xHC: Expanded Hyper-Connections

Xiangdong Zhang[1,2,3] **Xiaohan Qin**[1,3] **Tuo Dai**[3] **Xiaoming Shi**[3]
Huaijin Wu[1,3] **Yebin Yang**[1,3] **Zhuo Xia**[1,3] **Shaofeng Zhang**[4]
Yu Wang[3] **Yu Cheng**[5] **Junchi Yan**[1*]
[1]School of AI, Shanghai Jiao Tong University
[2]Shanghai AI Laboratory [3]Dots Studio, Xiaohongshu Inc.
[4]University of Science and Technology of China
[5]Nanyang Technological University

Abstract

Hyper-Connections (HC) extend the single residual stream of Transformers into N parallel streams, improving language model pre-training with modest additional FLOPs. Manifold-Constrained HC (mHC) stabilizes this formulation at scale by constraining residual mixing to doubly stochastic matrices. Large gains from $N{=}1$ to $N{=}4$ suggest residual-stream expansion as a promising scaling axis, yet existing HC-family methods typically stop at $N{=}4$. Our experiments on mHC reveal why: directly scaling N beyond 4 yields rapidly diminishing returns, as gains saturate while training compute grows sharply. We trace this to two bottlenecks: fixed-dimensional layer outputs cannot supply enough write-back information for more streams; meanwhile, generating the residual-mixing matrix makes the dominant cost scale cubically with N. To address both bottlenecks, we propose **xHC** (**E**x**p**anded **H**yper-**C**onnections), the first HC-family method to achieve meaningful expansion beyond $N{=}4$. xHC enriches write-back signals with local contextual features along the token sequence and uses a sparse residual-stream architecture that updates only k out of N streams while preserving dense access to all streams. In language model pre-training, xHC with $N{=}16$ and $k{=}4$ substantially outperforms both mHC and the vanilla residual baseline. On a 28B MoE model, **xHC improves the average downstream score by 3.1 points over mHC** (vs. +2.7 for mHC over vanilla) with only 3.0% additional training FLOPs over the vanilla baseline. Scaling law experiments show that xHC matches the loss of a vanilla baseline trained with $1.50\times$ the compute, and its continued improvement with larger N makes expansion rate a practical scaling axis for HC-family models. Code is available at `https://github.com/aHapBean/xHC`.

1 Introduction

Large language model architectures have advanced rapidly [27, 50, 14, 28, 40, 52, 54, 48, 47] through improved attention mechanisms [43, 53, 31, 1], Mixture-of-Experts [10, 23], and scaling in width, depth, and data [24, 20]. Yet the residual stream that carries token representations across layers remains largely unchanged: it still operates as a single identity pathway, offering no learnable control over cross-layer information flow [17]. Hyper-Connections (HC) [55] challenge this design by replacing the single stream with N parallel residual streams governed by learnable mixing matrices. Building on HC, Manifold-Constrained HC (mHC) [49] makes multi-stream residual training

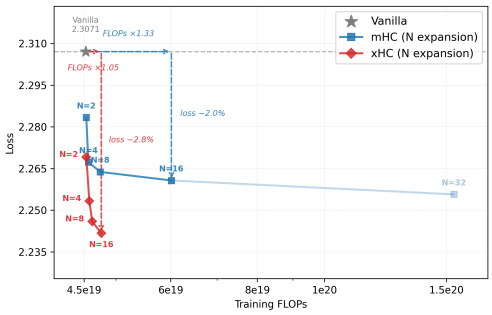
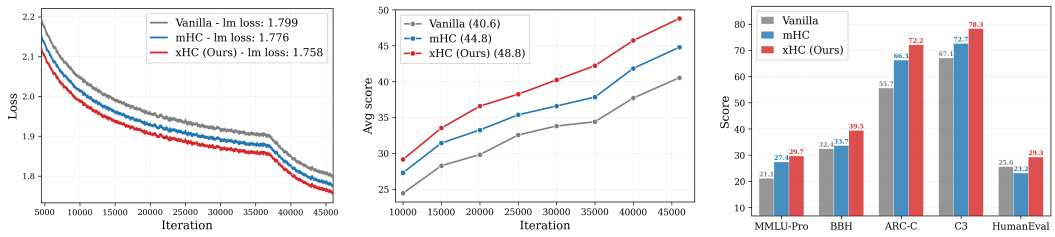


Figure 1: Expansion efficiency: loss vs. FLOPs using 2.5B MoEs (details in Table 3).

*Corresponding author: `yanjunchi@sjtu.edu.cn`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Training loss. (b) Average downstream score. (c) Benchmark-level gains.

Figure 2: **xHC delivers broad gains at 18B scale.** (a) xHC achieves lowest training loss. (b) The loss improvement translates into a significantly higher average downstream score across benchmarks in Table 1. (c) xHC improves representative benchmarks across reasoning, knowledge, and code.

stable at scale and represents the state-of-the-art variant to date. Residual-stream expansion increases the persistent state of models with only modest additional FLOPs, offering a form of *memory scaling* complementary to width and depth. The strong gains observed when expanding from $N{=}1$ to $N{=}4$ in prior results [55] suggest that this is a highly promising scaling axis. However, existing HC-family methods [55, 49, 51, 29] typically stop at $N{=}4$. We study this through mHC, the state-of-the-art HC formulation at scale. Our experiments in Figure 1 show that naively increasing N further yields rapidly diminishing returns: loss improves only marginally from $N{=}4$ to $N{=}16$, while training FLOPs increase by 32%, leaving this scaling axis largely unrealized.

We argue that this benefit-cost imbalance is not incidental, but stems from two distinct bottlenecks: one limits the benefit of larger N, while the other increases its cost. The first is an **information bottleneck**: each stream is meant to store a different weighted history of layer outputs, but each layer injects only a single write-back signal into the multi-stream state. As N grows, forming meaningful and diverse stream histories requires richer write-back information than this single signal can provide, making additional streams increasingly redundant. The second is a **computational bottleneck**: in mHC, the dominant cost comes from generating the residual mapping $\mathcal{H}_{\text{res}} \in \mathbb{R}^{N\times N}$ through an input-dependent projection. Because this projection predicts N^2 coefficients from an NC-dimensional residual state, its cost scales as $O(N^3C)$, making the cost of expansion grow much faster than its performance benefit. Together, these bottlenecks explain why simply adding more streams does not translate into proportionally more useful capacity. They also clarify what it would take to make expansion rate a true scaling axis: the model must supply more diverse write-back information while avoiding the cubic cost of scaling $\mathcal{H}_{\text{res}}$.

We therefore propose **xHC** (**E**x**p**anded **H**yper-**C**onnections), the first method to achieve *meaningful* expansion of HC-family models beyond $N{=}4$. Building on the stable multi-stream formulation of mHC, xHC is designed around a single goal: to make large-N expansion both effective and affordable, thereby turning expansion rate into a practical scaling axis. To increase the benefit of larger N, xHC introduces temporal feature augmentation along the causal token sequence: it enriches the write-back signal with low-cost features from neighboring tokens, using multi-scale causal convolutions followed by Gram–Schmidt orthogonalization [13] to supply more diverse information as the number of streams grows. To control the cost of larger N, xHC introduces a sparse residual-stream architecture that activates only k out of N streams for residual mixing and write-back, while keeping the read path dense so that every layer can still access the full N-stream state. These two designs are structurally decoupled yet highly synergistic: as N grows, temporal augmentation becomes more useful because additional streams require richer write-back information, while sparse residual updates become more valuable because they substantially reduce the extra FLOPs introduced by large-N expansion.

Empirically, xHC delivers substantial gains over both mHC and the vanilla residual baseline. As shown in Figure 2, xHC consistently improves both pre-training loss and downstream performance at the 18B MoE scale. It reaches a lower final training loss than mHC and the vanilla baseline (1.758 vs. 1.776 and 1.799), and improves the average downstream score from 44.8 with mHC to 48.8, while adding only 4.1% training FLOPs relative to the vanilla baseline.

