CSCI 5942: AI ENGINEERING, LECTURE 1

Scaling Laws

Christoffer Heckman
Department of Computer Science, University of Colorado Boulder

August 25, 2026

Quick overview

Power laws. How predictably does loss fall as model, data, and compute grow?

Kaplan 2020; straight lines across seven orders of magnitude

Chinchilla. Given a fixed compute budget, how big a model on how much data?

Hoffmann 2022; the 20-tokens-per-parameter rule, and when to break it

Codesign. How does a predictable loss curve set cluster size and data spend?

clusters, marginal dollars, and one currency: FLOPs

A live fit. Five Shakespeare models, one law, fitted in class.

trained on the Assignment 1 codebase; the last two runs fall off the line

Assignment 1

Transformers on Shakespeare, from scratch. Out Thursday.

- build it: attention, block, model; ~150 lines, tests included
- train it, sample some Shakespeare
- train a ladder of five sizes, fit a law, explain what you see

Every plot in today’s lecture marked "Assignment 1 codebase" came from it.

Submission on Gradescope; due date on the course page. Keep your trained checkpoints: the evaluation assignment in week 3 reuses them.

The Codebase

```python
class CausalSelfAttention(nn.Module):
    """Multi-head causal self-attention.

    Input  x: (B, T, C)  batch, time
    Output y: (B, T, C)
    """
    def forward(self, x):
        B, T, C = x.shape
        # --- YOUR IMPLEMENTATION HERE
        raise NotImplementedError


class MLP(nn.Module):
    """Expand 4x, GELU, project back."""
    def forward(self, x):
        # --- YOUR IMPLEMENTATION HERE ---
        raise NotImplementedError
```

B — batch. Independent sequences pushed through at once.

T — time. Position in the sequence, so the context length. The causal mask acts on this axis. 256 for A1.

C — channels. The width of the vector every position carries. 384 in the A1 base config. Also `d_model`, `n_embd`, `hidden_size`.

Still a representation, **not** a distribution over tokens. `lm_head` projects C → vocab at the very end, giving `(B, T, vocab)`.

```
data/prepare.py
model.py      ← the blanks are yours
train.py
sample.py
configs/      five sizes
tests/        four checks
  shapes · causality
  init · overfit-one-batch
```

PyTorch and numpy only. The causality test catches the two classic mask bugs before you train: attending to future tokens, and an off-by-one on the mask.

Power Laws

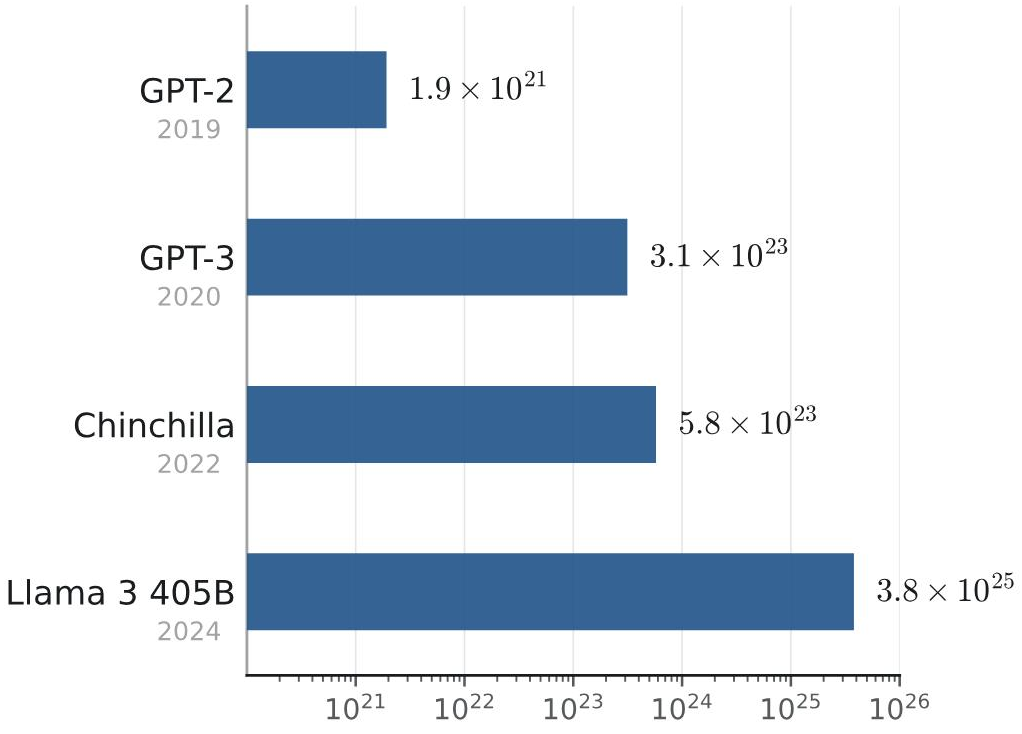
Five Years of Scale

model	year	parameters	training compute
GPT-2	2019	1.5B	$\sim 4 \times 10^{21}$
GPT-3	2020	175B	3.1×10^{23}
Chinchilla	2022	70B	5.8×10^{23}
Llama 3 405B	2024	405B	$\sim 3.8 \times 10^{25}$

Four orders of magnitude in five years. What does that physically mean?

Radford et al. (2019); Brown et al., NeurIPS 2020; Hoffmann et al., arXiv:2203.15556 (2022); Grattafiori et al., arXiv:2407.21783 (2024). Compute figures: Epoch AI, Notable AI Models (2026).

Five Years of Scale



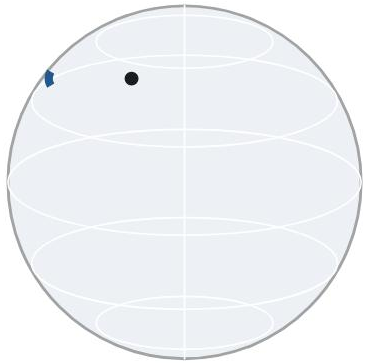
	energy (MWh)	carbon (tCO_2e)	the same carbon as flying a 737 this far, or as this much of six-a-day DEN-SFO service
GPT-2	7.9 (est.)	3.4 (est.)	293 km under a fifth of one flight
GPT-3	1,287	552	47,634 km 5 days
Chinchilla	1,152 (est.)	464 (est.)	40,033 km 4 days
Llama 3 405B	21,588 (est.)	8,930	770,461 km 12 weeks

estimates; sources on the citation line

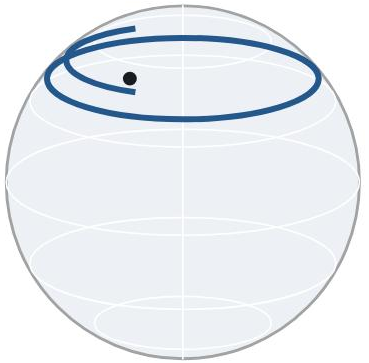
Compute: Epoch AI, Notable AI Models (2026); Brown et al., NeurIPS 2020, Table D.1; Hoffmann et al., arXiv:2203.15556; Grattafiori et al., arXiv:2407.21783. Energy and carbon: Patterson et al., arXiv:2104.10350, Table 4; Rae et al., arXiv:2112.11446, Appendix F; Meta, Llama 3.1 405B model card (2024). Flights: EMEP/EEA Guidebook 2023, type B738, whole aircraft DEN to SFO, combustion CO2 only. GPT-2 energy is our estimate scaled from GPT-3; Chinchilla energy is our estimate anchored on Gopher, which shares its compute budget.

Nineteen Times Around

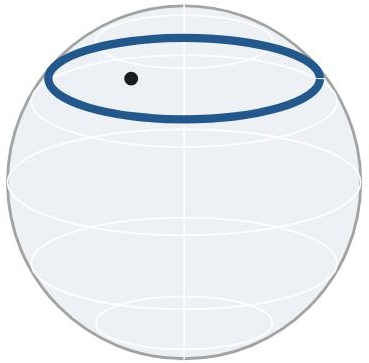
the same carbon, as distance flown by one 737



