two d-dimensional blocks in the concatenated readout. Because Norm is parameter-free RMSNorm across all $2d$ features, $\mathrm{Norm}(Sx) = S\mathrm{Norm}(x)$. Permuting the input weights of the mixing and write heads, and the two outputs of the mixing head, therefore gives $\alpha^j_{\mathrm{CrossV},\ell} = \alpha^{1-j}_{\mathrm{DAR},\ell}$ and $\beta^j_{\mathrm{CrossV},\ell} = \beta^j_{\mathrm{DAR},\ell}$. The branch input $\sum_j \alpha^j_\ell h^j_\ell$ is identical in both models, so the unchanged Transformer branch produces the same output. The partial-state transport gate ρ receives the same partial state and can retain its parameters; identical write coefficients then give identical partial states and history banks at every branch. Swapping the learned output queries likewise exchanges the two final readouts, whose sum is unchanged. Thus CrossV and DAR represent the same input-output functions.

Why blocks contain at least one full layer. With only one residual branch, there is no earlier partial state to transport, and the two-stream history state after branch ℓ is

$$H_\ell = [\beta^0_\ell u_\ell, \beta^1_\ell u_\ell]. \tag{20}$$

At every token position p, the positive write coefficients give

$$H^1_{\ell,p} = \frac{\beta^1_{\ell,p}}{\beta^0_{\ell,p}} H^0_{\ell,p}. \tag{21}$$

The two stored streams are collinear, so they cannot retain distinct summaries of the branch output. Independent queries can still select different depths, but they do not remove this within-state degeneracy. The transport matrix $M(\rho)$ is also unused because no earlier partial state exists. We therefore use one Transformer layer, containing an attention and a feed-forward branch, as the smallest block ($K = 1$, $S = 2$).

B Fused Residual-History Operators

The 7B sparse-MoE experiments use fused operators, whereas dense runs use direct PyTorch with torch.compile; both implement the same equations and objective. We retain the main-paper notation and make only token position p explicit; r indexes history states in $\mathcal{H}_b$, while "part" denotes the optional current partial candidate. The same equations reduce to Full DAR at $K = 1$, yielding a layer-wise history.

Block-form pseudocode. Figure 6 gives framework-level pseudocode for Block DAR. It retains the model logic while omitting fused-kernel and caching optimizations.

Packed history bank. History states $H_0, \ldots, H_{b-1}$ are stored separately from the optional current partial state P_b^{i-1}. At each append, a packing kernel stores the values in $H_r = [v_r^0, v_r^1]$ and their normalized cross-stream key views $\widehat{k}_r^j = \mathrm{Norm}_h(v_r^{1-j})$; these are only cached views of $\mathcal{H}_b$. For token p, its depth score is

$$a^j_{r\to\ell,p} = (q^j_\ell)^\top \widehat{k}^j_{r,p}. \tag{22}$$

History-bank prefix. For every (ℓ, p, j), the prefix kernel computes

$$m^j_{\ell,p} = \max_{0\le r<b} a^j_{r\to\ell,p}, \tag{23}$$

$$Z^j_{\ell,p} = \sum_{r=0}^{b-1} \exp(a^j_{r\to\ell,p} - m^j_{\ell,p}), \tag{24}$$

$$U^j_{\ell,p} = \sum_{r=0}^{b-1} \exp(a^j_{r\to\ell,p} - m^j_{\ell,p}) v^j_{r,p}. \tag{25}$$

