Attention Residuals

TECHNICAL REPORT OF ATTENTION RESIDUALS

Kimi Team
https://github.com/MoonshotAI/Attention-Residuals

Presented by: Vardaan Pahuja

Recap: Residual Connections

Instead of forcing a stack of layers to learn a full transformation **H(x)**, we let it learn only a correction to the input: **H(x)=x+F(x)**,

1. **Easier optimization**: hard for ordinary stacked layers to learn an exact identity map

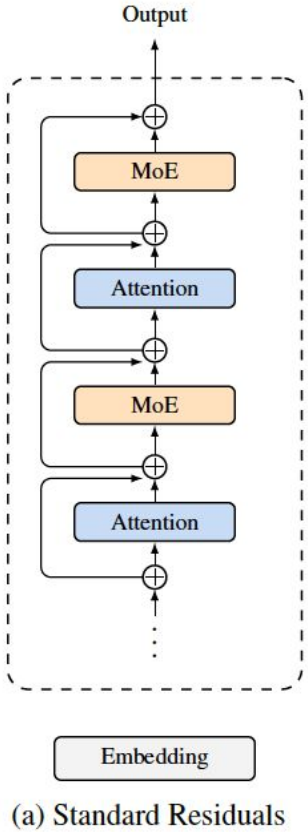
2. **Better gradient flow**: Residual paths provide a more direct route for gradients, so early layers can still receive useful learning signals

Motivation

- Standard PreNorm Transformer

$$h_l = h_{l-1} + f_l(h_{l-1}) \quad h_l = h_1 + \sum_{i=1}^{l-1} f_i(h_i)$$

Unrolled view



(a) Standard Residuals

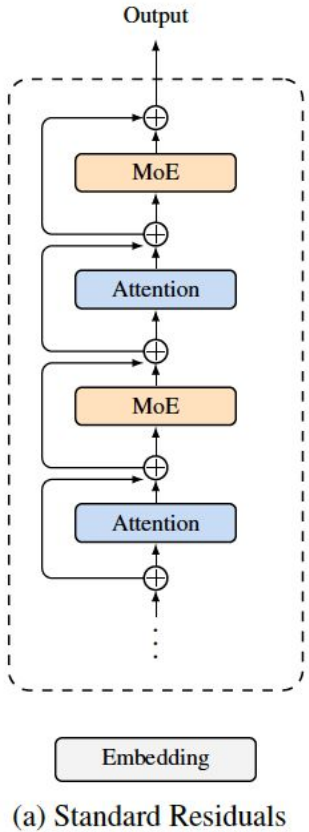
Motivation

- Standard PreNorm Transformer

$$h_l = h_{l-1} + f_l(h_{l-1}) \qquad h_l = h_1 + \sum_{i=1}^{l-1} f_i(h_i)$$

Unrolled view

- **Drawbacks**:
  - **No selective access**: different layers may want different earlier information, but they all receive the same accumulated state.



(a) Standard Residuals

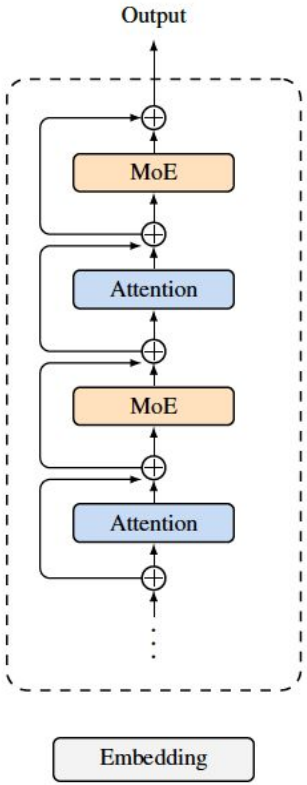
Motivation

- Standard PreNorm Transformer

$$h_l = h_{l-1} + f_l(h_{l-1}) \qquad h_l = h_1 + \sum_{i=1}^{l-1} f_i(h_i)$$

Unrolled view

- **Drawbacks**:
  - **No selective access**: different layers may want different earlier information, but they all receive the same accumulated state.
  - **Irreversible mixing**: once earlier outputs are merged, later layers cannot choose specific ones back out.



(a) Standard Residuals

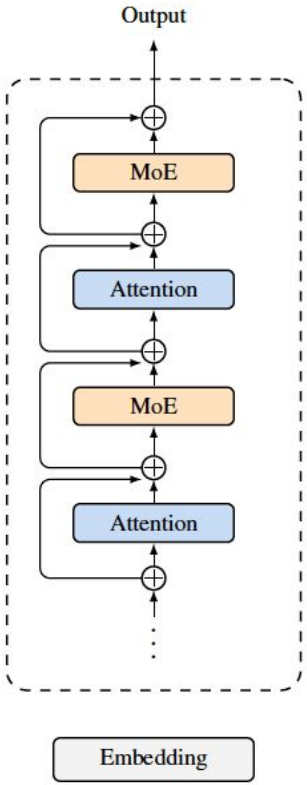
Motivation

- Standard PreNorm Transformer

$$h_l = h_{l-1} + f_l(h_{l-1}) \qquad h_l = h_1 + \sum_{i=1}^{l-1} f_i(h_i)$$

Unrolled view

- **Drawbacks**:
  - **No selective access**: different layers may want different earlier information, but they all receive the same accumulated state.
  - **Irreversible mixing**: once earlier outputs are merged, later layers cannot choose specific ones back out.
  - **Output growth**: hidden-state magnitude tends to increase with depth, so later layers must produce larger outputs from normalized inputs just to stay influential



(a) Standard Residuals

How did we get here?

- RNNs compress the past into one recurrent state
- Standard residuals compress the network's past over depth into one state

$$s_t = g(s_{t-1}, x_t)$$

How did we get here?

- RNNs compress the past into one recurrent state
- Standard residuals compress the network's past over depth into one state
- Transformers beat RNNs in sequence modeling by replacing **recurrence** with **attention over previous tokens**

$$s_t = g(s_{t-1}, x_t) \longrightarrow y_t = \sum_{j \le t} \alpha_{j \to t} \, v_j$$

Key upgrade: data-dependent weights (alpha) over many past states

Key Idea

- RNNs compress the past into one recurrent state
- Standard residuals compress the network's past over depth into one state
- Transformers beat RNNs in sequence modeling by replacing **recurrence** with **attention over previous tokens**
- **Attention Residuals** applies that same idea **across layers**

$$h_l = h_1 + \sum_{i=1}^{l-1} f_i(h_i) \longrightarrow h_l = \sum_{i<l} \alpha_{i \to l} \, v_i$$

Key upgrade: learn alpha to route information across depth (rather than using uniform weighting)