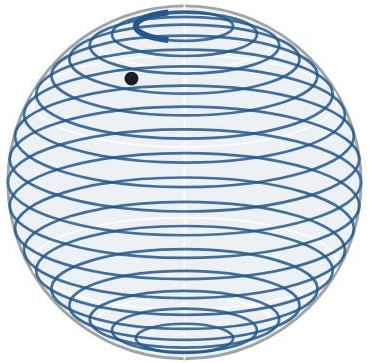
GPT-2
293 km
Denver to
Grand Junction



GPT-3
47,634 km
once around,
and a fifth



Chinchilla
40,033 km
once around
the Earth



Llama 3 405B
770,461 km
19 times
around

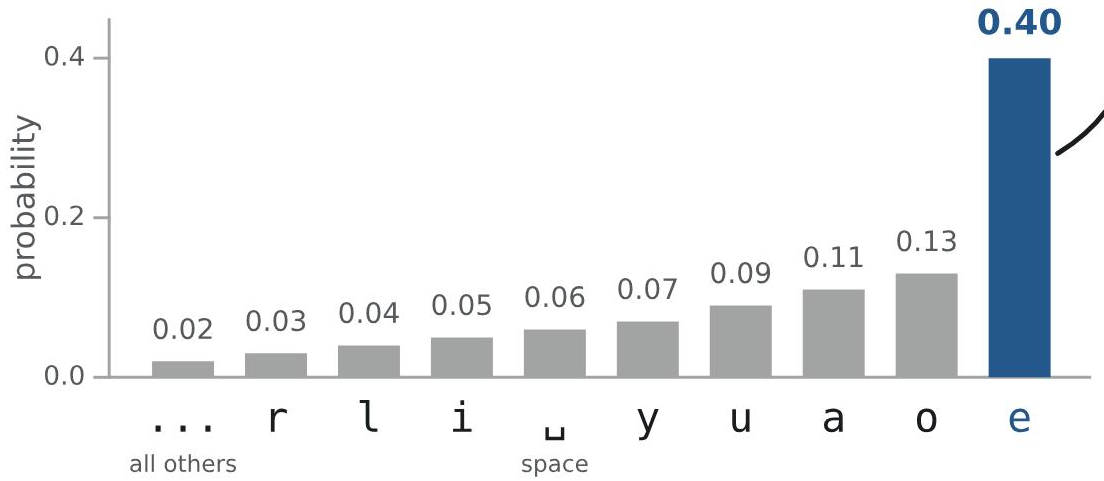
Carbon as in the previous slide. Distance at 1,553 km per 18 tCO2e, one whole Boeing 737-800, combustion CO2 only: EMEP/EEA Guidebook 2023, type B738. Earth circumference 40,075 km. Schedule equivalent on the previous slide: six departures a day at 18 tCO2e each, 108 tCO2e a day.

Loss

`to be, or not to b` **`e`** the character that actually came next

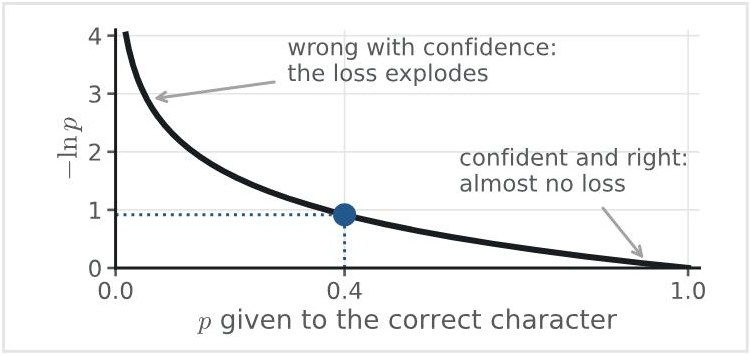
the model's only job: predict the next character

it answers with a probability for every character



the loss at this one position:

$$-\ln p(\texttt{e}) = -\ln 0.40 = 0.92$$



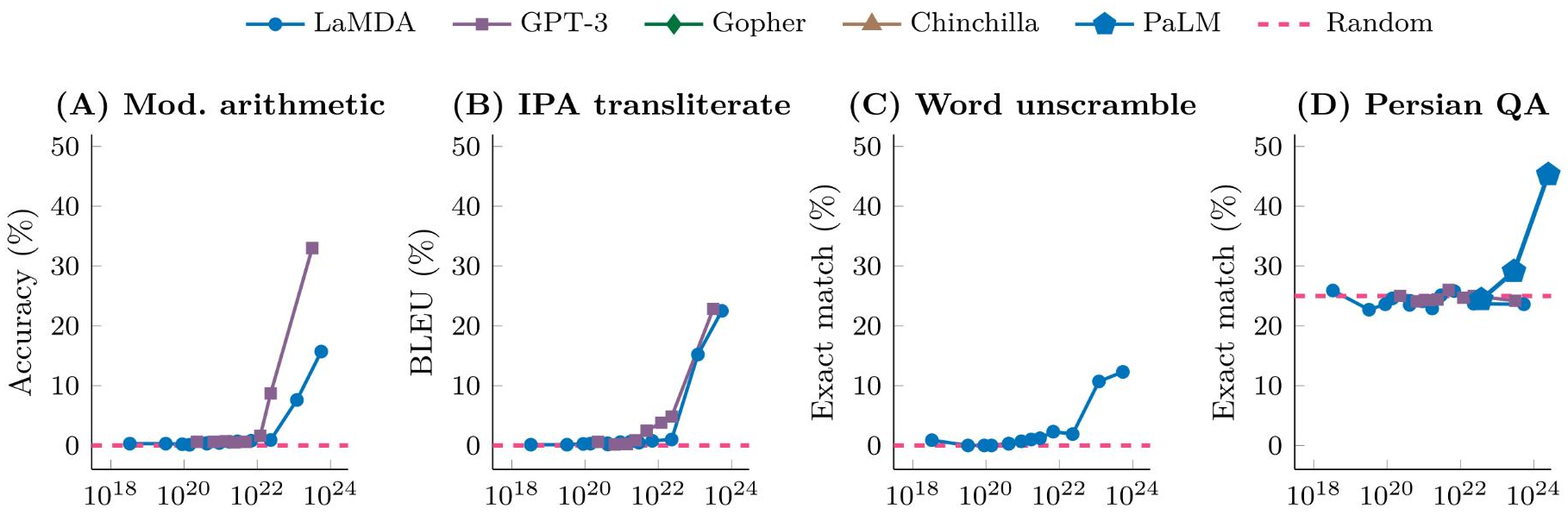
`to be, or not to b`**`e`**`, that is the question: whether 'tis nobler in the mind to suffer the`

$$L = \frac{1}{M}\sum_{i=1}^{M} -\ln p(c_i)$$

one number for the whole corpus; today's y-axis

c_i is the character that actually came next; M counts every character in the corpus; units: nats per character

Emergence, With Caveats



Much of the cliff is the scoring rule. Exact match needs every token right, so steady gains arrive as a jump; score the same answers by edit distance or Brier and the curve comes back.

And 10^{22} is not a threshold. Keep the metric, put pretraining loss on the x-axis, and the cliff sits at a loss the model has to reach. Different sizes reach it at different FLOPs.

Wei et al., "Emergent Abilities of Large Language Models," TMLR 2022, Figure 2 (first four panels). Schaeffer, Miranda, and Koyejo, "Are Emergent Abilities of Large Language Models a Mirage?" NeurIPS 2023. Du et al., "Understanding Emergent Abilities of Language Models from the Loss Perspective," arXiv:2403.15796 (2024).

Power Laws

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

$\alpha_N \approx 0.076$

parameters

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

$\alpha_D \approx 0.095$

data

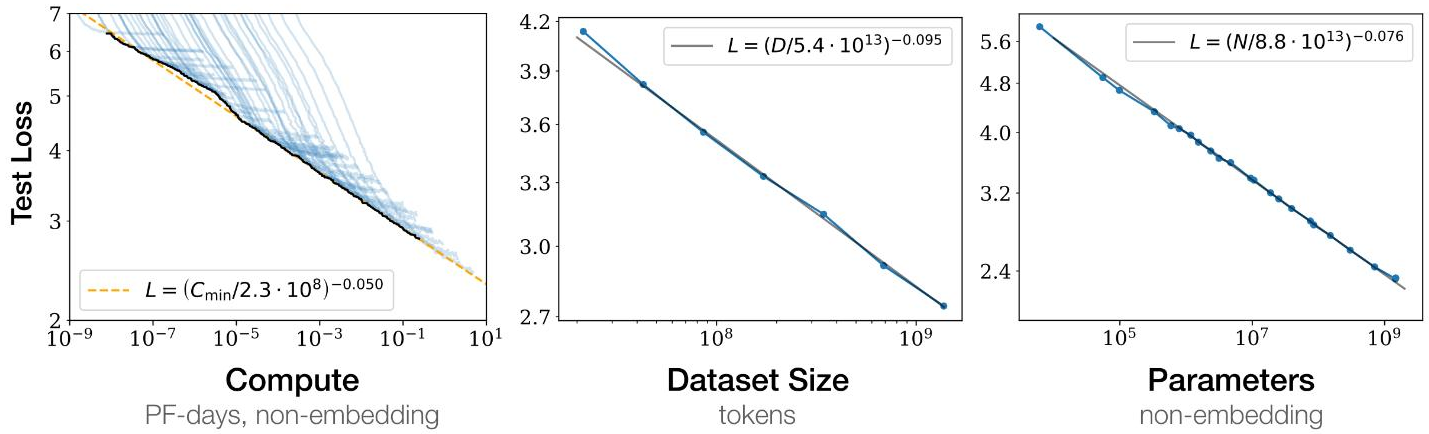
$$L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

$\alpha_C \approx 0.050$

compute

L cross-entropy loss, nats per token · N non-embedding parameters · D dataset size in tokens · C compute in PF-days

