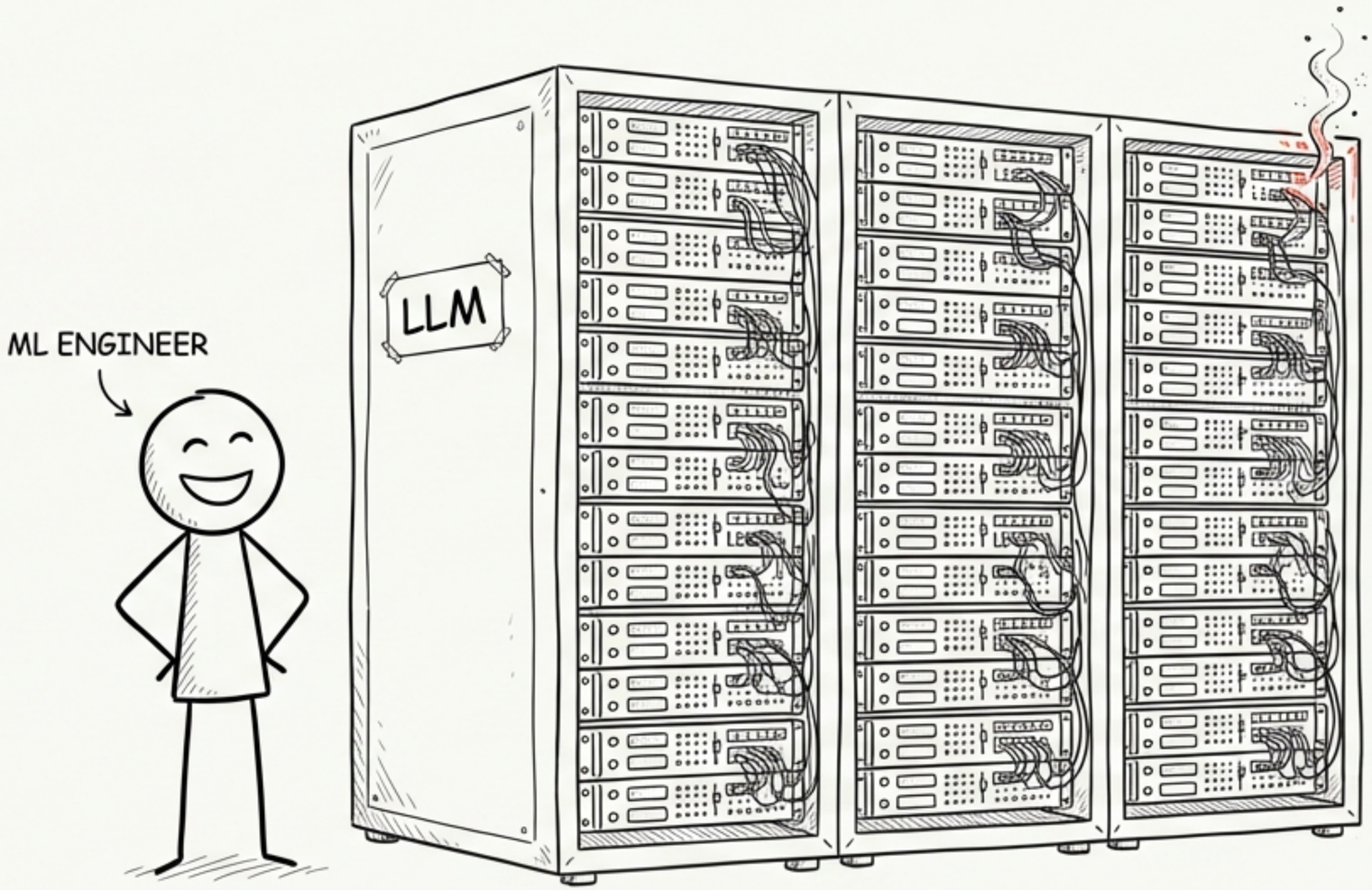
How We Stopped Our Giant AI Models From Spontaneously Combusting

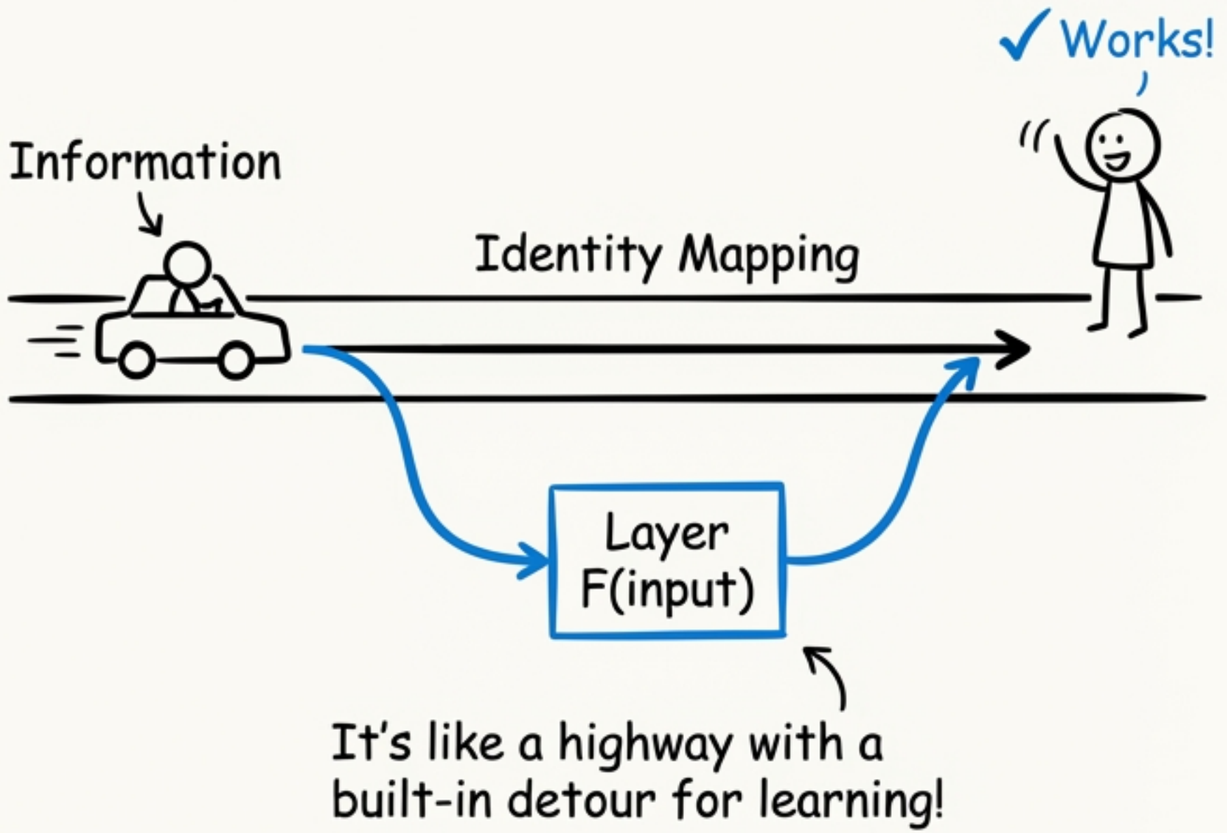
A story about an architectural flaw, a 57-year-old algorithm, and the Birkhoff Polytope.



