Google

Gemini Pretraining

Vlad Feinberg, vladf@google.com, vladfeinberg.com

Apr 2025

Agenda

01 Introduction

02 Classical Scaling

03 Small Model Customers

04 Inference-optimized Scaling

05 Closing Thoughts

01

Introduction

About Me

2017

P'17 COS + SML

w/ Kai: 3D CNN for MRI segmentation for connectome reconstruction

2018

ML Sys PhD
Berkeley RISE Lab
w/ Ion, Joey, Mike

Model-based Deep RL
MuJoCo (w/ Sergey)

2019

Dropped out for startup!

Sisu Data w/ Peter Bailis
Efficient DB cubing w/ FDR control via custom distributed lasso engine

2021

Google Cerebra

Quantizing Ads DNN for pCTR serving efficiency

2022

Google Brain ⟶ GDM

Optimizer work w/ Elad
Inference-efficient LLMs
Flash Pretraining Lead

Gemini

What This Talk Will Be About

- Classic Scaling
  - Basic methodologies for pre-training large language models (LLMs)
  - Foundational lessons learned in the field
- Inference-Optimized Scaling
  - How the above methodologies interact with practical serving needs.
  - Basic roofline methodology, but without sharding

What This Talk Won’t Be About

- Specialized capabilities research
  - image/audio/video in/out
  - long context
- Thinking or post-training
- Evals

Presumed Background Knowledge

- Language modelling concepts, decoder-only transformers
- Distributed computing

E.g., from level from previous papers read:

DeepSeek-V3 Technical Report, DeepSeek-AI et al., 2024 https://arxiv.org/abs/2412.19437

GSPMD: General and Scalable Parallelization for ML Computation Graphs, Yuanzhong Xu et al., 2021, https://arxiv.org/abs/2105.04663

Disclaimer: borrowing some amazing slides from fantastic co-workers, Jean-Baptiste Alayrac, Sebastian Borgeaud, and Jacob Austin

02

Classical Scaling

If I give you a certain amount of compute C (e.g. 1000 H100 for 30 days), what is the best LLM you can train?

What should be its size (=N)?
How many tokens (=D) should it be trained on?

If I give you a certain amount of compute `C` (e.g. 1000 H100 for 30 days), what is the best LLM you can train?

What should be its size (`=N`)?
How many tokens (`=D`) should it be trained on?

Note: for a transformer `C = 6 * N * D` is a very good approximation of FLOPs.*

*Excluding self-attention, an N-parameter decoder-only model requires 6N matmul FLOPs per token seen (2N for forward and 4N for backward), because each matmul performs one multiplication and one addition per pair of input values, and the backward pass includes two matmuls for each one in the forward pass.

How many FLOPs in each training step?

MLPs

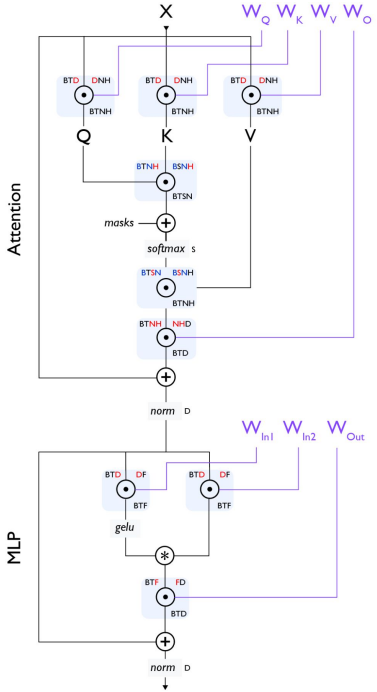
The MLPs of a transformer typically consist of 2 input matmuls that are element-wise combined and a single output matmul:

operation	train FLOPs	params
$A[B,T,D] \cdot W_{in1}[D,F]$	$6BTDF$	DF
$A[B,T,D] \cdot W_{in2}[D,F]$	$6BTDF$	DF
$\sigma(A_{in1})[B,T,F] * A_{in2}[B,T,F]$	$O(BTF)$	
$A[B,T,F] \cdot W_{out}[F,D]$	$6BTDF$	DF
	$\approx 18BTDF$	$3DF$

Attention

For multi-headed attention, let us assume equal head dimension H for **Q**,**K**,**V** projections, and estimate the cost of the **QKVO** matmuls:

operation	train FLOPs	params
$A[B,T,D] \cdot W_Q[D,N,H]$	$6BTDNH$	DNH
$A[B,T,D] \cdot W_K[D,N,H]$	$6BTDNH$	DNH
$A[B,T,D] \cdot W_V[D,N,H]$	$6BTDNH$	DNH
$A[B,T,N,H] \cdot W_O[N,H,D]$	$6BTDNH$	DNH
	$24BTDNH$	$4DNH$



symbol	dimension
B	batch
L	number of layers
T	sequence length (query)
S	sequence length (key value)
V	vocab
D	d_model, embedding dimension
F	MLP hidden dimension
H	attention head dimension
N	number of query heads

The dot-product attention operation is more subtle, effectively being a $TH \cdot HS$ matmul batched over the B, K dimensions, a softmax, and a $TS \cdot SH$ matmul again batched ov B, K dimensions. We highlight the batched dims in blue:

operation	train FLOPs
$Q[B,T,K,G,H] \cdot K[B,S,K,H]$	$6BTSKGH = 6BTSNH$
$\text{softmax}_S \; L[B,T,S,K,G]$	$O(BTSKG) = O(BTSN)$
$S[B,T,S,K,G] \cdot V[B,S,K,H]$	$6BTSKGH = 6BTSNH$
	$\approx 12BTSNH = 12BT^2NH$

Adding these up, we get 18BTDF + 24BTDNH = 6 * BT * (3DF+4DNH) = **6 * num tokens * parameter count (or 2 * ... for forward pass)**

Audience Q: What About MoEs? Why not Attn?

Why do we ask ourselves this question?

ML training before

- Maybe 2 stages; toy problem for iteration (CIFAR10) then you apply to Imagenet
- LR searches by doing multiple “final runs”.
  - The last data point is our test set!

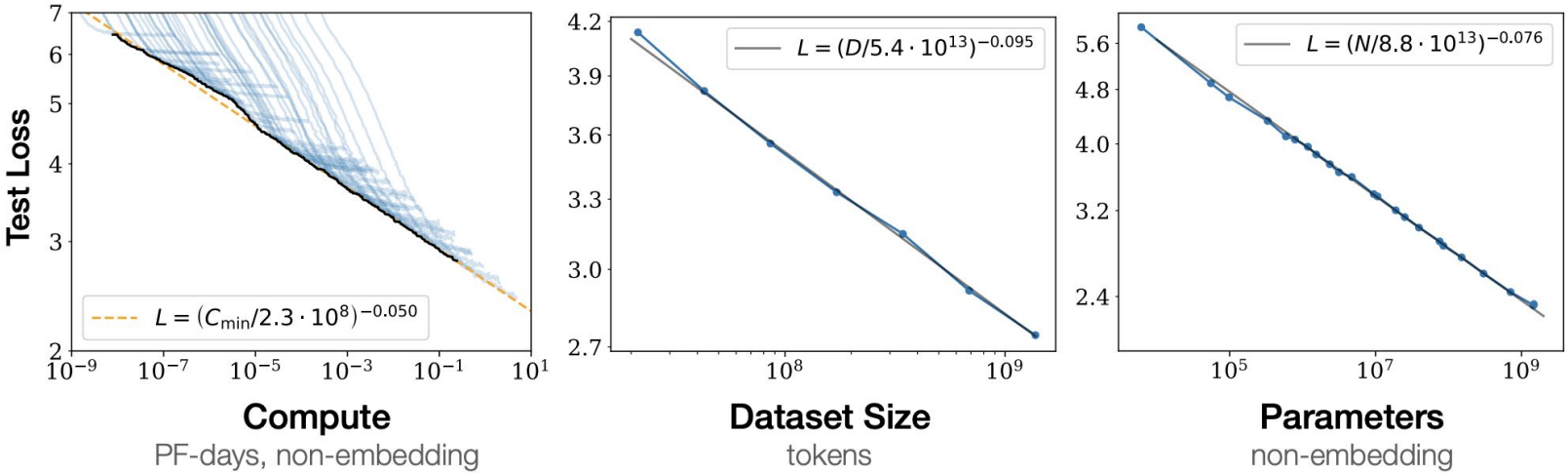
Now every next run requires **extrapolation**.