N_c, D_c, C_c fitted normalizers: 8.8×10^{13}, 5.4×10^{13}, 2.3×10^{8} · α the slope on log-log axes



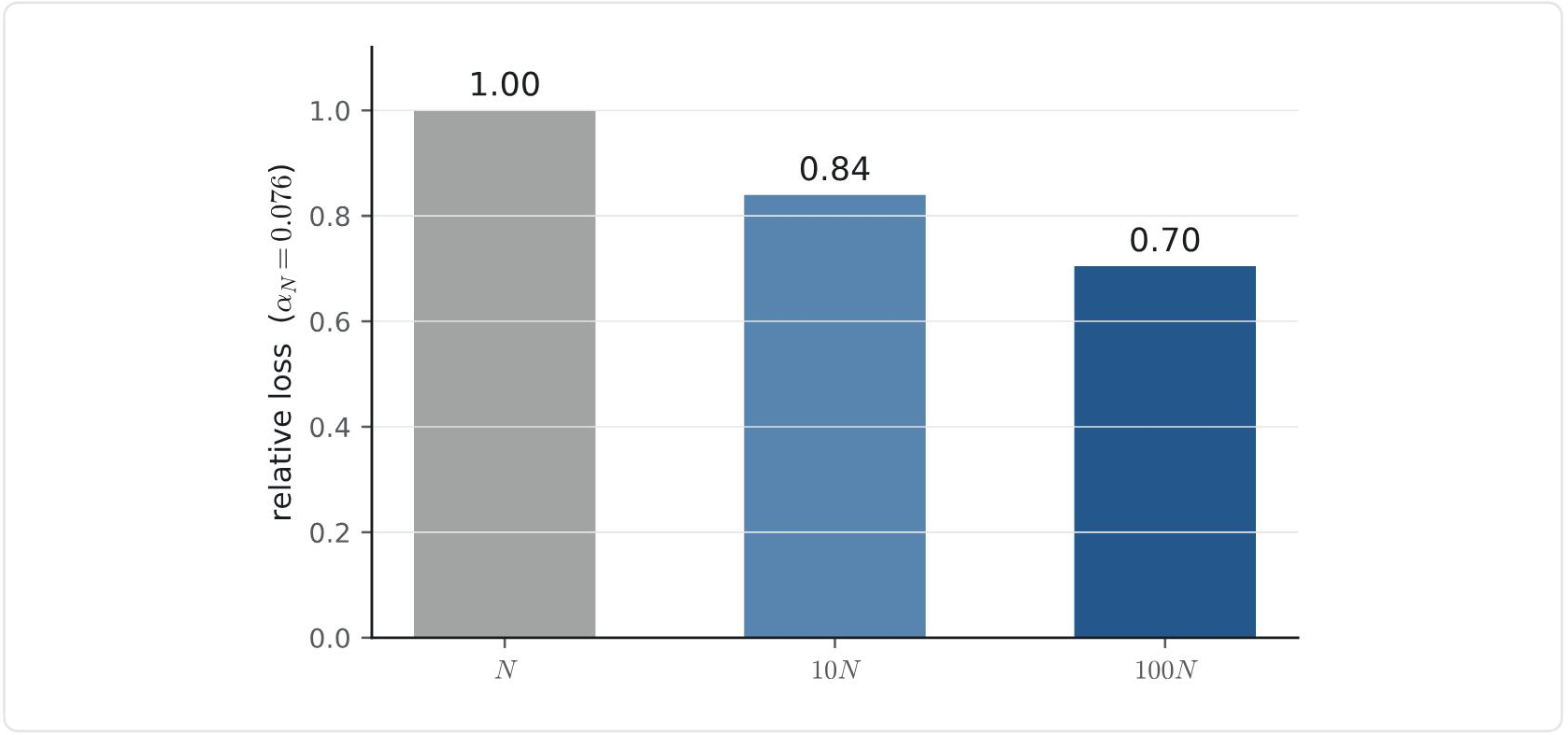
each law holds only while the other two are not the bottleneck

Kaplan et al., "Scaling Laws for Neural Language Models," arXiv:2001.08361 (2020), Figure 1.

Reading an Exponent

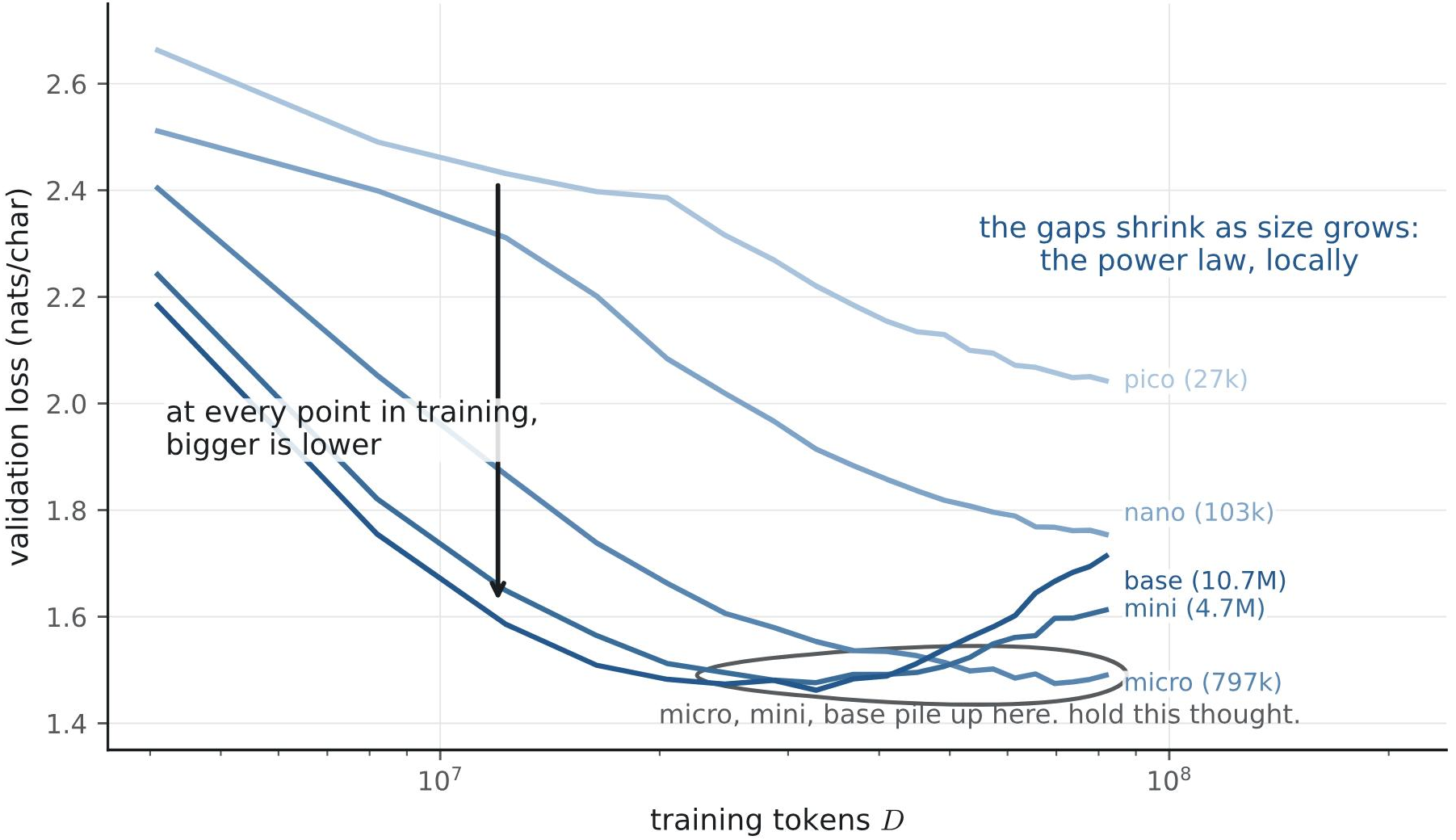
$10\times$ the parameters:

$$\frac{L(10N)}{L(N)} = 10^{-0.076} \approx 0.84$$



Two orders of magnitude take about a third off the loss.