For Nine Years, This Was Good Enough.

Since 2016, nearly every deep network has been built on the **Residual Connection** (in blue). It lets signals and gradients flow through deep models without vanishing.

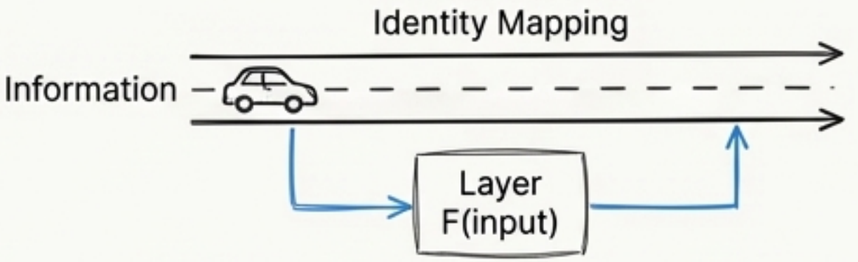
- The formula is simple: `output = F(input) + input`
- Its key property is **identity mapping**: the signal can pass pass through unmodified.



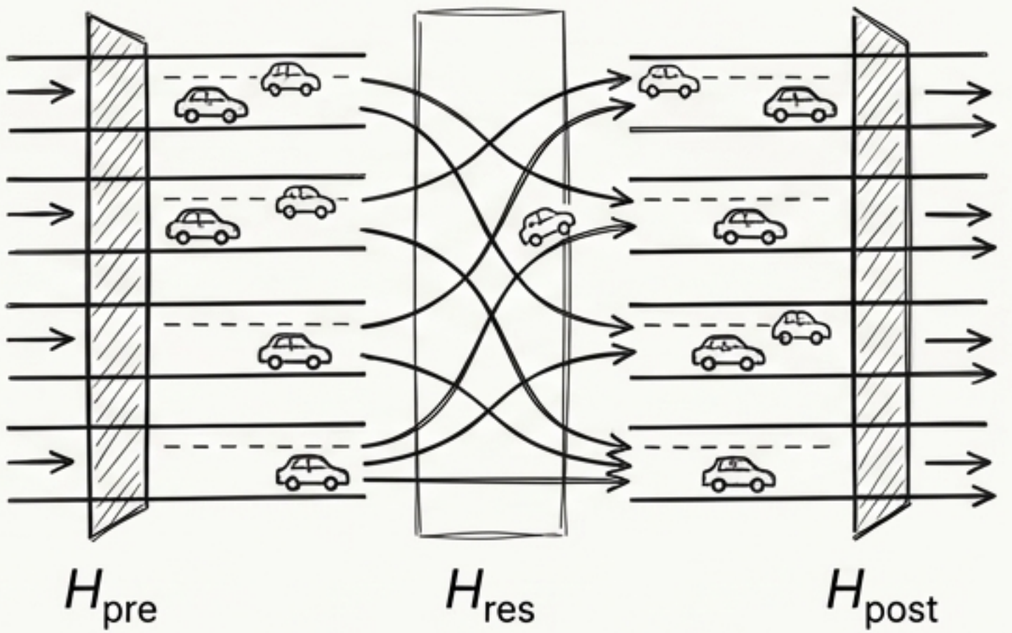
What if we built a superhighway?

As models got bigger, that single lane became a bottleneck. **Hyper-Connections (HC)** seemed like the answer:

- Expand the residual stream from 1 path to *n* parallel paths (e.g., n=4).
- Add learnable matrices (H_{res}, H_{pre}, H_{post}) to let the streams mix and exchange information.
- **The Promise**: More information bandwidth and better feature mixing, with minimal extra FLOPs.