More importantly, xHC changes the benefit-cost tradeoff of scaling the expansion rate itself. As shown in Figure 1, on a 2.5B MoE model, increasing N from 4 to 16 in mHC reduces loss by only 0.006 while increasing training FLOPs by 32%. In contrast, increasing N from 4 to 16 in xHC reduces loss by 0.012 with only 4% extra FLOPs. This shows that xHC makes larger N substantially more cost-effective, turning residual-stream expansion into a practical scaling axis.

This improved benefit-cost tradeoff also translates into better compute efficiency across model scales. Our scaling-law experiments in Figure 4 show that, to reach the same loss, the vanilla requires $1.50\times$ compute of xHC and mHC requires $1.19\times$ compute of xHC. This confirms that the improvement is systematic rather than confined to a specific model size. **Our contributions are as follows:**

1) We propose xHC, **the first HC-family method to achieve meaningful expansion beyond** $N{=}4$, turning expansion rate into a practical scaling axis.

2) We **identify two bottlenecks behind the diminishing returns** of larger N in mHC: limited write-back information supply and the cubic cost of dense residual mixing.

3) We **introduce two coordinated designs** for large-N expansion: temporal feature augmentation for richer write-back, and sparse residual updates for affordable residual mixing.

4) We **validate xHC on MoE models up to 28B parameters**, where it substantially outperforms mHC. Scaling laws and N-sweeps show that xHC improves compute efficiency and turns expansion rate into a viable scaling axis.

2 Related Work

Cross-Layer Information Flow. Residual connections [17] are the default mechanism for carrying token representations across layers in Transformer language models [43]. Most LLM architectures [28, 14, 50] keep this residual state as a single stream, while improving its optimization or weighting through gates, learned scaling, or depth-dependent rescaling [35, 42, 3, 44]. Other methods improve cross-layer access or feature reuse [21, 30, 41, 56], but operate over previous activations rather than expanding the persistent residual state. Closest to our work, Hyper-Connections (HC) [55] expand the residual stream into N learnable streams, and Manifold-Constrained HC (mHC) [49] stabilizes this formulation at scale via Sinkhorn-normalized residual mixing, achieving the strongest performance among HC-family methods. Other HC variants [51, 29] improve expressivity or memory efficiency, but existing HC-family methods largely operate at $N{=}4$, leaving open whether residual-stream expansion can become a true scaling axis for LLM pre-training. xHC bridges this gap by making larger-N expansion both meaningful and affordable.

Sparse Computation in Language Models. Sparsity is widely used to scale language models efficiently. Mixture-of-Experts (MoE) applies sparsity over computation: each token is routed to a subset of experts, decoupling total parameters from per-token FLOPs [33, 25, 10, 28, 40]. Sparse attention instead sparsifies over tokens, restricting each query to attend to a subset of the sequence and reducing long-context attention cost [6, 4, 53, 28]. xHC applies sparsity on a different dimension: *residual streams*. It activates only k out of N streams for residual mixing and write-back. This reduces the dominant residual-mixing cost from scaling with all N streams to scaling with only the k active streams, making large-N residual-memory expansion affordable.

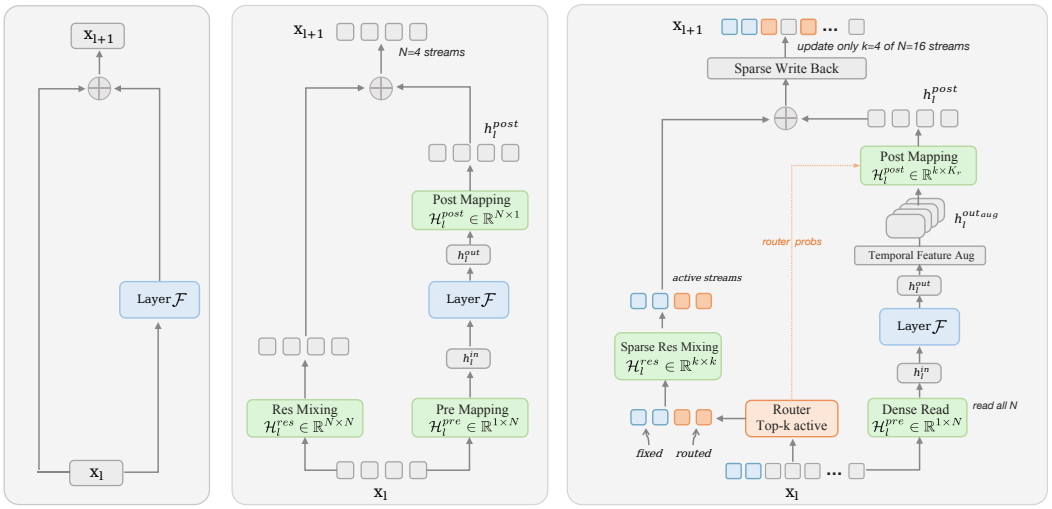
3 Method

We first recap the HC formulation and its manifold-constrained variant (§3.1), then analyze why scaling the expansion rate saturates in existing designs (§3.2). We then present xHC (§3.3; Figure 3), which uses two coordinated designs to make large-N expansion effective and affordable: temporal feature augmentation enriches the write-back signal (§3.3.1), while a sparse residual-stream architecture reduces the cost of large-N residual mixing (§3.3.2).

3.1 Preliminaries

Hyper-Connections. Standard residual connections [17] maintain a single residual stream across layers, limiting the capacity of cross-layer information flow. Hyper-Connections (HC) [55] generalize this design by maintaining N parallel residual streams, yielding significant pre-training improvements with modest additional FLOPs. Let $X_l = (x_{l,1}, \ldots, x_{l,N})^\top \in \mathbb{R}^{N\times C}$ denote the multi-stream state at layer l, where C is the hidden dimension. HC introduces three learnable mappings per layer: a *pre-mapping* $\mathcal{H}_l^{\mathrm{pre}} \in \mathbb{R}^{1\times N}$ that aggregates streams into a single input, a *post-mapping* $\mathcal{H}_l^{\mathrm{post}} \in \mathbb{R}^{N\times 1}$ that distributes the layer output back onto streams, and a *residual mapping* $\mathcal{H}_l^{\mathrm{res}} \in \mathbb{R}^{N\times N}$ that mixes streams. The single-layer update is:

$$X_{l+1} = \mathcal{H}_l^{\mathrm{res}}\, X_l \;+\; \mathcal{H}_l^{\mathrm{post}}\, \mathcal{F}(\mathcal{H}_l^{\mathrm{pre}}\, X_l,\; \mathcal{W}_l)\,, \tag{1}$$



(a) Residual Connection (b) Manifold-Constrained HC (mHC) (c) Expanded HC (xHC)

Figure 3: **Overview of xHC.** (a) A standard Transformer layer maintains a single residual stream. (b) mHC expands the residual state into N=4 streams with dense residual mixing and write-back. (c) xHC scales to N=16 with only k=4 active streams: it reads all streams, applies the sublayer $\mathcal{F}$ (Attn/MLP), augments MLP outputs, and sparsely writes back to selected streams. Blue/orange streams denote fixed/routed active streams. Restricting residual mapping to the k active streams reduces the dominant residual-mapping generation cost from $O(N^3C)$ to $O(k^3C)$.

where $\mathcal{F}(\cdot, \mathcal{W}_l)$ is the layer function (Attention or MLP) with parameters $\mathcal{W}_l$. In the dynamic variant, all three mappings are predicted from the current state via lightweight projections gated by $\tanh$.

