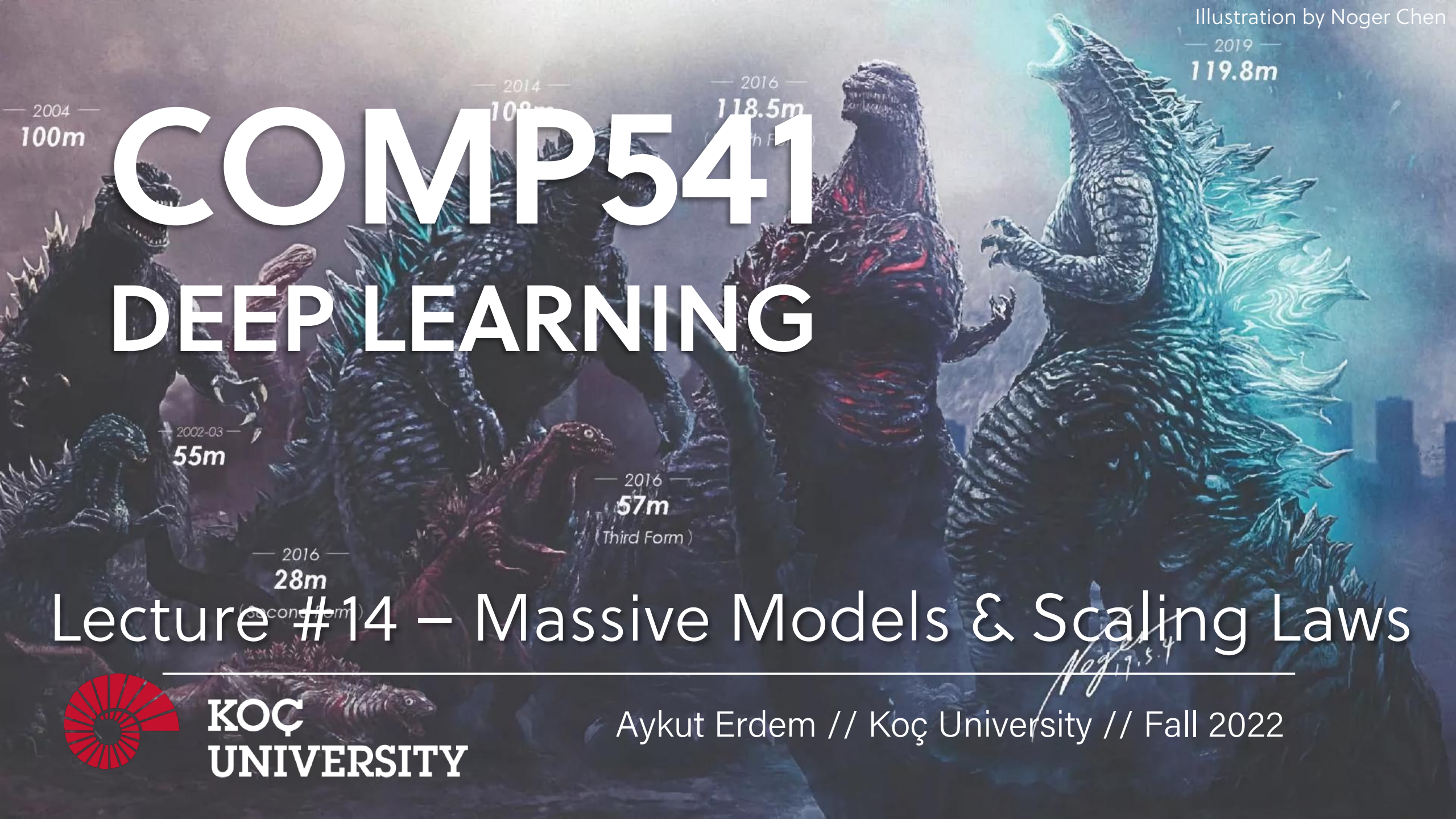


COMP541
DEEP LEARNING

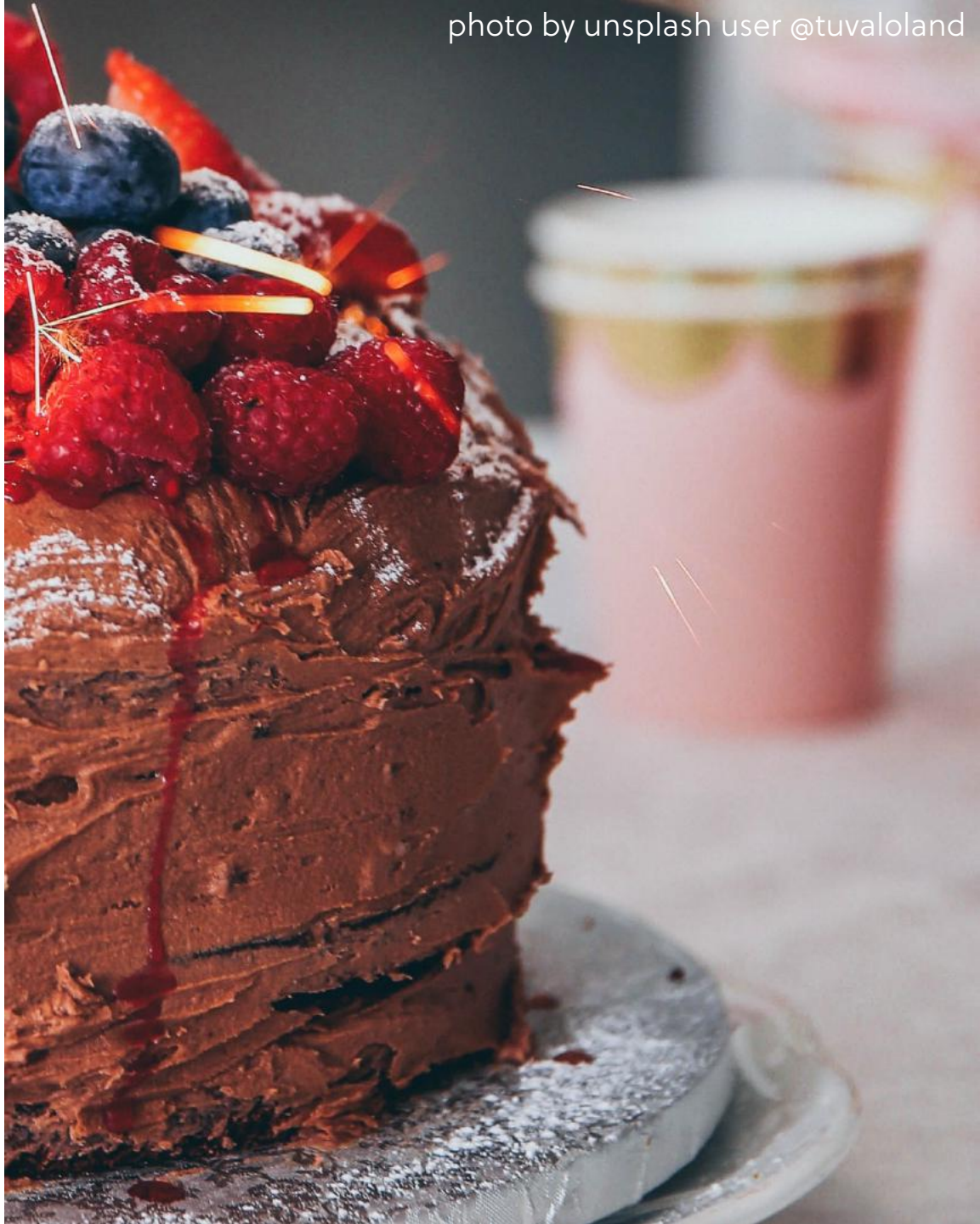
Lecture #14 – Massive Models & Scaling Laws

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Fall 2022

Previously on COMP541

- predictive / self-supervised learning
- self-supervised learning in NLP
- self-supervised learning in vision
- multimodal self-supervised learning



Lecture overview

- scaling laws
- massive text models
- applications of massive models

Disclaimer: Much of the material and slides for this lecture were borrowed from

—Marvin Zhang's Berkeley CS 182/282A class

—Stefanie Jegelka's MIT 6.S898 lecture on "Scaling Laws"

Lecture overview

- Today, we take a tour of the current landscape of massive models
- We will go over the high-level details of four of these models
- A natural question is whether moving in this direction is the right way to go; we will study this question theoretically, via **scaling laws**, as well as practically
- Finally, we will review some applications and current limitations of these models

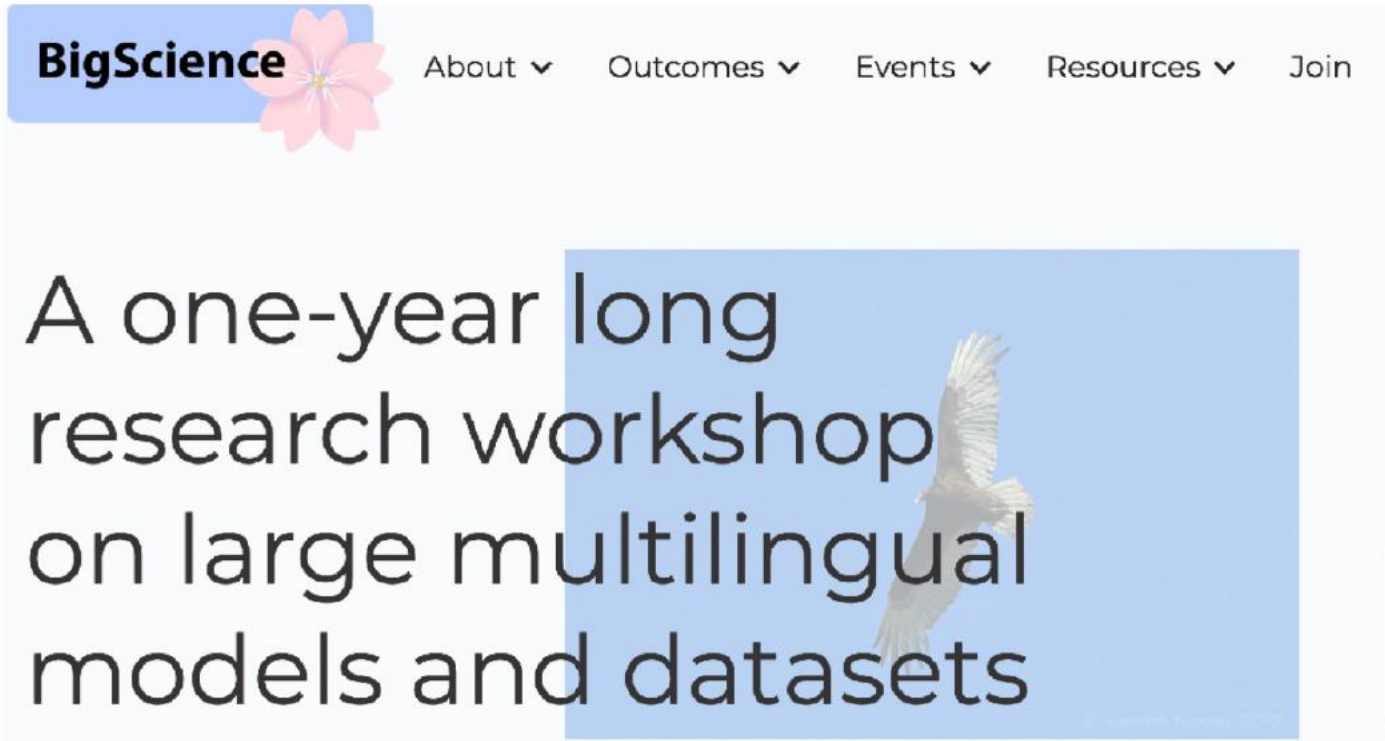
Scaling Laws

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

https://openai.com/blog/ai-and-compute/

If you had a given budget of compute, what model would you train on how much data?

