ON THE MIXING TRADE-OFF FOR EXPRESSIVE HYPER-CONNECTIONS: A GEOMETRIC CONTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Manifold-constrained Hyper-Connections (mHC) extends residual connections to multiple streams and stabilizes residual propagation by constraining stream-mixing residual matrices to the Birkhoff polytope, the set of doubly stochastic matrices. In this work, we investigate *how to effectively mix residual streams within the Birkhoff polytope*. We reveal a **mixing trade-off**: when **under-mixed** residual matrices stay close to the vertices of the Birkhoff polytope, hyper-connections will degenerate into a set of nearly independent streams; when **over-mixed** residual matrices are concentrated around the uniform barycenter of the polytope, streams will become less distinguishable from each other. We theoretically validate our findings by bounding the cumulative residual mixing, stream difference and gradient difference of mixing coefficients across layers. Motivated by these findings, we propose Contraction for Manifold-Constrained Hyper-Connections (cmHC), an exact doubly stochastic method which geometrically contracts the Birkhoff polytope toward its interior. This contraction mitigates under-mixing by forcing the residual matrices to explore deeper into the polytope, with a provable stream mixing bound and no new parameterization. Moreover, we apply a moderate contraction at the shallow layers to prevent over-mixing, enabling stable gradient propagation on extremely large-scale model pretraining. Extensive experiments on GPT-2 and mixture-of-experts at both 6B and 300B parameter scales demonstrate the effectiveness and scalability of cmHC. We also develop fused operators for cmHC on Ascend NPUs, reducing hyper-connection execution latency by up to 44.8%. Code is available at https://anonymous.4open.science/r/cmhc.

1 INTRODUCTION

Hyper-Connections (HC) (Zhu et al., 2025) extend standard residual connections (He et al., 2016) with learnable mixing across multiple feature streams, shown in Figure 1(a). As unconstrained mixing can destabilize signal propagation in deep networks, manifold-constrained Hyper-Connections (mHC) (Xie et al., 2026) constrains the residual matrices to the Birkhoff polytope, the set of doubly stochastic matrices. mHC-lite (Yang & Gao, 2026) provides an exact doubly stochastic constraint through learnable convex combinations of permutation matrices. Although these approaches exhibit training stability by doubly stochastic stream mixing, they do not ensure effective information exchange across streams. We study *how to mix streams more effectively in hyper-connections* by analyzing two ill-conditioned mixing states: under-mixing and over-mixing. See Appendix B for more related work.

Under-mixing. Prior work has observed an under-mixing state in hyper-connections: residual mixing matrices often remain close to the identity, limiting cross-stream interaction (Alimaskina et al., 2026). We further observe that this behavior is not specific to the identity: mixing can remain concentrated near other permutation vertices favored by initialization. In these cases, the streams are more permuted than actively mixed, limiting the expressivity of multi-stream architectures. To address this issue, existing methods have explored the interior of the Birkhoff polytope through different parameterizations of residual matrices (Liu et al., 2026; Dandachi & Diggs-Galligan, 2026; Guo et al., 2026), but they often rely on more involved constructions without a strictly guaranteed mixing effect, and their effectiveness is not validated on large-scale model pretraining. This motivates our **geometric contraction** strategy with *a provable stream mixing bound* and *no new parameterization*: we restrict the access to the permutation vertices and force the residual matrices to explore deeper

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.

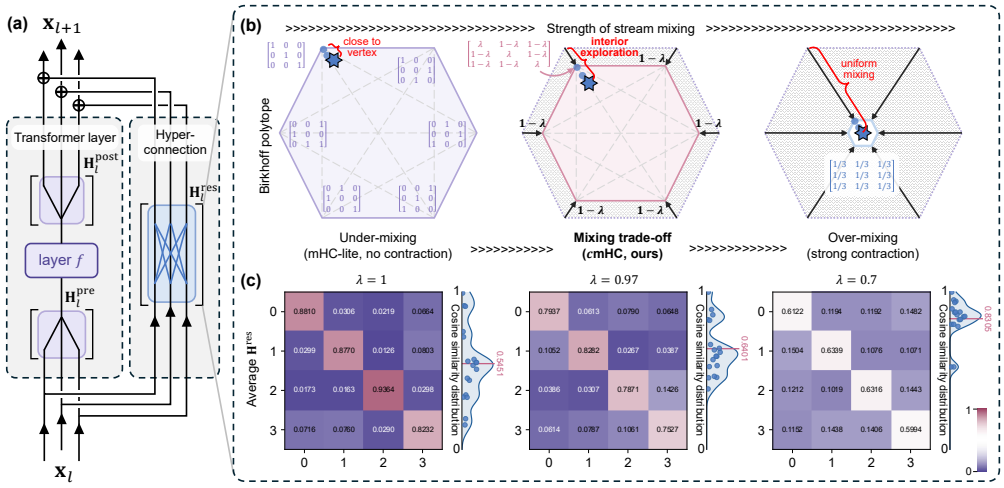


Figure 1: Illustration of our geometric contraction and its effect on stream mixing. **(a)** Architecture of hyper-connections at layer l. Our method focuses on the hyper-connection module. **(b)** Two-dimensional projections of the Birkhoff polytope for $n = 3$, along with the sampled trajectories of layer-averaged residual matrices during training, ended at the star ★. Traditional methods like `mHC-lite` do not encourage an interior exploration of the polytope, so the residual matrices remain close to a permutation vertex; our moderate contraction encourages interior exploration, while a strong contraction may drive the residual matrices toward the uniform barycenter. **(c)** Layer-averaged residual matrices and cosine similarity distributions between streams for $n=4$. A lower contraction coefficient λ indicates a stronger contraction.

into the polytope. We propose Contraction for Manifold-Constrained Hyper-Connections (`cmHC`), which contracts the polytope into a smaller region inside it by replacing permutation matrices with smoothed ones. This strictly guarantees a positive mixing while preserving exact double stochasticity. As illustrated in Figure 1(b), the contraction prevents residual mixing from degeneration around the permutation vertices, indicating guaranteed interior exploration.

The mixing trade-off. Another important aspect of our work is the revelation of the potential **over-mixing** issue that, to our knowledge, remains largely unexplored in recent efforts. Although they encourage active stream mixing by different constraints and parameterizations, we argue that **stronger mixing is not always better, because excessive mixing makes feature streams less distinguishable**. As shown in Figure 1(c), the average similarity between streams becomes closer to 1 when the contraction is strengthened. We theoretically show that the cumulative mixing bound depends jointly on contraction strength and the number of contracted layers. We further reveal the "*butterfly effect*" of stream mixing: even a numerically small contraction affects the stream difference and gradient difference of mixing coefficients, thereby amplifying over-mixing in deep networks. These findings motivate a simple but effective strategy in `cmHC`: we apply moderate contraction only at shallow layers, encouraging early information exchange while allowing near-identity mappings in deeper layers. We validate the effectiveness and scalability of `cmHC` by extensive experiments on GPT-2 and mixture-of-experts at both 6B and 300B parameter scales, showing how the trade-off design achieves a lower training loss while stabilizing gradient propagation on extremely large-scale model pretraining. Furthermore, despite no new parameterization required, we improve the efficiency of `cmHC` in industrial scenarios by developing fused operators on Ascend NPUs, reducing hyper-connection execution latency by up to 44.8%. In summary, our contributions are threefold:

- We theoretically and empirically reveal the mixing trade-off in `mHC`: under-mixing makes hyper-connections degenerate into a set of single streams, while over-mixing makes streams less distinguishable from each other.
- We propose `cmHC`, an exact doubly stochastic method which performs a moderate geometric contraction to the Birkhoff polytope without introducing additional parameters.
- We evaluate `cmHC` through extensive experiments from GPT-2 to 6B-MoE and 300B-MoE to validate its effectiveness and scalability, and develop efficient fused operators on Ascend NPUs to support efficient execution.

2 From Under-mixing to Geometric Contraction

Preliminary. Hyper-connections maintain n feature streams $\mathbf{x}_l \in \mathbb{R}^{n\times d}$ and update them as

$$\mathbf{x}_{l+1} = \mathbf{H}_l^{\text{res}}\mathbf{x}_l + \mathbf{H}_l^{\text{post}\top} f(\mathbf{H}_l^{\text{pre}}\mathbf{x}_l; \mathbf{\Theta}_l), \tag{1}$$

where $\mathbf{H}_l^{\text{res}} \in \mathbb{R}^{n\times n}$ mixes residual streams and $\mathbf{H}_l^{\text{pre}}, \mathbf{H}_l^{\text{post}} \in \mathbb{R}^{1\times n}$ aggregate and redistribute the feature streams, respectively. To ensure exact double stochasticity, `mHC-lite` parameterizes the residual matrix as a convex combination of the $n!$ permutation matrices $\mathbf{P}_m$:

$$\mathbf{H}_l^{\text{res}} = \sum_{m=1}^{n!} \alpha_{l,m}\mathbf{P}_m, \qquad \boldsymbol{\alpha}_l = \text{softmax}\left(a_l^{\text{res}}\mathbf{x}_l'\mathbf{W}_l^{\text{res}} + \mathbf{b}_l^{\text{res}}\right), \tag{2}$$

where $\mathbf{x}_l'$ is the normalized, flattened feature streams, and a_l^{res}, $\mathbf{W}_l^{\text{res}}$, and $\mathbf{b}_l^{\text{res}}$ are learnable parameters. This exactly constrains the residual matrices in a Birkhoff polytope $\mathcal{B}_n = \text{conv}\{\mathbf{P}_m\}_{m=1}^{n!}$ because of the Birkhoff–von Neumann theorem (Birkhoff, 1946); see Appendix A.2 for details.

2.1 Under-mixing in Hyper-Connections

Under-mixing has been widely considered a key bottleneck that limits the expressivity of hyper-connections. Intuitively, it makes the multi-stream architecture degenerate into a set of nearly independent streams. Prior work has observed this phenomenon in `mHC-lite`: in most layers, $\mathbf{H}_l^{\text{res}}$ stays close to the identity matrix with only small off-diagonal mass, instead of learning active stream mixing (Liu et al., 2026; Alimaskina et al., 2026). Geometrically, $\mathbf{H}_l^{\text{res}}$ remains near the identity vertex of the Birkhoff polytope during training, as shown in Figure 1(b).

We further observe that **under-mixing is not specific to the identity; it can occur around any permutation matrix favored at initialization**. In `mHC-lite`, the input $\mathbf{x}_0$ is first duplicated across n streams. Then, the diagonal entries of the residual bias $\mathbf{b}_l^{\text{res}}$ are set to 0, while the others are set to -8. This makes $\mathbf{H}_l^{\text{res}}$ close to the identity matrix through the softmax in (2). When we change the initialization of $\mathbf{b}_l^{\text{res}}$ to favor another permutation matrix, the average $\mathbf{H}^{\text{res}}$ stays close to that permutation during training, as shown in Figure 2.

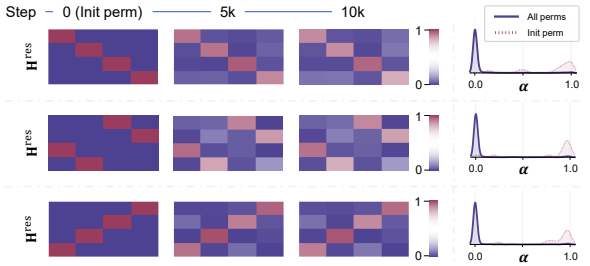


Figure 2: The under-mixing phenomenon under different initializations. **Left**: $\frac{1}{L}\sum_{l=0}^{L-1}\mathbf{H}_l^{\text{res}}$ at training steps 0 (initial state), 5k, and 10k. Across different initializations, $\frac{1}{L}\sum_{l=0}^{L-1}\mathbf{H}_l^{\text{res}}$ keeps close to the initial permutation matrix, not only the identity. **Right**: entry distributions of mixing coefficients $\boldsymbol{\alpha}$. The coefficient assigned to the initialized permutation ("Init perm") stays close to 1, while the majority of coefficients over all permutations ("All perms") stays close to 0.

From the perspective of training dynamics, the stream $\mathbf{x}_{l,m}$ favored by initialization receives a larger coefficient update, which pushes its mixing coefficient $\alpha_{l,m}$ toward 1 and suppresses the remaining coefficients. As a result, a single $\mathbf{P}_m$ dominates $\mathbf{H}_l^{\text{res}}$. Since $\mathbf{P}_m$ only permutes stream positions, cross-stream information exchange is suppressed. Figure 2 shows that the entry distribution of $\boldsymbol{\alpha}$ remains close to its initial near-Dirac state during training: the coefficient of the initialized permutation stays near 1, while most other permutation coefficients stay near 0.

2.2 cmHC: Geometric Contraction of the Birkhoff Polytope

To mitigate under-mixing, we introduce a geometric contraction $c(\cdot;\lambda) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ which replaces each permutation matrix $\mathbf{P}_m$ with a smoothed version:

$$\mathbf{P}_m' = c(\mathbf{P}_m;\lambda) = \lambda\mathbf{P}_m + \frac{1-\lambda}{n-1}(\mathbf{1}_n\mathbf{1}_n^\top - \mathbf{P}_m). \tag{3}$$

where $\lambda \in (1/n, 1)$ controls the contraction strength. The case $\lambda \to 1$ recovers `mHC-lite`, while $\lambda \to 1/n$ yields uniform mixing. The smoothed permutation matrices can be precomputed and replace the original bases without adding learnable parameters or changing their size.

Let $\mathbf{J}_n = \mathbf{1}_n\mathbf{1}_n^\top/n$ and $\lambda' = (n\lambda - 1)/(n-1)$. Using the coefficients from (2), the residual matrix is

$$\mathbf{H}_l^{\text{res}'} = \sum_{m=1}^{n!} \alpha_{l,m}\mathbf{P}'_m = c(\mathbf{H}_l^{\text{res}}; \lambda) = \lambda'\mathbf{H}_l^{\text{res}} + (1-\lambda')\mathbf{J}_n. \tag{4}$$

The resulting contraction polytope $\mathcal{B}_n(\lambda) = \text{conv}\{\mathbf{P}'_m\}_{m=1}^{n!}$ has the following properties.

Theorem 2.1 (Preservation of double stochasticity). *For $\lambda \in (1/n, 1)$, $\mathcal{B}_n(\lambda) \subseteq \mathcal{B}_n$. In particular, every contracted residual matrix is doubly stochastic and has spectral norm at most one.*

Theorem 2.2 (Lower bound of stream mixing). *For $\lambda \in (1/n, 1)$, every entry of a contracted residual matrix satisfies*

$$h_{l,ij}^{\text{res}'} \geq \min\left(\lambda, \frac{1-\lambda}{n-1}\right) > 0. \tag{5}$$

Proofs are provided in Appendices C.1 and C.2. Theorem 2.1 shows that c`mHC` preserves **exact double stochasticity**, ensuring $\|\mathbf{H}_l^{\text{res}'}\|_2 \leq 1$. Thus, for fixed mixing coefficients, the residual mixing operator and its transpose are non-expansive, preserving the stabilizing property of doubly stochastic mixing (Xie et al., 2026). Meanwhile, Theorem 2.2 guarantees a strictly positive contribution from every input stream to every output stream for $1/n < \lambda < 1$, preventing extreme under-mixing.

