Week 2 - Sep 9: Scaling laws: why foundation models are large

Catherine Cheng

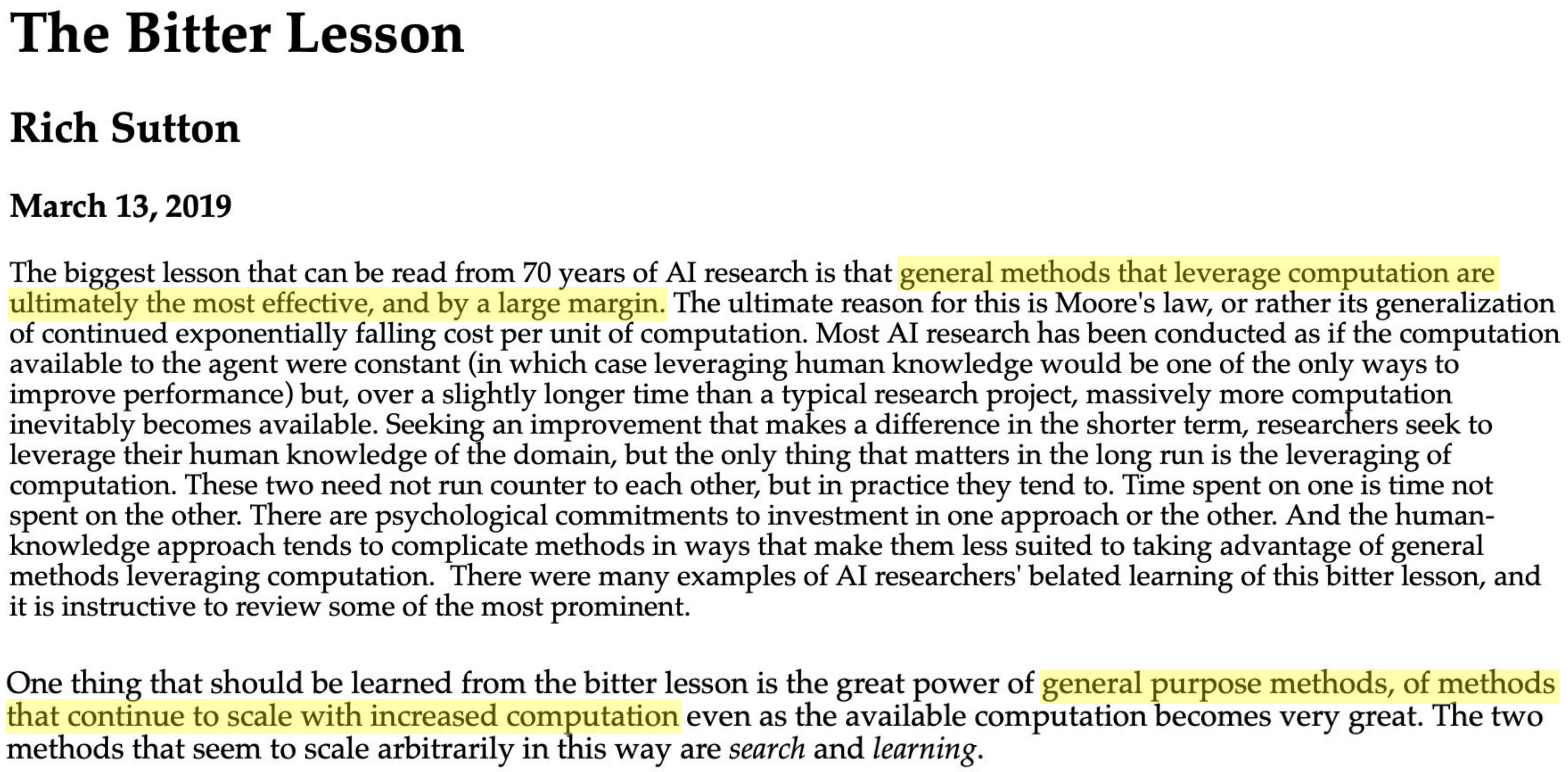
The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

(Slides adapted from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 4 by Sanjeev Arora)



The Bitter Lesson

Rich Sutton

March 13, 2019

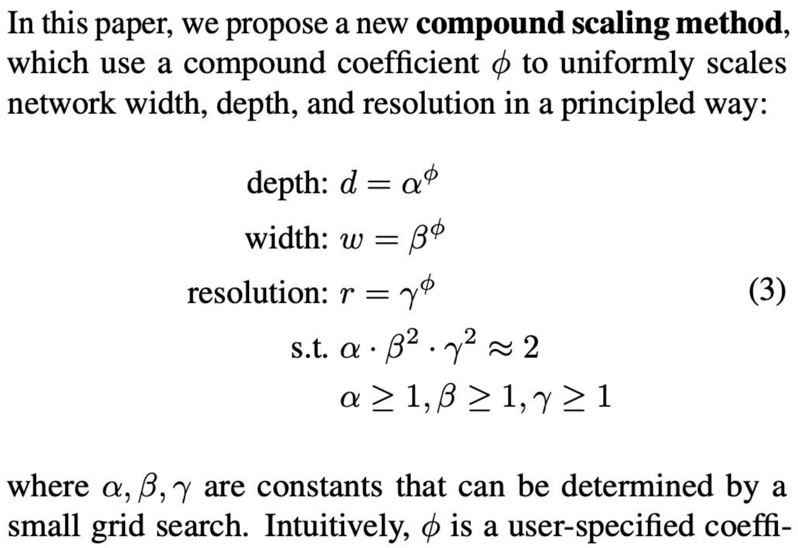
The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are *search* and *learning*.

(Slides adapted from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 4 by Sanjeev Arora)

What is scaling laws and why do we need them

- Scaling laws predict the model performance using a power-law when one of model size, dataset size, or compute budget varies
  - Early exploration of efficient (conv-)nets
    - Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International conference on machine learning. PMLR, 2019.



In this paper, we propose a new **compound scaling method**, which use a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way:

$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3}$$

where α, β, γ are constants that can be determined by a small grid search. Intuitively, ϕ is a user-specified coeffi-

(Slides adapted from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 4 by Sanjeev Arora)

What is scaling laws and why do we need them

- Scaling laws predict the model performance using a power-law when one of model size, dataset size, or compute budget varies
  - Early exploration of efficient (conv-)nets
  - What's a general recipe to scale model up for arbitrary tasks?
    - Transformers, autoregressive language modeling

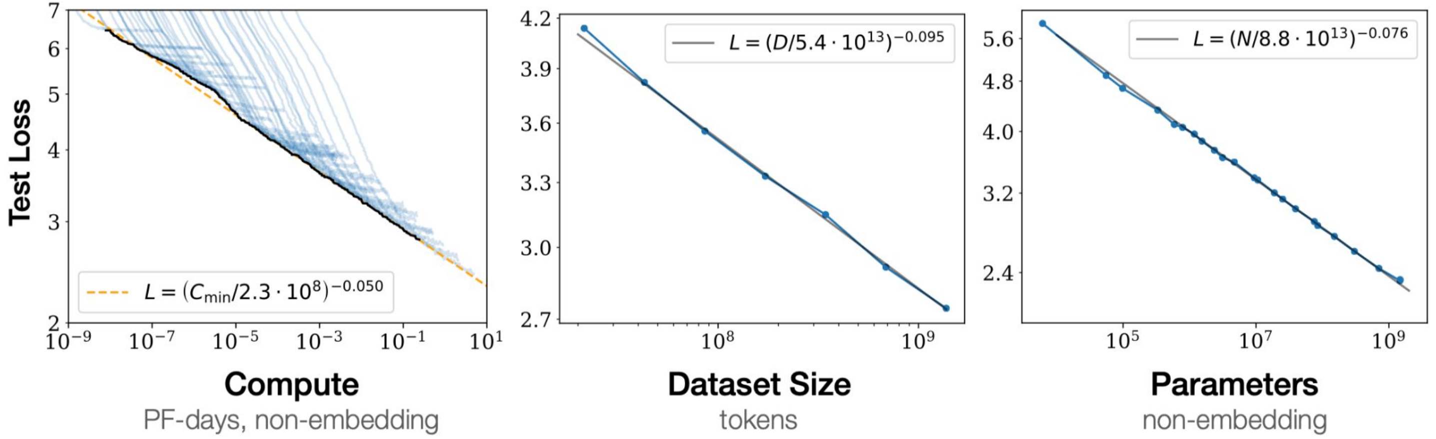
(Slides adapted from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 4 by Sanjeev Arora)

