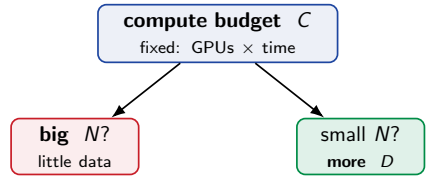
# You have 1,000 GPUs for a month. Now what?

A fixed compute budget $C$ is the real constraint: so many accelerators, for so many weeks. You get *one* training run. Two knobs to set:

- model size $N$ – how many parameters,
- data $D$ – how many training tokens.

They trade off, because $C \approx 6ND$. Spend it all on a **huge** model trained briefly? Or a **smaller** model trained on far more tokens?

Two papers, two answers, two years apart – and the second overturned how the whole field was training its models.



# Outline

## One budget, two knobs

Loss is a function of model size $N$, data $D$, and compute $C$.

# The problem, stated precisely

We treat the final pretraining loss as a function of two numbers: model size $N$ and training tokens $D$. Compute ties them together:

$$C \approx 6\,N\,D \qquad (\sim 6 \text{ FLOPs per parameter per token, forward + backward}).$$

Given a fixed budget $C$, find the allocation that minimizes loss:

$$N_{opt}(C),\ D_{opt}(C) \ = \ \underset{6ND = C}{\arg\min}\ L(N, D)$$

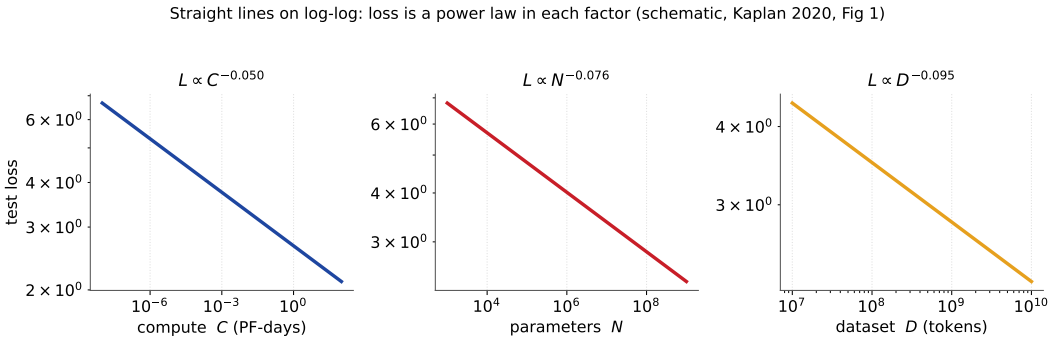
- Bigger $N$ at fixed $C \Rightarrow$ fewer tokens $D$ (train briefly).
- Bigger $D$ at fixed $C \Rightarrow$ smaller $N$ (train a small model longer).

Everything below is one question: for a given $C$, where is the sweet spot on the $N$ versus $D$ trade-off?

## Kaplan 2020: performance is predictable

Loss follows clean power laws across seven orders of magnitude.

# Loss is a power law in $N$, $D$, and $C$




Vary one factor – with the other two not bottlenecking – and the loss falls as a **straight line on a log-log plot**: a power law, holding over *seven* orders of magnitude. Model *shape* (depth vs width) barely matters; only scale does.

# The intuition: loss tracks the scarcer resource

Think of model size $N$ and data $D$ as **two ingredients**. Pouring more of the one you already have plenty of barely helps – the result is capped by whichever is **scarcer**.

- Huge model, tiny data $\Rightarrow$ extra parameters sit idle. You are **data-bottlenecked**.
- Lots of data, small model $\Rightarrow$ no room to absorb it. You are **capacity-bottlenecked**.
- Keep lowering the loss only by growing the *scarce* side – and, to keep going, **both together**.

Like a barrel of uneven staves: it holds water only up to the **shortest** one. Adding to the tall staves does nothing.

The next slide puts this in symbols – the loss is set by the *larger* (worse) of the $N$ and $D$ terms. It is also why Kaplan's "just go bigger" will turn out to under-train.

# Three power laws, one combined form

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad L(C_{\text{min}}) = \left(\frac{C_c}{C_{\text{min}}}\right)^{\alpha_C}$$

with tiny exponents $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$, $\alpha_C \approx 0.050$. Doubling $N$ cuts loss by only $2^{-0.076} \approx 5\%$ – diminishing returns, but *predictable* ones.

A single fit captures $N$ and $D$ together:

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

These fits are smooth enough to **extrapolate**: predict the loss of a model far bigger than any you have trained. That is what makes them "laws."