Contraction mapping. Equation (4) expresses c as an interpolation between the original matrix and the uniform barycenter $\mathbf{J}_n$. For $1/n < \lambda < 1$, we have $0 < \lambda' < 1$, and for any $\mathbf{H}, \mathbf{G} \in \mathcal{B}_n$,

$$\|c(\mathbf{H}; \lambda) - c(\mathbf{G}; \lambda)\|_F = \lambda'\|\mathbf{H} - \mathbf{G}\|_F. \tag{6}$$

Thus, c is a contraction mapping on $\mathcal{B}_n$ under the Frobenius norm, with contraction factor λ'. Since $c(\mathbf{J}_n; \lambda) = \mathbf{J}_n$, it also scales each matrix's distance to the barycenter by λ', mapping the original polytope to a smaller region in its relative interior. The general definition of a contraction mapping is provided in Appendix A.3.

As illustrated in Figure 1(b), the original permutation vertices are no longer reachable. Even if $\boldsymbol{\alpha}_l$ concentrates on a single component, $\mathbf{H}_l^{\text{res}'}$ becomes the smoothed $\mathbf{P}'_m$ rather than the permutation vertex $\mathbf{P}_m$. This indicates that the geometric contraction guarantees a positive cross-stream mixing, which mitigates the under-mixing effect.

3 The Mixing Trade-off

Stronger mixing promotes information exchange but may also reduce stream distinguishability. To examine this, we first analyze products of contracted residual matrices. We then study feature and gradient propagation using a local linear approximation to the Transformer branch, $\mathbf{H}^{\text{post}\top} f(\mathbf{H}^{\text{pre}}\mathbf{x}; \boldsymbol{\Theta}) \to \mathbf{H}^{\text{post}\top}\boldsymbol{\Theta}\mathbf{H}^{\text{pre}}\mathbf{x} = \mathbf{A}\mathbf{x}$, yielding

$$\mathbf{x}_{l+1} \approx (\mathbf{H}_l^{\text{res}'} + \mathbf{A}_l)\mathbf{x}_l. \tag{7}$$

Together, these analyses motivate a moderate contraction to shallow layers as a trade-off scheme.

3.1 Convergence of Cumulative Contraction

Equation (4) defines a contraction on the complete metric space $(\mathcal{B}_n(\lambda), \|\cdot\|_F)$ for $1/n < \lambda < 1$. By Banach's fixed-point theorem (Banach, 1922), repeated application of this contraction converges to a unique fixed point, that is, the uniform barycenter $\mathbf{J}_n$. Convergence to the barycenter indicates an over-mixing state: **all output streams assign the same importance to every input stream, rather than learning dynamic stream interactions**. Details are provided in Appendix A.3.

Across network layers, the mixing effect accumulates through products of residual matrices. To quantify how contraction strength affects this cumulative mixing, we use *Dobrushin's ergodicity*

coefficient (Gaubert & Qu, 2015) $\tau(\mathbf{H}_l^{\text{res}'}) \in [0, 1]$ to measure the convergence of a non-homogeneous matrix product. It is defined as

$$\tau(\mathbf{H}_l^{\text{res}'}) := 1 - \min_{i<j} \sum_{k=1}^{n} \min(h_{l,ik}^{\text{res}'}, h_{l,jk}^{\text{res}'}). \tag{8}$$

A smaller τ means that different rows share more common mass; when τ approaches 0, the rows become nearly identical, indicating stronger over-mixing. Then, the following theorem relates the convergence speed of cumulative stream mixing to the contraction strength.

Theorem 3.1 (Convergence of cumulative stream mixing). *Let $\lambda_l \in (1/n, 1]$ for $l = 0, \ldots, L-1$, where $\lambda_l = 1$ leaves layer l uncontracted. Define $\lambda_l' = (n\lambda_l - 1)/(n-1)$ and $\mathbf{H}_l^{\text{res}'} = \lambda_l' \mathbf{H}_l^{\text{res}} + (1 - \lambda_l')\mathbf{J}_n$. Then*

$$\tau\left(\prod_{l=0}^{L-1} \mathbf{H}_{L-1-l}^{\text{res}'}\right) \leq \prod_{l=0}^{L-1} \lambda_l'. \tag{9}$$

See Appendix C.3 for the proof. If all layers satisfy $\lambda_l' < 1$, the cumulative residual matrix approaches $\mathbf{J}_n$ **at an exponential rate** as L increases. This implies that limiting such contraction to a small number of layers is important for alleviating the exponential convergence.

Figure 3 visualizes the cumulative residual matrix $\prod_{l=0}^{L-1} \mathbf{H}_l^{\text{res}'}$ and its Dobrushin coefficient τ under different λ and L. When λ decreases from 1 to 0.97, τ drops by about 0.5, indicating a much stronger cumulative mixing effect. With $\lambda = 0.7$, τ becomes close to 0, and the cumulative residual matrix is nearly indistinguishable from the uniform barycenter of the Birkhoff polytope. And generally, increasing L further reduces τ, which is consistent with the decreasing cumulative upper bound.

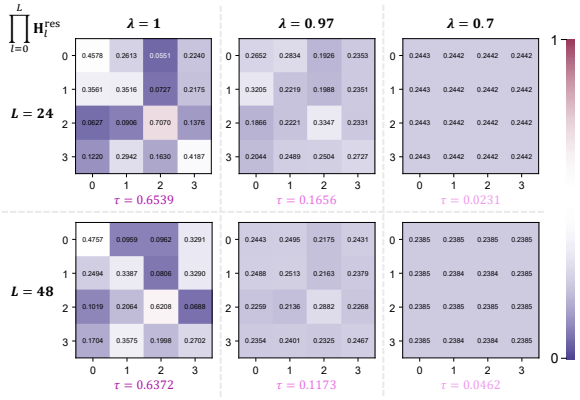


Figure 3: Cumulative residual matrix $\prod_{l=0}^{L-1} \mathbf{H}_l^{\text{res}'}$ and Dobrushin's ergodicity coefficient τ under different λ and network depth L. Note that the matrices slightly deviate from exact doubly stochasticity because of the inference-time low-precision conversion.

3.2 The Butterfly Effect of Geometric Contraction

This section reveals a "butterfly effect" caused by geometric contraction: even a numerically small contraction can not only accelerate the convergence of the cumulative residual matrix, but also affect inference and gradient propagation of $\boldsymbol{\alpha}$, thereby increasing the risk of over-mixing in mHC.

Stream difference $\|\tilde{\mathbf{x}}\|_F$ during inference. To highlight the differences among streams, we first apply a mean-difference decomposition to the feature streams:

$$\mathbf{x} = \underbrace{\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\mathbf{x}}_{\text{stream mean}} + \underbrace{(\mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\mathbf{x}}_{\text{stream difference } \tilde{\mathbf{x}}} = \mathbf{J}_n\mathbf{x} + \mathbf{\Pi}_n\mathbf{x}, \tag{10}$$

where $\mathbf{\Pi}_n = \mathbf{I}_n - \mathbf{J}_n$ and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Every column of the stream difference $\tilde{\mathbf{x}}$ sums to 0. To isolate how the contraction affects the stream differences, we align with the local linear approximation and do not consider the differences introduced by the non-linear Transformer branch: $\mathbf{\Pi}_n\mathbf{A}_l\mathbf{J}_n \approx \mathbf{0}$ for $l \geq 1$. We start from the output streams from the first layer $\tilde{\mathbf{x}}_1$ with stream difference norm $\|\tilde{\mathbf{x}}_1\|_F > 0$. The following theorem bounds the norm of the stream differences.

Theorem 3.2 (Norm upper bound of stream difference). *For $L \geq 2$ hyper-connection layers, the stream difference norm satisfies*

$$\|\tilde{\mathbf{x}}_k\|_F \leq \left(\prod_{l=1}^{k-1} (\lambda_l' + \ell_l)\right) \|\tilde{\mathbf{x}}_1\|_F \tag{11}$$

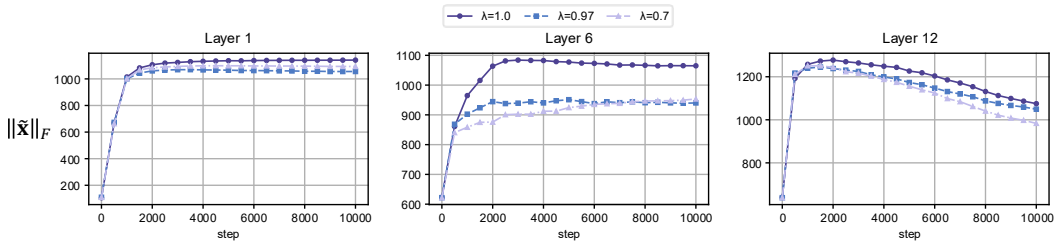


Figure 4: Frobenius norms of stream differences $\|\tilde{\mathbf{x}}\|_F$ over training steps.

for $2 \le k \le L$*, where* $\lambda_l' = \frac{n\lambda_l - 1}{n-1}$ *is the normalized contraction coefficient and* $\ell_l = \sup_{\mathbf{Z} \neq \mathbf{0}} \frac{\|\tilde{\mathbf{A}}_l \mathbf{Z}\|_F}{\|\mathbf{Z}\|_F}$ *is the Lipschitz constant of* $\tilde{\mathbf{A}}_l = \mathbf{\Pi}_n \mathbf{A}_l$.

See Appendix C.4 for the detailed proof. Theorem 3.2 indicates that for fixed ℓ_l and $\tilde{\mathbf{x}}_1$, a stronger contraction can reduce the upper bound of the stream difference norm. The multiplicative accumulation implies that contraction in earlier layers can suppress differences at later layers. This reveals the over-mixing risk: stronger information exchange can make streams less distinguishable.

We analyze the norm of stream differences $\|\tilde{\mathbf{x}}\|_F$ in Figure 4. Note that the upper bound of $\|\tilde{\mathbf{x}}\|_F$ is also affected by the training stability of the Transformer layer (ℓ_l), so deeper layers are not guaranteed to have smaller norms. Still, the stream difference norms for $\lambda = 0.97$ and $\lambda = 0.7$ are clearly lower than that of $\lambda = 1$. This verifies that geometric contraction suppresses inter-stream differences.

Gradient of the mixing coefficient $\boldsymbol{\alpha}$. We next analyze how contraction strength affects the learning of mixing coefficients $\boldsymbol{\alpha}$. Intuitively, simply replacing the preset permutation matrices $\mathbf{P}_m$ with smoother versions $\mathbf{P}'_m$ should not directly affect the learning of $\boldsymbol{\alpha}$. However, we find that geometric contraction in cmHC induces a smoother $\boldsymbol{\alpha}$ distribution by reducing the upper bound of the gradient difference between any α-pair of permutation matrices. Formally:

Theorem 3.3 (Gradient difference of different permutations)**.** *For two different permutation matrices* $\mathbf{P}_i, \mathbf{P}_j \in \mathbb{R}^{n \times n}$*, the gradient difference of the pretraining loss* $\mathcal{L}$ *w.r.t. their coefficients* $\alpha_{l,i}, \alpha_{l,j}$ *is bounded by*

$$|(\nabla_{\alpha_{l,i}} - \nabla_{\alpha_{l,j}})\mathcal{L}| \le 2\lambda_l' \|\nabla_{\tilde{\mathbf{x}}_{l+1}} \mathcal{L}\|_F \|\tilde{\mathbf{x}}_l\|_F. \quad (12)$$

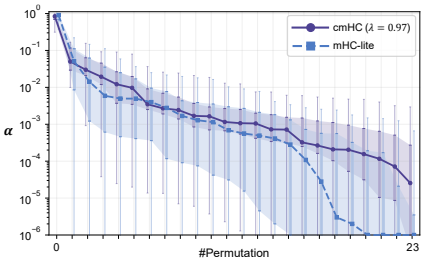


Figure 5: Distributions of mixing coefficients $\boldsymbol{\alpha}$ over the vertices of $\mathcal{B}_4$ (across all layers of nanoGPT-L). The 24 vertices are ranked by their α value. Curves indicate the medians. See Appendix E.3 for full results.

See Appendix C.5 for the detailed proof. From Theorem 3.2, contraction lowers the bound on $\|\tilde{\mathbf{x}}_l\|_F$. For a fixed incoming gradient $\|\nabla_{\tilde{\mathbf{x}}_{l+1}} \mathcal{L}\|_F$, this also lowers the corresponding upper bound on $|(\nabla_{\alpha_{l,i}} - \nabla_{\alpha_{l,j}})\mathcal{L}|$. This reveals that contraction tends to assign similar gradients to different coefficients of permutation matrices, consequently **leading to a smoother distribution of $\boldsymbol{\alpha}$**, which further promotes stream mixing. This explains why a numerically small contraction can substantially encourage mixing, while a strong contraction may instead increase the risk of over-mixing.

To examine this effect, we sort the coefficients $\boldsymbol{\alpha}$ over all vertices of $\mathcal{B}_4$ in descending order, and compare their distributions across all layers between `mHC-lite` and cmHC with $\lambda = 0.97$. Figure 5 shows that cmHC assigns more mass to the non-dominant vertices rather than only accumulating mass at vertex 0. This supports that geometric contraction induces a smoother $\boldsymbol{\alpha}$ distribution.

3.3 A Moderate Contraction to the Birkhoff Polytope

The analyses in Secs. 3.1 and 3.2 identify a risk of over-mixing: cumulative contraction drives $\mathbf{H}^{\text{res}'}$ exponentially toward the uniform barycenter of the Birkhoff polytope, and the contraction can also affect stream differences and induce a smoother coefficient distribution. These findings motivate a mixing trade-off strategy to control both strength and layers of the contraction. We **apply moderate contraction only to the first K layers while allowing near-identity mappings in deeper layers**:

$$\mathbf{P}'_{l,m} = \begin{cases} \lambda \mathbf{P}_m + \frac{1-\lambda}{n-1}(\mathbf{1}_n \mathbf{1}_n^\top - \mathbf{P}_m), & l \leq K-1, \\ \mathbf{P}_m, & l > K-1. \end{cases} \tag{13}$$

We set a single contraction coefficient λ shared by all first K layers. Contractions at shallow layers are more effective than at deeper layers because the contraction effect can accumulate and influence all subsequent layers. From a hierarchical perspective of deep representation learning, shallower layers tend to encode more general features and deeper layers become more specialized (Yosinski et al., 2014), so it should be more cautious to mix streams in deeper layers. This improves the expressivity of hyper-connection architectures while reducing the risk of over-mixing.