Overview

- Scaling Laws for Neural Language Models (Kaplan scaling law)
- Training Compute-Optimal Large Language Models (Chinchilla scaling law)
- Language Models are Few-Shot Learners (GPT3)

Kaplan Scaling Law (TL;DR)

- 10× increase in compute should be allocated to a 5.5× increase in model size and a 1.8× increase in training tokens
- large models should not be trained to their lowest possible loss to be compute optimal



(Slides adapted from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 4 by Sanjeev Arora)

Kaplan Scaling Law (TL;DR)

1. Performance depends strongly on scale, weakly on model shape
2. Smooth power laws
3. Universality of overfitting
4. Universality of training
5. Transfer improves with test performance
6. Sample efficiency
7. Convergence is inefficient
8. Optimal batch size

Kaplan Scaling Law (training setup)

- Model: decoder-only transformer
- Dataset: WebText
- BPE tokenizer, context length = 1024
- Compute for forward pass: $C_{\text{forward}} \approx 2N + 2n_{\text{layer}}n_{\text{ctx}}d_{\text{model}}$
- Estimated non-embedding compute per training token: $C \approx 6N$
- LR schedule with a 3000 step linear warmup followed by a cosine decay to 0
    - Convergence were largely independent of learning rate schedule

Kaplan Scaling Law (basic power law)

Varying factors:

1. Model size (from 768 to 1.5B non-embedding params)
2. Dataset size (from 22M to 23B tokens)
3. Shape (depth, width, attention heads, FFN dimension)

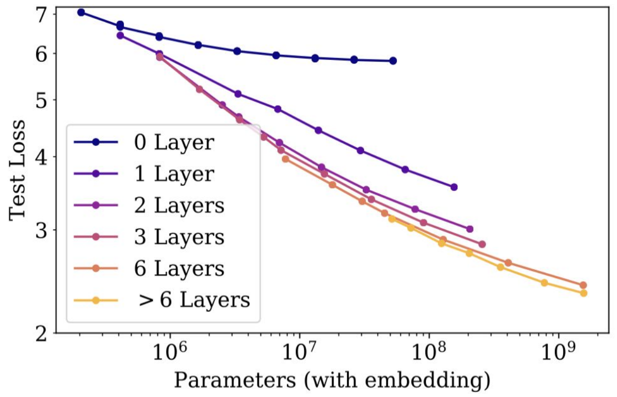
- Context length (mostly 1024)
- Batch size (mostly 2^19)

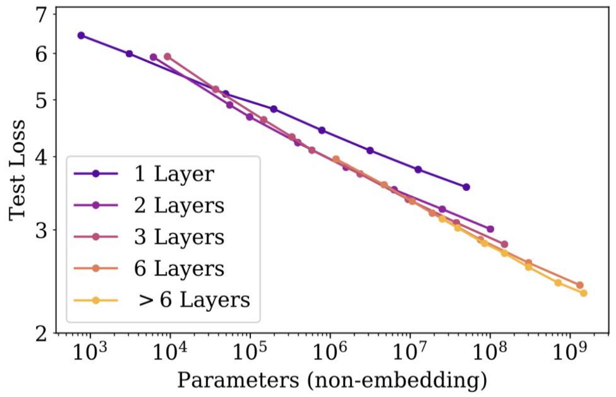
Kaplan Scaling Law (basic power law)

1. Model size

   Performance depends strongly on # layers when embedding parameters are included

   Performance converges to a single trend when embedding params are excluded

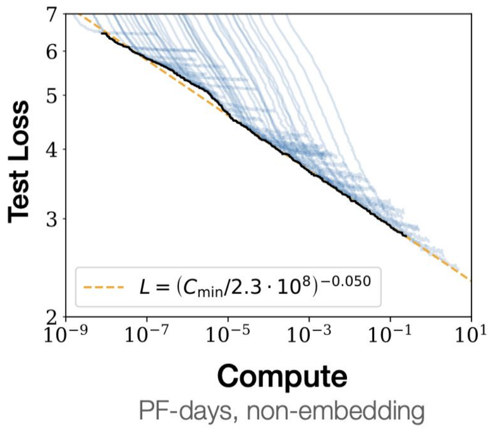




Kaplan Scaling Law (basic power law)

1. Model size

   Steady trend with non-embedding parameter count N, which can be fit into

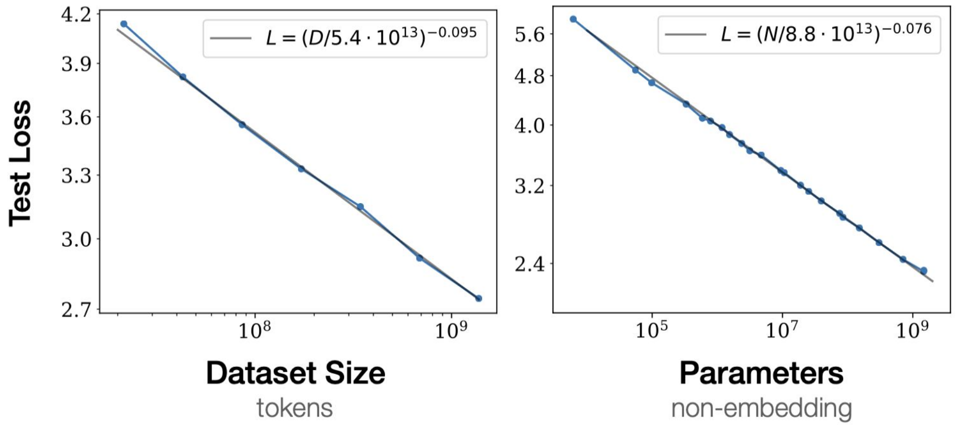


$$L(N) \approx \left(\frac{N_c}{N}\right)^{\alpha_N}$$

Kaplan Scaling Law (basic power law)

2. Dataset size and Compute

   Similar trend for dataset size D and compute C



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

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

Non-embedding compute is estimated as C=6NBS
Results in the plots have B (batch size) fixed so are not meant to be optimal

Kaplan Scaling Law (simultaneous dependence formulation)