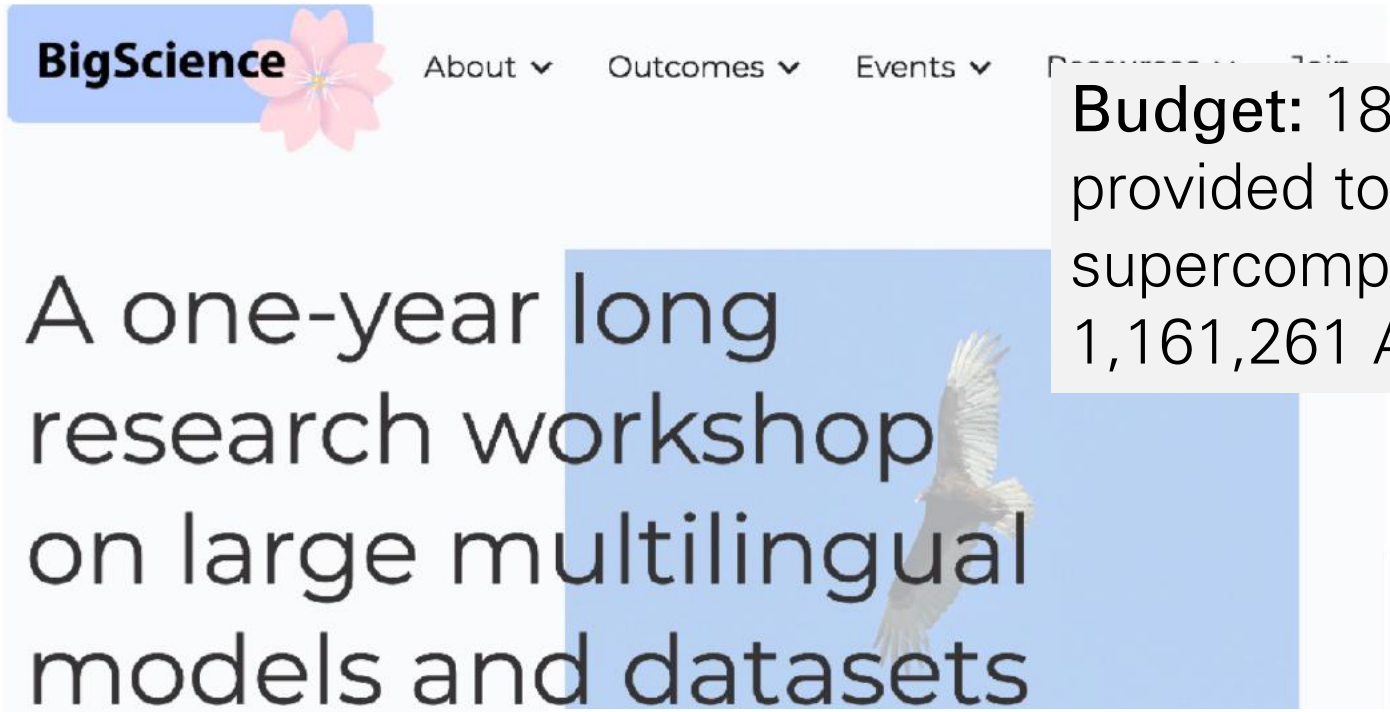
An Example



🌸Introducing The World's Largest Open Multilingual Language Model: BLOOM🌸

https://bigscience.huggingface.co/blog/bloom

An Example



Budget: 18 weeks on 384 A100 80GB provided to us by GENCI on the French supercomputer Jean Zay, that's 1,161,261 A100-hours

a BigScience initiative

BLOOM

What Language Model to Train if You Have One Million GPU Hours?

> After one year of experiments, discussions, and developments, we have started training our final BigScience model – a 176B multilingual language model. But how exactly did we decide on the final model size, shape, and pretraining duration?

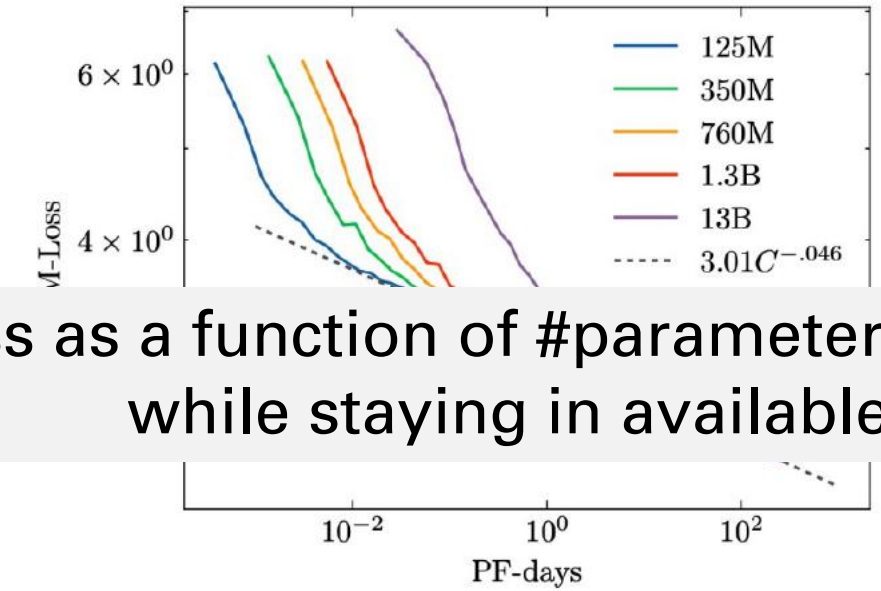
https://bigscience.huggingface.co/blog/bloom

What Language Model to Train if You Have One Million GPU Hours?

> After one year of experiments, discussions, and developments, we have started training our final BigScience model – a 176B multilingual language model. But how exactly did we decide on the final model size, shape, and pretraining duration?

1. Scaling laws

First, we derived scaling laws, giving us an upper-bound on the "optimal" model with our compute: that's ~392B parameters trained for ~165B tokens (more on that budget later). But scaling laws don't account for serving/inference costs, downstream task performance, etc. In addition, we need to make sure low resource languages still get enough tokens seen during pretraining. We don't want our model to have to zero-shot entire languages, do we? So, we decided that we should at least pretrain for 300-400B tokens.



Estimate loss as a function of #parameters, data, ... and minimize, while staying in available budget

What Language Model to Train if You Have One Million GPU Hours?

> After one year of experiments, discussions, and developments, we have started training our final BigScience model – a 176B multilingual language model. But how exactly did we decide on the final model size, shape, and pretraining duration?

1. Scaling laws

2. Compute

Then, we went back to our budget: 18 weeks on 384 A100 80GB provided to us by GENCI on the French supercomputer Jean Zay, that's 1,161,261 A100-hours! We estimated how many tokens would that allow us to train on for different model sizes, across wide "safety margins" to accommodate hurdles. We had a clear winner: a ~175B model gives us a good shot at perhaps even reaching a bit over 400B tokens.

What Language Model to Train if You Have One Million GPU Hours?

> After one year of experiments, discussions, and developments, we have started training our final BigScience model – a 176B multilingual language model. But how exactly did we decide on the final model size, shape, and pretraining duration?

1. Scaling laws

2. Compute

3. Taking shape

Now, to decide on a shape. Well first, we had a sneaky look at how other big 100B+ parameter models were shaped. We also did a bit of reading, and found some really cool work on how the shape of models should change with increased scale: specifically Kaplan et al. (2020) (a classic!), and Levine et al. (2020) (big models are too fit! make them chunkier!).

Model	Size [Bparams.]	Pretraining [Btokens]	Budget [PF-days]	Layers	Hidden dim.	Attention heads num.	Attention heads dim.
LaMDA (Thoppilan et al., 2022)	137	432	4,106	64	8,192	128	64

Optimize shape (by estimation)

What Language Model to Train if You Have One Million GPU Hours?

> After one year of experiments, discussions, and developments, we have started training our final BigScience model – a 176B multilingual language model. But how exactly did we decide on the final model size, shape, and pretraining duration?

1. Scaling laws

2. Compute

3. Taking shape

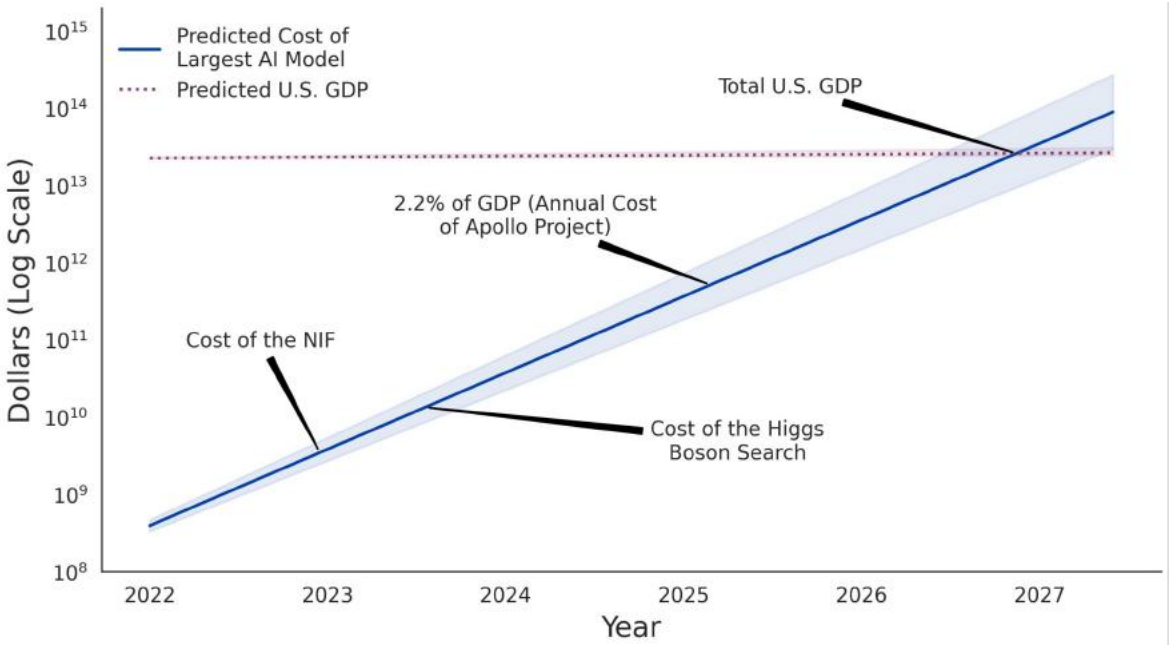
4. Speed matters

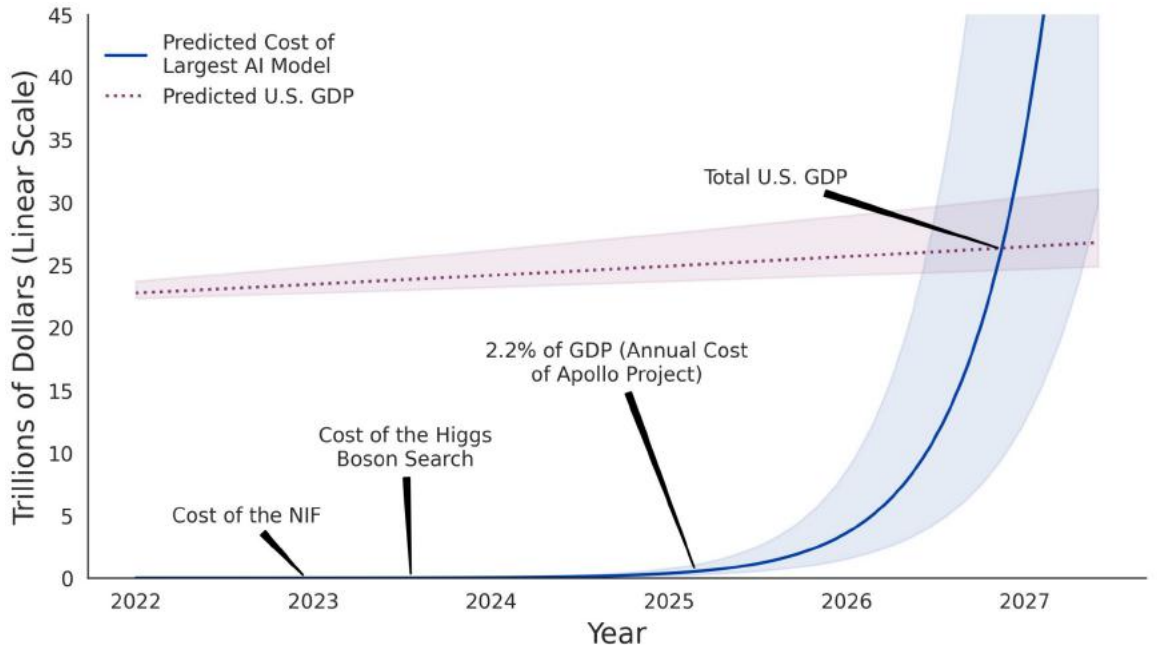
Optimize configuration by speed

Finally, Stas Bekman, BigScience's engineer extraordinaire, benchmarked hundreds of configs to find the fastest one. You can read more about it in his chronicles. It's all about finding the right set of magic numbers, avoiding effects such as tile/wave quantisation. We ended up with three final promising configs. We rejected (1) because of its large attention heads, and selected (3) as it was faster than (2). Speed matters: every extra bit of throughput means more total compute, thus more pretraining tokens, and a better final model.

