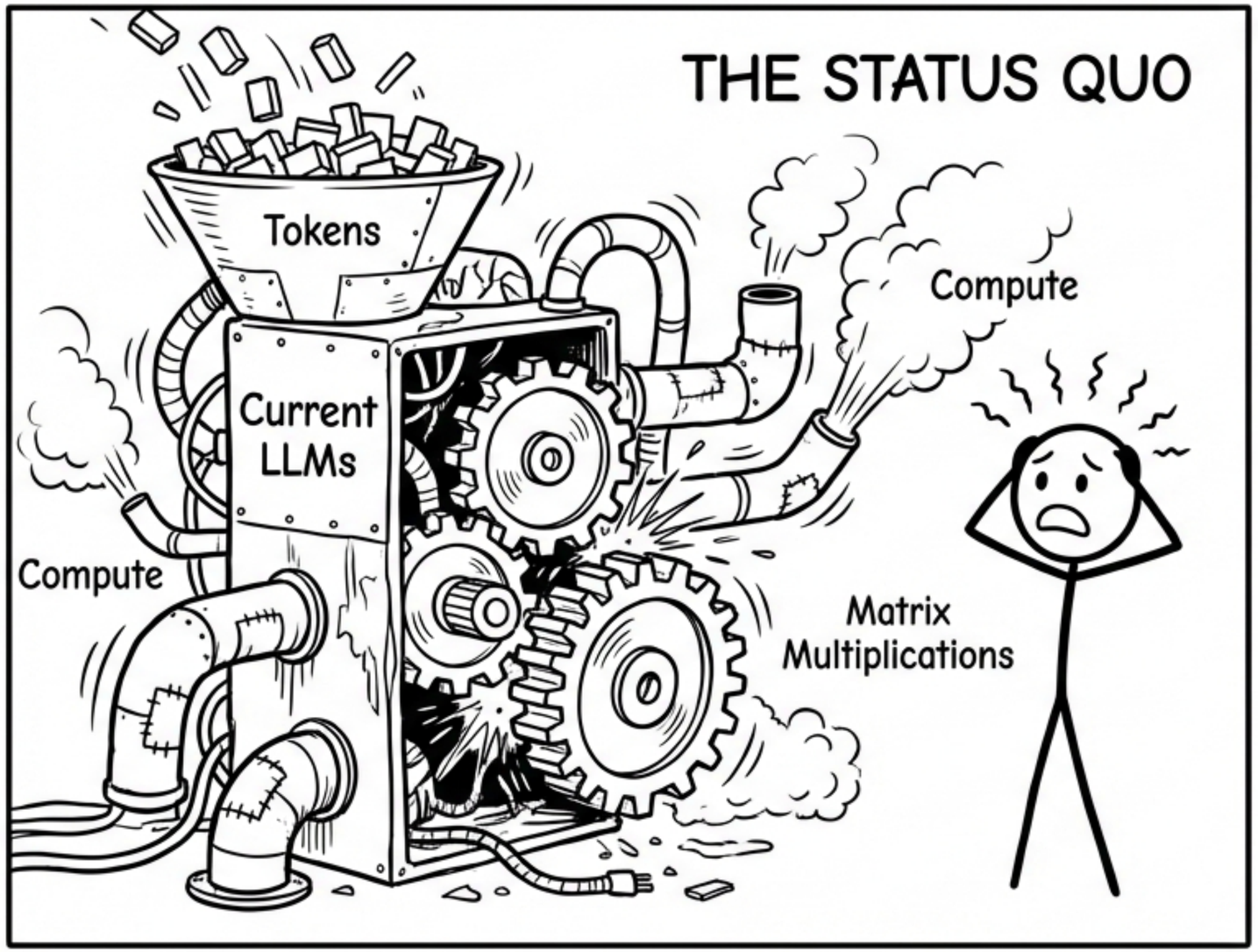
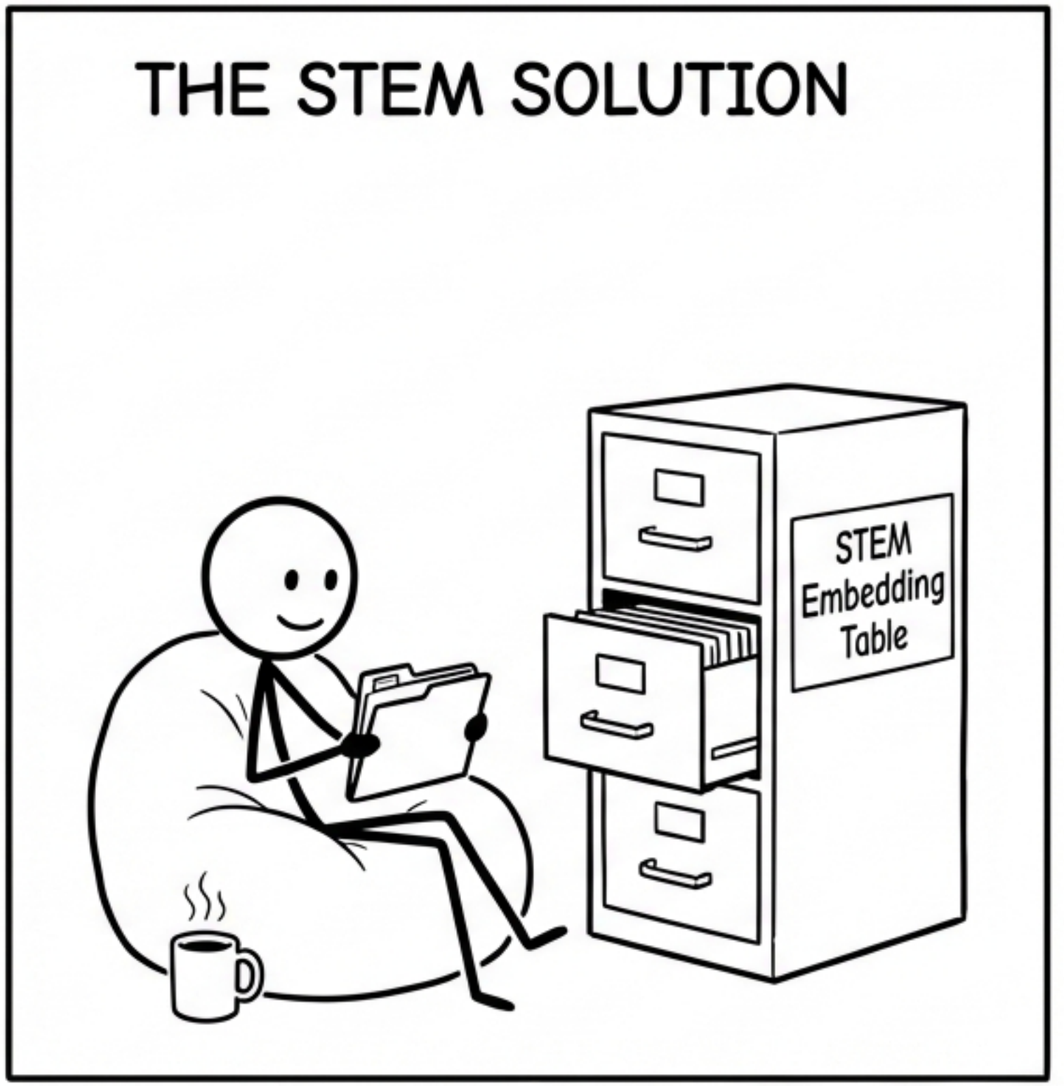
STEM: Scaling Transformers with Embedding Modules

Or: How I Learned to Stop Worrying and Love the Lookup Table.