- Proposed parameterization for model size and dataset size

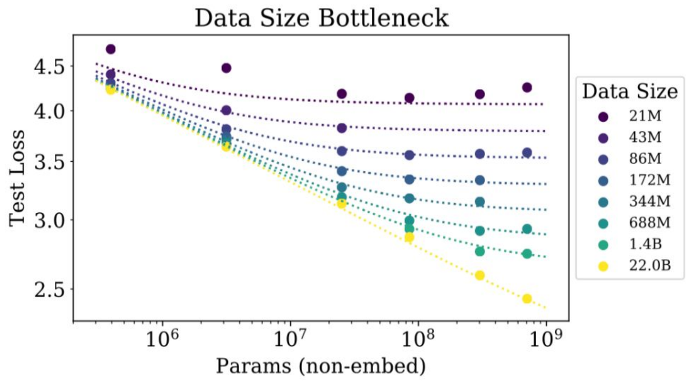
$$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$$

under 3 principles:

1. Changes in vocab size / tokenization rescale the loss by an overall factor
2. Approach L(N) with fixed N and D -> ∞ and approach L(D) with fixed D and N -> ∞
3. L(N,D) should be analytic at D = ∞ so that it has a series expansion in 1/Dwith integer powers (more speculative)

Kaplan Scaling Law (simultaneous dependence results)

- Left: For large D, performance is a straight power law in N
- Right: The extent of overfitting depends predominantly on the ratio $N^{\frac{\alpha_N}{\alpha_D}}/D$



We estimate that the variation in the loss with different random seeds is roughly 0.02, which means that to avoid overfitting when training to within that threshold of convergence we require

$$D \gtrsim (5 \times 10^3)\, N^{0.74} \tag{4.4}$$

Kaplan Scaling Law (simultaneous dependence formulation)

- There exists a critical batch size s.t. up to this batch size can be increased with very minimal degradation in compute-efficiency, which can be predicted using gradient noise scale, defined as

$$B_{\text{crit}}(L) \equiv \frac{E_{\min}}{S_{\min}}$$

Minimum # data to be processed

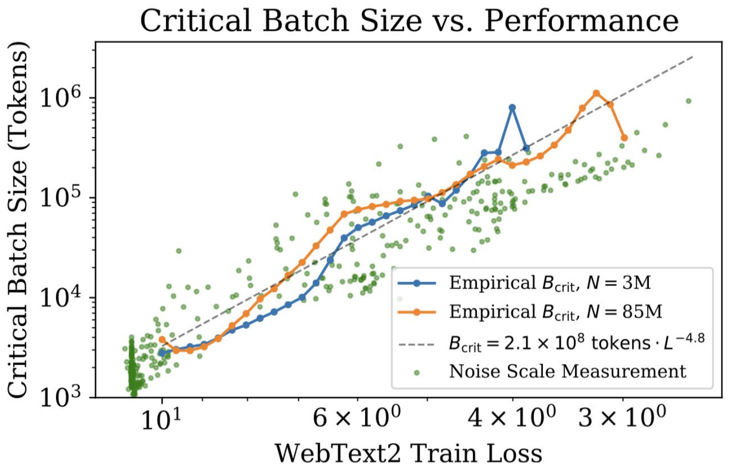
Minimum # training steps to achieve L

Kaplan Scaling Law (simultaneous dependence formulation)

- The critical batch size can be fit with a power-law in the loss

$$B_{\mathrm{crit}}(L) \approx \frac{B_*}{L^{1/\alpha_B}}$$

where $B_* \approx 2 \times 10^8$ and $\alpha_B \approx 0.21$.



Kaplan Scaling Law (simultaneous dependence formulation)

- Use the critical batch size to estimate the relation between # training steps with batch size 2^19

$$\left(\frac{S}{S_{\min}} - 1\right)\left(\frac{E}{E_{\min}} - 1\right) = 1$$

$$E = BS$$

$$B_{\text{crit}}(L) \equiv \frac{E_{\min}}{S_{\min}}$$

$$C = 6NBS$$

$$\longrightarrow$$

$$S_{\min}(S) \equiv \frac{S}{1 + B_{\text{crit}}(L)/B}$$

(minimum steps, at $B \gg B_{\text{crit}}$)

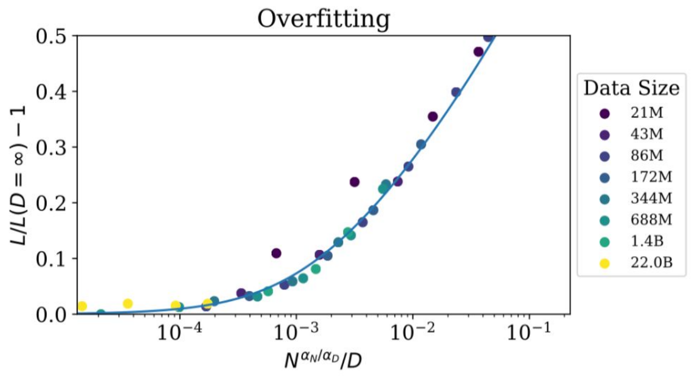
$$C_{\min}(C) \equiv \frac{C}{1 + B/B_{\text{crit}}(L)}$$

(minimum compute, at $B \ll B_{\text{crit}}$)

Kaplan Scaling Law (simultaneous dependence formulation)

- Fit to

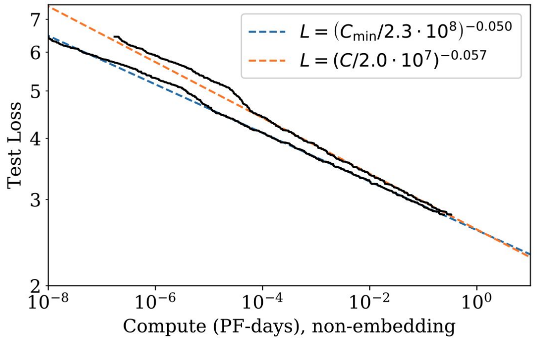
$$L(N, S_{\min}) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}}\right)^{\alpha_S}$$



Kaplan Scaling Law (simultaneous dependence formulation)

- More efficient training at the critical batch size

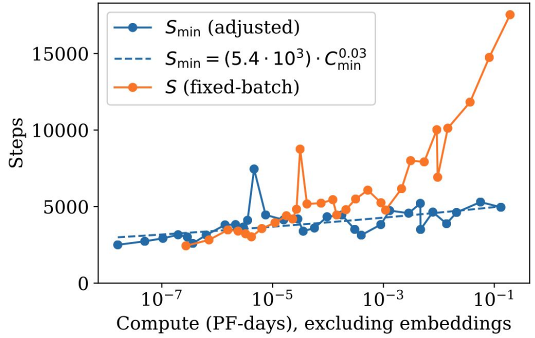
$$N(C_{\text{min}}) \propto (C_{\text{min}})^{0.73}$$



Kaplan Scaling Law (simultaneous dependence formulation)

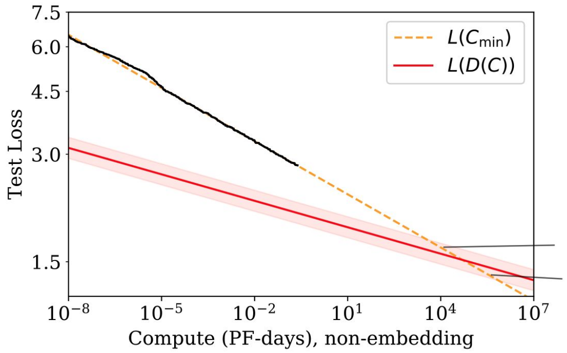
- The optimal number of steps will only grow very slowly with compute

$$N(C_{\min}) \propto (C_{\min})^{0.73} \longrightarrow S_{\min} \propto (C_{\min})^{0.03}$$



Kaplan Scaling Law (contradiction)