Here $m^j_{\ell,p}$, $Z^j_{\ell,p}$, and $U^j_{\ell,p}$ are the maximum logit, shifted denominator, and shifted value numerator. The kernel materializes the logits and max-shifted scores for backward, but not a second tensor of

```
def dual_block_attn_res(blocks, partial_block, q0, q1, norm):
    # blocks already include the duplicated token embedding
    states = blocks if partial_block is None else blocks + [partial_block]
    V0 = torch.stack([state[0] for state in states])
    V1 = torch.stack([state[1] for state in states])
    K0, K1 = norm(V1), norm(V0)             # cross-stream keys
    pi0 = softmax(einsum(q0, K0), dim=0)
    pi1 = softmax(einsum(q1, K1), dim=0)
    h0 = einsum(pi0, V0)                    # self-stream values
    h1 = einsum(pi1, V1)
    return alpha_beta_mix(h0, h1)            # branch input h and write gate beta

def mhc_update(partial_block, branch_out, beta, rho_gate):
    write = gated_write(beta, branch_out)
    if partial_block is None:                # first branch in a block
        return write
    rho = rho_gate(partial_block)
    return M(rho) @ partial_block + write     # constrained mHC transport

def forward(self, blocks, hidden_states):
    partial_block = hidden_states
    # apply Block DAR before attention
    h, beta = dual_block_attn_res(
        blocks, partial_block, self.attn_q0, self.attn_q1, self.attn_res_norm)

    # if reaching a block boundary, start a new block
    # block_size counts Transformer layers
    if self.layer_number % self.block_size == 0:
        blocks.append(partial_block)
        partial_block = None

    attn_out = self.attn(self.attn_norm(h))
    partial_block = mhc_update(
        partial_block, attn_out, beta, self.attn_rho_gate)

    # apply Block DAR before MLP
    h, beta = dual_block_attn_res(
        blocks, partial_block, self.mlp_q0, self.mlp_q1, self.mlp_res_norm)
    mlp_out = self.mlp(self.mlp_norm(h))
    partial_block = mhc_update(
        partial_block, mlp_out, beta, self.mlp_rho_gate)
    return blocks, partial_block
```

Figure 6: Block DAR pseudocode in the same control flow as Block AttnRes, with cross-stream retrieval and mHC partial-state updates. The variable `partial_block` corresponds to P_b^{i-1} in the main paper.

normalized weights; it directly returns

$$\zeta^j_{\ell,p} = m^j_{\ell,p} + \log Z^j_{\ell,p}, \qquad \widetilde{h}^{j,\mathrm{hist}}_{\ell,p} = U^j_{\ell,p}/Z^j_{\ell,p}. \tag{26}$$

Here $\zeta^j_{\ell,p}$ and $\widetilde{h}^{j,\mathrm{hist}}_{\ell,p}$ are the history-bank log-partition and retrieved state. The matched kernel family pairs the attention and feed-forward queries; DAR additionally evaluates both streams together.

Exact online-softmax merge. For branch $i > 1$ in block b, the additional candidate is P^{i-1}_b. Its cross-stream key, self-stream value, and score are

$$\widehat{k}^j_{\mathrm{part},\ell,p} = [\mathrm{Norm}_h(P^{i-1,1-j}_b)]_p, \tag{27}$$

$$v^j_{\mathrm{part},\ell,p} = (P^{i-1,j}_b)_p, \tag{28}$$

$$a^j_{\mathrm{part}\to\ell,p} = (q^j_\ell)^\top \widehat{k}^j_{\mathrm{part},\ell,p}. \tag{29}$$

The scalar

$$\lambda^j_{\ell,p} = \sigma(\zeta^j_{\ell,p} - a^j_{\mathrm{part}\to\ell,p}) \tag{30}$$

is the total softmax mass assigned to the history-bank candidates. Consequently, the residual-read kernel obtains exactly the main paper's retrieval over $\mathcal{C}_\ell = (\mathcal{H}_b, P^{i-1}_b)$:

$$h^j_{\ell,p} = \lambda^j_{\ell,p}\widetilde{h}^{j,\mathrm{hist}}_{\ell,p} + (1-\lambda^j_{\ell,p})v^j_{\mathrm{part},\ell,p}. \tag{31}$$

For $i = 1$, no partial candidate exists and $h^j_{\ell,p} = \widetilde{h}^{j,\mathrm{hist}}_{\ell,p}$. This is the two-phase online-softmax merge of Attention Residuals (Kimi Team et al., 2026); it is algebraically identical to normalizing all candidates together.