Analysis made in the **context of a parameterized LLM training recipe!**

Must already have architecture scaling, schedule defined for N, D.
Loss forecast implies model/recipe selection capability!

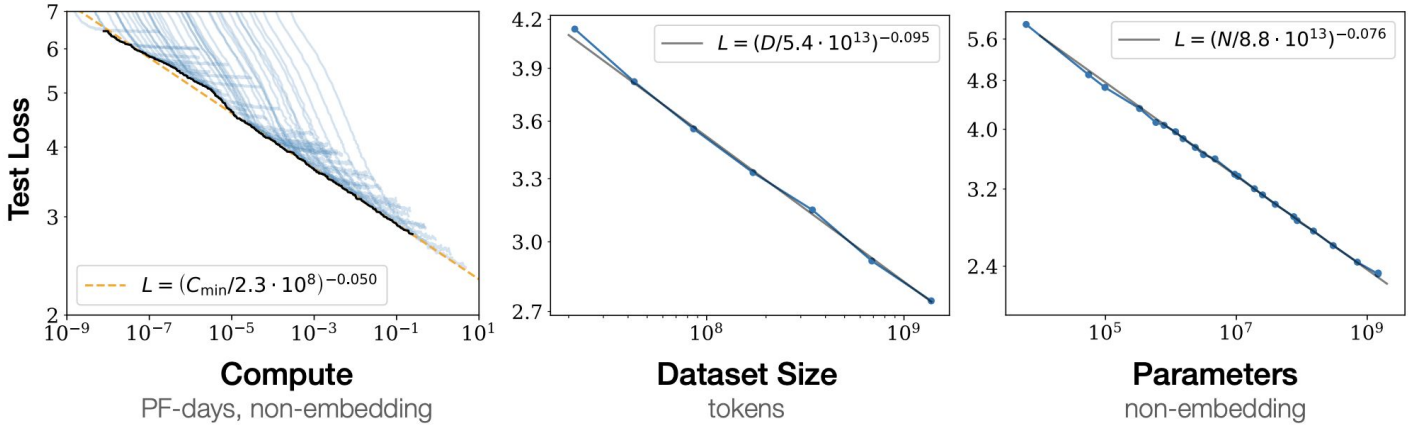
Scaling Laws

Kaplan *et al* (2020)[a] showed that for autoregressive transformers, the performance of much larger models is accurately predicted by a series of smaller models.



Scaling Laws

Given a fixed compute budget C, what is the optimal combination of (N, D) ?



"Maximally compute-efficient training would therefore be far more sample efficient than one might expect based on training small models to convergence, with data requirements growing very slowly as D ~ $C^{0.27}$ with training compute."

Their findings: With a 10x compute budget, parameters should increase by 5.37x and the amount of data by 1.86x.

Scaling Laws

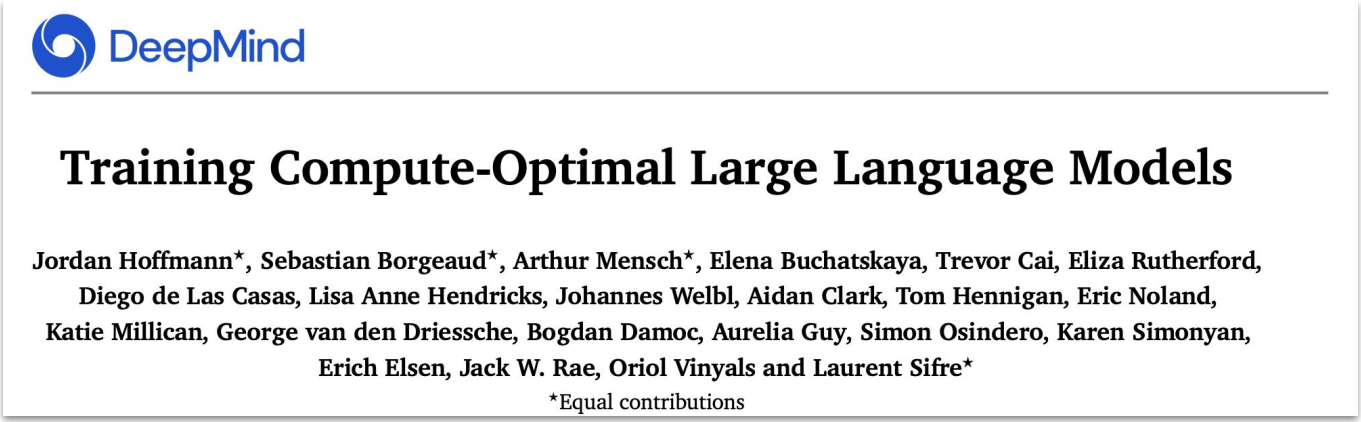
Their findings: With a 10x compute budget, parameters should increase by 5.37x and the amount of data by 1.86x.

Consequences for the industry: We should heavily invest in scaling the model size rather than the data size!

The catch:

- These “laws” are only empirical,
- The fitting of these laws depends a lot on the experimental setup as well as the implicit assumptions being made there.

Chinchilla paper from GDM [b] (March 2022)



Challenges one assumption from the Kaplan paper: Kaplan et al. run a single training run per model size and uses intermediate losses to estimate the loss at different token horizon.

Chinchilla paper: This is a bad approximation as you can get much better losses through proper learning rate decay. Only the final loss value is optimal.

Let's look together the consequences!

The IsoFlops approach

One of the 3 approaches in the paper

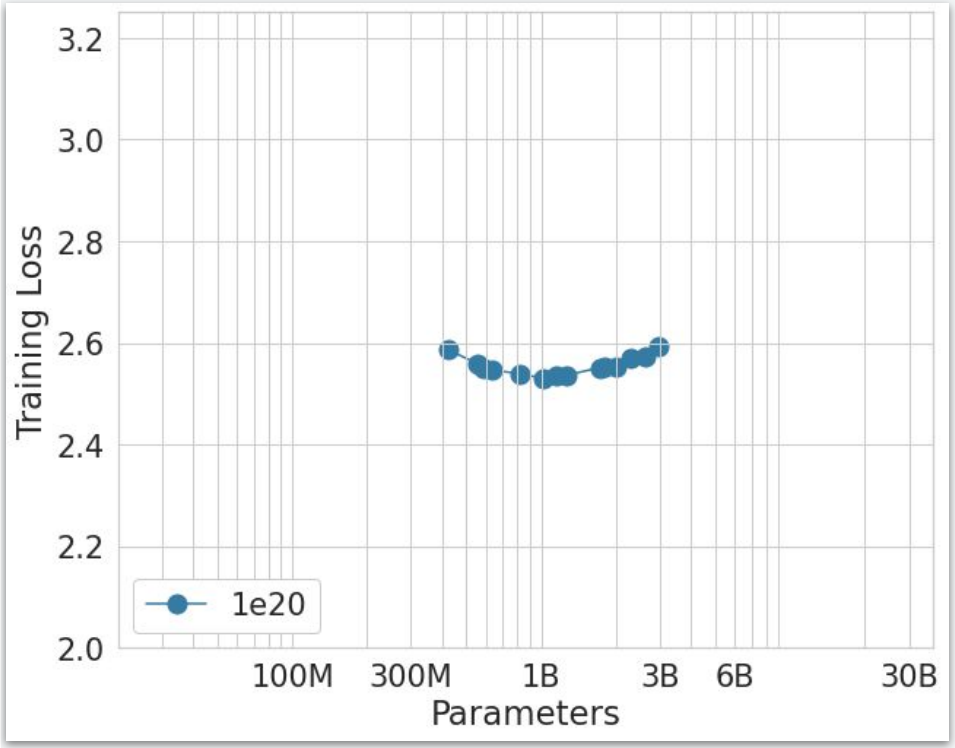
IsoFlops: Fix flops and vary model size and training tokens - $N_{opt}(C)$ and $D_{opt}(C)$