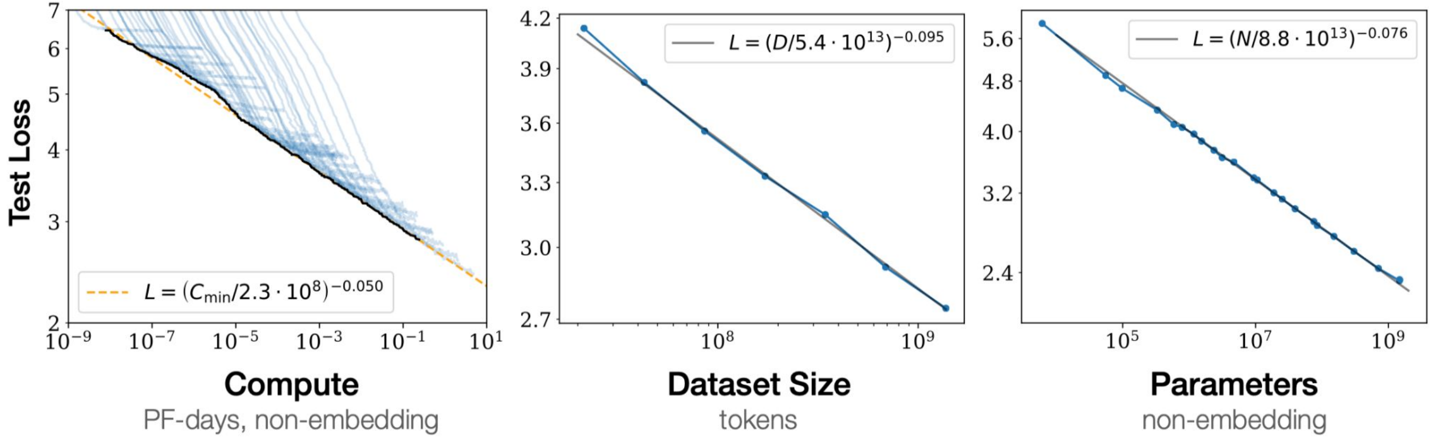
- At larger scales, the performance predicted by the L(C_min) scaling law decreases below what should be possible given the slow growth in training data with compute
  - Potential implication: an estimate of the point at which Transformer language models reach its maximal performance



The intersection point is sensitive to the precise power-law parameters

Recap: Kaplan Scaling Law

- 10× increase in compute should be allocated to a 5.5× increase in model size and a 1.8× increase in training tokens
- large models should not be trained to their lowest possible loss to be compute optimal



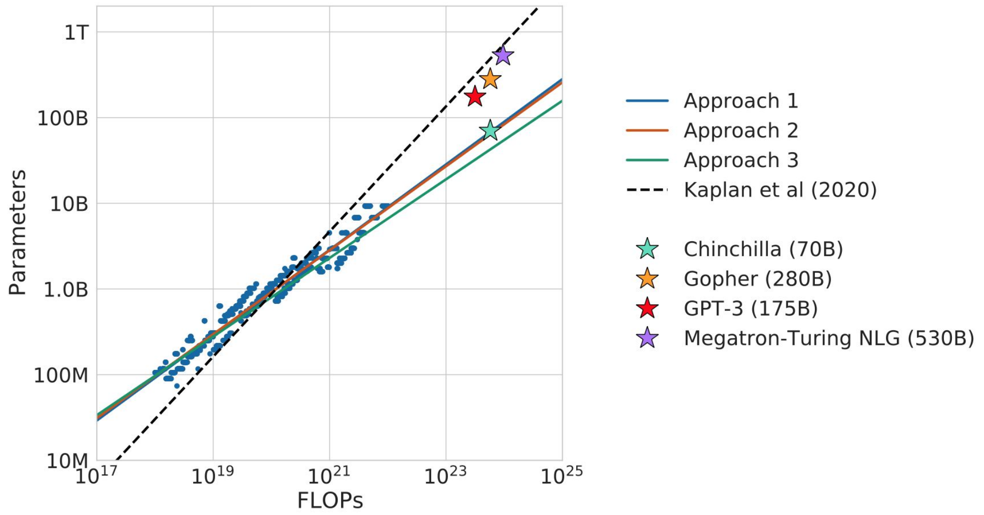
Chinchilla Scaling Law

- 10× increase in compute should be allocated to a ~~5.5×~~ increase in model size and a ~~1.8×~~ increase in training tokens
- large models should not be trained to their lowest possible loss to be compute optimal

“for compute-optimal training, the model size and the number of training tokens should be scaled equally”

Chinchilla Scaling Law (TL;DR)

- Large models should be substantially smaller and therefore trained much longer than is previously done according to the Kaplan scaling law, i.e. a lot of models are undertrained



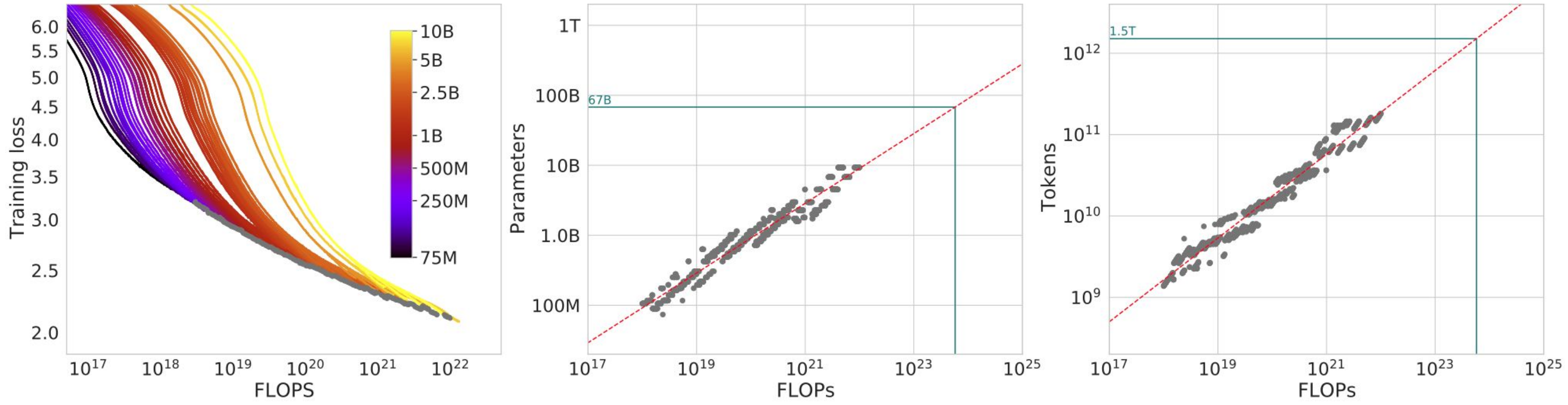
Chinchilla Scaling Law (TL;DR)