Fusion and backward boundaries. The fused residual read performs the exact history–partial merge in Equation 31 and predicts α_ℓ and β_ℓ. For $i > 1$, it also predicts $\rho_{b,i}$ and returns the transported state $M(\rho_{b,i})P^{i-1}_b$ together with the branch input h_ℓ. The unchanged branch f_ℓ executes separately; for attention, a post-branch epilogue fuses branch-output handling with the gated write, while the feed-forward path applies the same update directly. Backward retains the logits, shifted scores, and reduction outputs, but reconstructs packed v^j_r and normalized $\widehat{k}^j_r$ from the raw H_r states and recomputes current-partial keys as needed. Its custom kernels differentiate the residual-history path only; the Transformer branches remain outside it and are not recomputed. After the final block is appended, the learned output queries reuse the same history-bank read without a partial candidate. The current kernels reflect an initial optimization effort rather than a fully tuned implementation, leaving room for additional speedups through further kernel engineering.

C Experimental Configurations

Tables 6–11 report the detailed architecture, optimization, distributed-training, and parameter-count configurations for the dense and sparse-MoE experiments. Each reported result is obtained from a single training run.

The sparse backbone uses grouped-query attention (Ainslie et al., 2023), RoPE (Su et al., 2024), RMSNorm (Zhang & Sennrich, 2019), SwiGLU (Shazeer, 2020), and sparsely gated mixture-of-experts routing (Shazeer et al., 2017).

Table 6: Training and architecture details for the 1B language-modeling experiment. The same optimization recipe is used for all methods in the 1B comparison.

Setting	Value	Setting	Value
Backbone	Dense decoder-only Transformer	Hidden size	2048
Dataset	FineWeb-Edu-100B	Embedding size	1024
Training tokens	50B	Layers	20
Hardware	$8\times$ H800 GPUs	Attention / KV heads	16 / 8
GPU nodes	1	Feed-forward width	6656
Training framework	DeepSpeed	Block size K	2 layers
Compilation	torch.compile max-autotune	Optimizer	AdamW
Sequence length	4096	Peak learning rate	3.0×10^{-4}
Micro / global batch	2 / 256	Minimum learning rate	3.0×10^{-5}
Grad. accumulation	16	Warmup steps	2000
ZeRO stage	1	Weight decay	0.1
Activation recompute	Disabled	AdamW β_1, β_2	0.9, 0.95
Param./grad. sharding	Disabled	AdamW ϵ	1.0×10^{-8}
Random seed	42	Gradient clipping	1.0

Table 7: Architecture and training settings for the other dense model sizes. All runs use FineWeb-Edu-10B, DeepSpeed, torch.compile in `max-autotune` mode, and random seed 42. They run on a single node with eight H800 GPUs using micro/global batch sizes 2/256, 16 gradient-accumulation steps, ZeRO Stage 1, no activation recomputation or parameter/gradient sharding, and 300 warmup steps. Within each size, the residual mechanism changes across methods while the data order, dense decoder backbone, optimization recipe, batch size, and token budget are held fixed. The column K reports the block size, i.e., the number of Transformer layers per block, for block-form Attention Residuals and block-form DAR; Full Attention Residuals and Full DAR use $K = 1$.

Size	Tokens	Seq.	Batch	Layers	H/KV	d_{model}	d_{emb}	d_{ff}	K	mHC streams
0.1B	2B	4096	256	16	10/5	640	640	2560	2	4
0.3B	6B	4096	256	20	16/4	1024	1024	4096	2	4
0.5B	10B	4096	256	20	20/10	1280	1024	5120	2	4

Table 8: Exact dense-model parameter counts. "Excl." and "incl." indicate whether vocabulary parameters are excluded or included. The scale labels in the main paper refer to the nominal non-vocabulary model size.