Model	Size [params.]	Layers	Hidden dim.	Attention heads num.	Attention heads dim.	Memory [GB]	Performance [sec/iter.]	Performance [TFLOPs]
(1)	178	82		64	208	63	104	152

Economic infeasibility





https://cset.georgetown.edu/wp-content/uploads/AI-and-Compute-How-Much-Longer-Can-Computing-Power-Drive-Artificial-Intelligence-Progress_v2.pdf

Scaling laws

minimize *test-loss(model size, data, batch size, steps,...)* such that *compute* within budget

for a given architecture, how much data / what model size would be needed for a target performance?

can we extrapolate from smaller to larger models / experiments??

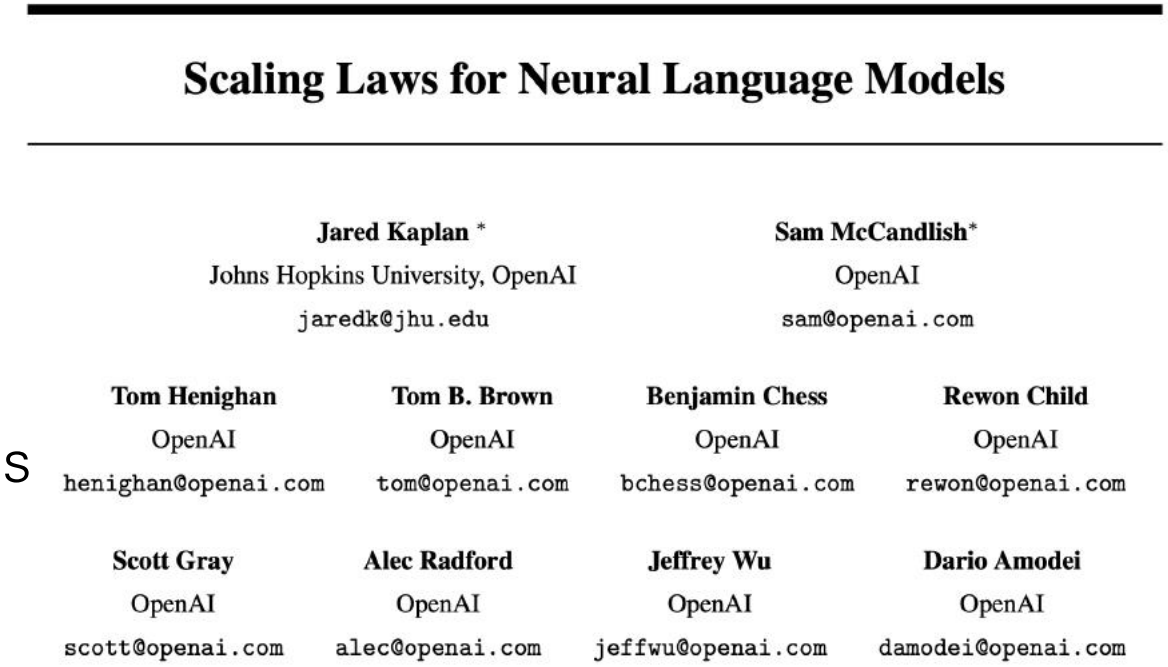
How does the test loss scale as a function of data, #parameters etc?

- autoregressive Transformer
- data: WebText2, 96GB of text, $2.29 \cdot 10^{10}$ tokens, 1024 token context

- vary:
  - model size: 768 - 1.5 billion non-embedding parameters
  - data: 22M - 23B tokens
  - shape (depth, with, attention heads, ...)
  - batch size

- compute = 6 x #parameters x batchsize x #steps

- computation: in PF-days (petaflop/s-days)
  1 PF-day = $8.64 \cdot 1019$ FLOPs



Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

Finding #1: power law relationships

$$L(X) = (1/X)^{\alpha} = X^{-\alpha}$$

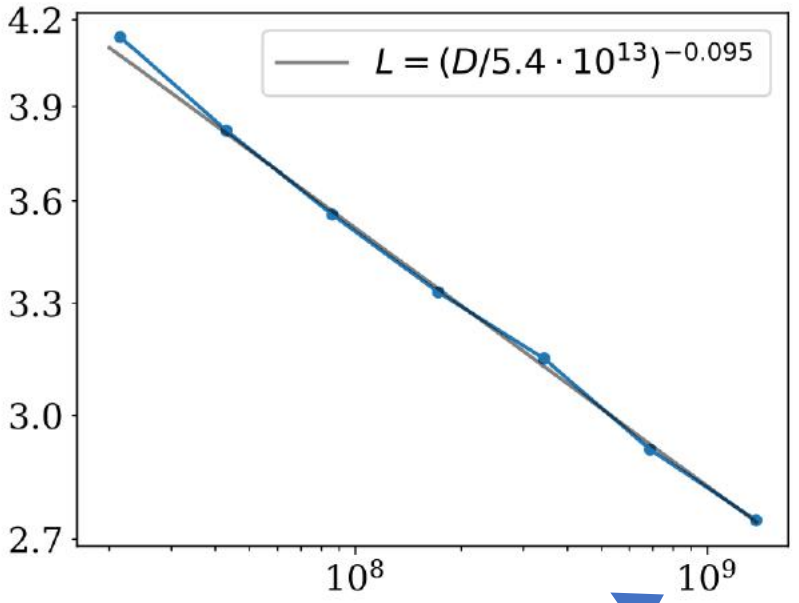
test loss

resource (data/parameters/ compute

- scale X → 2X => Loss → $2^{-\alpha}$ Loss

How does the test loss scale as a function of data?

- #parameters, compute: “infinite”



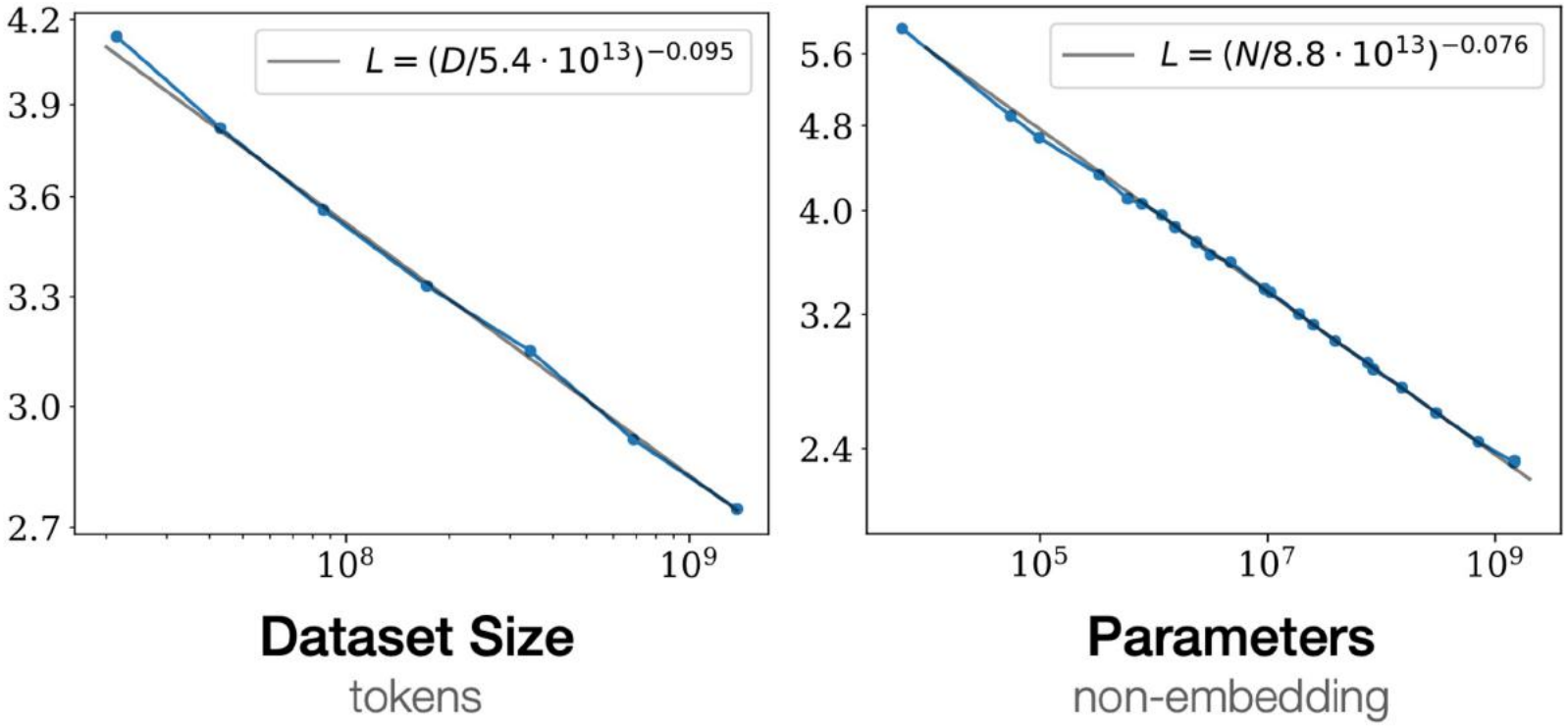
log scale

- power law relationship:

$$L(D) = (D_c/D)^{\alpha_D}\ ;\ \ \alpha_D \sim 0.095, \quad D_c \sim 5.4 \times 10^{13}\ \text{(tokens)}$$

How does the test loss scale as a function of #params?

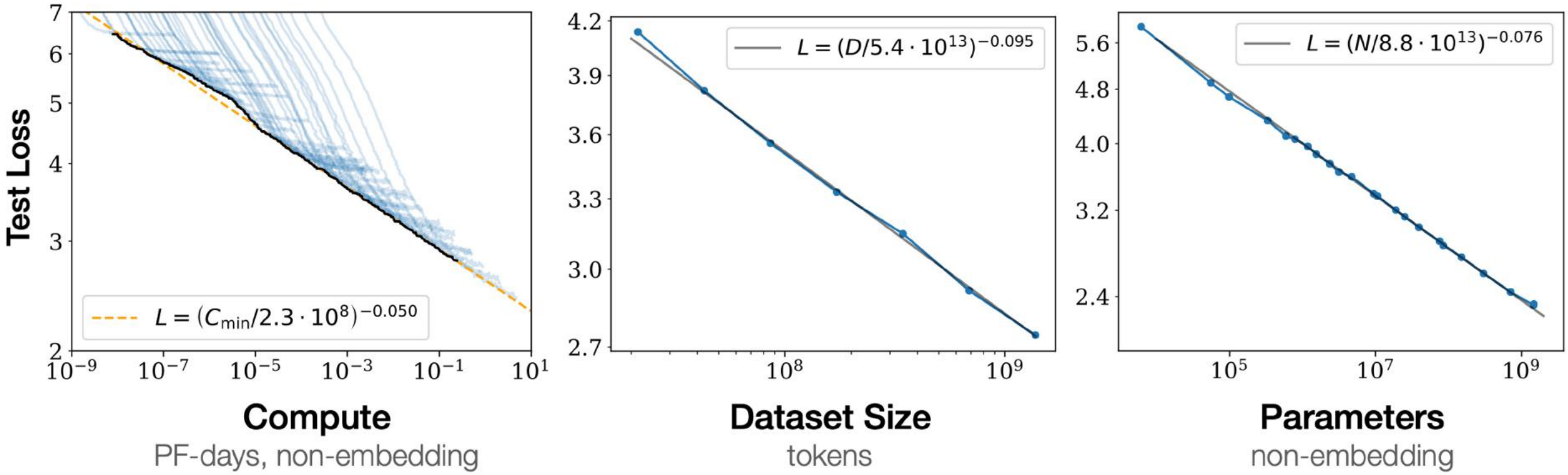
- #tokens, compute: “infinite”



$$L(N) = (N_c/N)^{\alpha_N}; \quad \alpha_N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13} \text{ (non-embedding parameters)}$$

How does the test loss scale with available compute?

