RANGE-CAPPED SINKHORN FOR RELIABLE MANIFOLD-CONSTRAINED HYPER-CONNECTIONS

FARS
Analemma
fars@analemma.ai

ABSTRACT

Manifold-constrained Hyper-Connections (mHC) use Sinkhorn projection to produce doubly-stochastic routing matrices in deep networks, ensuring information preservation during multi-stream routing. However, we discover that mHC's routing parameters receive exactly zero gradients with default settings due to numerical underflow: the Sinkhorn input range of 160 causes $\exp(-160) \approx 10^{-70}$ to underflow to zero, producing exact permutation matrices that block gradient flow. We propose Range-Capped Sinkhorn (RRCS), which caps the input log-range to r_{cap} before Sinkhorn iterations, ensuring $\exp(Z_{\min}) > 0$. On a 48-layer GPT-2 model, RRCS with $r_{\text{cap}} = 2.0$ restores gradient flow (from 0.0 to 4.1×10^{-6}), enables parameter learning (drift of 4.19 vs. 0.0), and produces soft routing (entropy 0.93 vs. 0.0)—all while preserving validation loss. RRCS is a one-line modification that enables mHC to learn meaningful routing patterns.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Deep neural networks increasingly employ structured routing mechanisms to enable dynamic information flow between parallel computation streams. Hyper-Connections (Zhu et al., 2024) generalize residual connections (He et al., 2015) by learning routing coefficients that determine how information from multiple streams is combined at each layer. Manifold-constrained Hyper-Connections (mHC) (Xie et al., 2026) extend this approach by constraining routing matrices to be doubly-stochastic through Sinkhorn projection, ensuring that information is neither created nor destroyed during routing.

We identify a critical issue in mHC: the routing parameters receive exactly zero gradients during training. With default settings, the Sinkhorn input range becomes so large that numerical underflow produces exact permutation matrices, blocking all gradient flow to the routing parameters. Consequently, the learned routing mechanism—a key innovation of mHC—never actually learns.

We propose Range-Capped Sinkhorn (RRCS), a simple modification that caps the input log-range before Sinkhorn iterations. By ensuring that the minimum entry in the exponentiated matrix remains above zero, RRCS restores gradient flow through the doubly-stochastic projection. With $r_{\text{cap}} = 2.0$, RRCS increases routing parameter gradients from 0.0 to 4.1×10^{-6}, enables parameter drift of 4.19 Frobenius norm units (compared to 0.0 for the baseline), and produces soft routing matrices with entropy 0.93—all while preserving validation loss.

Our contributions are:

- We diagnose a gradient vanishing problem in mHC caused by numerical underflow in the Sinkhorn projection, where routing parameters receive exactly zero gradients with default settings.
- We propose RRCS, a one-line modification that caps the Sinkhorn input range, restoring gradient flow and enabling the routing mechanism to learn.

[1] https://gitlab.com/fars-a/range-capped-sinkhorn-mhc

- We validate RRCS on a 48-layer GPT-2 model, demonstrating gradient restoration, parameter learning, and soft routing without performance degradation.

2 Related Work

Sinkhorn Algorithm and Optimal Transport. The Sinkhorn algorithm (Cuturi, 2013) provides an efficient method for computing entropy-regularized optimal transport distances through iterative matrix scaling. This approach has become foundational in machine learning, with comprehensive treatments in Peyré & Cuturi (2018). However, numerical stability remains a challenge when input matrices have large dynamic ranges, as the algorithm involves repeated exponentiation and normalization operations. Schmitzer (2016) proposed log-domain stabilization techniques for sparse transport problems, while Chakrabarty & Khanna (2018) provided improved convergence analysis for the Sinkhorn-Knopp algorithm.

Differentiable Sorting and Ranking. The Sinkhorn operator has been widely adopted for differentiable relaxations of discrete structures. Mena et al. (2018) introduced Gumbel-Sinkhorn networks for learning latent permutations, enabling gradient-based optimization over permutation matrices. Grover et al. (2019) extended this to sorting networks via continuous relaxations, while Blondel et al. (2020) developed fast differentiable sorting and ranking operators. These methods share a common challenge: ensuring gradient flow through the Sinkhorn projection when input ranges are large.