- *Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?*

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

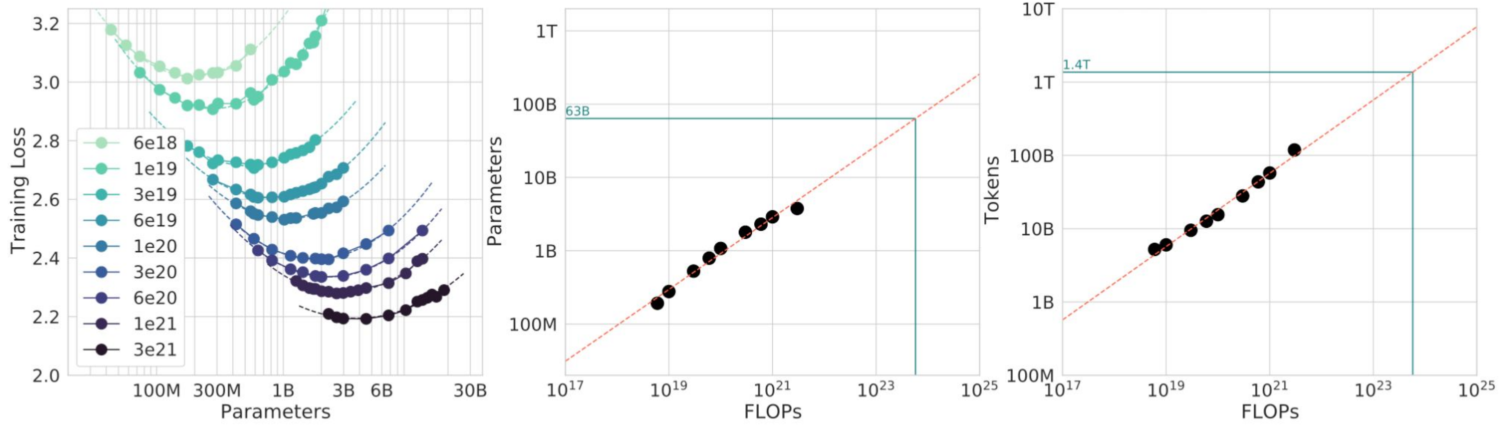
Chinchilla Scaling Law (Approach 1)

- Fix model sizes and vary number of training tokens
  - Train models (from 70M to > 10B params) for 4 different training sequences
  - Obtain a continuous mapping from FLOP count to training loss for each run
  - For each FLOP, determine the run that achieves the lowest loss
  - Find the model size that achieves the lowest loss with the required # training tokens
  - Fit power law to estimate the optimal model size and # training tokens for a given compute



Chinchilla Scaling Law (Approach 2)

- IsoFLOP profiles
  - Vary the model size for a fixed set of 9 different training FLOP counts (from 6*10^18 to 3*10^21 FLOPs) and monitor the final training loss
  - Plot the final loss against the param count for each FLOP budget to obtain the IsoFLOP curves
  - Fit a parabola to estimate the model size where the minimum loss is achieved
  - Fit a power law between FLOPs and model size and # training tokens



Chinchilla Scaling Law (Approach 3)

- Fitting a parametric loss function

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

  - E: Entropy of natural text
  - Second term: transformer of size N underperforms the ideal generative process
  - Last term: transformer is not trained to convergence due to finite optimization steps and dataset samples

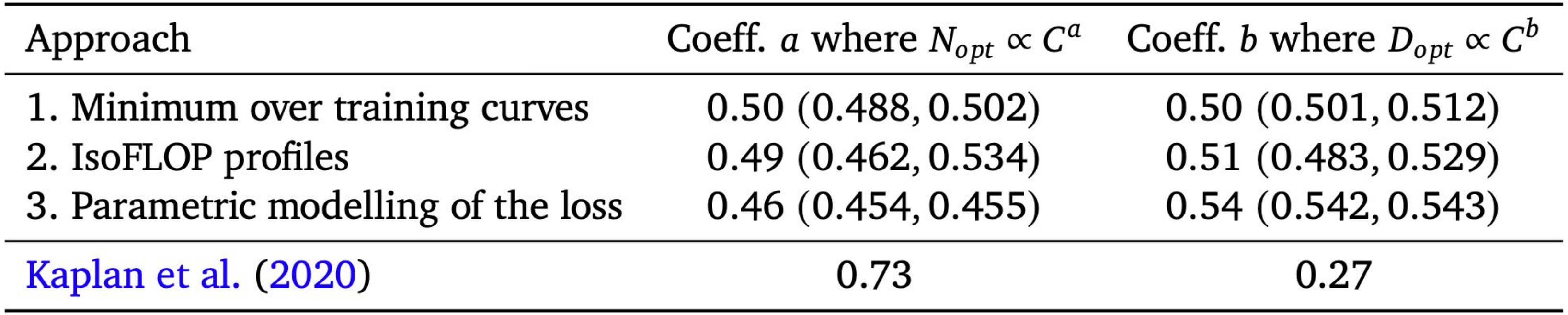
Chinchilla Scaling Law (Approach 3)

- Minimize the loss under the constraint $\text{FLOPs}(N, D) \approx 6ND$

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

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

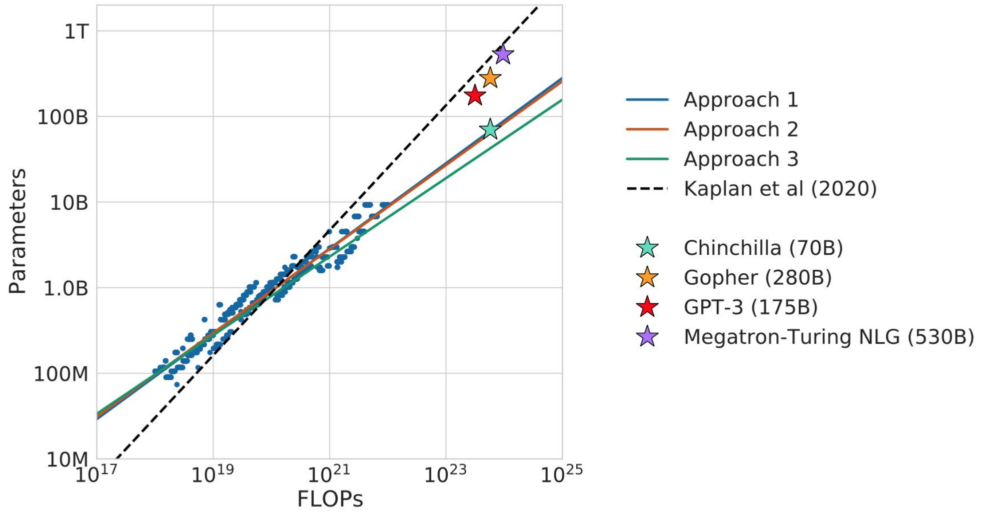
Chinchilla Scaling Law (Summary of estimations)



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 Scaling Law (Findings)

- Models (Gopher, GPT-3, Megatron-Turing NLG etc.) are largely over-sized / under-trained
- Smaller models should have been trained on more tokens to achieve the best performance

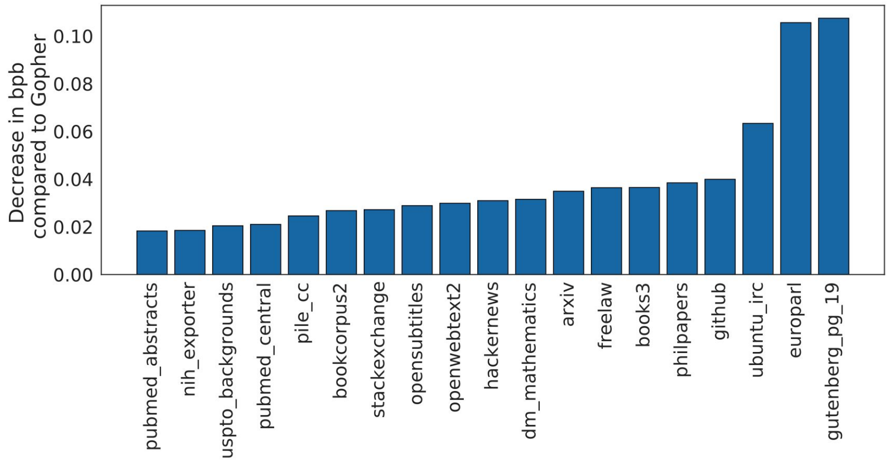


Chinchilla Scaling Law (Chinchilla, training)