# Kaplan's verdict: big models, stop early

Two empirical findings drive the conclusion:

- **Bigger models are more sample-efficient** – they hit a target loss in fewer tokens and fewer optimization steps.
- So at fixed $C$, the optimum is a **very large** model trained *well short of convergence*.

Read the exponents: a $10\times$ bigger budget $\Rightarrow$ model $\times 5.5$, data only $\times 1.8$. **Almost all new compute should buy parameters, not tokens.** "Big models may be more important than big data."

The allocation exponents:

$$N_{opt} \propto C^{0.73}, \qquad D_{opt} \propto C^{0.27}$$

## Chinchilla 2022: they were under-training

Re-derive the frontier carefully, and the advice flips.

# Predict first: 10× the compute – how much bigger a model?

You get 10× more compute than last time. By how much should you grow the **model**, and by how much the **data**?

# Predict first: 10× the compute – how much bigger a model?

You get 10× more compute than last time. By how much should you grow the **model**, and by how much the **data**?

**Kaplan (2020):** model ×**5.5**, data ×1.8 – almost all into size.

# Predict first: 10× the compute – how much bigger a model?

You get 10× more compute than last time. By how much should you grow the **model**, and by how much the **data**?

**Kaplan (2020):** model ×**5.5**, data ×1.8 – almost all into size.

**Chinchilla (2022):** model ×**3.2**, data ×**3.2** – split evenly ($\sqrt{10} \approx 3.2$).

# Predict first: 10× the compute – how much bigger a model?

You get 10× more compute than last time. By how much should you grow the **model**, and by how much the **data**?

**Kaplan (2020):** model ×**5.5**, data ×1.8 – almost all into size.

**Chinchilla (2022):** model ×**3.2**, data ×**3.2** – split evenly ($\sqrt{10} \approx 3.2$).

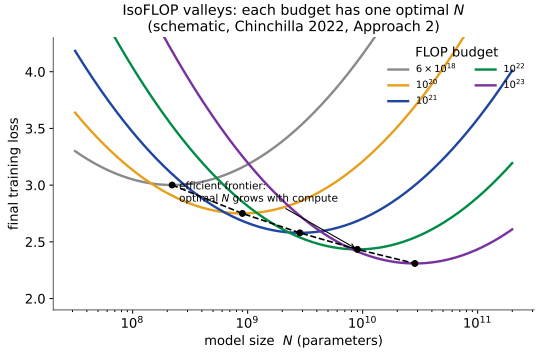
Same 10×, a very different model. Kaplan's rule builds models too big for the data they see. Chinchilla says: grow both together.

# Re-deriving the frontier three ways

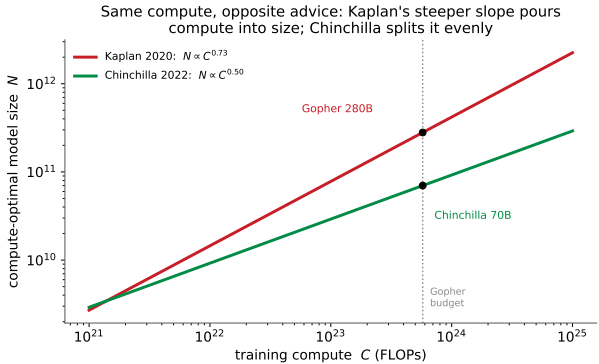
Hoffmann et al. train **400+** models (70M–16B params, 5B–500B tokens) and estimate $N_{opt}(C), D_{opt}(C)$ *three independent ways*:

1. minimum over training curves,
2. **IsoFLOP** profiles (vary $N$ at fixed $C$),
3. a parametric fit $L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$.

Why Kaplan differed: he used *one* fixed learning-rate schedule and mostly small (<100M) models – both bias the estimate toward *oversized* models.



# Scale $N$ and $D$ equally, not $N$ alone



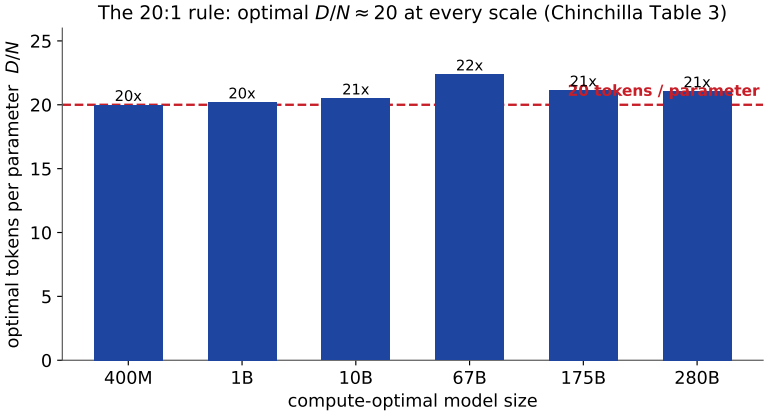

All three approaches agree:

$$N_{opt} \propto C^{\approx 0.50}, \quad D_{opt} \propto C^{\approx 0.50}$$

versus Kaplan's 0.73/0.27.