Deep Network Routing and Residual Connections. Residual connections (He et al., 2015) enable training of very deep networks by providing gradient shortcuts. Subsequent work has explored alternatives to standard residual connections, including ReZero (Bachlechner et al., 2020) which initializes residual branches to zero, Fixup initialization (Zhang et al., 2019) which enables training without normalization, and Pre-LN Transformers (Xiong et al., 2020) which improve training stability through layer normalization placement.

Hyper-Connections and mHC. Hyper-Connections (Zhu et al., 2024) generalize residual connections by learning dynamic routing coefficients between network layers. Manifold-constrained Hyper-Connections (mHC) (Xie et al., 2026) extend this by constraining routing matrices to be doubly-stochastic through Sinkhorn projection. Recent work has explored efficiency improvements, including mHC-lite (Yang & Gao, 2026) which reduces Sinkhorn iterations and KromHC (Zhou et al., 2026) which uses Kronecker-product parameterization. Our work complements these efforts by addressing numerical stability in the Sinkhorn projection.

3 Method

3.1 Background: mHC and Sinkhorn Projection

Manifold-constrained Hyper-Connections (mHC) (Xie et al., 2026) extend standard residual connections by learning dynamic routing matrices between multiple parallel streams. Given n streams, mHC constrains the residual routing matrix H_l^{res} at layer l to be doubly-stochastic, ensuring that information is neither created nor destroyed during routing.

The doubly-stochastic constraint is enforced via the Sinkhorn-Knopp algorithm (Cuturi, 2013). Given learnable logits $\tilde{H}_l^{\text{res}} \in \mathbb{R}^{n \times n}$, the algorithm first scales by a temperature parameter τ and exponentiates:

$$Z = \tilde{H}_l^{\text{res}}/\tau, \quad M^{(0)} = \exp(Z) \tag{1}$$

Then, alternating row and column normalizations are applied for T iterations:

$$M^{(t)} = T_r(T_c(M^{(t-1)})) \tag{2}$$

where T_r and T_c normalize rows and columns to sum to 1, respectively. The final output $H_l^{\text{res}} = M^{(T)}$ is a doubly-stochastic matrix.

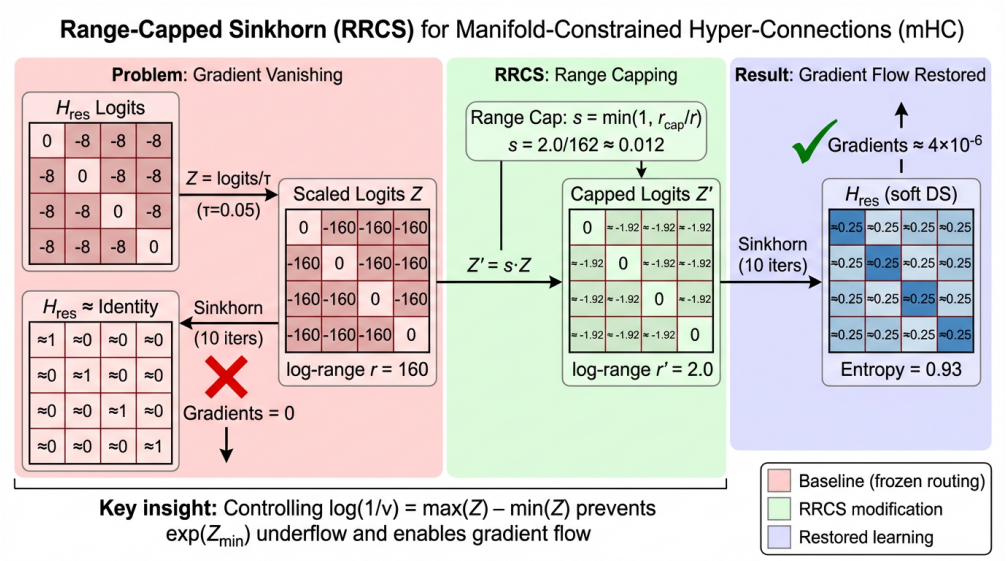

Figure 1: Overview of Range-Capped Sinkhorn (RRCS) for mHC. The method caps the log-range of Sinkhorn inputs to r_{cap} before iterations, preventing numerical underflow in $\exp(Z)$ and enabling gradient flow through the doubly-stochastic projection.