The IsoFlops approach

1. Fix a target FLOPs budget
2. Train a few models, vary model size

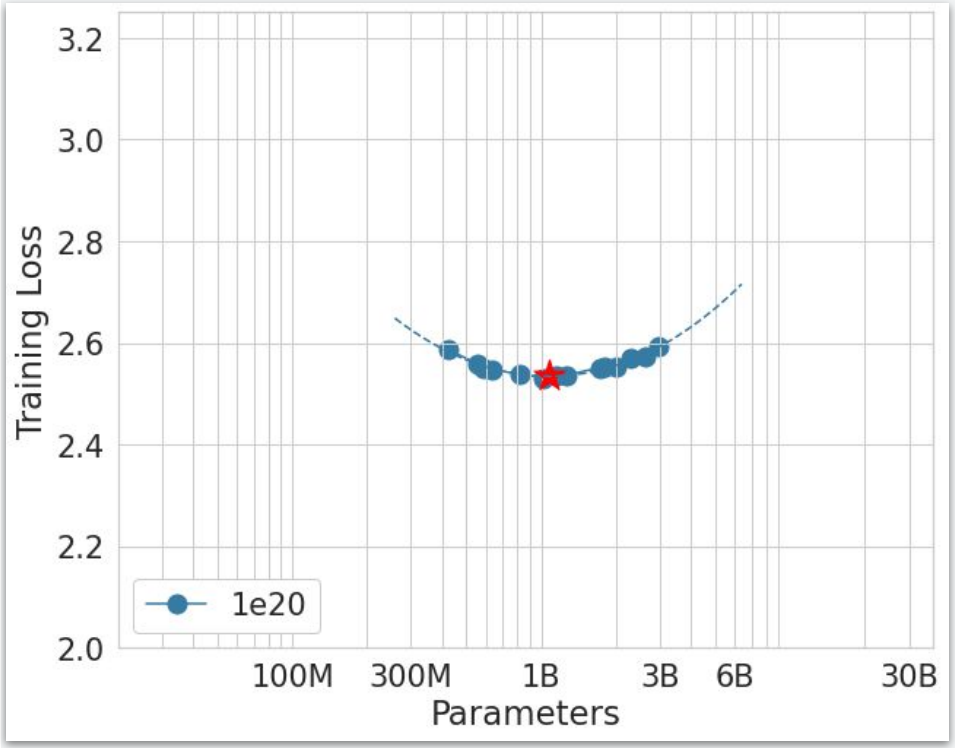
The IsoFlops approach

1. Fix a target FLOPs budget
2. Train a few models, vary model size



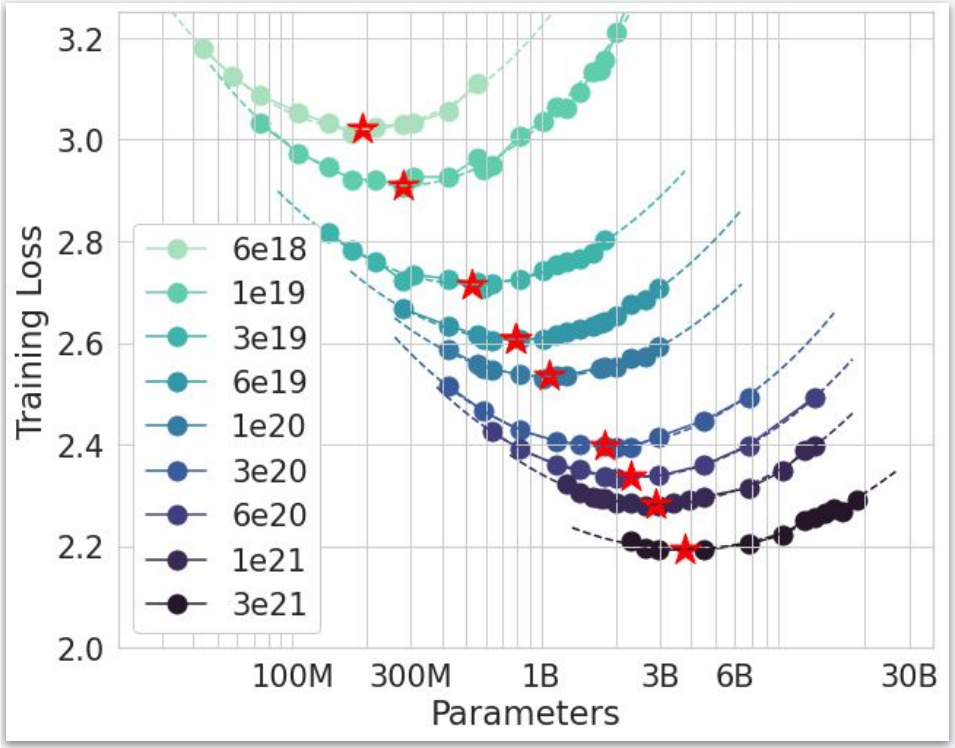
The IsoFlops approach

1. Fix a target FLOPs budget
2. Train a few models, vary model size
3. Fit a parabola and find the minimum



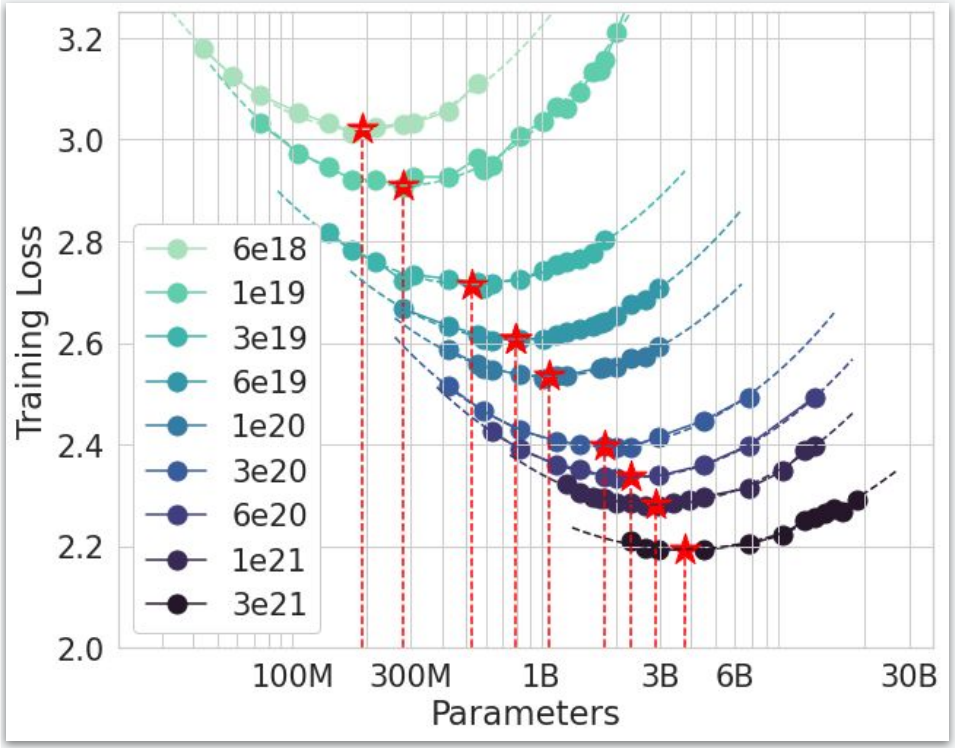
The IsoFlops approach

1. Fix a target FLOPs budget
2. Train a few models, vary model size
3. Fit a parabola and find the minimum
4. Repeat 1-3 for various FLOPs budgets