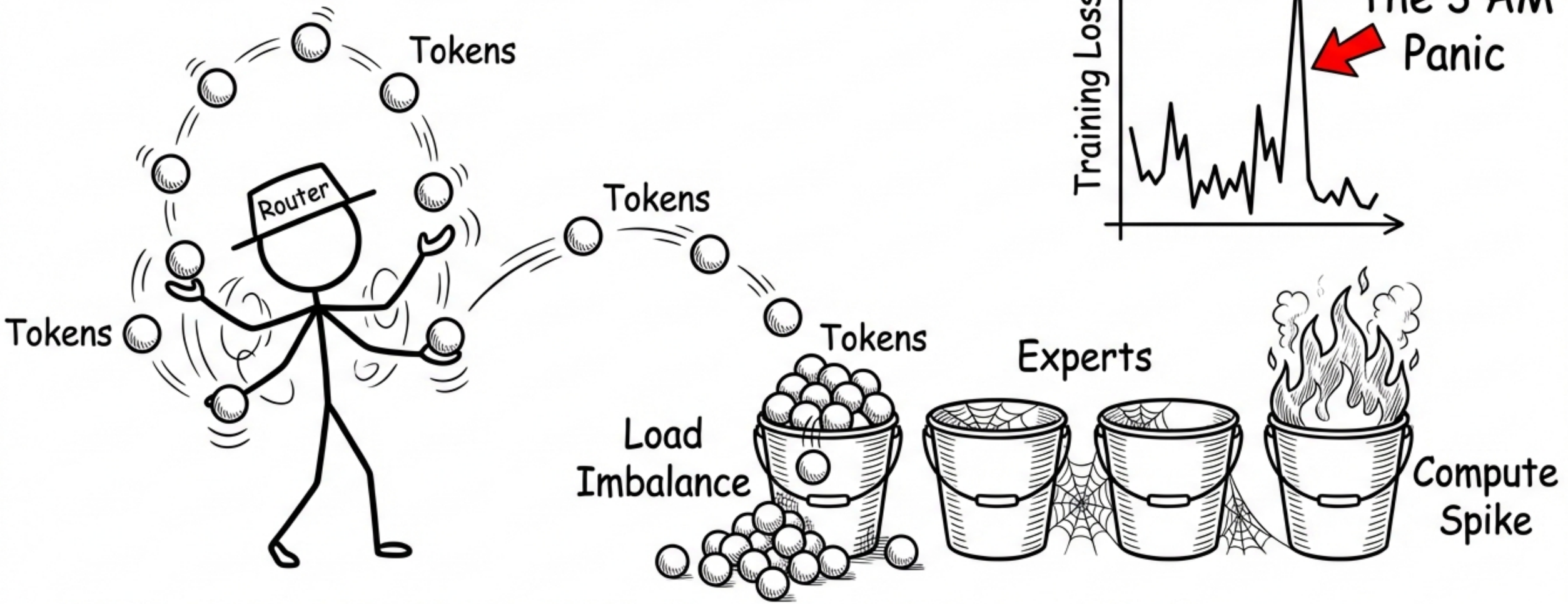




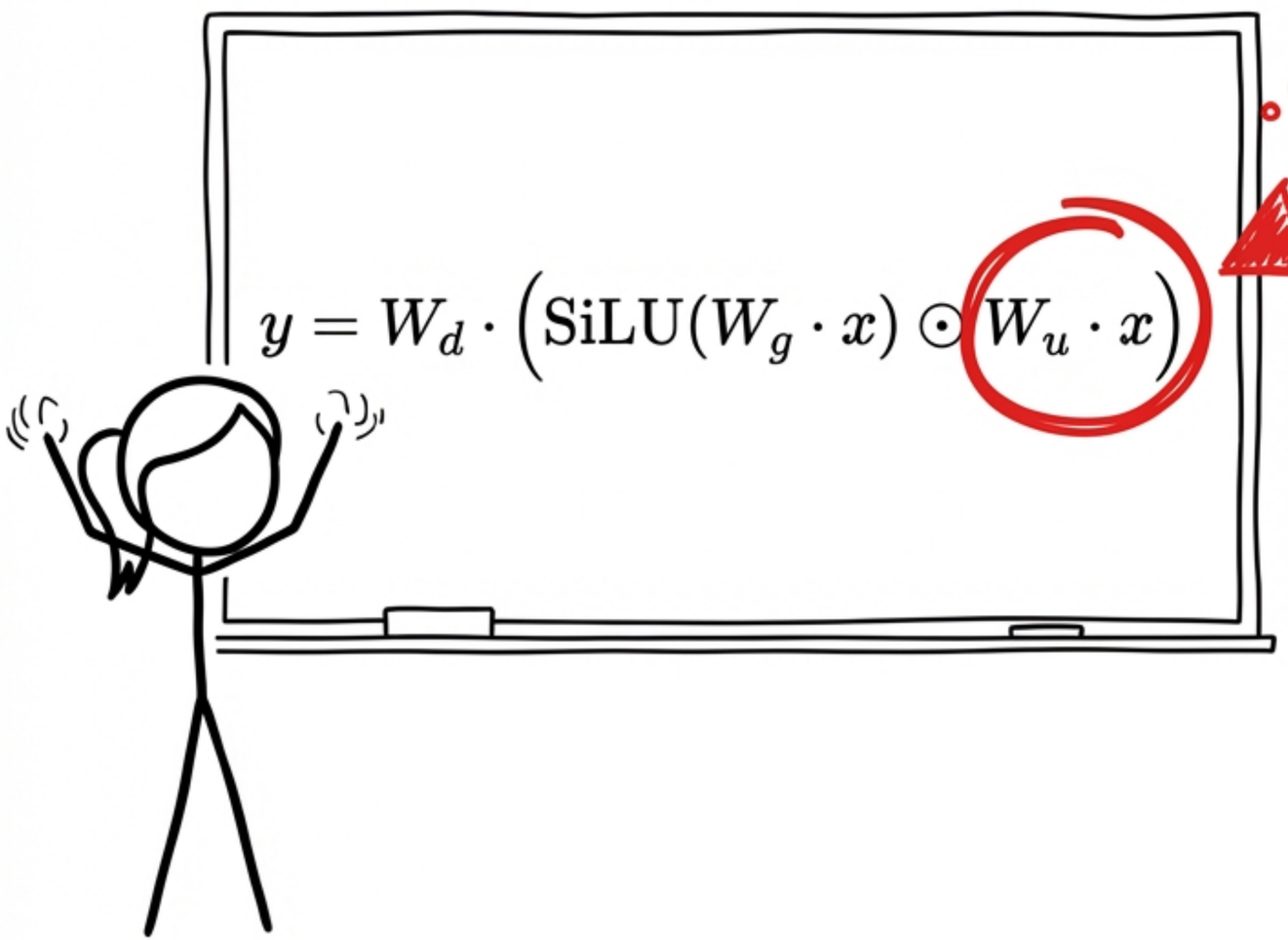
Authors: Sadhukhan, Cao, Dong, Zhao, et al. (Carnegie Mellon University & Meta AI)

(Because apparently, we spent billions on GPUs just to reinvent the

The Problem with MoE:



- Training Instability: Random loss spikes and expert under-utilization.
- Communication Overhead: Routing tokens across devices kills bandwidth.
- Complexity: Finer granularity = More headaches.