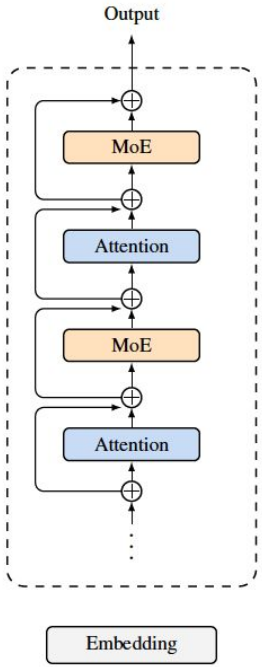
Key Idea

$$h_l = h_1 + \sum_{i=1}^{l-1} f_i(h_i)$$

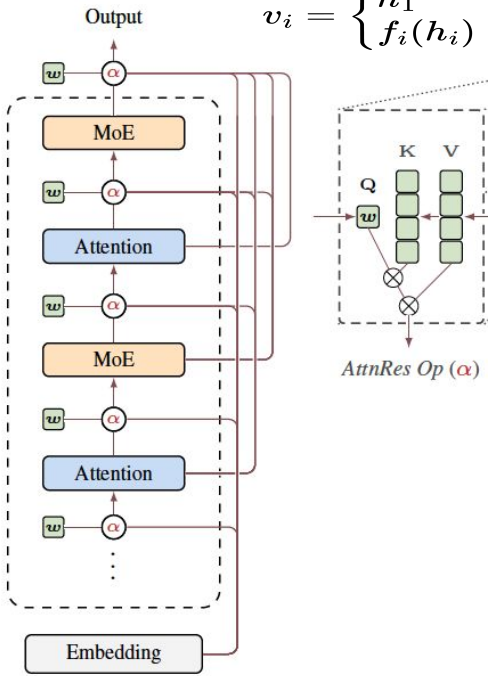
Key upgrade: learn alpha to route information across depth

$$h_l = \sum_{i<l} \alpha_{i \to l} \, v_i$$

$$v_i = \begin{cases} h_1 & i = 0 \\ f_i(h_i) & 1 \le i \le l-1 \end{cases}$$



(a) Standard Residuals



(b) Full Attention Residuals

Key Idea

So a layer can learn something like:

- “*for this computation, I mostly need the embedding and a mid-layer attention output*”
- “*for this token, I care more about recent layers*”
- “*for this MLP block, I want a different mixture than the self-attention block*”

Math Formulation

Define values over depth:

$$\boldsymbol{k}_i = \boldsymbol{v}_i = \begin{cases} \boldsymbol{h}_1 & i = 0 \\ f_i(\boldsymbol{h}_i) & 1 \le i \le l-1 \end{cases}$$

Layer-specific pseudo-query:

$$q_l = w_l \quad \text{(learned vector)}$$

Softmax attention weights over depth (with RMSNorm on keys):

$$\alpha_{i \to l} = \frac{\exp(q_l^\top \, \mathrm{RMSNorm}(v_i))}{\sum_{j=0}^{l-1} \exp(q_l^\top \, \mathrm{RMSNorm}(v_j))}$$

Layer input is a weighted sum: $\boldsymbol{h}_l = \sum_{i<l} \alpha_{i \to l} \, \boldsymbol{v}_i$

Intuition

- Depth becomes a “**memory**” of intermediate representations.
- Each layer learns where to read from that memory (**selective routing**).
- **RMSNorm** prevents large-magnitude layers from dominating the mix.
- In vanilla training, storing these activations overlaps with backprop (little extra memory).

Complexity (AttenRes)

Compute/memory (per token):

- **Arithmetic** ~ $O(L^2 d)$
- **Memory** ~ $O(Ld)$

Depth is usually < 1000 → arithmetic is modest

- Systems overhead dominates at scale.

Complexity (AttenRes)

Compute/memory (per token):

- **Arithmetic** ~ $O(L^2 d)$
- **Memory** ~ $O(Ld)$

Depth is usually < 1000 → arithmetic is modest

- Systems overhead dominates at scale.
- How can we reduce the overhead?

Complexity (Block AttnRes)

- **Partition layers into N blocks, summarize within blocks, attend only over block summaries**

Compute/memory (per token):
• **Arithmetic** ~ $O(L^2d)$ -> **$O(N^2d)$**

• **Memory** ~ $O(Ld)$ -> **$O(Nd)$**

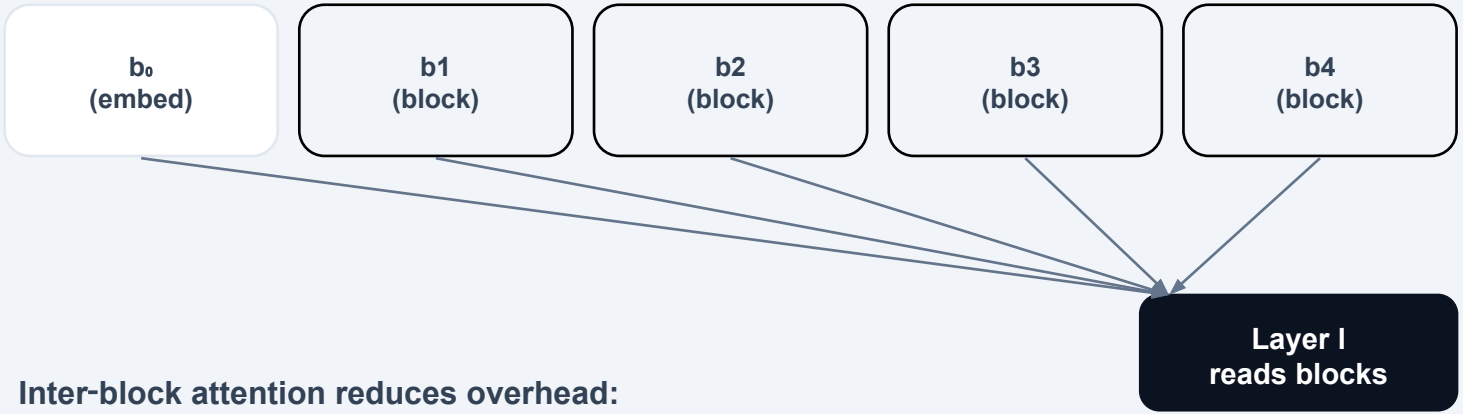
Depth is usually < 1000 → arithmetic is modest

- Systems overhead dominates at scale.

Block Attention Residuals

- Partition layers into N blocks, summarize within blocks, attend only over block summaries (plus token embedding).

Intra-block accumulation: $$b_n = \sum_{j \in B_n} f_j(h_j)$$



Inter-block attention reduces overhead:

- Attend over **N block summaries** (plus token embedding) instead of **L layer outputs**.
- Memory + pipeline communication drop from O(Ld) → O(Nd).
- Empirically, ~8 blocks recovers most of Full AttnRes gains.