4 Experiments

Setup. Following Yang & Gao (2026), we evaluate language modeling on two extensive web datasets: OpenWebText (Gokaslan & Cohen, 2019) and FineWeb-Edu (Lozhkov et al., 2024). We compare ~~c~~mHC with representative residual and hyper-connection baselines. (1) Vanilla residual connections "Residual" with one single stream $n = 1$ and multiple independent streams (*e.g.*, $n = 4$) without mixing. (2) Original HC (Zhu et al., 2025) without the Birkhoff constraint on the residual matrices. (3) mHC (Xie et al., 2026) with the Sinkhorn-Knopp iteration constraint. (4) mHC-lite (Yang & Gao, 2026) with the exact Birkhoff von-Neumann parameterization of the residual matrices. (5) KromHC (Zhou et al., 2026) which decomposes doubly stochastic matrices into Kronecker products of smaller ones. See Appendix D for more details.

4.1 Results on GPT-2

We train nanoGPT-S, nanoGPT-M, and nanoGPT-L from scratch for 10k iterations using the nanoGPT framework (Karpathy, 2025). These models have 12, 24, and 48 hyper-connection layers[1] and approximately 45M, 0.12B, and 0.36B parameters, respectively. All methods use $n = 4$ streams unless specified otherwise. Results are averaged over three runs.

Main comparative results. Table 1 shows that **~~c~~mHC with $\lambda = 0.97$ and $K = 2$ achieves the best average loss rank on both datasets**. On OpenWebText, this configuration outperforms both mHC and mHC-lite in training and validation loss at every scale, reducing validation loss over mHC-lite by 0.0073, 0.0024, and 0.0045, respectively. On FineWeb-Edu, ~~c~~mHC achieves the lowest training and validation losses for nanoGPT-M ($K = 10$) and the lowest training loss for nanoGPT-L ($K = 2$), while remaining competitive on nanoGPT-S. Besides, the gradient norms of ~~c~~mHC remain comparable to the traditional mHC and mHC-lite while significantly better than the unconstrained HC. These results demonstrate improved language modeling through moderate stream mixing without additional learnable parameters.

Convergence and stability. Figure 6 shows the training curves of the loss gap and gradient norm. ~~c~~mHC obtains the lowest training loss at convergence. Although HC appears to converge faster in the first 4k steps, its residual matrices are unconstrained, resulting in the largest training fluctuations among all methods. A similar pattern is observed in the gradient norm: HC continues to fluctuate by ~ 0.05 in later training stages and remains substantially higher than the other baselines, whereas ~~c~~mHC maintains one of the lowest and most stable gradient norms. Moreover, in Figure 6, all existing baselines have higher training losses than traditional residual connections with 4 independent streams, while ~~c~~mHC is the only method that achieves a lower training loss. See Appendix E.1 for more results on nanoGPT-M and nanoGPT-L.

Ablation on λ. With contraction applied to all layers, the moderate contraction $\lambda = 0.97$ consistently outperforms the strong contraction $\lambda = 0.70$ on both datasets. The validation-loss advantage grows from 0.0043 to 0.0143 on OpenWebText and from 0.0005 to 0.0125 on FineWeb-Edu as depth increases from 12 to 48 layers. These results support our mixing trade-off strategy that applies a moderate contraction.

[1] Each Transformer block contains two hyper-connection layers around attention and MLP layers, respectively.

Table 1: nanoGPT results on OpenWebText and FineWeb-Edu (mean ± standard deviation for 3 replicate experiments). All methods use 4 streams except "Residual, $n = 1$". *Avg. Loss Rank* marks the average rank over the Train Loss and Val. Loss. Best results are in **bold**.

Method	nanoGPT-S, #Layers=12			nanoGPT-M, #Layers=24			nanoGPT-L, #Layers=48			
	Train Loss	Val. Loss	Grad. Norm	Train Loss	Val. Loss	Grad. Norm	Train Loss	Val. Loss	Grad. Norm	*Avg. Loss Rank*
OpenWebText										
Residual, $n = 1$	$3.5677_{\pm 0.0006}$	$3.5636_{\pm 0.0007}$	$0.4239_{\pm 0.0194}$	$3.3371_{\pm 0.0008}$	$3.3363_{\pm 0.0007}$	$0.3452_{\pm 0.0099}$	$3.2311_{\pm 0.0009}$	$3.2442_{\pm 0.0001}$	$0.3330_{\pm 0.0064}$	11.8
Residual, $n = 4$	$3.4777_{\pm 0.0039}$	$3.4742_{\pm 0.0036}$	$0.3877_{\pm 0.0339}$	$3.2591_{\pm 0.0029}$	$3.2587_{\pm 0.0028}$	$0.3751_{\pm 0.0150}$	$3.1801_{\pm 0.0009}$	$3.1909_{\pm 0.0011}$	$0.4537_{\pm 0.0325}$	6.0
HC	$3.4749_{\pm 0.0017}$	$3.4796_{\pm 0.0072}$	$0.4300_{\pm 0.0584}$	$3.2697_{\pm 0.0008}$	$3.2681_{\pm 0.0006}$	$0.4990_{\pm 0.1291}$	$3.2232_{\pm 0.0013}$	$3.2356_{\pm 0.0011}$	$0.5519_{\pm 0.1373}$	9.7
mHC	$3.4813_{\pm 0.0059}$	$3.4775_{\pm 0.0064}$	$0.3882_{\pm 0.0269}$	$3.2562_{\pm 0.0016}$	$3.2558_{\pm 0.0013}$	$0.3734_{\pm 0.0146}$	$3.1795_{\pm 0.0020}$	$3.1920_{\pm 0.0020}$	$0.4665_{\pm 0.0357}$	6.0
mHC-lite	$3.4801_{\pm 0.0027}$	$3.4759_{\pm 0.0021}$	$0.3688_{\pm 0.0155}$	$3.2578_{\pm 0.0034}$	$3.2564_{\pm 0.0034}$	$0.3854_{\pm 0.0197}$	$3.1821_{\pm 0.0033}$	$3.1949_{\pm 0.0033}$	$0.4317_{\pm 0.0202}$	6.7
KromHC	$3.4809_{\pm 0.0024}$	$3.4766_{\pm 0.0029}$	$0.3898_{\pm 0.0338}$	$3.2612_{\pm 0.0041}$	$3.2602_{\pm 0.0041}$	$0.3867_{\pm 0.0177}$	$3.1807_{\pm 0.0019}$	$3.1934_{\pm 0.0024}$	$0.4589_{\pm 0.0322}$	8.0
cmHC, $\lambda = 0.70$, all layers	$3.4790_{\pm 0.0045}$	$3.4756_{\pm 0.0046}$	$0.3637_{\pm 0.0134}$	$3.2626_{\pm 0.0021}$	$3.2618_{\pm 0.0023}$	$0.3760_{\pm 0.0160}$	$3.2063_{\pm 0.0016}$	$3.2169_{\pm 0.0013}$	$0.4346_{\pm 0.0164}$	9.0
cmHC, $\lambda = 0.85$, all layers	$3.4770_{\pm 0.0025}$	$3.4732_{\pm 0.0026}$	$0.3634_{\pm 0.0166}$	$3.2579_{\pm 0.0031}$	$3.2575_{\pm 0.0029}$	$0.3758_{\pm 0.0150}$	$3.2012_{\pm 0.0012}$	$3.2119_{\pm 0.0014}$	$0.4334_{\pm 0.0155}$	6.7
cmHC, $\lambda = 0.97$, all layers	$3.4745_{\pm 0.0014}$	$3.4713_{\pm 0.0014}$	$0.3629_{\pm 0.0121}$	$3.2550_{\pm 0.0018}$	$3.2542_{\pm 0.0020}$	$0.3790_{\pm 0.0159}$	$3.1915_{\pm 0.0016}$	$3.2026_{\pm 0.0013}$	$0.4456_{\pm 0.0252}$	4.2
cmHC, $\lambda = 0.97$, $K = 10$	$3.4755_{\pm 0.0035}$	$3.4718_{\pm 0.0037}$	$0.3805_{\pm 0.0195}$	$3.2588_{\pm 0.0029}$	$3.2584_{\pm 0.0029}$	$0.3705_{\pm 0.0141}$	$\mathbf{3.1785}_{\pm 0.0009}$	$\mathbf{3.1893}_{\pm 0.0011}$	$0.4340_{\pm 0.0150}$	4.2
cmHC, $\lambda = 0.97$, $K = 5$	$3.4731_{\pm 0.0028}$	$3.4693_{\pm 0.0035}$	$0.3726_{\pm 0.0182}$	$3.2563_{\pm 0.0014}$	$3.2549_{\pm 0.0017}$	$0.3734_{\pm 0.0142}$	$3.1801_{\pm 0.0007}$	$3.1913_{\pm 0.0005}$	$0.4341_{\pm 0.0141}$	3.2
cmHC, $\lambda = 0.97$, $K = 2$	$\mathbf{3.4723}_{\pm 0.0025}$	$\mathbf{3.4686}_{\pm 0.0027}$	$0.3774_{\pm 0.0200}$	$\mathbf{3.2548}_{\pm 0.0006}$	$\mathbf{3.2540}_{\pm 0.0007}$	$0.3726_{\pm 0.0137}$	$3.1791_{\pm 0.0032}$	$3.1904_{\pm 0.0032}$	$0.4353_{\pm 0.0140}$	**1.3**
FineWeb-Edu										
Residual, $n = 1$	$3.5284_{\pm 0.0023}$	$3.5319_{\pm 0.0021}$	$0.4395_{\pm 0.0204}$	$3.3228_{\pm 0.0042}$	$3.3206_{\pm 0.0042}$	$0.3603_{\pm 0.0094}$	$3.2308_{\pm 0.0002}$	$3.2270_{\pm 0.0002}$	$0.3347_{\pm 0.0059}$	11.2
Residual, $n = 4$	$3.4644_{\pm 0.0017}$	$3.4684_{\pm 0.0017}$	$0.3645_{\pm 0.0127}$	$3.2489_{\pm 0.0005}$	$3.2474_{\pm 0.0008}$	$0.3689_{\pm 0.0136}$	$3.1837_{\pm 0.0023}$	$3.1819_{\pm 0.0023}$	$0.4587_{\pm 0.0297}$	5.3
HC	$3.4699_{\pm 0.0001}$	$3.4724_{\pm 0.0003}$	$0.4493_{\pm 0.0708}$	$3.2900_{\pm 0.0053}$	$3.2898_{\pm 0.0056}$	$2.7226_{\pm 2.2325}$	$3.2413_{\pm 0.0188}$	$3.2363_{\pm 0.0179}$	$0.9607_{\pm 0.6755}$	10.8
mHC	$3.4626_{\pm 0.0009}$	$3.4667_{\pm 0.0010}$	$0.3684_{\pm 0.0125}$	$3.2483_{\pm 0.0010}$	$3.2471_{\pm 0.0005}$	$0.3663_{\pm 0.0121}$	$3.1862_{\pm 0.0009}$	$3.1838_{\pm 0.0008}$	$0.4755_{\pm 0.0338}$	4.0
mHC-lite	$\mathbf{3.4621}_{\pm 0.0008}$	$\mathbf{3.4660}_{\pm 0.0013}$	$0.3709_{\pm 0.0137}$	$3.2480_{\pm 0.0026}$	$3.2477_{\pm 0.0023}$	$0.3742_{\pm 0.0163}$	$3.1884_{\pm 0.0020}$	$3.1846_{\pm 0.0021}$	$0.4502_{\pm 0.0250}$	4.3
KromHC	$3.4643_{\pm 0.0016}$	$3.4673_{\pm 0.0016}$	$0.3653_{\pm 0.0123}$	$3.2485_{\pm 0.0021}$	$3.2480_{\pm 0.0018}$	$0.3695_{\pm 0.0138}$	$3.1836_{\pm 0.0006}$	$\mathbf{3.1802}_{\pm 0.0004}$	$0.4531_{\pm 0.0249}$	4.3
cmHC, $\lambda = 0.70$, all layers	$3.4678_{\pm 0.0020}$	$3.4718_{\pm 0.0020}$	$0.3648_{\pm 0.0118}$	$3.2514_{\pm 0.0018}$	$3.2503_{\pm 0.0021}$	$0.3683_{\pm 0.0161}$	$3.2067_{\pm 0.0010}$	$3.2049_{\pm 0.0011}$	$0.4320_{\pm 0.0014}$	9.3
cmHC, $\lambda = 0.85$, all layers	$3.4652_{\pm 0.0022}$	$3.4691_{\pm 0.0019}$	$0.3653_{\pm 0.0130}$	$3.2466_{\pm 0.0017}$	$3.2454_{\pm 0.0019}$	$0.3750_{\pm 0.0174}$	$3.2059_{\pm 0.0019}$	$3.2043_{\pm 0.0019}$	$0.4393_{\pm 0.0155}$	6.0
cmHC, $\lambda = 0.97$, all layers	$3.4673_{\pm 0.0022}$	$3.4713_{\pm 0.0024}$	$0.3593_{\pm 0.0118}$	$3.2477_{\pm 0.0011}$	$3.2463_{\pm 0.0009}$	$0.3800_{\pm 0.0221}$	$3.1936_{\pm 0.0013}$	$3.1924_{\pm 0.0013}$	$0.4644_{\pm 0.0346}$	6.3
cmHC, $\lambda = 0.97$, $K = 10$	$3.4679_{\pm 0.0004}$	$3.4718_{\pm 0.0004}$	$0.3682_{\pm 0.0137}$	$\mathbf{3.2465}_{\pm 0.0019}$	$\mathbf{3.2453}_{\pm 0.0018}$	$0.3685_{\pm 0.0128}$	$3.1883_{\pm 0.0023}$	$3.1864_{\pm 0.0021}$	$0.4563_{\pm 0.0290}$	5.7
cmHC, $\lambda = 0.97$, $K = 5$	$3.4642_{\pm 0.0008}$	$3.4677_{\pm 0.0009}$	$0.3651_{\pm 0.0125}$	$3.2487_{\pm 0.0027}$	$3.2474_{\pm 0.0024}$	$0.3691_{\pm 0.0125}$	$3.1845_{\pm 0.0024}$	$3.1832_{\pm 0.0026}$	$0.4556_{\pm 0.0253}$	5.0
cmHC, $\lambda = 0.97$, $K = 2$	$3.4634_{\pm 0.0002}$	$3.4672_{\pm 0.0006}$	$0.3683_{\pm 0.0136}$	$3.2480_{\pm 0.0021}$	$3.2464_{\pm 0.0020}$	$0.3707_{\pm 0.0130}$	$\mathbf{3.1834}_{\pm 0.0012}$	$3.1817_{\pm 0.0014}$	$0.4364_{\pm 0.0181}$	**2.8**

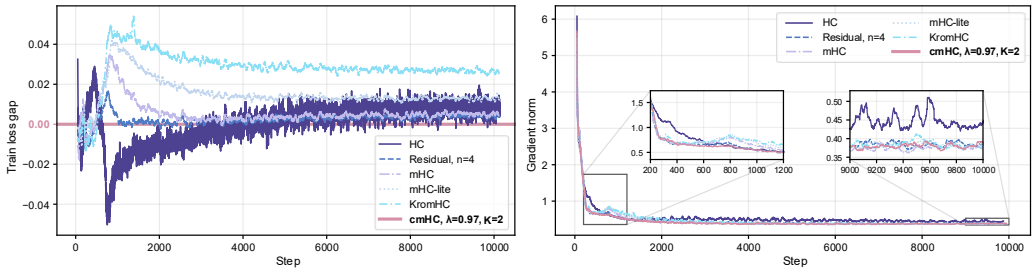


Figure 6: Loss gaps relative to cmHC and gradient norms (nanoGPT-S; OpenWebText).