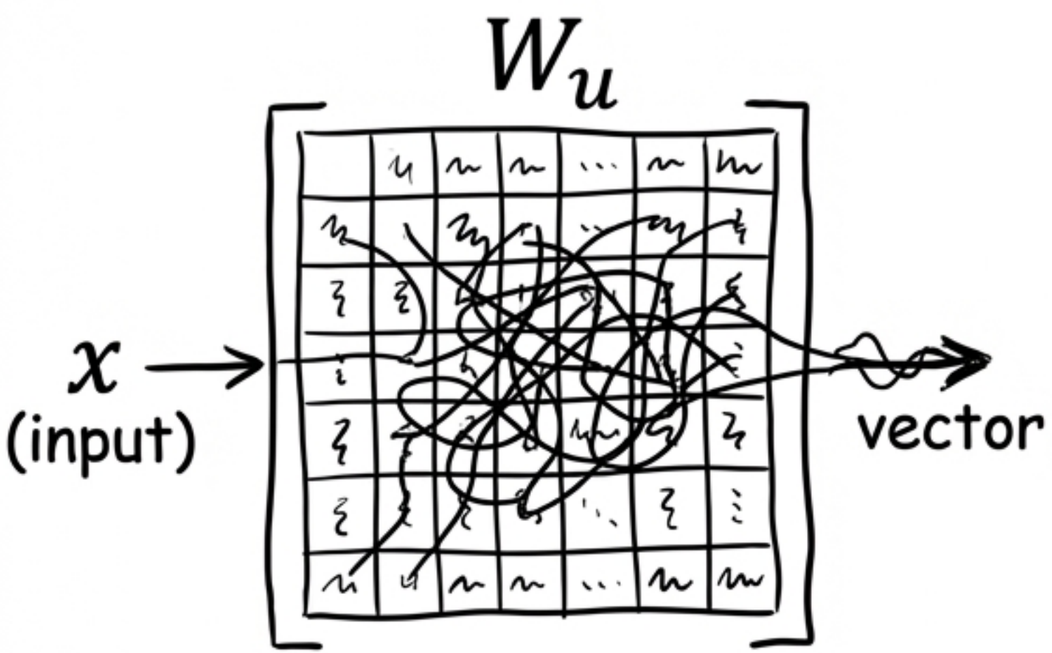
Wait... isn't this just an address lookup?

The Epiphany:

1. FFNs act as Key-Value memories.
2. The Up-Projection (W_u) just creates a key to find information.
3. If we already know the Token ID... why do massive matrix math to find its address?

The Architecture Swap

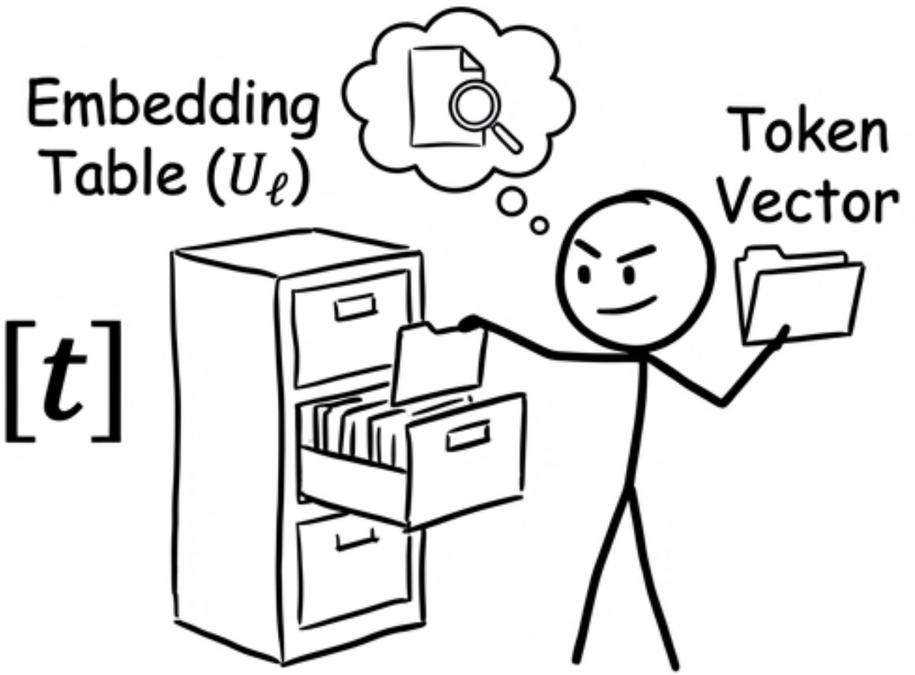
Dense FFN (Old)



Heavy Matrix Math.

$$W_u \cdot x \rightarrow U_\ell[t]$$

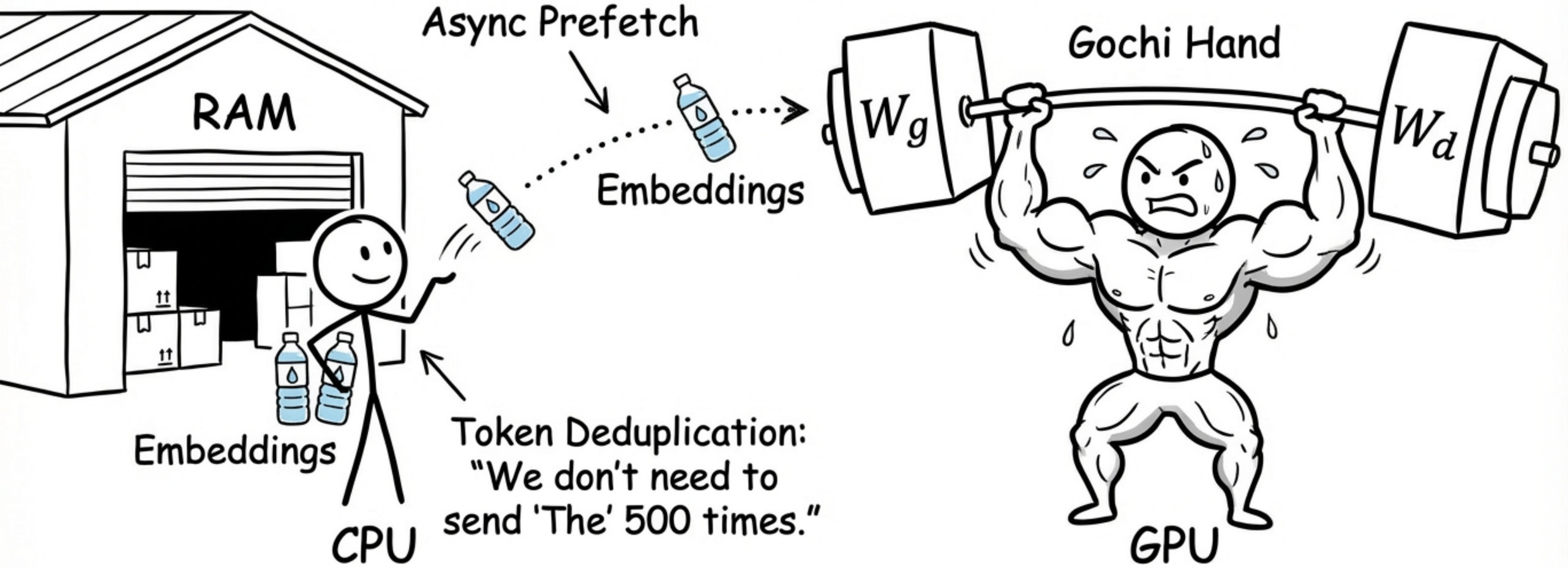
STEM (New)



Just grab the file.