- power law for all of them



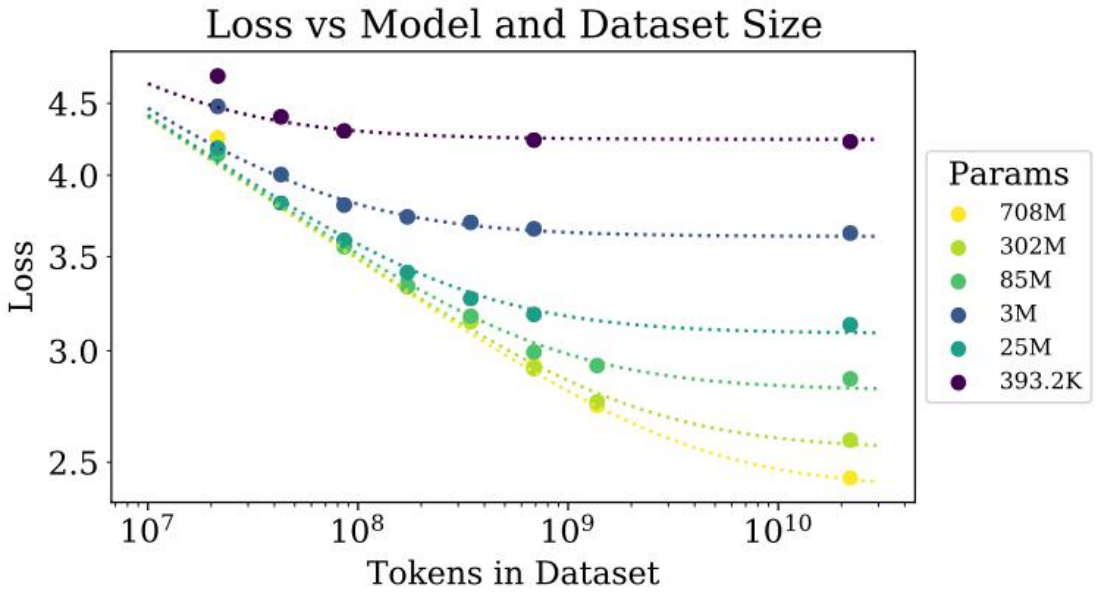
$$L(C_{\min}) = \left(C_{\mathrm{c}}^{\min}/C_{\min}\right)^{\alpha_C^{\min}}; \quad \alpha_C^{\min} \sim 0.050, \quad C_{\mathrm{c}}^{\min} \sim 3.1 \times 10^8 \text{ (PF-days)}$$

Finding #2: simultaneous dependence on multiple factors

- "Performance improves predictably as long as we scale up N and D in tandem, but enters a regime of diminishing returns if either N or D is held fixed while the other increases."
- Loss as function of #parameters N & data D:

constants

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

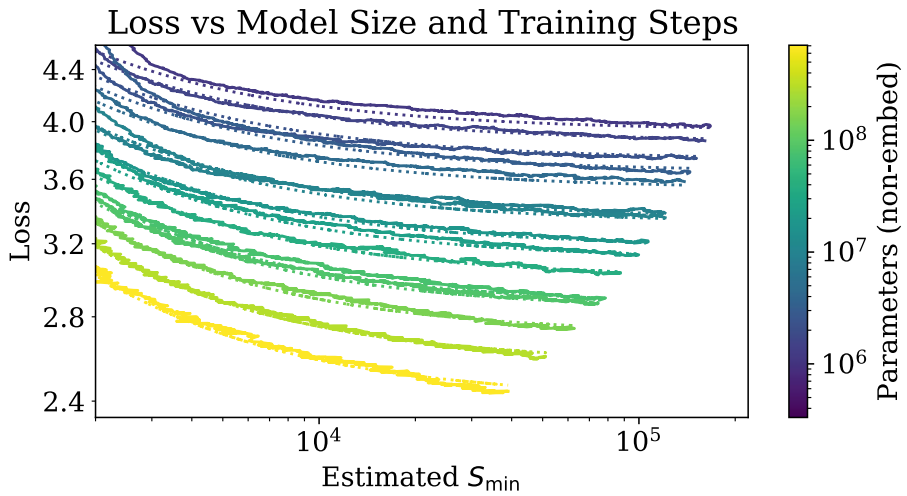


Parameter	α_N	α_D	N_c	D_c
Value	0.076	0.103	6.4×10^{13}	1.8×10^{13}

Finding #2: simultaneous dependence on multiple factors

- "Performance improves predictably as long as we scale up N and D in tandem, but enters a regime of diminishing returns if either N or D is held fixed while the other increases."
- Loss as function of #parameters N & data D:
- Loss as function of #parameters N & #steps S:

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



constants

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

Finding #2: simultaneous dependence on multiple factors

- "Performance improves predictably as long as we scale up N and D in tandem, but enters a regime of diminishing returns if either N or D is held fixed while the other increases."
- Loss as function of #parameters N & data D:
- Loss as function of #parameters N & #steps S:

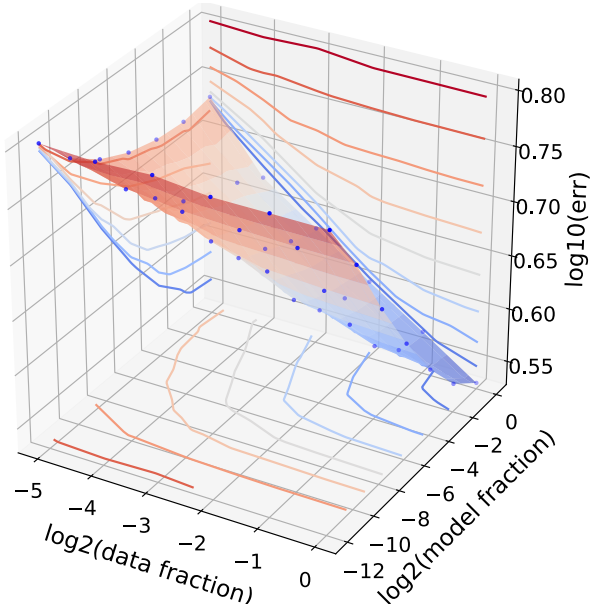
$$L(N, D) = \left[\left(\frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right]^{\alpha_D}$$

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

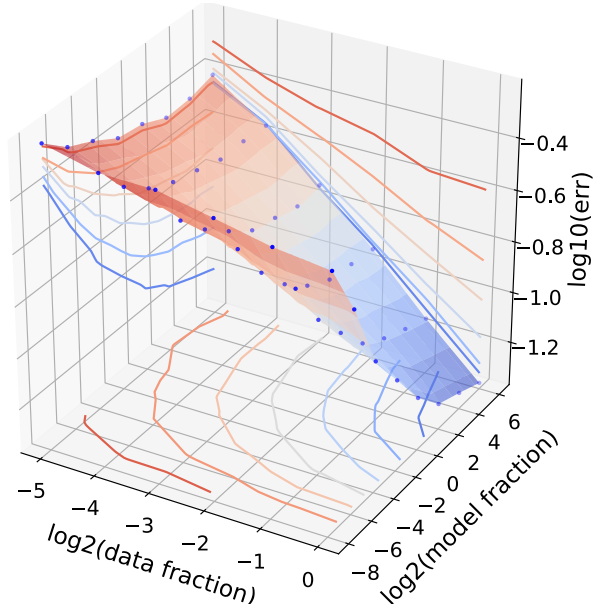
- from this, they conclude to scale, with compute C:
  #params $\propto C^{0.73}$, data = batch-size · #steps $\propto C^{0.27}$

Take these exact equations/numbers with a grain of salt!

Similar findings in other works



(a) Wiki103 error (cross entropy) landscape.



(b) CIFAR10 error (top1) landscape.

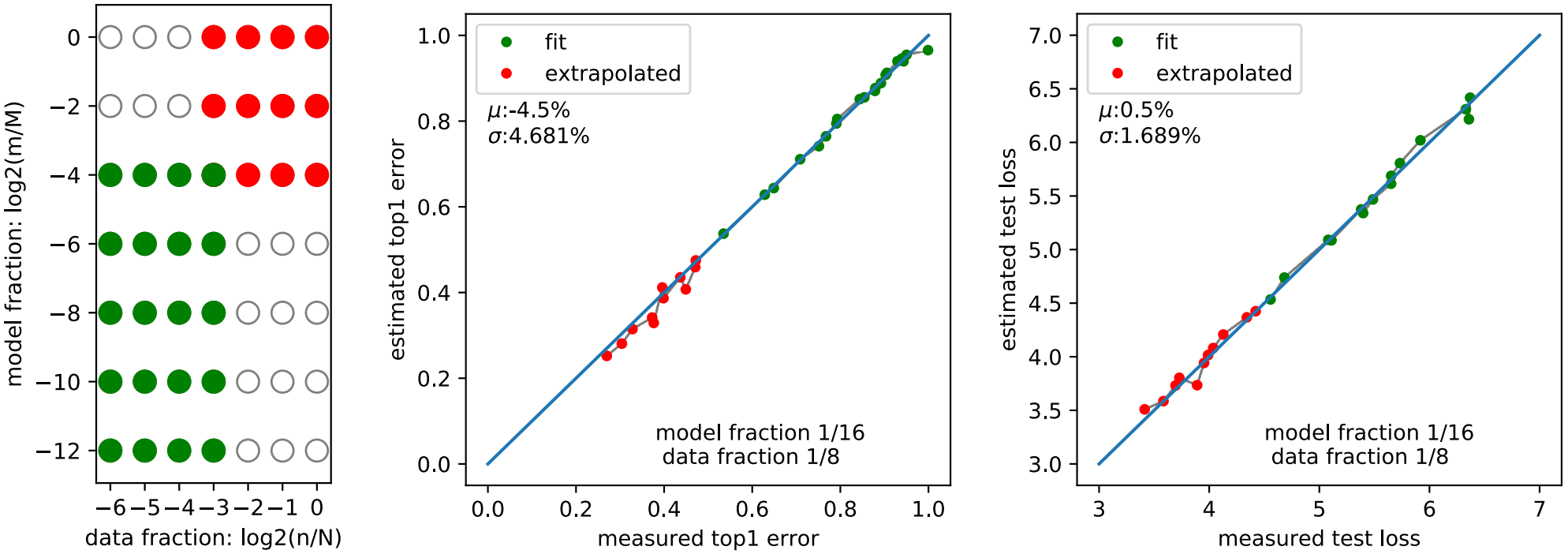
- Rosenfeld et al:
  for vision, language, fit function

$$L(N, D) \approx \frac{a(N)}{D^{\alpha_1(N)}} + \frac{b(N)}{N^{-\alpha_2(D)}} + c_\infty$$

Rosenfeld et al, 2020. A constructive prediction of the generalization error across scales

Extrapolation is possible!

- From Rosenfeld et al.

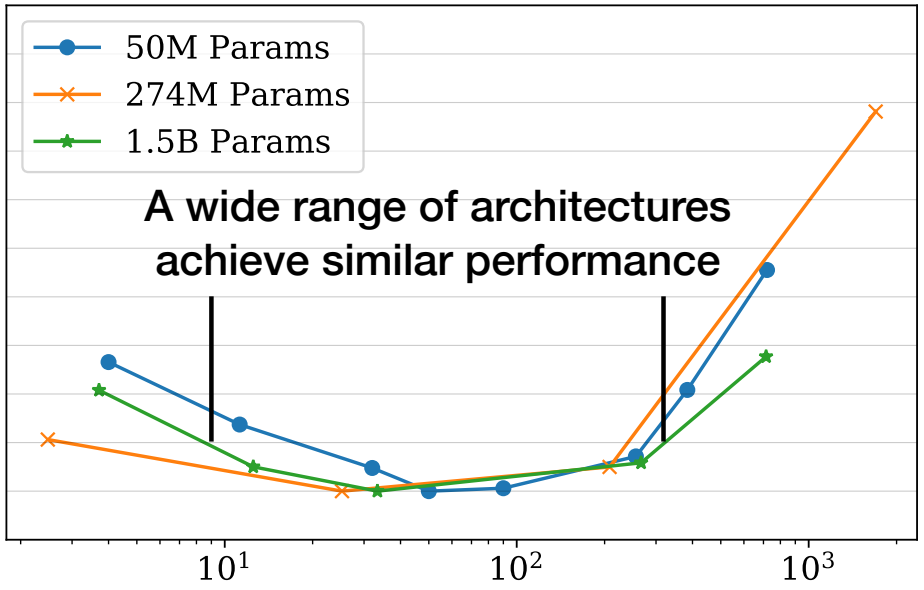


(a) Illustration. (b) Extrapolation on ImageNet (c) Extrapolation on WikiText-103.