Math Formulation

- For the i-th layer in block n, the value matrix is

$$\mathbf{V} = \begin{cases} [\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1}]^\top & \text{if } i = 1 \text{ (first layer of block } n) \\ [\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1}, \boldsymbol{b}_n^{i-1}]^\top & \text{if } i \geq 2 \text{ (subsequent layers)} \end{cases}$$

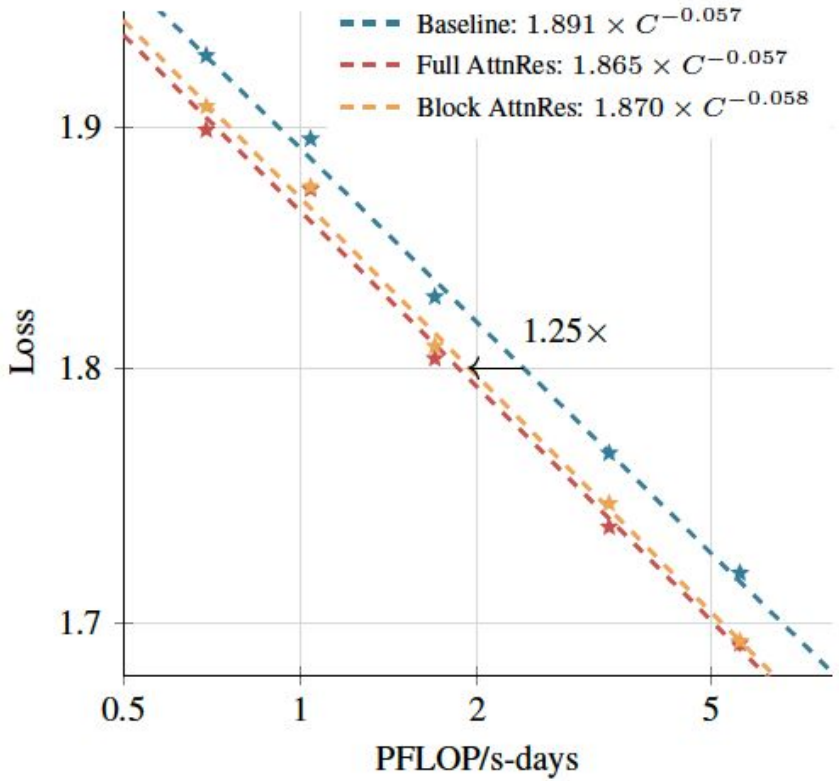
- In each block, the first layer receives the previous block representations and the token embeddings
- Subsequent layers in each block additionally attend to the partial sum b^{i−1}_n.

Training Efficiency

- At scale, activation recomputation + pipeline parallelism mean you don’t keep every layer output by default.
- Full AttnRes would require preserving + communicating all L layer outputs → O(Ld) cross-stage traffic.
- **Cache-based pipeline communication**: avoid redundant transfers by caching block summaries received during earlier stages

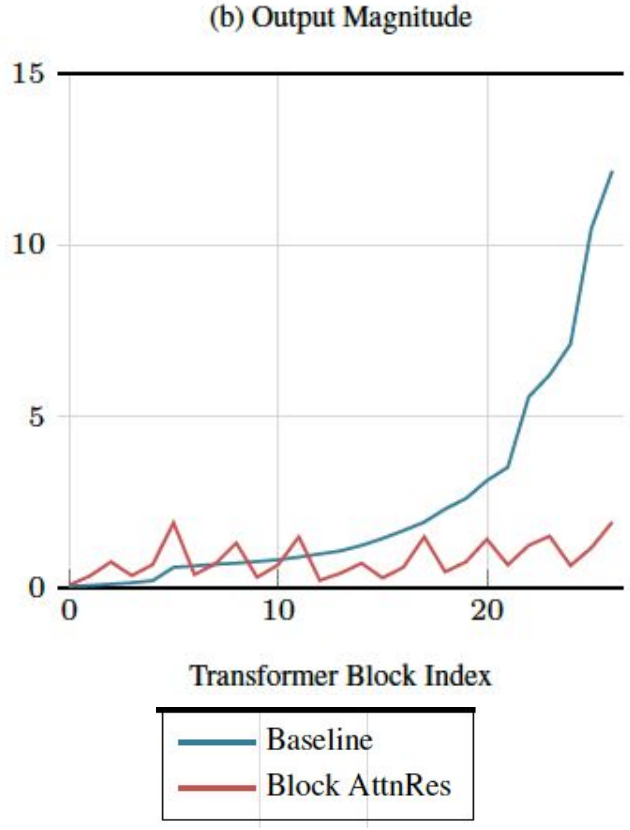
Scaling curves

- Both Full and Block AttnRes consistently outperform the baseline across all scales.
- Block AttnRes closely tracks Full AttnRes, recovering most of the gain at the largest scale.



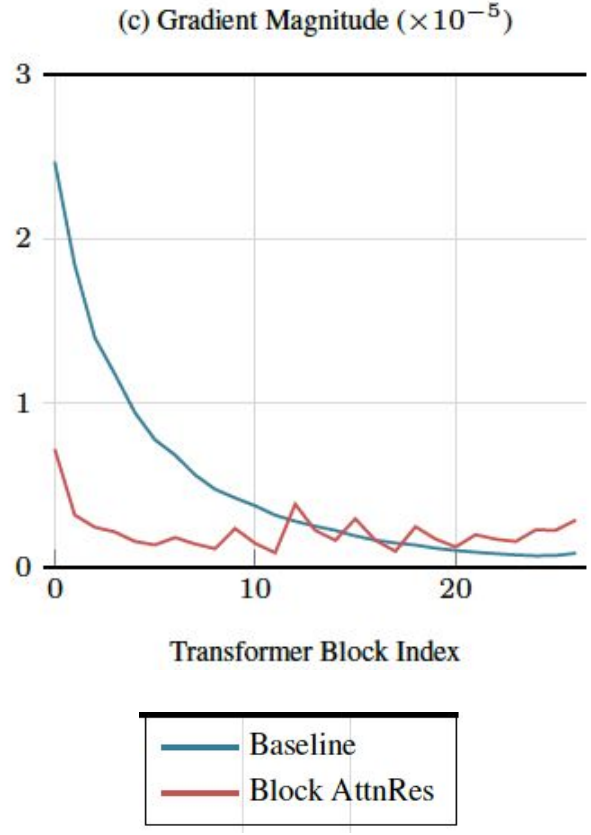
Training Dynamics: Output Magnitude

- As hidden-state magnitudes grow monotonically with depth, **deeper layers are compelled to learn increasingly large outputs** from fixed-scale normalized inputs to remain influential.
- Block AttnRes addresses this using selective aggregation