Residual Connection (Previous Slide)

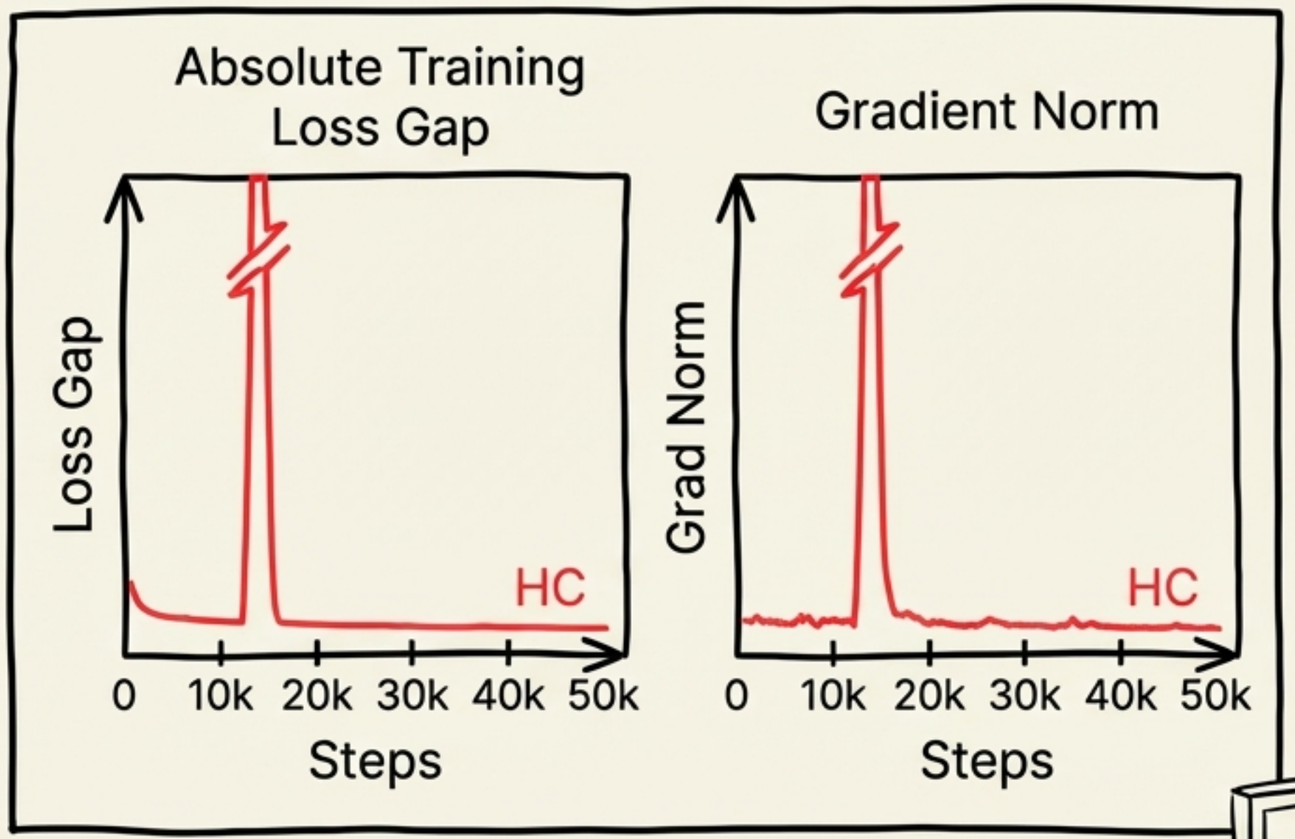


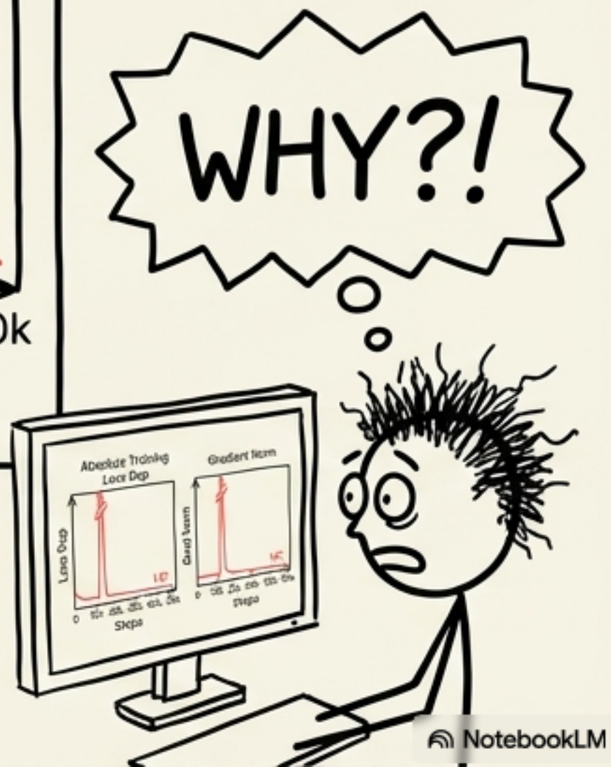
Hyper-Connection

At scale, the superhighway spontaneously combusts.

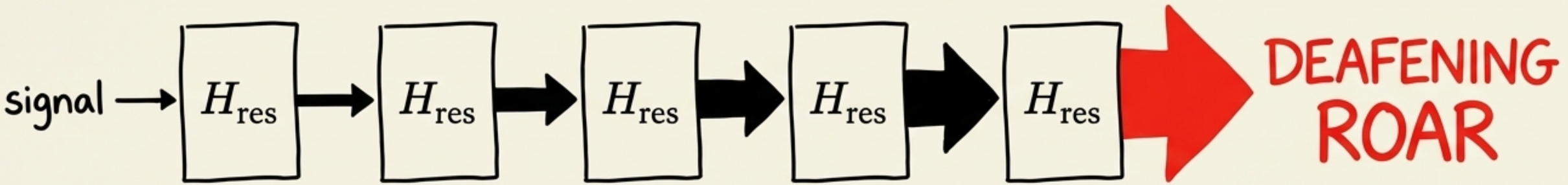
When training large models (e.g., 27B parameters), HC becomes catastrophically unstable.

- Around the 12k training step, the loss suddenly spikes.
- The gradient norm explodes, derailing the entire training process.





The Problem: A death by a thousand matrix multiplications.



The instability comes from the H_{res} **matrix**, which mixes the residual streams at each layer.

Across L layers, the total effect is a product of all the matrices: $(H_{resL-1}) * \ldots * (H_{res1})$.

This is the **composite mapping**.