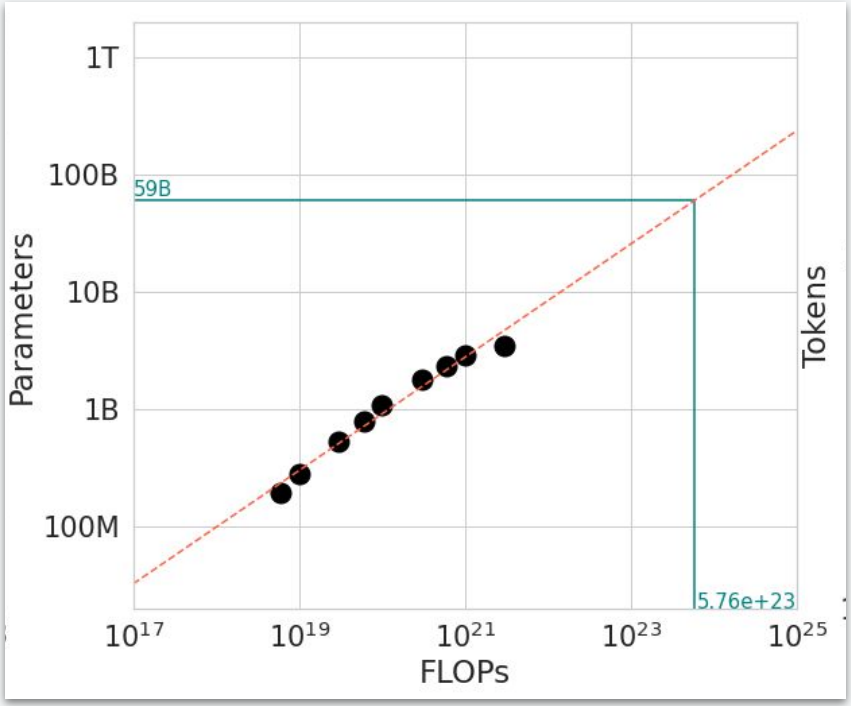
The IsoFlops approach

1. Fix a target FLOPs budget
2. Train a few models, vary model size
3. Fit a parabola and find the minimum
4. Repeat 1-3 for various FLOPs budgets



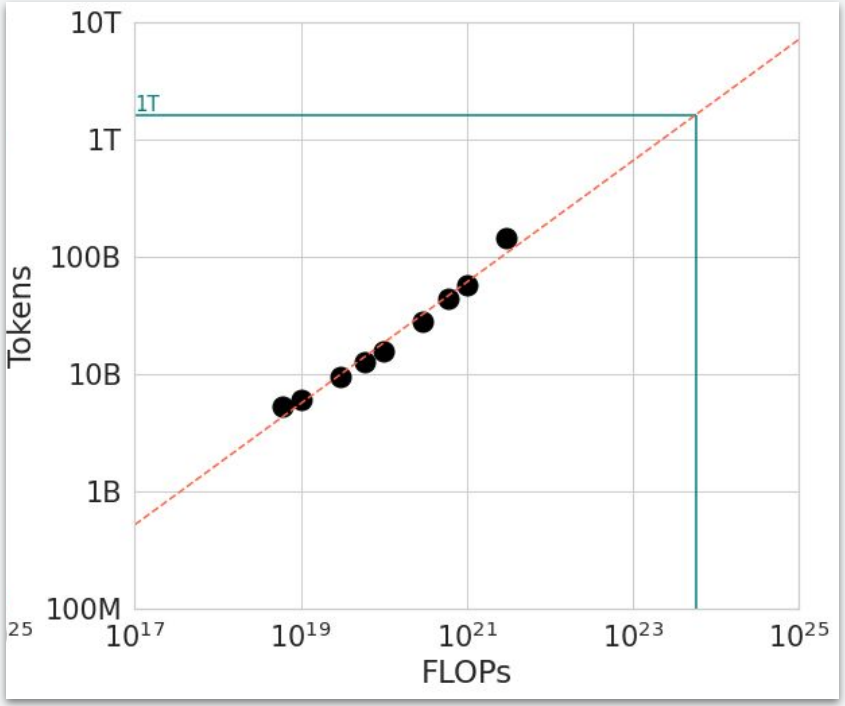
The IsoFlops approach

1. Fix a target FLOPs budget
2. Train a few models, vary model size
3. Fit a parabola and find the minimum
4. Repeat 1-3 for various FLOPs budgets
5. Fit a power law between FLOPs budget and optimal model size

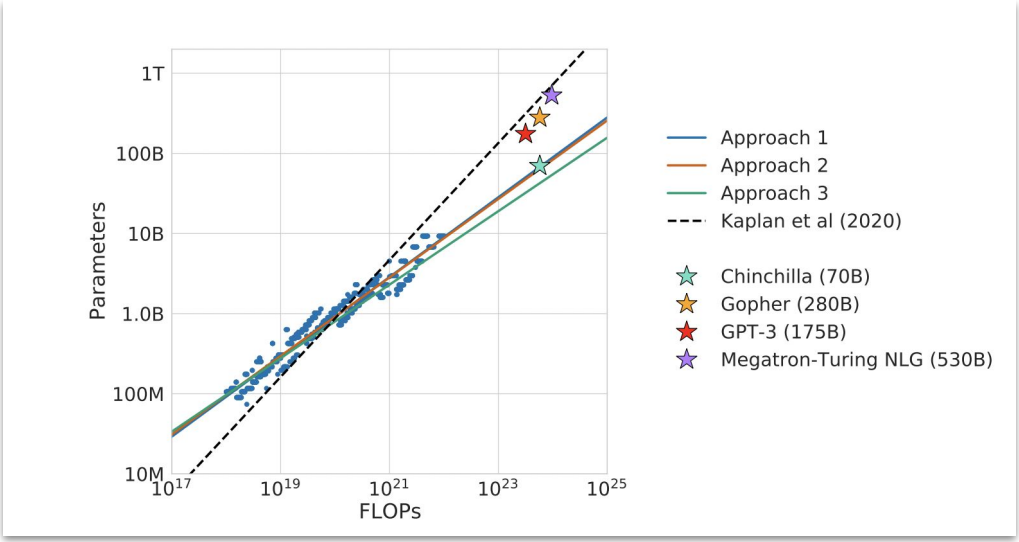


The IsoFlops approach

1. Fix a target FLOPs budget
2. Train a few models, vary model size
3. Fit a parabola and find the minimum
4. Repeat 1-3 for various FLOPs budgets
5. Fit a power law between FLOPs budget and optimal model size
6. Fit a power law between FLOPs budget and optimal dataset size

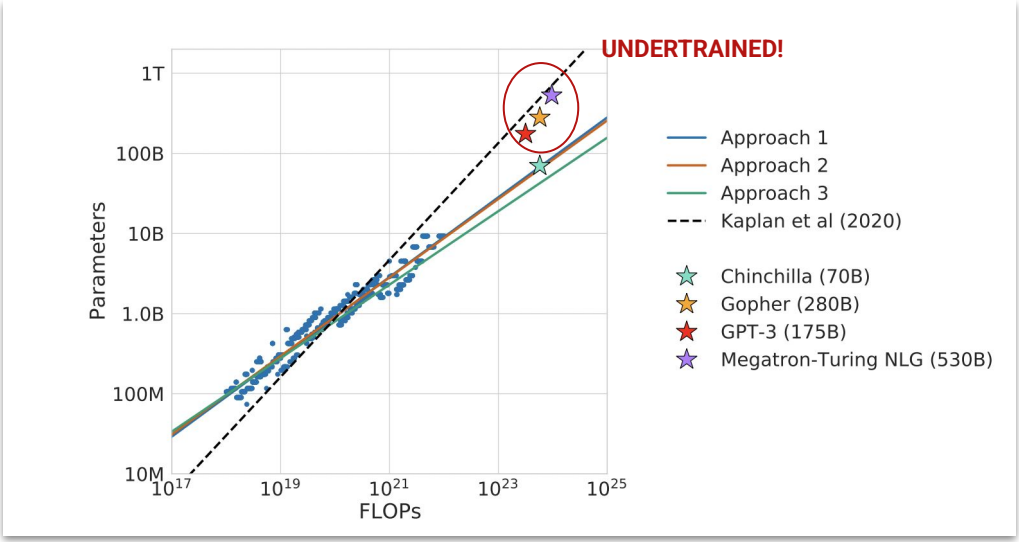


Chinchilla Scaling Laws



Chinchilla findings: the exponent in the power law is ~0.5, meaning model and data size should be scaled at the same rate! This is widely different from Kaplan et al.

