Training compute optimal Large Language Models

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch et al

January 10, 2025

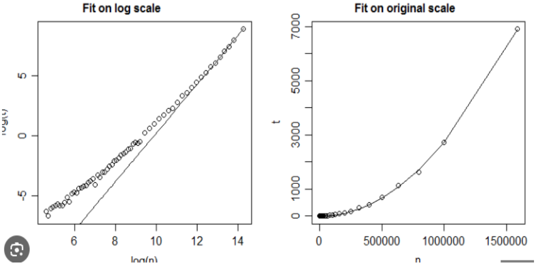
Problem

- FLOPs = floating point operations.
- **G**iven a fixed compute budget (number of FLOPs), how should the number of parameters and the number of training tokens scale relative to each other to have the most optimal model performance?

Previous work

- [1] showed that there is a power-law relationship between number of parameters in a model and performance.

Power-law relationship: $Y \propto X^d$



Previous work

- Misleading: Previous work [1] used a fix number of training tokens for all runs in their experiment.
- Status quo: [1] suggests that given additional compute budget, one should scale the number of parameters 5.5x and the number of tokens only 1.8x.
- Result: models are too large. Not enough training data to be optimal

To the point...

- The authors discover that given a fixed compute budget, it's most optimal to scale the number of parameters and the number of training tokens in equal portions.
- What that means: way less parameters, way more training data.

Novelties

- The authors discover novel scaling law for the number of parameters and the number of training tokens.
- Three approaches to discover scaling proportions
  - Fix model size and vary number of training tokens
  - Fix FLOP count and vary model size
  - Parametric estimation of an assumed functional form of the training loss using results from approach 1 & 2
- Smaller models significantly reduces inference costs
- Chinchilla model (70B parameters): using new scaling laws, the authors create a model that is more performant than the Gopher model (280B parameters) but with significantly less parameters.

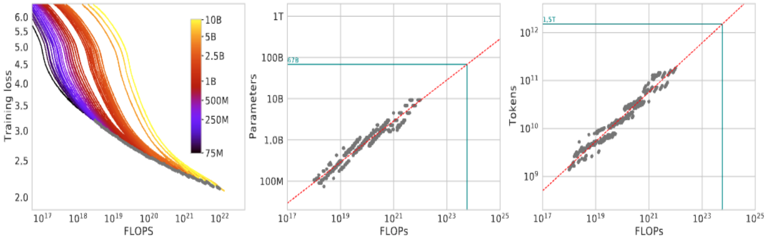
Optimal trade-off

Given a fixed compute budget, how do we trade-off model size and number of training tokens?

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } FLOPs(N,D)=C}{argmin} L(N, D)$$

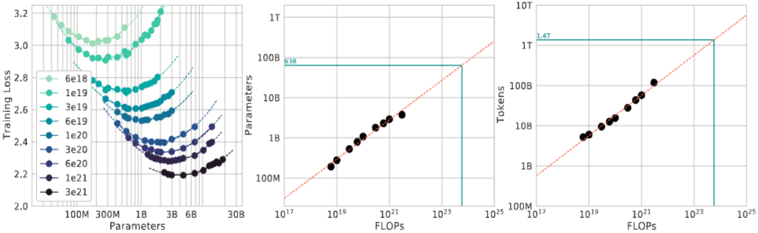
Estimating scaling laws: approach 1

- Fix model size and vary number of training tokens.
- Smooth and interpolate training loss curve for each model to obtain mapping from FLOP count to loss
- Fit a power-law estimator to the data points along the envelope to obtain optimal values for number of parameters and number of training tokens
- $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. $a = 0.5$ and $b = 0.5$ nvelope: think of it as the outer boundary formed by the most compute-efficient models.



Estimating scaling laws: approach 2

- Fix training FLOP count and vary model size.
- For each model size, adjust the number of tokens so the final FLOP count on the isoFLOP profile remains the same.
- Find the parameter count that results in the lowest loss for each isoFLOP curve
- Line color = isoFLOP profile.
- As in approach 1, they fit a power-law estimator to the data points with the lowest training loss for each isoFLOP profile
- $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. $a = 0.49$ and $b = 0.51$



Estimating scaling laws: Parametric estimation of loss

All final losses from approach 1 & 2 are modeled parametrically as a function of model parameter count and number of tokens seen.

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

- E is the lowest possible loss the model can achieve
- $\frac{A}{N^{\alpha}}$ Describes how the loss decreases as the model size N increases, with α controlling the rate of improvement.
- $\frac{B}{D^{\beta}}$ Describes how the loss decreases as the number of training tokens D increases, with β controlling the rate of improvement.