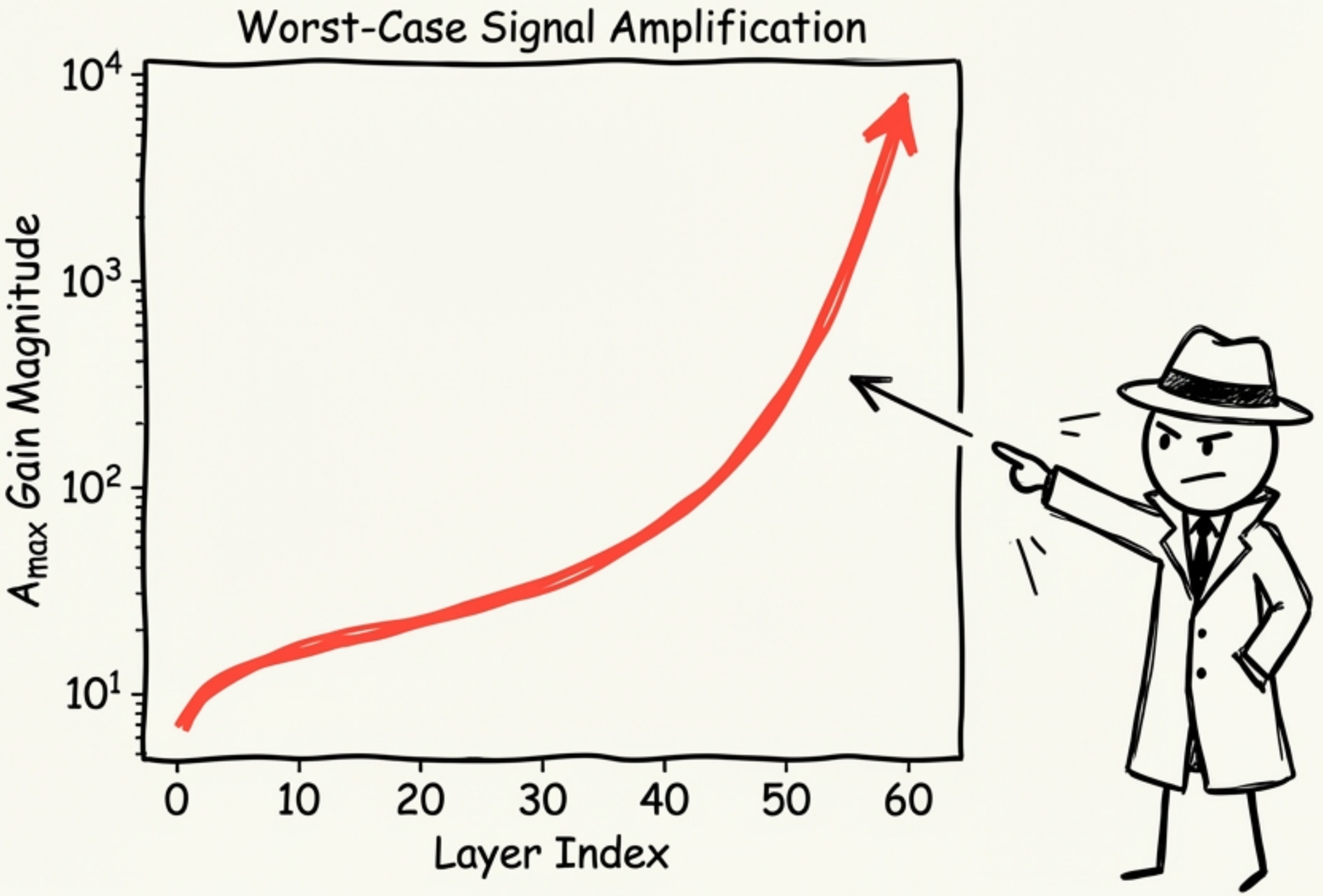
Because H_{res} is unconstrained, Because H_{res} is unconstrained, small signal amplifications at each layer compound exponentially.

What starts as a whisper from an early layer becomes a **deafening roar** by the end of the network.

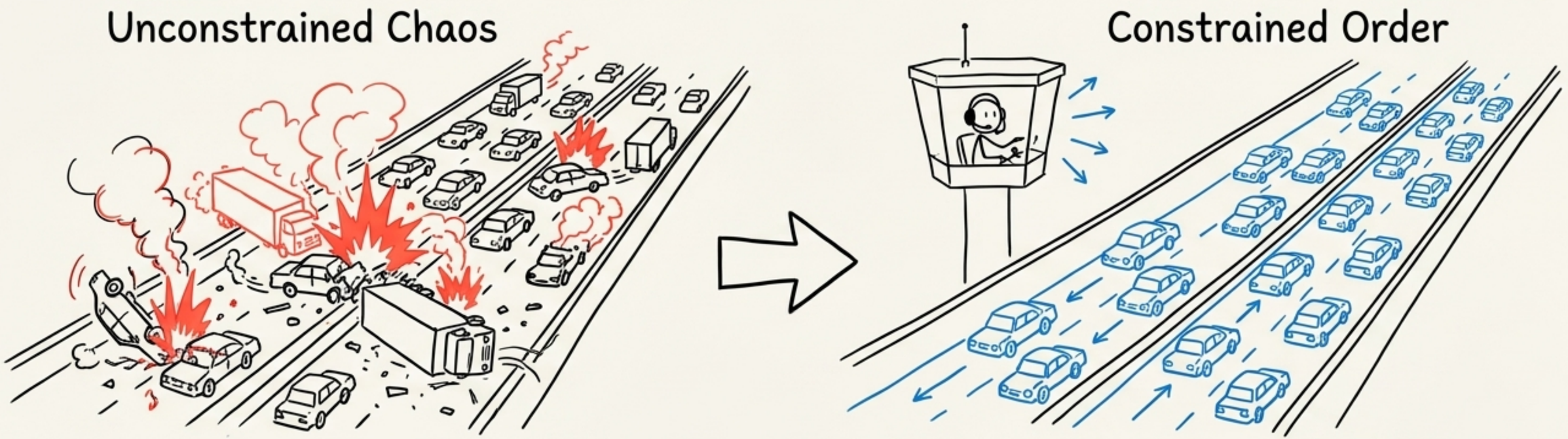
The evidence is clear: the signal gain reached 3,000x.