Nominal scale	Method	Excl. vocabulary (M)	Incl. vocabulary (M)
0.1B	Baseline	98.325	182.211
0.1B	AttnRes-Block	98.336	182.222
0.1B	AttnRes-Full	98.346	182.232
0.1B	DAR-Block	98.449	182.335
0.1B	DAR-Full	98.571	182.457
0.1B	mHC-4	100.292	184.178
0.3B	Baseline	304.129	438.347
0.3B	AttnRes-Block	304.151	438.368
0.3B	AttnRes-Full	304.170	438.388
0.3B	DAR-Block	304.377	438.595
0.3B	DAR-Full	304.621	438.839
0.3B	mHC-4	308.062	442.280
0.5B	Baseline	491.572	659.345
0.5B	AttnRes-Block	491.599	659.372
0.5B	AttnRes-Full	491.624	659.396
0.5B	DAR-Block	491.882	659.655
0.5B	DAR-Full	492.187	659.959
0.5B	mHC-4	496.489	664.261
1B	Baseline	1,069.631	1,338.067
1B	AttnRes-Block	1,069.674	1,338.110
1B	DAR-Block	1,070.127	1,338.563

Table 9: Architecture of the sparse-MoE backbone. The residual baseline, Attention Residuals, and DAR share this backbone, the pretokenized FineWeb-Edu-100B data, optimization schedule, and distributed-training configuration. Attention Residuals and DAR use block size $K = 3$; the residual mechanism and its matched fused implementation are the only method-specific components. The expert FFN width is the intermediate width of each routed expert.

Architecture setting	Value	Architecture setting	Value
Transformer layers	24	Context length	4096
Hidden size	2048	Expert FFN width	704
Attention heads / KV groups	16 / 8	Vocabulary size	131,072
Position encoding	RoPE	RoPE base	10^6
Normalization / activation	RMSNorm / SwiGLU	Linear biases	Disabled
Attention backend	Fused	Input–output embeddings	Tied
Block size K	3 layers	Experts	64
Experts selected per token	8	Router balancing	Auxiliary loss
Auxiliary-loss coefficient	0.01	Expert computation	Grouped GEMM
Token dispatcher	All-to-all		

Table 10: Exact sparse-MoE parameter counts. Active parameters are those used per token under top-8 expert routing; vocabulary parameters are reported separately because they contribute equally to all methods.

Method	Total excl. vocab. (B)	Total incl. vocab. (B)	Active excl. vocab. (B)	Active incl. vocab. (B)
Baseline	6.949	7.217	1.136	1.404
AttnRes	6.949	7.217	1.136	1.404
DAR	6.949	7.218	1.136	1.405

Table 11: Training and distributed-system configuration for the sparse-MoE comparison. TP, PP, EP, and CP denote tensor, pipeline, expert, and context parallelism. On eight GPUs, TP $= 2$ gives data-parallel size 4; micro-batch size 2 with 16 accumulation steps yields global batch 128. At context length 4096, 60,000 iterations correspond to 31.46B training tokens.

Training setting	Value	Training setting	Value
Dataset	FineWeb-Edu-100B (pretokenized)	Precision	BF16
Training framework	Megatron-Core	Activation recomputation	Selective
Training iterations	60,000	Training tokens	31.46B
Micro / global batch size	2 / 128	Hardware	1 node / 8 H800 GPUs
Optimizer	AdamW	Weight decay	0.1
AdamW β_1, β_2	0.9, 0.95	AdamW ϵ	10^{-8}
Peak / minimum learning rate	3×10^{-4} / 3×10^{-5}	Gradient clipping	1.0
Warmup	2,000 iterations	Decay schedule	Cosine over 60,000 iterations
Distributed optimizer	Enabled	Sequence parallelism	Enabled
TP / PP / EP / CP	2 / 1 / 4 / 1	Grad. accumulation	16
Data-loader mode	Cyclic	Random seed	42

D Detailed 1B Evaluation

The three 1B checkpoints are evaluated once under identical protocols. Paloma (Magnusson et al., 2024) uses its official evaluation implementation; Table 12 reports token perplexity for each evaluated domain group. For source s with N_s scored tokens, the aggregate token perplexity is

$$\mathrm{PPL}_{\mathrm{token}} = \exp\left(-\frac{\sum_{s=1}^{16}\sum_{t=1}^{N_s}\log p(x_{s,t})}{\sum_{s=1}^{16} N_s}\right). \tag{32}$$

Byte PPL and BPB use the same pooled log-likelihood normalized by the total number of bytes, with exponential and base-2 conversions, respectively. Thus, all aggregate metrics are token- or byte-level micro averages over the 16 sources, rather than macro averages of their source-level perplexities. The zero-shot results in the main paper use the default task formulations in lm-evaluation-harness (Biderman et al., 2024). The reported metric is length-normalized accuracy (`acc_norm`) for HellaSwag, PIQA, ARC-Easy, OpenBookQA, SocialIQA, and SciQ, and accuracy (`acc`) for WinoGrande and BoolQ. Because each checkpoint is evaluated once, the paper reports point estimates without variance or significance claims.

