Scaling Laws for Neural Language Models

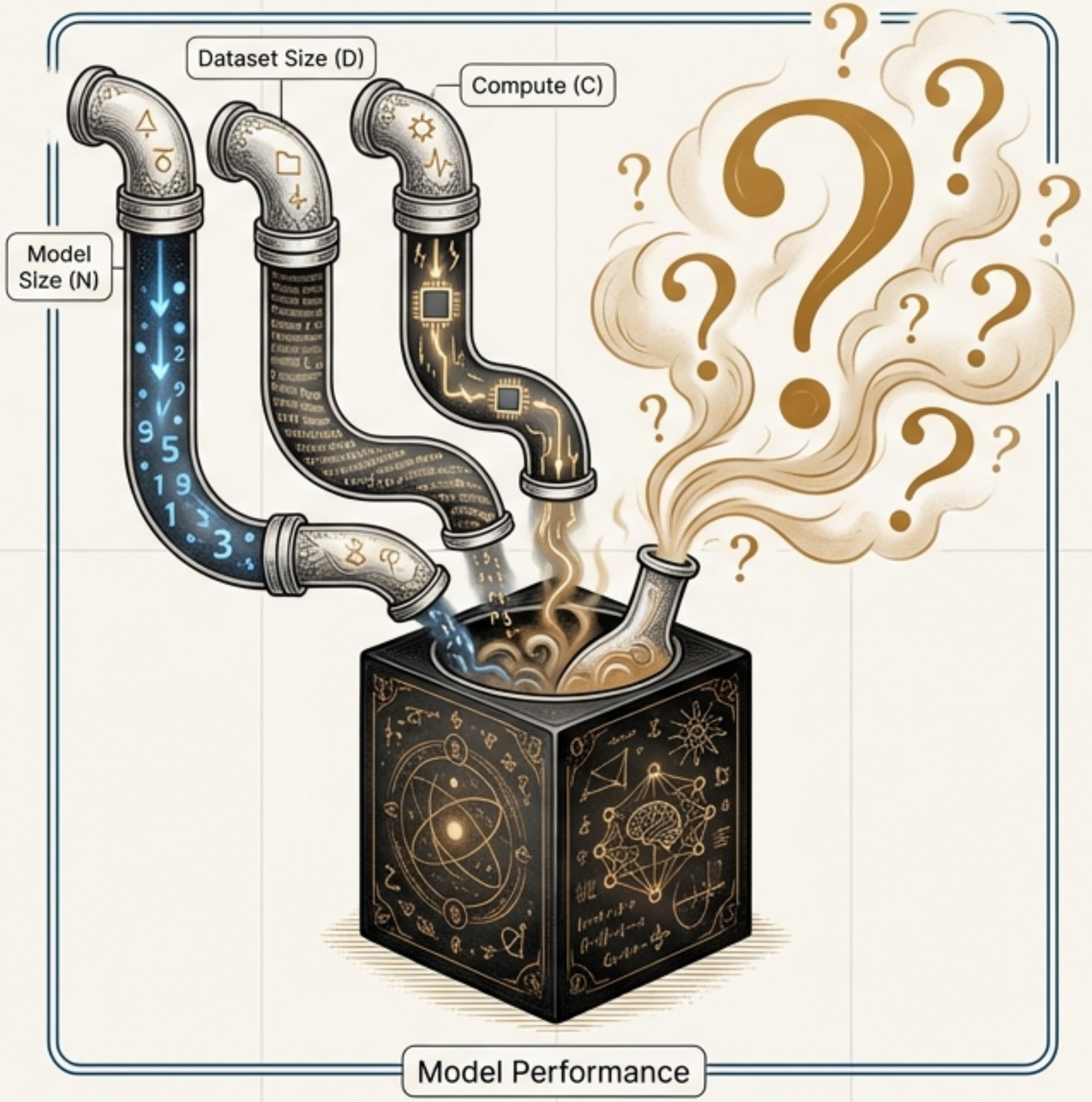
Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, Dario Amodei

OpenAI

The Scaling Problem: Are We Navigating Blind?