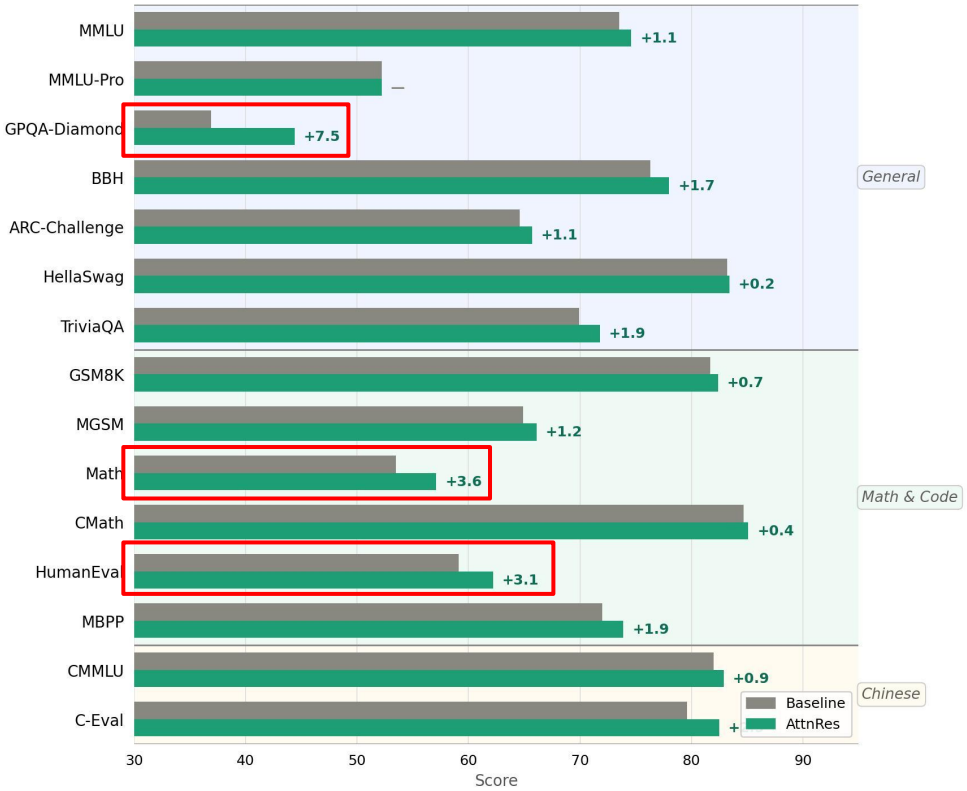
Training Dynamics: Gradient Magnitude

- **Baseline**: early layers sit underneath many downstream additions, so they can receive a large accumulated gradient signal simply because they are included in almost every later state
- Different layers can emphasize different block summaries or the embedding, which spreads influence more selectively rather than uniformly



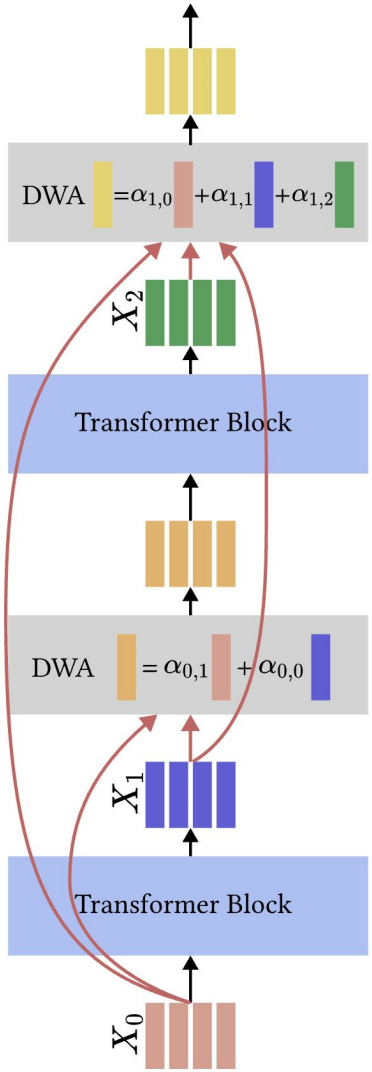
Downstream Results: biggest gains on multi-step reasoning

- More pronounced improvement on multi-step reasoning tasks (GPQA-Diamond, Minerva Math) as well as code generation (HumanEval).
- **Hypothesis**: improved depth-wise information flow benefits compositional tasks
- Later layers can selectively retrieve and build upon earlier representations.



Baseline: DenseFormer

- **DenseFormer** [1] grants each layer access to all previous outputs but combines them with fixed, **input-independent** scalar coefficients.



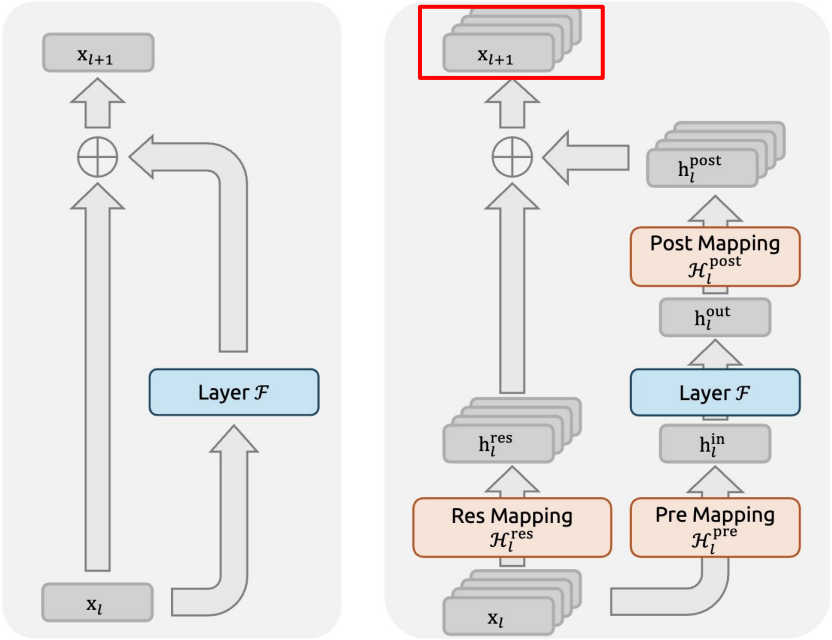
[1] Pagliardini, Matteo, et al. "Denseformer: Enhancing information flow in transformers via depth weighted averaging." NeurIPS 2024.

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

Baseline: Hyper-connections (ByteDance)

Instead of carrying a single hidden state through depth, the model carries **n parallel hidden vectors** per layer.

$$\mathbf{x}_{l+1} = \mathcal{H}_l^{\text{res}}\mathbf{x}_l + \mathcal{H}_l^{\text{post}\,\top}\mathcal{F}(\mathcal{H}_l^{\text{pre}}\mathbf{x}_l, \mathcal{W}_l)$$



(a) Residual Connection (b) Hyper-Connections (HC)

Zhu, Defa, et al. "Hyper-connections." ICLR 2025.