Ablation on K. Fixing $\lambda = 0.97$, we test the contraction on the first $K \in \{2, 5, 10\}$ and all layers. The results in Table 1 suggest that applying contraction to only a few layers works better than applying it to all layers, especially in deeper networks. Average loss ranks also improve as K decreases from 10 to 5 to 2. These results support our mixing trade-off strategy that encourages stream mixing on shallow layers.

4.2 Scalability: From 6B to 300B Parameters

Setup. We evaluate scalability using 6B and 300B parameter mixture-of-experts (MoE) (Shazeer et al., 2017) language models trained on FineWeb-Edu. Both adopt a DeepSeek-style architecture (DeepSeek-AI, 2024) with multi-latent attention and sparse MoE layers, using four residual streams. We set the cmHC contraction coefficient to $\lambda = 0.97$ for 6B and $\lambda = 0.96$ for 300B. Full architecture, training, and contraction settings are provided in Appendix D.1.

Figure 7 compares the training loss gaps relative to cmHC and the corresponding gradient norms. At the 6B scale, after an early transient, cmHC finishes training with loss advantages of 0.002 over mHC and 0.005 over mHC-lite at step 71.5k. Moreover, the gradient norm of cmHC keeps an advantage over the two baselines, indicating its training stability. From our scalability test on 300B-MoE, cmHC also attains a lower loss than mHC, with a gap of 0.0157 at step 2k; cmHC maintains smaller and smoother gradient norms, whereas mHC exhibits larger fluctuations. These results provide evidence that the benefit of our geometric contraction extends from 6B to substantially larger 300B models, including both lower losses and more stable optimization.

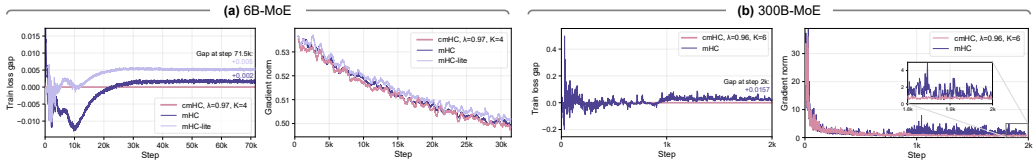


Figure 7: Loss gaps relative to cmHC and gradient norms (6B- and 300B-MoE; FineWeb-Edu).

Table 2: Total latency of the complete fused cmHC and mHC operator pipelines on Ascend 910C, including both forward and backward Pre and Post operators. Better latency time is in **bold**.

Input $[B, S, n, D]$	Total latency (μs)		Speedup	Reduction
	cmHC	mHC		
$[1, 4096, 4, 2048]$	$\mathbf{1,474.59}$	$2,670.02$	$1.81\times$	44.8%
$[1, 4096, 4, 3584]$	$\mathbf{2,706.02}$	$3,923.59$	$1.45\times$	31.0%
$[1, 4096, 4, 7168]$	$\mathbf{6,472.56}$	$8,523.01$	$1.32\times$	24.1%

4.3 Efficient Fused Operators on Ascend NPUs

Although the geometric contraction introduces no additional asymptotic computation, a direct framework implementation still launches several small operators and repeatedly accesses the widened multi-stream state. We therefore implement fused Pre and Post operators in Ascend C (Huawei Technologies Co., Ltd., 2025). The Pre operator combines coefficient generation, stream aggregation, and the projection of $\boldsymbol{\alpha}_l$ onto the contracted permutation matrices $\{\mathbf{P}'_{l,m}\}_{m=1}^{n!}$; the Post operator fuses residual mixing and post expansion. Dense projections run on cube cores, whereas reductions, nonlinearities, and the small-n mixing operations run on vector cores. The contracted bases are prepared once per layer and cached, so the kernels implement exactly the same parameterization as (3). Architecture diagrams, kernel data paths, backward equations, and memory-layout details are provided in Appendix F.

We benchmark the complete fused-operator pipeline on an Ascend 910C server with CANN 9.1.0. All workloads use $B = 1$, $S = 4096$, and $n = 4$; the Transformer sublayer is excluded so that the comparison isolates the hyper-connection operators. As shown in Table 2, our implementation reduces total forward + backward latency by 44.8%, 31.0%, and 24.1% as D increases from $2,048$ to $7,168$. The benefit is largest at smaller hidden dimensions, where launch and intermediate-memory overhead constitute a larger share of execution time. Detailed Pre/Post breakdowns and comparison with an unfused MindSpore (MindSpore, 2022) implementation are reported in Appendix F.

See §E for more experiment results.

5 Conclusion

We study how manifold-constrained hyper-connections mix information across feature streams. We first reveal that residual matrices in traditional methods tend to stay close to the permutation vertices of the Birkhoff polytope, leading to under-mixing, while excessive mixing can drive streams toward a uniform state and suppress their differences. Motivated by this mixing trade-off, we propose cmHC, which geometrically contracts the Birkhoff polytope toward its interior, providing a provable bound on stream mixing without introducing any new parameterization. Our analysis bounds cumulative residual mixing, stream differences and gradient differences of mixing coefficients across layers, showing that the effect of contraction depends jointly on its strength and the number of contracted layers. This motivates a simple but effective strategy that applies a moderate contraction to shallow layers to prevent over-mixing. Our comparative experiments and analyses of training dynamics from GPT-2 to 6B- and 300B-MoE models validate the effectiveness and scalability of cmHC. We further develop fused operators for cmHC on Ascend NPUs, reducing hyper-connection execution latency by up to 44.8%, to further support industrial use.

AI USE STATEMENT

We used generative AI tools to assist with drafting and revising portions of the manuscript and to improve the clarity, organization, and readability of the text. We reviewed and edited the AI-assisted content to ensure that it accurately reflects our methods, theoretical findings, and experimental results. We take full responsibility for the final content of this paper, including all claims, interpretations, and any material produced with the assistance of generative AI.

REFERENCES

Ekaterina Alimaskina, Gleb Molodtsov, and Aleksandr Beznosikov. Analyzing stream collapse in hyper-connections: From diagnosis to mitigation. In *ICML 2026 Workshop on Weight-Space Symmetries: from Foundations to Practical Applications*, 2026.

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

Garrett Birkhoff. Three observations on linear algebra. *Univ. Nac. Tacuman, Rev. Ser. A*, 5:147–151, 1946.

Torque Dandachi and Sophia Diggs-Galligan. go-mHC: Direct parameterization of manifold-constrained hyper-connections via generalized orthostochastic matrices. *arXiv:2604.02309*, 2026.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv:2405.04434*, 2024.

Stephane Gaubert and Zheng Qu. Dobrushin's ergodicity coefficient for Markov operators on cones. *Integral Equations and Operator Theory*, 81(1):127–150, 2015.

Aaron Gokaslan and Vanya Cohen. OpenWebText corpus, 2019. URL http://Skylion007.github.io/OpenWebTextCorpus.

Yuxuan Gu, Wuyang Zhou, Huijun Xing, and Danilo Mandic. TEMPER: Tensorized efficient manifold-constrained parameterization for expressive residual routing. *arXiv:2608.07851*, 2026.

Haoqiang Guo, Xuyi Chen, Bo Ke, Yishu Lei, Ziyang Xu, Shikun Feng, Ximen, and Wenhan Luo. oHC: Orthogonal hyper-connections on SO(4) via quaternions. *arXiv:2609.02672*, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Huawei Technologies Co., Ltd. About Ascend C. CANN Documentation, https://www.hiascend.com/document/detail/en/canncommercial/850/opdevg/Ascendcopdevg/atlas_ascendc_map_10_0002.html, 2025.

Huawei Technologies Co., Ltd. CANN 9.1.0 Documentation. Ascend Developer Zone, https://www.hiascend.com/document/detail/zh/CANNCommunityEdition/910/index/index.html, 2026.

Andrej Karpathy. nanochat: The best ChatGPT that $100 can buy, 2025. URL https://github.com/karpathy/nanochat.

Zhaoyi Liu, Haichuan Zhang, and Ang Li. Beyond the Birkhoff polytope: Spectral-sphere-constrained hyper-connections. *arXiv:2603.20896*, 2026.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-Edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Anton Lyubinin. TBP-mHC: full expressivity for manifold-constrained hyper connections through transportation polytopes. *arXiv:2605.21724*, 2026.

MindSpore. MindSpore technical white paper. MindSpore Documentation, https://www.mindspore.cn/docs/en/r1.7/design/technical_white_paper.html, 2022. Accessed: 2026-08-27.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *ICLR*, 2024.

William Peng, Josheev Rai, Kevin Tseng, Siwei Wang, and Sean Wu. Ablate and rescue: A causal analysis of residual stream hyper-connections. *arXiv:2603.14833*, 2026.

Biswa Sengupta, Jinhua Wang, and Leo Brunswic. JPmHC dynamical isometry via orthogonal hyper-connections. *arXiv:2602.18308*, 2026.

Noam Shazeer. GLU variants improve Transformer. *arXiv:2002.05202*, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

John von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games (AM-28), Volume II*, 2:5–12, 1953.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Mingyu Xu, Kuai Yu, et al. mHC: Manifold-constrained hyper-connections. In *ICML*, 2026.

Yongyi Yang and Jianyang Gao. mHC-lite: You don't need 20 Sinkhorn-Knopp iterations. In *ICLR 2026 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2026.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, pp. 3320–3328, 2014.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, pp. 12381–12392, 2019.

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

Wuyang Zhou, Yuxuan Gu, Giorgos Iacovides, and Danilo Mandic. KromHC: Manifold-constrained hyper-connections with Kronecker-product residual matrices. In *ICML*, 2026.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *ICLR*, 2025.

APPENDIX CONTENTS

A Preliminary **13**

A.1 Manifold-Constrained Hyper-Connections (mHC) 13

A.2 mHC-lite: Birkhoff-von Neumann Parameterization 13

A.3 Contraction Mapping and the Uniform Fixed Point 13

B Related Work **14**

C Proofs **15**

C.1 Proof of Theorem 2.1 15

C.2 Proof of Theorem 2.2 15

C.3 Proof of Theorem 3.1 16

C.4 Proof of Theorem 3.2 16

C.5 Proof of Theorem 3.3 17

D Experiment Setup **18**

D.1 Experiment Setup for Large-Scale MoEs 18

E Full Results **19**

E.1 Training Loss and Gradient Norm on GPT-2 19

E.2 Visualization of Residual Matrices 19

E.3 Smoothness of the Mixing Coefficient 21

E.4 Scaling to More Streams 22

F Implementation Details of the Fused Operator **22**

F.1 Architecture-Aware Fusion Principles 22

F.2 Fused Pre Forward Kernel 23

F.3 Fused Pre Backward Kernel and State Reuse 24

F.4 Fused Post Forward and Backward Kernels 25

F.5 Operator-Level Performance 25

A PRELIMINARY

A.1 MANIFOLD-CONSTRAINED HYPER-CONNECTIONS (mHC)

To resolve the training instability of HC caused by the unconstrained residual matrix $\mathbf{H}^{\text{res}}$, DeepSeek proposed Manifold-Constrained Hyper-Connections (mHC) (Xie et al., 2026) which constrains $\mathbf{H}^{\text{res}}$ to the space of doubly stochastic matrices, known as the *Birkhoff polytope* $\mathcal{B}_n$. A matrix $\mathbf{U} \in \mathbb{R}^{n\times n}$ is doubly stochastic if it satisfies:

$$u_{ij} \geq 0 \; \forall i, j \in \{1, \ldots, n\}, \quad \sum_{i=1}^{n} u_{ij} = 1 \; \forall i, \quad \sum_{j=1}^{n} u_{ij} = 1 \; \forall j. \tag{14}$$

By forcing $\mathbf{H}^{\text{res}}$ to be doubly stochastic, the mixing between feature streams becomes a convex combination. As the spectral norm of a doubly stochastic matrix is bounded by 1, it prevents signal amplification and guarantees the mathematical preservation of the identity mapping. In standard mHC, projection onto $\mathcal{B}_n$ is achieved via the iterative Sinkhorn-Knopp algorithm:

$$\mathbf{H}^{\text{res}}_{(0)} = \exp(\mathbf{S}), \quad \mathbf{H}^{\text{res}}_{(t+1)} = \text{Diag}\left(\mathbf{H}^{\text{res}}_{(t)}\mathbf{1}_n\right)^{-1} \mathbf{H}^{\text{res}}_{(t)} \text{Diag}\left(\mathbf{1}_n^\top \mathbf{H}^{\text{res}}_{(t)}\right)^{-1}, \tag{15}$$

where $\mathbf{S} \in \mathbb{R}^{n\times n}$ is a learned logit matrix.

The components of a mHC block can be formalized as

$$\begin{cases} \mathbf{x}'_l = \text{RMSNorm}(\text{flatten}(\mathbf{x}_l)), \\ \mathbf{H}^{\text{pre}}_l = \sigma\left(a^{\text{pre}}_l \mathbf{x}'_l \mathbf{W}^{\text{pre}}_l + \mathbf{b}^{\text{pre}}_l\right), \\ \mathbf{H}^{\text{post}}_l = 2\sigma\left(a^{\text{post}}_l \mathbf{x}'_l \mathbf{W}^{\text{post}}_l + \mathbf{b}^{\text{post}}_l\right), \\ \mathbf{H}^{\text{res}}_l = \text{Sinkhorn-Knopp}\left(a^{\text{res}}_l \text{mat}\left(\mathbf{x}'_l \mathbf{W}^{\text{res}}_l\right) + \mathbf{b}^{\text{res}}_l\right), \end{cases} \tag{16}$$

where $\mathbf{W}^{\text{pre}}_l, \mathbf{W}^{\text{post}}_l \in \mathbb{R}^{nd\times n}$, $\mathbf{W}^{\text{res}}_l \in \mathbb{R}^{nd\times n^2}$ and $\mathbf{b}^{\text{pre}}_l, \mathbf{b}^{\text{post}}_l \in \mathbb{R}^{1\times n}$, $\mathbf{b}^{\text{res}}_l \in \mathbb{R}^{1\times n^2}$ are linear projection weights and biases, respectively; $a^{\text{pre}}_l, a^{\text{post}}_l$ and a^{res}_l are learnable scalars; $\text{mat}: \mathbb{R}^{1\times n^2} \to \mathbb{R}^{n\times n}$ is a reshape map; σ is the Sigmoid function.