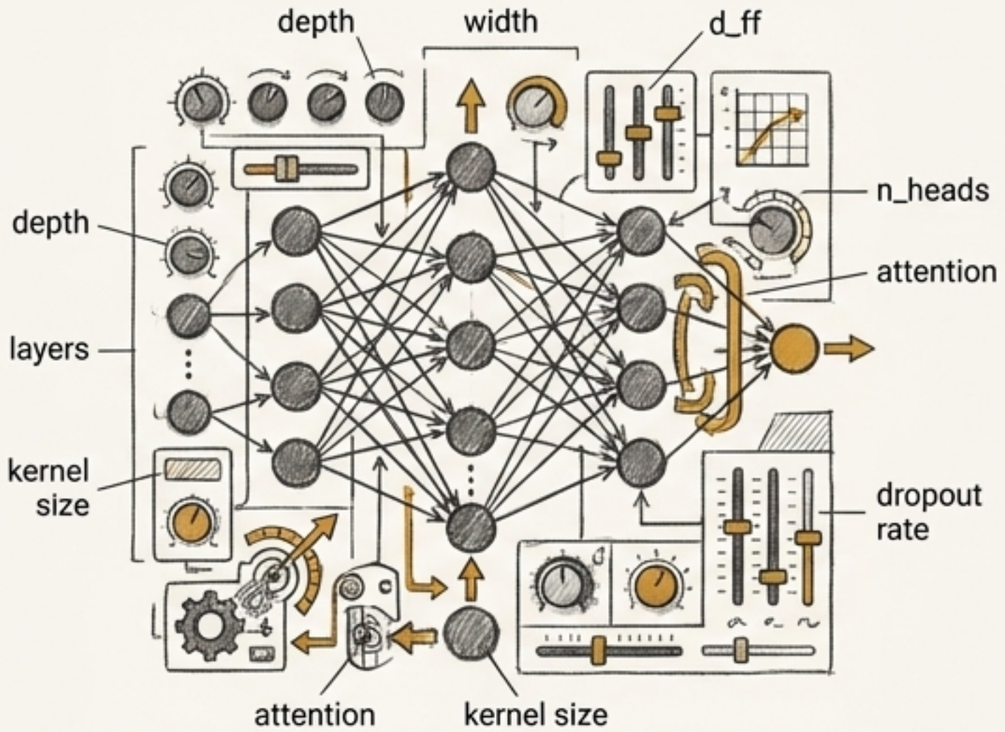
Training state-of-the-art language models requires massive investment in compute, data, and engineering. Yet, our approach is often heuristic.

- We mix ingredients (parameters, data, compute) hoping for optimal performance, but without a predictive formula.
- This ‘alchemy” is inefficient, resource-intensive, and carries significant risk.
- How should we allocate a given compute budget? More data? A bigger model? More training steps? The answers are not obvious.



From Architectural Tinkering to Universal Laws

The Old Way



- Focus on tweaking architecture: depth vs. width, number of attention heads, feed-forward dimension.
- Results are often model-specific and don't generalize into predictable principles.

The New Question

$$L = f(N, D, C)$$

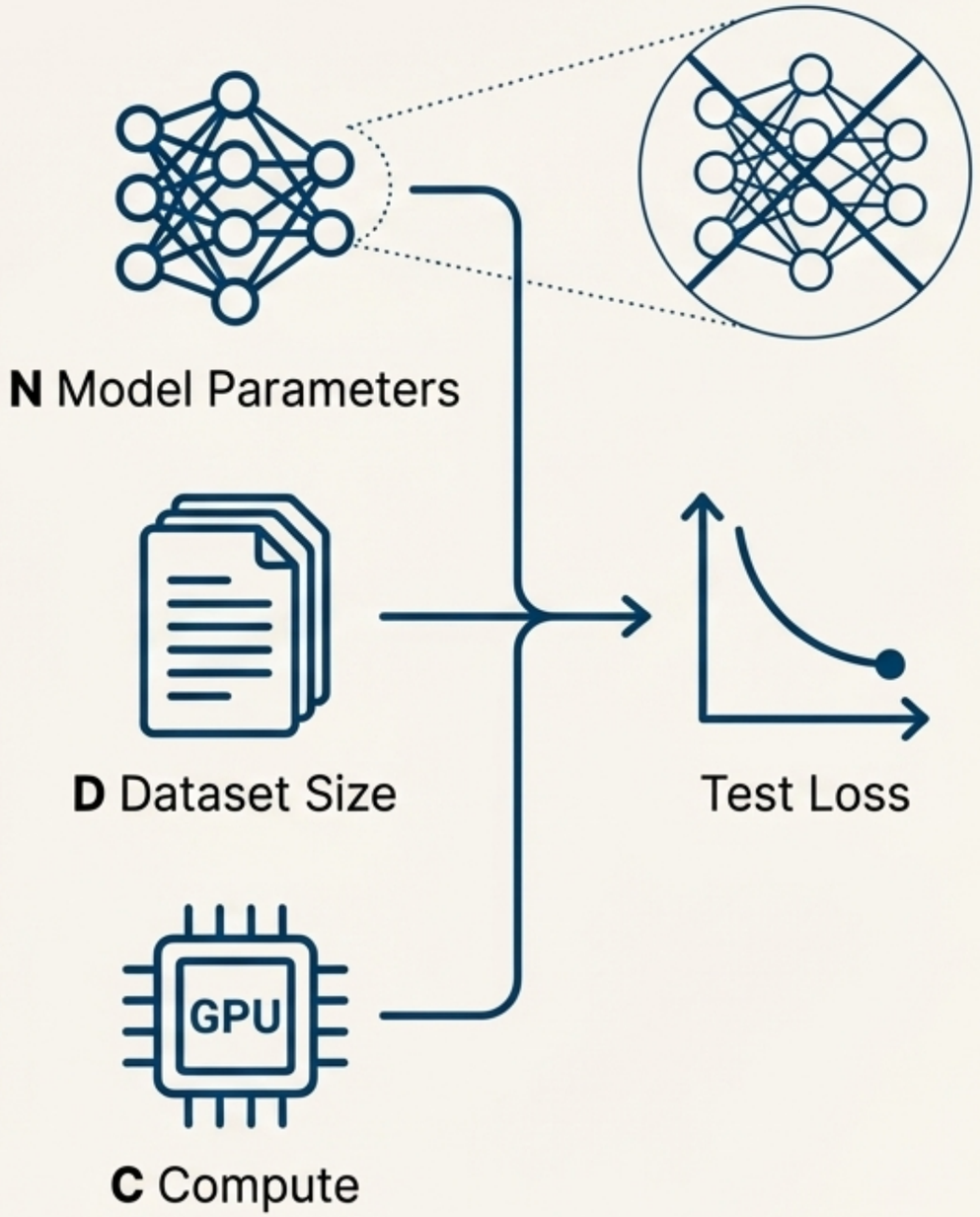
- Can we discover predictable, quantitative relationships that govern model performance at scale, independent of minor architectural changes?