Power Laws, Locally



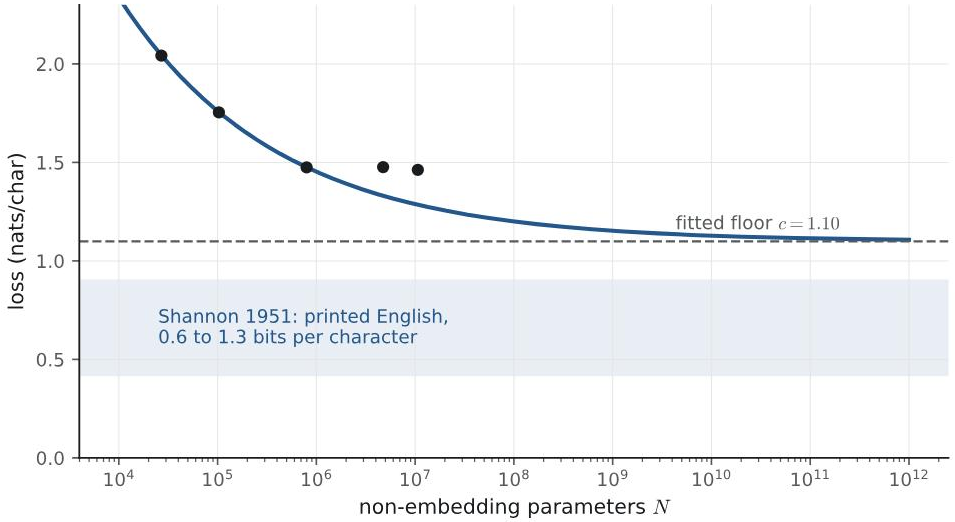
Five transformers, 27k to 10.6M non-embedding parameters, 82M characters of Shakespeare. Assignment 1 codebase.

The Floor

$$L(N) = c + a\,N^{-\alpha}$$

c is the entropy the model cannot remove.

Mid-word, the next character is forced. At a word boundary the author had a real choice, and no model recovers what was never determined.

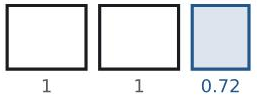


$-\log_2 p$ gives bits

$-\ln p$ gives nats, the log at base e

1 nat $= 1/\ln 2 = 1.443$ bits

$e \approx 2.72$ equally likely choices:



each one has $p = 1/e$,

so $-\ln p = 1$ nat, exactly

our floor: 1.10 nats/char $=$ 1.59 bits/char

$1.10 \times 1.443 = 1.59$

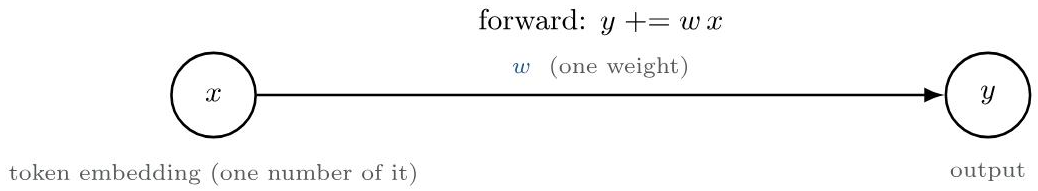
Not the architecture: c survives $N \to \infty$. Not English alone: 1.59 bits against Shannon's 0.6 to 1.3. That gap is our setup.

Shannon, "Prediction and Entropy of Printed English," Bell System Technical Journal 30:1 (1951). Henighan et al., "Scaling Laws for Autoregressive Generative Modeling," arXiv:2010.14701 (2020).

2 CHINCHILLA

Chinchilla

Six FLOPs per Token



One weight w, one token, forward:

$$y \mathrel{+}= w\,x \qquad \Rightarrow \qquad 2\ \text{FLOPs (multiply, add)}$$

Backward, two gradients per weight:

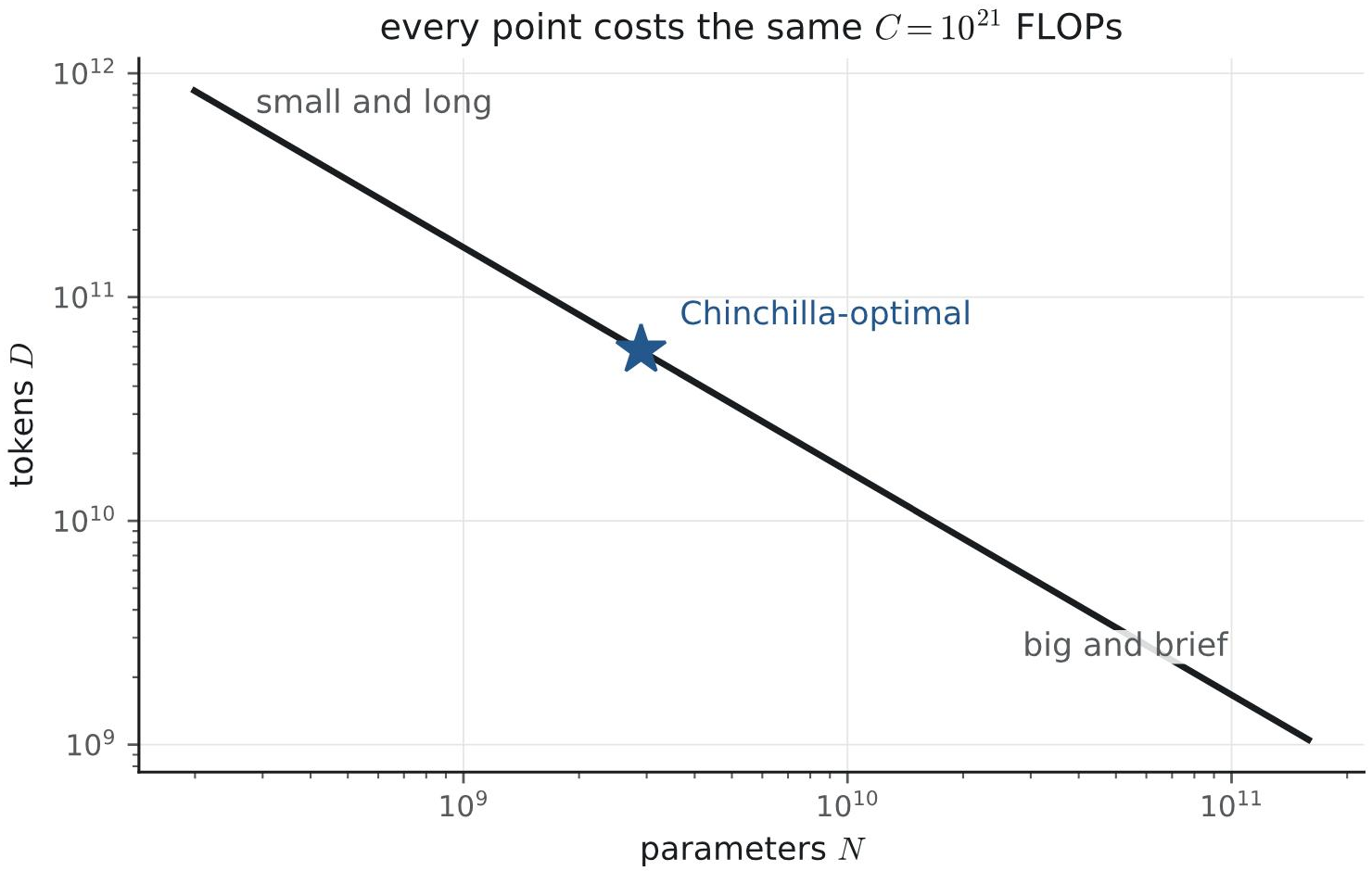
$$\frac{\partial L}{\partial x} = w \cdot \frac{\partial L}{\partial y} \qquad \frac{\partial L}{\partial w} = x \cdot \frac{\partial L}{\partial y} \qquad \Rightarrow \qquad 4\ \text{FLOPs}$$

