MV-HC: MULTI-VIEW HYPER-CONNECTIONS FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hyper-Connections (HC) improve residual routing by expanding the hidden state into multiple interacting streams, while recent variants such as mHC, mHC-lite, and KromHC improve the stability and scalability of this multi-stream formulation. However, these methods typically initialize all streams from the same representation, diversifying *routing* but not the source information carried by each stream. We introduce **Multi-View Hyper-Connections (MV-HC)**, which instead initializes dedicated hyper-streams from complementary tokenization views of the same input. An auxiliary representation is causally aligned to the main-token sequence and propagated through KromHC-style manifold-constrained residual mixing, while a lightweight gated reduction combines the heterogeneous streams before the LM head. We evaluate MV-HC from 60M to 1.5B parameters and observe consistent improvements over KromHC in validation and zero-shot downstream performance. At 1.5B, MV-HC reduces BPB from 0.628 to $\mathbf{0.617}$, improves CORE from 55.2 to $\mathbf{56.7}$, and improves the broader language-modeling/BBH evaluation average from 35.1 to $\mathbf{36.7}$. These gains require only $\mathbf{0.14}\%$ additional parameters, while training throughput remains within 1% of KromHC (62.9k vs. 63.5k tokens/s). These results show that complementing multi-path residual routing with multi-view source representations provides a simple and efficient way to enrich hyper-connections.

1 INTRODUCTION

Residual connections are a fundamental component of modern deep neural networks and large language models. Given a hidden representation $\mathbf{z}_l \in \mathbb{R}^{B\times S\times d}$ at layer l, a standard residual layer is commonly written as

$$\mathbf{z}_{l+1} = \mathbf{z}_l + \mathcal{F}_l(\mathbf{z}_l), \tag{1}$$

where $\mathcal{F}_l(\cdot)$ denotes a learnable transformation such as attention, an MLP, or a Transformer block. The success of this formulation is largely attributed to its identity mapping property, which allows the input signal to propagate directly across depth. This property improves optimization stability and has enabled the scaling of very deep architectures. From a macro-architecture perspective, residual connections define how information is routed and merged across layers. While the standard residual connection provides a simple and effective single-stream pathway, its topology restricts information flow to one residual route. Recent macro-design studies have therefore explored richer residual routing mechanisms by increasing the number of residual streams and introducing multiple interacting pathways (Srivastava et al., 2015; He et al., 2016; Chai et al., 2020; Fang et al., 2023; Xie et al., 2023; Mak & Flanigan, 2025; Xiao et al., 2025; Zhu et al., 2024). Among them, Hyper-Connections (HC) generalize residual connections by maintaining an expanded stream tensor $\mathbf{X}_l \in \mathbb{R}^{B\times S\times n\times d}$, where n is the residual-stream expansion rate. A single HC layer can be written as

$$\mathbf{X}_{l+1} = \mathbf{H}_l^{\mathrm{res}}\mathbf{X}_l + \mathbf{H}_l^{\mathrm{post}\top}\mathcal{F}_l\left(\mathbf{H}_l^{\mathrm{pre}}\mathbf{X}_l\right), \tag{2}$$

where $\mathbf{H}_l^{\mathrm{res}}$ mixes residual streams, $\mathbf{H}_l^{\mathrm{pre}}$ aggregates streams before the main layer, and $\mathbf{H}_l^{\mathrm{post}}$ maps the layer output back into the expanded stream space. By increasing the topological complexity of residual propagation, HC provides richer information flow than conventional residual connections. However, unconstrained residual mixing can weaken the identity mapping property that makes residual networks stable. When $\mathbf{H}_l^{\mathrm{res}}$ is freely learned, the product of residual mixing matrices across

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.

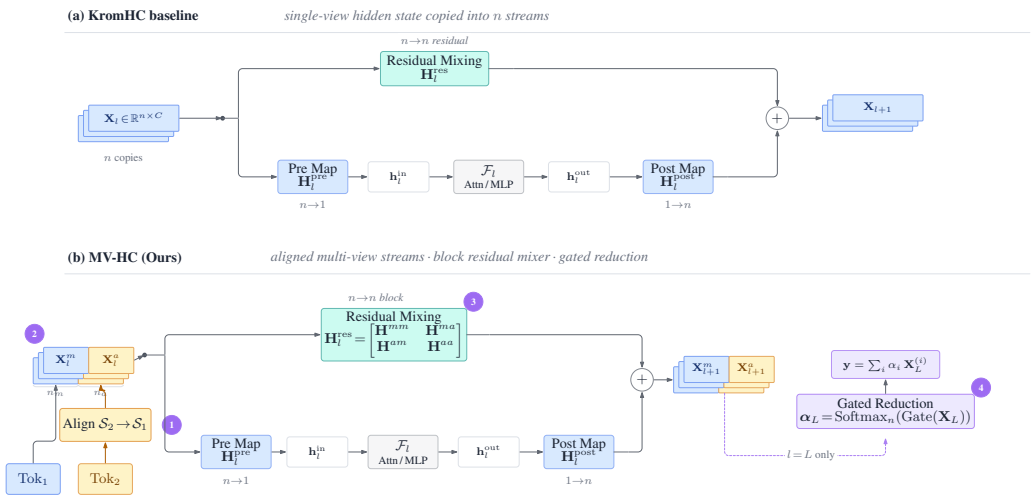

Figure 1: **KromHC vs. MV-HC. (a)** KromHC initializes all residual streams from the same representation. **(b)** MV-HC causally aligns an auxiliary tokenization view to the main sequence, initializes dedicated main and auxiliary stream groups, allows them to interact through KromHC-style constrained residual mixing, and applies a token-wise gated reduction before the LM head. Please refer to Appendices A.1 and A.17 for details.

layers may amplify, attenuate, or distort the residual signal, causing optimization instability and limiting scalability. Manifold-Constrained Hyper-Connections (mHC) address this issue by constraining $\mathbf{H}_l^{\text{res}}$ to the Birkhoff polytope, i.e., the set of doubly stochastic matrices (Xie et al., 2025). This makes residual mixing behave as a convex combination of streams and helps preserve the global feature mean during propagation. However, mHC relies on iterative Sinkhorn-Knopp projection (Sinkhorn & Knopp, 1967), which can be approximate and infrastructure-sensitive. mHC-lite (Yang & Gao, 2026) avoids repeated Sinkhorn-Knopp iterations by parameterizing doubly stochastic matrices through the Birkhoff-von Neumann theorem (Birkhoff, 1946), but its parameter complexity grows factorially with the residual stream width. KromHC (Zhou et al., 2026) further improves scalability by parameterizing the residual matrix as a Kronecker product of smaller doubly stochastic matrices,

$$\mathbf{H}_l^{\text{res}} = \mathbf{U}_l^{(1)} \otimes \mathbf{U}_l^{(2)} \otimes \cdots \otimes \mathbf{U}_l^{(R)}. \tag{3}$$

where R is the number of Kronecker factors and each $U_l^{(r)}$ is doubly stochastic. This construction is closely related to tensor-network ideas, where high-dimensional structures are represented through products or contractions of smaller factors to improve parameter efficiency (Novikov et al., 2015; Kolda & Bader, 2009; Cichocki et al., 2016). Since the Kronecker product of doubly stochastic matrices remains doubly stochastic, KromHC guarantees exact doubly stochasticity while reducing the parameter complexity of manifold-constrained residual mixing. Despite these advances, existing HC, mHC, mHC-lite, and KromHC mainly improve the *topology* of residual information flow while deriving all residual streams from the same hidden-state signal. In typical implementations, the previous hidden state $\mathbf{z}_l$ is expanded or repeated into multiple streams:

$$\mathbf{X}_l = \text{Expand}(\mathbf{z}_l) \in \mathbb{R}^{B \times S \times n \times d}. \tag{4}$$

Although these streams may later evolve differently through learned mixing, their source representation is identical. Thus, existing hyper-connection methods mainly diversify residual routing, but they do not explicitly diversify the source information carried by each stream. This motivates our central question: *Can hyper-connections be enriched not only by diversifying residual topology, but also by initializing residual streams from complementary views of the input?*

We propose **Multi-View Hyper-Connections (MV-HC)**, a tokenizer-augmented extension of manifold-constrained hyper-connections for language models. The goal of MV-HC is not to replace or repair the main tokenizer, but to provide hyper-connections with additional input-derived signals. Given the same raw input sequence, the main tokenizer produces the standard language-model representation $\mathbf{Z}^{\text{main}} \in \mathbb{R}^{B \times S_1 \times d}$, while an auxiliary tokenizer or segmentation mechanism, such as character-level, byte-level, word-level, or morphology-aware tokenization, produces

$\mathbf{Z}^{\text{aux}} \in \mathbb{R}^{B\times S_2\times d}$. Multi-granularity and tokenizer-free language modeling studies have shown that different input granularities can expose complementary lexical, orthographic, morphological, or boundary-level information (Clark et al., 2022; Xue et al., 2022; Tay et al., 2021; Yu et al., 2023). Prior work has also explicitly combined multiple tokenization granularities within a single model, including character–subword fusion in CharBERT, fine- and coarse-grained tokenization in LICHEE and AMBERT, and joint character–word modeling in Lattice-BERT (Ma et al., 2020; Zhang et al., 2021; Lai et al., 2021; Guo et al., 2021). Related approaches further exploit alternative segmentation views through multi-view regularization or mutually informed character and subword representations (Wang et al., 2021; 2024). MV-HC uses this observation not as a replacement for subword tokenization, but as a mechanism for constructing fresh auxiliary sources for hyper-stream initialization. Since the main and auxiliary views may have different sequence lengths, MV-HC first aligns the auxiliary representation to the main-token positions:

$$\widetilde{\mathbf{Z}}^{\text{aux}} = \text{Align}\left(\mathbf{Z}^{\text{aux}}, \mathbf{Z}^{\text{main}}\right) \in \mathbb{R}^{B\times S_1\times d}. \tag{5}$$