Manifold-Constrained HC. When $\mathcal{H}_l^{\text{res}}$ is unconstrained, the composed mapping $\prod_l \mathcal{H}_l^{\text{res}}$ across depth can amplify or attenuate signals, breaking the identity mapping property that keeps deep residual training stable. mHC [49] addresses this by projecting $\mathcal{H}_l^{\text{res}}$ onto the Birkhoff polytope of doubly stochastic matrices via the Sinkhorn–Knopp algorithm:

$$\mathcal{H}_l^{\text{res}} = \text{SK}(\exp(\alpha_l^{\text{res}} \cdot (x_l' W_l^{\text{res}}) + b_l^{\text{res}})) \in \mathbb{R}^{N\times N}, \tag{2}$$

where $x_l' = \text{RMSNorm}(\text{flatten}(X_l)) \in \mathbb{R}^{NC}$ is the RMSNorm-normalized flattened multi-stream state, $W_l^{\text{res}} \in \mathbb{R}^{NC\times N^2}$ predicts the residual-mixing logits, α_l^{res} and b_l^{res} are learnable scale and bias, and $\text{SK}(\cdot)$ denotes Sinkhorn normalization [34] that enforces row and column sums to one. This constraint preserves stable information propagation across depth, enabling practical large-scale deployment at N=4 with significant performance gains.

3.2 Why Scaling N Saturates in mHC

Understanding this saturation is essential for turning residual-stream expansion from a small-N improvement into a practical scaling axis. Prior results [55] show that early residual-stream expansion is highly effective: increasing N from 1 to 4 brings substantial gains at modest FLOPs cost (less than 2%). This suggests that scaling N could improve model performance by increasing residual memory, providing an axis orthogonal to width and depth. However, existing HC-family methods [55, 51, 29, 49] typically stop at N=4, leaving open whether this axis remains meaningful at larger N. To answer this, we sweep $N \in \{2, 4, 8, 16, 32\}$ in mHC under matched training recipes. As shown in Figure 1, scaling beyond N=4 quickly runs into diminishing returns: increasing N from 4 to 16 reduces loss by only 0.006, while training FLOPs increase by 32%. This unfavorable benefit-cost tradeoff motivates us to examine what prevents larger N from translating into useful capacity. We identify two bottlenecks behind this saturation.

Information Supply Bottleneck. In mHC, each of the N streams accumulates a distinct weighted combination of historical layer outputs. However, the write-back to stream i at layer l takes the form:

$$\Delta x_{l,i} = h_{l,i}^{\text{post}} \cdot \text{out}, \tag{3}$$

where $h_{l,i}^{\text{post}}$ is a scalar and $\text{out} \in \mathbb{R}^C$ is the layer output. Although $h_{l,i}^{\text{post}}$ is input-dependent and stream-specific, the newly injected information at each layer is spanned by only one write-back

component, out: different streams can assign different weights to this component, but cannot draw from different components. This can be sufficient at small N, but as N grows, additional streams need more diverse write-back components to form non-redundant histories. Without such diversity, extra streams become increasingly redundant. Section 4.5 provides evidence for this bottleneck.

Cost Bottleneck. The second bottleneck is on the cost side. In mHC, generating $\mathcal{H}_l^{\mathrm{res}} \in \mathbb{R}^{N\times N}$ from the NC-dimensional state requires predicting N^2 coefficients, leading to an $O(N^3C)$ input-dependent projection cost. This makes large-N expansion increasingly expensive even when the additional performance gain is marginal. Concretely, mHC at N=16 adds roughly 32% extra FLOPs relative to its N=4 setting for only limited loss reduction on 2.5B MoE, as shown in Figure 1.

The two bottlenecks compound: the information bottleneck limits the *benefit* of larger N, while the cost bottleneck increases its *price*. Together they collapse the return on investment of scaling N.

3.3 xHC: Expanded Hyper-Connections

xHC addresses both bottlenecks to make expansion rate N a practical scaling axis (Figure 3). It enriches write-back signals with local contextual features and adopts an asymmetric sparse residual-stream architecture: only k active streams undergo residual mixing and write-back, while dense read keeps every layer connected to all N streams. This reduces the dominant $\mathcal{H}_l^{\mathrm{res}}$ generation cost from $O(N^3C)$ to $O(k^3C)$, with k=4 and $N = 16$ in our main setting. Other implementation details follow mHC unless stated otherwise.

3.3.1 Enriching the Write-Back Signal

Temporal Feature Augmentation. As analyzed in §3.2, large-N expansion saturates because each layer provides only a single write-back component to all streams. A direct fix would be to compute a separate output for each stream, but this would multiply the layer FLOPs by N. Instead, xHC enriches the write-back basis by borrowing low-cost local contextual information from neighboring tokens. Because autoregressive prediction already conditions on surrounding context, neighboring hidden outputs are semantically compatible with the current token and can be integrated into its write-back signal. We aggregate this local information with lightweight causal depthwise 1D convolutions, which preserve autoregressive order and add small overhead [16, 46, 15].

To describe local convolution, we temporarily restore the sequence dimension and write $\mathrm{out} \in \mathbb{R}^{S\times C}$, where S is the sequence length. We apply r causal depthwise convolutions with kernel sizes $\{\kappa_1, \ldots, \kappa_r\}$ to the layer output. Different kernel sizes capture neighboring-token information at different contextual ranges, providing multi-granularity write-back components that expand the write-back basis beyond a single layer output. We concatenate these components with the original output:

$$\mathrm{out}_{\mathrm{aug}} = [\,\mathrm{out};\ \mathrm{DWConv}_{\kappa_1}(\mathrm{out});\ \ldots;\ \mathrm{DWConv}_{\kappa_r}(\mathrm{out})\,] \in \mathbb{R}^{S\times K_r\times C}, \tag{4}$$

where $K_r = r + 1$. In our main setting, $r = 3$ with kernel sizes $\{4, 8, 12\}$, so $K_r = 4$. The convolutions are per-channel and causal, adding only $C\sum_{j=1}^{r}\kappa_j$ parameters per layer. We apply temporal augmentation only after MLP (MoE FFN) layers: attention already mixes positions, and adding temporal augmentation after attention empirically destabilizes training.

Gram–Schmidt Orthogonalization. Because depthwise convolutions operate channel-wise, their outputs can retain a strong component aligned with the original layer output. If these correlated components are directly combined by $\mathcal{H}_l^{\mathrm{post}}$, the augmented write-back signal may contain redundant components and amplify the original direction in an uncontrolled manner, making write-back scaling less predictable at large N. We therefore apply modified Gram–Schmidt orthogonalization [13] over the K_r components. Let $g_j = \mathrm{DWConv}_{\kappa_j}(\mathrm{out})$ denote the j-th convolutional component, and initialize $v_1 = \mathrm{out}$. For $j = 1, \ldots, r$, we compute:

$$v_{j+1} = g_j - \sum_{i=1}^{j}\frac{\langle g_j, v_i\rangle}{\langle v_i, v_i\rangle}v_i. \tag{5}$$

After orthogonalization, we redefine $\mathrm{out}_{\mathrm{aug}} = [v_1; \ldots; v_{K_r}] \in \mathbb{R}^{S\times K_r\times C}$ and use these orthogonalized components for all subsequent write-back operations. The projection is computed per token along the C-dimensional space and adds small overhead.

3.3.2 Sparse Residual-Stream Architecture

The cost bottleneck comes from residual mixing over all N streams. xHC reduces this cost by updating only k active streams out of N, while keeping dense read access to the full N-stream state. We describe the forward pass in three steps: routing, reading, and writing.