Table 12: Token perplexity across the 16 reported Paloma domain groups for the 1B models. Lower is better. DAR is best on 15 groups; Attention Residuals is best on 4chan.

Domain group	Baseline	AttnRes	DAR
4chan	631.49	**518.60**	536.45
C4 100 Domains	37.56	37.16	**35.98**
C4	44.04	43.75	**42.33**
Dolma v1.5	102.37	96.63	**93.23**
100 Programming Langs.	41,241.97	24,985.74	**23,731.25**
100 Subreddits	87.58	86.48	**84.12**
Falcon RefinedWeb	51.08	50.27	**48.79**
Gab	8,203.23	7,966.30	**7,435.71**
M2D2 S2ORC	52.05	49.84	**48.26**
M2D2 Wikipedia	47.63	46.86	**45.35**
Manosphere	94.71	93.53	**91.01**
mC4	76.08	73.76	**71.39**
PTB	53.38	50.92	**47.78**
RedPajama	590.75	508.56	**501.21**
Twitter AAE	8,396.23	8,083.35	**7,606.84**
Wikitext-103	28.06	26.86	**26.31**

E Additional Analyses

We report the router-balance and historical retrieval diagnostics referenced in the main paper.

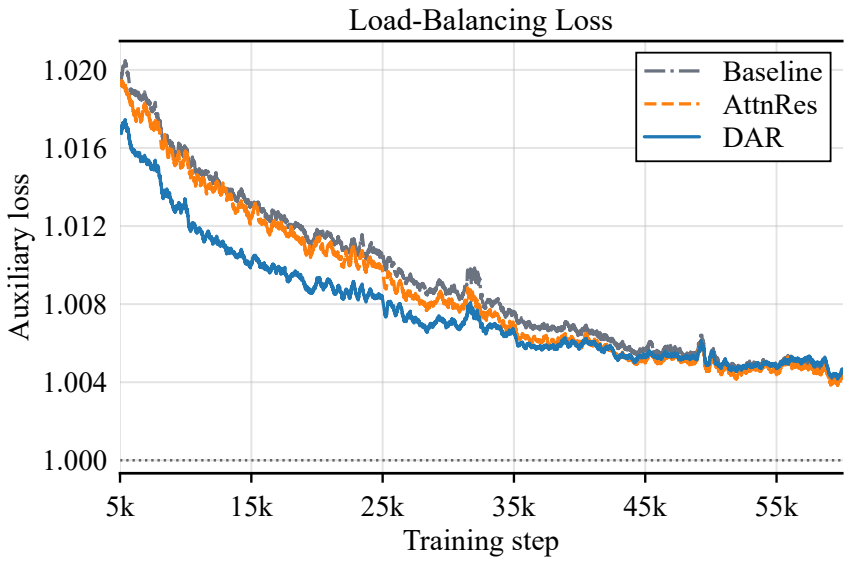
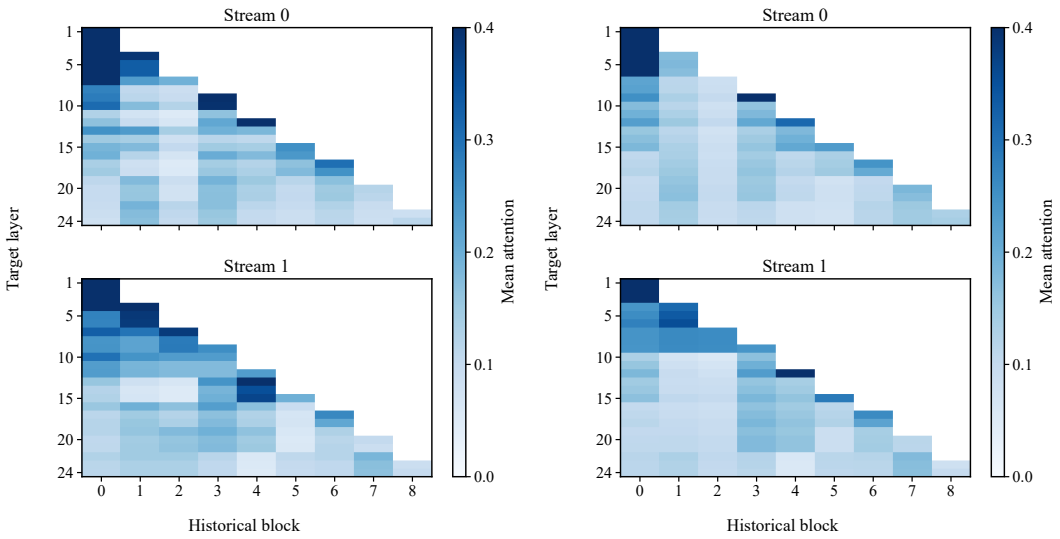