The initial hyper-stream tensor is then constructed from both the main and auxiliary views:

$$\mathbf{X}_0 = \text{MVExpand}\left(\mathbf{Z}^{\text{main}}, \widetilde{\mathbf{Z}}^{\text{aux}}\right) \in \mathbb{R}^{B\times S_1\times n\times d}. \tag{6}$$

The resulting streams are processed by an HC-style layer as in Eq. 2, using a constrained residual mixer such as mHC or KromHC. To recover the standard language-model hidden shape, MV-HC introduces a gated stream-reduction module after the final MV-HC layer. Let $\mathbf{X}_L^{(k)} \in \mathbb{R}^{B\times S_1\times d}$ denote the k-th residual stream at the last layer. We compute

$$\mathbf{Z}^{\text{out}} = \sum_{k=1}^{n} \alpha_L^{(k)} \odot \mathbf{X}_L^{(k)}, \tag{7}$$

where $\alpha_L^{(k)}$ is a learned token-wise gate for stream k at the final layer. The output $\mathbf{Z}^{\text{out}} \in \mathbb{R}^{B\times S_1\times d}$ is then passed to the LM head. Therefore, the main tokenizer still defines the autoregressive prediction positions and output vocabulary, while the auxiliary tokenizer only enriches the internal residual-stream representation. In this way, MV-HC extends hyper-connections from multi-path residual routing to multi-view residual representation learning. The auxiliary view provides complementary information extracted directly from the raw input signal, while manifold-constrained residual mixing preserves stable propagation across hyper-streams. Our contributions are summarized as follows:

- We identify a limitation of existing HC, mHC, mHC-lite, and KromHC formulations: their residual streams are typically initialized from the same hidden-state signal, so hyper-connections mainly diversify residual routing rather than the source information carried by each stream.
- We propose Multi-View Hyper-Connections (MV-HC), a tokenizer-augmented framework that initializes hyper-streams from complementary input-derived views while preserving compatibility with standard autoregressive language modeling.
- We introduce an alignment module that maps auxiliary-token representations to the main-token sequence length, enabling heterogeneous tokenization granularities to be integrated into the hyper-connection pathway.
- We combine multi-view stream initialization with KromHC-style manifold-constrained residual mixing to preserve stable propagation while improving representational diversity.
- We propose a gated stream-reduction mechanism that adaptively merges heterogeneous residual streams back into the standard language-model hidden state.

2 METHOD

As KromHC provides exact doubly stochastic residual matrices with lower parameter and memory overhead than previous manifold-constrained HC variants, we implement MV-HC on top of the KromHC formulation. Our method keeps the main KromHC layer computation unchanged, but modifies the residual-stream construction and the final stream reduction. Specifically, MV-HC replaces repeated same-view stream expansion with multi-view stream initialization, and replaces uniform stream summation with gated stream reduction. The auxiliary tokenizer pathway and alignment details are described in Appendix A.1.

2.1 BACKGROUND: KROMHC RESIDUAL STREAM

Let $\mathbf{z}_l \in \mathbb{R}^{B\times S\times d}$ be the hidden representation at layer l, where B is the batch size, S is the sequence length, and d is the hidden dimension. KromHC expands the single hidden state into n residual streams:

$$\mathbf{X}_l \in \mathbb{R}^{B\times S\times n\times d}. \tag{8}$$

In the original implementation, the initial stream tensor is obtained by repeating the same hidden state across the stream dimension:

$$\mathbf{X}_0^{(k)} = \mathbf{z}_0, \qquad k = 1, \ldots, n, \tag{9}$$

where k indexes the residual streams. The KromHC layer then follows the standard HC-style update:

$$\mathbf{X}_{l+1} = \mathbf{H}_l^{\text{res}}\mathbf{X}_l + \mathbf{H}_l^{\text{post}\top}\mathcal{F}_l\left(\mathbf{H}_l^{\text{pre}}\mathbf{X}_l\right), \tag{10}$$

where $\mathcal{F}_l(\cdot)$ denotes the main neural branch, e.g., attention, MLP, or a Transformer block. The pre-mixing matrix $\mathbf{H}_l^{\text{pre}} \in \mathbb{R}^{1\times n}$ aggregates the n streams into a single hidden representation before the main branch, while $\mathbf{H}_l^{\text{post}} \in \mathbb{R}^{1\times n}$ writes the branch output back into the expanded stream space. Therefore, the expensive branch $\mathcal{F}_l$ is applied only to a tensor of shape $\mathbb{R}^{B\times S\times d}$, not independently to all streams. The residual stream mixer $\mathbf{H}_l^{\text{res}} \in \mathbb{R}^{n\times n}$ is parameterized by KromHC as a Kronecker product of smaller doubly stochastic matrices. Let

$$n = \prod_{r=1}^{R} i_r, \qquad \mathbf{U}_l^{(r)} \in \mathbb{R}^{i_r\times i_r}, \qquad r = 1, \ldots, R, \tag{11}$$

where R is the number of Kronecker factors and each $\mathbf{U}_l^{(r)}$ is doubly stochastic. Then

$$\mathbf{H}_l^{\text{res}} = \bigotimes_{r=1}^{R} \mathbf{U}_l^{(r)}. \tag{12}$$

Since the Kronecker product of doubly stochastic matrices remains doubly stochastic, $\mathbf{H}_l^{\text{res}}$ preserves the stability property of manifold-constrained residual mixing while avoiding the dense parameterization cost of mHC and the factorial growth of mHC-lite. In practice, KromHC folds the stream dimension into the batch dimension, so the conceptual tensor $\mathbf{X}_l \in \mathbb{R}^{B\times S\times n\times d}$ is represented as $\mathbb{R}^{(Bn)\times S\times d}$ during computation.

2.2 MULTI-VIEW STREAM INITIALIZATION

The key limitation of Eq. equation 9 is that all streams are initialized from the same hidden-state signal. MV-HC instead initializes different stream groups from different views of the same raw input. Given an input sequence, the main tokenizer produces tokens of length S_1, and the auxiliary tokenizer produces tokens of length S_2. After embedding and projection, we obtain

$$\mathbf{Z}^{\text{main}} \in \mathbb{R}^{B\times S_1\times d}, \qquad \mathbf{Z}^{\text{aux}} \in \mathbb{R}^{B\times S_2\times d}. \tag{13}$$

Since S_1 and S_2 may differ, the auxiliary representation is aligned to the main-token positions:

$$\widetilde{\mathbf{Z}}^{\text{aux}} = \text{Align}\left(\mathbf{Z}^{\text{aux}}, \mathbf{Z}^{\text{main}}\right) \in \mathbb{R}^{B\times S_1\times d}. \tag{14}$$

The alignment is constrained to preserve autoregressive causality: at main-token position i, only auxiliary units whose complete raw-text spans lie within the observed prefix may contribute. Within this causal mask, alignment can use span-based mean pooling, overlap-weighted pooling, or lightweight attention pooling; implementation details are provided in Appendix A.4. We divide the total number of streams into main-view and auxiliary-view groups, i.e. $n = n_m + n_a$. The MV-HC stream initialization is then

$$\mathbf{X}_0 = \left[\mathbf{X}_0^{\text{main}}, \mathbf{X}_0^{\text{aux}}\right] \in \mathbb{R}^{B\times S_1\times n\times d}, \tag{15}$$

where

$$\mathbf{X}_0^{\text{main}} = \text{Repeat}_{n_m}\left(\mathbf{Z}^{\text{main}}\right), \qquad \mathbf{X}_0^{\text{aux}} = \text{Repeat}_{n_a}\left(\widetilde{\mathbf{Z}}^{\text{aux}}\right). \tag{16}$$

For example, when $n = 4$ and $n_m = n_a = 2$, MV-HC initializes two streams from the main-tokenizer view and two streams from the aligned auxiliary view:

$$\mathbf{X}_0 = \text{Stack}\left(\mathbf{Z}^{\text{main}}, \mathbf{Z}^{\text{main}}, \widetilde{\mathbf{Z}}^{\text{aux}}, \widetilde{\mathbf{Z}}^{\text{aux}}\right). \tag{17}$$

This is the first modification relative to KromHC: stream expansion is no longer a repetition of a single representation, but a multi-view initialization from complementary input-derived signals.

2.3 AUXILIARY-CARRYING STREAM PROPAGATION

To keep the auxiliary signal available beyond the first layer, MV-HC reserves a subset of residual streams as auxiliary-carrying streams. At layer l, we write the stream tensor as

$$\mathbf{X}_l = \left[\mathbf{X}_l^{\text{main}}, \mathbf{X}_l^{\text{aux}}\right], \tag{18}$$

where $\mathbf{X}_l^{\text{main}} \in \mathbb{R}^{B\times S_1\times n_m\times d}$ and $\mathbf{X}_l^{\text{aux}} \in \mathbb{R}^{B\times S_1\times n_a\times d}$. The auxiliary stream group is not a fixed copy of $\widetilde{\mathbf{Z}}^{\text{aux}}$; rather, it evolves through the network:

$$\widetilde{\mathbf{Z}}^{\text{aux}} \to \mathbf{X}_0^{\text{aux}} \to \mathbf{X}_1^{\text{aux}} \to \cdots \to \mathbf{X}_L^{\text{aux}}.$$

Thus, later layers can access an edited auxiliary-view representation carried by the hyper-streams. Assuming the stream order places the n_m main streams before the n_a auxiliary streams, the KromHC residual matrix can be interpreted in block form according to this stream partition:

$$\mathbf{H}_l^{\text{res}} = \begin{bmatrix} \mathbf{H}_l^{mm} & \mathbf{H}_l^{ma} \\ \mathbf{H}_l^{am} & \mathbf{H}_l^{aa} \end{bmatrix}, \tag{19}$$

where

$$\mathbf{H}_l^{mm} \in \mathbb{R}^{n_m\times n_m}, \quad \mathbf{H}_l^{ma} \in \mathbb{R}^{n_m\times n_a}, \quad \mathbf{H}_l^{am} \in \mathbb{R}^{n_a\times n_m}, \quad \mathbf{H}_l^{aa} \in \mathbb{R}^{n_a\times n_a}.$$