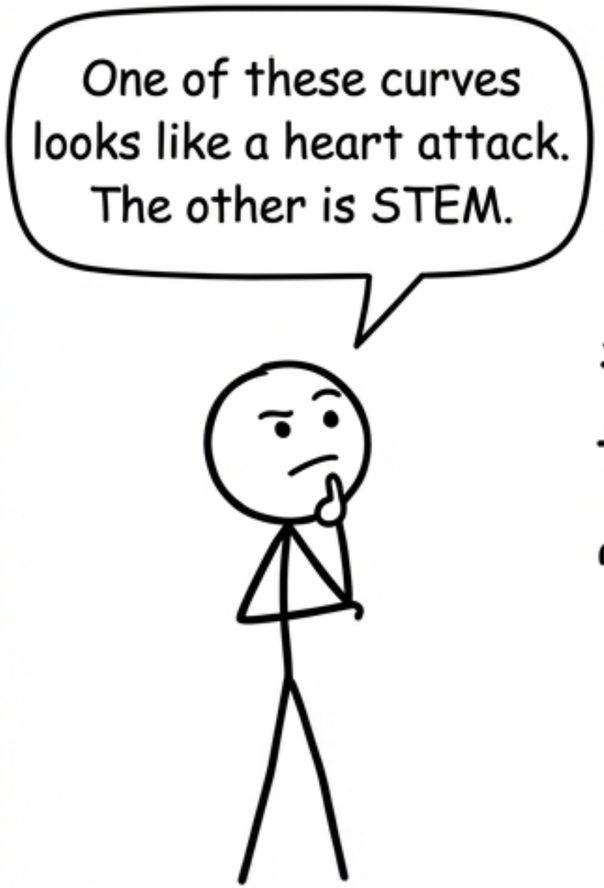
We keep the Gate (W_g) and Down-Projection (W_d) dense.
We only replace the Up-Projection with a static lookup.

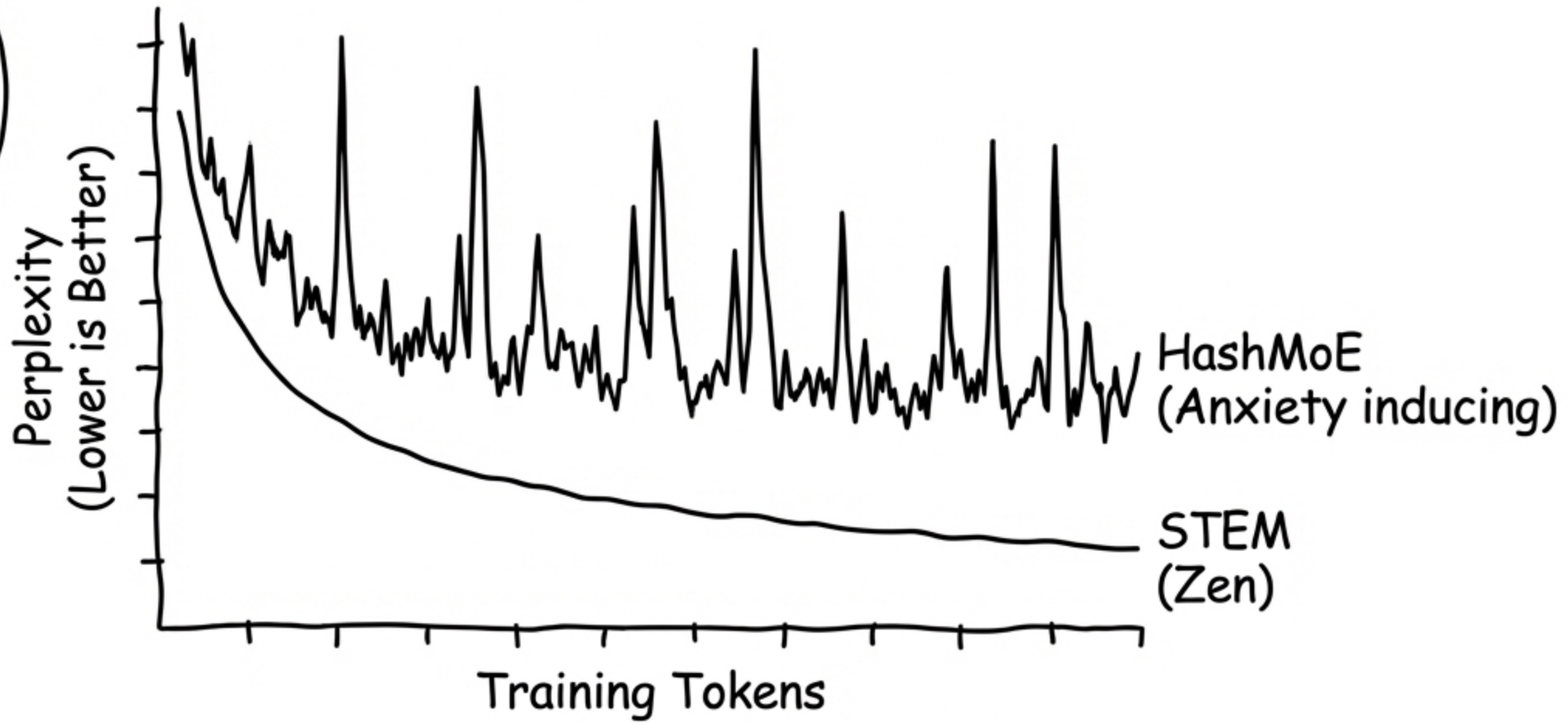
CPU Offloading



Efficiency: Eliminates ~1/3 of FFN parameters from active memory.