We measured the worst-case signal amplification (the "Amax Gain Magnitude") of the composite mapping in a 27B model.

- The gain grows exponentially with model depth.
- In deeper layers, the signal was amplified by over three orders of magnitude.
- This is not a tuning issue; it's a fundamental architectural instability.



The Fix: Force the matrices to behave.



The solution is **Manifold-Constrained Hyper-Connections** (mHC). Instead of trying to regularize the chaos, we constrain it out of existence.

- We project the `H_res` matrix onto a specific mathematical manifold: the **Birkhoff Polytope**.
- This forces `H_res` to become a **doubly stochastic matrix**.

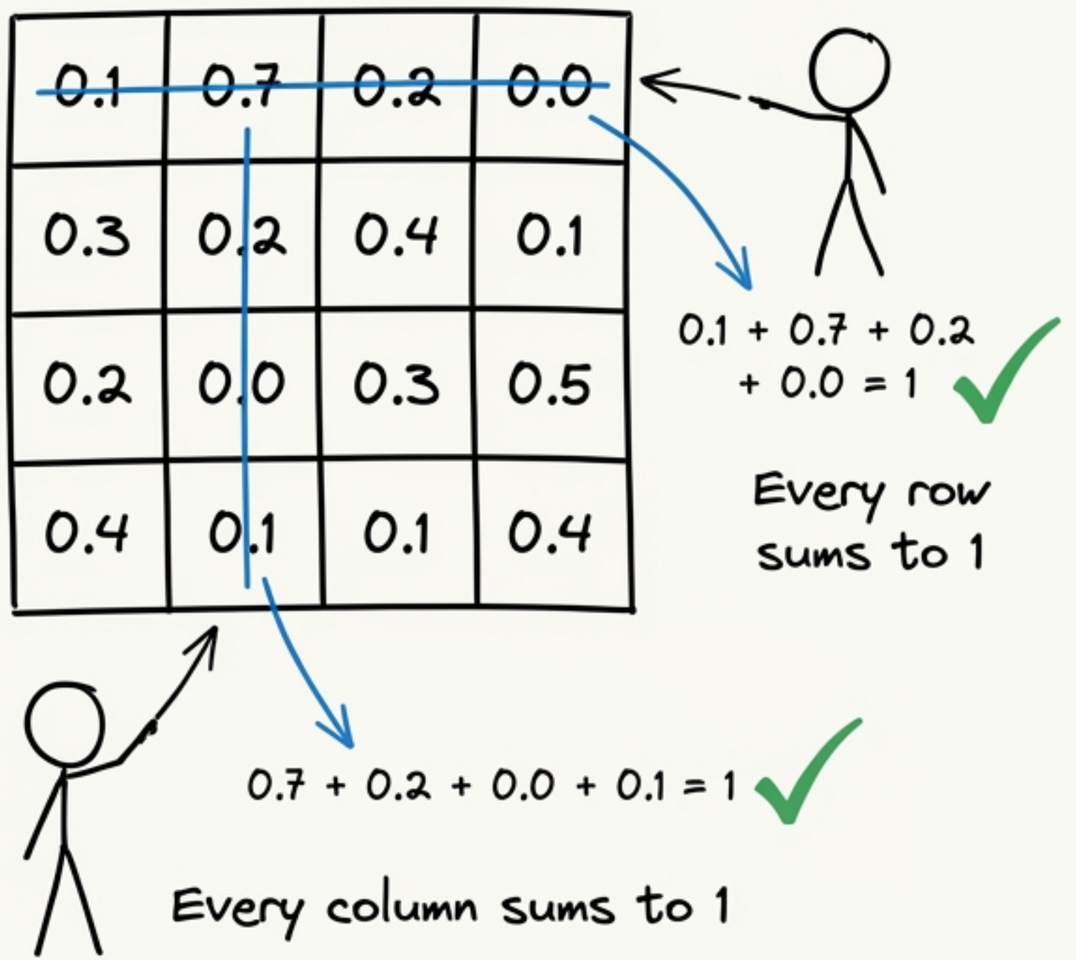
What's a "Doubly Stochastic Matrix"?

It's a matrix with two simple, powerful rules:

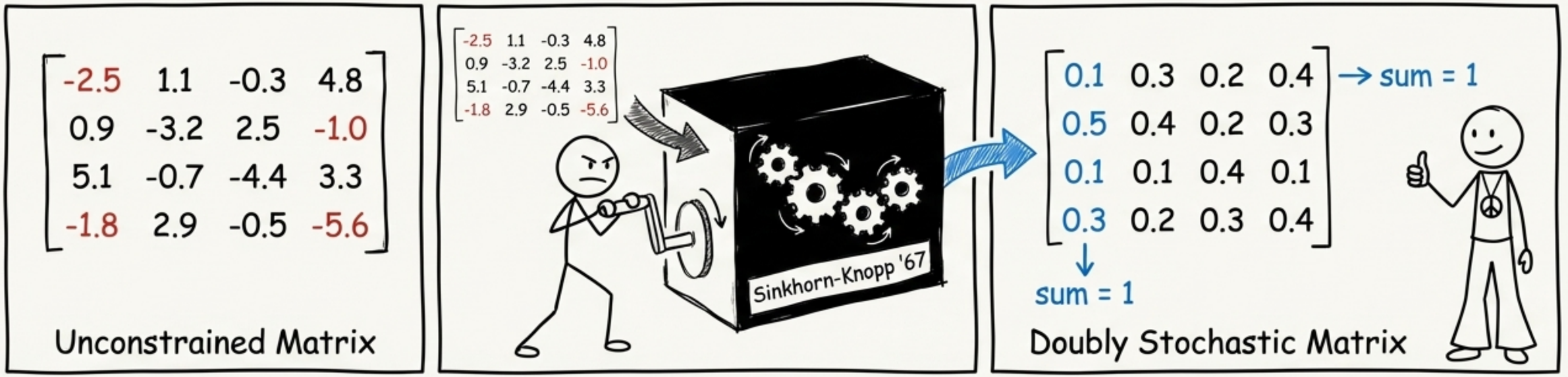
1. All entries are non-negative.
2. Every row AND every column must sum to exactly 1.