Here, the row index denotes the destination stream group and the column index denotes the source stream group. Thus, $\mathbf{H}_l^{mm}$ mixes main streams into main streams, $\mathbf{H}_l^{ma}$ maps auxiliary streams into main streams, $\mathbf{H}_l^{am}$ maps main streams into auxiliary streams, and $\mathbf{H}_l^{aa}$ mixes auxiliary streams into auxiliary streams. The residual propagation term becomes

$$\begin{bmatrix} \bar{\mathbf{X}}_{l+1}^{\text{main}} \\ \bar{\mathbf{X}}_{l+1}^{\text{aux}} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_l^{mm} & \mathbf{H}_l^{ma} \\ \mathbf{H}_l^{am} & \mathbf{H}_l^{aa} \end{bmatrix} \begin{bmatrix} \mathbf{X}_l^{\text{main}} \\ \mathbf{X}_l^{\text{aux}} \end{bmatrix}. \tag{20}$$

Equivalently,

$$\bar{\mathbf{X}}_{l+1}^{\text{main}} = \mathbf{H}_l^{mm}\mathbf{X}_l^{\text{main}} + \mathbf{H}_l^{ma}\mathbf{X}_l^{\text{aux}}, \tag{21}$$

$$\bar{\mathbf{X}}_{l+1}^{\text{aux}} = \mathbf{H}_l^{am}\mathbf{X}_l^{\text{main}} + \mathbf{H}_l^{aa}\mathbf{X}_l^{\text{aux}}. \tag{22}$$

The block notation does not introduce a new residual mixer; it only exposes how the KromHC residual matrix routes information within and across the main and auxiliary stream groups.

2.4 MV-HC LAYER UPDATE

Given the multi-view stream tensor $\mathbf{X}_l \in \mathbb{R}^{B\times S_1\times n\times d}$, the pre-mixing module aggregates the n streams into the input of the main branch. Specifically, $\mathbf{H}_l^{\text{pre}} \in \mathbb{R}^{1\times n}$ contracts with $\mathbf{X}_l$ over the stream dimension:

$$\mathbf{U}_l = \sum_{k=1}^{n} H_{l,k}^{\text{pre}}\mathbf{X}_l^{(k)} \in \mathbb{R}^{B\times S_1\times d}, \tag{23}$$

where $\mathbf{X}_l^{(k)} \in \mathbb{R}^{B\times S_1\times d}$ denotes the k-th stream. Equivalently, this can be written compactly as $\mathbf{U}_l = \mathbf{H}_l^{\text{pre}}\mathbf{X}_l$, with the multiplication understood as contraction over the stream dimension. Because $\mathbf{X}_l$ contains both main-view and auxiliary-view streams, $\mathbf{U}_l$ is a learned mixture of both sources. The main branch output is then

$$\mathbf{Y}_l = \mathcal{F}_l\left(\mathbf{U}_l\right), \qquad \mathbf{Y}_l \in \mathbb{R}^{B\times S_1\times d}. \tag{24}$$

Finally, $\mathbf{Y}_l$ is written back to the stream space using $\mathbf{H}_l^{\text{post}} \in \mathbb{R}^{1\times n}$. This operation broadcasts the branch output to the n streams with learned stream-wise weights:

$$\left(\mathbf{H}_l^{\text{post}\top}\mathbf{Y}_l\right)^{(k)} = H_{l,k}^{\text{post}}\mathbf{Y}_l, \qquad k = 1, \ldots, n, \tag{25}$$

so that $\mathbf{H}_l^{\text{post}\top}\mathbf{Y}_l \in \mathbb{R}^{B\times S_1\times n\times d}$. The full stream update is therefore

$$\mathbf{X}_{l+1} = \mathbf{H}_l^{\text{res}}\mathbf{X}_l + \mathbf{H}_l^{\text{post}\top}\mathbf{Y}_l, \tag{26}$$

where $\mathbf{H}_l^{\text{res}}\mathbf{X}_l$ also denotes multiplication over the stream dimension. Since $\mathbf{H}_l^{\text{res}}$ is generated by the KromHC Kronecker-product construction, it remains doubly stochastic and preserves stable residual propagation. At the same time, the auxiliary-view information is carried across depth through $\mathbf{X}_l^{\text{aux}}$.

2.5 GATED STREAM REDUCTION

After the final layer, KromHC reduces the stream tensor by summing over all streams. This is appropriate when every stream starts from the same representation, but MV-HC streams may contain heterogeneous main-view and auxiliary-view information. Therefore, MV-HC uses a learned gated reduction. Let the final stream tensor be

$$\mathbf{X}_L = [\mathbf{X}_L^{(1)}, \ldots, \mathbf{X}_L^{(n)}], \qquad \mathbf{X}_L^{(k)} \in \mathbb{R}^{B \times S_1 \times d}.$$

We compute token-wise stream scores and normalize them over the stream dimension:

$$\boldsymbol{\alpha}_L = \mathrm{Softmax}_n \left(\mathrm{Gate} \left(\mathbf{X}_L \right) \right), \qquad \boldsymbol{\alpha}_L \in \mathbb{R}^{B \times S_1 \times n}. \tag{27}$$

where $\mathrm{Softmax}_n(\cdot)$ denotes softmax normalization over the stream dimension, and $\mathrm{Gate}(\cdot)$ is a lightweight token-wise linear projection shared across streams. Concretely, for each stream k,

$$g_L^{(k)} = \mathbf{w}_g^\top \mathbf{X}_L^{(k)} + b_g, \qquad g_L^{(k)} \in \mathbb{R}^{B \times S_1}, \tag{28}$$

and the scores $\{g_L^{(k)}\}_{k=1}^n$ are stacked along the stream dimension before applying $\mathrm{Softmax}_n$. The final hidden state is

$$\mathbf{Z}^{\mathrm{out}} = \sum_{k=1}^{n} \boldsymbol{\alpha}_L^{(k)} \odot \mathbf{X}_L^{(k)} \in \mathbb{R}^{B \times S_1 \times d}. \tag{29}$$

Here, $\odot$ denotes element-wise multiplication with broadcasting over the hidden dimension. The resulting $\mathbf{Z}^{\mathrm{out}}$ has the standard language-model hidden shape and can be passed directly to the LM head:

$$\mathbf{P} = \mathrm{Softmax} \left(\mathbf{Z}^{\mathrm{out}} \mathbf{W}_{\mathrm{vocab}} \right). \tag{30}$$

The main tokenizer therefore still defines the autoregressive prediction positions and output vocabulary; the auxiliary tokenizer only enriches the internal residual-stream representation.

3 EXPERIMENTS

We evaluate MV-HC on autoregressive language model pretraining and downstream evaluation. Our main protocol follows the same setting used for KromHC: we compare against the standard residual connection, mHC, mHC-lite, and KromHC under matched data, main tokenizer, optimizer, training schedule, residual stream width, and evaluation suite. In addition to the two model sizes used in KromHC, we include a larger $\sim$1.5B-parameter model to study whether the gains of MV-HC remain stable at a more practical LLM scale. Unless otherwise specified, MV-HC uses BPE as the main tokenizer and a byte-level tokenizer as the auxiliary tokenizer, and unless otherwise stated, all reported downstream results are evaluated in the zero-shot setting.

3.1 EXPERIMENTAL SETUP

We run experiments on three transformer scales: a small model with approximately 60M parameters and $D = 6$ transformer blocks, a medium model with approximately 186M parameters and $D = 12$ transformer blocks, and a large model with approximately 1.5B parameters and $D = 36$ transformer blocks. The main experiments use $n = 4$ residual streams, following the KromHC comparison setting. For each method, we replace the original residual connections in every transformer block with the corresponding residual mixing module, while keeping the attention, MLP, normalization, main tokenizer, optimizer, and training data unchanged. Following the KromHC setup, we train all models on FineWeb-Edu (Lozhkov et al., 2024) using a token-to-parameter ratio of approximately 20:1, as suggested by Hoffmann et al. (Hoffmann et al., 2022). For the 60M and 186M models, we follow the same token-to-parameter ratio used in KromHC. For the 1.5B model, we use the same training recipe and scale the token budget proportionally, unless otherwise stated. The baselines and MV-HC are trained from scratch under the same compute configuration for each model size. MV-HC is implemented on top of the KromHC residual formulation. It keeps the Kronecker-product residual mixing structure but augments it with a multi-view auxiliary path. This allows MV-HC to preserve the compact parameterization of KromHC while adding extra view-conditioned residual information. The auxiliary tokenizer, view projection, and view gate are counted in the reported parameter overhead. We report pretraining loss, validation bits-per-byte (BPB), CORE score, commonsense reasoning accuracy, language modeling and BigBench-Hard task accuracy, and training throughput.

Table 1: Training and validation results under matched pretraining settings.