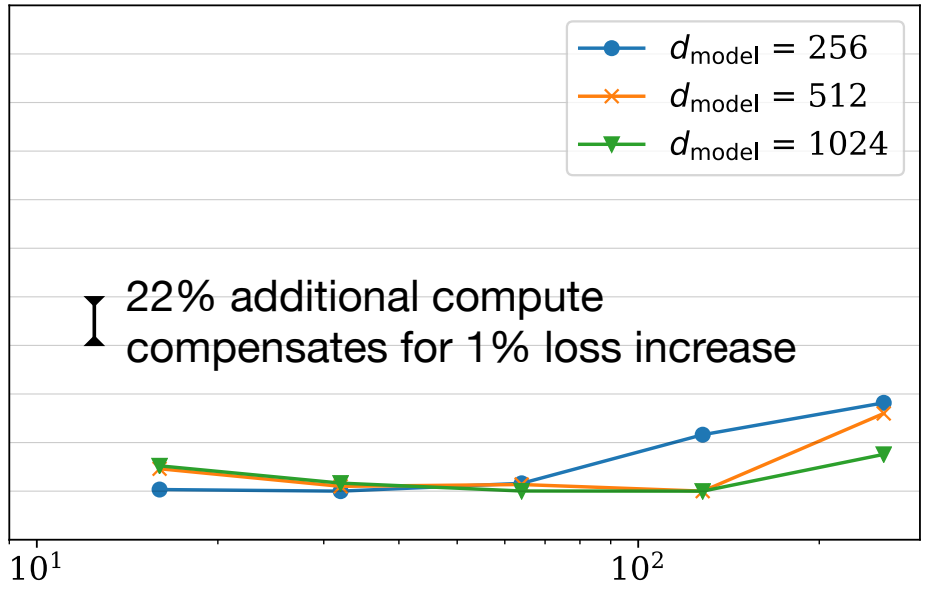
Rosenfeld et al, 2020. A constructive prediction of the generalization error across scales

Finding #3: dependence on model shape

- "Performance depends strongly on scale, weakly on model shape"



Aspect Ratio (d_{model} / n_{layer})



Attention Head Dimension (d_{model} / n_{head})

25M Parameters

But: Dependence on shape for self-attention layers

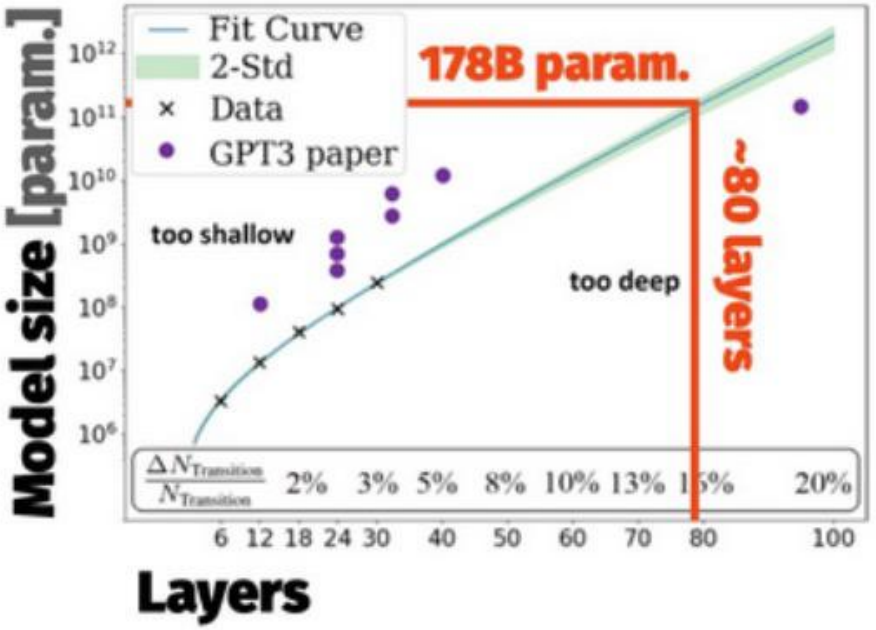
- Levine et al 2020: **relation between width and depth in self-attention** - theoretically & empirically
- **depth threshold** $H_{th}(d_x) \sim \log(d_x)$ as function of network width (latent dim): (theoretical result uses "separation rank" as capacity - ability to model dependencies between subsets of variables)
- **depth efficiency**: network of depth $\leq H_{th}(d_x)$ cannot be replicated by a shallower network (unless width grows doubly-exponentially in depth)
  - separation rank increases polynomially in width, doubly-exponentially in depth
- **balanced growth ("depth inefficiency"):** for depth $> H_{th}(d_x)$, ability to model input dependencies increases similarly with depth and width — separation rank increases as exp(depth x width)
- => make networks not only deeper, but also wider: if not wide enough, can't use excess layers efficiently. Claim: GPT3 is too narrow

Levine et al 2020, Limits to depth-efficiencies of self-attention

Optimal predicted depth-width relation

- used e.g. in BLOOM:

The Depth-to-Width Interplay in Self-Attention, Levine et al.

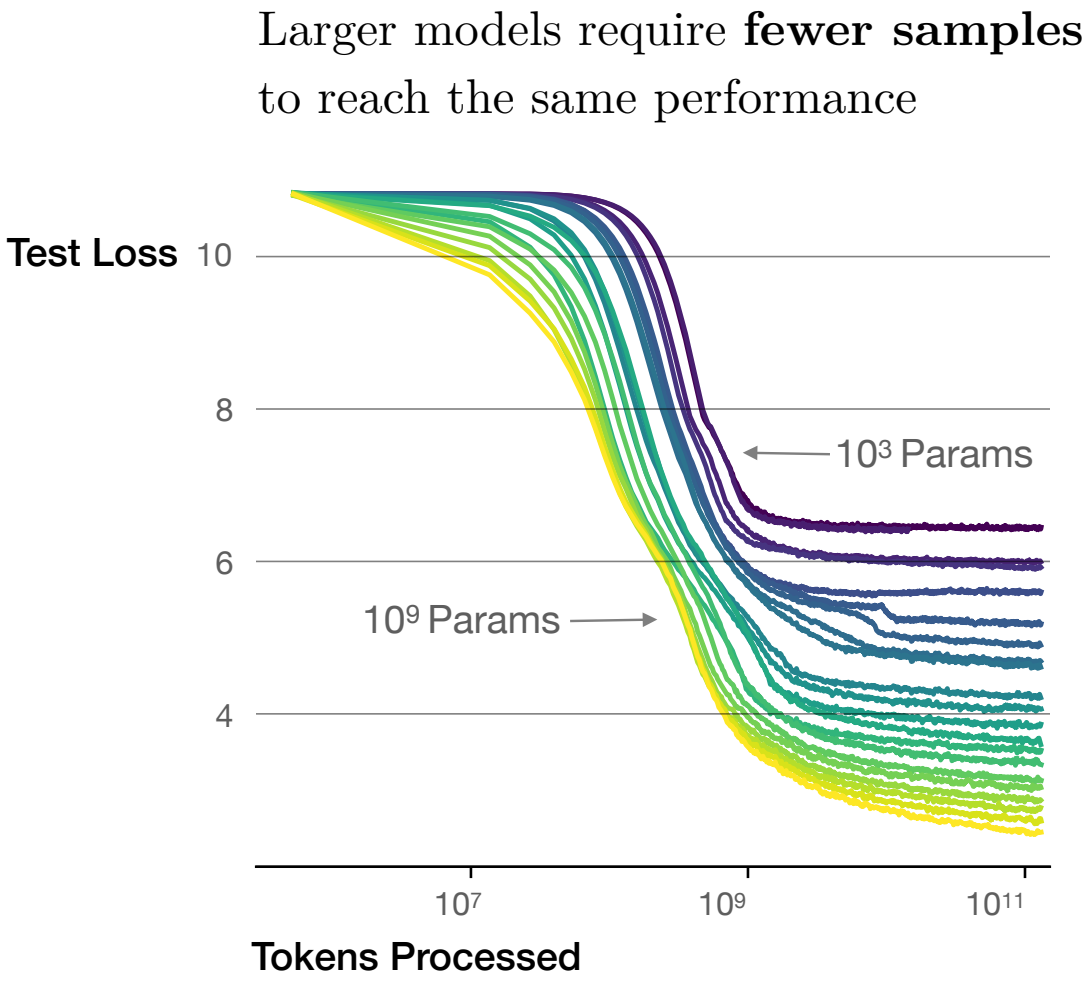


Predicts an optimal width (hidden dim.) vs depth ratio.
At 178B parameters, the recommended depth is ~80 layers.

Best fit: configuration (1) and (2).

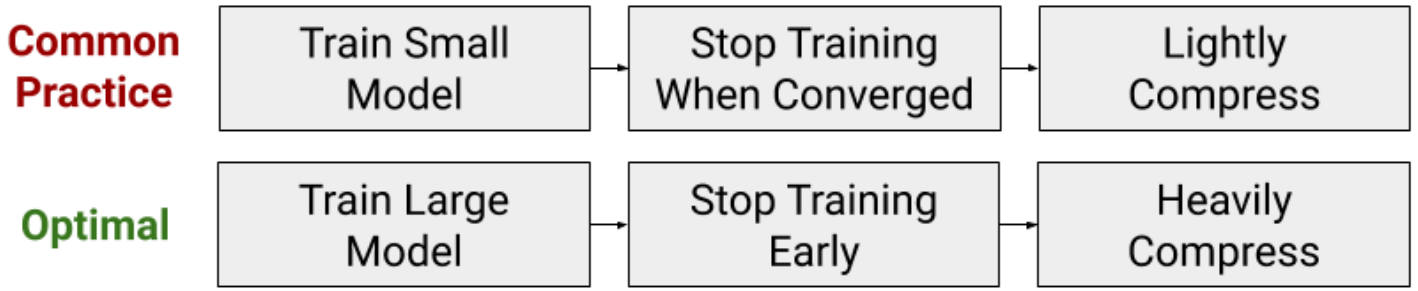
Similarly, for vision models: scaling depth and width in proper relation improves performance (EfficientNet, Tan & Le 2019)

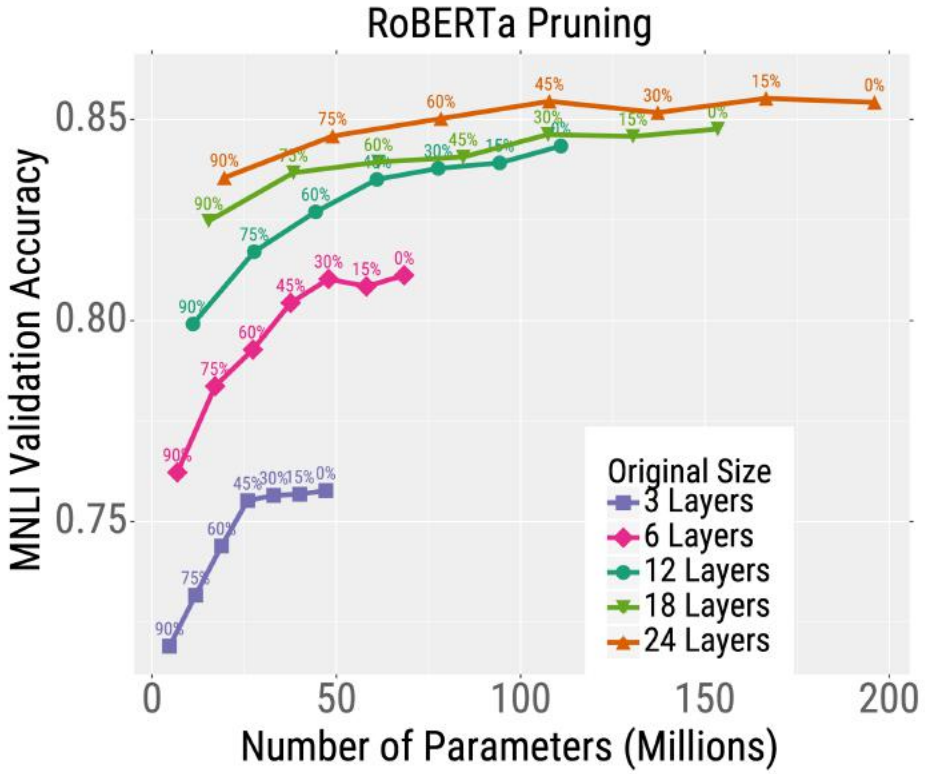
Finding #4: larger models require fewer samples to reach the same performance



Similar findings

- Li et al, 2020: larger models train faster, not much loss via compression





https://bair.berkeley.edu/blog/2020/03/05/compress/

Take-aways from Kaplan et al.