A Principled Approach: Isolating the Variables of Scale

To find the laws, we must first isolate the core components of scale. The study focuses on three key factors:

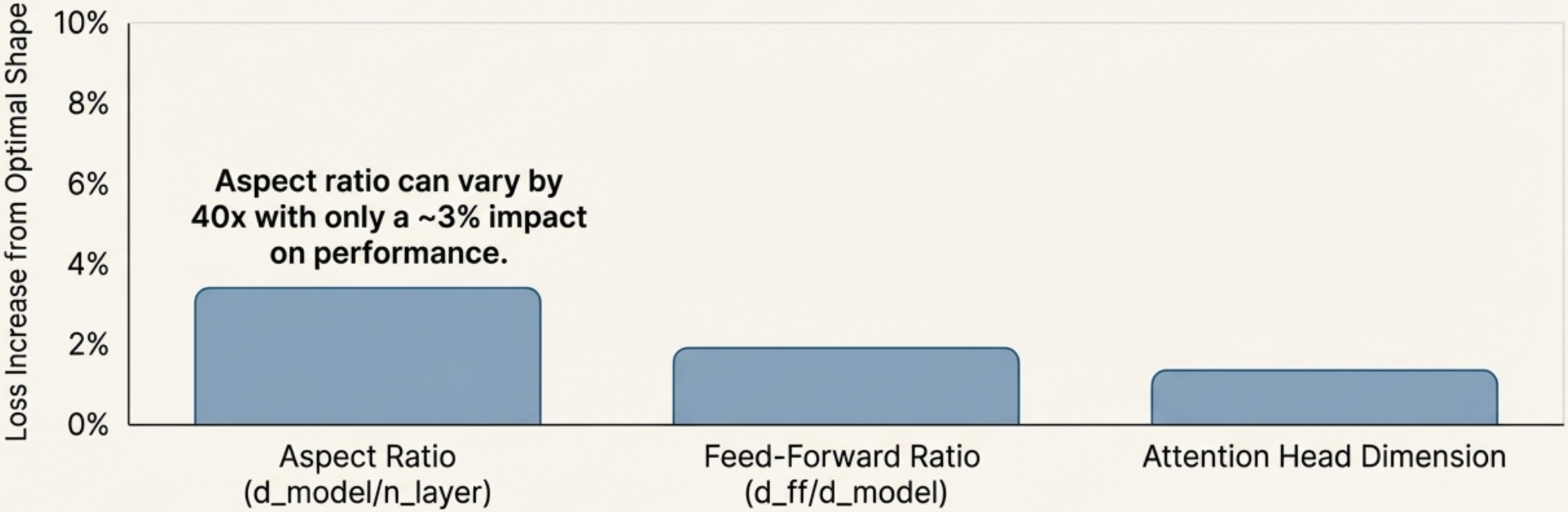
- **N**: The number of non-embedding model parameters.
- **D**: The size of the dataset in tokens.
- **C**: The total training compute (estimated as $C \approx 6NBS$).

A Crucial Distinction: The study deliberately excludes vocabulary and positional embedding parameters from **N**. This decision reveals significantly cleaner, more predictable scaling laws.