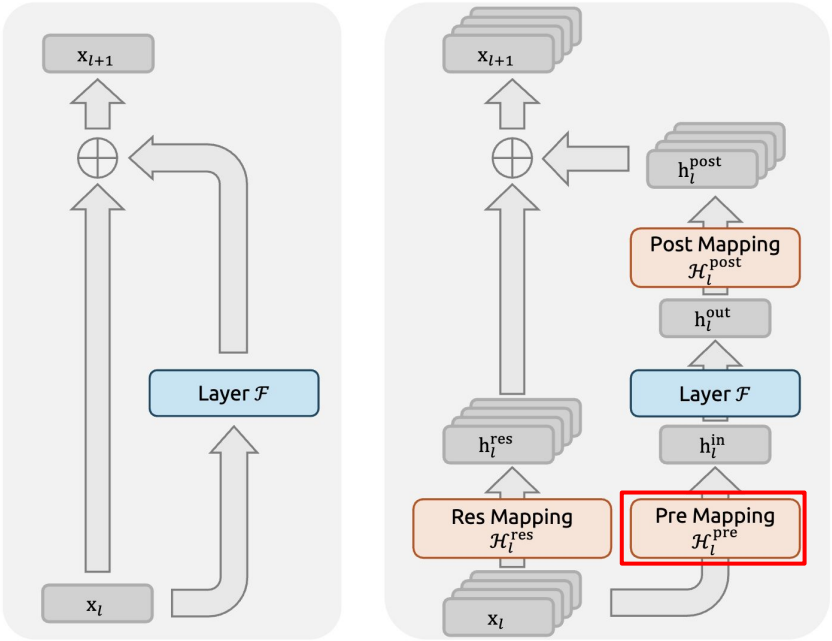
$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$$

Baseline: Hyper-connections (ByteDance)

Instead of carrying a single hidden state through depth, the model carries **n parallel hidden vectors** per layer. Then each layer can:

1. **Mix those hidden vectors to form the actual input** to the current layer (H^{pre}_l ∈ R^{1×n})

$$\mathbf{x}_{l+1} = \mathcal{H}_l^{\text{res}}\mathbf{x}_l + \mathcal{H}_l^{\text{post}\,\top}\mathcal{F}(\mathcal{H}_l^{\text{pre}}\mathbf{x}_l, \mathcal{W}_l)$$



(a) Residual Connection (b) Hyper-Connections (HC)

Zhu, Defa, et al. "Hyper-connections." ICLR 2025.

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

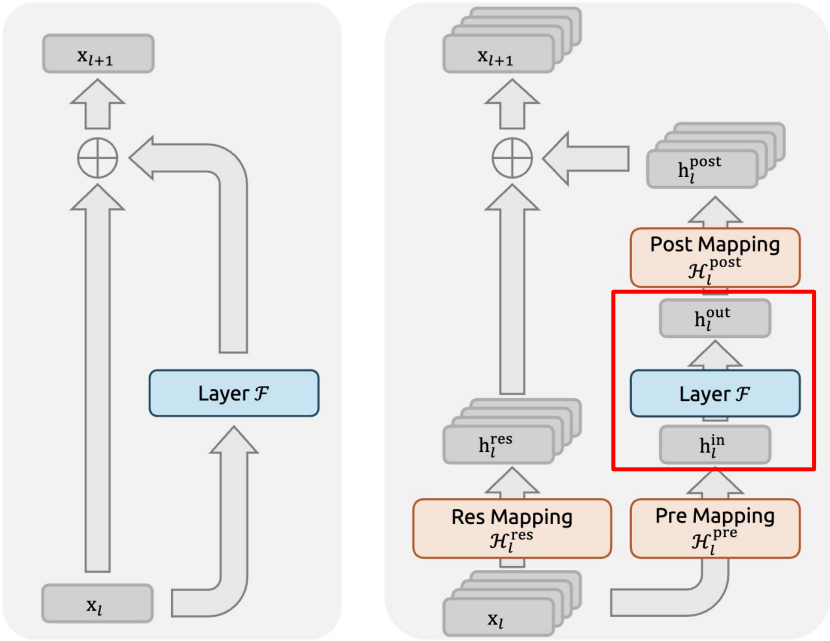
Baseline: Hyper-connections (ByteDance)

Instead of carrying a single hidden state through depth, the model carries **n parallel hidden vectors** per layer. Then each layer can:

1. **Mix those hidden vectors to form the actual input** to the current layer (H^{pre}_l ∈ R^{1×n})

2. **Apply the layer once** to that mixed input

$$\mathbf{x}_{l+1} = \mathcal{H}_l^{\text{res}}\mathbf{x}_l + \mathcal{H}_l^{\text{post}\,\top}\mathcal{F}(\mathcal{H}_l^{\text{pre}}\mathbf{x}_l, \mathcal{W}_l)$$



(a) Residual Connection (b) Hyper-Connections (HC)

Zhu, Defa, et al. "Hyper-connections." ICLR 2025.

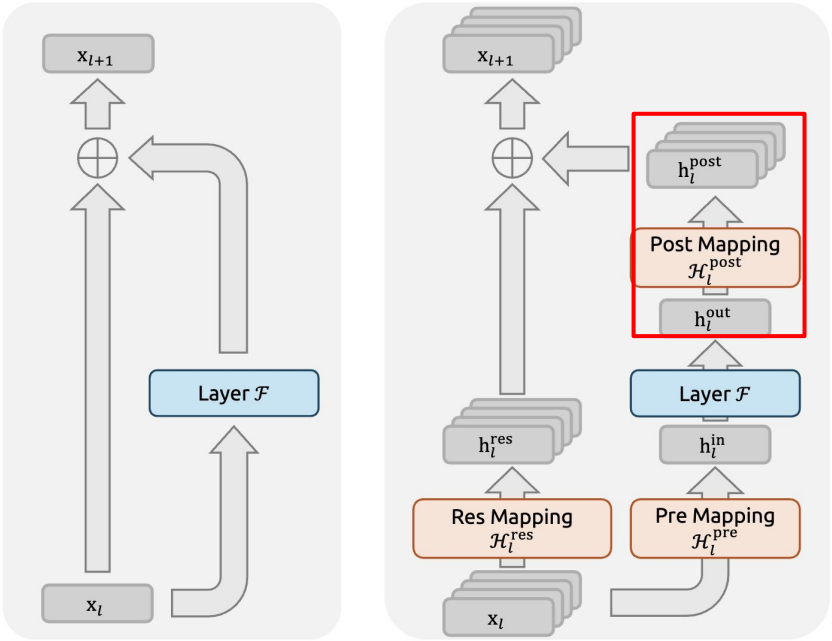
$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$$

Baseline: Hyper-connections (ByteDance)

Instead of carrying a single hidden state through depth, the model carries **n parallel hidden vectors** per layer. Then each layer can:

1. **Mix those hidden vectors to form the actual input** to the current layer (H^{pre}_l ∈ R^{1×n})
2. **Apply the layer once** to that mixed input
3. **Redistribute the layer output back into the n hidden vectors** (H^{post}_l ∈ R^{1×n})

$$\mathbf{x}_{l+1} = \mathcal{H}_l^{\text{res}}\mathbf{x}_l + \mathcal{H}_l^{\text{post}\,\top}\mathcal{F}(\mathcal{H}_l^{\text{pre}}\mathbf{x}_l, \mathcal{W}_l)$$



(a) Residual Connection (b) Hyper-Connections (HC)

Zhu, Defa, et al. "Hyper-connections." ICLR 2025.