Stream Routing. A router selects the k streams to update at each layer. Let $\tilde{x}_l = \mathrm{LN}(\mathrm{flatten}(X_l)) \in \mathbb{R}^{NC}$ denote the LayerNorm-normalized [2] flattened multi-stream state used by the router. The router observes the full N-stream state and produces per-stream importance scores:

$$s = \sigma(\tilde{x}_l W_r) \in \mathbb{R}^N, \tag{6}$$

where $W_r \in \mathbb{R}^{NC \times N}$ is a learned projection and σ is the sigmoid function. We use sigmoid rather than softmax to reduce winner-take-all routing and avoid inactive streams. For stability, we use a fixed-plus-routed scheme: the first m streams are always active with routing weight 1, and the remaining $k - m$ active streams are selected by TopK routing over the non-fixed streams [25, 33]. Let $\mathcal{I} = (\mathcal{I}_1, \ldots, \mathcal{I}_k)$ list the selected stream indices:

$$\{\mathcal{I}_1, \ldots, \mathcal{I}_k\} = \{1, \ldots, m\} \cup \mathrm{TopK}_{j>m}(s_j, k-m), \qquad p_i = \begin{cases} 1, & \mathcal{I}_i \le m, \\ s_{\mathcal{I}_i}, & \mathcal{I}_i > m. \end{cases} \tag{7}$$

Here, TopK returns stream indices, and p_i is the routing weight for the i-th selected stream, $i = 1, \ldots, k$. The rows of $X_{\mathrm{active}} = X_l[\mathcal{I}, :]$ follow the same order.

Dense Read. A naive sparse variant would sparsify both reading and writing, but this can disconnect cross-layer information flow: streams updated at one layer may not be selected for reading by the next, shortening the effective path through which routed information propagates. xHC therefore keeps the read path dense:

$$\mathrm{input}_l = \sum_{i=1}^{N} h_{l,i}^{\mathrm{pre}} \cdot x_{l,i}, \quad \text{where } \mathcal{H}_l^{\mathrm{pre}} = f_{\mathrm{pre}}(X_l) \in \mathbb{R}^{1 \times N}. \tag{8}$$

Here, $h_{l,i}^{\mathrm{pre}}$ denotes the i-th entry of $\mathcal{H}_l^{\mathrm{pre}}$. Thus every layer can access the full N-stream state, preserving cross-layer information flow even though only k streams are updated. Ablations in Section 4.5 confirm the importance of dense read.

Sparse Residual Update. In mHC, $\mathcal{H}_l^{\mathrm{res}}$ and $\mathcal{H}_l^{\mathrm{post}}$ are generated from the full N-stream state. xHC instead applies the same generators only to the active state X_{active}:

$$\mathcal{H}_l^{\mathrm{res}} = \mathrm{SK}(f_{\mathrm{res}}(X_{\mathrm{active}})) \in \mathbb{R}^{k \times k}, \tag{9}$$

$$\mathcal{H}_l^{\mathrm{post}} = f_{\mathrm{post}}(X_{\mathrm{active}}) \in \mathbb{R}^{k \times K_r}. \tag{10}$$

Here, f_{res} and f_{post} follow the residual and post-mapping generators of mHC, but operate on the active streams. The post-mapping $\mathcal{H}_l^{\mathrm{post}}$ generalizes the mHC form from $\mathbb{R}^{N \times 1}$ to $\mathbb{R}^{k \times K_r}$, so that each active stream can independently combine the K_r augmented write-back components. This reduces the dominant residual-mapping generation cost from $O(N^3 C)$ to $O(k^3 C)$ while retaining the Sinkhorn constraint on the active stream subset.

The active write-back signal is computed by contracting $\mathcal{H}_l^{\mathrm{post}}$ with the augmented components:

$$\Delta X_{\mathrm{active},i} = p_i \sum_{j=1}^{K_r} \mathcal{H}_{l,i,j}^{\mathrm{post}} \, \mathrm{out}_{\mathrm{aug},j}, \quad i = 1, \ldots, k, \tag{11}$$

where the token dimension is omitted for clarity. The active streams are then updated by:

$$X_{\mathrm{active}}^{\mathrm{new}} = \mathcal{H}_l^{\mathrm{res}} X_{\mathrm{active}} + \Delta X_{\mathrm{active}}. \tag{12}$$

Here p_i is the routing weight for the i-th active stream: fixed streams use $p_i = 1$, while routed streams use their sigmoid scores. We apply p_i only to the new write-back term, leaving residual mixing ungated. Updated streams are scattered back to the full state, while non-active streams carry forward unchanged for future dense reads.

Table 1: Downstream evaluation on 18B and 28B MoE models. All numbers are scores (%; higher is better). mHC uses N=4, and xHC uses N=16, k=4; all methods use comparable training FLOPs.

	18B			28B		
Benchmark	Vanilla	mHC	xHC	Vanilla	mHC	xHC
Language Understanding & Knowledge						
MMLU	48.9	54.7	**57.2**	54.6	56.8	**60.5**
MMLU-Pro	21.1	27.4	**29.7**	30.1	34.9	**36.0**
MMLU-Redux	46.4	49.9	**52.8**	50.6	53.9	**56.4**
Reasoning & Mathematics						
BBH	32.4	33.7	**39.5**	41.7	**43.6**	43.4
CommonsenseQA	54.6	56.6	**60.9**	60.5	63.9	**69.6**
ARC-Challenge	55.7	66.3	**72.2**	70.8	74.9	**77.7**
GSM8K	37.7	44.5	**48.4**	50.3	56.3	**59.2**
Code						
HumanEval	25.6	23.2	**29.3**	27.4	26.8	**31.1**
LCBench	9.9	12.2	**14.6**	15.1	14.8	**17.9**
Chinese						
CMMLU	42.7	47.6	**50.4**	47.6	50.1	**53.4**
CEval	44.5	48.8	**52.4**	50.2	51.2	**54.9**
C3	67.1	72.7	**78.3**	75.2	78.7	**82.5**
Average	40.6	44.8	**48.8**	47.8	50.5	**53.6**

3.3.3 Why Both Designs Are Needed

The two designs address complementary bottlenecks. Temporal feature augmentation makes additional streams more informative, but alone leaves dense residual mixing expensive. Sparse residual updates make large N affordable, but alone leave the write-back signal information-limited. Together, they make large-N expansion both meaningful and affordable, with ablations in Section 4.5 confirming that neither design alone recovers the full benefit of xHC.

4 Experiments

We evaluate xHC through language model pre-training experiments across model scales, compute budgets, expansion rates, and ablation settings. The evaluation first compares xHC against mHC and the vanilla residual baseline on 18B and 28B MoE models, then studies compute efficiency through scaling-law experiments. Finally, N-sweep and ablation studies examine whether larger expansion rates remain beneficial under xHC and whether the two designs address the large-N bottlenecks.

4.1 Experimental Setup

Training setup. We conduct experiments in the language model pre-training setting using Mixture-of-Experts (MoE) Transformer models. Unless otherwise stated, we apply xHC and mHC to the same DeepSeekMoE-style [10] Transformer backbone. Specifically, the backbone uses grouped-query attention (GQA) [1] and 144 experts with top-8 routing, where each expert is implemented as a SwiGLU-based FFN [32]. We report main downstream results at two MoE scales: an 18B-total, 1.7B-activated model and a 28B-total, 2.7B-activated model. We additionally use a 10B MoE model for ablations and a 2.5B MoE model for N-sweep experiments. Unless otherwise specified, xHC uses N=16 total streams with k=4 active streams. All models are trained with a context length of 8192 tokens on a data mixture covering English, Chinese, code, mathematics, and reasoning corpora. Full hyperparameters and training details are provided in Appendix A.

Baselines. We compare xHC primarily against mHC (N=4) [49], which has comparable training FLOPs to xHC and serves as the strongest HC-family method at scale. We also include a standard residual-stream Transformer [43, 17] as the vanilla baseline.