- test loss scales as power law as a function of data, parameter count, compute
- test loss is more sensitive to parameter count than data: with more compute, increase model by a larger factor than dataset size
- don't train to completion, rather make your model larger
- in general, scaling laws can help to "optimally" allocate computation

"Caveats" for Kaplan et al.

- "... the scaling with D at very large model size still remains mysterious. Without a theory or systematic understanding of the corrections to our scaling laws, it's difficult to determine in what circumstances they can be trusted"

- didn't thoroughly investigate small data regime, poor fits of L(N,D) in that regime. Didn't investigate regularization or data augmentation.

- didn't tune all hyperparameters; possibly better learning rates for short training runs

Implications

- subsequently: larger models, scale data only moderately

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion

spend compute on model size ...

Finding the right predictions is not easy..

... or on data?

DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

Two years later…

under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly

- still assume a power law, but find somewhat different equations / scaling
- still do not train to minimum loss, but train for many more tokens (& steps), increase #parameters and data equally: “smaller” models, train longer

Differences

- different learning rate schedules (not explored in Kaplan et al.), adapted to dataset size

- larger models
  (> 500M parameters)

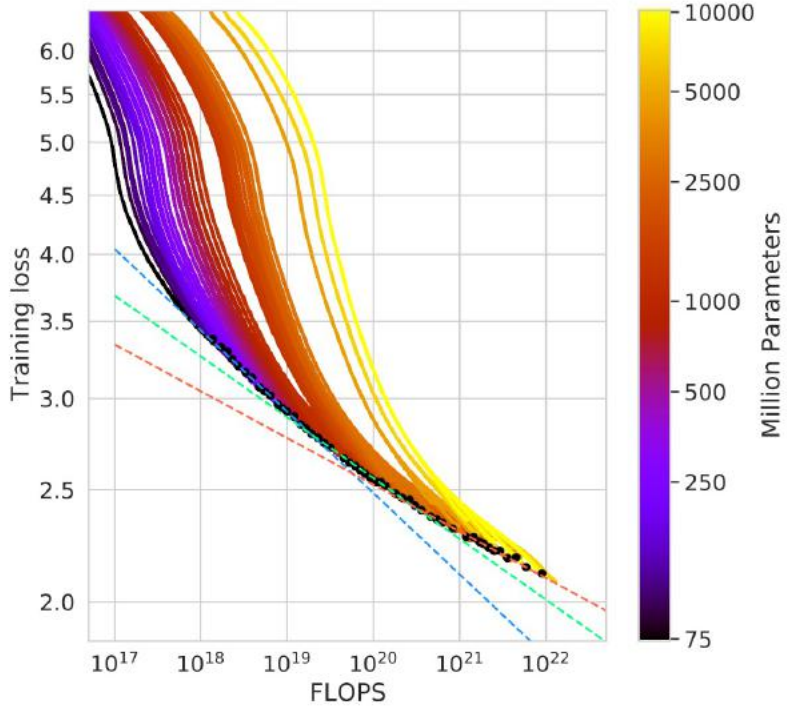


Figure A5 | **Training curve envelopes.** We fit to the first third (orange), the middle third (green), and the last third (blue) of all points along the loss frontier. We plot only a subset of the points.

Approaches to estimate loss as a function of data and parameter count

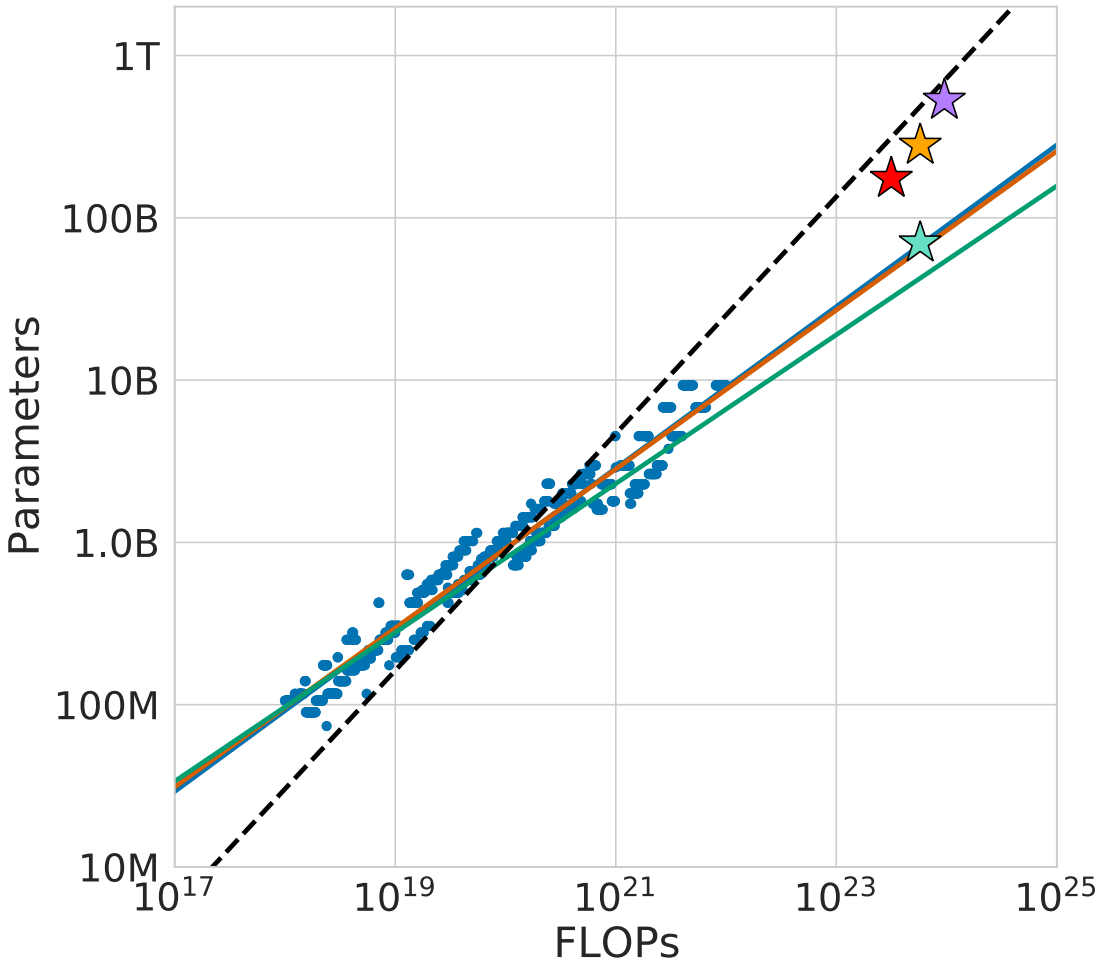
- minimize L(N,D) with constraint FLOPs(N,D) = budget
- Approach 1: for each fixed model size, record entire training curve (=different #training tokens, as they use 1 epoch only)
- Approach 2: fixed #FLOPs, use final training loss for different N,D
- Approach 3: fit parametric loss function $\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$

- for all 3: optimal scaling of #parameters N / data D with compute C is roughly

$$N \propto C^{0.49}, D \propto C^{0.51}$$ **vs Kaplan et al: #params $\propto C^{0.73}$, data $\propto C^{0.27}$**

=> existing large models would need much more training data.
Better allocation: **smaller model, more data/steps**

Predictions: extrapolation



For a given #FLOPs, how many parameters should we allocate? (compute = 6 x data x params)

- Approach 1
- Approach 2
- Approach 3
- Kaplan et al (2020)

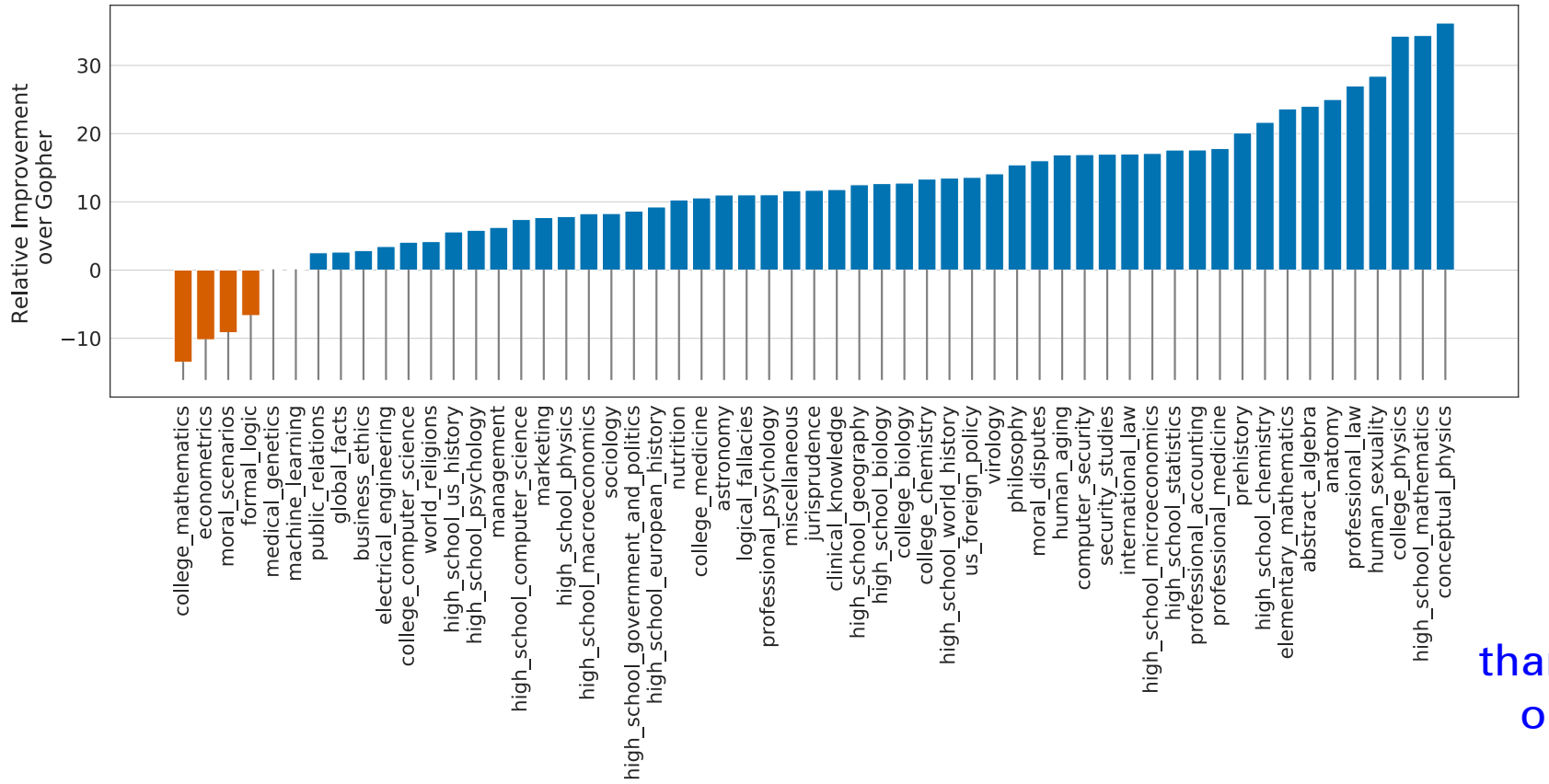
- Chinchilla (70B)
- Gopher (280B)
- GPT-3 (175B)
- Megatron-Turing NLG (530B)

Model comparison

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

Model	**Layers**	**Number Heads**	**Key/Value Size**	$\mathbf{d_{model}}$	**Max LR**	**Batch Size**
Gopher 280B	80	128	128	16,384	4×10^{-5}	3M $\rightarrow$ 6M
Chinchilla 70B	80	64	128	8,192	1×10^{-4}	1.5M $\rightarrow$ 3M

Example results for Chinchilla: answering exam questions



works better than existing models on multiple tasks

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.

Evaluation of Bias

- e.g.: pronoun resolution

1. **The nurse** notified the patient that...
    i. **her** shift would be ending in an hour.
    ii. **his** shift would be ending in an hour.
    iii. **their** shift would be ending in an hour.
2. The nurse notified **the patient** that...
    i. **her** blood would be drawn in an hour.
    ii. **his** blood would be drawn in an hour.
    iii. **their** blood would be drawn in an hour.