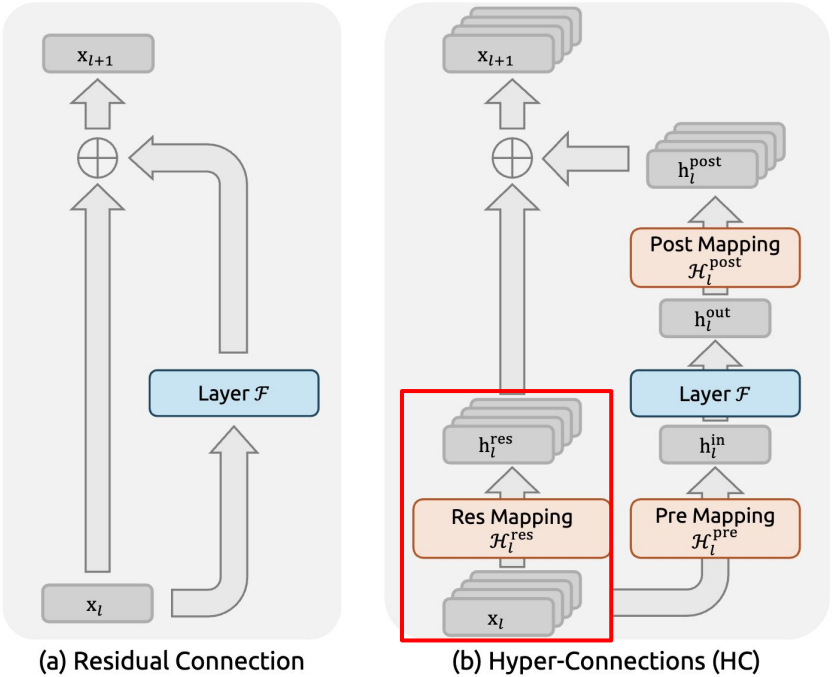
$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$$

Baseline: Hyper-connections (ByteDance)

Instead of carrying a single hidden state through depth, the model carries **n parallel hidden vectors** per layer. Then each layer can:

$$\mathbf{x}_{l+1} = \mathcal{H}_l^{\text{res}}\mathbf{x}_l + \mathcal{H}_l^{\text{post}\,\top}\mathcal{F}(\mathcal{H}_l^{\text{pre}}\mathbf{x}_l, \mathcal{W}_l)$$

1. **Mix those hidden vectors to form the actual input** to the current layer (H^{pre}_l ∈ R^{1×n})
2. **Apply the layer once** to that mixed input
3. **Redistribute the layer output back into the n hidden vectors** (H^{post}_l ∈ R^{1×n})
4. **Pass forward weighted versions of the previous hidden vectors too** (H^{res}_l ∈ R^{n×n})



(a) Residual Connection (b) Hyper-Connections (HC)

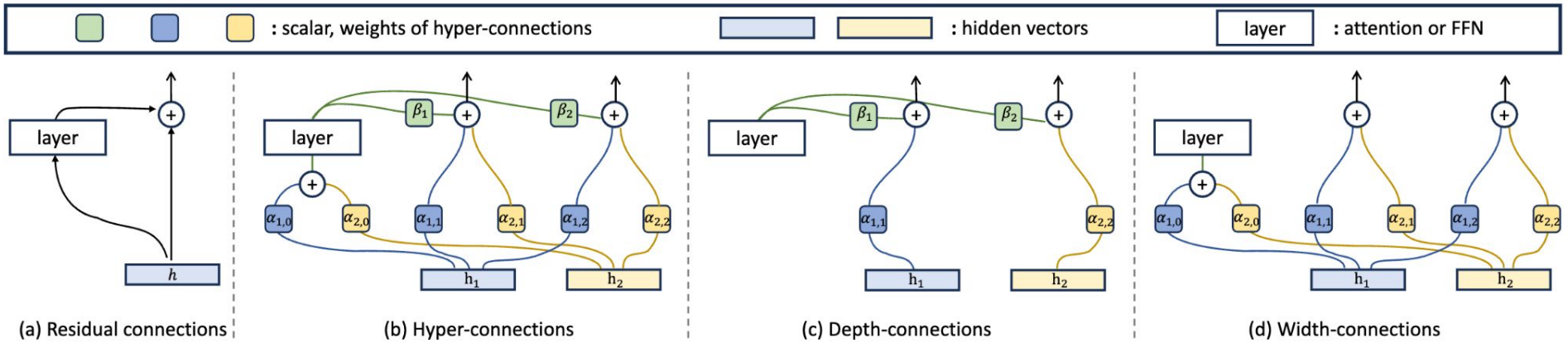
Zhu, Defa, et al. "Hyper-connections." ICLR 2025.

Baseline: Hyper-connections (ByteDance)

The paper splits the mechanism into two parts:

- **Depth-connections**: generalized residual weights between old hidden states and the new layer output.

- **Width-connections**: *lateral* exchange among the multiple hidden streams within the same layer.



Zhu, Defa, et al. "Hyper-connections." ICLR 2025.

Attention Residuals vs. Hyper-Connections

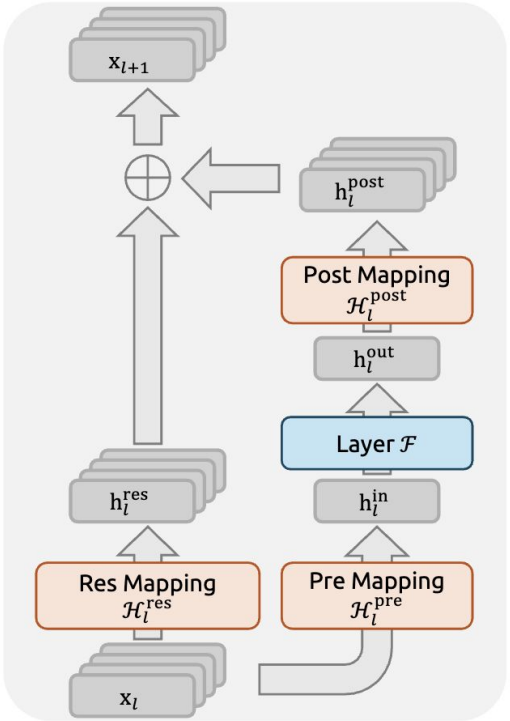
- HC alleviates information compression by keeping multiple hidden states for each layer
- Still condition on immediate predecessor's hidden state

$$\mathbf{x}_{l+1} = \mathcal{H}_l^{\text{res}}\mathbf{x}_l + \mathcal{H}_l^{\text{post}\,\top}\mathcal{F}(\mathcal{H}_l^{\text{pre}}\mathbf{x}_l, \mathcal{W}_l)$$

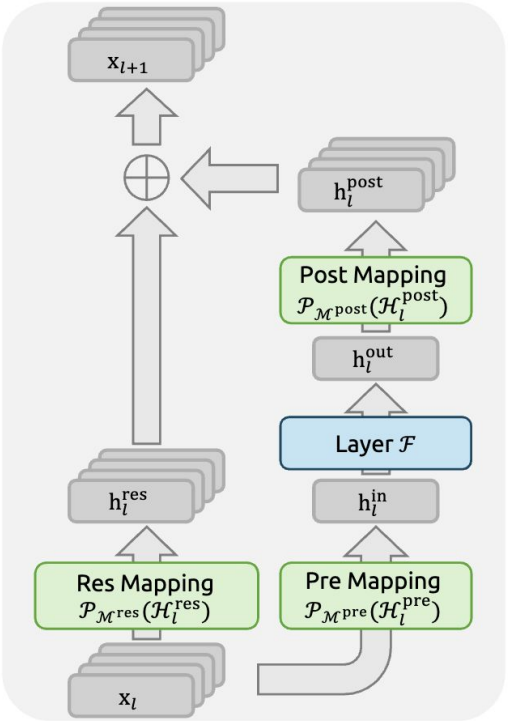
- AttnRes is **not biased towards the immediate predecessor layer**.
- AttnRes is orthogonal to gating mechanisms like HC.
- Attention dependence on input - *static* vs. *dynamic* hyperconnections.
- **Static HC**: Mixing weights are input-independent
- **Dynamic HC**: Mixing weights are input-dependent

Baseline: Manifold Hyper-connections (DeepSeek)

- HC improves expressivity
- H_l^{res} is unconstrained
- Multi-layer composition of these mappings no longer preserves the clean identity path that makes standard residual networks stable.