Downstream Evaluation. We evaluate on a benchmark suite covering four aspects of LLM performance: language understanding and knowledge (MMLU [18], MMLU-Pro [45], MMLU-Redux [12]); reasoning and mathematics (BBH [38], CommonsenseQA [39], ARC-Challenge [7], GSM8K [8]); code generation and understanding (HumanEval [5], LCBench [9]); and Chinese language under-

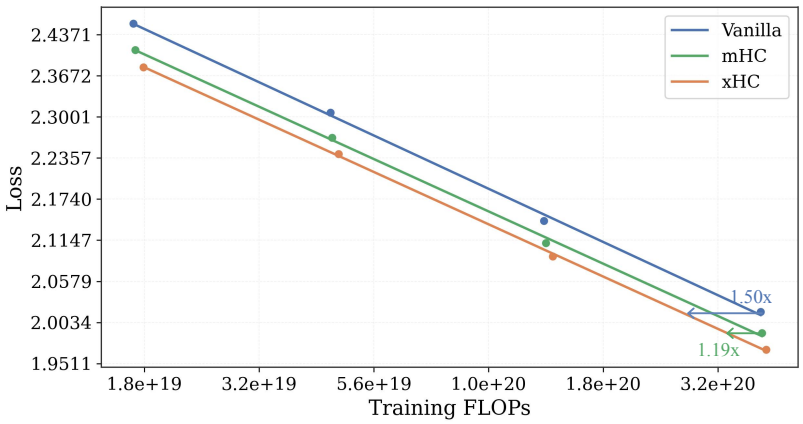


Figure 4: Scaling-law comparison. xHC traces a consistently lower loss curve than both mHC and the vanilla residual baseline across training compute.

standing (CMMLU [26], CEval [22], C3 [37]). Together, these benchmarks provide a compact yet diverse view of whether improved residual-stream scaling translates into stronger generalization after pre-training. Evaluation protocols and shot settings are provided in Appendix A.

4.2 Main Results

We compare xHC against mHC and the vanilla residual baseline on 18B and 28B MoE models. Table 1 summarizes downstream results under comparable training-FLOPs budgets, with xHC adding only 3.0% training FLOPs over the vanilla baseline at 28B.

xHC outperforms both mHC and the vanilla baseline across both scales. At 18B, the average score improves from 44.8 with mHC to 48.8 with xHC, with representative gains on ARC-Challenge (+5.9), BBH (+5.8), C3 (+5.6), and HumanEval (+6.1). At 28B, the average score improves from 50.5 to 53.6 (+3.1) over mHC, with representative gains on CommonsenseQA (+5.7), HumanEval (+4.3), C3 (+3.8), and MMLU (+3.7). The consistent improvements on the 28B MoE model further demonstrate that xHC remains effective in larger pre-training regimes.

4.3 Scaling Laws

The downstream results above establish the advantage of xHC at two discrete scales. We next ask whether this advantage persists across a broader compute range. To fit scaling laws [24], we train a separate suite of four models for each method under matched recipes, spanning training compute from 1.7×10^{19} to 4.0×10^{20} FLOPs. We fit a shifted power law [20, 19]:

$$\mathcal{L}(C) = AC^{-\alpha} + E, \tag{13}$$

where C denotes training FLOPs, and A and α are fitted parameters, and $E = 0.72$ is the estimated irreducible loss. We fit by linear regression in log space and plot the resulting curves with the offset E included. Detailed settings are provided in Appendix B.

Figure 4 shows that xHC traces a lower fitted loss curve than both mHC and the vanilla baseline across the measured compute range. At the largest compute point, xHC reduces loss by about 1.1% relative to mHC and 2.4% relative to the vanilla baseline. To quantify this advantage, we use the losses achieved by the largest vanilla baseline and mHC models as representative targets, and read from the fitted xHC curve the compute required to match each target. The resulting matched-loss comparison shows that the baseline and mHC require about $1.50\times$ and $1.19\times$ the compute of xHC, respectively. This comparison, together with the consistently lower fitted curve, indicates that xHC provides a systematic compute-efficiency improvement across the measured model family, rather than only improving specific downstream evaluation points.

4.4 Scaling the Expansion Rate

We evaluate whether xHC turns the expansion rate N into a meaningful scaling axis. We sweep $N \in \{2, 4, 8, 16\}$ in xHC on a 2.5B MoE model (details in Appendix A), and compare against mHC at matched expansion rates. In mHC, all N streams participate in dense residual mixing, so the

Table 2: Ablation study. Parentheses show extra training FLOPs over vanilla baseline. "Sparse" is the sparse architecture, "D. Read" is Dense Read, and "Fixed" is the number of always-active streams.

Method	N	Temp Aug.	Sparse	D. Read	k	Fixed	Router	Val. Loss ↓
(1) mHC (+0.6%)	4	-	-	-	-	-	-	2.004
(2) mHC (+18.8%)	16	-	-	-	-	-	-	1.998
(3) mHC w/ Temp Aug (+20.1%)	16	✓	-	-	-	-	-	1.984
(4) **xHC** (+3.3%)	16	✓	✓	✓	4	2	Sigmoid	1.983
(5) w/o D.Read & Fixed Streams	16	✓	✓	✗	4	0	Sigmoid	1.997
(6) w/o Dense Read	16	✓	✓	✗	4	2	Sigmoid	1.985
(7) w/o Fixed Streams	16	✓	✓	✓	4	0	Sigmoid	1.986
(8) k=2	16	✓	✓	✓	2	1	Sigmoid	1.991
(9) k=8	16	✓	✓	✓	8	2	Sigmoid	1.982
(10) Softmax Router	16	✓	✓	✓	4	2	Softmax	1.988

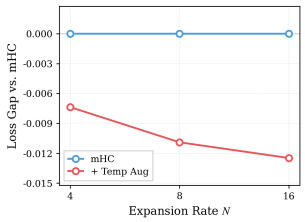


Figure 5: Information bottleneck ablation. The gain grows with N.

residual-mixing overhead grows rapidly with N. In xHC, increasing N enlarges residual-memory capacity while the sparse update cost remains controlled by k. Figure 1 plots language modeling loss during training against training FLOPs as N increases. Under xHC, loss decreases consistently from N=2 to N=16, with each doubling of N yielding a clear improvement at small additional FLOPs. In contrast, mHC shows rapid saturation: increasing N from 4 to 16 reduces loss by only 0.006 while increasing training FLOPs by 32%. Over the same range, xHC reduces loss by 0.012 with only 4% additional FLOPs. These results show that xHC substantially improves the benefit-cost tradeoff of residual-stream expansion, making larger N useful under a practical compute budget.

4.5 Ablation Study

Ablation studies use a 10B MoE model trained under a matched data budget, providing a controlled setting for isolating the effect of each xHC design choice. Model configurations and training details are provided in Appendix A. We report validation loss on the Pile test set [11].

Incremental Construction. Following the motivation of xHC, we start from mHC at N=16 and incrementally add the two ingredients of xHC: temporal feature augmentation and a sparse residual-stream architecture. Temporal feature augmentation targets the information bottleneck by enriching write-back information, while sparse residual updates target the cost bottleneck by making large-N affordable. As shown in Table 2 (1)–(4), adding temporal feature augmentation to mHC improves validation loss from 1.998 to 1.984, confirming the benefit of enriching the write-back signal at large N. Adding the sparse residual-stream architecture then preserves this validation loss (1.983) while reducing FLOPs overhead from 20.1% to 3.3%, showing that sparse updates make the enriched large-N residual state affordable. Together, these results support the design of xHC: effective large-N expansion requires both richer write-back information and controlled residual-update cost.

Information Bottleneck Ablations. We add temporal feature augmentation alone to dense mHC at $N \in \{4, 8, 16\}$, using three multi-scale causal convolution branches by default. Figure 5 shows that the loss gap relative to mHC becomes increasingly negative as N grows. This trend is consistent with our diagnosis that the information bottleneck becomes more severe at larger expansion rates, where additional streams require richer write-back information to form non-redundant histories.

Sparse Architecture Ablations. Table 2 validates the designs that keep sparse residual updates effective. The central risk of sparse write is information disconnection: a stream updated at one layer may not be selected at the next, breaking cross-layer propagation. Removing both Dense Read and fixed streams exposes this risk, degrading loss to 1.997. Dense Read mitigates this by letting every layer access all N streams, so information in any stream remains visible regardless of routing; with fixed streams present, removing Dense Read increases loss from 1.983 to 1.985. Fixed streams further stabilize sparse updates by providing guaranteed write targets; removing them increases loss from 1.983 to 1.986. Separately, the active-stream budget k controls the sparsity–quality tradeoff: k=2 under-provisions active streams (1.991), k=8 yields a marginal gain (1.982) at higher cost, and k=4 balances the two. Finally, Sigmoid routing outperforms Softmax (1.983 vs. 1.988), consistent with avoiding winner-take-all routing that can leave streams persistently inactive.