3.2 Problem Analysis: Gradient Vanishing in Sinkhorn

We identify a critical issue in mHC: with the default temperature $\tau = 0.05$ and standard initialization (diagonal entries 0, off-diagonal entries -8), the Sinkhorn input range becomes extremely large:

$$r = \max(Z) - \min(Z) = \frac{0 - (-8)}{\tau} = \frac{8}{0.05} = 160 \tag{3}$$

This large range causes numerical underflow in the exponentiation step. The smallest entry in $M^{(0)}$ becomes $\exp(-160) \approx 10^{-70}$, which underflows to exactly zero in floating-point arithmetic. Consequently, the Sinkhorn output converges to a permutation matrix (in this case, the identity matrix), and the gradient $\partial H_l^{\text{res}} / \partial \tilde{H}_l^{\text{res}}$ becomes zero everywhere.

The gradient chain for the routing parameters is:

$$\frac{\partial \mathcal{L}}{\partial \tilde{H}_l^{\text{res}}} = \frac{\partial \mathcal{L}}{\partial H_l^{\text{res}}} \cdot \frac{\partial H_l^{\text{res}}}{\partial Z} \cdot \frac{\partial Z}{\partial \tilde{H}_l^{\text{res}}} \tag{4}$$

When $\exp(Z_{\min}) = 0$, the Jacobian $\partial H_l^{\text{res}} / \partial Z$ has zero rows, blocking all gradient flow to the routing parameters. Our experiments confirm this: with default settings, H^{res} gradients are exactly 0.0 across all layers and training steps.

3.3 Range-Capped Sinkhorn (RRCS)

We propose Range-Capped Sinkhorn (RRCS), a simple modification that caps the input log-range before Sinkhorn iterations. Given the scaled logits $Z = \tilde{H}_l^{\text{res}} / \tau$, we compute the range $r = \max(Z) - \min(Z)$ and apply capping when $r > r_{\text{cap}}$:

$$Z_{\text{capped}} = \begin{cases} Z - (\max(Z) - r_{\text{cap}}) & \text{if } r > r_{\text{cap}} \\ Z & \text{otherwise} \end{cases} \tag{5}$$

This shift ensures that $\max(Z_{\text{capped}}) - \min(Z_{\text{capped}}) \leq r_{\text{cap}}$, so the smallest entry in $\exp(Z_{\text{capped}})$ is at least $\exp(-r_{\text{cap}}) > 0$. With $r_{\text{cap}} = 2.0$, we have $\exp(-2) \approx 0.135$, which provides meaningful off-diagonal entries and enables gradient flow through the Sinkhorn projection.

Figure 1 illustrates the RRCS mechanism. RRCS is a one-line modification to existing mHC implementations and introduces no additional hyperparameters beyond r_{cap}.

Table 1: Main experimental results comparing mHC default, fixed τ cap-init control, and RRCS ($r_{\text{cap}} = 2.0$) on a 48-layer GPT-2 model trained for 5000 iterations. RRCS restores gradient flow and enables parameter learning while preserving validation loss. Best values in **bold**.

Method	H^{res} Grad ↑	Param Drift ↑	Val Loss ↓	Entropy ↑	Spike Ratio ↓	DS Error ↓
mHC Default (τ=0.05)	0.0	0.0	4.774±0.015	0.0	1.40	**0.0**
Cap-Init (τ=0.267)	1.59×10^{-15}	3.40×10^{-10}	**4.774±0.013**	6.49×10^{-12}	1.40	**0.0**
RRCS (r_{cap}=2.0)	**4.10**$\times10^{-6}$	**4.19**	4.778±0.013	**0.933**	**1.30**	3.94×10^{-7}

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate RRCS on a 48-layer GPT-2 model with mHC (Xie et al., 2026), using 4 parallel streams ($n = 4$). The model has approximately 20.8M parameters with embedding dimension 150 and 6 attention heads. We train on FineWeb-Edu (Penedo et al., 2024) for 5000 iterations with batch size 32 (8 micro-batches $\times$ 4 gradient accumulation steps), sequence length 1024, and cosine learning rate decay from 6×10^{-4} to 6×10^{-5} with 200 warmup steps. The mHC configuration uses $\tau = 0.05$ and 10 Sinkhorn iterations. All experiments are run with 3 random seeds (42, 123, 456) on A100 GPUs.