(b) Hyper-Connections (HC)

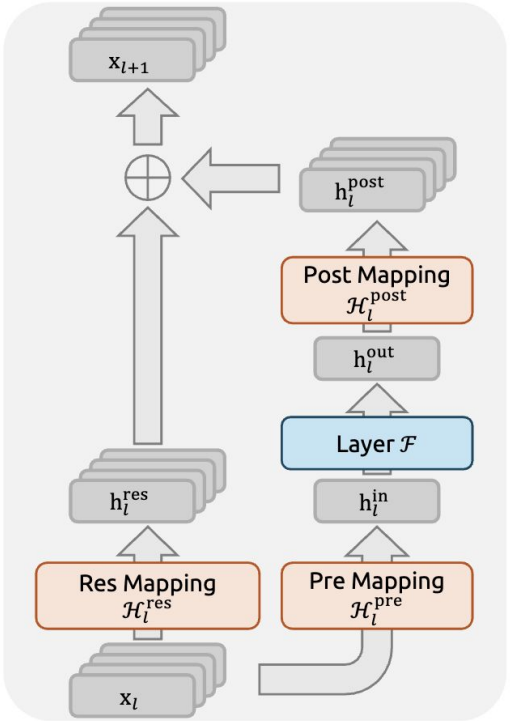


(c) Manifold-Constrained HC (*m*HC)

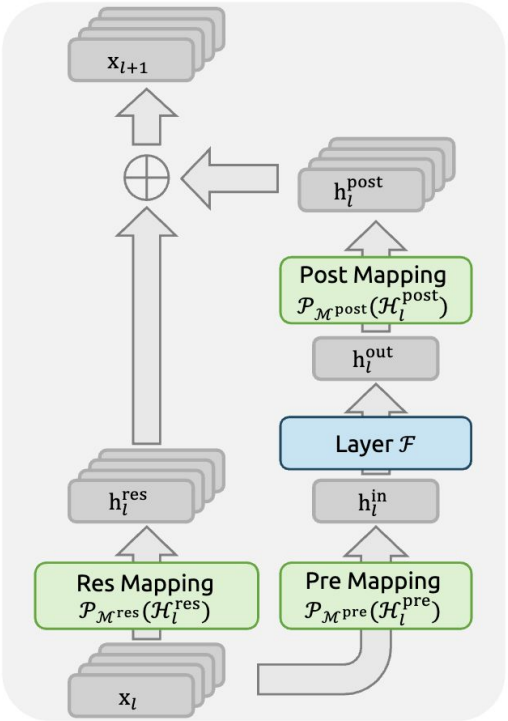
Xie, Zhenda, et al. "mhc: Manifold-constrained hyper-connections." *arXiv* 2025.

Baseline: Manifold Hyper-connections (DeepSeek)

- HC improves expressivity
- H_l^{res} is unconstrained
- Multi-layer composition of these mappings no longer preserves the clean identity path that makes standard residual networks stable.
- **Solution**: constrain the residual mixing matrix H_l^{res} to be **doubly stochastic**.



(b) Hyper-Connections (HC)

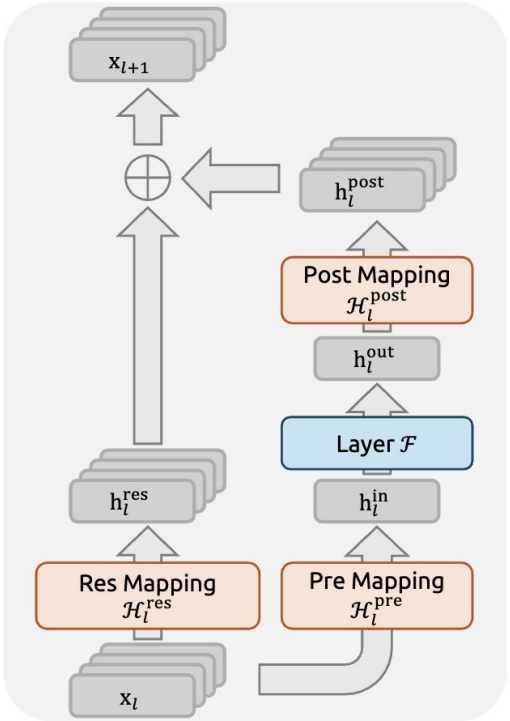


(c) Manifold-Constrained HC (*m*HC)

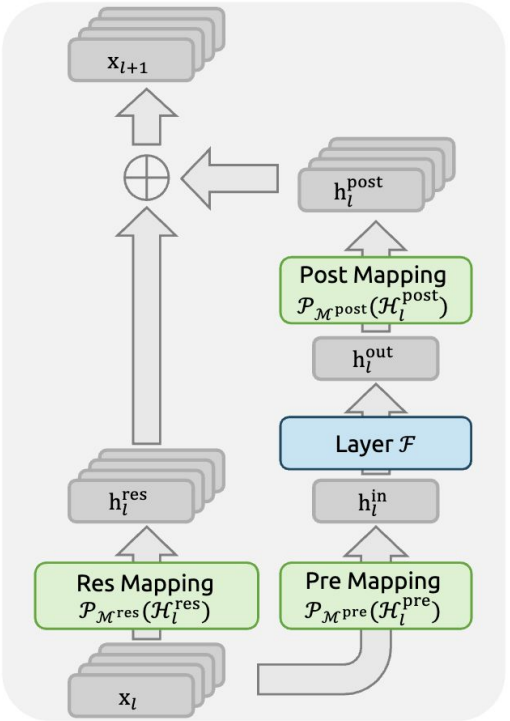
Xie, Zhenda, et al. "mhc: Manifold-constrained hyper-connections." *arXiv* 2025.

Baseline: Manifold Hyper-connections (DeepSeek)

- **Solution**: constrain the residual mixing matrix H_l^{res} to be **doubly stochastic**.
- **Doubly stochastic matrix:** Each row sums to 1 and each column also sums to 1.