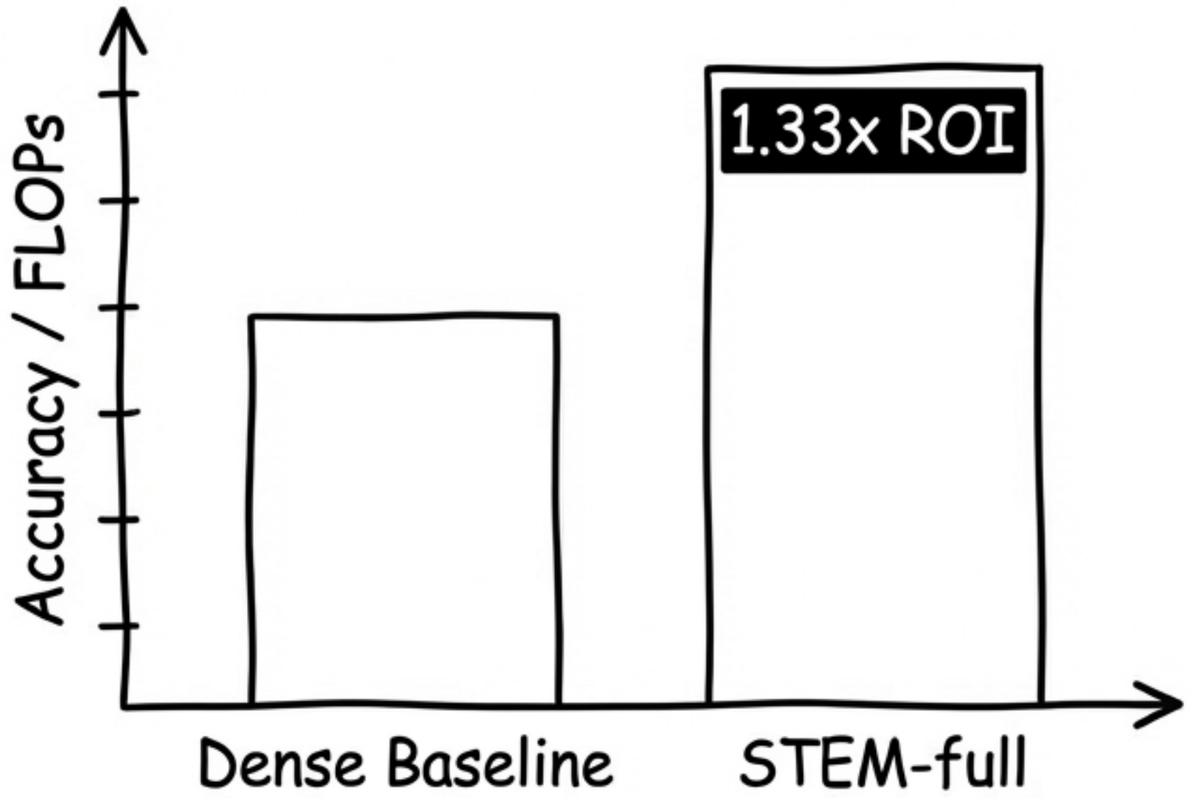
Validation Perplexity vs. Training Tokens

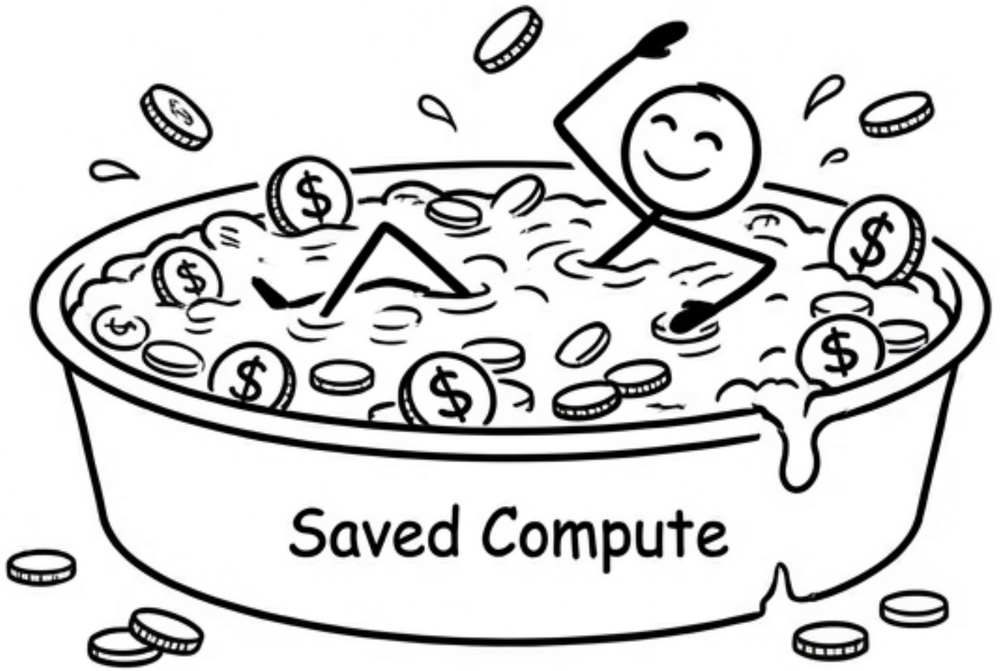




- Zero loss spikes.
- Lower perplexity with fewer FLOPs.
- Proven on 350M and 1B scales.

Training Return on Investment (ROI)



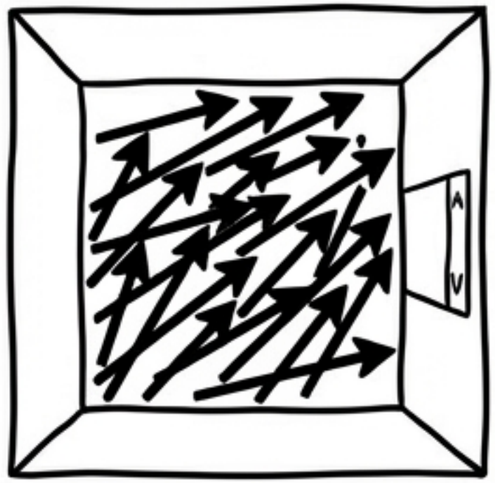


The Metric: Training ROI = Accuracy / FLOPs.
The Result: Replacing FFNs with STEM yields 1.33x the ROI.
The Saving: Reduces per-token FLOPs by ~33% (by deleting the Up-Projection).

We get smarter models for cheaper. Finance is happy.

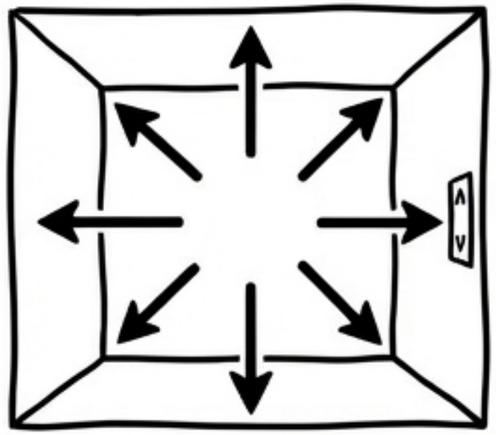
Angular Spread (Geometry)

Dense FFN
(High Cosine Similarity)



Crowded. Hard to find your car.

STEM
(Large Angular Spread)

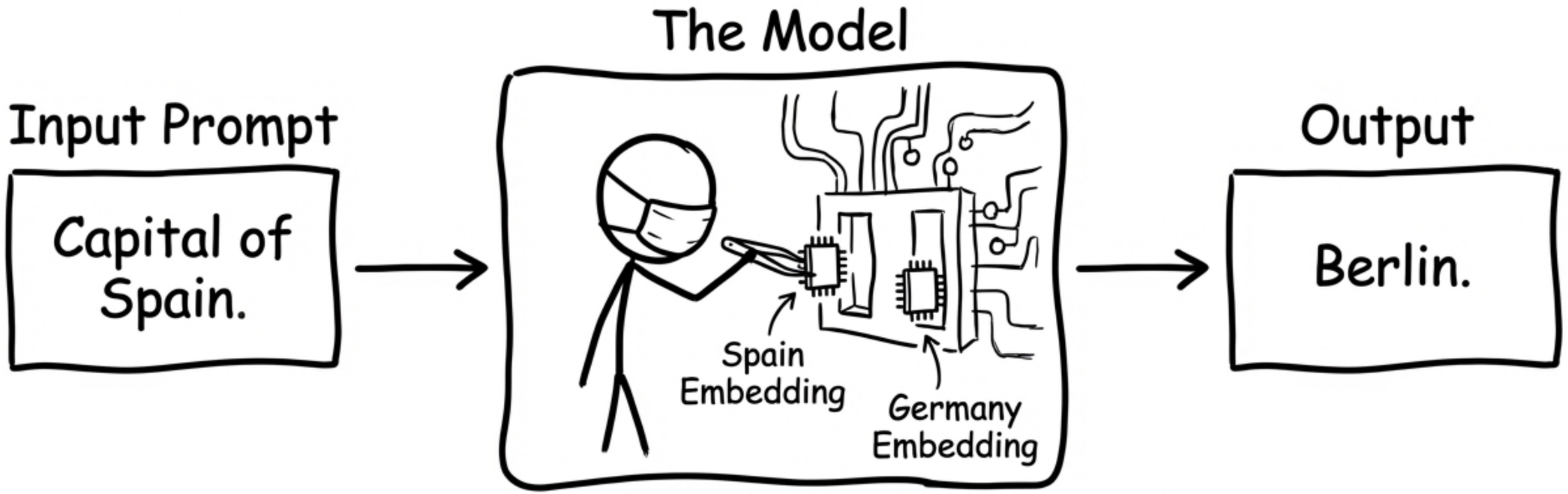


Room for activities.
Distinct concepts.