$$C \approx (2+4)\,N\,D = 6\,N\,D$$

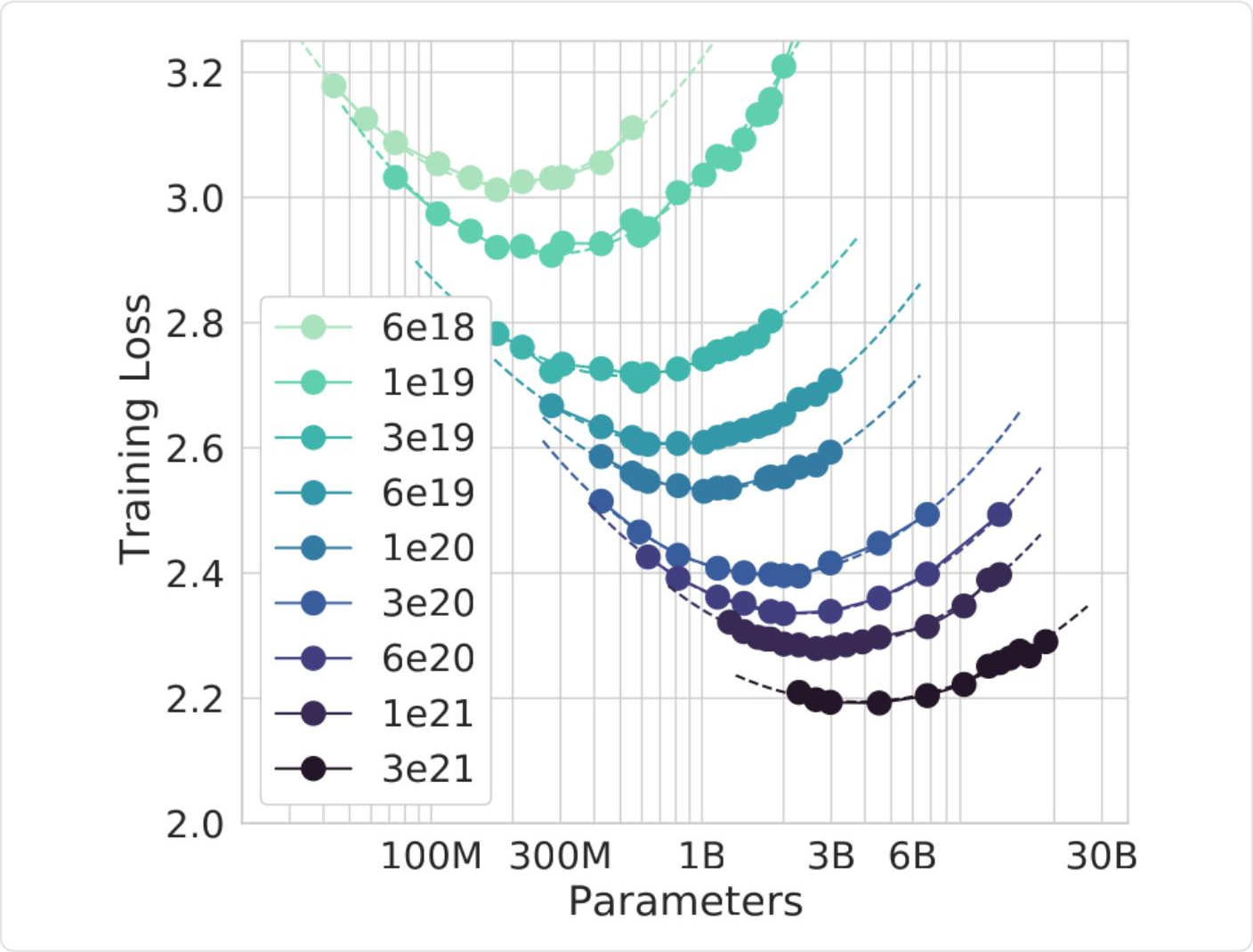
attention adds $O(T)$ FLOPs per token per layer; small until context gets long

Allocation

$$C \approx 6\,N\,D$$



IsoFLOP Profiles



Hoffmann et al., "Training Compute-Optimal Large Language Models," arXiv:2203.15556 (2022), Figure 3.

The Parametric Fit

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

$E = 1.69$	$A = 406.4$	$B = 410.7$	$\alpha = 0.34$	$\beta = 0.28$
the floor	size term	data term	size exponent	data exponent

Minimize subject to $C = 6ND$:

$$N^*(C) \propto C^{\frac{\beta}{\alpha+\beta}} = C^{0.454} \qquad D^*(C) \propto C^{\frac{\alpha}{\alpha+\beta}} = C^{0.546}$$

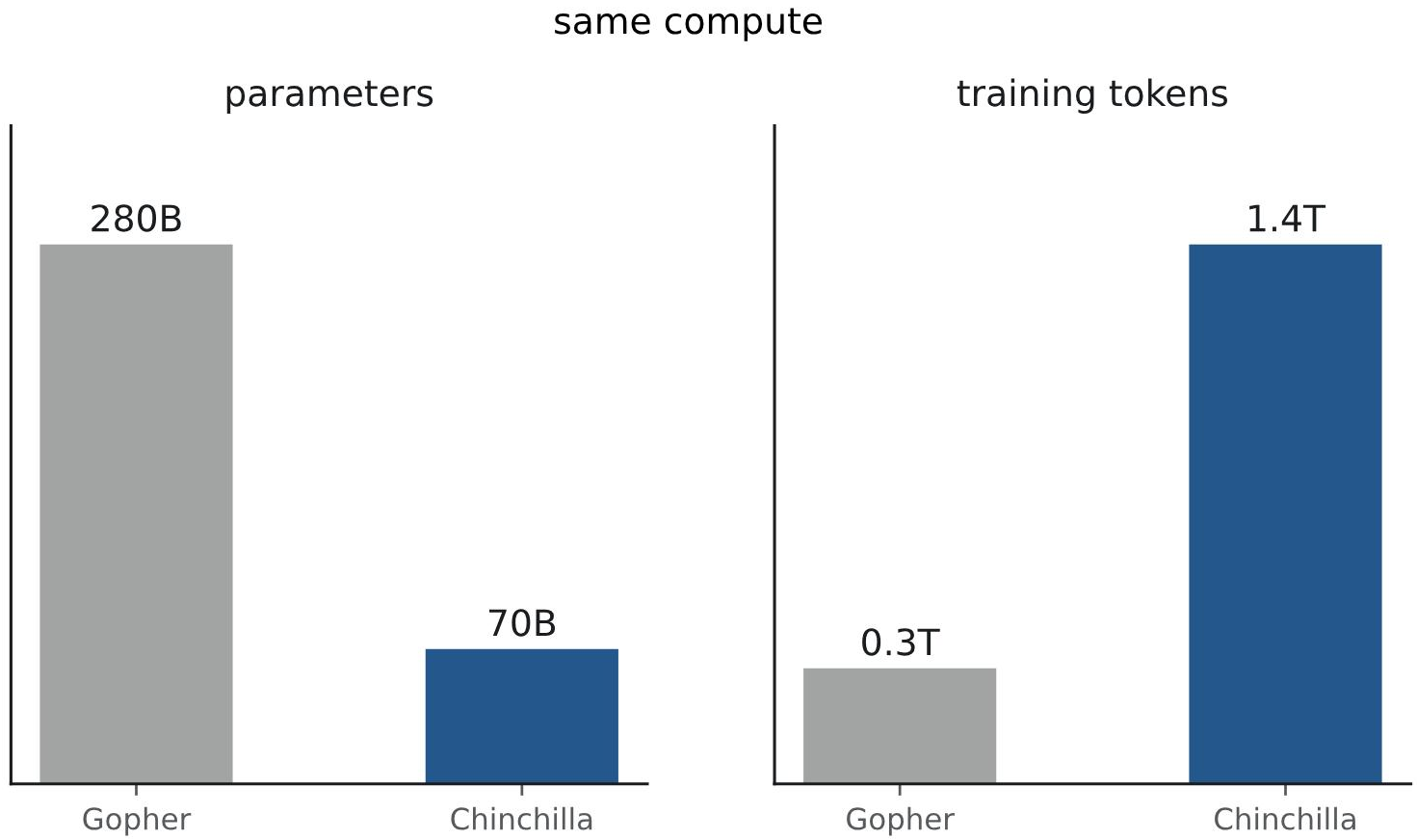
Both exponents land near one half: grow *N* and *D* together.

near one half because $\alpha \approx \beta$; that near-equality is the empirical fact

Hoffmann et al., "Training Compute-Optimal Large Language Models," arXiv:2203.15556 (2022), approach 3.

Chinchilla

$$N^* \propto C^{0.5} \qquad D^* \propto C^{0.5} \qquad \frac{D^*}{N^*} \approx 20$$



Hoffmann et al., "Training Compute-Optimal Large Language Models," arXiv:2203.15556 (2022). Rae et al., "Scaling Language Models: Methods, Analysis & Insights from Training Gopher," arXiv:2112.11446 (2021).

Sizing on a Budget

$C = 10^{21}$ FLOPs. How big a model?