Finding #1: Scale is King, Shape is Secondary

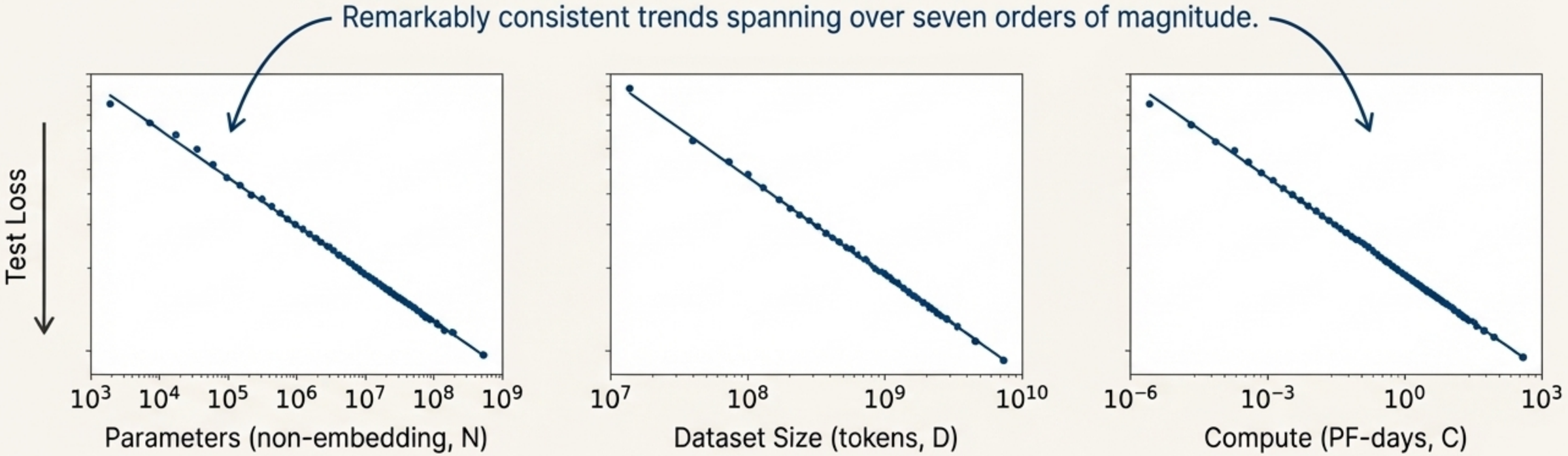
Performance depends very weakly on model **shape (depth, width, etc.) when** the number of non-embedding parameters (N) is held constant.



For example, a “wide and shallow” model (6 layers, d_model=4288) performs within 3% of a “deep and narrow” model (48 layers, d_model=1600).

Finding #2: Performance Scales as Predictable Power Laws

Language model performance (cross-entropy loss) improves smoothly and predictably with scale. The relationship is not chaotic but follows a power law.



- Loss scales as a power law when Model Size (N), Dataset Size (D), or Compute (C) is the limiting factor.
- There are no signs of deviation at the largest scales tested, suggesting the laws hold for even bigger models.

The Laws of Scale: Quantifying Performance

We can express these predictable relationships as simple equations.

Model Size (N)

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

$$\alpha_N \approx 0.076$$

Dataset Size (D)

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}$$

$$\alpha_D \approx 0.095$$

Compute (C_{min})

$$L(C_{\min}) = \left(\frac{C_c}{C_{\min}}\right)^{\alpha_C}$$

$$\alpha_C \approx 0.050$$

The exponents (α) quantify the performance improvement. For example, doubling model size (N) reduces the loss by a factor of $2^{-0.076}$, or about 5%.

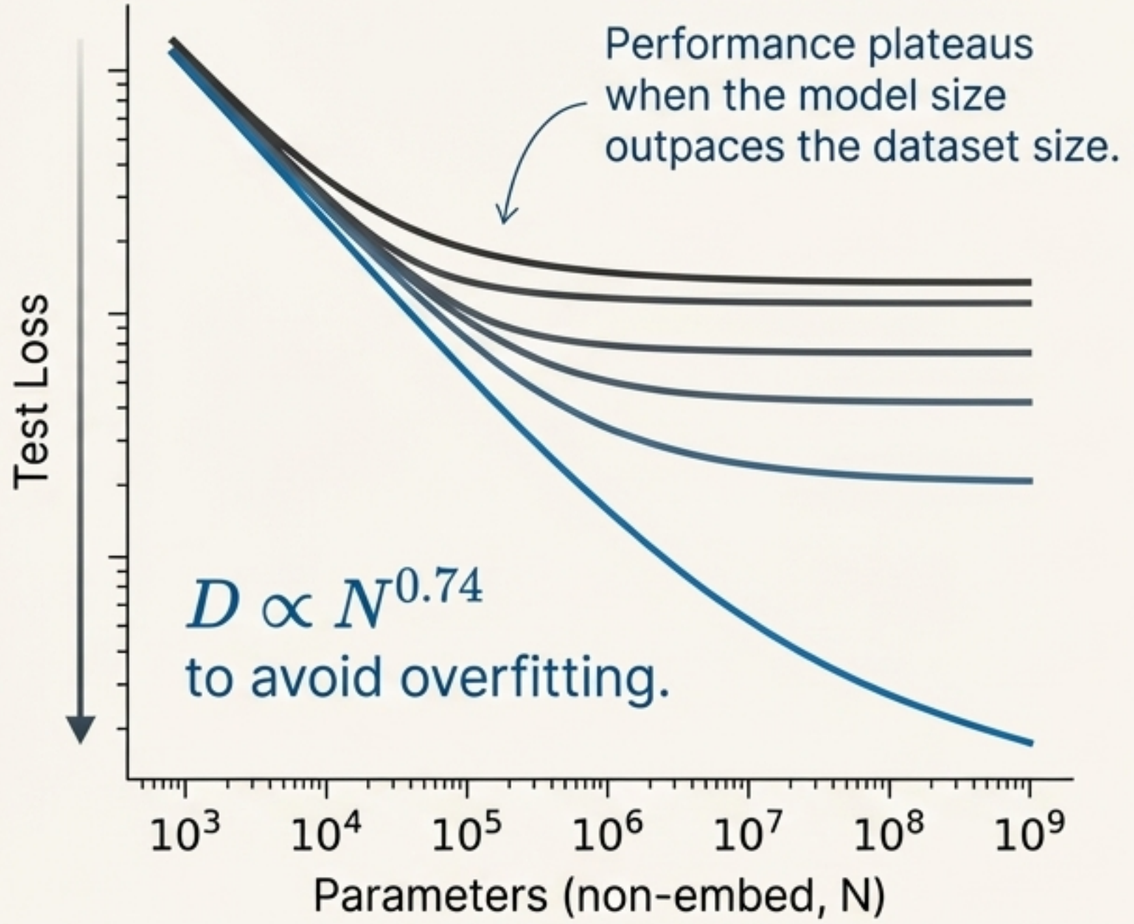
Finding #3: Overfitting is No Longer a Mystery