Model	Method	Train ↓	BPB ↓	CORE ↑	Δ Params
60M	Residual	3.214	0.786	41.1	0.00M
	mHC	3.198	0.779	42.2	3.84M
	mHC-lite	3.201	0.781	42.3	0.51M
	KromHC	3.185	0.772	42.4	0.18M
	MV-HC	**3.162**	**0.764**	**43.2**	0.31M
186M	Residual	2.982	0.721	46.2	0.00M
	mHC	2.968	0.715	44.5	7.68M
	mHC-lite	2.961	0.713	44.4	1.02M
	KromHC	2.936	0.704	47.7	0.24M
	MV-HC	**2.911**	**0.696**	**48.6**	0.43M
1.5B	Residual	2.514	0.642	53.8	0.00M
	mHC	2.503	0.637	54.1	31.20M
	mHC-lite	2.499	0.635	54.4	4.18M
	KromHC	2.481	0.628	55.2	1.36M
	MV-HC	**2.452**	**0.617**	**56.7**	2.05M

3.2 Initialization

All hyper-connection variants are initialized so that the residual path is close to an ordinary residual connection at the start of training. For MV-HC, the Kronecker residual mixing factors are initialized to approximate the identity transformation. The residual scaling parameters are initialized with small positive values, and the identity-associated bias entries are initialized to dominate the residual mixing distribution. The multi-view branch is also initialized conservatively. The auxiliary view projection uses small initial weights. Therefore, MV-HC starts from a stable residual-like regime and gradually learns to use the auxiliary views during training.

3.3 Training and Validation Metrics

We evaluate MV-HC against the standard residual baseline, mHC, mHC-lite, and KromHC using the same pretraining and evaluation protocol. Table 1 summarizes the final training loss, validation BPB, CORE score, and additional trainable parameters. The training objective is the standard autoregressive cross-entropy loss, $\mathcal{L}_{\mathrm{CE}} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t})$. To compare models independently of tokenizer granularity, we report validation bits-per-byte, computed as $\mathcal{L}_{\mathrm{BPB}} = \frac{\mathcal{L}_{\mathrm{CE}}}{\ln(2)} \times \frac{\text{Total Tokens}}{\text{Total Bytes}}$. We also follow the KromHC evaluation setting and report CORE, a centered-accuracy summary over a fixed suite of downstream tasks for measuring general language understanding.

As shown in Table 1, MV-HC obtains the best validation BPB and CORE score across all three model scales. The gains are more pronounced for the 1.5B model, suggesting that the benefit of multi-view residual streams may increase with model capacity. Figure 2 further compares the training-loss trajectories in a zoomed middle-stage window. MV-HC follows a smooth optimization path and shows slightly better convergence than KromHC and the other hyper-connection variants.

3.4 Downstream Task Performance

We evaluate the models on the same downstream benchmark groups used in the KromHC comparison. Tables 2 and 3 report detailed results for commonsense reasoning and language modeling tasks. MV-HC obtains the highest average score at all three scales. MV-HC improves over KromHC on the average commonsense score at all three model scales, including an improvement from 55.3 to 56.7 at the 1.5B scale. This suggests that the multi-view path provides useful residual information beyond the Kronecker residual mixing used in KromHC. Table 3 shows that MV-HC also improves the average score on language modeling, question answering, BBH-style subtasks, LSAT, and language identification. The improvement is consistent across model sizes.

3.5 Efficiency

We measure the training throughput of each method using the average number of processed tokens per second during pretraining. The goal of this experiment is not to claim that hyper-connections

Table 2: Commonsense and reasoning benchmark results. Accuracy is reported in percentage.

Model	Method	Avg	HS	HS-ZS	PIQA	ARC-E	ARC-C	COPA	CSQA	OBQA	WG	WGrande	BoolQ
60M	Residual	39.8	27.0	29.0	60.4	42.8	21.8	55.0	31.8	23.8	59.3	49.2	37.4
60M	mHC	40.8	26.4	28.4	60.8	42.8	22.4	60.0	23.0	24.8	55.3	50.4	54.4
60M	mHC-lite	41.0	27.4	28.2	56.4	42.8	24.4	53.0	35.0	25.6	53.4	50.0	54.4
60M	KromHC	41.1	28.0	27.8	59.0	41.4	23.0	54.0	28.8	25.4	53.5	50.0	60.8
60M	**MV-HC**	**43.5**	**28.4**	**29.6**	60.6	43.1	**25.2**	**61.0**	34.2	**26.8**	58.7	**51.6**	58.9
186M	Residual	46.2	36.6	35.4	65.4	58.4	26.4	63.0	36.6	32.6	57.5	52.0	44.0
186M	mHC	47.5	36.2	37.6	64.2	58.6	27.4	64.0	32.4	33.6	60.4	53.2	54.6
186M	mHC-lite	44.4	35.2	36.2	64.6	58.6	27.2	63.0	23.6	30.4	58.6	49.2	41.6
186M	KromHC	47.7	36.4	38.4	65.0	57.8	27.6	66.0	30.0	31.2	58.6	54.8	58.4
186M	**MV-HC**	**49.3**	**37.2**	**39.1**	**66.2**	**59.4**	**28.8**	**67.0**	35.1	33.0	**61.1**	**55.7**	**59.8**
1.5B	Residual	53.8	47.5	46.1	71.8	66.2	38.5	72.0	43.4	37.8	64.0	59.3	45.5
1.5B	mHC	54.1	47.8	46.9	72.4	66.8	39.1	72.0	44.1	38.4	64.8	59.7	43.5
1.5B	mHC-lite	54.4	48.1	47.3	72.1	67.0	39.6	73.0	44.6	38.8	65.0	60.1	43.3
1.5B	KromHC	55.3	48.8	48.0	73.0	68.2	40.7	74.0	45.2	39.4	65.6	61.2	43.7
1.5B	**MV-HC**	**56.7**	**50.2**	**49.6**	**74.4**	**69.8**	**42.3**	**76.0**	**46.8**	**41.1**	**67.4**	**62.6**	**43.9**

Table 3: Language modeling, BigBench-Hard subtasks, and additional evaluation results. Accuracy is reported in percentage.

Model	Method	Avg	Lamb	SQuAD	CoQA	BBH-QA	BBH-CS	BBH-Op	BBH-Dyck	LSAT	LangID
60M	Residual	16.2	18.6	0.0	5.6	20.0	40.8	9.1	4.6	23.5	23.4
60M	mHC	15.4	17.4	0.0	4.8	10.6	42.0	5.2	9.2	23.5	26.0
60M	mHC-lite	16.9	19.6	0.4	4.6	19.4	38.2	5.7	6.4	29.6	28.0
60M	KromHC	17.3	19.0	0.2	5.8	14.0	40.8	8.6	11.4	27.8	27.8
60M	**MV-HC**	**18.9**	**20.3**	**0.8**	**6.3**	19.8	**43.1**	**9.8**	**12.2**	28.9	**29.0**
186M	Residual	23.7	29.2	10.8	13.8	39.2	38.8	12.9	15.8	27.8	25.4
186M	mHC	22.9	31.6	5.8	13.0	39.6	42.0	16.7	13.0	20.4	24.0
186M	mHC-lite	23.3	30.0	8.4	14.2	36.6	42.6	14.8	10.0	27.0	26.2
186M	KromHC	24.0	30.4	8.2	15.4	40.4	44.6	11.9	13.6	26.1	25.0
186M	**MV-HC**	**26.2**	**32.2**	10.2	**16.6**	**41.8**	**45.7**	**17.3**	**16.4**	**28.4**	**27.1**
1.5B	Residual	33.2	42.0	21.8	25.4	47.2	50.8	21.5	24.0	34.5	32.0
1.5B	mHC	33.7	42.5	22.2	25.8	47.8	51.2	22.1	24.5	34.9	32.3
1.5B	mHC-lite	34.2	43.1	22.8	26.2	48.3	51.9	22.7	25.1	35.2	32.8
1.5B	KromHC	35.1	44.0	23.6	27.1	49.2	52.8	23.5	26.2	35.9	33.4
1.5B	**MV-HC**	**36.7**	**45.7**	**25.1**	**28.7**	**51.0**	**54.2**	**25.2**	**27.8**	**37.6**	**35.0**

improve raw throughput over the standard residual baseline. In fact, all HC-based methods introduce additional operations for stream expansion, stream mixing, and pre/post residual routing, and therefore are expected to have lower throughput than a standard residual model. Instead, this experiment evaluates the *extra overhead* introduced by MV-HC relative to existing hyper-connection variants, especially KromHC. MV-HC adds an auxiliary-view pathway, including auxiliary embedding, alignment, projection, and gated stream reduction. However, the main Transformer branch is still executed only once per layer, and the residual mixer keeps the same compact KromHC-style Kronecker-product structure. As a result, the additional cost of MV-HC remains close to KromHC. Table 4 reports throughput for the 1.5B model. Compared with the residual baseline, all hyper-connection methods are slower. Among HC-based methods, KromHC achieves the highest throughput, while MV-HC remains very close to KromHC despite using an additional auxiliary view. This shows that the proposed multi-view extension improves representation quality with only a small extra overhead beyond KromHC.

3.6 Ablation Study

We conduct ablation studies on the 186M model with $n = 4$ residual streams. First, Table 4 separates the contribution of the auxiliary tokenizer view from the final gated stream reduction. Adding the same gated reduction to KromHC gives only a small improvement, suggesting that the gate alone does not explain the full gain. MV-HC without the gate still improves over KromHC, indicating that the auxiliary view contributes useful information to the hyper-streams. The full MV-HC model performs best, suggesting that the gate is most effective when combining heterogeneous main-view and auxiliary-view streams. Second, Table 5 studies additional MV-HC design choices. Removing the auxiliary view path reduces both validation and downstream performance, while removing the



Figure 2: Zoomed-in view of the training loss trajectories for the 1.5B model during the middle stage of training. MV-HC maintains a consistently competitive loss trajectory and converges slightly below KromHC, suggesting that the auxiliary tokenizer view improves representation learning without disrupting the stable optimization behavior inherited from KromHC.

Model	Method	Tokens/sec ↑	Rel. Thr. ↑	Overhead ↓
1.5B	Residual	145.7k	1.00×	0.0%
1.5B	HC	60.2k	0.41×	58.7%
1.5B	mHC	52.2k	0.36×	64.2%
1.5B	mHC-lite	60.4k	0.41×	58.5%
1.5B	KromHC	**63.5k**	**0.44×**	**56.4%**
1.5B	MV-HC	62.9k	0.43×	56.8%

Table 4: Training throughput comparison across hyper-connection variants. Throughput is measured in tokens/sec on 8× NVIDIA H100 80GB without specialized kernels. All hyper-connection variants introduce overhead relative to the standard residual baseline, while MV-HC remains close to KromHC among HC-based methods.