- STEM embeddings naturally learn to respect each other's personal space.
- Low pairwise cosine similarity = Less interference.
- Result: Better Knowledge Storage Capacity.

Interpretability & Knowledge Editing:

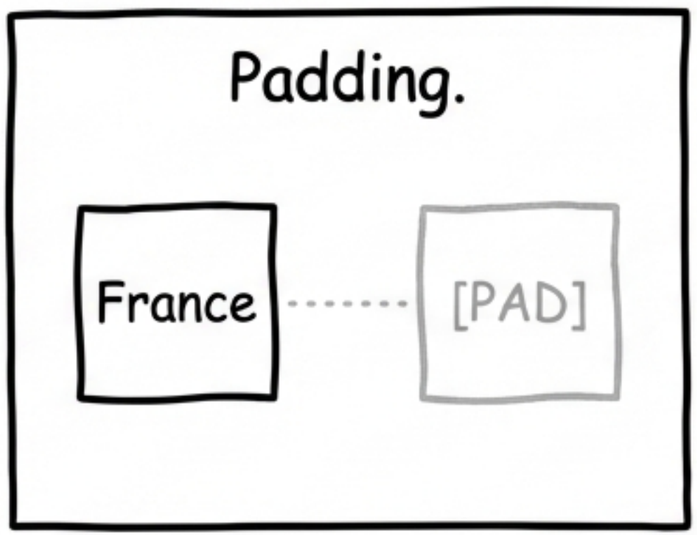
Since embeddings are tied to Token IDs, we know exactly where "Spain" lives. We can swap facts without retouching weights or the prompt.



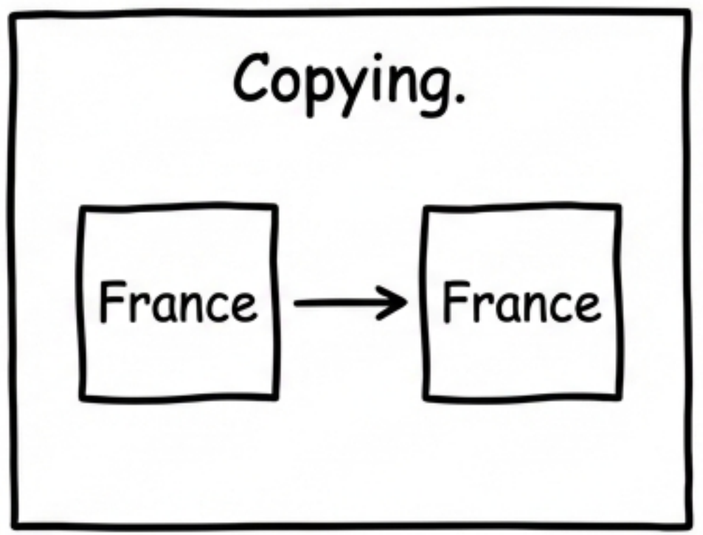
Quote: "Surgical intervention without the mess."

Hacking the Token Length Mismatch

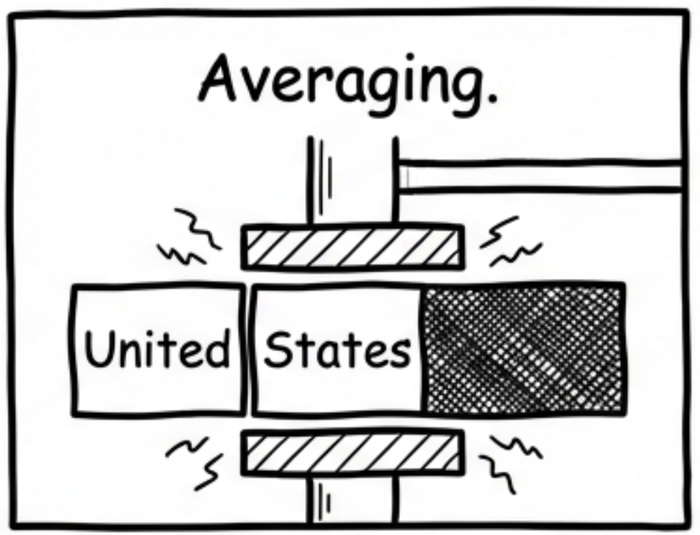
How to turn "France" (1 token) into "United States" (2 tokens).



The Filler Strategy.



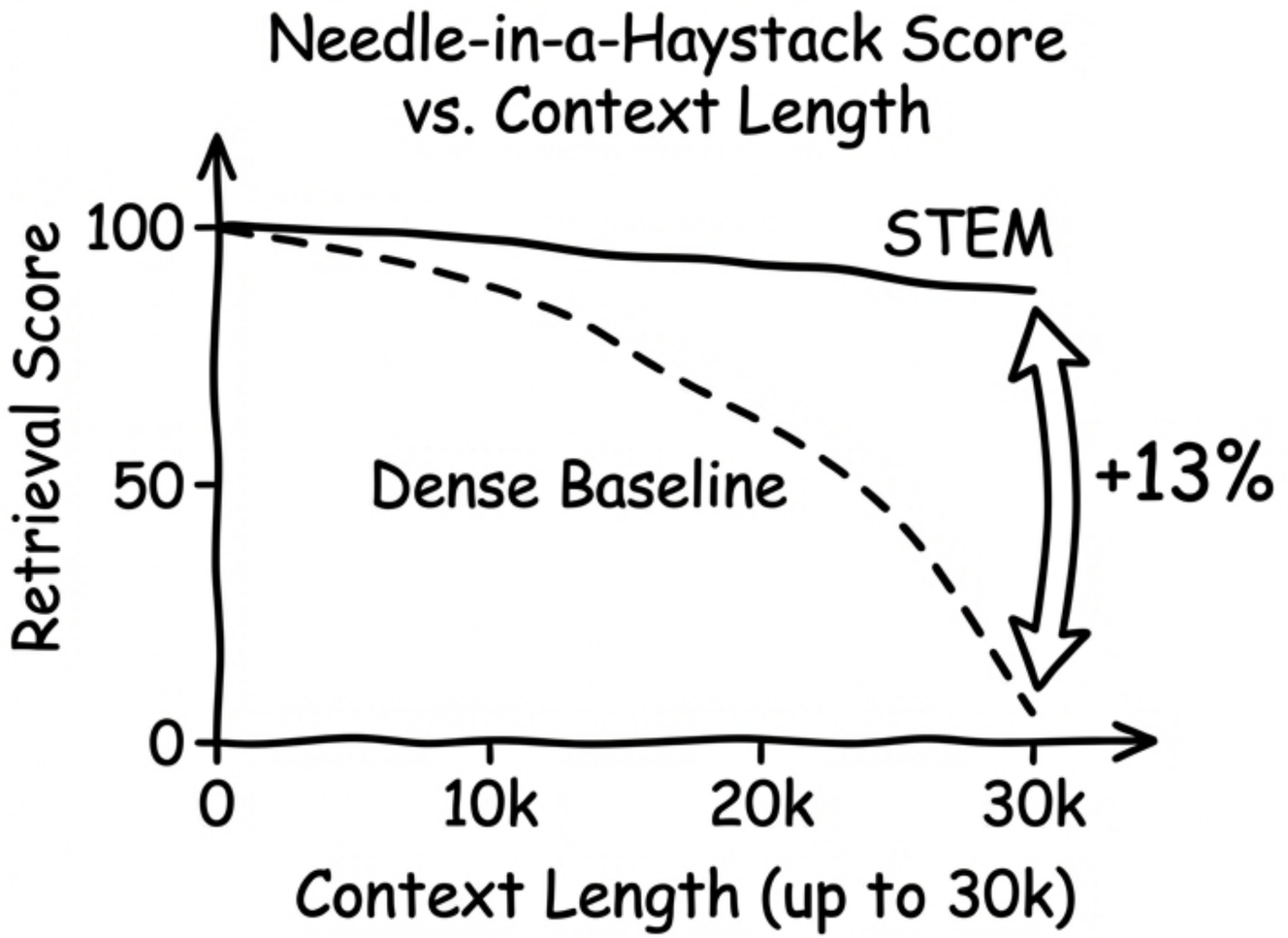
The Clone Strategy.