Estimating scaling proportions: Parametric estimation of loss

$$\min_{A,B,E,\alpha,\beta} \sum_{Runs\ i} Huber_\delta(log\,\hat{L}(N_i, D_i) - \log L_i)$$

$$Huber_\delta = \begin{cases} \frac{1}{2}r^2, & \text{for r} \leq \delta \\ \delta(\text{r} - \frac{1}{2}\delta), & \text{for r} > \delta \end{cases}, \text{ where } r = y_i - \hat{y}_i$$

- Predicted values come from $\hat{L}(N_i, D_i)$
- Observed values $\log L_i$ come from the experiment runs in approach 1 & 2
- Estimate parameters to understand loss behavior for N and D

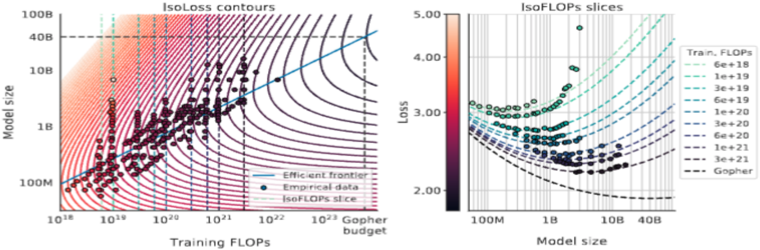
Find N_{opt} and D_{opt}

Apply compute constraint $C \propto NxD$ yields

$$N_{opt}(C) = G(\frac{C}{6})^a, \ D_{opt}(C) = G^{-1}(\frac{C}{6})^b, \text{ where } G = (\frac{\alpha A}{\beta B})^{\frac{1}{\alpha+\beta}} \quad (1)$$

$$a = \frac{\beta}{\alpha + \beta}, \text{ and } b = \frac{\alpha}{\alpha + \beta}$$

- From this, the authors get $a = 0.46$ and $b = 0.54$
- Plot shows conours of fitted function $\hat{L}$
- The blue line is a closed form efficient frontier using (1)



Findings

- All three approaches yield similar predictions for scaling proportions
- model size and amount of training data should be increased in equal proportions
- Suggests current LLMs are much too large considering the compute budgets

Table 2 | **Estimated parameter and data scaling with increased training compute.** The listed values are the exponents, a and b, on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Our analysis suggests a near equal scaling in parameters and data with increasing compute which is in clear contrast to previous work on the scaling of large models. The 10^{th} and 90^{th} percentiles are estimated via bootstrapping data (80% of the dataset is sampled 100 times) and are shown in parenthesis.

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

Chinchilla

- To test their scaling proportions, the authors use Gopher as a baseline and create a model called Chinchilla.
- Based on the FLOP count used to train Gopher, they leverage the scaling laws and pick 70B parameters and 1.4T tokens
- Gopher 240B parameters
- The model architecture and training is virtually the same as Gopher
- trained on MassiveText (same as Gopher)

Results

	# Tasks	Examples
Language Modelling	20	WikiText-103, The Pile: PG-19, arXiv, FreeLaw, ...
Reading Comprehension	3	RACE-m, RACE-h, LAMBADA
Question Answering	3	Natural Questions, TriviaQA, TruthfulQA
Common Sense	5	HellaSwag, Winogrande, PIQA, SIQA, BoolQ
MMLU	57	High School Chemistry, Astronomy, Clinical Knowledge, ...
BIG-bench	62	Causal Judgement, Epistemic Reasoning, Temporal Sequences, ...

Table 5 | **All evaluation tasks.** We evaluate *Chinchilla* on a collection of language modelling along with downstream tasks. We evaluate on largely the same tasks as in Rae et al. (2021), to allow for direct comparison.

Language modeling: bits-per-byte

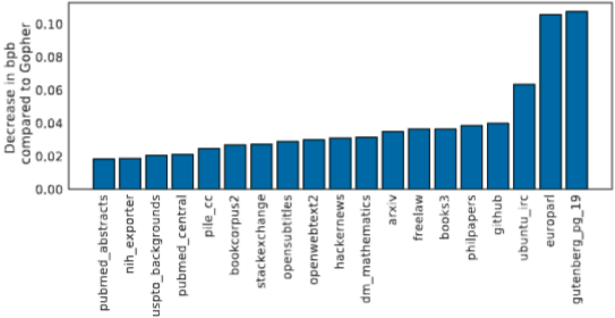


Figure 5 | **Pile Evaluation.** For the different evaluation sets in The Pile (Gao et al., 2020), we show the bits-per-byte (bpb) improvement (decrease) of *Chinchilla* compared to *Gopher*. On all subsets, *Chinchilla* outperforms *Gopher*.

MMLU: mulitple choice QA across 57 topics

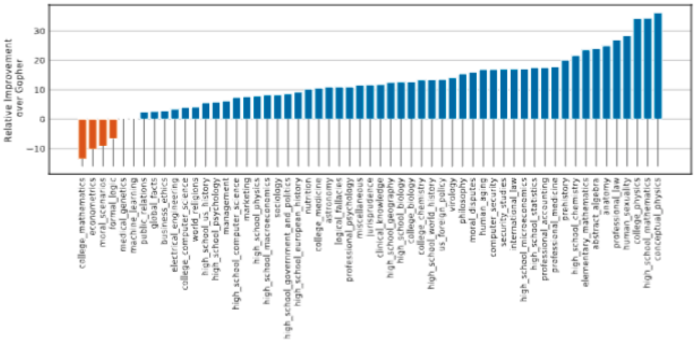


Figure 6 | **MMLU results compared to *Gopher*** We find that *Chinchilla* outperforms *Gopher* by 7.6% on average (see Table 6) in addition to performing better on 51/57 individual tasks, the same on 2/57, and worse on only 4/57 tasks.