Table 5: Ablation studies on the 186M model with $n = 4$. The upper block isolates the effects of the auxiliary view and gated reduction, while the lower block evaluates additional MV-HC components.

Variant	Aux. View	Gate	Train Loss ↓	BPB ↓	CORE ↑	Avg. Down. ↑
Auxiliary-view and gated-reduction ablation						
KromHC	No	No	–	0.704	47.7	–
KromHC + gate	No	Yes	–	0.702	47.9	–
MV-HC w/o gate	Yes	No	–	0.702	48.2	–
Full MV-HC	Yes	Yes	–	**0.696**	**48.6**	–
Component ablation						
KromHC baseline	No	No	2.936	0.704	47.7	24.0
MV-HC w/o auxiliary view	No	Yes	2.928	0.702	47.9	24.3
MV-HC w/o gate	Yes	No	2.923	0.702	48.2	24.4
MV-HC with shared tokenizer	Yes	Yes	2.917	0.698	48.3	24.7
Full MV-HC	Yes	Yes	**2.911**	**0.696**	**48.6**	**26.2**

view gate also leads to a weaker result than the full model. Sharing the auxiliary tokenizer across layers performs better than removing the auxiliary pathway, but still remains below full MV-HC. Overall, these ablations suggest that the improvements come from the combination of multi-view stream initialization and adaptive stream reduction, rather than from a single added component.

4 Conclusion

We introduced **Multi-View Hyper-Connections (MV-HC)**, which initializes residual streams from complementary tokenization views instead of repeated copies of the same representation. Built on KromHC, MV-HC adds causally aligned multi-view stream initialization and gated reduction while retaining the underlying residual update. Across models from 60M to 1.5B, MV-HC consistently improves validation and downstream performance over KromHC with minimal additional overhead. Controlled ablations further indicate that the gains come from preserving heterogeneous views in distinct hyper-streams rather than from extra parameters or gating alone. Future work will explore multiple simultaneous views, larger stream widths, and larger model scales.

References

Garrett Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucuman Rev. Ser. A*, 1946.

Yekun Chai, Shuo Jin, and Xinwen Hou. Highway transformer: Self-gating enhanced self-attentive networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6887–6900, 2020.

Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimization part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9(4-5):249–429, 2016.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.

Yanwen Fang, Yuxi Cai, Jintai Chen, Jingyu Zhao, Guangjian Tian, and Guodong Li. Cross-layer retrospective retrieving via layer attention. *arXiv preprint arXiv:2302.03985*, 2023.

Weidong Guo, Mingjun Zhao, Lusheng Zhang, Di Niu, Jinwen Luo, Zhenhua Liu, Zhenyang Li, and Jianbo Tang. LICHEE: Improving language model pre-training with multi-grained tokenization. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 1383–1392, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.119. URL https://aclanthology.org/2021.findings-acl.119/.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. In *SIAM Review*, 2009.

Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Lattice-BERT: Leveraging multi-granularity representations in Chinese pre-trained language models. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1716–1731, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.137. URL https://aclanthology.org/2021.naacl-main.137/.

Anton Lozhkov et al. FineWeb-Edu: The finest collection of educational content, 2024. Dataset. DOI: 10.57967/hf/2497.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. CharBERT: Character-aware pre-trained language model. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 39–50, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.4. URL https://aclanthology.org/2020.coling-main.4/.

Brian Mak and Jeffrey Flanigan. Residual matrix transformers: Scaling the size of the residual stream. *arXiv preprint arXiv:2506.22696*, 2025.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks. 2015.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 1967.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*, 2021.

Xinyi Wang, Sebastian Ruder, and Graham Neubig. Multi-view subword regularization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 473–482, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.40. URL https://aclanthology.org/2021.naacl-main.40/.

Yilin Wang, Xinyi Hu, and Matthew Gormley. Learning mutually informed representations for characters and subwords. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3201–3213, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.202. URL https://aclanthology.org/2024.findings-naacl.202/.

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

Shufang Xie, Huishuai Zhang, Junliang Guo, Xu Tan, Jiang Bian, Hany Hassan Awadalla, Arul Menezes, Tao Qin, and Rui Yan. Residual: Transformer with dual residual connections. *arXiv preprint arXiv:2304.14802*, 2023.

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

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

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

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 36:78808–78823, 2023.

Xinsong Zhang, Pengshuai Li, and Hang Li. Ambert: A pre-trained language model with multi-grained tokenization. In *Findings of the association for computational linguistics: ACL-IJCNLP 2021*, pp. 421–435, 2021.

Wuyang Zhou, Yuxuan Gu, Giorgos Iacovides, and Danilo Mandic. Kromhc: Manifold-constrained hyper-connections with kronecker-product residual matrices. *arXiv preprint arXiv:2601.21579*, 2026.

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

A Appendix

A.1 Auxiliary Tokenizer Embedding and Alignment

This appendix describes the auxiliary tokenizer pathway used in MV-HC. The purpose of this pathway is not to replace the main tokenizer or change the language-modeling objective. Instead, it provides an additional input-derived representation that can initialize auxiliary hyper-streams.

A.2 Two Tokenizers and Two Embedding Tables

Given a raw input sequence x, MV-HC applies two tokenization functions. The main tokenizer $\mathcal{T}_{\text{main}}$ produces the standard token sequence used by the language model:

$$\mathbf{t}^{\text{main}} = \mathcal{T}_{\text{main}}(x) \in \mathbb{N}^{B\times S_1}, \tag{31}$$

where S_1 is the main-token sequence length. An auxiliary tokenizer $\mathcal{T}_{\text{aux}}$ produces an alternative view of the same input:

$$\mathbf{t}^{\text{aux}} = \mathcal{T}_{\text{aux}}(x) \in \mathbb{N}^{B\times S_2}, \tag{32}$$

where S_2 may be different from S_1. For example, $\mathcal{T}_{\text{main}}$ may be a BPE tokenizer, while $\mathcal{T}_{\text{aux}}$ may be a character-level, byte-level, word-level, or morphology-aware tokenizer.

Since the two tokenizers generally have different vocabularies, MV-HC uses two separate embedding tables:

$$\mathbf{E}^{\text{main}} \in \mathbb{R}^{|\mathcal{V}_{\text{main}}|\times d}, \qquad \mathbf{E}^{\text{aux}} \in \mathbb{R}^{|\mathcal{V}_{\text{aux}}|\times d_a}. \tag{33}$$

The main embedding produces

$$\mathbf{Z}^{\text{main}} = \mathbf{E}^{\text{main}}\left(\mathbf{t}^{\text{main}}\right) \in \mathbb{R}^{B\times S_1\times d}, \tag{34}$$

where d is the model hidden dimension. The auxiliary embedding produces

$$\mathbf{Z}^{\text{aux}} = \mathbf{E}^{\text{aux}}\left(\mathbf{t}^{\text{aux}}\right) \in \mathbb{R}^{B\times S_2\times d_a}, \tag{35}$$

where d_a is the auxiliary embedding dimension. In practice, d_a can be smaller than d to reduce overhead.

Because the auxiliary representation must eventually be combined with the main hidden state, we project it to the model dimension:

$$\mathbf{Z}^{\text{aux},d} = \mathbf{Z}^{\text{aux}}\mathbf{W}_p, \qquad \mathbf{W}_p \in \mathbb{R}^{d_a\times d}, \tag{36}$$

so that

$$\mathbf{Z}^{\text{aux},d} \in \mathbb{R}^{B\times S_2\times d}. \tag{37}$$

A.3 Why Separate Embedding Tables Are Needed

Using a shared embedding table is generally inappropriate because the two tokenizers assign different meanings to token IDs. For example, a BPE token ID may correspond to a subword unit, while the same integer ID under a character tokenizer may correspond to a character. Therefore, the token IDs from $\mathcal{T}_{\text{main}}$ and $\mathcal{T}_{\text{aux}}$ are not semantically aligned. Separate embedding tables allow each tokenizer to learn representations appropriate for its own vocabulary and granularity.

The only case where embedding sharing is natural is when the two tokenizers share the same vocabulary and token-ID semantics. Since MV-HC is designed for heterogeneous tokenization views, we use independent embedding tables by default and only merge the representations after projecting the auxiliary embeddings to the main model dimension.

A.4 Causal Alignment from Auxiliary Tokens to Main Tokens

The main and auxiliary tokenizers can produce different sequence lengths, so the auxiliary representation must be aligned to the main-token positions before stream initialization. Importantly, because MV-HC is used for autoregressive language modeling, this alignment must not expose information from raw-text positions that lie beyond the current main-token prefix.

The alignment module maps

$$\text{Align} : \mathbb{R}^{B\times S_2\times d} \to \mathbb{R}^{B\times S_1\times d}. \tag{38}$$

Given the projected auxiliary representation $\mathbf{Z}^{\text{aux},d}$, we compute

$$\widetilde{\mathbf{Z}}^{\text{aux}} = \text{Align}\left(\mathbf{Z}^{\text{aux},d}, \mathbf{Z}^{\text{main}}\right) \in \mathbb{R}^{B\times S_1\times d}. \tag{39}$$

Because both token sequences are derived from the same raw input, each token can be associated with a character span. Let the i-th main token cover $[a_i, b_i)$ and the j-th auxiliary token cover

$[c_j, d_j)$. At main-token position i, only auxiliary tokens whose complete span has already entered the observed autoregressive prefix are eligible for alignment. We therefore define the causally valid set

$$\mathcal{A}_{\text{causal}}(i) = \{j \mid [a_i, b_i) \cap [c_j, d_j) \neq \emptyset, \quad d_j \leq b_i\}. \tag{40}$$

The condition $d_j \leq b_i$ ensures that an auxiliary token is never exposed before its full raw-text span has been observed. This is particularly important when the auxiliary tokenizer is coarser than the main tokenizer: an auxiliary unit that spans several main tokens becomes available only after its complete span lies inside the current prefix.