How much data do we need as we scale up model size?

$$\mathcal{L}(N, D) = \left[\left(\frac{N_c}{N}\right)^{\alpha N/\alpha D} + \left(\frac{D_c}{D}\right]^{\alpha D}$$

Data requirements grow sub-linearly with model size.

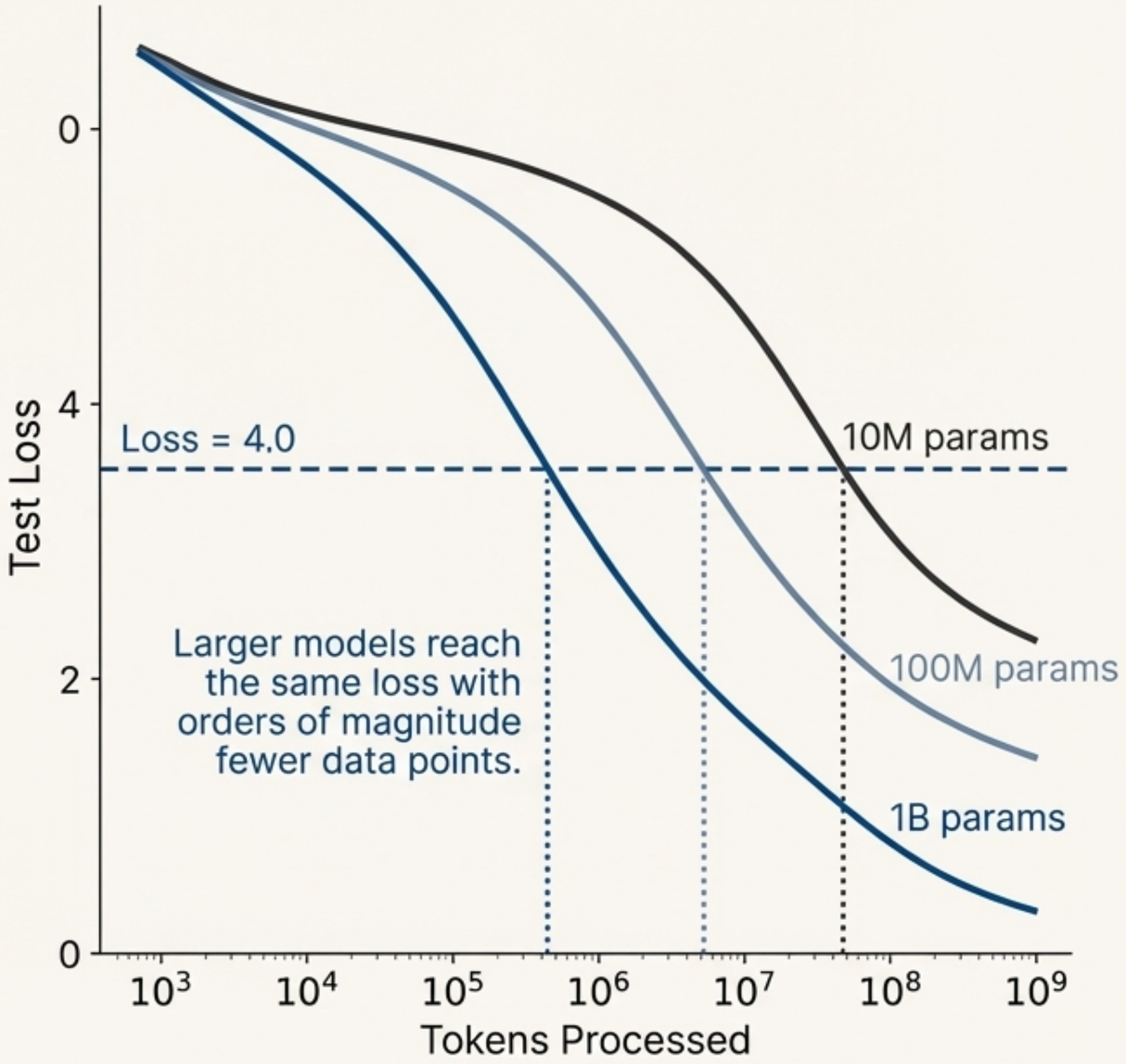
- The performance penalty depends on the ratio $N^{0.74}/D$.
- To maintain performance, every 8x increase in model size requires only a ~5x increase in dataset size.