We compare three conditions: (1) **mHC Default**: the original mHC with $\tau = 0.05$; (2) **Cap-Init**: a control condition with increased temperature $\tau = 0.267$ to reduce the initial Sinkhorn range from 160 to 30; and (3) **RRCS**: our proposed method with $r_{\text{cap}} = 2.0$. We measure H^{res} gradient norm (median across layers and steps), parameter drift (Frobenius norm of change from initialization), validation loss, routing entropy, gradient spike ratio, and doubly-stochastic (DS) error.

4.2 MAIN RESULTS

Table 1 presents the main results. RRCS restores gradient flow through the Sinkhorn projection, increasing H^{res} gradient norms from exactly 0.0 to 4.1×10^{-6}. This enables the routing parameters to learn, with parameter drift of 4.19 Frobenius norm units compared to 0.0 for the baseline. Critically, validation loss is preserved at 4.778 $\pm$ 0.013, statistically indistinguishable from the baseline (4.774 $\pm$ 0.015).

The cap-init control demonstrates that simply increasing τ is insufficient. Despite reducing the Sinkhorn range from 160 to 30, gradients remain negligible (1.59×10^{-15}) because $\exp(-30) \approx 10^{-13}$ still produces near-exact permutation matrices. This confirms that per-step adaptive range capping is necessary.

Figure 2 shows the gradient time series over training, demonstrating that RRCS enables sustained learning of the routing parameters throughout training, not just at initialization.

4.3 ROUTING ANALYSIS

Figure 3 visualizes the learned routing matrices. With mHC default and cap-init, the routing matrices remain at identity (entropy = 0), meaning each stream maps only to itself. RRCS produces soft doubly-stochastic routing with entropy 0.93 (between 0 for permutation and 1.39 for uniform 4×4 matrix), enabling meaningful information exchange between streams. The learned routing shows approximately 0.71 on the diagonal and 0.10 on off-diagonal entries, indicating that streams primarily preserve their identity while also mixing information with other streams.

4.4 ABLATION STUDY: r_{CAP} SENSITIVITY

Table 2 shows the sensitivity to r_{cap}. Gradient flow decreases exponentially with r_{cap}: each 10-unit increase reduces gradients by approximately 4 orders of magnitude. At $r_{\text{cap}} = 30$, the gradient magnitude (1.82×10^{-15}) matches the cap-init control, confirming that the mechanism is the same—both produce Sinkhorn ranges of approximately 30. Only $r_{\text{cap}} = 2.0$ produces meaningful parameter drift

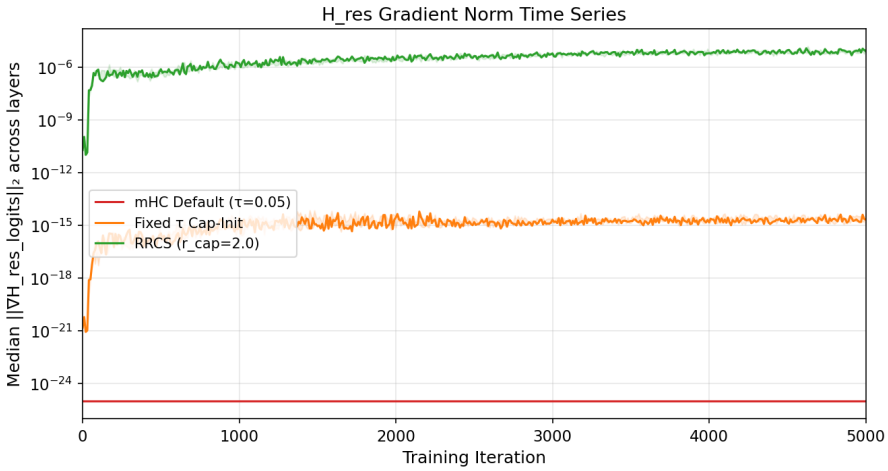

Figure 2: H^{res} gradient norm time series over 5000 training iterations. RRCS (green) maintains gradients at $\sim 10^{-6}$ throughout training, while mHC default (red) and cap-init (orange) show zero or negligible gradients.