$$C = 6ND, \quad D = 20N \quad \Rightarrow \quad C = 120\,N^2$$

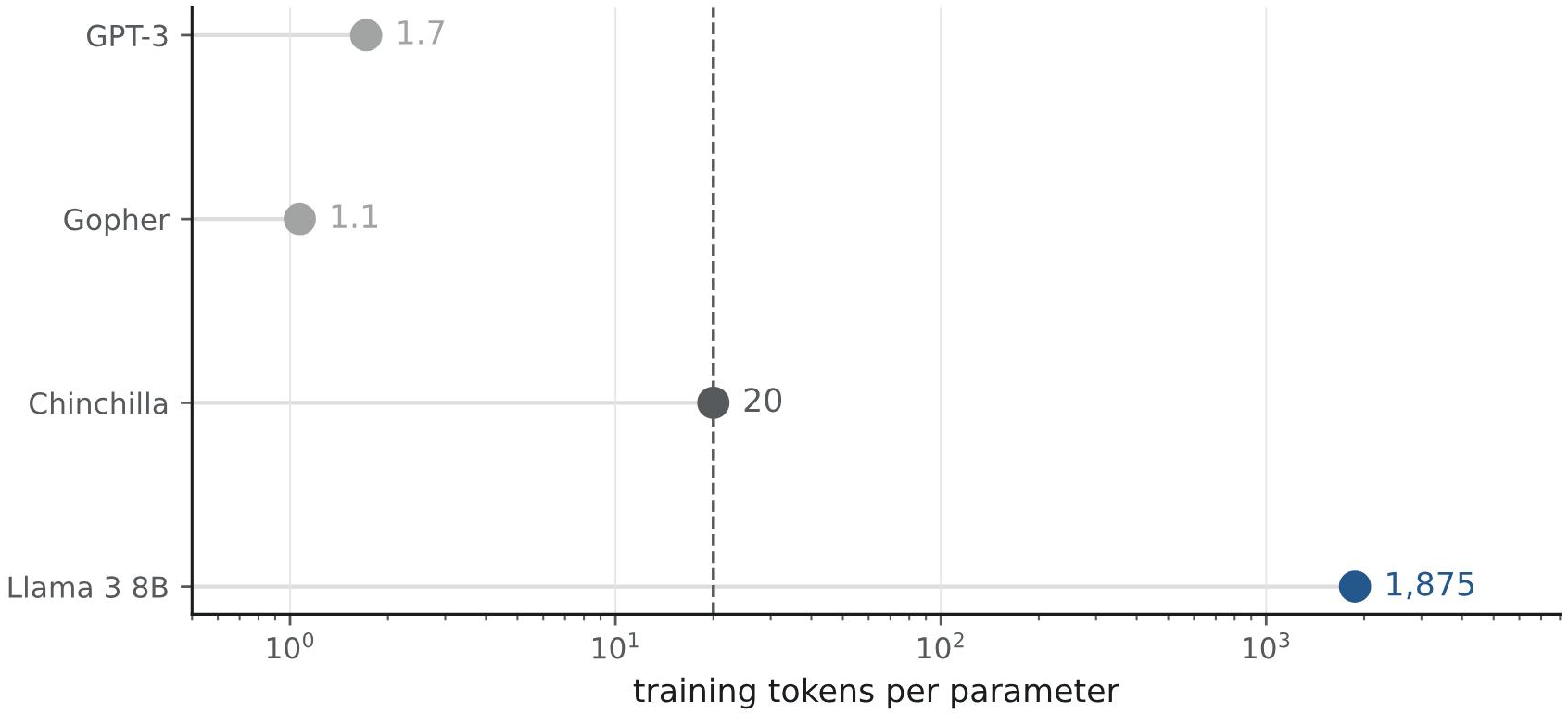
$$N^* = \sqrt{\frac{10^{21}}{120}} \approx 2.9\text{B} \qquad D^* \approx 58\text{B}$$

The budget determines both numbers: 2.9B parameters, 58B tokens.

check: Chinchilla's own budget, $C = 5.8 \times 10^{23}$, gives $\sqrt{C/120} \approx 70$B. That is Chinchilla.

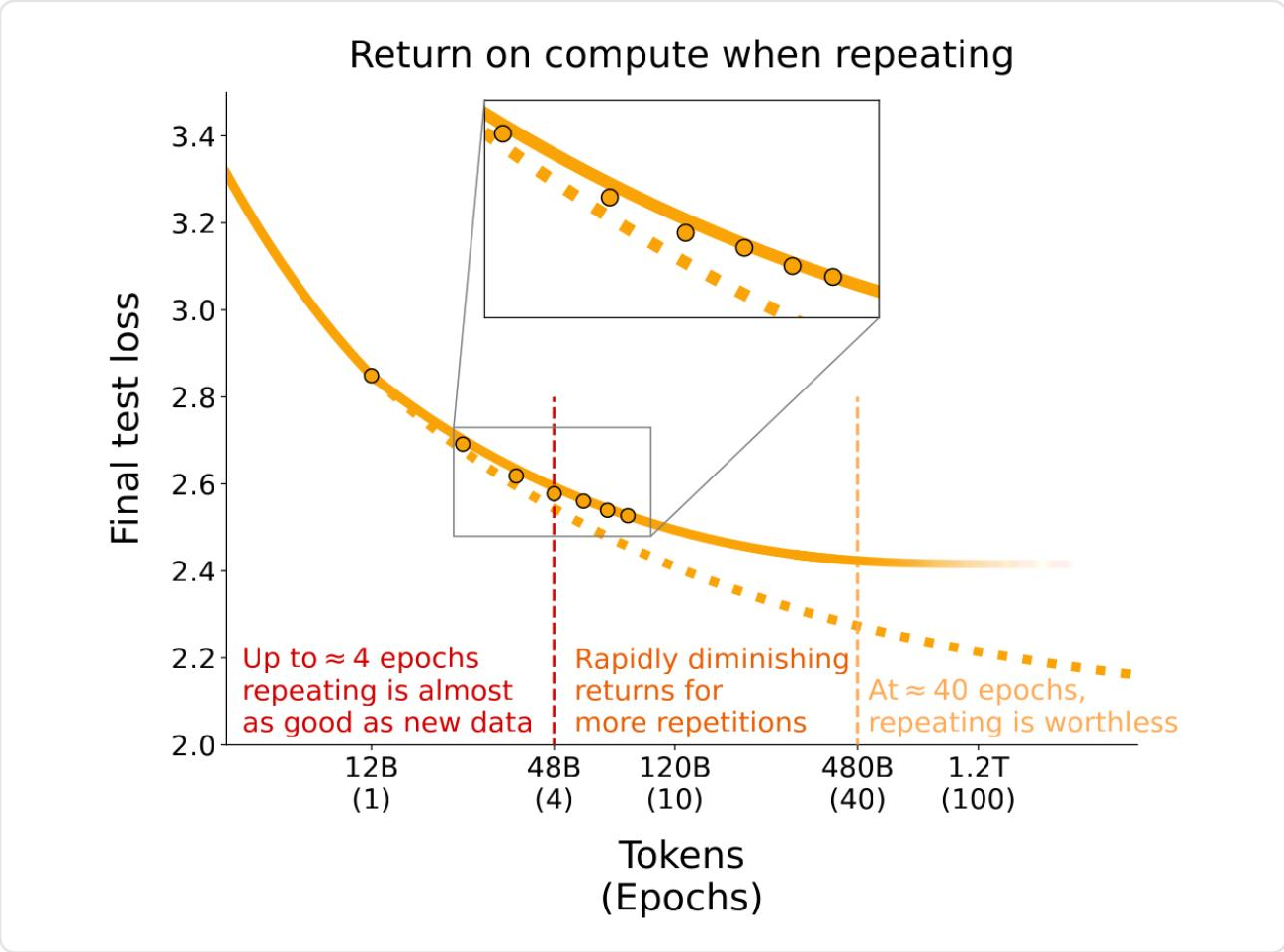
Overtraining

The 20:1 ratio minimizes loss for a fixed training budget. It says nothing about what serving costs.



Grattafiori et al., "The Llama 3 Herd of Models," arXiv:2407.21783 (2024). Sardana et al., "Beyond Chinchilla-Optimal: Accounting for Inference in Language Model Scaling Laws," ICML 2024.