Figure 7: All three sparse-MoE models converge toward balanced aggregate expert utilization. The unscaled router auxiliary loss specified in Table 9 is shown from 5K to 60K steps using a 300-step moving average. Under the adopted normalization, the dotted reference at 1.0 denotes uniform aggregate utilization; the closely matched curves show that the residual mechanism does not materially alter router balance.

Gate asymmetry protocol. For Figure 9, we instrument the trained 0.5B DAR and DAR-SelfKV models on four FineWeb-Edu-10B validation sequences of length 4096 and average their MLP write gates over the final 128 token positions. At Transformer layer n, let β_n^j be the post-MLP branch-write gate for stream j and $R_{n,j}$ its rescue gain. We define the write and rescue gaps as

$$g_n = \mathbb{E}_{x,t}[\beta_n^1 - \beta_n^0], \qquad d_n = R_{n,1} - R_{n,0},$$

where the expectation averages over validation sequences and token positions. We omit the residual keep gate ρ because it enters the symmetric two-stream mixing matrix and does not by itself



(a) Pre-attention retrieval (b) Pre-MLP retrieval

Figure 8: Historical retrieval weights for 7B sparse-MoE DAR, averaged over eight 4096-token sequences. Rows are target layers and columns are historical blocks; blank cells are unavailable. Both branch types exhibit the same stream separation.

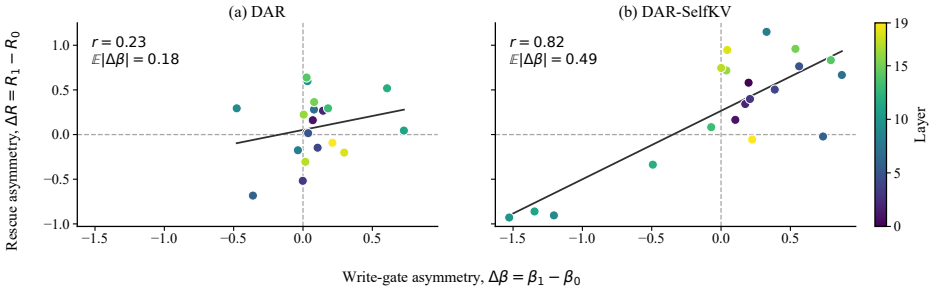


Figure 9: DAR-SelfKV exhibits stronger write-gate asymmetry and a stronger gate–rescue association than DAR. Each point is one MLP layer, colored by depth; lines are least-squares fits and both panels use identical axes.

favor either stream. DAR-SelfKV has a larger mean absolute write gap than DAR (0.49 versus 0.18), and its write and rescue gaps are more strongly correlated across layers ($r = 0.82$ versus 0.23). Thus, SelfKV's high cross-stream similarity masks imbalanced information flow: one stream receives stronger writes and becomes functionally dominant, whereas DAR uses both trajectories more evenly.