Implication #1: Larger Models are Dramatically More Sample-Sample-Efficient

The scaling laws have a direct and powerful consequence: larger models learn faster from data.

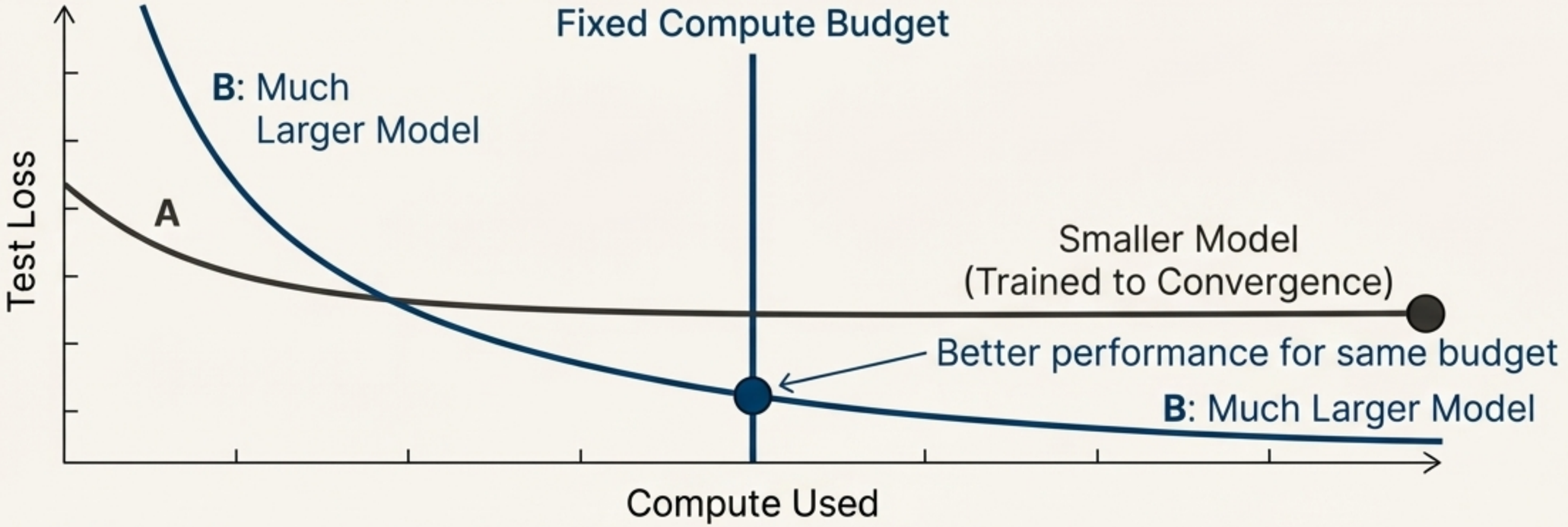
- A larger model reaches the same performance level (test loss) with fewer training steps and fewer data points.
- This holds true across the entire training process.
- Sample efficiency improves significantly with model scale, making large models a more efficient way to extract information from a dataset.



Implication #2: The Optimal Strategy is Not What You Think

The Conventional Wisdom: Train models for a long time, until their performance converges.