- Same architecture and setup as Gopher except:
    - Use a slightly different subset distribution of MassiveText to account for the increased # training tokens
    - Use AdamW for better downstream performance
    - Use a slightly modified SentencePiece tokenizer and no NFKC normalization for better representation of math and chemistry
    - Forward and backward passes are computed in bfloat16 while storing a float32 copy of weights in the distributed optimizer states

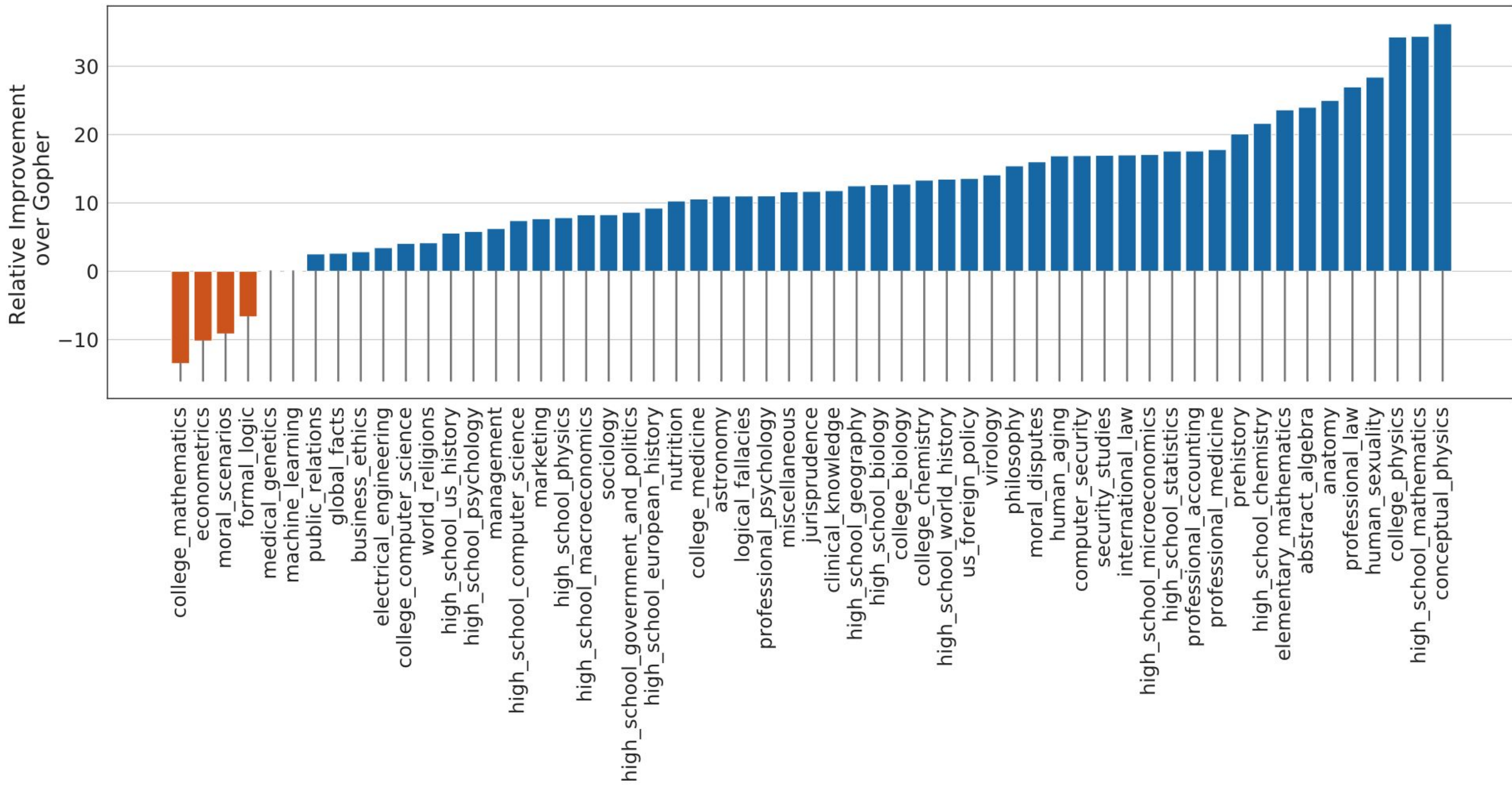
Chinchilla Scaling Law (Chinchilla, results)

- Language modeling
  - Show a consistent decrease in bits-per-byte (bpb) of Chinchilla compared to Gopher on all subsets of The Pile
  - Chinchilla is trained on 4×more data than Gopher and there may be data contamination



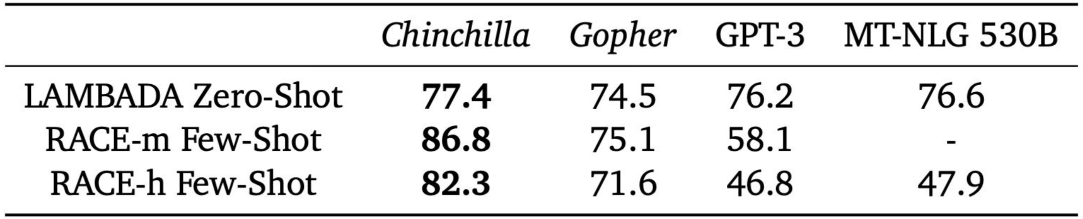
Chinchilla Scaling Law (Chinchilla, results)

- MMLU
  - Except on 4 tasks (college_mathematics, econometrics, moral_scenarios, and formal_logic), Chinchilla outperforms Gopher
- Similar trend on BIG-bench


Relative Improvement over Gopher
30
20
10
0
−10
college_mathematics
econometrics
moral_scenarios
formal_logic
medical_genetics
machine_learning
public_relations
global_facts
business_ethics
electrical_engineering
college_computer_science
world_religions
high_school_us_history
high_school_psychology
management
high_school_computer_science
marketing
high_school_physics
high_school_macroeconomics
sociology
high_school_government_and_politics
high_school_european_history
nutrition
college_medicine
astronomy
logical_fallacies
professional_psychology
miscellaneous
jurisprudence
clinical_knowledge
high_school_geography
high_school_biology
college_biology
college_chemistry
high_school_world_history
us_foreign_policy
virology
philosophy
moral_disputes
human_aging
computer_security
security_studies
international_law
high_school_microeconomics
high_school_statistics
professional_accounting
professional_medicine
prehistory
high_school_chemistry
elementary_mathematics
abstract_algebra
anatomy
professional_law
human_sexuality
college_physics
high_school_mathematics
conceptual_physics