5 Conclusion

We presented xHC, a method for making residual-stream expansion in HC-family models effective and affordable beyond the common N=4 setting. We showed that directly scaling N in mHC is limited by two bottlenecks: insufficient write-back information and the cubic cost of dense residual-mapping

generation. xHC addresses these bottlenecks with temporal feature augmentation and a sparse residual-stream architecture that preserves dense read access while updating only k active streams. Across MoE pre-training experiments, xHC scales to $N{=}16$ with $k{=}4$, substantially outperforms mHC and the vanilla residual baseline on models up to 28B parameters, and improves matched-loss compute efficiency in scaling-law experiments. These results show that xHC turns expansion rate from a small fixed setting into a practical scaling axis for HC-family models.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Scientific Research Innovation Capability Support Project for Young Faculty (U40) of the Ministry of Education of China (SRICSPYF-ZY2025019) and Xiaohongshu Inc.

References

[1] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. 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*, pages 4895–4901, 2023.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] T. Bachlechner, B. P. Majumder, H. Mao, G. Cottrell, and J. McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in artificial intelligence*, pages 1352–1361. PMLR, 2021.

[4] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[5] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[6] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[7] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[8] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

[10] D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. 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)*, pages 1280–1297, 2024.

[11] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[12] A. P. Gema, J. O. J. Leang, G. Hong, A. Devoto, A. C. M. Mancino, R. Saxena, X. He, Y. Zhao, X. Du, M. R. G. 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)*, pages 5069–5096, 2025.

[13] G. H. Golub and C. F. Van Loan. *Matrix computations*. JHU press, 2013.

[14] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[15] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[16] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[19] T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[20] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[22] Y. Huang, Y. Bai, Z. Zhu, J. Zhang, J. Zhang, T. Su, J. Liu, C. Lv, Y. Zhang, Y. 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.

[23] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[24] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[25] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[26] H. Li, Y. Zhang, F. Koto, Y. Yang, H. Zhao, Y. Gong, N. Duan, and T. Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11260–11285, 2024.

[27] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[28] A. Liu, A. Mei, B. Lin, B. Xue, B. Wang, B. Xu, B. Wu, B. Zhang, C. Lin, C. Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

[29] Z. Liu, H. Zhang, and A. Li. Beyond the birkhoff polytope: Spectral-sphere-constrained hyper-connections. *arXiv preprint arXiv:2603.20896*, 2026.

[30] M. Pagliardini, A. Mohtashami, F. Fleuret, and M. Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

[31] N. Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[32] N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[33] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[34] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[35] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[36] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[37] K. Sun, D. Yu, D. Yu, and C. Cardie. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155, 2020.

[38] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. Le, E. Chi, D. 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*, pages 13003–13051, 2023.

[39] A. Talmor, J. Herzig, N. Lourie, and J. Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, 2019.

[40] K. Team, Y. Bai, Y. Bao, Y. Charles, C. Chen, G. Chen, H. Chen, H. Chen, J. Chen, N. Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[41] K. Team, G. Chen, Y. Zhang, J. Su, W. Xu, S. Pan, Y. Wang, Y. Wang, G. Chen, B. Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

[42] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 32–42, 2021.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[44] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

[45] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. 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.

[46] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

[47] H. Wu, B. Li, Y. Yang, Y. Tu, Z. Zhou, J. Chen, and J. Yan. Achieving low-bit muon through subspace preservation and grid quantization. In *The Fourteenth International Conference on Learning Representations*, 2026.

[48] H. Wu, L. Li, H. Huang, Y. Tu, J. Zhang, M. Yu, and J. Yan. Hshare: Fast llm decoding by hierarchical key-value sharing. In *International Conference on Learning Representations*, volume 2025, pages 89412–89428, 2025.

[49] Z. Xie, Y. Wei, H. Cao, C. Zhao, C. Deng, J. Li, D. Dai, H. Gao, J. Chang, K. Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

[50] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

[52] Y. Yang, H. Wu, F. Guo, L. Yao, X. Qin, J. Wang, D. Zhang, and J. Yan. Jtok: On token embedding as another axis of scaling law via joint token self-modulation. *arXiv preprint arXiv:2602.00800*, 2026.

[53] J. Yuan, H. Gao, D. Dai, J. Luo, L. Zhao, Z. Zhang, Z. Xie, Y. Wei, L. Wang, Z. Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097, 2025.

[54] X. Zhang, D. Zhang, S. Zhang, X. Qin, Y. Cheng, and J. Yan. Nitp: Next implicit token prediction for llm pre-training. *arXiv preprint arXiv:2605.24956*, 2026.

[55] D. Zhu, H. Huang, Z. Huang, Y. Zeng, Y. Mao, B. Wu, Q. Min, and X. Zhou. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2024.

[56] L. Zhu, Y. Fang, B. Liao, S. Wang, T. Cheng, Z. Huang, C. Chen, L. Wei, Y. Zeng, Y. Wang, et al. Mixture-of-depths attention. *arXiv preprint arXiv:2603.15619*, 2026.

Table 3: Model configurations and training hyperparameters for the main model scales. The 2.5B setting is used for N-sweep experiments, the 10B setting for ablations, and the 18B/28B settings for main results.

Attribute	**2.5B**	**10B**	**18B**	**28B**
Total Parameters	2.5B	10B	18B	28B
Active Parameters	0.5B	1.4B	1.7B	2.7B
Layers	15	15	28	32
Leading Dense Layers	1	1	1	1
MoE Layers	14	14	27	31
Hidden Dimension	1024	2080	2112	2560
FFN Dimension (dense layer)	3104	6144	6144	6144
Attention Heads	8	16	16	20
KV Groups (GQA)	4	8	8	10
Head Dimension	128	128	128	128
Routed Experts	144	144	144	144
Shared Experts	1	1	1	1
Active Experts (Top-k)	8	8	8	8
Expert FFN Dimension	320	704	672	768
Shared Expert Dimension	320	704	672	768
Normalization	RMSNorm ($\epsilon = 10^{-5}$)			
Position Embedding	RoPE ($\theta = 50000$)			
Activation	SwiGLU			
xHC Expansion Rate N	16	16	16	16
xHC Active Streams k	4	4	4	4
xHC Fixed Streams m	2	2	2	2
Temporal Conv Branches r	3	3	3	3
Conv Kernel Sizes	{4, 8, 12}			
Gram–Schmidt Orthogonalization	✓			
Temporal Augmentation Position	After MLP (MoE FFN) only			
Stream Router Activation	Sigmoid			
Sinkhorn Iterations	20			
Gating Factor Init α	0.01			
Sequence Length	8192			
Vocab Size	152064			
Tokenizer	Qwen2			
Global Batch Size	160	288	–	768
Training Tokens	–	–	–	–
Warmup Steps	500	500	500	500
Optimizer	AdamW (β_1=0.9, β_2=0.95, ϵ=10^{-15})			
Base Learning Rate	6.95e-4	4.82e-4	3.97e-4	3.5e-4
LR Schedule	WSD (warmup–stable–decay, exponential decay)			
Min LR Ratio	0.1			
Weight Decay	0.1			
Gradient Clipping	1.0			

A Experimental Details and Hyperparameters

Model Architectures. All MoE backbones follow a DeepSeekMoE-style design: one leading dense layer followed by a stack of MoE layers. Each MoE layer contains routed experts and shared experts with top-8 sigmoid routing. We use grouped-query attention (GQA) with query-key LayerNorm (QK Norm). Detailed configurations are listed in Table 3.