A.2 MHC-LITE: BIRKHOFF-VON NEUMANN PARAMETERIZATION

A known issue of mHC's Sinkhorn-Knopp algorithm is that it does not guarantee exact double stochasticity. As it requires multiple iterations (typically $t \geq 10$ or 20) to converge, the column sum of $\mathbf{H}^{\text{res}}_l$ may deviate from 1 under insufficient iterations. To address this, mHC-lite (Yang & Gao, 2026) learns a direct convex combination of a set of pre-defined permutation matrices $\mathbf{P}_m$:

$$\mathbf{H}^{\text{res}}_l = \sum_{m=1}^{n!} \alpha_{l,m} \mathbf{P}_m, \quad \boldsymbol{\alpha}_l = \text{softmax}(a^{\text{res}}_l \mathbf{x}'_l \mathbf{W}^{\text{res}}_l + \mathbf{b}^{\text{res}}_l), \tag{17}$$

and other components are identical to (16). This strategy guarantees exact double stochasticity because of the Birkhoff-von Neumann Theorem:

Theorem A.1 (Birkhoff-von Neumann (Birkhoff, 1946; von Neumann, 1953))**.** *Any doubly stochastic matrix $\mathbf{U} \in \mathcal{B}_n$ can be represented as a convex combination of permutation matrices:*

$$\mathbf{U} = \sum_{m=1}^{n!} \alpha_m \mathbf{P}_m, \tag{18}$$

where $\boldsymbol{\alpha} \in [0,1]^{n!}$ is a probabilistic simplex vector and $\{\mathbf{P}_m\}_{m=1}^{n!} \in \mathbb{R}^{n\times n}$ are permutation matrices.

A.3 CONTRACTION MAPPING AND THE UNIFORM FIXED POINT

We provide the formal definition of a contraction mapping and establish the fixed point and convergence rate of the geometric map used in ~~c~~mHC.

Definition A.1 (Contraction mapping). *Let $(\mathcal{M}, d)$ be a metric space. A map $c : \mathcal{M} \to \mathcal{M}$ is a contraction if there exists $0 \le k < 1$ such that*

$$d(c(u), c(v)) \le k\, d(u, v), \qquad \forall u, v \in \mathcal{M}. \tag{19}$$

Theorem A.2 (Banach fixed-point theorem (Banach, 1922)). *A contraction on a non-empty complete metric space has a unique fixed point. Starting from any point in the space, repeated application of the map converges to this fixed point.*

Applicability to the Birkhoff polytope. The Birkhoff polytope $\mathcal{B}_n$ is non-empty and closed in the finite-dimensional normed space $(\mathbb{R}^{n\times n}, \|\cdot\|_F)$, and is therefore complete. For a fixed $\lambda \in (1/n, 1)$, define

$$\mathbf{J}_n = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top, \qquad \lambda' = \frac{n\lambda - 1}{n - 1} \in (0, 1). \tag{20}$$

Equation (4) gives $c(\mathbf{H}; \lambda) = \lambda'\mathbf{H} + (1 - \lambda')\mathbf{J}_n$. Since $\mathbf{H}, \mathbf{J}_n \in \mathcal{B}_n$ and $\mathcal{B}_n$ is convex, this map sends $\mathcal{B}_n$ into itself. Moreover,

$$\|c(\mathbf{H}; \lambda) - c(\mathbf{G}; \lambda)\|_F = \lambda'\|\mathbf{H} - \mathbf{G}\|_F. \tag{21}$$

Thus, all conditions of Banach's fixed-point theorem hold.

Unique fixed point. The fixed point $\mathbf{H}_*$ satisfies

$$\mathbf{H}_* = \lambda'\mathbf{H}_* + (1 - \lambda')\mathbf{J}_n. \tag{22}$$

Since $1 - \lambda' > 0$, this implies $\mathbf{H}_* = \mathbf{J}_n$. Therefore, the unique fixed point is the uniform barycenter of the Birkhoff polytope.

Convergence under repeated contraction. Let $\mathbf{H}^{(0)} \in \mathcal{B}_n$ and define $\mathbf{H}^{(t+1)} = c(\mathbf{H}^{(t)}; \lambda)$. Subtracting $\mathbf{J}_n$ gives

$$\mathbf{H}^{(t+1)} - \mathbf{J}_n = \lambda'(\mathbf{H}^{(t)} - \mathbf{J}_n). \tag{23}$$

By induction,

$$\mathbf{H}^{(t)} = \mathbf{J}_n + (\lambda')^t(\mathbf{H}^{(0)} - \mathbf{J}_n), \qquad \|\mathbf{H}^{(t)} - \mathbf{J}_n\|_F = (\lambda')^t\|\mathbf{H}^{(0)} - \mathbf{J}_n\|_F. \tag{24}$$

Hence, repeated contraction converges geometrically to $\mathbf{J}_n$, with stronger contraction (smaller λ, hence smaller λ') yielding faster convergence. At this limit, every output stream receives the same uniform mixture of input streams.

B Related Work

Multi-stream residual connections. Hyper-Connections (Zhu et al., 2025) extend residual connections (He et al., 2016) with learnable aggregation, mixing, and redistribution across multiple streams. To stabilize residual propagation, `mHC` (Xie et al., 2026) constrains residual mixing to doubly stochastic matrices through Sinkhorn normalization, while `mHC-lite` (Yang & Gao, 2026) uses convex combinations of permutation matrices for an exact doubly stochastic constraint. There are also improvements focusing on the efficiency of computing and scalability on more streams, such as Kronecker decomposition (Zhou et al., 2026), tensorization (Gu et al., 2026) and sparse stream update (Zhang et al., 2026). We discuss the mixing effectiveness of manifold-constrained hyper-connections, as existing research has revealed their under-mixing phenomenon, also referred to as "stream collapse" (Alimaskina et al., 2026) or "identity degradation" (Liu et al., 2026); this harms the expressivity of multi-stream design.

Exploration of the Birkhoff polytope. A line of recent work explores *expansions* of the Birkhoff polytope to address under-mixing. For example, Liu et al. (2026) constrain the residual matrices into a spectral sphere which expands the doubly stochastic constraint to the negative domain. Guo et al. (2026) constrain residual matrices to $SO(n)$ rotation groups and parameterizes $SO(4)$ using pairs of unit quaternions. However, such expansions cannot strictly guarantee a positive mixing effect inside the Birkhoff polytope without provable stream mixing bounds. Another line of work focuses on a wider feasible region inside the Birkhoff polytope to encourage interior exploration. For example,

Alimaskina et al. (2026) recommend a non-symmetric initialization to unbind residual matrices from the polytope vertices; Lyubinin (2026) covers the polytope with recursive transport construction; Dandachi & Diggs-Galligan (2026) and Sengupta et al. (2026) introduce the orthogonal constraint to expand the spectral reach and ensure the stability of the gradient spectrum of hyper-connections. However, these approaches often rely on more involved constructions and parameterizations, such as Cayley Transform and per-stream scaling vectors, while our approach introduces NO additional parameters and specialized constructions at all.

Moreover, the effectiveness of most existing approaches is not validated on extremely large-scale model pretraining. To our knowledge, the largest parameter scale used for research-based manifold-constrained hyper-connections is 28B by Zhang et al. (2026), while we conduct scalability tests on a 300B mixture-of-experts model to validate the effectiveness and training stability of our method.

Stream similarity and over-mixing. A recent empirical analysis (Peng et al., 2026) shows that high representational similarity between streams does not imply functional interchangeability, as the non-redundant part of streams can still play asymmetric roles to model predictions. This confirms the risk of over-mixing from another perspective: when streams become nearly identical to each other, the redundancy between streams can be amplified and the informative asymmetry can be weakened.

C Proofs

C.1 Proof of Theorem 2.1

Theorem 2.1 (Preservation of double stochasticity). *For $\lambda \in (1/n, 1)$, $\mathcal{B}_n(\lambda) \subseteq \mathcal{B}_n$. In particular, every contracted residual matrix is doubly stochastic and has spectral norm at most one.*

Proof. By the linearity of the convex combination, we have

$$\mathcal{c}(\mathbf{H}_l^{\text{res}}; \lambda) = \sum_{m=1}^{n!} \alpha_{l,m} \left(\lambda \mathbf{P}_m + \frac{1-\lambda}{n-1} (\mathbf{1}_n \mathbf{1}_n^\top - \mathbf{P}_m) \right) \tag{25}$$

$$= \lambda \left(\sum_{m=1}^{n!} \alpha_{l,m} \mathbf{P}_m \right) + \frac{1-\lambda}{n-1} \left(\mathbf{1}_n \mathbf{1}_n^\top - \sum_{m=1}^{n!} \alpha_{l,m} \mathbf{P}_m \right) \tag{26}$$

$$= \lambda \mathbf{H}_l^{\text{res}} + \frac{1-\lambda}{n-1} (\mathbf{1}_n \mathbf{1}_n^\top - \mathbf{H}_l^{\text{res}}) \tag{27}$$

for any doubly stochastic matrix $\mathbf{H}_l^{\text{res}}$. As $\mathbf{H}_l^{\text{res}} \mathbf{1}_n = \mathbf{1}_n$, $\mathbf{1}_n^\top \mathbf{H}_l^{\text{res}} = \mathbf{1}_n^\top$, we have

$$\mathcal{c}(\mathbf{H}_l^{\text{res}}; \lambda) \mathbf{1}_n = \lambda \mathbf{H}_l^{\text{res}} \mathbf{1}_n + \frac{1-\lambda}{n-1} (\mathbf{1}_n \mathbf{1}_n^\top - \mathbf{H}_l^{\text{res}}) \mathbf{1}_n = \left(\lambda - \frac{1-\lambda}{n-1} + \frac{n(1-\lambda)}{n-1} \right) \mathbf{1}_n = \mathbf{1}_n, \tag{28}$$

$$\mathbf{1}_n^\top \mathcal{c}(\mathbf{H}_l^{\text{res}}; \lambda) = \mathbf{1}_n^\top \lambda \mathbf{H}_l^{\text{res}} + \frac{1-\lambda}{n-1} \mathbf{1}_n^\top (\mathbf{1}_n \mathbf{1}_n^\top - \mathbf{H}_l^{\text{res}}) = \mathbf{1}_n^\top \left(\lambda - \frac{1-\lambda}{n-1} + \frac{n(1-\lambda)}{n-1} \right) = \mathbf{1}_n^\top. \tag{29}$$

So $\mathcal{c}(\mathbf{H}_l^{\text{res}}; \lambda)$ is doubly stochastic and thus $\mathcal{B}_n(\lambda) \subseteq \mathcal{B}_n$. For any $\mathbf{H}_l^{\text{res}'} \in \mathcal{B}_n(\lambda)$, $\|\mathbf{H}_l^{\text{res}'}\|_1 = \|\mathbf{H}_l^{\text{res}'}\|_\infty = 1$, so the spectral norm satisfies $\|\mathbf{H}_l^{\text{res}'}\| \leq \sqrt{\|\mathbf{H}_l^{\text{res}'}\|_1 \|\mathbf{H}_l^{\text{res}'}\|_\infty} = 1$.

□

C.2 Proof of Theorem 2.2

Theorem 2.2 (Lower bound of stream mixing). *For $\lambda \in (1/n, 1)$, every entry of a contracted residual matrix satisfies*

$$h_{l,ij}^{\text{res}'} \geq \min \left(\lambda, \frac{1-\lambda}{n-1} \right) > 0. \tag{30}$$

Proof. For each entry of $\mathbf{H}_l^{\text{res}'}$, we have

$$h_{l,ij}^{\text{res}'} = \sum_{m=1}^{n!} \alpha_{l,m} \left(\lambda p_{m,ij} + \frac{1-\lambda}{n-1}(1 - p_{m,ij}) \right) \tag{31}$$

$$= \lambda \sum_{p_{m,ij}=1} \alpha_{l,m} + \frac{1-\lambda}{n-1} \sum_{p_{m,ij}=0} \alpha_{l,m} \tag{32}$$

$$= \lambda \sum_{p_{m,ij}=1} \alpha_{l,m} + \frac{1-\lambda}{n-1} \left(1 - \sum_{p_{m,ij}=1} \alpha_{l,m} \right). \tag{33}$$

As $0 \leq \sum_{p_{m,ij}=1} \alpha_{l,m} \leq 1$, $h_{l,ij}^{\text{res}'}$ is a convex combination of λ and $\frac{1-\lambda}{n-1}$, hence $h_{l,ij}^{\text{res}'} \geq \min\left(\lambda, \frac{1-\lambda}{n-1}\right)$. □

C.3 Proof of Theorem 3.1

Theorem 3.1 (Convergence of cumulative stream mixing). *Let $\lambda_l \in (1/n, 1]$ for $l = 0, \ldots, L-1$, where $\lambda_l = 1$ leaves layer l uncontracted. Define $\lambda_l' = (n\lambda_l - 1)/(n-1)$ and $\mathbf{H}_l^{\text{res}'} = \lambda_l' \mathbf{H}_l^{\text{res}} + (1 - \lambda_l')\mathbf{J}_n$. Then*

$$\tau\left(\prod_{l=0}^{L-1} \mathbf{H}_{L-1-l}^{\text{res}'} \right) \leq \prod_{l=0}^{L-1} \lambda_l'. \tag{34}$$

Proof. For $1/n < \lambda_l \leq 1$, the affine form of $\mathbf{H}_l^{\text{res}'}$ and the nonnegativity of $\mathbf{H}_l^{\text{res}}$ give

$$h_{l,ij}^{\text{res}'} = \lambda_l' h_{l,ij}^{\text{res}} + \frac{1-\lambda_l'}{n} \geq \frac{1-\lambda_l}{n-1} \geq 0. \tag{35}$$

This includes $\lambda_l = 1$, where the entrywise lower bound is zero and $\mathbf{H}_l^{\text{res}'} = \mathbf{H}_l^{\text{res}}$ remains doubly stochastic.

For every layer l, the entry-wise lower bound (35) gives

$$\sum_{k=1}^{n} \min(h_{l,ik}^{\text{res}'}, h_{l,jk}^{\text{res}'}) \geq \frac{n(1-\lambda_l)}{n-1} \quad \Rightarrow \quad \tau(\mathbf{H}_l^{\text{res}'}) \leq 1 - \frac{n(1-\lambda_l)}{n-1} = \lambda_l' \leq 1. \tag{36}$$

Dobrushin's coefficient is sub-multiplicative ($\tau(\mathbf{AB}) \leq \tau(\mathbf{A})\tau(\mathbf{B})$), so

$$\tau\left(\prod_{l=0}^{L-1} \mathbf{H}_{L-1-l}^{\text{res}'} \right) \leq \prod_{l=0}^{L-1} \tau(\mathbf{H}_l^{\text{res}'}) \leq \prod_{l=0}^{L-1} \lambda_l'. \tag{37}$$

□

C.4 Proof of Theorem 3.2

Theorem 3.2 (Norm upper bound of stream difference). *For $L \geq 2$ hyper-connection layers, the stream difference norm satisfies*

$$\|\tilde{\mathbf{x}}_k\|_F \leq \left(\prod_{l=1}^{k-1} (\lambda_l' + \ell_l) \right) \|\tilde{\mathbf{x}}_1\|_F \tag{38}$$

for $2 \leq k \leq L$, where $\lambda_l' = \frac{n\lambda_l - 1}{n-1}$ is the normalized contraction coefficient and $\ell_l = \sup_{\mathbf{Z} \neq \mathbf{0}} \frac{\|\tilde{\mathbf{A}}_l \mathbf{Z}\|_F}{\|\mathbf{Z}\|_F}$ is the Lipschitz constant of $\tilde{\mathbf{A}}_l = \mathbf{\Pi}_n \mathbf{A}_l$.