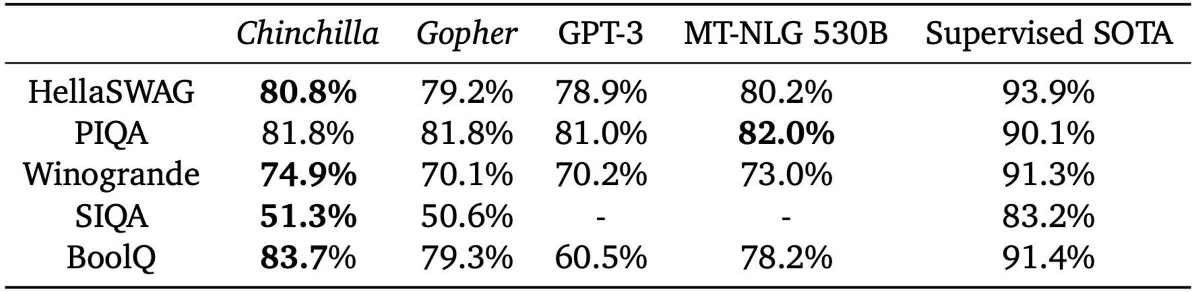
Chinchilla Scaling Law (Chinchilla, results)

- 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

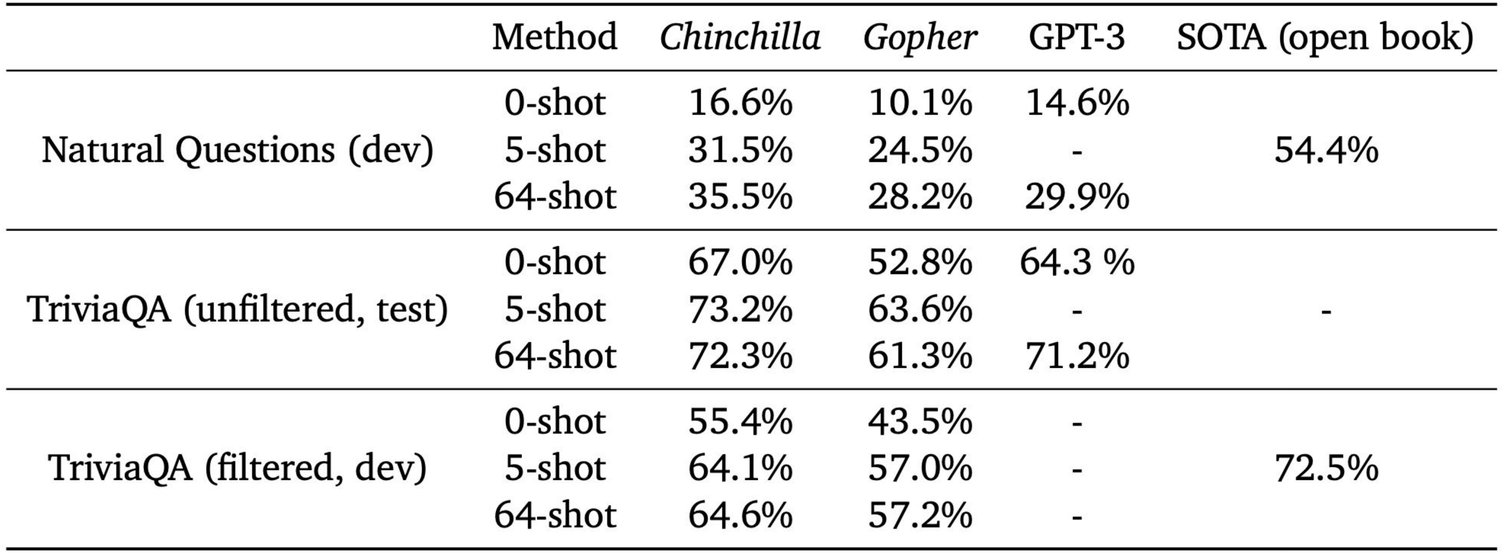
- Common sense: PIQA, SIQA, Winogrande, HellaSwag, BoolQ



	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%

Chinchilla Scaling Law (Chinchilla, results)

- Closed-book QA



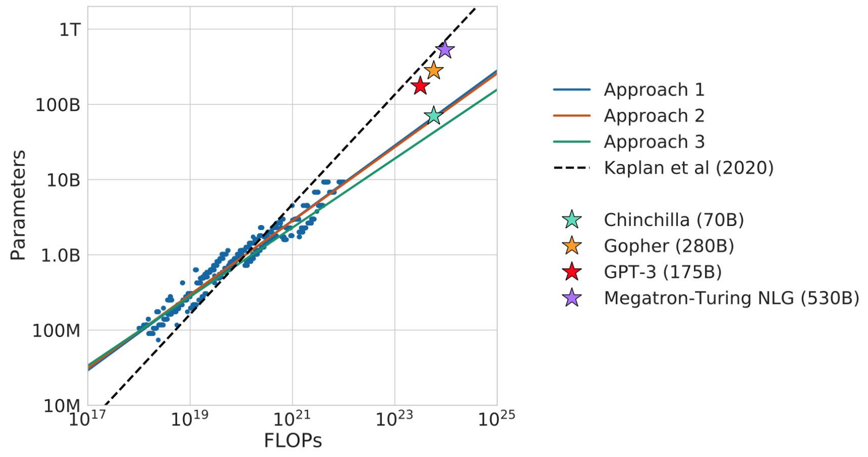
	Method	*Chinchilla*	*Gopher*	GPT-3	SOTA (open book)
	0-shot	16.6%	10.1%	14.6%	
Natural Questions (dev)	5-shot	31.5%	24.5%	-	54.4%
	64-shot	35.5%	28.2%	29.9%	
	0-shot	67.0%	52.8%	64.3 %	
TriviaQA (unfiltered, test)	5-shot	73.2%	63.6%	-	-
	64-shot	72.3%	61.3%	71.2%	
	0-shot	55.4%	43.5%	-	
TriviaQA (filtered,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

Chinchilla Scaling Law (Chinchilla, results)

- Gender bias and toxicity
  - Generally overcome gender stereotypes
  - Negligible effect on toxic text generation
  - Toxicity levels in unconditional text generation are largely independent of the model quality

Recap: Chinchilla Scaling Law

- For compute-optimal training, the model size and the number of training tokens should be *scaled equally*
- Large models should be substantially smaller and therefore trained much longer, i.e. a lot of models are undertrained



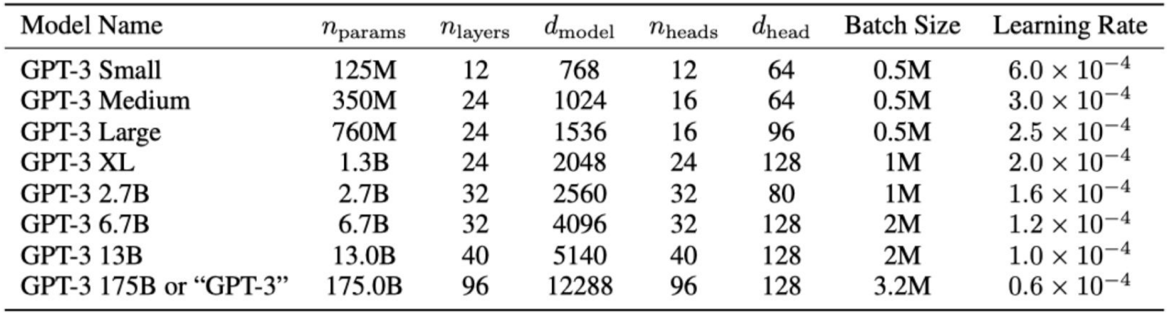
GPT-3 (TL;DR)

- An autoregressive language model of 175B parameters, 10x larger than any previous LMs
- Introduced the concept of “in-context learning”, and showed competitive performance

(Slides adapted from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 1 by Danqi Chen)

Overview of GPT-3

- GPT-3 is a Transformer decoder only trained on large amounts of unlabeled text
- All models were trained on 300B tokens