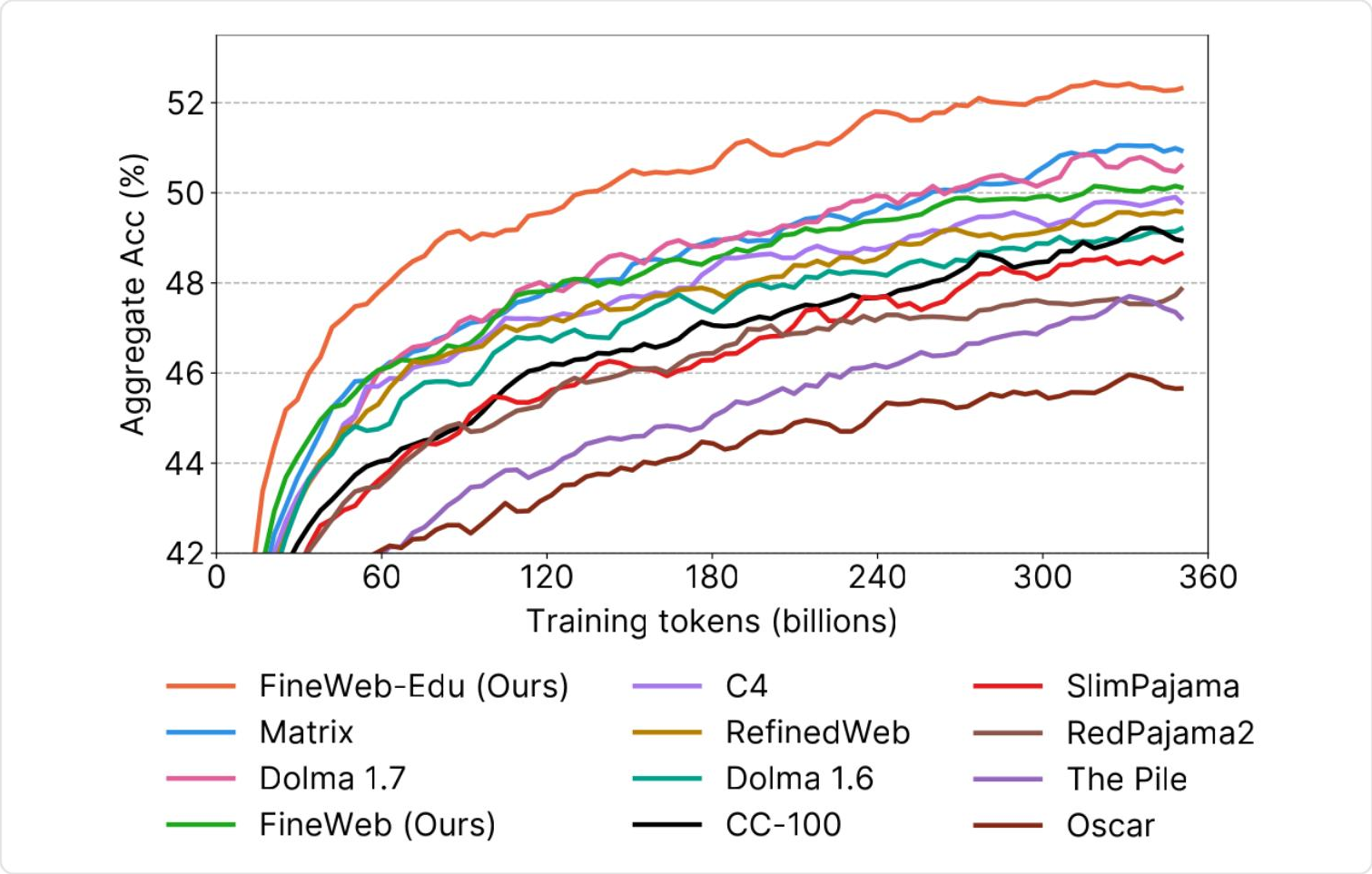
The Data Wall



Our Shakespeare ladder trains at epoch 80, off the right edge of this plot.

Muennighoff et al., "Scaling Data-Constrained Language Models," NeurIPS 2023, Figure 1 (left).

Quality



Filtering buys tokens that count, so a smaller corpus can act like a larger one, IMO.

Penedo et al., "The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale," NeurIPS 2024, Figure 10.

3 CODESIGN

Codesign

What a Budget Buys

1,000 H100s $\times$ 30 days $\times$ 40% MFU?

MFU is model FLOPs utilization: the fraction of the chips' peak rate the run actually reaches.

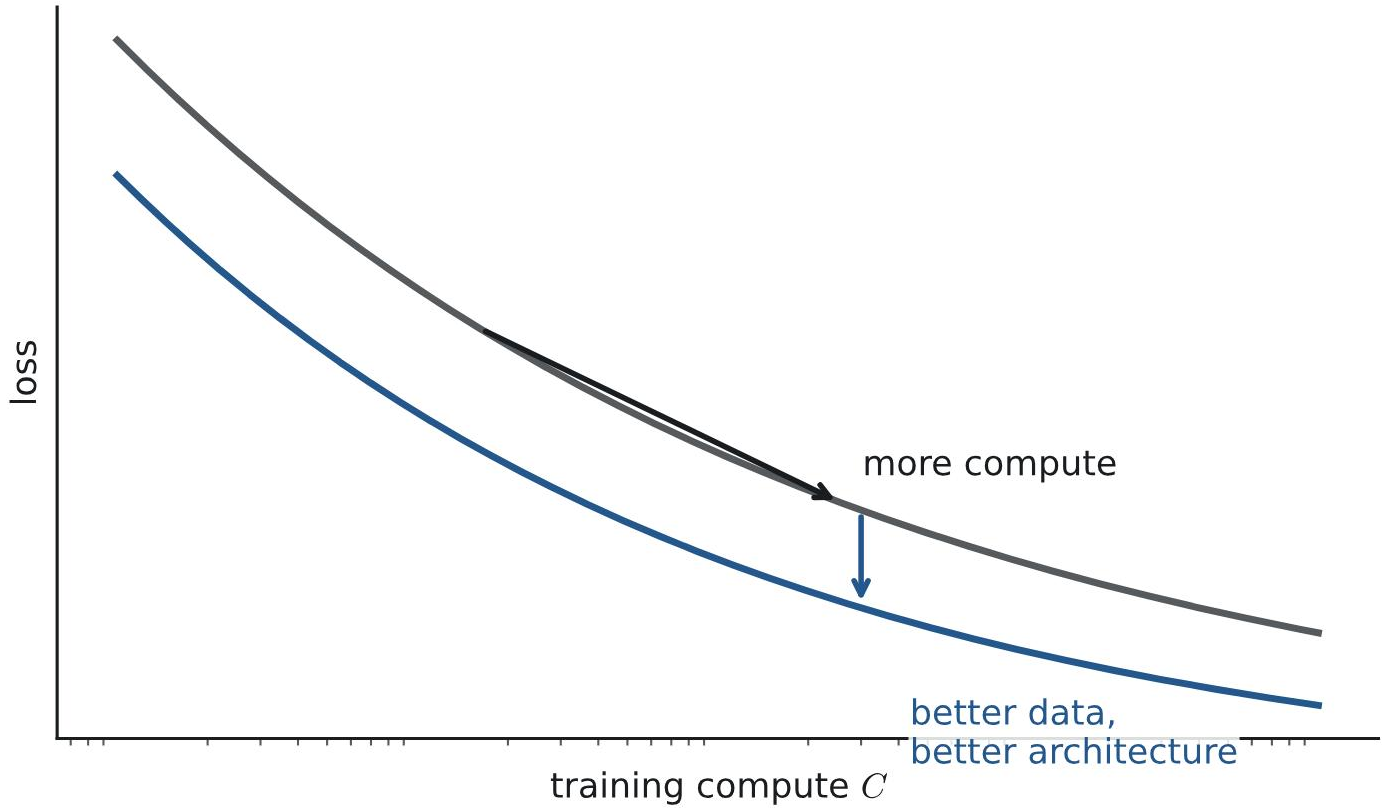
$$C \approx 1.0 \times 10^{24}\ \text{FLOPs}$$

$$N^* = \sqrt{C/120} \approx 93\text{B} \qquad D^* \approx 1.9\text{T}$$

Overtraining then says: build it smaller, train it longer.

at $2 per GPU-hour, this month of cluster time is about $1.4M, before any inference cost

One Currency



Measure the shift, price it in FLOPs.

4 LIVE

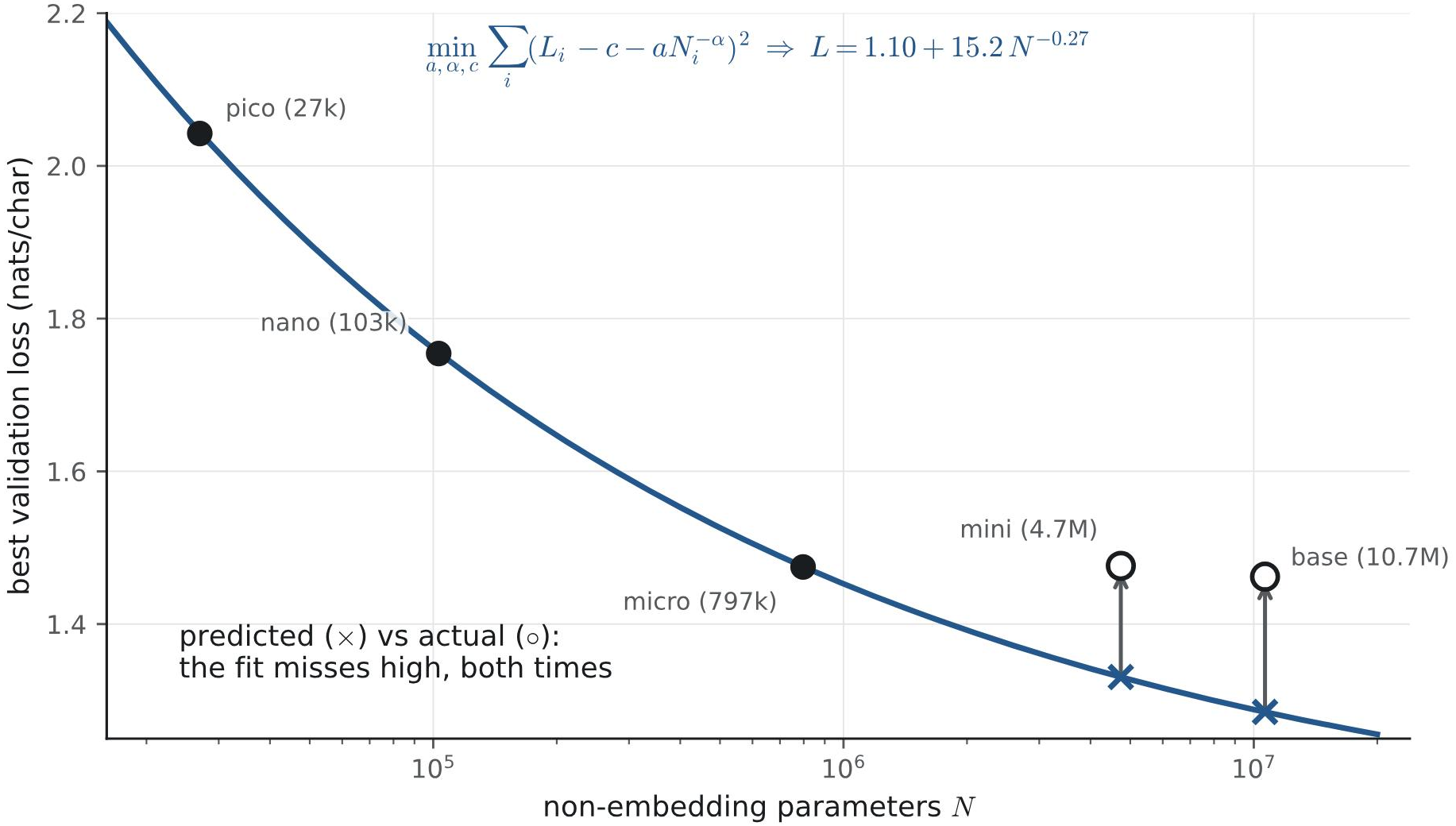
Let’s Play

The Ladder

run	layers	width	N	D
pico	2	32	27k	82M
nano	2	64	103k	82M
micro	4	128	797k	82M
mini	6	256	4.7M	82M
base	6	384	10.6M	82M
your prediction			mini: _____	base: _____