Proof. Since $\mathbf{H}_l^{\text{res}'}$ is doubly stochastic, $\mathbf{\Pi}_n \mathbf{H}_l^{\text{res}'} = \mathbf{H}_l^{\text{res}'} - \mathbf{J}_n = \mathbf{H}_l^{\text{res}'} \mathbf{\Pi}_n$. Applying $\mathbf{\Pi}_n$ to (7) and substituting (4) gives

$$\tilde{\mathbf{x}}_{l+1} \approx \mathbf{H}_l^{\text{res}'} \tilde{\mathbf{x}}_l + \mathbf{\Pi}_n \mathbf{A}_l \mathbf{x}_l \tag{39}$$

$$= \lambda_l' \mathbf{H}_l^{\text{res}} \tilde{\mathbf{x}}_l + \tilde{\mathbf{A}}_l \tilde{\mathbf{x}}_l + \mathbf{\Pi}_n \mathbf{A}_l \mathbf{J}_n \mathbf{x}_l \tag{40}$$

$$\approx (\lambda_l' \mathbf{H}_l^{\text{res}} + \tilde{\mathbf{A}}_l) \tilde{\mathbf{x}}_l, \tag{41}$$

where $\tilde{\mathbf{A}}_l = \mathbf{\Pi}_n \mathbf{A}_l$ and $\lambda_l' = \frac{n\lambda_l - 1}{n-1} \in (0, 1)$. For $c\texttt{mHC}$ with $L \geq 2$ hyper-connection layers, the stream difference for any output layer index $2 \leq k \leq L$ is

$$\tilde{\mathbf{x}}_k \approx \left(\prod_{l=1}^{k-1} (\lambda_{k-l}' \mathbf{H}_{k-l}^{\text{res}} + \tilde{\mathbf{A}}_{k-l}) \right) \tilde{\mathbf{x}}_1. \tag{42}$$

As $\tilde{\mathbf{A}}_l$ is a finite-dimensional linear operator, there exists a Lipschitz constant $\ell_l = \sup_{\mathbf{Z} \neq \mathbf{0}} \frac{\|\tilde{\mathbf{A}}_l \mathbf{Z}\|_F}{\|\mathbf{Z}\|_F}$ for every layer l which satisfies $\|\tilde{\mathbf{A}}_l \mathbf{Z}\|_F \leq \ell_l \|\mathbf{Z}\|_F$ for $\forall \mathbf{Z} \in \mathbb{R}^{n \times d}$. Since $\|\mathbf{H}_l^{\text{res}}\|_2 \leq 1$, the approximate recurrence in (39) yields the following bound:

$$\|\tilde{\mathbf{x}}_{l+1}\|_F \leq \|\lambda_l' \mathbf{H}_l^{\text{res}} \tilde{\mathbf{x}}_l\|_F + \|\tilde{\mathbf{A}}_l \tilde{\mathbf{x}}_l\|_F \leq (\lambda_l' + \ell_l) \|\tilde{\mathbf{x}}_l\|_F \tag{43}$$

$$\Rightarrow \quad \|\tilde{\mathbf{x}}_k\|_F \leq \left(\prod_{l=1}^{k-1} (\lambda_l' + \ell_l) \right) \|\tilde{\mathbf{x}}_1\|_F. \tag{44}$$

□

C.5 Proof of Theorem 3.3

Theorem 3.3 (Gradient difference of different permutations). *For two different permutation matrices $\mathbf{P}_i, \mathbf{P}_j \in \mathbb{R}^{n \times n}$, the gradient difference of the pretraining loss $\mathcal{L}$ w.r.t. their coefficients $\alpha_{l,i}, \alpha_{l,j}$ is bounded by*

$$|(\nabla_{\alpha_{l,i}} - \nabla_{\alpha_{l,j}})\mathcal{L}| \leq 2\lambda_l' \|\nabla_{\tilde{\mathbf{x}}_{l+1}} \mathcal{L}\|_F \|\tilde{\mathbf{x}}_l\|_F. \tag{45}$$

Proof. From (39), the gradient of the stream difference at layer l is

$$\nabla_{\tilde{\mathbf{x}}_l} \mathcal{L} = (\lambda_l' \mathbf{H}_l^{\text{res}} + \tilde{\mathbf{A}}_l)^\top \nabla_{\tilde{\mathbf{x}}_{l+1}} \mathcal{L} \quad \Rightarrow \quad \nabla_{\tilde{\mathbf{x}}_{l+1}} \mathcal{L} = \left(\prod_{k=l+1}^{L-1} (\lambda_k' \mathbf{H}_k^{\text{res}} + \tilde{\mathbf{A}}_k) \right)^\top \nabla_{\tilde{\mathbf{x}}_L} \mathcal{L}. \tag{46}$$

This shows that all later contractions λ_k' with $k > l$ affect the gradient received by layer l.

We then analyze the update of $\boldsymbol{\alpha}$. From (2) and (39), we have

$$\frac{\partial \tilde{\mathbf{x}}_{l+1}}{\partial \alpha_{l,m}} = \lambda_l' \frac{\partial \mathbf{H}_l^{\text{res}}}{\partial \alpha_{l,m}} \tilde{\mathbf{x}}_l + \underbrace{\frac{\partial \tilde{\mathbf{A}}_l}{\partial \alpha_{l,m}}}_{0} \tilde{\mathbf{x}}_l = \lambda_l' \mathbf{P}_m \tilde{\mathbf{x}}_l. \tag{47}$$

By the multivariate chain rule,

$$\frac{\partial \mathcal{L}}{\partial \alpha_{l,m}} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial (\tilde{\mathbf{x}}_{l+1})_{ij}} \frac{\partial (\tilde{\mathbf{x}}_{l+1})_{ij}}{\partial \alpha_{l,m}} = \lambda_l' \sum_{i,j} \left(\nabla_{\tilde{\mathbf{x}}_{l+1}} \mathcal{L} \right)_{ij} \left(\mathbf{P}_m \tilde{\mathbf{x}}_l \right)_{ij}, \tag{48}$$

and therefore

$$\nabla_{\alpha_{l,m}} \mathcal{L} = \lambda_l' \left\langle \nabla_{\tilde{\mathbf{x}}_{l+1}} \mathcal{L},\ \mathbf{P}_m \tilde{\mathbf{x}}_l \right\rangle_F, \tag{49}$$

where $\langle \mathbf{U}, \mathbf{V} \rangle_F := \sum_{i,j} u_{ij} v_{ij}$ is the Frobenius inner product. Substituting (46) and (42) into (49), we obtain

$$\nabla_{\alpha_{l,m}} \mathcal{L} = \lambda_l' \left\langle \overbrace{\left(\prod_{k=l+1}^{L-1} (\lambda_k' \mathbf{H}_k^{\text{res}} + \tilde{\mathbf{A}}_k) \right)^\top \nabla_{\tilde{\mathbf{x}}_L} \mathcal{L}}^{\lambda_{>l}' \text{ part: backward constraint}},\ \mathbf{P}_m \overbrace{\left(\prod_{k=1}^{l-1} (\lambda_{k-l}' \mathbf{H}_{k-l}^{\text{res}} + \tilde{\mathbf{A}}_{k-l}) \right) \tilde{\mathbf{x}}_1}^{\lambda_{<l}' \text{ part: forward constraint}} \right\rangle_F. \tag{50}$$

This indicates that the update of $\alpha_{l,m}$ is affected by three factors: (1) the contraction at the current layer λ'_l; (2) the contractions in deeper layers $\{\lambda'_{>l}\}$ introduced by backpropagation; and (3) the contractions in earlier layers $\{\lambda'_{<l}\}$ introduced by the forward process.

Now we compute the gradient difference between any α-pair permutation matrices. For two different permutation matrices $\mathbf{P}_i$ and $\mathbf{P}_j$, we have

$$(\nabla_{\alpha_{l,i}} - \nabla_{\alpha_{l,j}})\mathcal{L} = \lambda'_l \left(\langle \nabla_{\tilde{\mathbf{x}}_{l+1}}\mathcal{L},\ \mathbf{P}_i\tilde{\mathbf{x}}_l\rangle_F - \langle \nabla_{\tilde{\mathbf{x}}_{l+1}}\mathcal{L},\ \mathbf{P}_j\tilde{\mathbf{x}}_l\rangle_F\right) \tag{51}$$

$$= \lambda'_l \left\langle \nabla_{\tilde{\mathbf{x}}_{l+1}}\mathcal{L},\ (\mathbf{P}_i - \mathbf{P}_j)\tilde{\mathbf{x}}_l\right\rangle_F. \tag{52}$$

By the Cauchy-Schwarz inequality,

$$|(\nabla_{\alpha_{l,i}} - \nabla_{\alpha_{l,j}})\mathcal{L}| \le \lambda'_l \|\nabla_{\tilde{\mathbf{x}}_{l+1}}\mathcal{L}\|_F \|(\mathbf{P}_i - \mathbf{P}_j)\tilde{\mathbf{x}}_l\|_F. \tag{53}$$

Since a permutation matrix is orthogonal, left multiplication by it preserves the Frobenius norm. With $\|\mathbf{P}_i - \mathbf{P}_j\|_2 \le \|\mathbf{P}_i\|_2 + \|\mathbf{P}_j\|_2 = 2$, we reach the final result:

$$|(\nabla_{\alpha_{l,i}} - \nabla_{\alpha_{l,j}})\mathcal{L}| \le 2\lambda'_l \|\nabla_{\tilde{\mathbf{x}}_{l+1}}\mathcal{L}\|_F \|\tilde{\mathbf{x}}_l\|_F. \tag{54}$$

□

D Experiment Setup

Datasets. OpenWebText (Gokaslan & Cohen, 2019) is built from Reddit-linked webpages, and FineWeb-Edu (Lozhkov et al., 2024) is an education-oriented subset of FineWeb. For OpenWebText, we first tokenize it with GPT-2 BPE tokenizer and use a small validation split; this gives ∼9.04B training tokens and ∼4.43M validation tokens after preprocessing. For FineWeb-Edu, we use the "sample-10BT" split which contains roughly 10B GPT-2 tokens, and construct a validation split following the same pipeline. This gives ∼9.90B training tokens and ∼49.10M validation tokens.

Hardware and hyperparameters. The nanoGPT experiments are conducted on 8× 80GB NVIDIA H100 GPUs. For these comparative experiments, all models are pretrained for 10k iterations. For a fair comparison, parameter initializations and nanoGPT-related hyperparameters of `cmHC` strictly follow the configurations in `mHC-lite` (Yang & Gao, 2026).

D.1 Experiment Setup for Large-Scale MoEs

Architecture. Table 3 summarizes the 6B and 300B configurations. Here, depth counts Transformer blocks; each block contains separate attention and feed-forward hyper-connections. Both models use bias-free linear projections, SwiGLU feed-forward networks (Shazeer, 2020), RMSNorm (Zhang & Sennrich, 2019) with $\epsilon = 10^{-6}$, query/key normalization, and zero attention and hidden dropout. Both use rotary position embeddings (Su et al., 2024) with base 10,000; the 300B command explicitly selects YaRN (Peng et al., 2024) with both magnitude scaling factors set to 1. The 6B configuration additionally enables one multi-token prediction (MTP) layer with MTP auxiliary loss weight 0.1 and hyper-connections disabled.

Routing. Both configurations use sigmoid routing scores, expert bias updates, and sequence-level auxiliary load-balancing loss with coefficient 10^{-3}. The expert bias update rates are 10^{-4} for 6B and 10^{-3} for 300B, and the scaling factors for routed outputs are 2.0 and 2.5, respectively.

Hyper-connection settings. Both runs enable `cmHC` with $n = 4$ residual streams. In the 6B implementation, we use $\lambda = 0.97$ and $K = 4$. For 300B, the run uses $\lambda = 0.96$ and $K = 6$. For `mHC`, both configurations specify 20 Sinkhorn iterations.

Data and optimization. Both runs use tokenized FineWeb-Edu data. Table 4 reports training steps. Token counts are computed as completed steps multiplied by global batch size and sequence length; they count processed token positions.

Table 3: Architecture settings for the large-scale experiments. Expert widths are intermediate dimensions. Model scales follow the experiment designations.

Setting	6B-MoE	300B-MoE
Transformer blocks	28	48
Hidden size	2,048	7,168
Dense feed-forward width	5,632	18,432
Initial dense blocks	2	3
MoE layers	26	45
Attention heads	16	128
Query low-rank dimension	512	1,536
Key/value low-rank dimension	256	512
Non-rotary query/key head dimension	128	128
Rotary query/key head dimension	64	64
Value head dimension	128	128
Num of experts	64	256
Selected routed experts per token	4	6
Routed expert width	512	1,152
Shared expert intermediate width (total)	1,024	1,152
Vocabulary size	131,072	131,072
Context length	4,096	4,096
Residual streams	4	4

Table 4: Training settings and completed training lengths. Token counts use the completed steps. A dash denotes a value not specified in the supplied configuration.

Setting	6B-MoE	300B-MoE
Optimizer	AdamW	Muon
Learning rate after warmup	10^{-5}	2×10^{-4}
Warmup steps	2,000	200
Schedule after warmup	Constant	Constant
Configured training steps	71,500	2,000
Global batch size	512	4,096
Micro-batch size	1	1
Tokens per optimization step	2,097,152	16,777,216
Processed tokens	149.95B	33.55B
Gradient clipping norm	1.0	1.0
Optimizer β_1, β_2 settings	$(0.9, 0.95)$	$(0.9, 0.95)$
AdamW ϵ	10^{-8}	—
Main computation precision	BF16	BF16
Tensor/context parallel degree	1/1	1/1
Pipeline parallel degree	1	10
Expert parallel degree	8	8

E Full Results

E.1 Training Loss and Gradient Norm on GPT-2

Figure 8 shows full results for nanoGPT-M and nanoGPT-L, similar to Figure 6. The loss gap is measured relative to ~~cmHC~~, so a positive value indicates a higher loss than ours. ~~cmHC~~ maintains a lower training loss than all compared residual and hyper-connection baselines at both depths. The gradient norms indicate that ~~cmHC~~ preserves the training stability of mHC in deeper models.