Chinchilla Scaling Laws



Chinchilla findings: the exponent in the power law is ~0.5, meaning model and data size should be scaled at the same rate! This is widely different from Kaplan et al.

Consequences: Given a compute budget, models should be smaller and trained for longer. Kaplan's scaling laws meant that models were undertrained – which is obviously bad given bigger models are more expensive to serve and use downstream!

Slight refinements

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

Joint loss, eval fit.

To make a change, compare baseline vs candidate laws.

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

The End of Scaling?

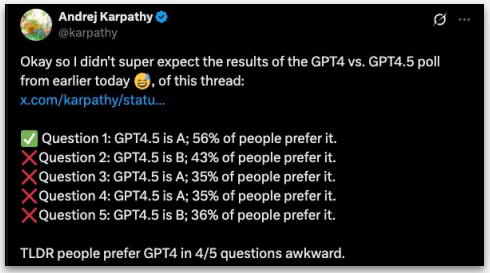
GPT-4.5 Doomers: LMSys is not the end-all-be-all.

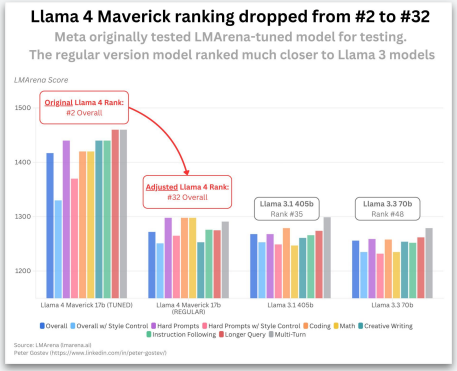
Llama 4 Maverick demonstrated that ranking can be volatile and overfit to human preference.

More importantly:

1. Better NN design still coming
2. Data from new sources being added

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score
1	1	Gemini-2.5-Pro-Exp-03-25	1437
2	2	ChatGPT-4o-latest (2025-03-26)	1406
2	4	Grok-3-Preview-02-24	1402
2	2	GPT-4.5-Preview	1397
5	8	Gemini-2.0-Flash-Thinking-Exp-01-21	1380





Google

Better Algorithms

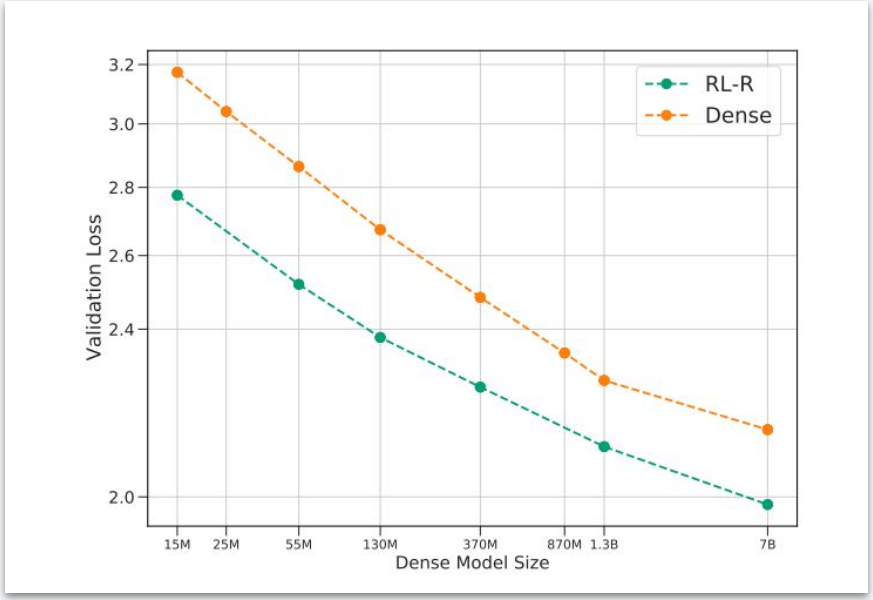
MoE scaling laws are better, but have implications for token hunger. We're running out of internet!

Table 1: Values of the fitted coefficients.

Model	a	α	b	β	g	γ	c
MoE	18.1	0.115	30.8	0.147	2.1	0.58	0.47
Dense	16.3	0.126	26.7	0.127	-	-	0.47

Beta is the data-dependent exponent; alpha is parameter.

Notice relative data hunger compared to dense!



At same active param count and fixed 100B token training, MoE 64E improves on dense

More Data Sources

Unsurprisingly, this is where we spend most of our time, even as modelling people. Probably half my focus this year so far.

1. Multimodal Data
    1. Audio, visual, 3D, videos, etc.

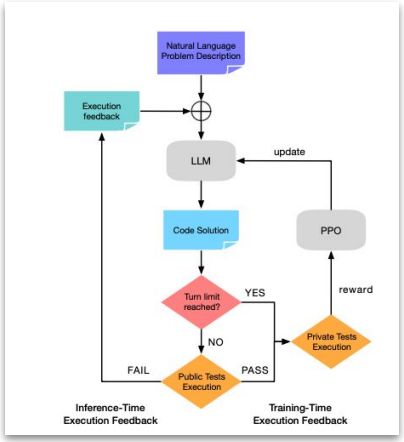
2. Synthetic Data
    1. Without filter, it can help in the Stein's paradox sense (Jain et al 2024)

    2. Tradeoff: Generation Quality vs. Filtering

$$\widehat{\boldsymbol{\theta}}_{JS} = \left(1 - \frac{(m-2)\sigma^2}{\|\mathbf{y}\|^2}\right)\mathbf{y}. \qquad \mathrm{E}\left[\left\|\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right\|^2\right], \qquad \mathbf{Y} \sim N_m(\boldsymbol{\theta}, \sigma^2 I),$$

$$\widehat{R}_{n,m}(\boldsymbol{\theta}; \alpha) := \frac{1-\alpha}{n}\sum_{i=1}^{n} \ell(\boldsymbol{\theta}; \boldsymbol{z}_i) + \frac{\alpha}{m}\sum_{i=1}^{m} \ell(\boldsymbol{\theta}; \boldsymbol{z}_i^s) + \Omega(\boldsymbol{\theta}),$$