xHC Configuration. Unless otherwise specified, xHC uses N=16 total expansion streams with k=4 active streams per layer, of which m=2 are always-active fixed streams and $k-m$=2 are dynamically selected by the stream router. The router is a single linear projection applied to the LayerNorm-normalized, flattened $N\times C$ residual state, producing N stream scores through a sigmoid activation. Temporal feature augmentation uses r=3 causal depthwise convolution branches with kernel sizes $\{4, 8, 12\}$, applied only after MLP (MoE FFN) layers, followed by Gram–Schmidt orthogonalization. Sinkhorn normalization for $\mathcal{H}_l^{\text{res}}$ uses 20 iterations, and the gating factor α is initialized to 0.01. During xHC training, we observe that rare extreme activations can hinder the convergence of Sinkhorn normalization, leaving some row sums greater than one and amplifying

the forward signal. To improve stability, we apply row-sum clamping after Sinkhorn normalization by replacing each row $\mathcal{H}^{\text{res}}_{l,i:}$ with $\mathcal{H}^{\text{res}}_{l,i:} / \max(\sum_j \mathcal{H}^{\text{res}}_{l,ij}, 1)$. This operation only rescales rows whose sums exceed one, and we observe that it stabilizes training without degrading the performance.

N-sweep Configuration. The N-sweep experiments in Section 4.4 use the 2.5B setting in Table 3. Table 4 lists the xHC active-stream configuration for each expansion rate. For mHC, we sweep dense expansion rates $N \in \{2, 4, 8, 16\}$, where all N streams participate in residual mixing and write-back. Other training settings, including tokenizer, context length, data mixture, optimizer, learning-rate schedule, and training-token count, are kept matched across compared runs.

Table 4: N-sweep configuration for xHC.

Expansion rate N	Active streams k	Fixed streams m
2	1	0
4	2	1
8	4	2
16	4	2

Training Hyperparameters. At each model scale, xHC, mHC, and the vanilla residual baseline are trained with matched optimization recipes and data budgets, so that differences can be attributed to the residual-stream design rather than the training setup. We use AdamW with β_1=0.9, β_2=0.95, ϵ=10^{-15}, weight decay 0.1, and gradient clipping at 1.0. The learning rate follows a warmup–stable–decay (WSD) schedule with 500 warmup steps, exponential decay, and a minimum-to-peak learning-rate ratio of 0.1. The base learning rate and global batch size are scaled with model size, as listed in Table 3. Training runs use up to 128 GPUs.

Evaluation Protocol. All benchmarks are evaluated under the OpenCompass framework. The few-shot settings are: MMLU/MMLU-Pro/MMLU-Redux/CMMLU/CEval (5-shot), ARC-Challenge (25-shot), CommonsenseQA (7-shot), LCBench (5-shot), GSM8K (4-shot), BBH/C3 (3-shot), and HumanEval (0-shot).

B Scaling Law Experiment Configuration

We train four model sizes for each method (vanilla baseline, mHC, and xHC) under matched recipes, spanning training compute from approximately 1.7×10^{19} to 4.0×10^{20} FLOPs. Table 5 lists the key model configurations used for scaling-law fitting. All models follow the same DeepSeekMoE-style template as Table 3, with 144 routed experts, top-8 routing, one shared expert, and one leading dense layer, while varying model width, depth, and expert FFN size. Scaling laws are fitted on the final language modeling (LM) loss.

We fit a shifted power law $\mathcal{L}(C) = AC^{-\alpha} + E$ to the final LM loss of each method, where C denotes training FLOPs and $E = 0.72$ is an estimated irreducible loss. Following our fitting code, we perform linear regression on $\log_2(\mathcal{L} - E)$ against $\log_{10} C$ and convert the resulting fit back to the shifted power-law form above. The fitted parameters are reported in Table 6.

C Training FLOPs and Parameter Overhead Analysis

We derive the per-layer parameter overhead introduced by xHC and mHC and report the resulting training FLOPs cost.

C.1 Per-Layer Parameter Overhead

Each Transformer layer contains two sublayers (attention and MLP). Let C denote the hidden dimension, N the expansion rate, k the number of active streams, K_r the temporal augmentation rank, and $\{\kappa_1, \ldots, \kappa_r\}$ the depthwise convolution kernel sizes. Temporal augmentation is applied only after the MLP sublayer.

xHC. Each sublayer adds a stream router (N^2C), a pre-mapping generator (N^2C), and a residual-mapping generator (k^3C). The post-mapping generator operates at rank K_r for the MLP sublayer

Table 5: Scaling-law model configurations. All models use context length 8192, RoPE [36] (θ=50000), 144 routed experts, top-8 routing, 500 warmup steps, and the same data mixture.

Active Parameters†	Total Parameters	Layers	MoE Layers	Hidden Dimension	FFN Dimension (dense layer)	Expert FFN Dimension	Q/KV Heads	Training Budget	Global Batch Size	Base Learning Rate
180M	1.08B	16	15	656	1920	224	8/4	–	128	8.28e-4
340M	2.20B	15	14	1024	3104	320	8/4	–	160	6.95e-4
460M	3.74B	23	22	1040	3072	352	8/4	–	208	5.74e-4
1.10B	9.46B	15	14	2080	6144	704	16/8	–	288	4.82e-4

†Activated parameters per token, excluding embeddings.

Table 6: Fitted scaling-law parameters.

Method	A	α
Baseline	109.303	0.0936
mHC (N=4)	99.139	0.0920
xHC (N=16, k=4)	97.703	0.0919

(with temporal augmentation) and rank 1 for the attention sublayer. Depthwise convolution weights are added only to the MLP sublayer. The total per-layer count is:

$$P_{\text{xHC}} = \left(4N^2 + 2k^3 + k^2 + k^2 K_r + \textstyle\sum_{i=1}^{r} \kappa_i\right) \cdot C\,. \tag{14}$$

mHC. All N streams are active (k=N), with no router, no temporal augmentation (K_r=1), and no convolution:

$$P_{\text{mHC}} = (4N^2 + 2N^3) \cdot C\,. \tag{15}$$

The $2N^3C$ residual-mapping term is the source of cubic cost scaling.

Comparison. With K_r=4 and kernel sizes $\{4, 8, 12\}$, xHC has the following per-layer overheads under representative active-stream budgets:

$$P_{\text{xHC, }N=4,\,k=2} = 124\,C, \qquad P_{\text{xHC, }N=16,\,k=4} = 1256\,C,$$
$$P_{\text{mHC, }N=4} = 192\,C, \qquad P_{\text{mHC, }N=16} = 9216\,C.$$

xHC at N=16, k=4 uses **7.3× fewer** parameters per layer than dense mHC at the same expansion rate, while providing the same N=16 residual-memory capacity.

C.2 Training FLOPs Overhead

The parametric training FLOPs overhead per token introduced by the residual-stream module is:

$$F_{\text{HC}} = 6 \cdot P_{\text{HC}} \cdot L\,, \tag{16}$$

where P_{HC} is instantiated as P_{xHC} or P_{mHC} depending on the method. The factor of 6 accounts for forward and backward passes through parametric projections. This estimate excludes non-parametric operations such as Sinkhorn normalization, residual mixing, LayerNorm on the multi-stream state, and Gram–Schmidt projections.

We compute the backbone training FLOPs per token F_{backbone} using the standard $6\times$ approximation for linear layers, plus the sequence-dependent attention cost $6Sn_q d_h L$ (assuming causal attention with average context length $S/2$) and the LM-head projection $6VC$ [24]. The overhead percentage is $\Delta\% = F_{\text{HC}}/F_{\text{backbone}} \times 100$. Table 7 reports the overhead across methods and scales.

xHC at N=16 incurs 3.0% additional training FLOPs on the 28B model (4.1% on 18B), while dense mHC at the same expansion rate incurs 22.3% (18.9% on 18B). These results show that sparse residual updates keep the cost of large-N expansion modest across both scales.