E.2 Visualization of Residual Matrices

Figure 9 visualizes every residual matrix of the attention and MLP layers in ~~cmHC~~. Despite that the contraction is only applied to the first two layers, we can observe the dynamic stream mixing in later layers like layer 3 and 5. Deeper layers also participate in stream mixing, but most of them remain close to near-identity mappings. This behavior matches our expectation: shallow-layer mixing has

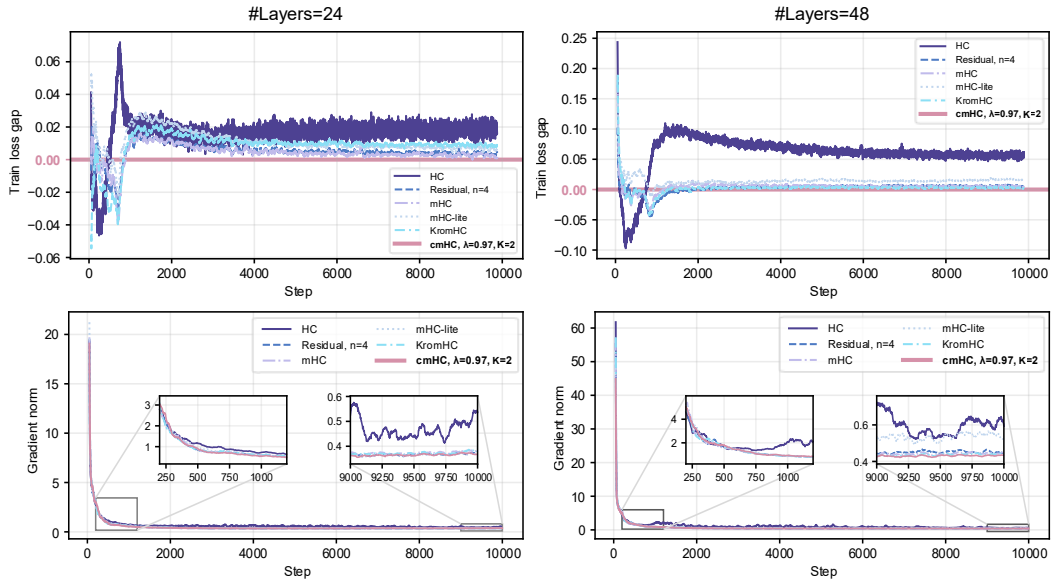


Figure 8: Loss gaps relative to cmHC and gradient norms (nanoGPT-M and nanoGPT-L; OpenWebText).

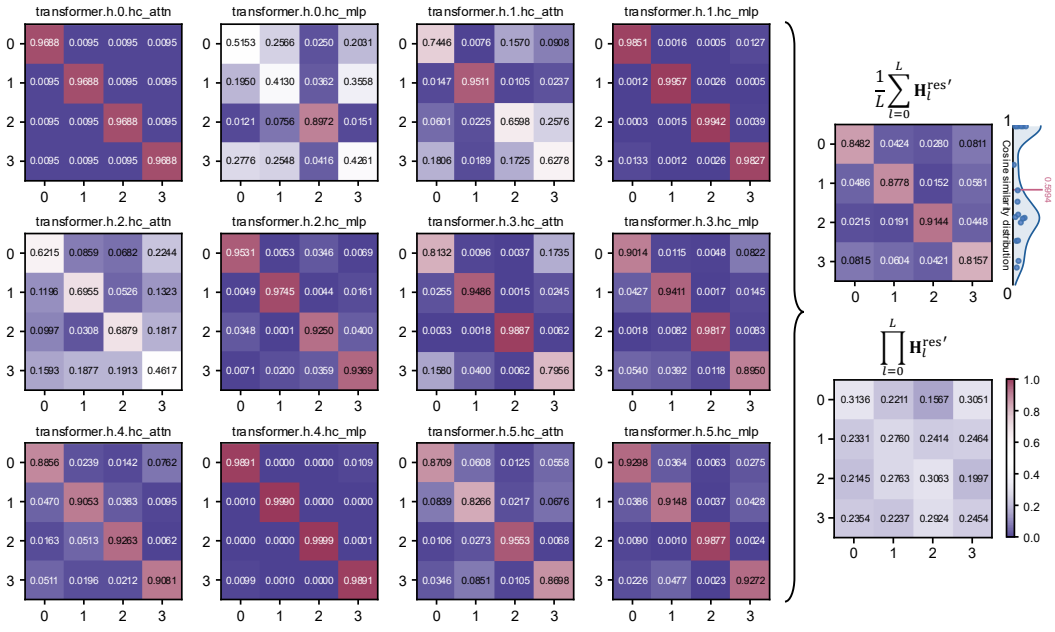


Figure 9: Full heatmaps of residual matrices (cmHC, $\lambda = 0.97$, $K = 2$; nanoGPT-L). *Left*: residual matrices $\mathbf{H}_l^{\text{res}}$ for every l. *Right*: average and product of residual matrices, and cosine-similarity distribution between streams.

a stronger cumulative effect on the network, while allowing later layers to stay near-identity helps prevent over-mixing. The aggregated views further support this point: as the average residual matrix is near identity, the product of residual matrices is already close to, but still distinguishable from, the uniform barycenter $\mathbf{J}_n$. This contrast between the average and product residual matrices illustrates the mixing trade-off: encouraging stream mixing while keeping the streams distinguishable.

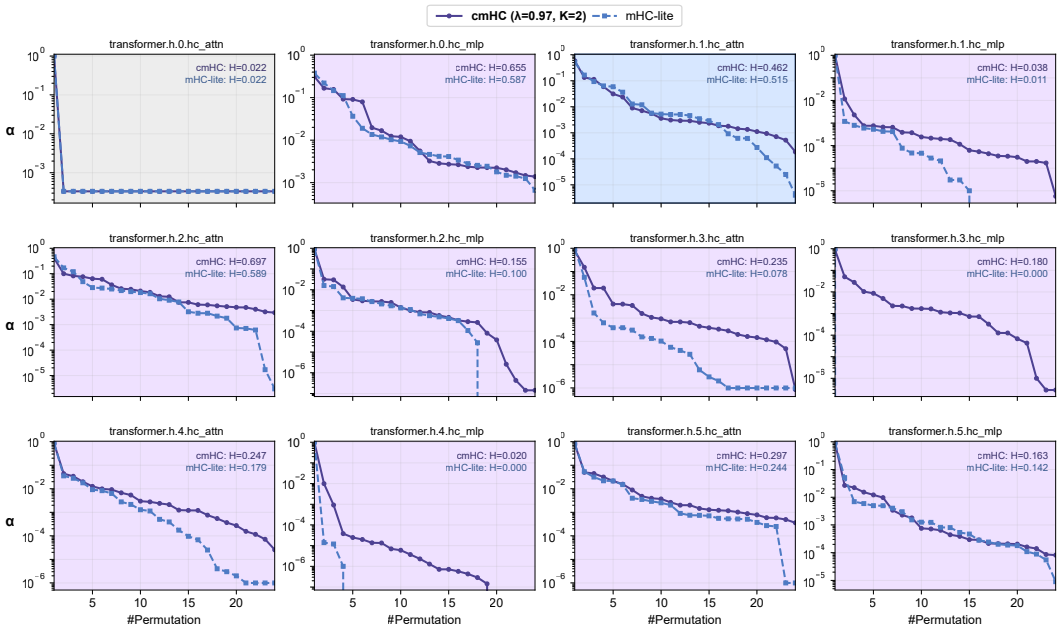


Figure 10: Full experiment results: distributions of mixing coefficients $\boldsymbol{\alpha}_l$ over the vertices of $\mathcal{B}_4$ for every l (nanoGPT-L). A purple (blue) background marks cases where the entropy $H(\boldsymbol{\alpha}_l)$ of cmHC is larger (smaller) than that of mHC-lite.

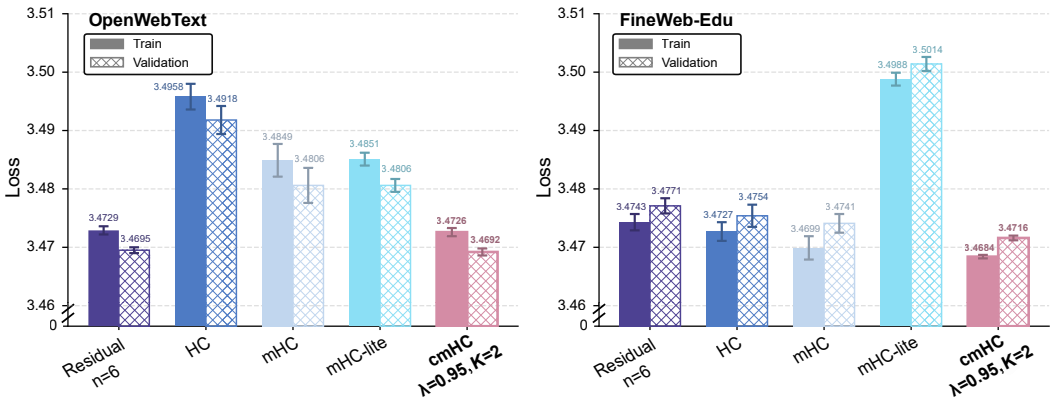


Figure 11: Comparative results with 6 residual streams (nanoGPT-S).

E.3 Smoothness of the Mixing Coefficient

This section shows the original full version of Figure 5. We further report the *normalized entropy* $H(\boldsymbol{\alpha}_l) \in [0, 1]$ to measure the smoothness of the distribution:

$$H(\boldsymbol{\alpha}_l) = \frac{-\sum_{m=1}^{n!} \alpha_{l,m} \log \alpha_{l,m}}{\log n!}, \tag{55}$$

where a larger value indicates a smoother distribution. Figure 10 shows the distributions of $\boldsymbol{\alpha}_l$ and their normalized entropy for every layer l. In the majority of layers, cmHC assigns more mass to the non-dominant vertices and achieves higher normalized entropy than mHC-lite, despite that only the first 2 layers are contracted. This supports our conclusion that even a numerically small geometric contraction can induce a smoother $\boldsymbol{\alpha}$ distribution in deeper layers.

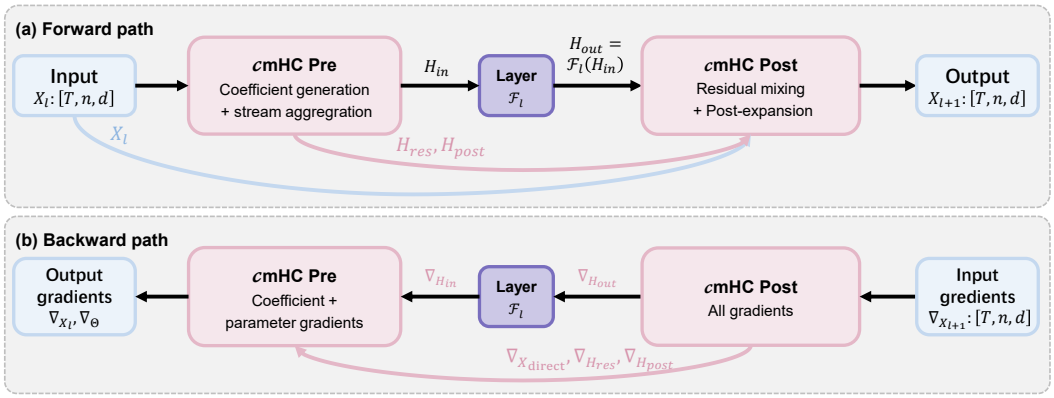


Figure 12: Fused-operator pipeline for cmHC. The backward path reverses the forward order and routes direct residual-path gradients from Post backward to Pre backward.

E.4 Scaling to More Streams

Figure 11 shows our experiments when scaling nanoGPT models to more hyper-connection streams. We use $\lambda = 0.95$ such that the non-diagonal entries in $\mathbf{P}'_m$ remain 0.01 like $\lambda = 0.97$ for 4 streams. cmHC keeps the superior performance on both datasets when streams are expanded to 6.

F Implementation Details of the Fused Operator

To reduce the execution overhead of cmHC, we develop an efficient fused-operator implementation. This section describes its design and implementation and compares its operator-level performance with the mHC fused-operator pipeline and an equivalent unfused primitive-operator implementation.

We implement the fused operators in Ascend C, a C/C++-compatible programming language and API system for developing custom kernels on Ascend NPUs (Huawei Technologies Co., Ltd., 2025). A cmHC layer places two fused operator boundaries around the model sublayer. The fused Pre operator produces the sublayer input and residual-path coefficients; the fused Post operator then performs residual mixing and post expansion. Training executes the corresponding operators in reverse order. The implementation contract is

$$(H_{\text{in}}, H_{\text{res}}, H_{\text{post}}) = \text{Pre}(X_l), \quad h_l^{\text{out}} = \mathcal{F}_l(H_{\text{in}}), \quad X_{l+1} = \text{Post}(X_l, H_{\text{res}}, h_l^{\text{out}}, H_{\text{post}}). \quad (56)$$

Here the capitalized names follow the operator-interface convention: $X_l \equiv \mathbf{x}_l$, $H_{\text{res}} \equiv \mathbf{H}_l^{\text{res}'}$, $H_{\text{in}} \equiv \mathbf{H}_l^{\text{pre}}\mathbf{x}_l$, and $H_{\text{post}} \equiv \mathbf{H}_l^{\text{post}}$. Thus these tensors are the implementation-level counterparts of the quantities defined in (1) and (3). The model sublayer $\mathcal{F}_l$ remains outside the fusion boundary because it may be an attention or feed-forward block. The resulting Pre and Post boundaries fuse only the surrounding connection logic, avoiding repeated reads of the widened residual state and launches of small framework operators. Figure 12 shows the forward and backward organization.

F.1 Architecture-Aware Fusion Principles

Let B, S, n, and D denote batch size, sequence length, residual-stream count, and per-stream hidden dimension, and let $T = BS$. The implementation specializes $n = 4$, while T and D are large. Accordingly, large projections use cube cores, whereas reductions, nonlinearities, coefficient transformations, and small-n stream mixing use vector cores. Table 5 summarizes this mapping.

Memory-layout and cache policies are selected by input shape. Small workloads retain a dense ND workspace, whereas sufficiently large T and D use a blocked NZ workspace at vector–cube handoffs. Large one-pass tensors selectively bypass L2, while weights, cached bases, and intermediates reused across stages remain cacheable. These conditional paths preserve useful locality for small inputs and avoid conversion and cache pollution at larger scales.

Table 5: Design principles and their mapping to the target architecture.

