Scaling Laws

Kaplan · Chinchilla · Emergent Abilities · Inference-Time Scaling

What are scaling laws?

Discovery: language model performance follows **smooth power-law relationships** as you scale up three key variables — predictably, over many orders of magnitude.

Model Size N

Number of parameters (non-embedding)

117M $\rightarrow$ 175B $\rightarrow$ 1T+

Data Size D

Number of training tokens

1B $\rightarrow$ 300B $\rightarrow$ 15T

Compute C

Total FLOPs spent on training

$10^{18} \rightarrow 10^{24}$ FLOPs

Power law: $L = \left(\frac{K}{X}\right)^{\alpha}$ where $L =$ loss, $X \in \{N, D, C\}$

On a log-log plot, this is a straight line: $\log L = -\alpha \log X + \text{const}$

Why this matters: you can **predict** the performance of a 100B model by extrapolating from smaller experiments. This saves millions of dollars.

Kaplan et al. (2020), "Scaling Laws for Neural Language Models" — OpenAI

Kaplan et al. (2020): the power-law relationships

Model size:

$$L(N) = \left(\frac{N_c}{N}\right)^{0.076}$$

$N_c = 8.8 \times 10^{13}$

$10\times$ params $\rightarrow$ 19% less loss

Data size:

$$L(D) = \left(\frac{D_c}{D}\right)^{0.095}$$

$D_c = 5.4 \times 10^{13}$

$10\times$ data $\rightarrow$ 24% less loss

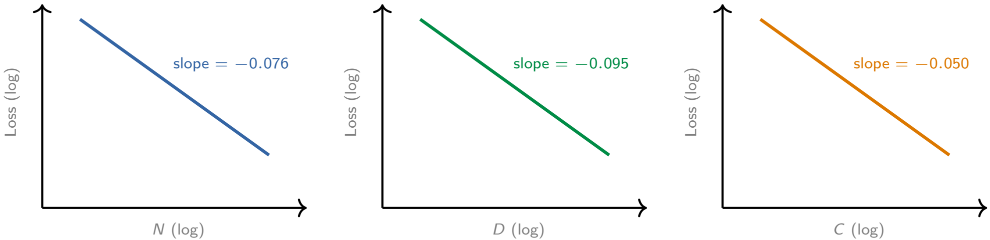
Compute:

$$L(C) = \left(\frac{C_c}{C}\right)^{0.050}$$

$C_c = 3.1 \times 10^{8}$ PF-days

$10\times$ compute $\rightarrow$ 12% less loss

On a log-log plot, each relationship is a straight line:



Kaplan: how to spend your compute budget

Given a fixed compute budget C, how should you split it between N and D?
Since $C \approx 6ND$, making N bigger means less data (fewer passes), and vice versa.

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

"Train big models on relatively little data"

10$\times$ more compute:

Increase model size by $\sim$5.4$\times$
Increase data by only $\sim$1.9$\times$

$\approx$ 1.7 tokens per parameter

Key insight:

Larger models are more **sample-efficient** — they reach lower loss with fewer tokens

So: prioritize model size!

GPT-3 followed Kaplan: 175B parameters trained on only 300B tokens $\approx$ 1.7 tokens per parameter. **This turned out to be sub-optimal.**

Problem: Kaplan counted only non-embedding parameters, biasing results at smaller scales

The compute approximation: $C \approx 6ND$

$$C \quad \approx \quad 6 \cdot N \cdot D$$

Total FLOPs $\approx$ 6 $\times$ parameters $\times$ training tokens

Where d … ne from?

$\times$ 2

Each parameter involves a multiply + accumulate = 2 FLOPs per param per token (forward pass)

$\times$ 3

Backward pass costs $\approx 2\times$ the forward pass
Forward: $1\times$,
Backward: $2\times$
Total: $3\times$ forward

$= \quad 6$

2 (FLOPs/param/token) $\times$ 3 (fwd + bwd) = 6 FLOPs per parameter per token

Example — GPT-3: $N = 175$B, $D = 300$B tokens

$C \approx 6 \times 175 \times 10^9 \times 300 \times 10^9 = 3.15 \times 10^{23}$ FLOPs $\approx$ 3,640 PetaFLOP-days

The fundamental trade-off: for fixed C, increasing N means decreasing D (and vice versa). Scaling laws tell us the **optimal split** that minimizes loss.

Chinchilla (2022): Kaplan was wrong

Hoffmann et al. (DeepMind): trained
400+ models from 70M to 16B params.
Found that Kaplan's recommendation to prioritize model size was **sub-optimal**.

Three independent approaches, same conclusion:

Approach 1

Fix model sizes,
vary number of tokens.
Find minimum loss
for each compute level.

Approach 2

IsoFLOP profiles:
fix compute budget,
vary model size.
Find optimal N per C.