Reading comprehension

	Chinchilla	*Gopher*	GPT-3	MT-NLG 530B
LAMBADA Zero-Shot	**77.4**	74.5	76.2	76.6
RACE-m Few-Shot	**86.8**	75.1	58.1	-
RACE-h Few-Shot	**82.3**	71.6	46.8	47.9

Table 7 | **Reading comprehension.** On RACE-h and RACE-m (Lai et al., 2017), *Chinchilla* considerably improves performance over *Gopher*. Note that GPT-3 and MT-NLG 530B use a different prompt format than we do on RACE-h/m, so results are not comparable to *Gopher* and *Chinchilla*. On LAMBADA (Paperno et al., 2016), *Chinchilla* outperforms both *Gopher* and MT-NLG 530B.

Big bench

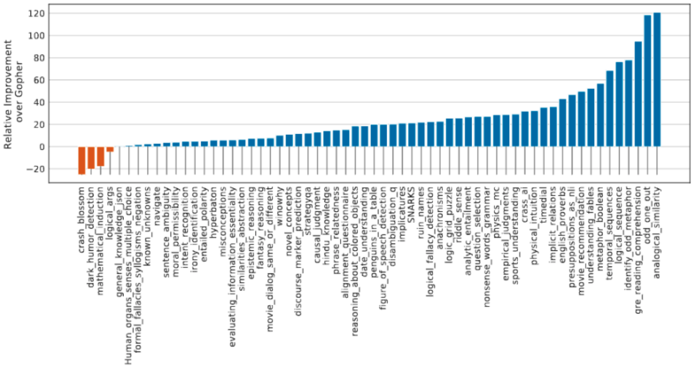


Figure 7 | **BIG-bench results compared to *Gopher*** *Chinchilla* out performs *Gopher* on all but four BIG-bench tasks considered. Full results are in Table A7.

Common sense

	Chinchilla	*Gopher*	GPT-3	MT-NLG 530B	Supervised SOTA
HellaSWAG	**80.8%**	79.2%	78.9%	80.2%	93.9%
PIQA	81.8%	81.8%	81.0%	**82.0%**	90.1%
Winogrande	**74.9%**	70.1%	70.2%	73.0%	91.3%
SIQA	**51.3%**	50.6%	-	-	83.2%
BoolQ	**83.7%**	79.3%	60.5%	78.2%	91.4%

Table 8 | **Zero-shot comparison on Common Sense benchmarks.** We show a comparison between *Chinchilla*, *Gopher*, and MT-NLG 530B on various Common Sense benchmarks. We see that *Chinchilla* matches or outperforms *Gopher* and GPT-3 on all tasks. On all but one *Chinchilla* outperforms the much larger MT-NLG 530B model.

Question answering

	Method	*Chinchilla*	*Gopher*	GPT-3	SOTA (open book)
Natural Questions (dev)	0-shot	16.6%	10.1%	14.6%	54.4%
	5-shot	31.5%	24.5%	-	
	64-shot	35.5%	28.2%	29.9%	
TriviaQA (unfiltered, test)	0-shot	67.0%	52.8%	64.3 %	-
	5-shot	73.2%	63.6%	-	
	64-shot	72.3%	61.3%	71.2%	
TriviaQA (filtered, dev)	0-shot	55.4%	43.5%	-	72.5%
	5-shot	64.1%	57.0%	-	
	64-shot	64.6%	57.2%	-	

Table 9 | **Closed-book question answering.** For Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017), *Chinchilla* outperforms *Gopher* in all cases. On Natural Questions, *Chinchilla* outperforms GPT-3. On TriviaQA we show results on two different evaluation sets to allow for comparison to GPT-3 and to open book SOTA (FiD + Distillation (Izacard and Grave, 2020)).

Gender bias

	Chinchilla	*Gopher*
All	78.3%	71.4%
Male	71.2%	68.0%
Female	79.6%	71.3%
Neutral	84.2%	75.0%

	Chinchilla	*Gopher*
Male *gotcha*	62.5%	59.2%
Male *not gotcha*	80.0%	76.7%
Female *gotcha*	76.7%	66.7%
Female *not gotcha*	82.5%	75.8%

Table 10 | **Winogender results. Left:** *Chinchilla* consistently resolves pronouns better than *Gopher*. **Right:** *Chinchilla* performs better on examples which contradict gender stereotypes (*gotcha* examples). However, difference in performance across groups suggests *Chinchilla* exhibits bias.

Bibliography I

[1] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.