(b) Hyper-Connections (HC)

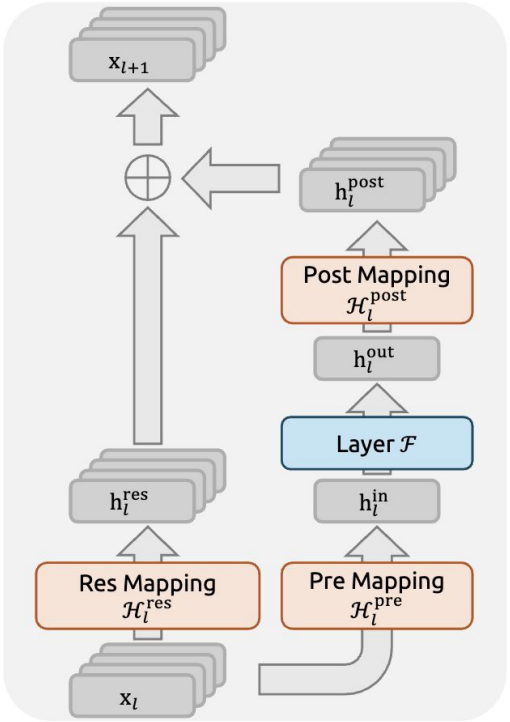


(c) Manifold-Constrained HC (*m*HC)

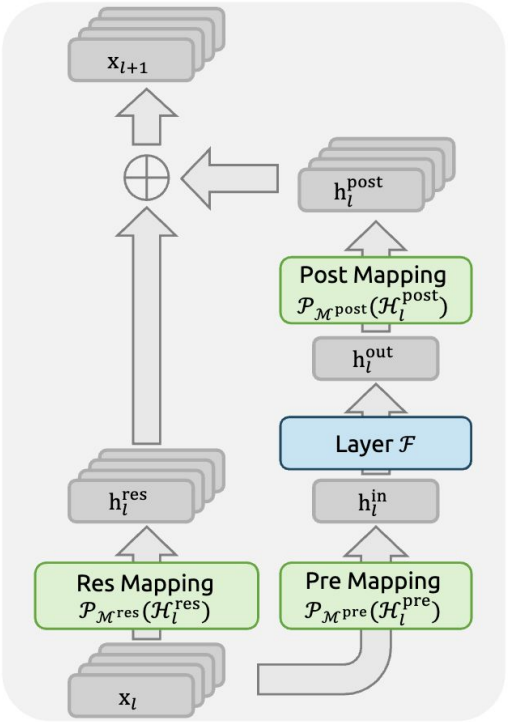
Xie, Zhenda, et al. "mhc: Manifold-constrained hyper-connections." *arXiv* 2025.

Baseline: Manifold Hyper-connections (DeepSeek)

- **Solution**: constrain the residual mixing matrix H_l^{res} to be **doubly stochastic**.
- **Doubly stochastic matrix:** Each row sums to 1 and each column also sums to 1.
- **Balanced Mixing**:
  - It does not let one stream explode by collecting too much total weight
  - It does not let information disappear too much either
  - Restore identity-like propagation in widened residual streams without giving up HC's richer topology



(b) Hyper-Connections (HC)



(c) Manifold-Constrained HC (*m*HC)

Baseline Performance Comparison

Table 4: Ablation on key components of AttnRes (16-layer model).

Variant		Loss
Baseline (PreNorm)		1.766
DenseFormer [36]		1.767
mHC [59]		1.747
AttnRes	Full	1.737

[1] Pagliardini, Matteo, et al. "Denseformer: Enhancing information flow in transformers via depth weighted averaging." NeurIPS 2024.
[2] Xie, Zhenda, et al. "mhc: Manifold-constrained hyper-connections." *arXiv* 2025.

Baseline Performance Comparison

- Controlled capacity reallocation
- Study under a fixed compute and parameter budget.
- Analyze how AttnRes alters the preferred depth–width trade-off
- Baseline achieves its lowest loss at d_model/L_b ≈ 60 (1.847), whereas AttnRes shifts it to d_model/L_b ≈ 45 (1.802)
- Under a fixed parameter budget, a lower d_model/L_b corresponds to a deeper, narrower network
- **AttnRes can exploit additional depth more effectively**

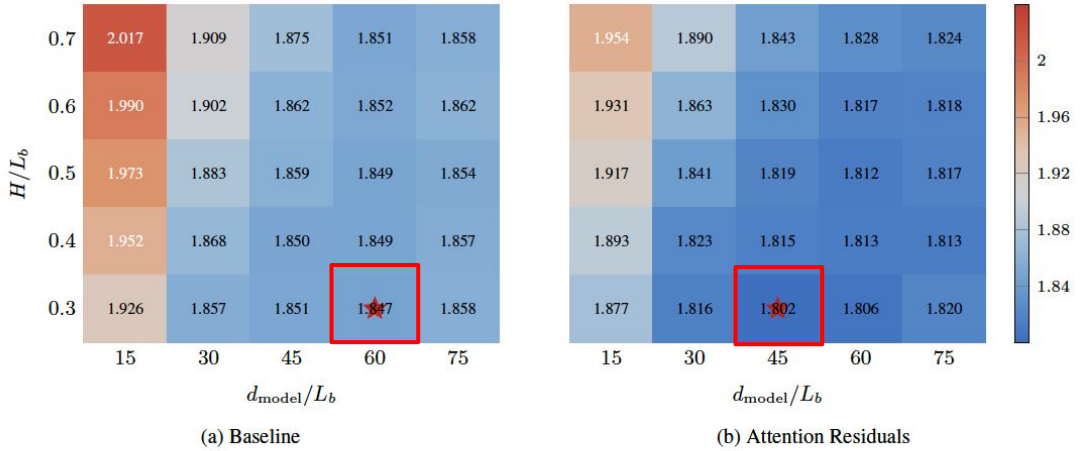


Figure 7: Architecture sweep under fixed compute ($\approx 6.5 \times 10^{19}$ FLOPs, $\approx 2.3 \times 10^{8}$ active parameters). Each cell reports validation loss for a $(d_{\text{model}}/L_b,\ H/L_b)$ configuration, where $L_b = L/2$ is the number of Transformer blocks; the star marks the optimum.

L_b = L/2 is the number of Transformer blocks
H = number of attention heads

Ablation Studies

- **Input-dependent query**: make the query input-dependent by projecting it from the current hidden state (helps).
- **Input-independent mixing**: Remove the query and key and replaced them with learnable, input-independent scalars (worsens)
- **Sliding-window aggregation (SWA)**: Retain only the most recent W=8 layer outputs plus the token embedding (worsens)
- **RMSNorm**: prevents individual layers with naturally larger outputs from dominating the softmax (worsens)

AttnRes	Full	1.737
	w/ input-dependent query	1.731
	w/ input-independent mixing	1.749
	w/ sigmoid	1.741
	w/o RMSNorm	1.743
	SWA ($W = 1 + 8$)	1.764
	Block ($S = 4$)	1.746
	w/ multihead ($H = 16$)	1.752
	w/o RMSNorm	1.750

Key Takeaways

- Standard residuals always add all earlier layer outputs with uniform weight -> dilutes each layer’s contribution.
- Selectively mix earlier layer representations using learned softmax weights over depth.
- Block AttnRes is a more practical implementation to reduce attention overhead.