Approach 3

Fit parametric model:
$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$
Minimize analytically.

$$N_{\text{opt}} \propto C^{0.50} \qquad D_{\text{opt}} \propto C^{0.50}$$

"Scale model size and data equally"

$$D_{\text{opt}} \approx 20 \times N$$

$\approx$ 20 tokens per parameter

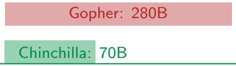
Hoffmann et al. (2022), "Training Compute-Optimal Language Models" — NeurIPS 2022

Chinchilla vs. Gopher: the empirical proof

Same compute budget, radically different allocation

	Gopher	**Chinchilla**
Parameters	280B	**70B**
Training tokens	300B	**1.4T**
Tokens / parameter	$\sim$1.07	$\sim$20
Training compute	5.76×10^{23} FLOPs	5.76×10^{23} FLOPs
MMLU accuracy	60.0%	**67.5%**

Parameters:



Training data:

A 4× smaller model trained on 4× more data, using the same compute, uniformly outperformed Gopher, GPT-3 (175B), Jurassic-1 (178B), and MT-NLG (530B)

Inference cost is also 4× cheaper (4× fewer parameters to serve)

Kaplan vs. Chinchilla: the key differences

	Kaplan (2020)	Chinchilla (2022)
N_{opt} scaling	$C^{0.73}$	$C^{0.50}$
D_{opt} scaling	$C^{0.27}$	$C^{0.50}$
Tokens per parameter	$\sim$1.7	$\sim$20
Philosophy	"Train big, stop early"	"Scale N and D equally"
Parameter counting	Non-embedding only	All parameters
Experimental scale	Smaller models	Up to 16B (validated at 70B)

Why do they disagree?

Parameter counting

Kaplan excluded embedding params, biasing results at smaller scales

Experimental scale

Kaplan used smaller models. Chinchilla trained 400+ models up to 16B

LR schedule

Different hyperparameter tuning approaches led to different optima

Consensus today: Chinchilla scaling ($D \approx 20N$) is the accepted baseline. But in practice, labs now **over-train** beyond even Chinchilla-optimal (see later).

The Chinchilla loss model

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

$E = 1.69$

Irreducible loss
(entropy of natural

$\frac{A}{N^{\alpha}}$

$A = 406.4$
$\alpha = 0.34$

$\frac{B}{D^{\beta}}$

$B = 410.7$
$\beta = 0.28$

Optimize:

Given $C = 6ND$,
minimize L

Interpretation: loss has three components — an irreducible floor E,
a penalty for too few parameters (A/N^{α}),
and a penalty for too little data (B/D^{β}).
The optimal model balances both penalties equally.

Result: minimizing L subject to $C = 6ND$ gives

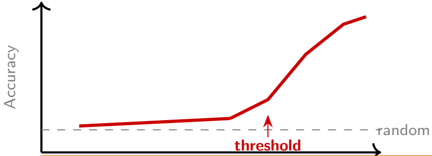
$$N_{\text{opt}} = G \cdot \left(\frac{C}{6}\right)^{a}, \quad D_{\text{opt}} = G^{-1} \cdot \left(\frac{C}{6}\right)^{b} \quad \text{where } a \approx b \approx 0.5$$

The near-equal exponents ($a \approx b$) mean N and D should scale at the same rate

Emergent abilities (Wei et al., 2022)

An ability is emergent if it is **absent in smaller models** but **appears in larger ones** — it cannot be predicted by extrapolating from smaller-scale performance.

The phase transition pattern:



Ability	Threshold
Multi-digit arithmetic	∼100B params
Word unscrambling	∼100B params
Chain-of-thought	∼100B params
Instruction following	∼100B+
College exams (MMLU)	∼100B+

Wei et al. documented **137 emergent abilities** across BIG-Bench and other benchmarks. Near-random performance until a critical scale, then a sharp jump.

But is this real? Are these genuine phase transitions in capability, or an artifact of how we measure performance?

Wei et al. (2022), "Emergent Abilities of Large Language Models" — TMLR

Are emergent abilities a mirage?

Schaeffer et al. (2023): emergent abilities are an artifact of **metric choice**, not fundamental changes in model behavior. NeurIPS 2023 Oral.

Exact-match accuracy

Discontinuous metric:
either 100% correct or 0%

"$2 + 3 = 5$" $\rightarrow$ correct
"$2 + 3 = 6$" $\rightarrow$ wrong

Shows sudden emergence

$\longrightarrow$ **same model!**

Token-level log-likelihood

Continuous metric:
measures partial knowledge

Probability of correct token
increases *smoothly* with scale

Shows gradual improvement

Key finding: 92%+ of BIG-Bench "emergent" abilities use nonlinear metrics. Switch to continuous metrics $\rightarrow$ smooth scaling. No phase transition.

Current resolution: capability improves **smoothly** (continuous metrics), but practical task success can appear **sudden** (discontinuous metrics). Both perspectives are "correct" — it depends on what you measure.

Schaeffer et al. (2023), "Are Emergent Abilities of Large Language Models a Mirage?" — NeurIPS 2023

Scaling laws beyond language

Power-law scaling is not language-specific. The same functional form $L = (K/X)^{\alpha}$ holds across domains with remarkably similar exponents.