D Additional Ablation Studies

We provide additional ablations on temporal feature augmentation and stream-budget selection, covering augmentation placement, Gram–Schmidt orthogonalization, multi-scale temporal components, and the balance between fixed and routed streams. Unless otherwise specified, experiments are

Table 7: Parameter and training FLOPs overhead. P/layer is the per-layer HC parameter count; $\Delta P\%$ is the parameter overhead relative to the vanilla backbone's activated parameters; $\Delta F\%$ is the training FLOPs overhead.

	18B (C=2112, L=28)			**28B** (C=2560, L=32)		
	P/layer	$\Delta P\%$	$\Delta F\%$	P/layer	$\Delta P\%$	$\Delta F\%$
mHC (N=4)	405K (192C)	0.5%	0.7%	492K (192C)	0.6%	0.5%
mHC (N=16)	19.5M (9216C)	26.3%	18.9%	23.6M (9216C)	30.2%	22.3%
xHC (N=16, k=4)	2.65M (1256C)	3.5%	4.1%	3.22M (1256C)	4.1%	3.0%

Table 8: Additional ablations on temporal feature augmentation. Lower validation loss is better.

Variant	Val. Loss ↓
xHC default	1.983
w/o Gram–Schmidt orthogonalization	1.984
+ Attention-side temporal augmentation	1.985

conducted under the same 10B MoE setting as Section 4.5, and we report validation loss on the Pile test set.

Attention-side Temporal Feature Augmentation. In xHC, temporal feature augmentation is applied only after MLP (MoE FFN) layers rather than after attention layers. This design follows the different roles of the two sublayers: attention already performs content-dependent information mixing across tokens, whereas the MLP sublayer processes each token independently and therefore provides a cleaner place to inject lightweight local contextual features. We also test adding the same temporal convolution after attention layers. As shown in Table 8, this variant gives a similar validation loss but is slightly worse than the default design by 0.002, indicating that attention-side temporal augmentation brings little additional benefit while introducing extra computation and implementation complexity. We therefore use temporal augmentation only after MLP layers throughout the paper.

Effect of Gram–Schmidt Orthogonalization. Section 3.3.1 argues that convolutional components can retain strong components aligned with the original layer output. Directly combining these correlated components may amplify the original write-back direction and make write-back scaling less predictable. We therefore apply Gram–Schmidt orthogonalization to remove collinear components and keep the augmented write-back components distinct. As shown in Table 8, removing Gram–Schmidt orthogonalization gives nearly the same validation loss as the default xHC variant at the 10B ablation scale, indicating that this normalization does not remove useful signal. However, at the 18B scale, the cosine similarity between convolutional branches and the main branch can exceed 0.7, and removing Gram–Schmidt orthogonalization leads to training instability. We therefore retain Gram–Schmidt orthogonalization as a low-cost conditioning step that improves large-scale training stability while preserving validation performance.

Table 9: Ablation on the number of temporal convolution branches. All variants use dense mHC at N=16 without sparse residual updates.

Variant	**Conv Branches**	Val. Loss ↓
mHC	0	1.998
mHC + Temp Aug (single-scale)	1	1.989
mHC + Temp Aug (multi-scale)	3	1.984

Effect of multi-scale temporal components. We ablate the number of temporal convolution branches used in temporal feature augmentation. This experiment is conducted on dense mHC at N=16, without the sparse residual-stream architecture, so that the effect of enriching the write-back signal can be isolated. As shown in Table 9, adding one causal convolution branch improves validation loss from 1.998 to 1.989, confirming that local temporal information provides useful additional write-back signal. Using three branches with different kernel sizes further improves the loss to 1.984. This supports our design choice: multiple convolution branches provide write-back components over

different temporal ranges, making the augmented signal more diverse and more effective for large-N expansion.

D.1 Routing and Stream-Budget Sensitivity

We provide guidance on selecting the active-stream budget k and fixed-stream count m for xHC at N=16. Fixed streams provide persistent paths across layers, while the remaining $k - m$ active streams are selected dynamically by the sigmoid router.

Fixed streams. Table 10 varies m at k=4 and includes a k=5, m=3 configuration to examine the effect of retaining two routed streams. At k=4, one and two fixed streams yield similar validation losses (1.982 and 1.983), both lower than fully dynamic routing (1.986). Increasing m to 3 leaves only one routed stream and gives a loss of 1.986; increasing k to 5 restores two routed streams and recovers a loss of 1.983. These results support balancing persistent and dynamically selected paths, rather than maximizing the number of fixed streams. We use m=2 to retain two persistent paths alongside two routed streams.

Table 10: Sensitivity to the fixed-stream count at N=16 on the 10B MoE model. All configurations use sigmoid routing and the same temporal augmentation.

Active streams k	Fixed streams m	Routed streams $k - m$	Val. Loss ↓
4	0	4	1.986
4	1	3	1.982
4	2 (default)	2	1.983
4	3	1	1.986
5	3	2	1.983

Active-stream budget. Table 2 compares $(k, m) = (2, 1)$, $(4, 2)$, and $(8, 2)$, with validation losses of 1.991, 1.983, and 1.982, respectively. Increasing k from 4 to 8 at fixed m=2 improves loss by only 0.001 while increasing the $O(k^3C)$ residual-mapping generation cost by a factor of eight. Thus, k=4 provides a practical balance between quality and computation.

Default configuration. We use N=16, k=4, m=2 in both the 18B and 28B experiments without retuning the stream budgets at each scale. The nearby configurations above show that this choice is not an isolated optimum. When residual-state memory is constrained, reducing N provides a separate capacity–cost trade-off, as examined in Section 4.4.

E Limitations and Outlook

This work validates xHC on MoE Transformers up to 28B parameters, which represents an important and practical setting for modern large-scale pre-training. While this provides strong evidence for the effectiveness of xHC in contemporary MoE architectures, extending the evaluation to dense Transformers and to post-training stages, such as supervised fine-tuning and related alignment procedures, remains a natural next step. In addition, although xHC introduces only small training-FLOPs overhead, the expanded N-stream residual state can increase activation memory. Co-designing xHC with memory-efficient training techniques, such as activation checkpointing and quantized residual states, is therefore a promising direction.

F Broader Impact

This work introduces an architectural improvement for Transformer pre-training and does not directly enable new applications or interact with sensitive data. However, by improving the compute efficiency of language model training, xHC could contribute to the development of more capable models, which may carry dual-use risks such as generating disinformation or enabling malicious automation. These risks are shared broadly across efficiency-oriented research and are best addressed through responsible release practices, usage monitoring, and alignment techniques applied at the deployment stage rather than at the architectural level.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We state the contributions and scope in the abstract (lines 11-21) and introduction (lines 86-95).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not present formal theoretical results or proofs. The analysis is empirical and mechanistic, supported by ablations and scaling experiments rather than theorem-style claims.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4 and Appendix A provide the model configurations, training hyperparameters, evaluation protocols, and ablation settings needed to reproduce the main experimental comparisons.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release the implementation code for xHC, together with instructions for reproducing the main experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Experiments section and Appendix A report model configurations, optimizer settings, learning-rate schedules, context length, evaluation benchmarks, and few-shot settings. Compared methods use matched data budgets at each scale.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following standard practice for large-scale language model pre-training, we report deterministic benchmark scores and training losses rather than error bars, since repeating all large-scale runs is prohibitively expensive. To mitigate run-specific effects, we evaluate across multiple model scales, compute budgets, and ablation settings, including scaling-law experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Training runs use up to 128 GPUs, as reported in Appendix A. Appendix C provides analytical training FLOPs estimates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix F.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: To our knowledge, the proposed xHC does not pose a risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite the existing methods, benchmarks, and model/backbone designs used or discussed in the paper, and our experiments follow standard usage of these assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not introduce new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method does not use LLMs as an important, original, or non-standard component. The LLM usage, if present, is limited to writing, editing, or formatting and does not affect the methodology or experiments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.