Using mean pooling, the aligned auxiliary representation is

$$\widetilde{\mathbf{Z}}_i^{\text{aux}} = \begin{cases} \dfrac{1}{|\mathcal{A}_{\text{causal}}(i)|} \displaystyle\sum_{j \in \mathcal{A}_{\text{causal}}(i)} \mathbf{Z}_j^{\text{aux},d}, & |\mathcal{A}_{\text{causal}}(i)| > 0, \\ \mathbf{0}, & |\mathcal{A}_{\text{causal}}(i)| = 0. \end{cases} \tag{41}$$

The zero vector indicates that no causally valid auxiliary unit is available for that main-token position and avoids introducing information from a future span.

More generally, weighted pooling can be used over the same causally valid set:

$$\widetilde{\mathbf{Z}}_i^{\text{aux}} = \sum_{j \in \mathcal{A}_{\text{causal}}(i)} w_{ij} \mathbf{Z}_j^{\text{aux},d}, \qquad \sum_{j \in \mathcal{A}_{\text{causal}}(i)} w_{ij} = 1. \tag{42}$$

The weights w_{ij} may be uniform, proportional to span overlap, or produced by a lightweight attention-pooling module. In all cases, the causal mask in Eq. equation 40 is applied before pooling.

A.5 Example: BPE Main Tokenizer and Character Auxiliary Tokenizer

Consider the input word

`unhappy`.

Assume that the main BPE tokenizer produces

`["un", "happy"]`,

so that $S_1 = 2$, while a character-level auxiliary tokenizer produces

`["u", "n", "h", "a", "p", "p", "y"]`,

and therefore $S_2 = 7$.

The main-token spans are

$$\texttt{"un"} \to [0, 2), \qquad \texttt{"happy"} \to [2, 7),$$

while the character-token spans are

$$\texttt{"u"} \to [0, 1), \quad \texttt{"n"} \to [1, 2), \quad \texttt{"h"} \to [2, 3), \quad \ldots, \quad \texttt{"y"} \to [6, 7).$$

At the first main-token position, whose observed prefix ends at character boundary $b_1 = 2$, only `u` and `n` satisfy the causal condition $d_j \leq b_1$. Therefore,

$$\widetilde{\mathbf{Z}}_{\texttt{un}}^{\text{aux}} = \frac{\mathbf{z}_{\texttt{u}} + \mathbf{z}_{\texttt{n}}}{2}. \tag{43}$$

The characters `h`, `a`, `p`, `p`, and `y` are not available at this position because they lie outside the observed prefix.

At the second main-token position, the prefix extends through $b_2 = 7$, and the characters overlapping `"happy"` are all causally available:

$$\widetilde{\mathbf{Z}}_{\texttt{happy}}^{\text{aux}} = \frac{\mathbf{z}_{\texttt{h}} + \mathbf{z}_{\texttt{a}} + \mathbf{z}_{\texttt{p}_1} + \mathbf{z}_{\texttt{p}_2} + \mathbf{z}_{\texttt{y}}}{5}. \tag{44}$$

Thus, the auxiliary representation is converted from

$$\mathbb{R}^{B \times 7 \times d} \quad \text{to} \quad \mathbb{R}^{B \times 2 \times d},$$

while preserving the autoregressive information boundary of the main tokenizer.

Coarser auxiliary tokenizers. The causal constraint is especially important when the auxiliary tokenizer is coarser than the main tokenizer. Consider again `unhappy`, but suppose the auxiliary tokenizer represents the complete word as one token,

$$\texttt{"unhappy"} \to [0, 7),$$

while the BPE tokenizer produces

$$\texttt{"un"} \to [0, 2), \qquad \texttt{"happy"} \to [2, 7).$$

Although the auxiliary word overlaps the first BPE token, it is *not* available at that position because its right boundary is $7 > b_1 = 2$. Consequently,

$$\mathcal{A}_{\text{causal}}(1) = \emptyset, \qquad \widetilde{\mathbf{Z}}_1^{\text{aux}} = \mathbf{0}.$$

The word-level representation becomes available only once its full span has entered the autoregressive prefix. Therefore, coarse auxiliary segmentation cannot reveal characters belonging to future main-token positions.

A.6 Stream Initialization After Alignment

After causal alignment, both views have the same sequence length and hidden dimension:

$$\mathbf{Z}^{\text{main}} \in \mathbb{R}^{B \times S_1 \times d}, \qquad \widetilde{\mathbf{Z}}^{\text{aux}} \in \mathbb{R}^{B \times S_1 \times d}. \tag{45}$$

By construction, $\widetilde{\mathbf{Z}}_i^{\text{aux}}$ depends only on auxiliary units whose complete raw-text spans are contained in the prefix available at main-token position i; hence the subsequent stream initialization does not alter the autoregressive information boundary.

They can therefore initialize different groups of MV-HC streams. If $n = n_m + n_a$, then

$$\mathbf{X}_0^{\text{main}} = \text{Repeat}_{n_m}\left(\mathbf{Z}^{\text{main}}\right), \qquad \mathbf{X}_0^{\text{aux}} = \text{Repeat}_{n_a}\left(\widetilde{\mathbf{Z}}^{\text{aux}}\right). \tag{46}$$

The complete stream tensor is

$$\mathbf{X}_0 = \left[\mathbf{X}_0^{\text{main}}, \mathbf{X}_0^{\text{aux}}\right] \in \mathbb{R}^{B \times S_1 \times n \times d}. \tag{47}$$

This tensor is then passed to the MV-HC layers. The main and auxiliary streams evolve jointly through the residual-stream mixer, while the language-modeling objective remains defined exclusively over the main-tokenizer positions.

A.7 Effect of Auxiliary Tokenizer Choice

We further study how the choice of auxiliary tokenizer affects MV-HC when the main tokenizer is fixed to BPE. Since the auxiliary tokenizer is used only to initialize additional hyper-streams, the autoregressive objective and output vocabulary remain defined by the BPE tokenizer. For every auxiliary tokenizer, alignment obeys the causal span constraint described in Appendix A.4: an auxiliary unit can contribute at main-token position i only after its complete raw-text span lies inside the observed prefix. This distinction is mostly trivial for finer-grained byte- and character-level views, but is important for word-level and morphology-aware tokenizers whose units can span multiple BPE positions. We compare byte-level, character-level, morphology-aware, and word-level auxiliary tokenizers under the same model, training budget, and residual stream width.

Figure 3 compares the convergence behavior of different auxiliary tokenizers. Overall, finer-grained auxiliary views perform better in this setting. Byte-level and character-level views achieve lower training loss than the word-level view. Besides providing complementary orthographic and boundary-level information, fine-grained tokenizers have the practical advantage that their units are naturally contained within short main-token spans and therefore provide dense causally available auxiliary signals. In contrast, a coarser word- or morphology-level auxiliary unit is available only after its full span has entered the autoregressive prefix, which can make the auxiliary signal sparser at earlier positions.

We do not interpret this result as evidence that one auxiliary tokenizer is universally superior. The relative usefulness of each view is likely data- and domain-dependent. Byte-level tokenization is

Table 6: Ablation on auxiliary tokenizer choice with BPE as the main tokenizer. Results are reported for the 1.5B model with $n = 4$ residual streams.

Method	Auxiliary Tokenizer	Train Loss ↓	BPB ↓	CORE ↑
KromHC	None	2.481	0.628	55.2
MV-HC	Word-level	2.474	0.625	55.6
MV-HC	Morphology-aware	2.466	0.622	56.0
MV-HC	Character-level	2.458	0.619	56.4
MV-HC	Byte-level	**2.452**	**0.617**	**56.7**

Table 7: Controlled ablations on the 186M model ($n = 4$). *Het.* denotes heterogeneous main/auxiliary hyper-streams.

Method	Aux	Routing	BPB ↓	CORE ↑
Residual	N	–	0.721	46.2
+ AuxConcat	Y	–	0.714	46.8
+ GatedAux	Y	–	0.711	47.1
KromHC	N	HC	0.704	47.7
+ gate	N	HC	0.702	47.9
+ ParamMatch	N	HC	0.703	47.8
+ AuxFuse	Y	HC	0.699	48.2
MV-HC w/o gate	Y	HC+Het.	0.702	48.2
Full MV-HC	Y	HC+Het.	**0.696**	**48.6**

a general option because it can represent arbitrary text without out-of-vocabulary issues, while character-level tokenization provides an interpretable fine-grained view. Morphology-aware tokenization may be more useful for morphologically rich languages or domains where word structure is important, whereas word-level tokenization can be less complementary to BPE and produces causally available representations less frequently when one word spans several main-token positions.

A.8 Isolating the source of MV-HC gains

To determine whether the improvement of MV-HC comes specifically from maintaining heterogeneous views in separate hyper-connection streams, rather than from auxiliary information, additional parameters, or gated fusion alone, we compare a set of controlled variants in Table 7. Adding the auxiliary view through simple input concatenation already improves the residual baseline from 0.721 to 0.714 BPB and from 46.2 to 46.8 CORE, confirming that the auxiliary tokenization contains useful complementary information. A stronger gated-fusion variant further improves to 0.711 BPB and 47.1 CORE, but remains clearly below KromHC (0.704/47.7), indicating that auxiliary fusion alone does not account for the gains of MV-HC. Similarly, adding the final gate to KromHC produces only a small improvement ($0.704 \rightarrow 0.702$ BPB and $47.7 \rightarrow 47.9$ CORE), while the parameter-matched KromHC control remains close to the original model (0.703/47.8). These results suggest that neither the gate nor the additional trainable capacity is sufficient to explain the improvement. Most importantly, *KromHC+AuxFuse*, which uses the same auxiliary view but fuses it before stream expansion so that all streams share the same fused representation, reaches 0.699 BPB and 48.2 CORE. Thus, providing auxiliary information to KromHC is beneficial, but still falls short of full MV-HC (0.696/48.6), where the main and auxiliary representations remain distinct across the residual streams. Moreover, MV-HC without the final gate already reaches 0.702 BPB and 48.2 CORE, showing that heterogeneous multi-view routing itself provides a substantial benefit over KromHC. The full model achieves the best result, improving KromHC by 0.008 BPB and 0.9 CORE points overall.