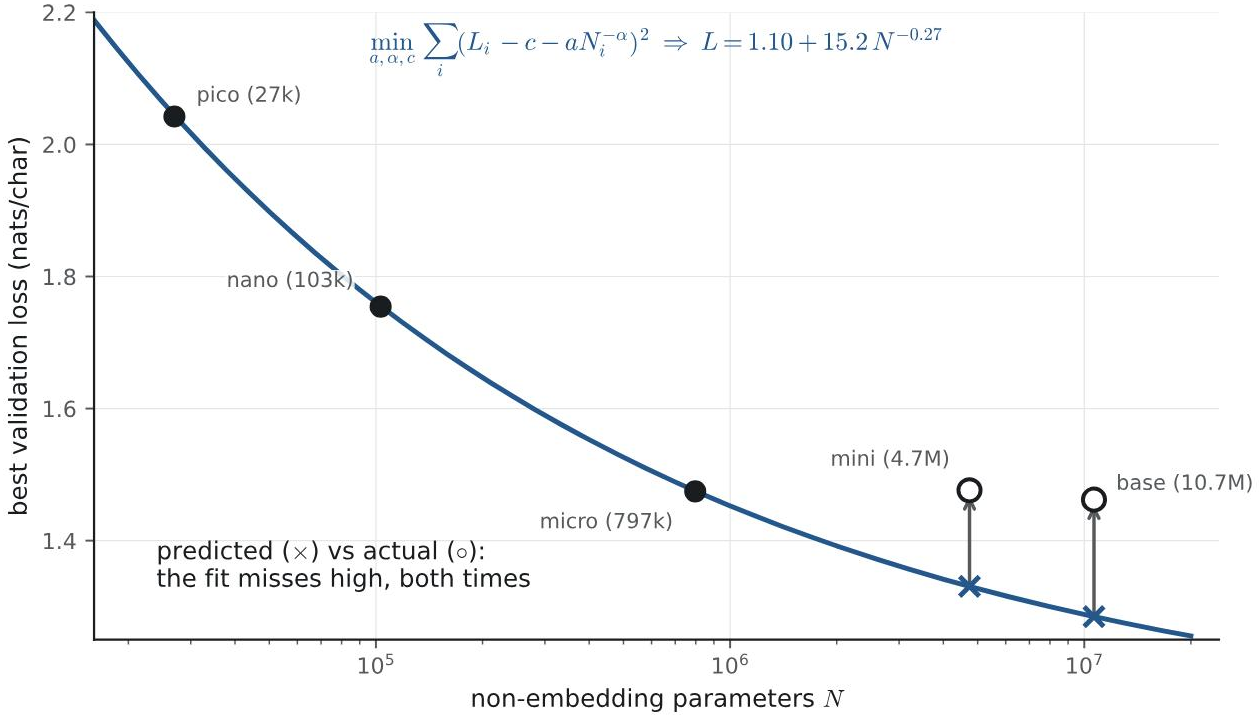
Fit the three smallest. **Predict mini and base.**

The Fit



Assignment 1 codebase, five runs, best validation loss per run. You fit this yourself in Part 3.

What Just Happened?

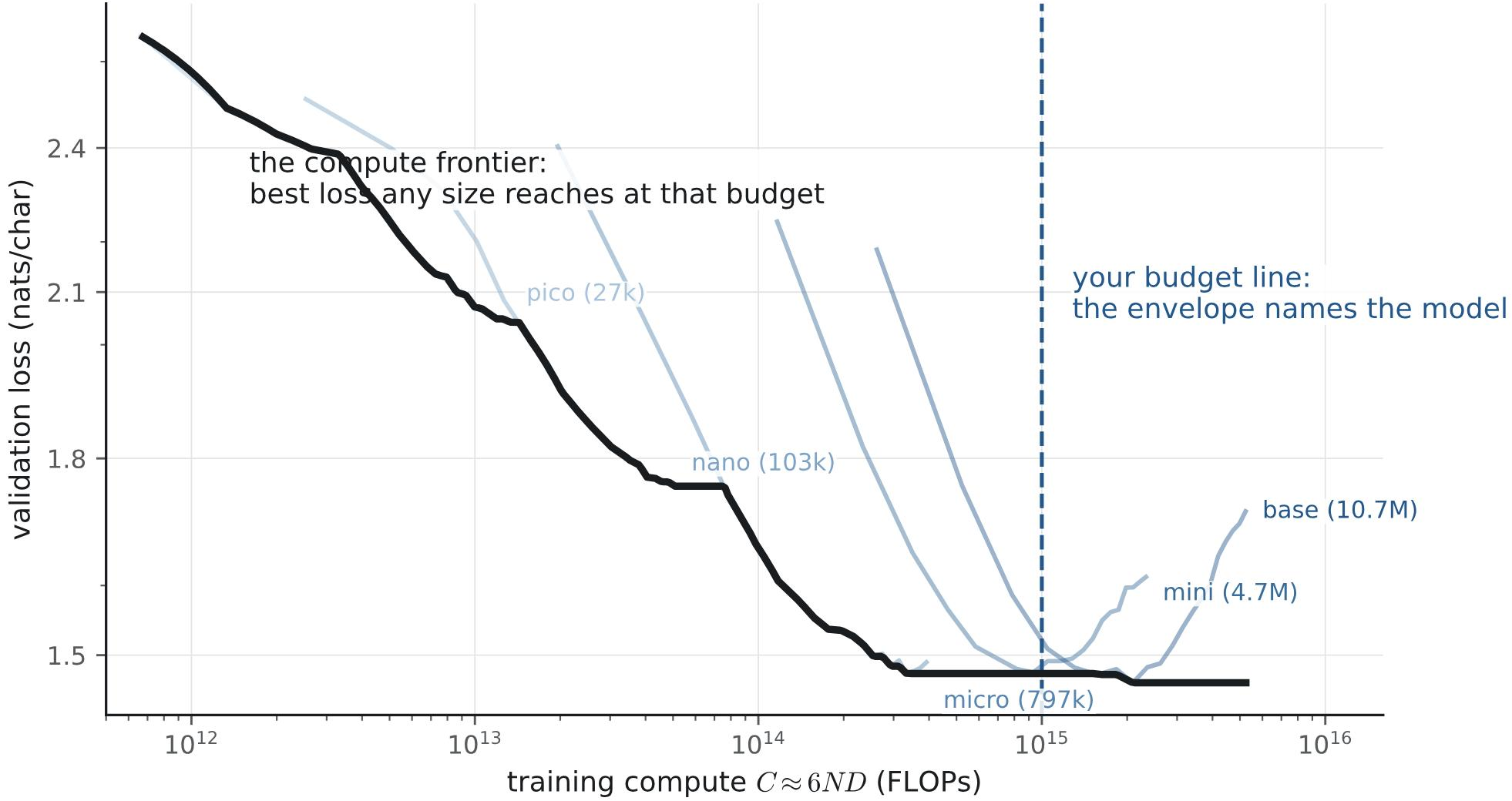


The fit assumed unlimited data.

base sits at 7.7 tokens per parameter, on epoch 80.

Kaplan’s power law at the bottom, Muennighoff’s data wall at the top, Chinchilla’s ratio predicting where.

The Frontier



Same five runs, x-axis now training compute C = 6ND accumulated during each run.

Reading

Required. Hoffmann et al. 2022, "Training Compute-Optimal Large Language Models."

Optional. Kaplan 2020 · Muennighoff 2023 · Llama 3 herd 2024 · Sardana 2024

Assignment 1, Part 3 is today's demo, done by you.

your seeds, your fit, your explanation of the miss

Thursday: Architectures. Bring a shape table.

any open model's config: n_layer, n_head, d_model, vocab; we count its parameters by hand