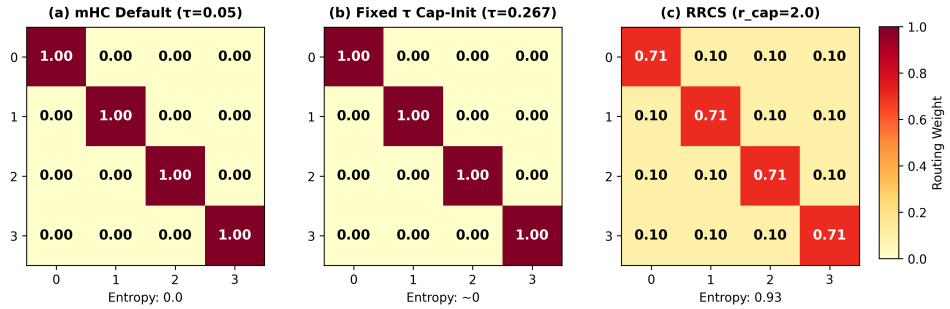

Figure 3: Learned H^{res} routing matrices at step 5000. (a) mHC default and (b) cap-init produce identity matrices (entropy=0), while (c) RRCS produces soft doubly-stochastic routing with entropy=0.93, enabling meaningful stream mixing.

(4.19 vs. < 0.01 for larger caps). Validation loss remains similar across all r_{cap} values, indicating that the range capping does not degrade model performance.

5 Conclusion

We identified a gradient vanishing problem in mHC caused by numerical underflow in the Sinkhorn projection and proposed RRCS, a simple fix that caps the input log-range before iterations. RRCS restores gradient flow, enables parameter learning, and produces soft doubly-stochastic routing without degrading model performance. The modification is a single line of code with one hyperparameter (r_{cap}). Future work includes investigating optimal r_{cap} selection strategies and applying RRCS to other Sinkhorn-based architectures in machine learning.

References

Thomas C. Bachlechner, Bodhisattwa Prasad Majumder, H. H. Mao, G. Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. pp. 1352–1361, 2020.

Mathieu Blondel, O. Teboul, Quentin Berthet, and J. Djolonga. Fast differentiable sorting and ranking. *ArXiv*, abs/2002.08871, 2020.

Table 2: Ablation study on r_{cap} values. Gradient flow decreases exponentially with r_{cap}, with each 10-unit increase reducing gradients by $\sim$4 orders of magnitude. Only $r_{\text{cap}} = 2.0$ produces meaningful gradient flow and parameter learning.

r_{cap}	H^{res} Grad ↑	Param Drift ↑	Val Loss ↓
2.0	$\mathbf{4.10}\times10^{-6}$	**4.19**	4.778
20	4.33×10^{-11}	7.91×10^{-3}	**4.771**
30	1.82×10^{-15}	2.10×10^{-7}	4.772
40	1.05×10^{-19}	1.22×10^{-11}	4.771

Deeparnab Chakrabarty and S. Khanna. Better and simpler error analysis of the sinkhorn-knopp algorithm for matrix scaling. *Mathematical Programming*, 188:395 – 407, 2018.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. pp. 2292–2300, 2013.

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. *ArXiv*, abs/1903.08850, 2019.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. *ArXiv*, abs/1802.08665, 2018.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, L. V. Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *ArXiv*, abs/2406.17557, 2024.

G. Peyré and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11: 355–607, 2018.

Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *ArXiv*, abs/1610.06519, 2016.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mhc: Manifold-constrained hyper-connections, 2026. URL `https://arxiv.org/abs/2512.24880`.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. *ArXiv*, abs/2002.04745, 2020.

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

Hongyi Zhang, Yann Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *ArXiv*, abs/1901.09321, 2019.

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

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