Language
GPT, LLaMA
Text generation
$\alpha_N \approx 0.08$

Vision
ViT models
Image classification
Zhai et al. 2022

Video
Video generation
Autoregressive
Henighan 2020

Multimodal
Image + text
CLIP, etc.
Shukor 2025

A new scaling axis: inference-time compute

Snell et al. (2024): instead of making models bigger, spend more compute at **inference time** — search over reasoning paths, verify with reward models.

A smaller model + test-time compute can outperform a **14$\times$ larger** model.

OpenAI o1 (2024): AIME math competition 15.6% $\rightarrow$ **83.3%** by scaling inference compute. o1 "thinks longer" on harder problems (chain-of-thought at test time).

Three axes of scaling: pre-training compute (Kaplan/Chinchilla), data quality, inference compute

The over-training trend

Chinchilla optimizes training efficiency.
But in deployment, you pay per inference.
A **smaller** model trained on **more** data gives similar quality at lower serving cost.

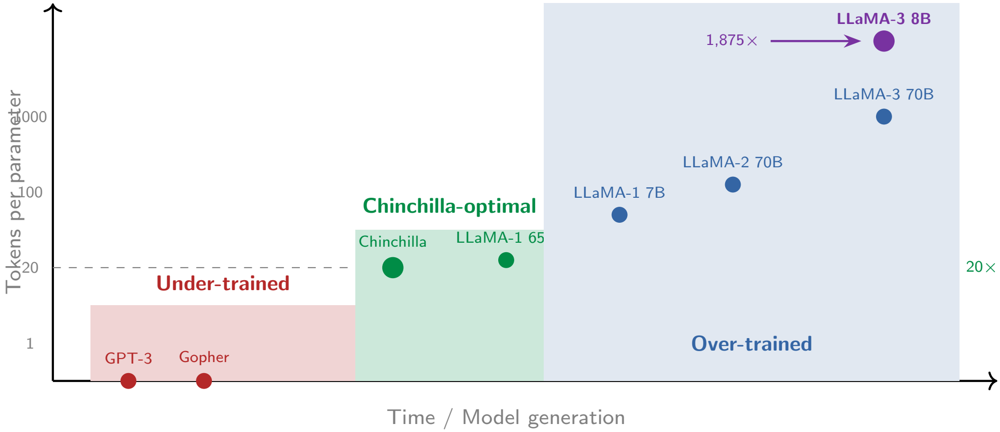
Model	**Params**	**Tokens**	**Tok/Param**	**vs. Chinchilla (20:1)**
GPT-3 (2020)	175B	300B	1.7	12× under-trained
Gopher (2021)	280B	300B	1.1	19× under-trained
Chinchilla (2022)	70B	1.4T	20	1× optimal
LLaMA-1 7B (2023)	7B	1.0T	143	7× over-trained
LLaMA-1 65B (2023)	65B	1.4T	22	$\sim$1× near-optimal
LLaMA-2 70B (2023)	70B	2.0T	29	1.4× over-trained
LLaMA-3 8B (2024)	8B	15T	1,875	94× extreme over-train

Key insight: LLaMA-1 7B (trained on 1T tokens) was competitive with **GPT-3 175B** on many benchmarks. 25× smaller → 25× cheaper to serve!

The principle: training is a one-time cost.
Inference is paid **every request, forever.**
Over-training = spend more on training to save on inference at scale.

Loss keeps improving well past Chinchilla-optimal, just with diminishing returns

The three eras of compute allocation



The scaling playbook

1. Performance scales as a power law with model size, data, and compute.
$L(N) \propto N^{-0.076}$, $L(D) \propto D^{-0.095}$, $L(C) \propto C^{-0.050}$ (Kaplan 2020)

2. Scale N and D equally for compute-optimal training ($D \approx 20N$).
Chinchilla 70B beat Gopher 280B with the same compute budget (Hoffmann 2022)

3. In practice, over-train smaller models for cheaper inference.
LLaMA-3 8B: 15T tokens (1,875 tok/param). Training cost is one-time; inference is forever.

4. New capabilities emerge at scale (whether real or metric-dependent).
137 emergent abilities documented (Wei 2022). Debate ongoing (Schaeffer 2023).

5. Inference-time compute is the new frontier.
Smaller model + more thinking at test time can beat 14× larger model (Snell 2024).

Further reading

Scaling Laws

- Kaplan et al. (2020), "Scaling Laws for Neural Language Models" (OpenAI)
- Hoffmann et al. (2022), "Training Compute-Optimal Large Language Models" (Chinchilla)
- Muennighoff et al. (2024), "Scaling Data-Constrained Language Models"

Emergent Abilities & Scaling

- Wei et al. (2022), "Emergent Abilities of Large Language Models"
- Schaeffer et al. (2023), "Are Emergent Abilities of LLMs a Mirage?"

Inference-Time Scaling

- Snell et al. (2024), "Scaling LLM Test-Time Compute Optimally"
- Brown et al. (2024), "Large Language Monkeys: Scaling Inference Compute with Repeated Sampling"

Questions?

Next: Hallucination & Grounding