Why this works:

- **No Signal Explosion**: The spectral norm is ≤ 1. The matrix cannot amplify the signal.
- **Stable Over Depth**: The product of two doubly stochastic matrices is also doubly stochastic. The stability is guaranteed, no matter how deep the model.
- **Fair Information Mixing**: It acts as a 'convex combination of permutations.' It shuffles and averages information, but can't play favorites.



The solution came from a 1967 paper on numerical analysis.

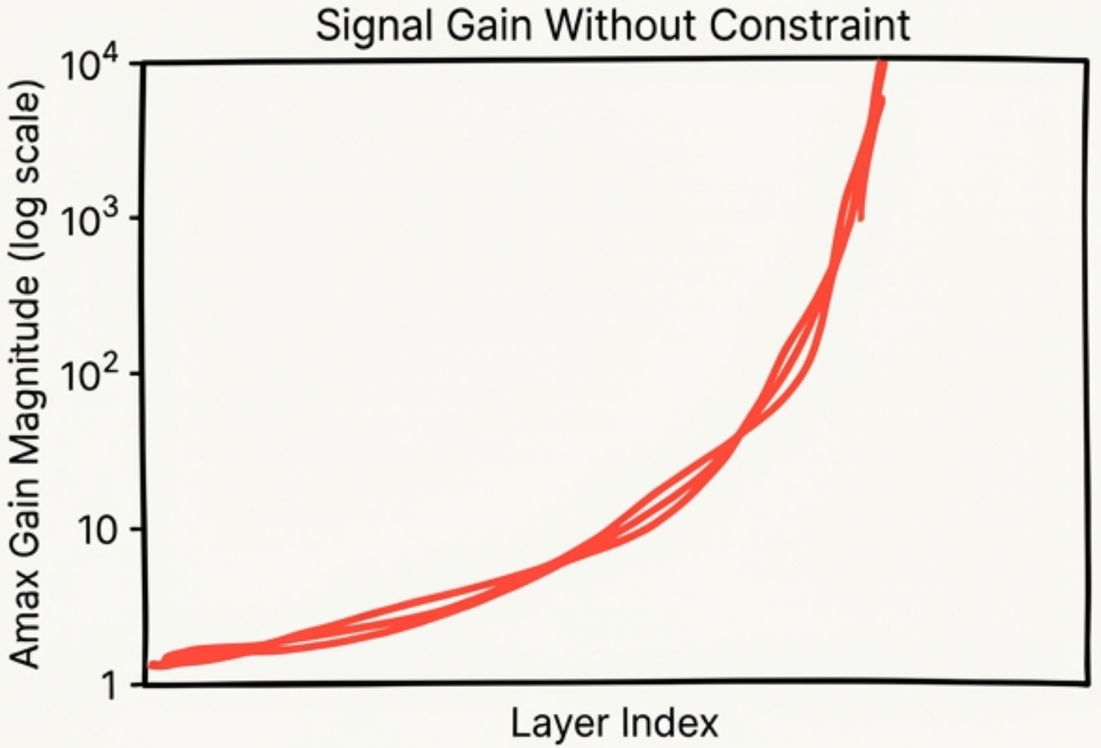


To enforce the constraint, we use the **Sinkhorn-Knopp algorithm**.

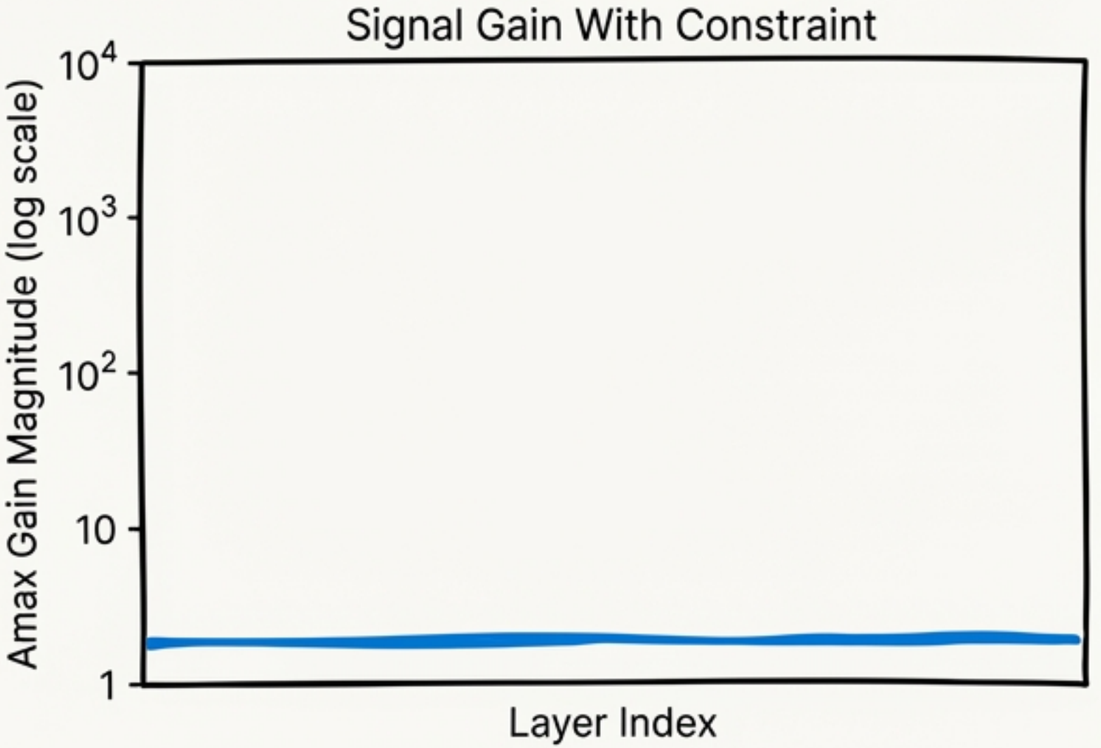
- It's a simple iterative process:
  1. Make all matrix entries positive (using exp).
  2. Normalize all rows to sum to 1.
  3. Normalize all columns to sum to 1.
  4. Repeat ~20 times.
- This reliably converges to the nearest doubly stochastic matrix.