Table 8: Late-layer residual-stream similarity across context lengths. Results are reported on the 186M model with n = 4. Values are mean pairwise cosine similarity averaged over the final four Transformer layers; lower is more diverse.

Context	mHC	KromHC	**MV-HC**
4K	0.916	0.889	**0.786**
8K	0.914	0.897	**0.792**
16K	0.924	0.906	**0.799**
32K	0.929	0.911	**0.806**
128K	0.934	0.917	**0.817**
256K	0.937	0.919	**0.824**

A.9 Residual-Stream Representation Diversity

A central motivation of MV-HC is that hyper-connection streams should differ not only in their residual routing, but also in the information they carry. Existing HC variants initialize their streams from the same hidden representation, whereas MV-HC initializes different stream groups from the main and auxiliary input views. We therefore directly analyze whether this difference produces persistent representational diversity across depth.

For each layer l, let $\mathbf{X}_l^{(k)} \in \mathbb{R}^{B\times S\times d}$ denote the k-th residual stream before the $\mathbf{H}_l^{\text{pre}}$ aggregation. For every pair of distinct streams $i \neq j$, we compute token-wise cosine similarity

$$c_{l,t}^{(i,j)} = \frac{\mathbf{X}_{l,t}^{(i)\top}\mathbf{X}_{l,t}^{(j)}}{\|\mathbf{X}_{l,t}^{(i)}\|_2\|\mathbf{X}_{l,t}^{(j)}\|_2 + \epsilon}. \tag{48}$$

We first average over valid token positions within each sequence, then over the $\binom{n}{2}$ unique stream pairs, and finally equally over evaluation samples. Thus, sequences contribute equally to the reported mean regardless of their length. Lower cosine similarity corresponds to greater residual-stream diversity. We evaluate matched 186M checkpoints with $n = 4$ streams on held-out sequences with context lengths of 4K, 8K, 16K, 32K, 128K, and 256K tokens. Long-context settings beyond the native training length are treated as extrapolation tests when applicable. Importantly, similarity is measured on the expanded residual tensor $\mathbf{X}_l$ before stream contraction, rather than on the single hidden representation passed to the Transformer branch. As shown in Table 8, MV-HC exhibits consistently lower pairwise stream similarity than mHC and KromHC throughout the last four layers. In other words, the average similarity over the last four layers remains substantially lower for MV-HC across all evaluated context lengths.

A.10 Effect of Auxiliary Fusion Baseline

To distinguish the effect of multi-view input fusion from the effect of multi-view hyper-connections, we include a standard residual baseline with auxiliary fusion. In this variant, the auxiliary representation is aligned to the main-token sequence, concatenated with the main embedding, and projected back to the model dimension:

$$\mathbf{Z}^{\text{fuse}} = \text{Proj}\left(\left[\mathbf{Z}^{\text{main}}, \widetilde{\mathbf{Z}}^{\text{aux}}\right]\right) \in \mathbb{R}^{B\times S_1\times d}.$$

The resulting representation is then passed to a standard Transformer with ordinary residual connections. This baseline uses the same auxiliary tokenizer and alignment module as MV-HC, but does not use expanded hyper-streams or KromHC-style residual mixing. Table 9 compares MV-HC with a simple auxiliary-fusion baseline. The Residual + AuxConcat variant benefits from the auxiliary tokenizer view, indicating that the aligned auxiliary representation itself provides useful information. However, it remains below KromHC and full MV-HC, suggesting that simple early fusion is not sufficient to explain the gains of MV-HC. Compared with AuxConcat, MV-HC keeps the auxiliary information in separate hyper-streams and allows it to interact with the main streams through constrained residual mixing across layers. This suggests that the improvement is due to the combination of auxiliary-view information and multi-stream hyper-connection routing, rather than only the presence of an additional tokenizer.

Table 9: Ablation separating simple multi-view input fusion from multi-view hyper-connection routing. Results are reported on the 186M model with $n = 4$ streams for HC-based methods.

Variant	Aux. View	HC Routing	BPB ↓	CORE ↑
Residual	No	No	0.721	46.2
Residual + AuxConcat	Yes	No	0.714	46.8
KromHC	No	Yes	0.704	47.7
MV-HC w/o gate	Yes	Yes	0.702	48.2
Full MV-HC	Yes	Yes	**0.696**	**48.6**

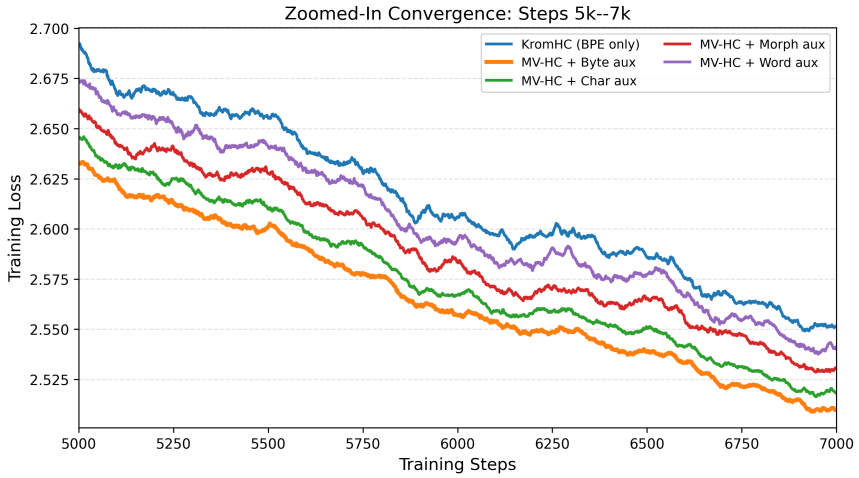

Figure 3: Zoomed-in convergence comparison for MV-HC with a BPE main tokenizer and different auxiliary tokenizers. The auxiliary tokenizer affects the training trajectory, with byte-level and character-level views showing slightly lower loss in this setting. We treat this as an empirical trend rather than a universal ranking, since the best auxiliary view may depend on the dataset, language, and domain.

A.11 Complexity and Implementation Overhead

This section discusses the computational and memory overhead of MV-HC. We separate the cost into four parts: the main Transformer branch, the residual-stream mixer, the auxiliary-tokenizer pathway, and the gated stream reduction.

A.12 Main Transformer Branch

MV-HC keeps the expensive Transformer computation unchanged relative to the underlying HC/KromHC model. At each layer, the multi-stream tensor

$$\mathbf{X}_l \in \mathbb{R}^{B\times S_1\times n\times d}$$

is first aggregated by the pre-mixing module into

$$\mathbf{U}_l = \mathbf{H}_l^{\text{pre}}\mathbf{X}_l \in \mathbb{R}^{B\times S_1\times d},$$

where the multiplication denotes contraction over the stream dimension. The main branch $\mathcal{F}_l$, e.g., attention and MLP, is applied only to $\mathbf{U}_l$:

$$\mathbf{Y}_l = \mathcal{F}_l(\mathbf{U}_l).$$

Therefore, attention and MLP are not applied independently to all n streams. As a result, the dominant Transformer FLOPs remain the same order as the base model:

$$\mathcal{O}(BS_1^2d) \quad \text{for attention}$$

and

$$\mathcal{O}(BS_1 d d_{\text{ff}}) \quad \text{for the MLP,}$$

where d_{ff} is the feed-forward hidden dimension.

A.13 Residual-Stream Mixing Cost

The additional HC-style computation comes from mixing over the residual-stream dimension. For a dense stream mixer, the cost scales as

$$\mathcal{O}(BS_1 d n^2),$$

because an $n \times n$ stream-mixing matrix is applied to n streams at every token position and hidden dimension. If MV-HC keeps the same total number of streams as the KromHC baseline and only reallocates them into main-view and auxiliary-view groups, i.e.,

$$n = n_m + n_a$$

is fixed, then the residual-stream mixing cost is essentially unchanged. The block notation

$$\mathbf{H}_l^{\text{res}} = \begin{bmatrix} \mathbf{H}_l^{mm} & \mathbf{H}_l^{ma} \\ \mathbf{H}_l^{am} & \mathbf{H}_l^{aa} \end{bmatrix}$$

does not by itself add computation; it only assigns semantic roles to stream groups. If auxiliary streams are added on top of an existing n_m-stream model, the total stream width becomes $n = n_m + n_a$. For a dense mixer, the residual-stream mixing cost increases from

$$\mathcal{O}(BS_1 d n_m^2)$$

to

$$\mathcal{O}(BS_1 d (n_m + n_a)^2).$$

The extra dense mixing cost is therefore

$$\mathcal{O}\big(BS_1 d (2 n_m n_a + n_a^2)\big)\,.$$

In our implementation, MV-HC is built on KromHC because its Kronecker-product parameterization scales more favorably with the number of streams than dense mHC-style residual matrices or mHC-lite permutation combinations.