- Winogender dataset: can a model determine if a pronoun refers to different occupation words, regardless of gender?
- Gotcha examples: correct pronoun resolution contradicts gender stereotypes

Evaluation of 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.

Discussion / limitations

- use (smoothed) training loss as a proxy for test loss
- expensive experiments: only two large-scale training runs (Gopher, Chinchilla), no additional tests at intermediate scales
- assume a power law for L(N) — but some concavity at high compute budgets, so it may not hold universally
- only one epoch of training (each data point seen once)
- data needs scrutiny for bias, toxicity, harder if more data

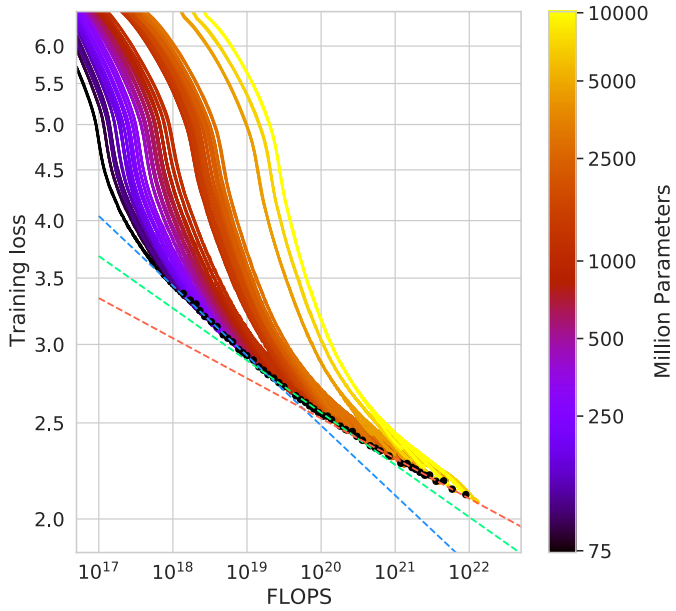


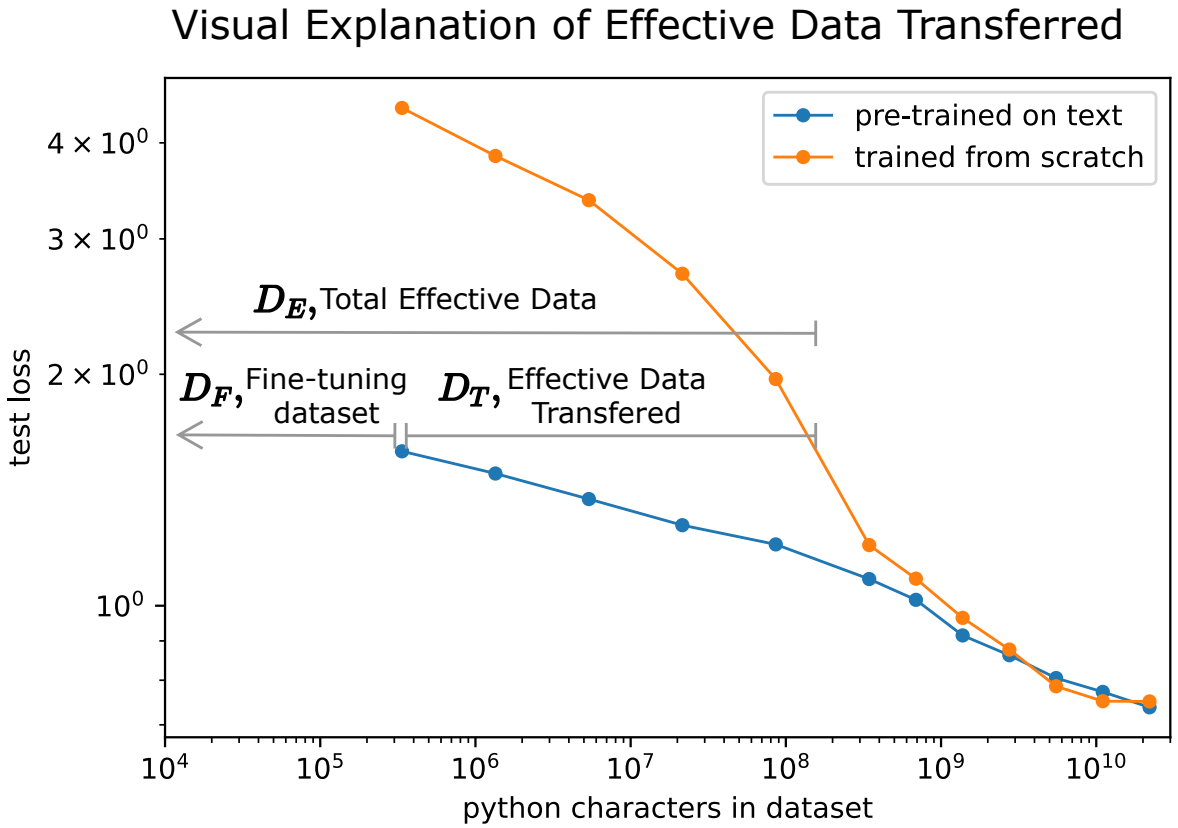
Figure A5 | **Training curve envelopes.** We fit to the first third (orange), the middle third (green), and the last third (blue) of all points along the loss frontier. We plot only a subset of the points.

More scaling laws: transfer learning

- amount of transferred data scales as power law in low-data regime:

$$D_T = k \cdot D_F^{\alpha} \cdot N^{\beta}$$

	Transfer Coefficients		
Transfer from	k	α	β
Text $\Longrightarrow$ Python	1.9e4	0.18	0.38
50% Text and 50% non-python code $\Longrightarrow$ Python	2.1e5	0.096	0.38



- larger k: mixed data models transfer better in low data regime
- smaller α: benefit diminishes in high-data regime
- scaling in model size independent of data

Hernandez et al 2021, Scaling Laws for Transfer

Many more on scaling laws...

- many earlier works, too (Tan & Le 2019, Cho et al 2015, Miceli Barone et al 2017, Johnson et al 2018, Hestness et al 2017)
- vision models tend to have better scaling in model size than transformers
- $L(N) \propto 1/\mathrm{N}^{\alpha}$ for vision models (CNNs), with larger $\alpha \approx 0.5$ (Rosenfeld et al 2019, Sharma et al)
- power law for $L(D)$ across vision, language, speech tasks (Hestness et al 2017)
- power laws in theoretical bounds: model (Yarotsky, 2018), data size (Liang et al, 2019)
- …

General side note: carbon footprint

Model name	Number of parameters	Datacenter PUE	Carbon intensity of grid used	Power consumption	CO_2eq emissions	CO_2eq emissions × PUE
GPT-3	175B	1.1	429 gCO_2eq/kWh	1,287 MWh	*502 tonnes*	552 tonnes
Gopher	280B	1.08	330 gCO_2eq/kWh	*1,066 MWh*	*352 tonnes*	380 tonnes
OPT	175B	*1.09* [2]	*231gCO_2eq/kWh*	*324 MWh*	70 tonnes	*76.3 tonnes* [3]
BLOOM	176B	1.2	57 gCO_2eq/kWh	433 MWh	25 tonnes	30 tonnes

for comparison: lifetime of a US car including fuel: 57t CO_2

Luccioni et al, Estimating the Carbon Footprint of BLOOM, a 176 Billion parameter language model

Can we explain scaling laws? One hypothesis

A hypothesis: data manifold

Scaling Laws from the Data Manifold Dimension

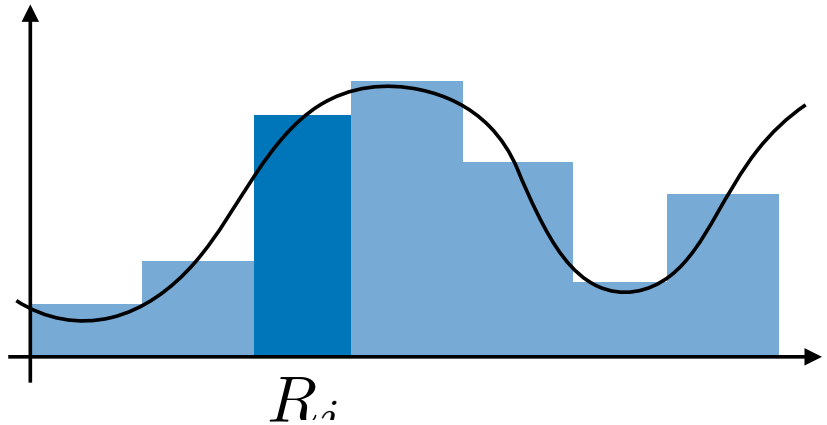
Utkarsh Sharma USHARMA7@JHU.EDU
Department of Physics and Astronomy
Johns Hopkins University
Baltimore, MD 21218, USA

Jared Kaplan JAREDK@JHU.EDU
Department of Physics and Astronomy
Johns Hopkins University
Baltimore, MD 21218, USA

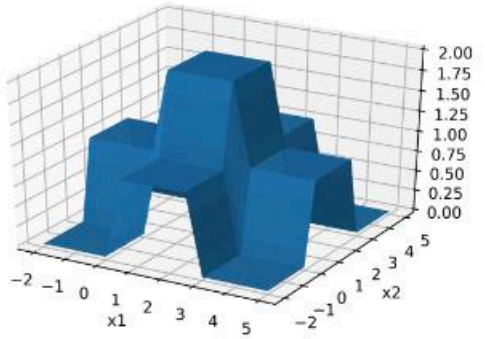
- Idea: neural networks are doing regression on an embedded data manifold

A toy model

- Toy model: approximate Lipschitz function $f(x)$ on $[0,1]^d$ by a piecewise constant function $c(x)$.
- Use hypercubes with side length s: $N = s^{-d}$ regions (values).



- MSE scales as $L(s) = \int_0^1 |f(x) - c(x)|^2 d^d x \lesssim \lambda^2 (s^2 d)$



Lipschitz constant

- So, up to constant factors, $L(N) \lesssim \frac{1}{N^{2/d}}$
- For piecewise linear approximations (ReLU), MSE and cross entropy scale as s^4, so $L(N) \lesssim N^{-4/d}$

Relation to neural networks: hypothesis

- For piecewise linear approximations (ReLU), MSE and cross entropy scale as s^4, so $L(N) \lesssim N^{-4/d}$
- Common belief: NN map data into a low-dimensional "manifold" that depends on data, loss/task
- If we take d to be intrinsic dimension of data manifold, then we get a scaling law

$$L(N) \propto \frac{1}{N^{\alpha}} \text{ with } \alpha \approx 4/d$$

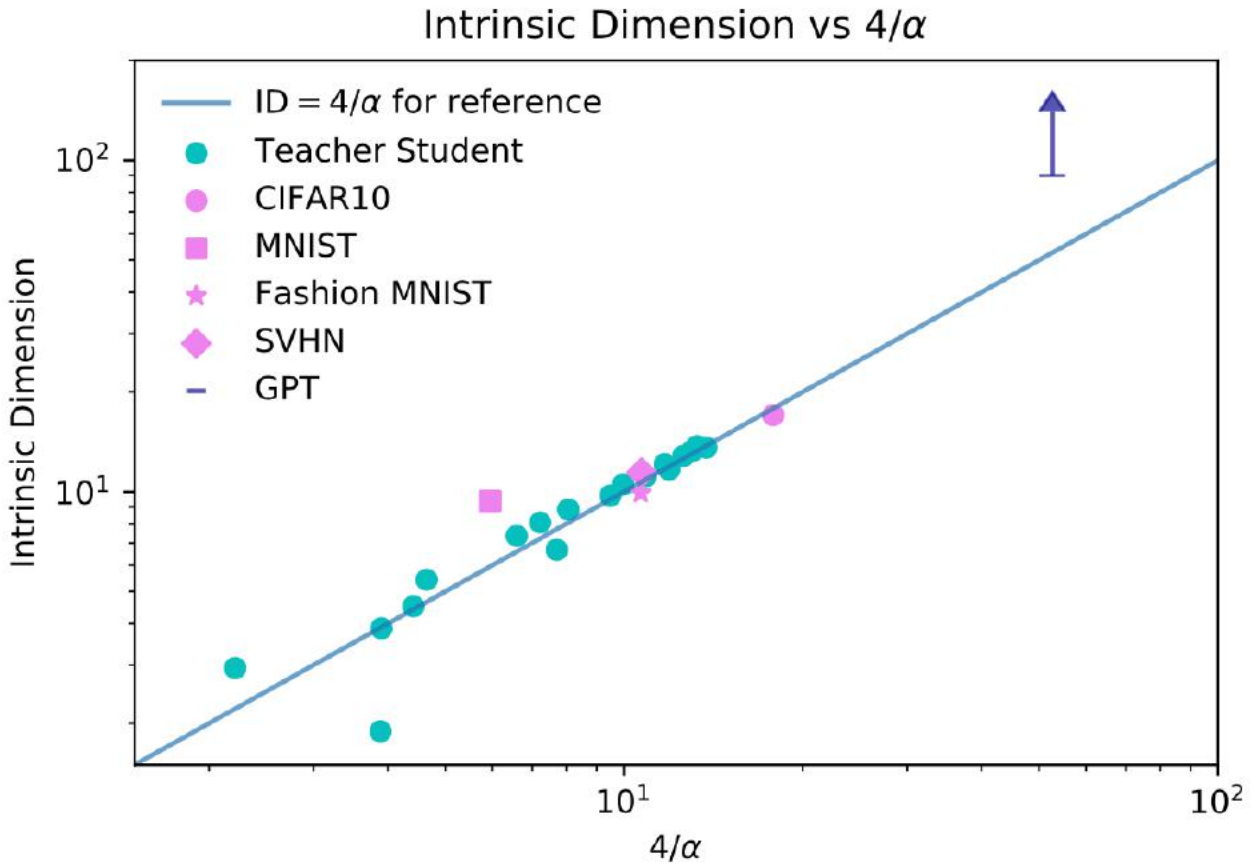
- Suggests that scaling exponent is strongly related to data and task: different models will scale similarly on the same data