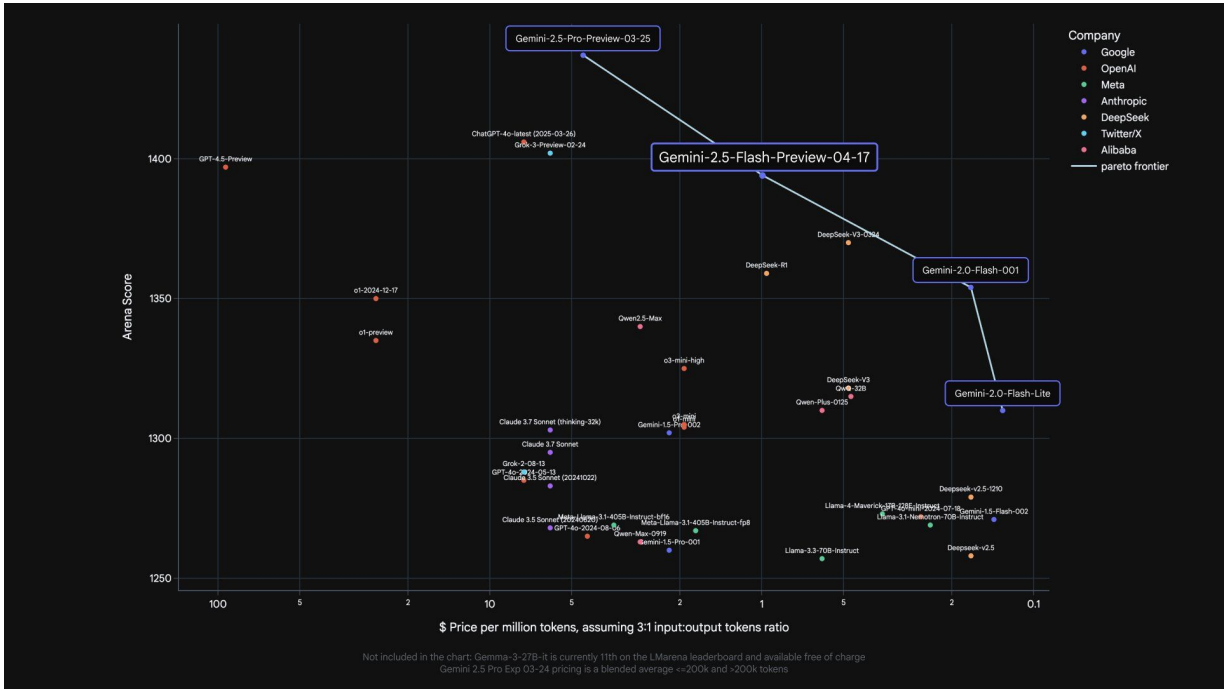
For appropriate alpha…



03

Small Model Customers

Inference Efficiency Goals



What Motivates Being on the Pareto Frontier

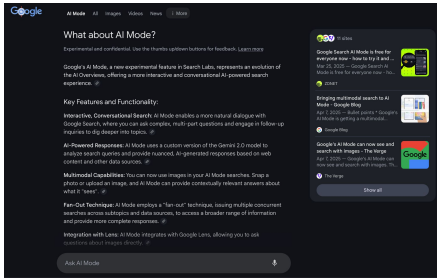
Key Google Use Cases demand

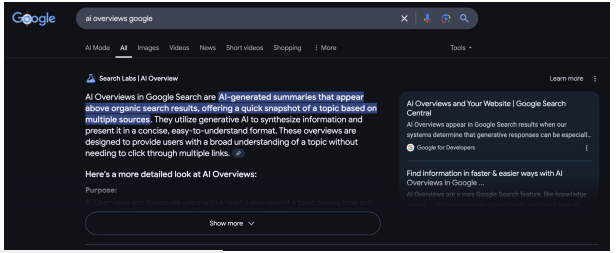
(1) higher volume servicing
(2) real time

All of these require Flash (+Flash-lite)

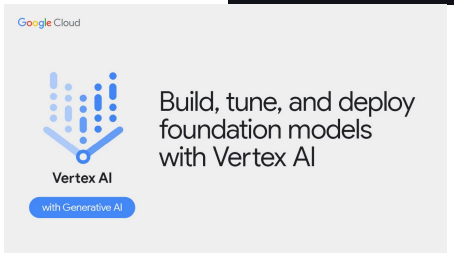
- Free Tier Gemini App (chatbot)
- AIO
- AIM
- Vertex AI (finetuning, deploying)
- AI Studio (generation API)

Gemini





Google AI Studio



Real-time #1: Astra

Shout out to Tara Sainath for the video!

Real time #2: Mariner

Shout out to Anmol Gulati for the video!

Why Do Real-time Use Cases Imply Smaller Models

Some quick back of envelopes

- Consider a web interaction agent with:
  - 128k prefill, but only 8k incremental
  - 128 decode (say it takes that many tokens to get an action)
- Suppose further that we don't want more than a second of latency between actions.
- And say 250ms is scaffolding, load balancing, request validation, kv cache retrieval etc. (optimistic!)

An experiment with Llama3-70B and v5e chips...
Assume fully compute bound on prefill and hbm on decode

```
# layers, model, hidden, heads, kv heads, key size, embedding
L, D, F, H, V, K, E = 80, 8192, 3.5 * 8192, 64, 8, 128, 128256
nparam = L*(3*D*F+2*D*K*H+2*V*D*H)/1e9 + E*D/1e9; print(f'{nparam:.1f}') # 68.8 B
# v5e specs bf16
hbm_bw = 819 # GBpS
bf16_Tflops = 197 # Teraflops
ici_bw = 50 # GBps
# prefill on 8k incremental for 128k context
C, T = 128 * 1024, 8192
prefill_1chip = 2 * T * nparam / bf16_Tflops / 1000; print(f'{prefill_flop:.1f}') # 5.7 s
```

1-off napkin math For Llama3-70B Inference on v5e

Uh oh… 5.7 seconds for 1 chip. So to hit 0.5 sec api limit we already need to have a 4x4 prefill station of v5e

Audience Q: how would we shard on 4x4?

```
per_token_step = (0.75 - prefill_1chip / 16) / 128 * 1000; print(f'{per_token_step:.2f}') # 3.06
# need 3ms / token to make time limit!
decode_chips = 2 * 4
decode_memory = (2 * V * L * K * C / 1e9 + nparam) / 2 # int4 assum
full_step = decode_memory / hbm_bw / decode_chips * 1000; print(f'{full_step:.2f}') # 6.89
# but just memory on 2x4 takes 10ms! ps. why did I stop at 2x4?
# how to bridge? SPEED!
# we'd need ~2.5x acceptance rate
```

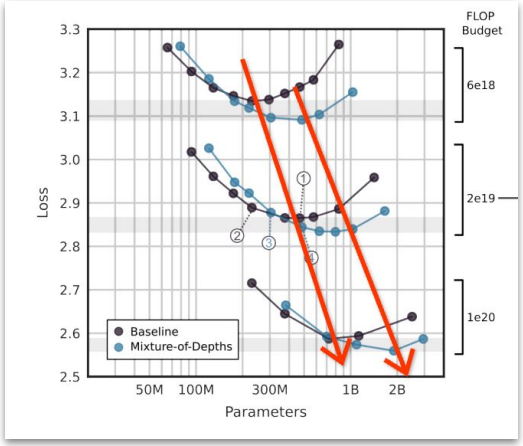
```
one_query_chiptime = prefill_1chip + per_token_step * 128 / 1000 * 8
# 8.86 v5e-seconds per query
# 10k QPS? better have 90k chips ... times 2x
# plus even hitting 50% of rooflines would be great
# plus you need to overprovision for load variability
```

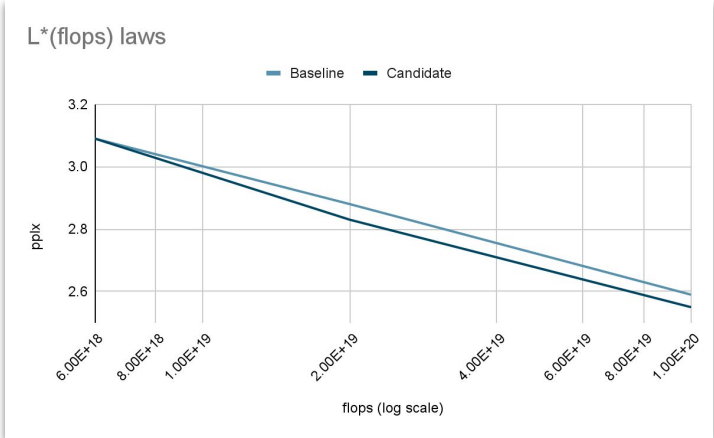
Chinchilla-style Scaling Ignores Inference Cost

How changes get adopted in classical setting:

1. Derive L*(flops) baseline
2. L*(flops) candidate

To the right, isoflop-style
Can also use L(N,D) fits and 6ND.