A.14 Auxiliary Path Cost

The auxiliary pathway introduces three sources of overhead: auxiliary tokenization, auxiliary embedding/projection, and alignment. Let S_2 be the auxiliary-token sequence length and d_a be the auxiliary embedding dimension. The auxiliary embedding lookup produces

$$\mathbf{Z}^{\text{aux}} \in \mathbb{R}^{B \times S_2 \times d_a},$$

and the projection to the main hidden dimension costs approximately

$$\mathcal{O}(BS_2 d_a d)$$

for a full linear projection from d_a to d. In practice, d_a can be chosen much smaller than d to reduce this overhead.

For alignment, we use a span-based gather-pooling operation. If each main token attends to at most K auxiliary tokens, then the alignment produces

$$\widetilde{\mathbf{Z}}^{\text{aux}} \in \mathbb{R}^{B \times S_1 \times d}$$

with cost approximately

$$\mathcal{O}(BS_1 K d).$$

For character-level or byte-level auxiliary tokenizers, S_2 can be larger than S_1, but K is usually small when alignment is based on character-span overlap. Thus, the online alignment operation is lightweight compared with the attention and MLP branches, while the main additional model-side cost comes from the auxiliary embedding/projection path.

Auxiliary tokenization and construction of alignment indices can be performed offline during dataset preprocessing. Specifically, for each training example we can precompute and store

`input_ids`, `aux_input_ids`, `align_index`, `align_mask`,

and optionally `align_weight`. During training, the dataloader only loads these tensors, and the model performs auxiliary embedding lookup, projection, and gather-pooling. If auxiliary tokenization and alignment construction are performed online inside the dataloader, their CPU-side cost may be non-negligible, especially for byte-level or character-level tokenizers. Therefore, we recommend offline preprocessing for large-scale training.

A.15 Gated Reduction Cost

Standard KromHC reduces the final streams by summing over the stream dimension. MV-HC replaces this with a gated reduction:

$$\mathbf{Z}^{\text{out}} = \sum_{k=1}^{n} \alpha_L^{(k)} \odot \mathbf{X}_L^{(k)}.$$

In our default implementation, the gate is a token-wise linear projection from $\mathbb{R}^d$ to a scalar score for each stream, followed by a softmax over the stream dimension. The gate computation costs approximately

$$\mathcal{O}(BS_1nd),$$

and the weighted stream summation has the same order. For small stream counts, this overhead is minor compared with attention and MLP computation.

A.16 Activation Memory

MV-HC increases activation memory mainly through the expanded stream tensor and the auxiliary pathway. The stream tensor stores activations of size

$$\mathcal{O}(BS_1nd),$$

which is the same order as KromHC for the same total stream width n. If MV-HC keeps n fixed and only reallocates streams into main and auxiliary groups, the stream activation memory is unchanged relative to KromHC. If MV-HC adds auxiliary streams on top of the original streams, activation memory increases linearly with the new total number of streams $n_m + n_a$.

The auxiliary pathway additionally stores

$$\mathcal{O}(BS_2d_a)$$

for auxiliary embeddings and

$$\mathcal{O}(BS_1d)$$

for the aligned auxiliary representation. The alignment metadata requires

$$\mathcal{O}(BS_1K)$$

memory for `align_index` and `align_mask`, where K is the maximum number of auxiliary tokens assigned to each main token. Compared with Transformer activations, this metadata cost is usually small.

A.17 MV-HC Algorithm

Algorithm 1 summarizes the forward computation of MV-HC. The algorithm starts from a raw input sequence x and constructs two complementary views of the same input. The main tokenizer $\mathcal{T}_{\text{main}}$ produces the standard language-model token sequence, while the auxiliary tokenizer $\mathcal{T}_{\text{aux}}$ produces an alternative tokenization view. After embedding, these two paths produce $\mathbf{Z}^{\text{main}} \in \mathbb{R}^{B\times S_1\times d}$ and $\mathbf{Z}^{\text{aux}} \in \mathbb{R}^{B\times S_2\times d}$, respectively. Since the two tokenizers may generate different sequence lengths, the auxiliary representation is aligned to the main-token positions through $\text{Align}(\cdot)$, yielding $\widetilde{\mathbf{Z}}^{\text{aux}} \in \mathbb{R}^{B\times S_1\times d}$.

After alignment, MV-HC initializes the residual streams from both views. The main representation is repeated into n_m main-view streams, while the aligned auxiliary representation is repeated into n_a auxiliary-view streams. The complete initial stream tensor is then formed as

$$\mathbf{X}_0 = [\mathbf{X}_0^{\text{main}}, \mathbf{X}_0^{\text{aux}}] \in \mathbb{R}^{B\times S_1\times n\times d}, \qquad n = n_m + n_a.$$

This step is the main difference from standard KromHC, where all n streams are initialized by repeating the same hidden representation. In MV-HC, different stream groups carry different input-derived information from the beginning.

For each layer l, the multi-view stream tensor is first aggregated by the pre-mixing weights $\mathbf{H}_l^{\text{pre}}$. This aggregation contracts over the stream dimension:

$$\mathbf{U}_l = \sum_{k=1}^{n} H_{l,k}^{\text{pre}} \mathbf{X}_l^{(k)} \in \mathbb{R}^{B\times S_1\times d}.$$

Thus, the main branch $\mathcal{F}_l$, such as attention, MLP, or a full Transformer block, receives a single standard hidden representation rather than processing all streams independently. The branch output is then written back into the stream space through $\mathbf{H}_l^{\text{post}}$, while the previous streams are propagated through the KromHC residual mixer $\mathbf{H}_l^{\text{res}}$. The layer update is

$$\mathbf{X}_{l+1} = \mathbf{H}_l^{\text{res}} \mathbf{X}_l + \mathbf{H}_l^{\text{post}\top} \mathbf{Y}_l.$$

This preserves the KromHC-style stable residual propagation while allowing main-view and auxiliary-view streams to interact across layers.

After the final layer, MV-HC reduces the stream tensor back to the standard language-model hidden shape. Instead of using the uniform sum reduction used in KromHC, MV-HC computes token-wise gates over the stream dimension:

$$\boldsymbol{\alpha}_L = \text{Softmax}_n(\text{Gate}(\mathbf{X}_L)).$$

Here, $\text{Softmax}_n(\cdot)$ denotes normalization across the n streams, and $\text{Gate}(\cdot)$ is implemented as a lightweight token-wise linear projection by default. The final hidden state is obtained by the weighted stream sum

$$\mathbf{Z}^{\text{out}} = \sum_{k=1}^{n} \boldsymbol{\alpha}_L^{(k)} \odot \mathbf{X}_L^{(k)} \in \mathbb{R}^{B\times S_1\times d}.$$

This gated reduction allows the model to adaptively control how much information to use from each main or auxiliary stream at each token position. Finally, $\mathbf{Z}^{\text{out}}$ is passed to the standard LM head to produce output probabilities. Therefore, MV-HC keeps the main tokenizer as the prediction interface while using the auxiliary tokenizer only to enrich the internal residual-stream representation.

Algorithm 1 Multi-View Hyper-Connections (MV-HC)

Require: Input x, tokenizers $\mathcal{T}_{\text{main}}, \mathcal{T}_{\text{aux}}$, layers $\{\mathcal{F}_l\}_{l=0}^{L-1}$, streams $n = n_m + n_a$.
Ensure: Output probabilities $\mathbf{P}$.

$\mathbf{t}^{\text{main}} \leftarrow \mathcal{T}_{\text{main}}(x)$, $\mathbf{t}^{\text{aux}} \leftarrow \mathcal{T}_{\text{aux}}(x)$.
$\mathbf{Z}^{\text{main}} \leftarrow \mathbf{E}^{\text{main}}(\mathbf{t}^{\text{main}}) \in \mathbb{R}^{B\times S_1\times d}$.
$\mathbf{Z}^{\text{aux}} \leftarrow \mathbf{E}^{\text{aux}}(\mathbf{t}^{\text{aux}}) \in \mathbb{R}^{B\times S_2\times d_a}$.
$\mathbf{Z}^{\text{aux},d} \leftarrow \mathbf{Z}^{\text{aux}}\mathbf{W}_p \in \mathbb{R}^{B\times S_2\times d}$.
$\widetilde{\mathbf{Z}}^{\text{aux}} \leftarrow \text{Align}(\mathbf{Z}^{\text{aux},d}, \mathbf{Z}^{\text{main}}) \in \mathbb{R}^{B\times S_1\times d}$.
$\mathbf{X}_0^{\text{main}} \leftarrow \text{Repeat}_{n_m}(\mathbf{Z}^{\text{main}})$.
$\mathbf{X}_0^{\text{aux}} \leftarrow \text{Repeat}_{n_a}(\widetilde{\mathbf{Z}}^{\text{aux}})$.
$\mathbf{X}_0 \leftarrow [\mathbf{X}_0^{\text{main}}, \mathbf{X}_0^{\text{aux}}] \in \mathbb{R}^{B\times S_1\times n\times d}$.
for $l = 0, \ldots, L-1$ **do**
    $\mathbf{U}_l \leftarrow \sum_{k=1}^{n} H_{l,k}^{\text{pre}} \mathbf{X}_l^{(k)}$.
    $\mathbf{Y}_l \leftarrow \mathcal{F}_l(\mathbf{U}_l)$.
    $\mathbf{X}_{l+1} \leftarrow \mathbf{H}_l^{\text{res}} \mathbf{X}_l + \mathbf{H}_l^{\text{post}\top} \mathbf{Y}_l$.
end for
$\boldsymbol{\alpha}_L \leftarrow \text{Softmax}_n(\text{Gate}(\mathbf{X}_L))$.
$\mathbf{Z}^{\text{out}} \leftarrow \sum_{k=1}^{n} \boldsymbol{\alpha}_L^{(k)} \odot \mathbf{X}_L^{(k)}$.
$\mathbf{P} \leftarrow \text{Softmax}(\mathbf{Z}^{\text{out}}\mathbf{W}_{\text{vocab}})$.
return $\mathbf{P}$.