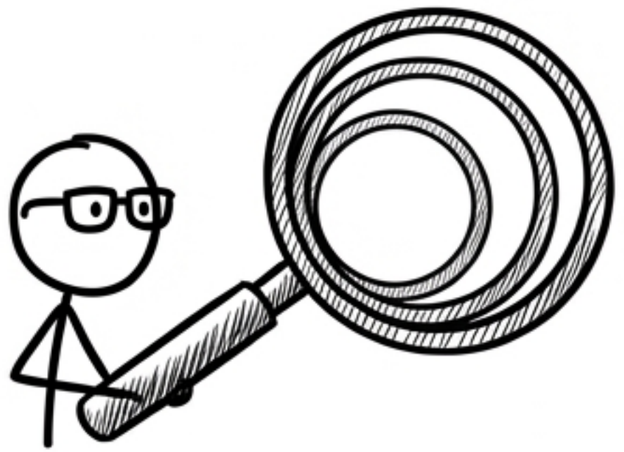


The Diplomatic Solution (Averaging).

Challenge: Source and Target words have different lengths.
Result: All these methods work surprisingly well. The model adapts instantly.

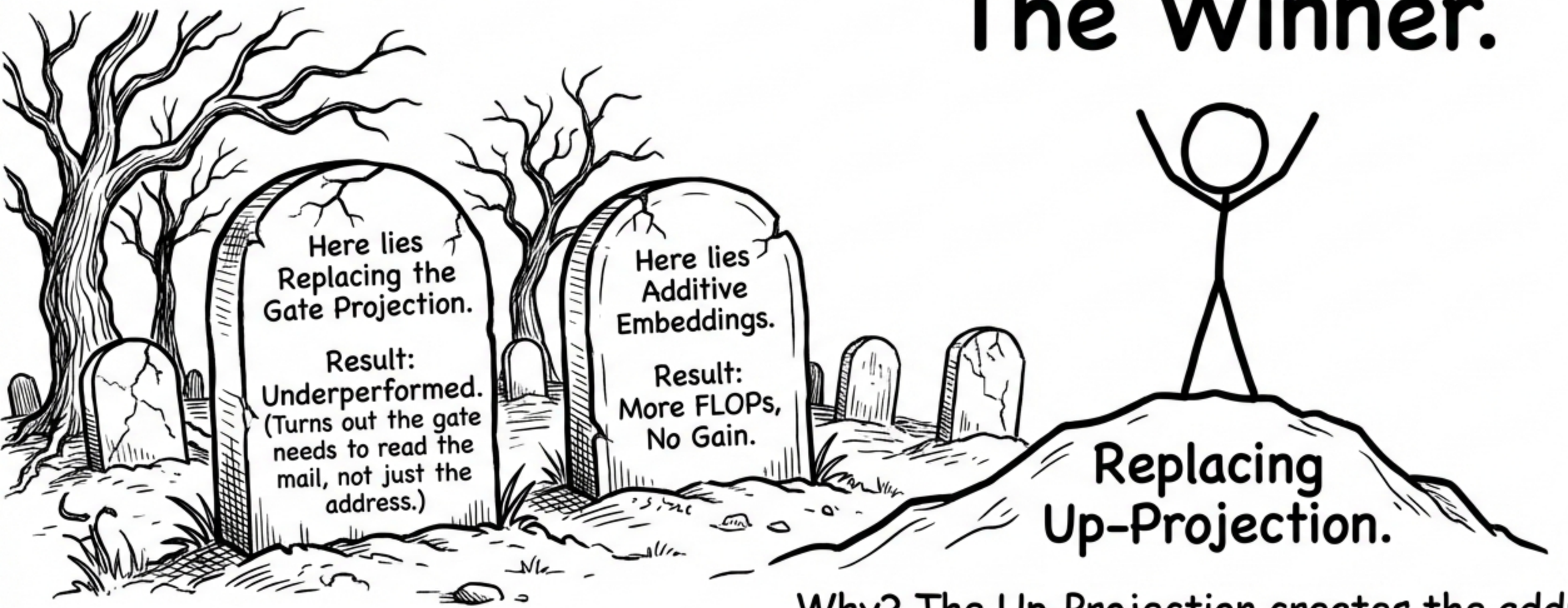
Test-Time Capacity Scaling: More Reading = Bigger Model





As context gets longer, more unique tokens appear.
→ More unique tokens = More STEM embeddings activated.
→ The model effectively gets bigger the more you read.