**Equal scaling:** every doubling of model size should come with a doubling of training tokens. The two lines diverge fast – at the Gopher budget, Kaplan says 280B, Chinchilla says ∼70B.

# The rule of thumb: ∼20 tokens per parameter




Turn the frontier into a number: across every scale, compute-optimal training wants $D/N \approx 20$. A 1B model wants ∼20B tokens; a 70B model wants ∼1.4T.

# The proof: Chinchilla vs Gopher

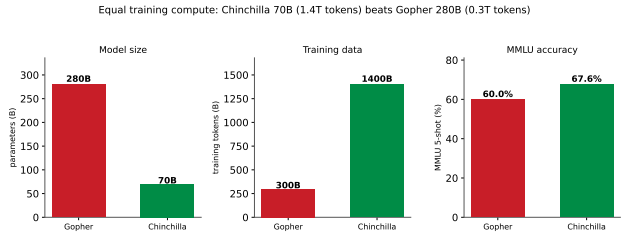
Same compute, opposite bets – and the smaller model wins.

# The clean experiment: equal compute, opposite bets

Same FLOP budget, two designs:

| | Gopher | Chinchilla |
|---|---|---|
| Params | 280B | **70B** |
| Tokens | 300B | **1.4T** |
| Compute | *equal* | |

Chinchilla is 4× smaller, trained on ∼4× more data – exactly what the frontier predicts.



# Chinchilla wins almost everywhere

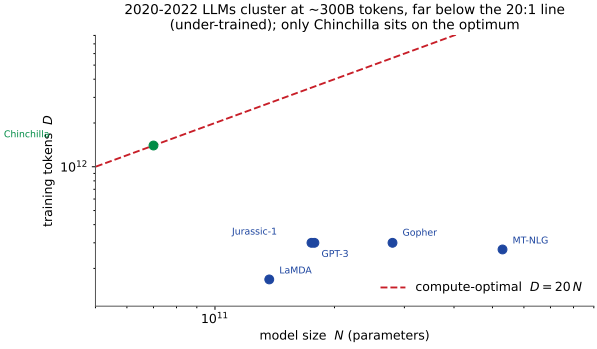
- **MMLU:** 67.6% vs Gopher's 60.0% (+7.6%) – beating expert forecasters' own June-2023 prediction.
- Uniformly better on The Pile, BIG-bench (+10.7%), common-sense, and question-answering benchmarks.
- RACE-h +10.7, TruthfulQA +14.1 (0-shot) – large jumps from better use of the same compute.

And because it is 4× smaller, Chinchilla is far **cheaper to fine-tune and to serve** – the win compounds at inference time.

## What it means for everyone else

Most models of the era were simply the wrong shape.

# The whole field had been under-training




Following Kaplan and GPT-3, everyone trained to ∼300B tokens *regardless* of model size – GPT-3, Jurassic-1, Gopher, and MT-NLG all cluster there.

Against the 20:1 line they are all **over-sized and under-trained**: compute went into parameters that more tokens would have used better.

# Beyond compute-optimal: the inference argument

Chinchilla optimizes *training* compute. But a deployed model is trained *once* and served *billions* of times – and inference cost scales with $N$, not $D$.

So it pays to push **even past** 20:1: take a *smaller* model and train it on *far more* tokens than compute-optimal. You spend more at training to save forever at inference.

This is the **Llama philosophy**: Llama's 7B was trained on $\sim$1T tokens ($\sim$140:1), well beyond Chinchilla-optimal, precisely because inference – not training – dominates the real-world cost.

# Recap

- Loss is a smooth **power law** in model size $N$, data $D$, and compute $C$ (Kaplan 2020) – so allocation is an optimization: $\arg\min_{6ND=C} L(N, D)$.
- **Kaplan:** spend new compute mostly on *size* – $N \propto C^{0.73}$, $D \propto C^{0.27}$.
- **Chinchilla:** that under-trains. Scale $N$ and $D$ **equally** ($\approx C^{0.5}$ each) – about **20 tokens per parameter**.
- **Proof:** Chinchilla 70B / 1.4T beats Gopher 280B / 300B at *equal compute* (MMLU +7.6%).
- Most 2020–2022 LLMs were oversized and under-trained; inference cost pushes the optimum even *smaller and longer* (Llama).

**The one line:** for a fixed budget, do not just go bigger – balance parameters and tokens, then bias toward more tokens if you have to serve the model.