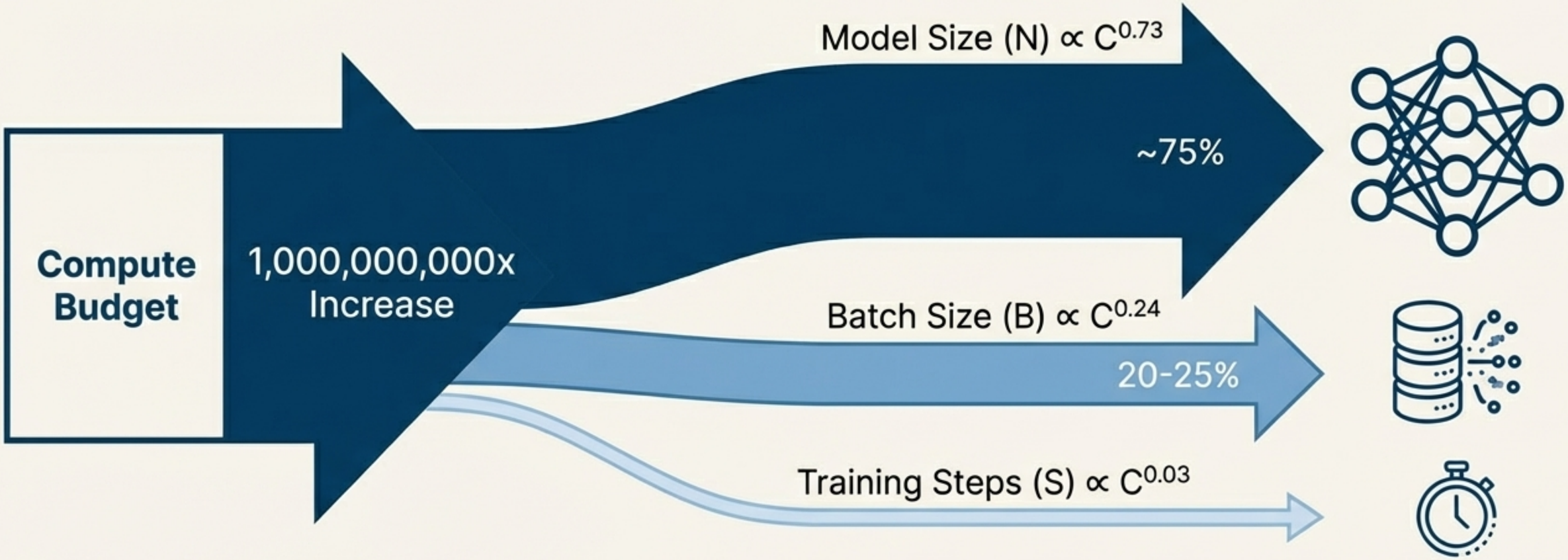
The New Rule: For a fixed compute budget, the optimal strategy is to train a very large model and stop ***significantly short of convergence.***



Why it Works: The steep, initial learning curve of a very large model yields a better loss for a given amount of compute than training a smaller model to its performance plateau.

A New Rulebook for Allocating Compute

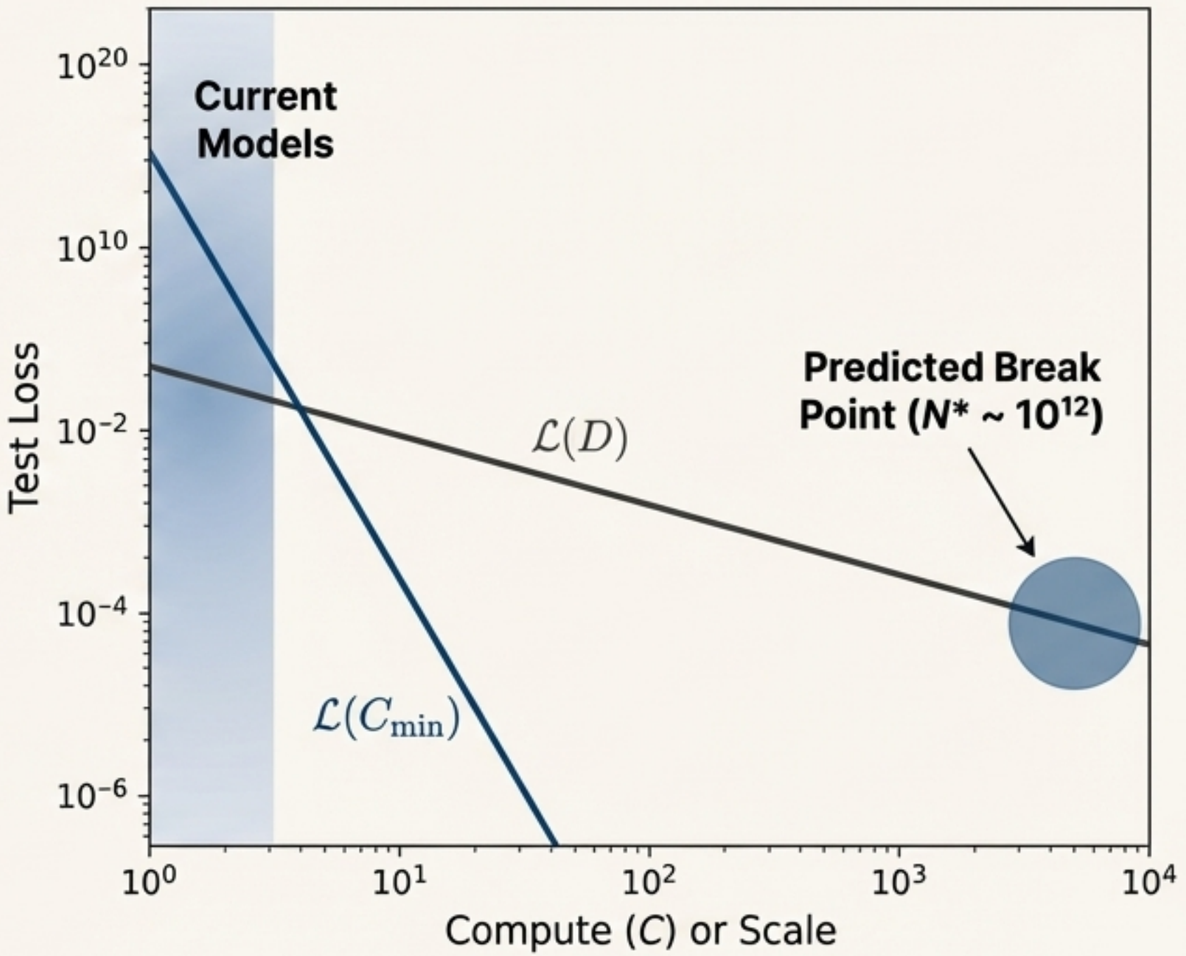
As the total compute budget (`C`) increases, it should be spent primarily on increasing model size (`N`).