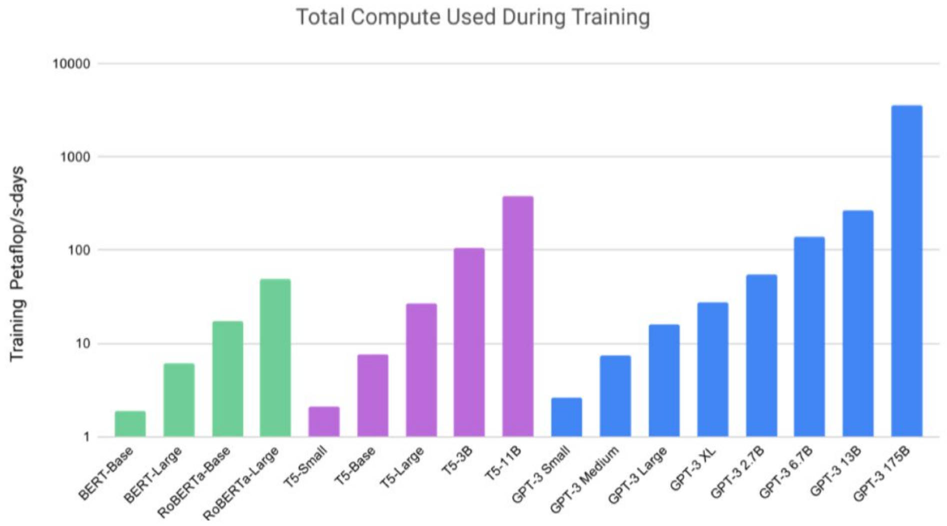


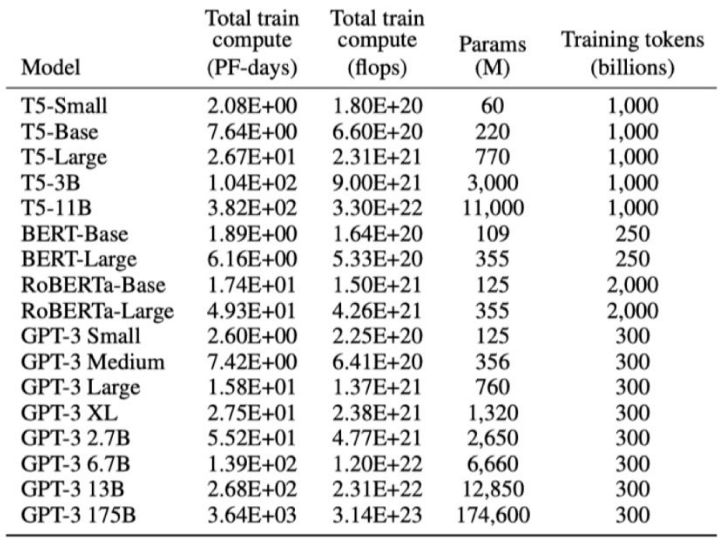
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

- Scaling laws (next week): "scaling of validation loss should be approximately a smooth power law as a function of size"
- Larger models typically use a larger batch size but require a smaller learning rate
- **Context window size = 2048**
- Use a lot of "model parallelism" during training
- Use Adam optimizer $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$

(Slides from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 1 by Danqi Chen)

GPT-3: training compute



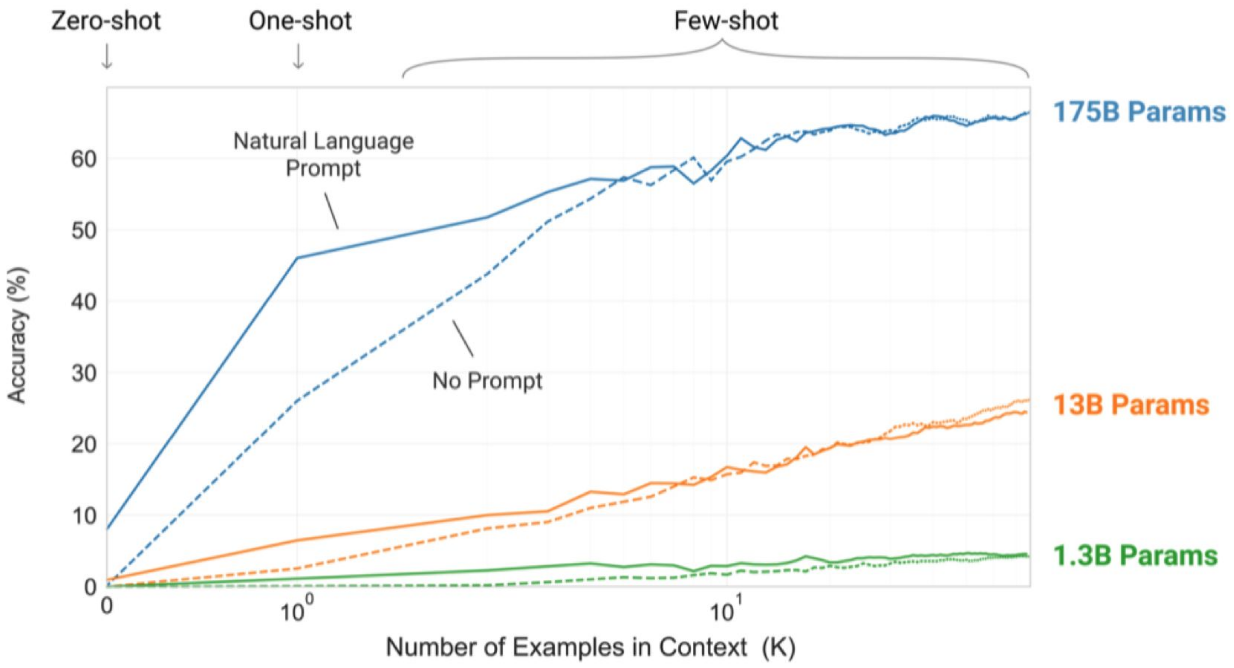


Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)
T5-Small	2.08E+00	1.80E+20	60	1,000
T5-Base	7.64E+00	6.60E+20	220	1,000
T5-Large	2.67E+01	2.31E+21	770	1,000
T5-3B	1.04E+02	9.00E+21	3,000	1,000
T5-11B	3.82E+02	3.30E+22	11,000	1,000
BERT-Base	1.89E+00	1.64E+20	109	250
BERT-Large	6.16E+00	5.33E+20	355	250
RoBERTa-Base	1.74E+01	1.50E+21	125	2,000
RoBERTa-Large	4.93E+01	4.26E+21	355	2,000
GPT-3 Small	2.60E+00	2.25E+20	125	300
GPT-3 Medium	7.42E+00	6.41E+20	356	300
GPT-3 Large	1.58E+01	1.37E+21	760	300
GPT-3 XL	2.75E+01	2.38E+21	1,320	300
GPT-3 2.7B	5.52E+01	4.77E+21	2,650	300
GPT-3 6.7B	1.39E+02	1.20E+22	6,660	300
GPT-3 13B	2.68E+02	2.31E+22	12,850	300
GPT-3 175B	3.64E+03	3.14E+23	174,600	300

“We train much larger models on many fewer tokens”

(Slides from COS 597R (Fall 2024): Deep Dive into Large Language Models Lecture 1 by Danqi Chen)

A summary of results


Zero-shot
One-shot
Few-shot
Natural Language Prompt
No Prompt
175B Params
13B Params
1.3B Params
Accuracy (%)
10^0
10^1
Number of Examples in Context (K)