The difference is... not subtle.

HC - Unconstrained



mHC - Constrained



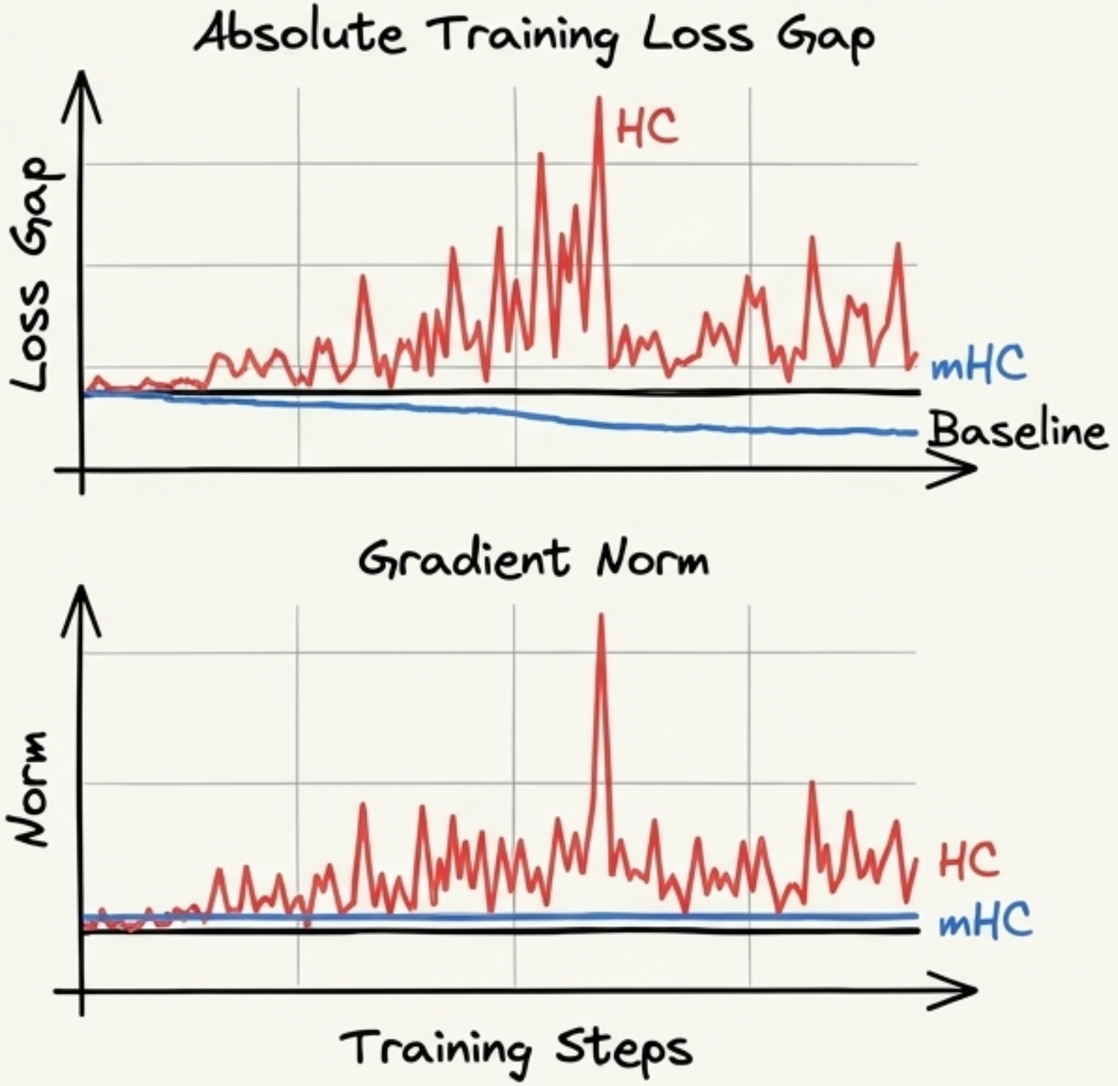
By constraining the composite mapping, the maximum signal gain is reduced by **three orders of magnitude**. The network is structurally stable.

In a real 27B model, the chaos is gone.

The stability isn't just theoretical. During training:

- **HC**: Experiences the same loss spike and gradient explosion we saw earlier.
- **mHC**: Training proceeds smoothly, just like the baseline, but converges to a better final loss.

Stable Training Curve (mHC)



Stability unlocks better performance.

Across a wide range of downstream benchmarks, **mHC consistently outperforms both the baseline and the unstable HC**.

It doesn't just survive training; **it learns better**.

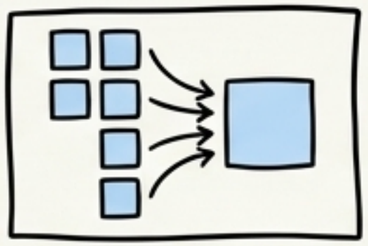
Benchmark	Baseline	HC	mHC (Winner)
BBH	43.8	48.9	**51.0** (+2.1 vs HC)
DROP (F1)	47.0	51.6	**53.9** (+2.3 vs HC)
GSM8K	46.7	53.2	**53.8**
MMLU	59.0	63.0	**63.4**

Especially strong gains on reasoning tasks like BBH and DROP.