The **future** of performance gains lies in dramatically larger models, not longer training runs.

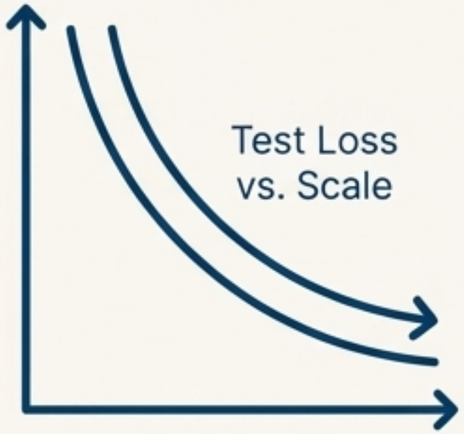
Limitations and The Road Ahead

- **Caveats:** These power laws must eventually break down. The exact point is unknown but predicted to be orders of magnitude away ($N^* \sim 10^{12}$ parameters).
- **Future Directions:** This predictive framework allows for more strategic research:
  - Pushing the limits of model parallelism to train ever-larger models.
  - Improving optimizers for better early-training dynamics.
  - Investigating if these laws apply to other domains like images, audio, or video.

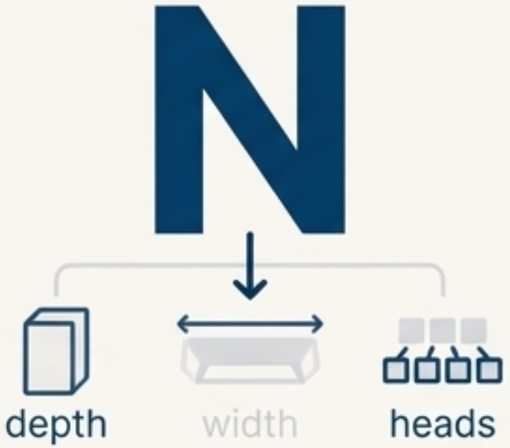


From Alchemy to a Science of Scale

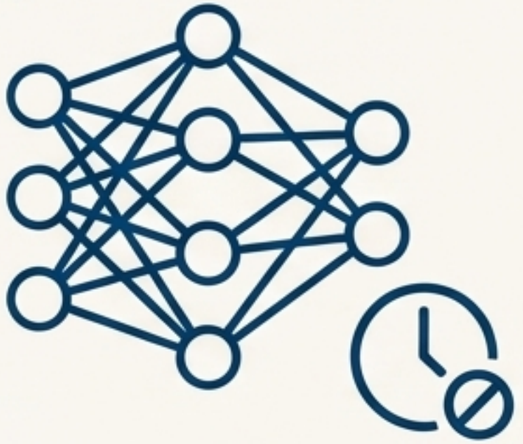
We have moved from heuristic exploration to a **predictive science** of training large language models.



1. Performance is predictable: Test loss follows smooth, simple power laws with respect to model size, dataset size, and compute.



2. Scale is the primary driver: The number of non-embedding parameters ('N') is far more important than specific architectural choices.



3. The optimal strategy is bigger, not longer: To best use a fixed compute budget, train a much larger model for fewer steps.

Q&A

Appendix: Summary of Power Laws

Key Trend Equations

Condition	Governing Equation
Limited N, large D	$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$
Limited D, large N	$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}$
Compute-Optimal	$L(C_{\min}) = \left(\frac{C_c}{C_{\min}}\right)^{\alpha_C}$
Combined N and D (Overfitting)	$L(N,D) = \left[\left(\frac{N_c}{N}\right)^{\alpha_N/\alpha_D} + \left(\frac{D_c}{D}\right)\right]^{\alpha_D}$
Finite Steps S (Infinite Data)	$L(N,S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}}\right)^{\alpha_S}$

Empirical Fitted Values

Parameter	Value
α_N	0.076
α_D	0.095
α_C	0.050
α_B	0.21
α_S	0.76