Google

04

Inference-Aware Scaling

Inference-Aware Laws

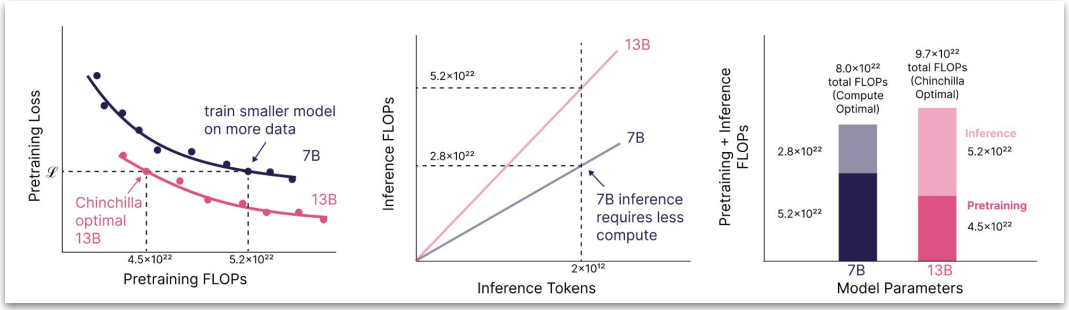
Most direct answer to previous question:

$$N^*(\ell, D_{\text{inf}}), D_{\text{tr}}^*(\ell, D_{\text{inf}}) = \underset{N, D_{\text{tr}} | L(N, D_{\text{tr}}) = \ell}{\arg\min} \; 6ND_{\text{tr}} + 2ND_{\text{inf}}. \tag{3}$$

Globally optimize flops between training and inference?

$$N^*(\ell, D_{\text{inp}}, D_{\text{out}}), D_{\text{tr}}^*(\ell, D_{\text{inp}}, D_{\text{out}}) = \underset{N, D_{\text{tr}} | L(N, D_{\text{tr}}) = \ell}{\arg\min} \; \frac{6ND_{\text{tr}}C_{\text{tr}}}{\text{MFU}_{\text{tr}}} + 2NC_{\text{inf}}\left[\frac{D_{\text{inp}}}{\text{MFU}_{\text{inp}}} + \frac{D_{\text{out}}}{\text{MFU}_{\text{out}}}\right] \tag{6}$$

Beyond Chinchilla-Optimal: Accounting for Inference in Language Model Scaling Laws, Sardana et al., 2024, https://arxiv.org/abs/2401.00448



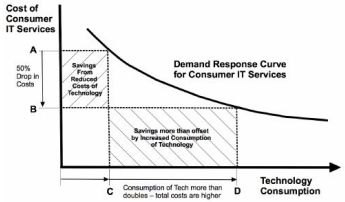
Challenges with Inference-Aware Laws

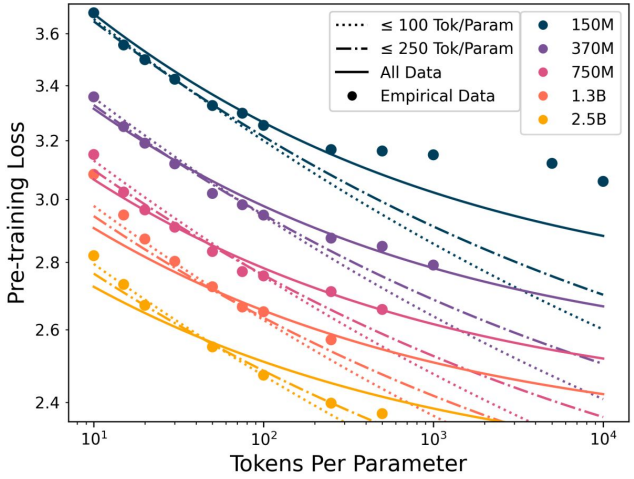
Non-homogeneity of Compute

1. Inference-optimized chips. Also global optimization is not how cross-org planning actually works.
2. But in principle can adjust the formulas for “business cost”

Deeper issue (1): non-forecastability of `Dinf`

1. Jevon’s paradox.
2. Market expansion from quality improvements





Deeper (2): badness of fit

(See Fig5/tbl1 to the right)

Data	α	β	A	B	E
$\leq$100 tok/param	0.08	0.13	7.199	25.97	0.17
$\leq$250 tok/param	0.13	0.16	14.23	39.54	0.98
$\leq$500 tok/param	0.13	0.16	17.07	35.80	0.95
All Data	0.18	0.24	33.66	138.9	1.45
Chinchilla	0.34	0.28	406.4	410.7	1.69

#1: Addressing Badness of Fit from Heavily Overtrained models

Modelling Under Data Constraints

Scaling Data-Constrained Language Models

https://arxiv.org/abs/2305.16264

D was opaque and recipe-specific.
You wouldn't be blamed for assuming iid

New dimension: intentionally unique data, L(N, U, R)

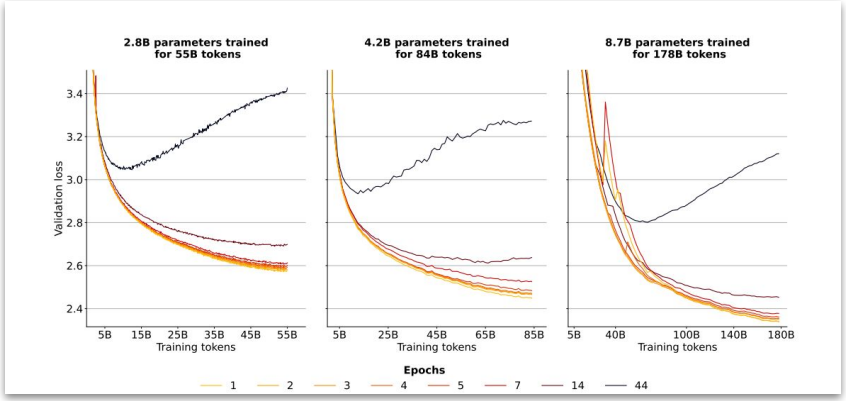
$$D' = U_D + U_D R_D^* (1 - e^{\frac{-R_D}{R_D^*}}) . \quad (5)$$

Note that for $R_D = 0$ (no repetitions), $D' = U_D = D$. For $R_D \ll R_D^*$, $e^{-R_D/R_D^*} \approx 1 - \frac{R_D}{R_D^*}$ and so

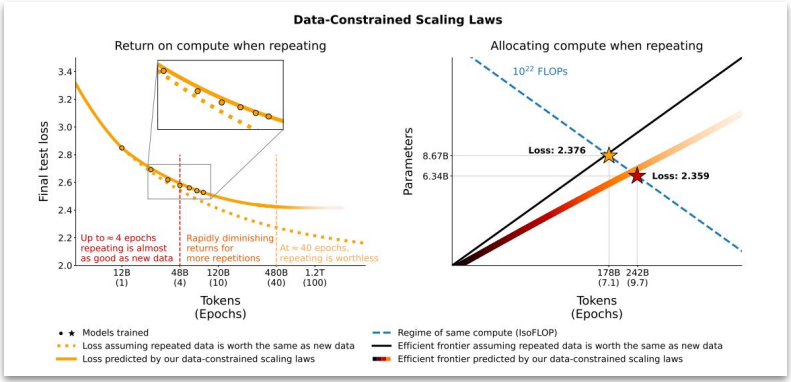
$$D' \approx U_D + U_D R_D^* (1 - 1 + R_D/R_D^*) = U_D(1 + R_D) = D$$

fitted

Upshot: yet smaller models, more resilient to repeats.



Intentionally subset data



Google

#2: What about `Dinf`?

Still, What About Dinf

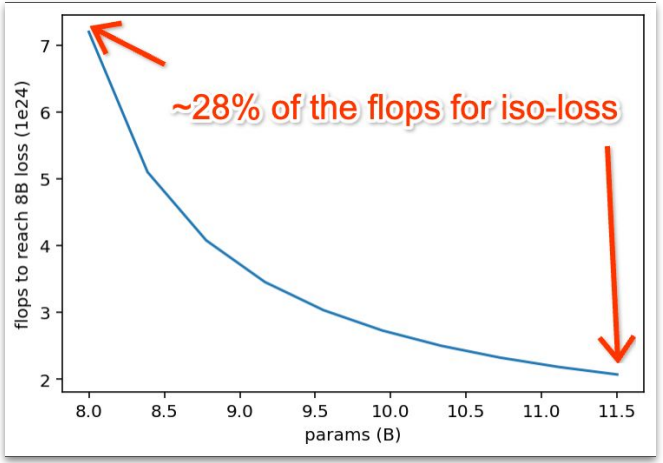
Llama3 : Dinf = inf! https://arxiv.org/abs/2407.21783

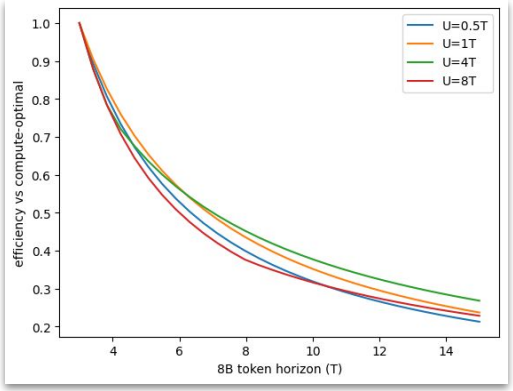
"Both our 8B and 70B parameter models continued to improve log-linearly after we trained them on up to 15T tokens." link

Could be quite valid for open source! Just pick sizes and train on *all your data*!