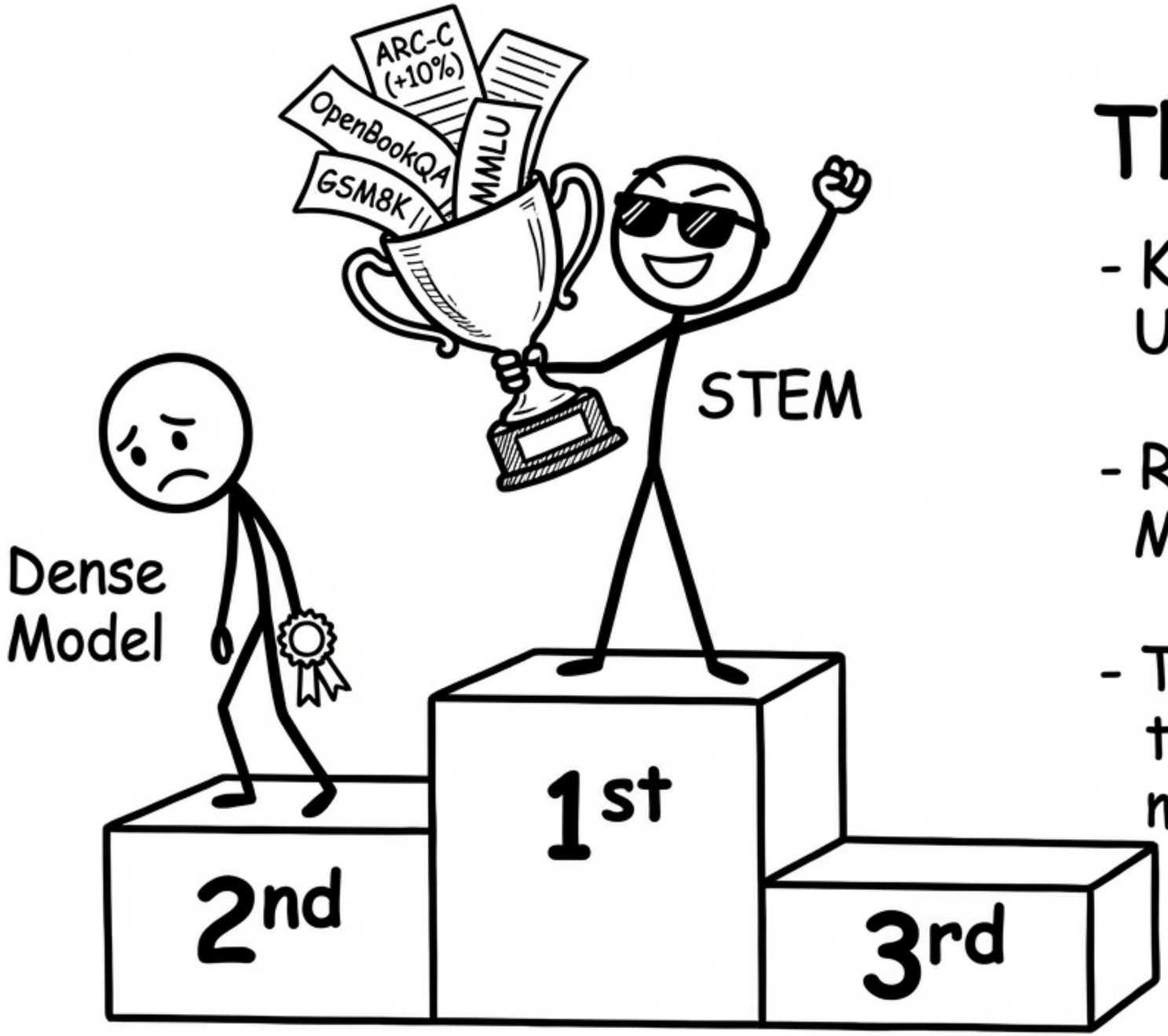
The Graveyard of Failed Approaches



The Winner.

Replacing Up-Projection.

Why? The Up-Projection creates the address.
The Gate reads the mail.
We made the address static (Token ID)
but kept the reader dynamic.

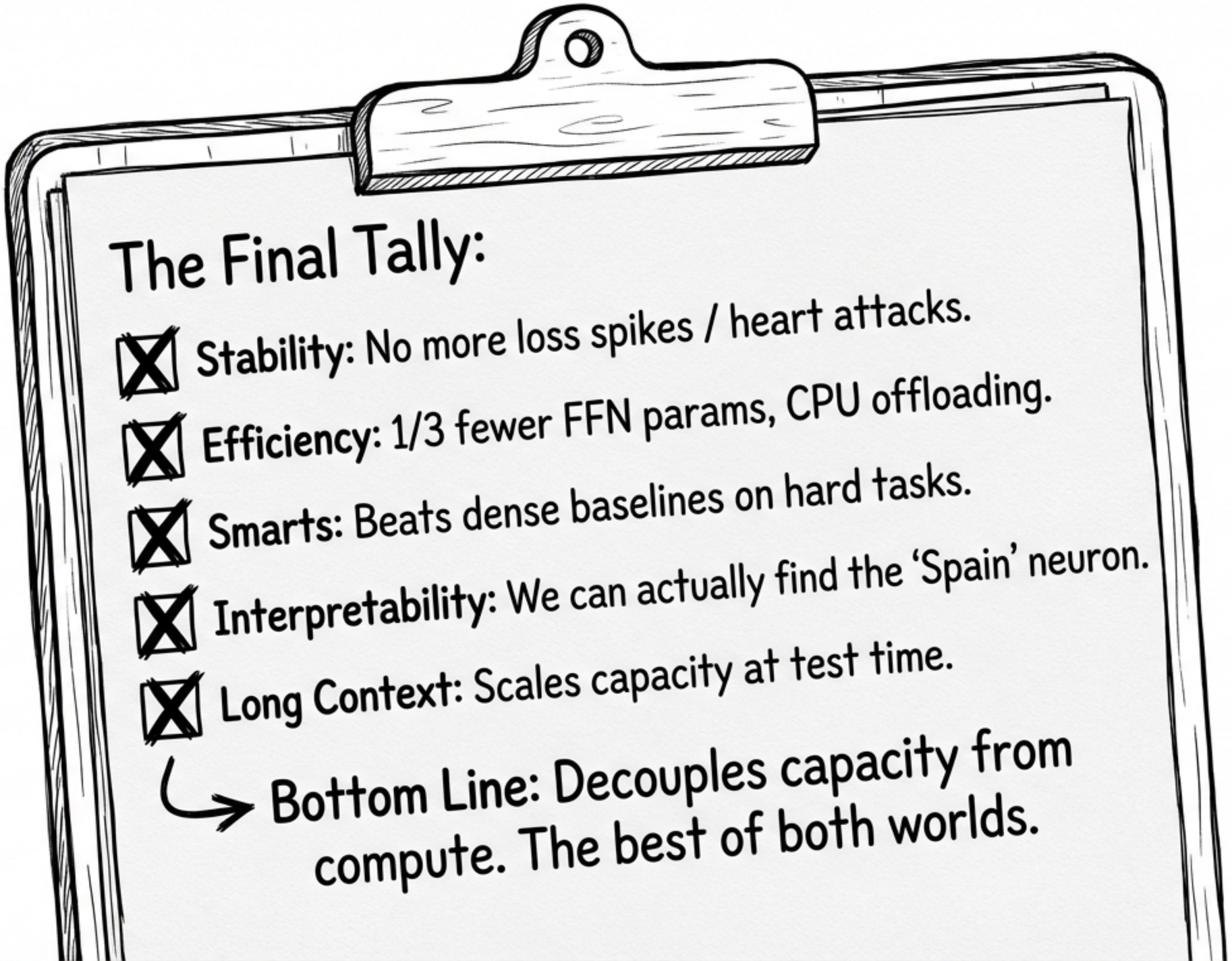


The Scoreboard:

- Knowledge Intensive Tasks: Up to 9-10% improvement.

- Reasoning: Strong gains on Math and Multi-tasking.

- Takeaway: Dumbing down the architecture actually made it smarter.


The Final Tally:
☒ Stability: No more loss spikes / heart attacks.
☒ Efficiency: 1/3 fewer FFN params, CPU offloading.
☒ Smarts: Beats dense baselines on hard tasks.
☒ Interpretability: We can actually find the 'Spain' neuron.
☒ Long Context: Scales capacity at test time.
↳ Bottom Line: Decouples capacity from compute. The best of both worlds.

REFERENCES

Paper: STEM: Scaling Transformers with Embedding Modules (2026)

Team: Sadhukhan, Cao, Dong, Zhao, et al.

Code: https://github.com/Infini-AI-Lab/STEM