Making it fast enough for reality.

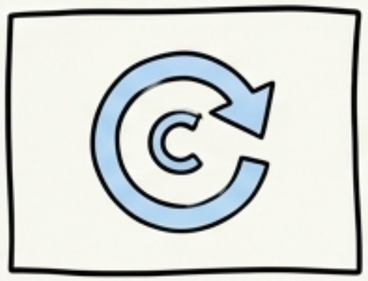
A 4x wider residual stream means more memory traffic. Without optimization, this would be too slow.

Here’s how we made it practical:



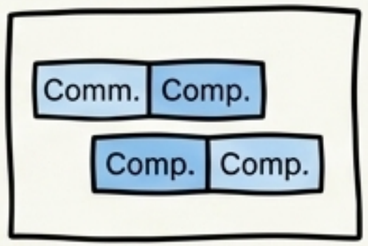
Kernel Fusion

Fusing multiple GPU operations into one to minimize memory I/O.



Selective Recomputation

Discarding some intermediate results and re-computing them during the backward pass to save memory.



DualPipe Overlapping

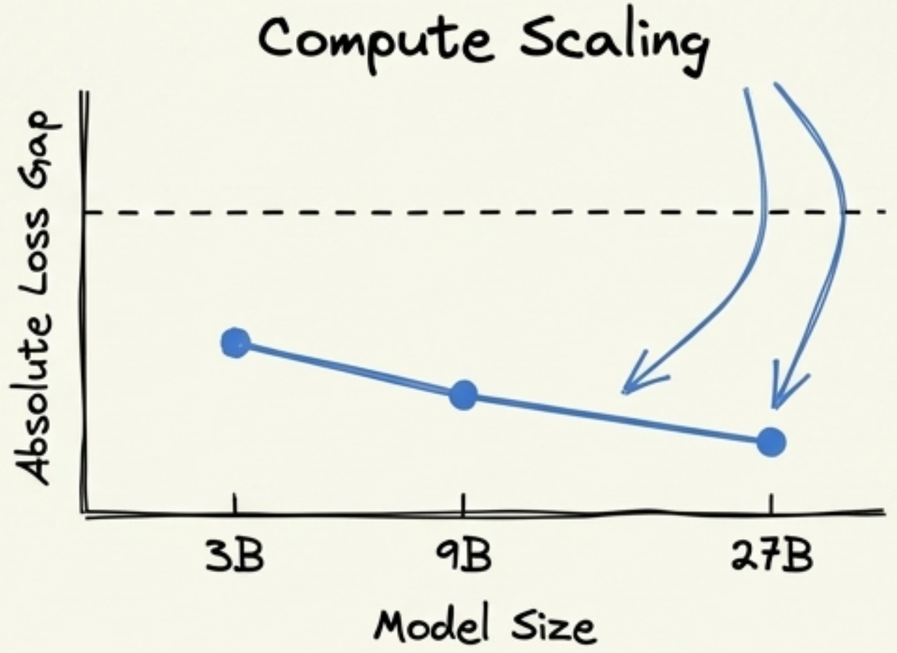
Overlapping communication and computation to keep the GPUs busy.

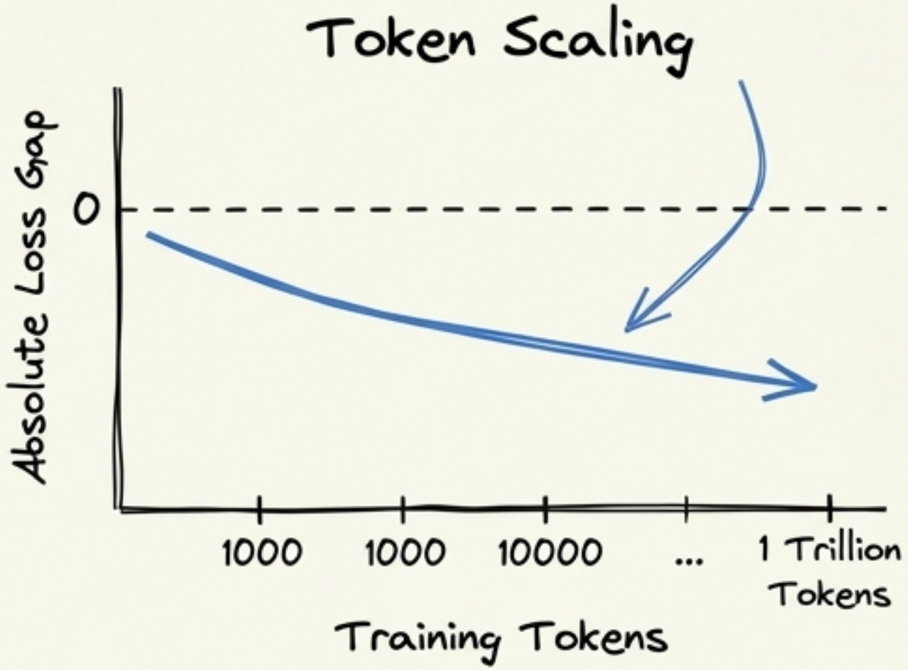
All these stability and performance gains come at a cost of only **6.7% additional training time overhead**.

The advantage holds as we scale.

The performance gains from mHC are not a fluke. They are robust across different scales.

- **Compute Scaling:** The loss improvement over the baseline is maintained as models scale from 3B to 9B to 27B parameters.
- **Token Scaling:** The advantage also persists as we train on more data (up to 1 trillion tokens).





Model scaling is no longer just about stacking layers.

mHC shows that how information flows through a model—its macro-architecture—is as important as the design of individual components.

- Mathematical constraints can unlock new, more stable architectures.
- Stability and expressiveness don’t have to be a trade-off.
- This is a blueprint for how next-generation models might scale, connecting manifold theory, architecture, and systems engineering.