We could be doing research with those flops! Use this forecast to estimate how much regret we got. Assuming 5 epochs, data-scarce law from prev paper. Colab . Job is to push the curves right.

With L(U, N, R) we can back out ideal shrunk datasets to match loss (5 epochs)





Distillation

Adds other axes to everything we've discussed so far!

Will not say much here; beside the fact that there are other dimensions now too.

Distillation Scaling Laws (Busbridge et al 2025)

Table 2. Scenarios considered in our scaling law applications.

Compute Scenario	δ_T^{Lgt}	δ_T^{Pre}	Description
Best case (fully amortized teacher)	0	0	The teacher produces no additional FLOPs and so we are free to choose the teacher L_T^* that minimizes the student cross-entropy.
Teacher inference	1	0	We don't account for the teacher cost because the teacher already exists, or we intend to use the teacher as e.g. a server model. We still need to pay to use it for distilling a student.
Teacher pretraining	0	1	The teacher needs training, but we store the logits for re-use, either during training, or after training for distilling into sufficiently many students.
Teacher pretraining + inference	1	1	The teacher needs training and we pay for distilling into one student, the worst case scenario.

$$\text{FLOPs} \approx \underbrace{3F(N_S)D_S}_{\substack{\text{Student}\\\text{Training}}} + F(N_T)\Big(\underbrace{\delta_T^{\text{Lgt}} D_S}_{\substack{\text{Teacher}\\\text{Logits}}} + \underbrace{\delta_T^{\text{Pre}} 3D_T}_{\substack{\text{Teacher}\\\text{Training}}}\Big) \quad (9)$$

How to spend flops with teacher?

Student Capacity Gap?

$$\underbrace{L_S(N_S, D_S, L_T)}_{\text{Student cross-entropy}} = \underbrace{L_T}_{\text{Teacher cross-entropy}} + \underbrace{\frac{1}{L_T^{c_0}}\left(1+\left(\frac{L_T}{\widetilde{L}_S d_1}\right)^{1/f_1}\right)^{-c_1 f_1}\left(\frac{A}{N_S^{\alpha'}}+\frac{B}{D_S^{\beta'}}\right)^{\gamma'}}_{\text{Student ability to mimic teacher}} \quad (8)$$

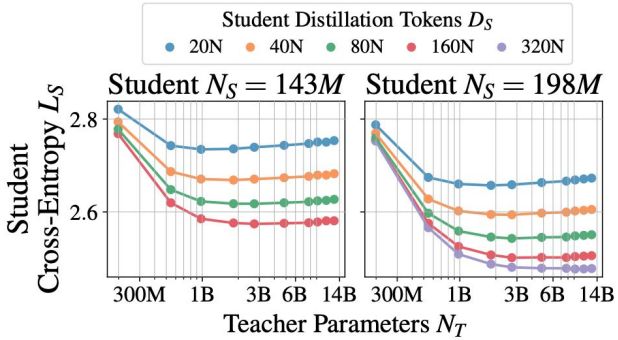


Figure 4. **Fixed M Teacher/Fixed M Student.** Students of two sizes trained with different $M_S = D_S/N_S = 20$ ratios are distilled from teachers with $M_T = D_T/N_T \approx 20$.

Fig4, eq8

(1) very weak effect from up-trend; and not typical regime

(2) Missing part of the story. Teacher pplx can be arbitrarily weakened by just adding temperature!
Take a really good teacher -> Eq8 predicts bad distill -> but add high temp and it will be good?

(3) In practice, you can James-stein this away with weight tuning with supervised objective

Distill blog post story

Distill as variance reduction. Better teacher will just reduce bias

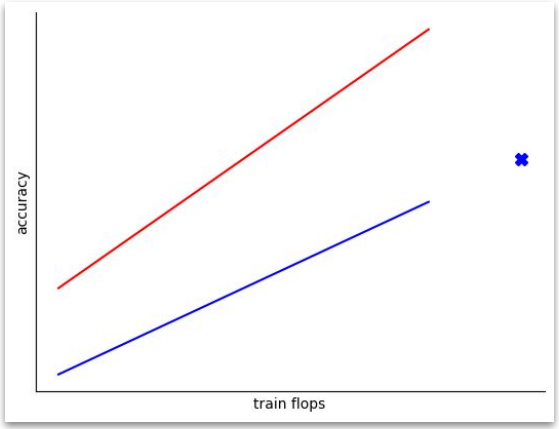
The underlying classification problem has $(X, Y) \sim$  where noisy $Y|X$ is still stochastic. One can imagine that for a logistic loss, the variance of the training loss from n examples derived from the dataset $(X, \mathbb{E}[Y|X])$ is smaller than that of the training loss from n examples of (X, Y). For a probabilistic binary model p_θ parameterized by θ:

$$\text{var}\ \frac{1}{n}\sum_i \log(1+\exp(-Y_i p_\theta(X_i))) \geq \text{var}\ \frac{1}{n}\sum_i \log(1+\exp(-\mathbb{E}[Y_i|X_i] p_\theta(x_i)))\ ,$$

05

Closing Thoughts

Scaling Trends and Inference Implications



Scaling Work has two flavors

1. Adding points to Quality x Model Size plot
2. Increasing the slope of the plot

Gemini tick-tock (Flash goal to match Pro of previous gen)

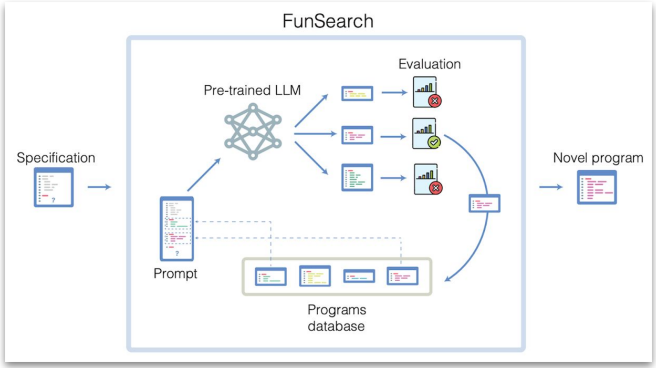
Inference Efficiency Work: Compression work grows with both scaling aspects

1. Development of better distillation recipes
2. Quantization
3. Serving-friendly model design changes

Future Pretrain Research Ideas – Without Big Costs!

Common refrain: pretraining is expensive, only can be researched in industry

- Developing hardware-focussed kernels is the hot-loop for research now. Kernel programming languages, compiler tools, developer tools that make this easier are crucial. Or come up with the next flash attention.
- Quantization entering a new frontier from vector quant
- Funsearch-style inference vs quality tradeoffs
  For LLM-in-the-loop for search.
- Scaling laws are brittle, dataset dependent.
  - L(N, D, etc.) – of course we can add more dims to improve fit.
    Least squares vs MLE & formal stats model
    Imply different scaling recommendations! Formalize.
  - Rather than grid (N, D) where do we get max info gain? Active learn...



Key Resources

- Scaling Laws for Neural Language Models, Kaplan et al., 2020, https://arxiv.org/abs/2001.08361
- Beyond Chinchilla-Optimal: Accounting for Inference in Language Model Scaling Laws, Sardana et al., 2024, https://arxiv.org/abs/2401.00448
- Scaling Data-Constrained Language Models, Muennighoff et al., 2023, https://arxiv.org/abs/2305.16264
- Distillation Scaling Laws, Busbridge et al., 2025, https://arxiv.org/abs/2502.08606
- How to Scale Your Model, Austin et al., https://jax-ml.github.io/scaling-book/
- Efficiently Scaling Transformer Inference, Pope et al., 2022, https://arxiv.org/abs/2211.05102
- Unified Scaling Laws for Routed Language Models, Clark et al., 2022, https://arxiv.org/abs/2202.01169