Design consideration	Kernel realization	Intended effect
Heterogeneous arithmetic	Dense projections on cube cores; reductions, nonlinearities, and scale-accumulate paths on vector cores	Match arithmetic granularity to the core type
Small stream dimension	Use vector scaling, accumulation, and reduction for $n = 4$ Post mixing	Avoid GEMM setup, padding, and format conversion for tiny matrices
Limited on-chip storage	Tile token and hidden dimensions; retain small coefficients and accumulators in Unified Buffer (UB)	Increase on-chip reuse within the UB capacity
Global-memory traffic	Fuse common-input operations; select ND/NZ workspace layouts at vector–cube handoffs; selectively bypass L2 for one-pass tensors	Reduce intermediate traffic, layout conversion, and cache pollution
Mixed-precision execution	Keep activations in BF16/FP16 and coefficients and accumulators in FP32	Reduce bandwidth while preserving coefficient accuracy
Launch and scheduling overhead	Produce all outputs of each Pre or Post direction in one launch	Amortize launch cost and avoid repeated state scans

Pre combines vector and cube cores because it contains both high-dimensional projections and coefficient transformations. Post uses vector cores only: its 4×4 coefficient matrix cannot amortize a cube-core schedule, whereas its D-wide scale–accumulate loops map directly to vector instructions.

F.2 Fused Pre Forward Kernel

For the Pre implementation, we use $D_{\text{flat}} = nD$, $R = n!$, and $Q = R + 2n$ with $n = 4$, $R = 24$ and $Q = 32$. Table 6 summarizes the tensors that determine its forward data path.

Table 6: Primary tensor interface of the fused mHC Pre forward operator.

Tensor	Role	Shape	Data type
X_l	Multi-stream input	$[B, S, n, D]$	BF16/FP16
Φ	Projection weight	$[Q, nD]$	FP32
s	Group-wise scale	$[3]$	FP32
b	Projection bias	$[Q]$	FP32
$\mathbf{P}'_l$	Contracted permutation matrix	$[R, n, n]$	FP32
H_{in}	Aggregated sublayer input	$[B, S, D]$	BF16/FP16
H_{post}	Post-mapping coefficient	$[B, S, n]$	FP32
H_{res}	Residual-mixing matrix	$[B, S, n, n]$	FP32

Let $\mathbf{P}_m$ denote the m-th permutation matrix and $\mathbf{J}_n = \mathbf{1}_n\mathbf{1}_n^\top/n$ the uniform doubly stochastic matrix. Before kernel launch, the framework uses the normalized contraction coefficient $\lambda'_l = (n\lambda_l - 1)/(n-1)$ from Sec. 2.2 to construct

$$\mathbf{P}'_{l,m} = \lambda'_l\mathbf{P}_m + (1 - \lambda'_l)\mathbf{J}_n, \qquad m = 1, \ldots, R, \tag{57}$$

which is algebraically identical to (3). Under the depth-aware strategy, $\lambda_l = 1$ (and hence $\lambda'_l = 1$) beyond the first K layers. The resulting basis tensor $\mathbf{P}'_l$ is cached by the framework and passed to the forward and backward Pre kernels; the fused kernels therefore receive the contracted bases rather than the scalar λ_l.

The operator exposes one entry and executes the three stages in Figure 13. Event synchronization and GM workspace enforce producer–consumer ordering between vector and cube cores.

Stage 1: projection and RMS statistics. The vector cores cast BF16/FP16 input tiles to FP32 and write them to workspace. The same scan evaluates

$$\rho = \left(\text{mean}(X_l^2) + \epsilon_{\text{norm}}\right)^{-1/2} \tag{58}$$

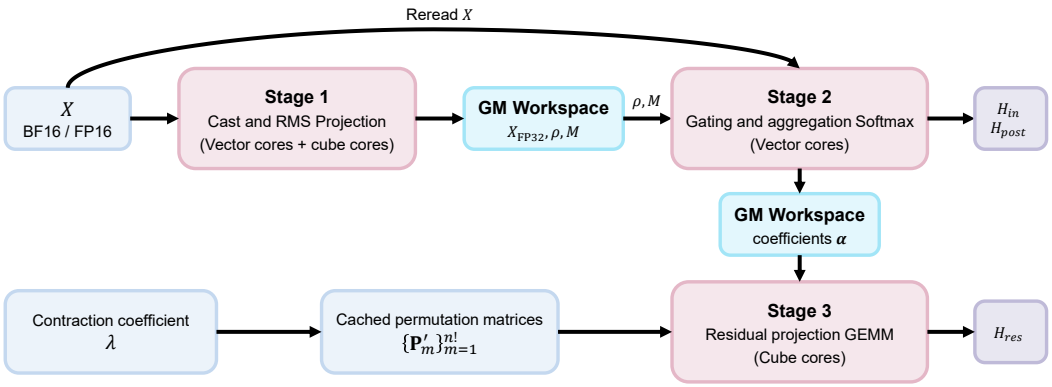


Figure 13: Three-stage data path of the fused mHC Pre forward operator, including framework-side preparation of the contracted bases.

for each token. The cube cores consume the converted workspace and compute

$$M = \mathrm{FP32}(X_l)\Phi^{\mathsf{T}}, \qquad M \in \mathbb{R}^{T\times Q}. \tag{59}$$

Cross-core events prevent the projection from reading incomplete workspace tiles.

For sufficiently large workloads, the vector cores transform each FP32 tile from ND to NZ in UB before writing the workspace. The cube cores then consume preformatted tiles through contiguous NZ transfers instead of converting ND data on the read path. Host tiling retains the ND path for smaller inputs, where the vector-side transformation cannot be amortized, and initializes only the selected matrix-multiplication implementation. The NZ workspace aligns the D_{flat} dimension to the hardware block size and preserves a fixed physical stride for tail tiles.

Stage 2: coefficient generation and stream aggregation. The vector cores apply group-wise scale and bias to M and ρ, generating the Pre, Post, and R-dimensional residual coefficients. A stable FP32 softmax processes the residual segment. The same stage rereads low-precision X_l and reduces its weighted streams into H_{in}, avoiding separate gating, softmax, and aggregation operators. For a token batch, the residual coefficients form $\boldsymbol{A} \in \mathbb{R}^{T\times R}$, whose rows are the coefficient vectors $\boldsymbol{\alpha}_l$ from (2). They enter workspace for Stage 3 and are optionally exported for backward reuse.

Stage 3: residual-matrix projection. After a barrier makes the coefficient tiles visible, the cube cores evaluate

$$H^{\flat}_{\mathrm{res}} = \boldsymbol{A}(\mathbf{P}'_l)^{\flat}, \qquad [T,R]\times[R,n^2]\to[T,n^2]. \tag{60}$$

For $n=4$, this is an FP32 $[T,24]\times[24,16]$ projection using the framework-cached bases. Work is partitioned along the token dimension, with explicit tail handling.

The Pre path uses low-precision activation I/O and an FP32 coefficient path. X_l and H_{in} remain in BF16/FP16; projection weights, logits, softmax coefficients, and matrix multiplications use FP32. The cast input, projection result, inverse RMS factor, and residual coefficients cross GM workspace only at required vector-cube boundaries.

F.3 Fused Pre Backward Kernel and State Reuse

Pre backward fuses the gradients of stream aggregation, coefficient generation, softmax, and residual projection. The vector cores reduce the upstream H_{in} gradient against the original streams and prepare tiles for the projection-weight gradient. With $\nabla_H = \nabla_H \mathcal{L} \in \mathbb{R}^{T\times n^2}$ denoting the residual-matrix gradient, the kernel evaluates

$$\nabla_{\boldsymbol{A}} = \nabla_H((\mathbf{P}'_l)^{\flat})^{\mathsf{T}}, \tag{61}$$

$$\nabla_Z = \boldsymbol{A}\odot\left(\nabla_{\boldsymbol{A}} - \mathrm{reduce_sum}(\boldsymbol{A}\odot\nabla_{\boldsymbol{A}})\right), \tag{62}$$

where the reduction is independent for each token. The vector path reconstructs sigmoid derivatives from saved Pre and Post outputs, propagates the inverse-RMS gradient, and writes ∇_M to workspace.

The same cached $\mathbf{P}'_l$ used in forward is read as a constant input; neither $\mathbf{P}'_l$ nor λ_l receives a gradient in this operator.

The cube cores then execute the two high-dimensional matrix multiplications

$$\nabla_X^{\text{proj}} = \nabla_M \Phi, \qquad \nabla_\Phi = \nabla_M^{\mathsf{T}}\, \text{FP32}(X_l). \tag{63}$$

The vector cores combine ∇_X^{proj} with the aggregation, RMS, and optional direct Post gradients. Gradients of Φ, s, and b are reduced deterministically; scale and bias use compensated accumulation.

Training reuses the states in Table 7 instead of recomputing Pre. At $n = 4$, they contain 65 FP32 values (260 bytes) per token. Because H_{post} is already a forward output, only the remaining 61 values are optional training outputs; inference disables them.

Table 7: Forward states reused by the fused `cmHC` Pre backward operator for $n = 4$.

State	Shape	Data type	FP32 values/token
Inverse RMS factor	$[T, 1]$	FP32	1
Projection output M	$[T, 32]$	FP32	32
Pre coefficients	$[T, 4]$	FP32	4
Post coefficients	$[T, 4]$	FP32	4
Residual coefficients $\boldsymbol{A}$	$[T, 24]$	FP32	24
Total	–	FP32	65

F.4 Fused Post Forward and Backward Kernels

For token t, output stream i, and input stream j, Post computes

$$X_{l+1,t,i,:} = H_{\text{post},t,i}\, h^{\text{out}}_{l,t,:} + \sum_{j=1}^{n} H_{\text{res},t,j,i}\, X_{l,t,j,:}. \tag{64}$$

Input and output activations use BF16/FP16; H_{res} and H_{post} use FP32. The $n = 4$ implementation supports $[B, S, n, D]$ and flattened $[T, n, D]$ layouts and requires D to be divisible by 128.

Post forward uses vector cores only. Each core processes a token-hidden tile, initializes an FP32 accumulator with $H_{\text{post},i} h^{\text{out}}_l$, and applies four residual-stream scale-accumulates. The result is cast once on GM writeback. Small coefficients remain on chip, h^{out}_l is reused across output streams, and double-buffered queues overlap transfer with computation.

Tiling selects between two reuse schedules. If all n input tiles fit in UB, they remain resident across output streams; otherwise, inputs are streamed while the FP32 accumulator remains resident. This avoids UB overcommitment without changing the arithmetic.

Given the upstream gradient $\nabla_Y = \nabla_{X_{l+1}}$, the Post backward operator jointly evaluates

$$\nabla_{X_l,t,j,:} = \sum_{i=1}^{n} H_{\text{res},t,j,i} \nabla_{Y,t,i,:}, \qquad \nabla_{H_{\text{res}},t,j,i} = \sum_{d=1}^{D} X_{l,t,j,d} \nabla_{Y,t,i,d}, \tag{65}$$

$$\nabla_{h^{\text{out}},t,:} = \sum_{i=1}^{n} H_{\text{post},t,i} \nabla_{Y,t,i,:}, \qquad \nabla_{H_{\text{post}},t,i} = \sum_{d=1}^{D} h^{\text{out}}_{l,t,d} \nabla_{Y,t,i,d}. \tag{66}$$

All four gradient families are produced in one launch. Tokens are distributed across vector cores, and the hidden dimension uses aligned tiles with an explicit tail path. The Post vector and residual matrix remain in UB; vector multiply, multiply–accumulate, and reduction implement the small products. The coefficient gradients accumulate in FP32 and are written once per token, while activation gradients are cast and written per tile. Fusion reuses ∇_Y, X_l, and the coefficients, avoiding four scans of D.

F.5 Operator-Level Performance

We benchmark on an Ascend 910C server with CANN 9.1.0 (Huawei Technologies Co., Ltd., 2026). CANN (Compute Architecture for Neural Networks) provides the operator library, compiler toolchain,

and runtime for Ascend processors. The mHC comparison operators are the fused implementations from this release. Both pipelines use the same Post implementation, isolating their implementation difference to Pre.

All workloads use $B = 1$, $S = 4,096$, $n = 4$, and $D \in \{2048, 3584, 7168\}$. Latency is reported in microseconds by input shape. Speedup is $T_{\text{baseline}}/T_{\text{ours}}$. The model sublayer is excluded, and Pre and Post were timed independently. The complete-pipeline totals and reductions are reported in Table 2; below we provide the detailed Pre/Post breakdowns.

Table 8 isolates the differing Pre stage. The forward operator achieves $3.82\times$, $2.51\times$, and $1.93\times$ speedups. The shape-aware layout and cache policies are most consequential at large D, where wider rows otherwise increase cube-side layout-conversion cost and cache pressure. The backward gain narrows with D, reaching $1.46\times$ at $D = 7,168$ as projection gradients dominate fixed-size coefficient work.

Table 8: Latency of the fused $\mathcal{C}$mHC Pre and mHC Pre operators.

Input $[B, S, n, D]$	Fwd latency (μs)		Fwd speedup	Bwd latency (μs)		Bwd speedup
	$\mathcal{C}$mHC	mHC		$\mathcal{C}$mHC	mHC	
$[1, 4096, 4, 2048]$	**266.70**	1,017.50	3.82×	**444.40**	889.03	2.00×
$[1, 4096, 4, 3584]$	**485.96**	1,217.54	2.51×	**879.57**	1,365.56	1.55×
$[1, 4096, 4, 7168]$	**1,168.57**	2,255.84	1.93×	**2,082.52**	3,045.70	1.46×

Table 9 reports the identical Post cost in both pipelines. It accounts for 51.8%, 49.5%, and 49.8% of our forward-plus-backward cost, diluting the larger Pre-only speedups.

Table 9: Latency of the fused Post forward and backward operators used in the complete pipeline.

Input $[B, S, n, D]$	Fwd latency (μs)	Bwd latency (μs)	Total latency (μs)
$[1, 4096, 4, 2048]$	109.32	654.17	763.49
$[1, 4096, 4, 3584]$	160.20	1,180.29	1,340.49
$[1, 4096, 4, 7168]$	993.84	2,227.63	3,221.47

Finally, Table 10 compares our fused Pre operator with an unfused MindSpore composition (MindSpore, 2022). This isolates the benefit of custom-kernel fusion, while the mHC comparison evaluates the contraction-specific design. The two comparisons are separate benchmark series; absolute latencies for our operator are comparable only within each table.

Our fused Pre operator achieves $4.25\times$–$4.85\times$ forward and $3.16\times$–$4.90\times$ backward speedup over the MindSpore composition, confirming that fusion removes substantial launch and intermediate memory overhead. As D grows, high-dimensional work and the identical Post cost reduce the complete-pipeline gain.

Table 10: Latency of our fused Pre operator and the MindSpore primitive-operator composition.

Input $[B, S, n, D]$	Fwd latency (μs)		Fwd speedup	Bwd latency (μs)		Bwd speedup
	Ours	MindSpore		Ours	MindSpore	
$[1, 4096, 4, 2048]$	**314.17**	1,524.50	4.85×	**620.95**	3,042.44	4.90×
$[1, 4096, 4, 3584]$	**567.43**	2,607.34	4.59×	**1,259.12**	5,455.29	4.33×
$[1, 4096, 4, 7168]$	**1252.86**	5,325.69	4.25×	**3,517.94**	11,119.79	3.16×