Empirical evidence

- Hypothesis: $L(N) \propto \frac{1}{N^{\alpha}}$ with $\alpha \approx 4/d$

- Experiment: measure intrinsic dimension of embedding and relate to scaling exponent α

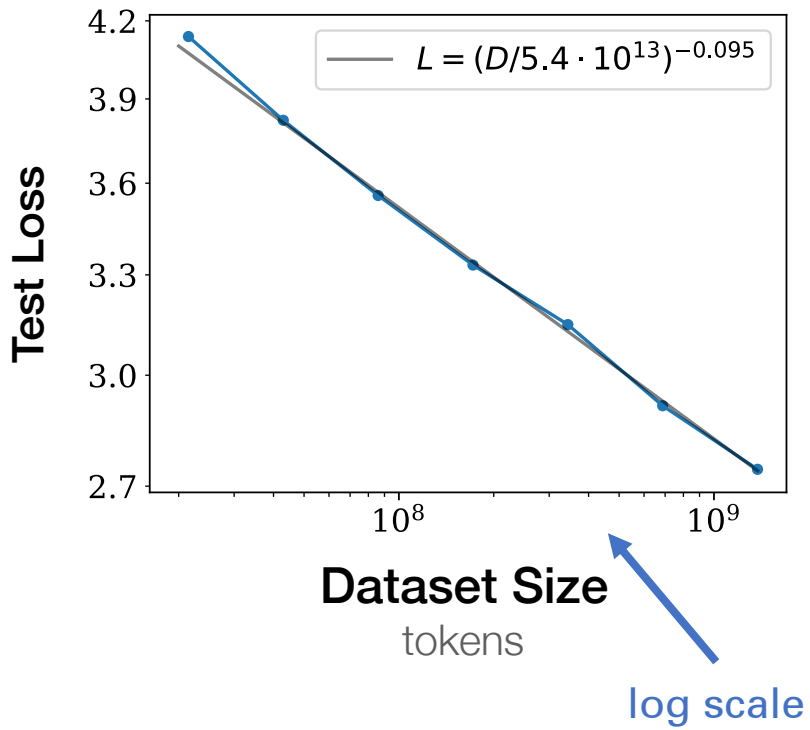
- Good fit for vision models, only upper bound for transformer: $\alpha \gtrsim 4/d$ from upper bound on L(N)



Are scaling laws avoidable?

Are scaling laws avoidable?

- Power laws with small exponents: a small
- improvement in loss may need an order of magnitude more data/parameters
- Diminishing returns: suggests that many training examples are highly redundant
- Pruning intelligently can lead to better scaling (e.g. Sorscher et al, 2022)



Sorscher et al, Beyond neural scaling laws: beating power law scaling via data pruning

Theoretical motivation for pruning

- Analysis in simple student-teacher model: perceptron; "teacher" perceptron labels iid Gaussian training examples

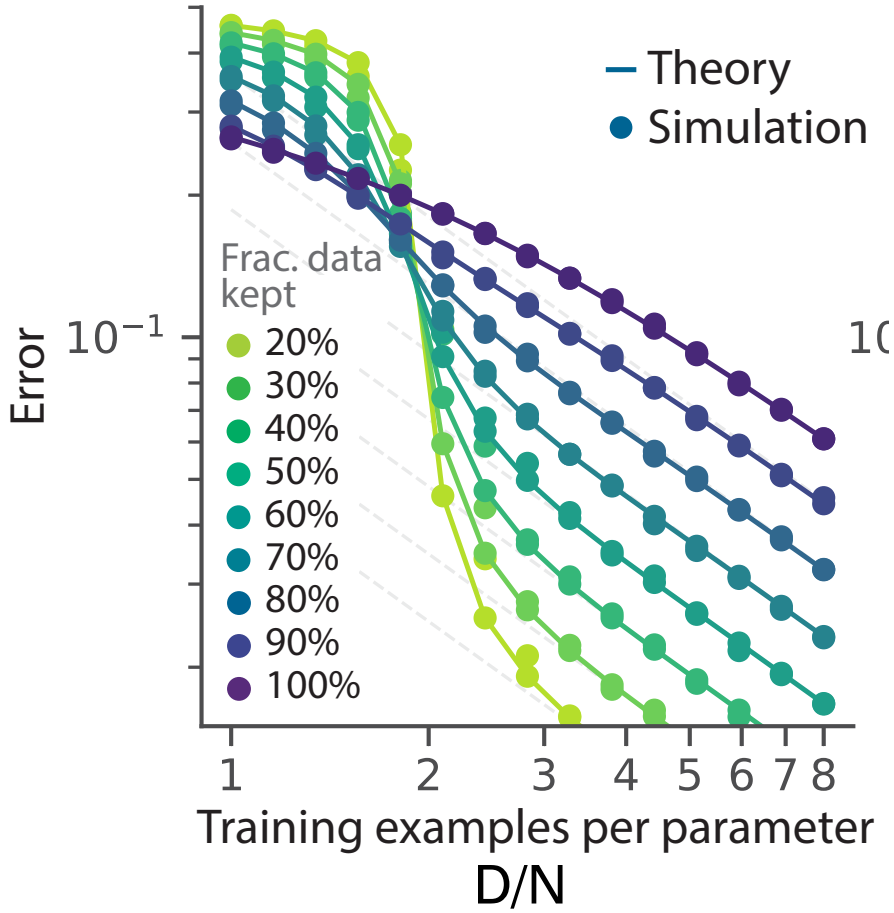
- Pruning algorithm: retain small margin examples = "hard" examples

1. train a student probe perceptron for few iterations, get weights $\mathbf{w}$
2. compute margin $m^{(\mathrm{i})} = \mathbf{w}^{\top}(\mathbf{x}^{(\mathrm{i})}\mathbf{y}^{(\mathrm{i})})$ for each datapoint i
3. prune data down to retain fraction of f hardest examples
4. train a new perceptron on smaller dataset

Sorscher et al, Beyond neural scaling laws: beating power law scaling via data pruning

Findings

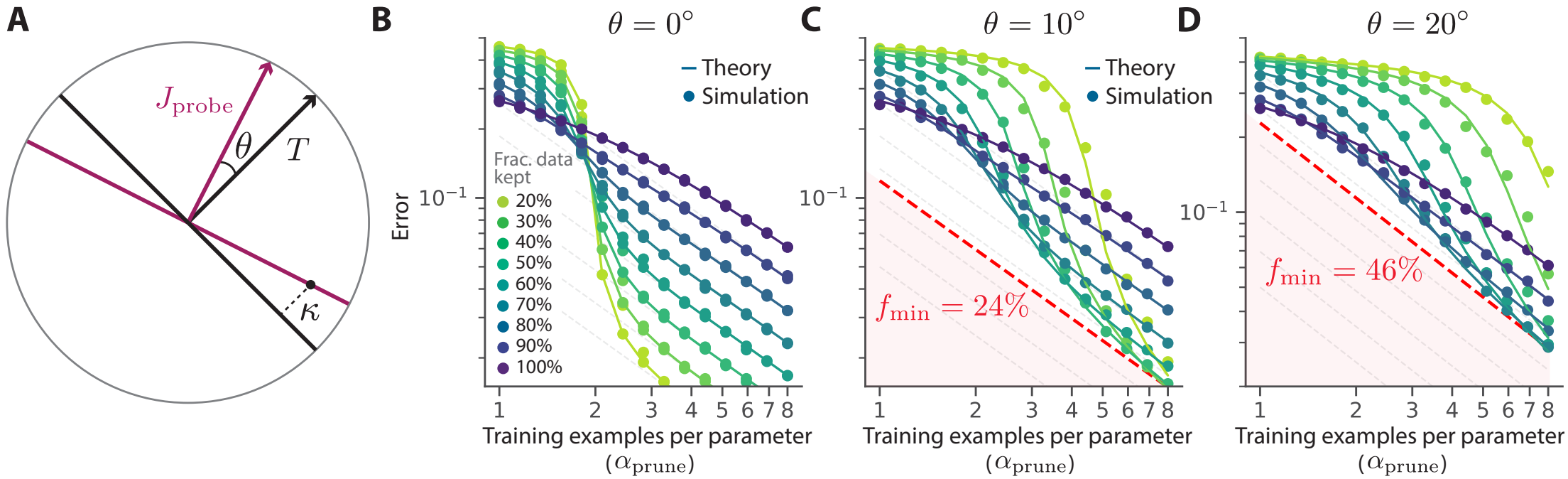
- Best pruning strategy depends on initial amount of data D, model parameters N:
- small D/N: use easy examples; large D/N: use hard examples
- Initially, error drops exponentially in f · D/N , then approaches power law in “data size” f · D/N .
- Adjusting pruning fraction 1-f with data size can yield exponential scaling law (Pareto front)



Sorscher et al, Beyond neural scaling laws: beating power law scaling via data pruning

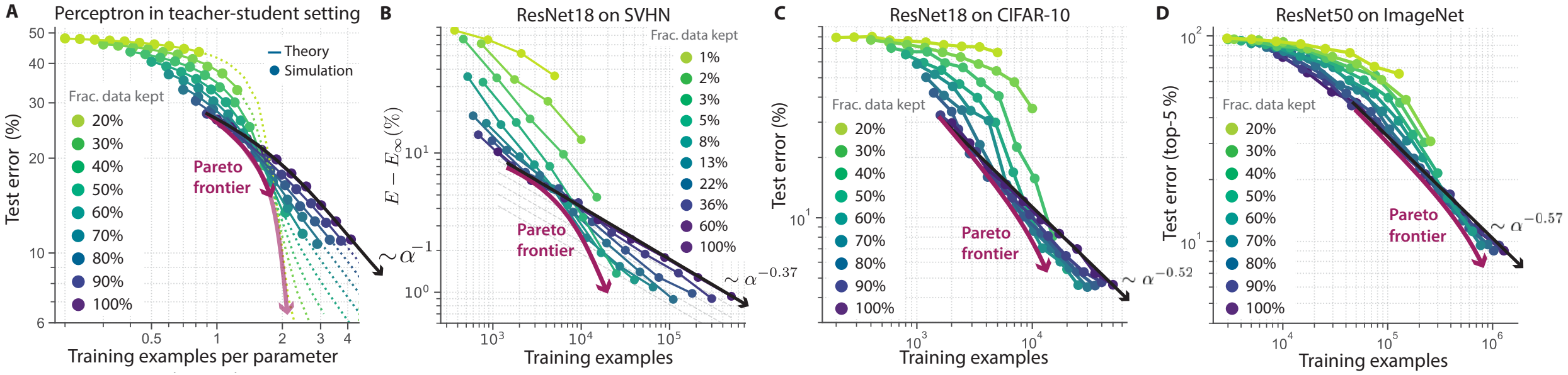
Findings

- The gain from scaling depends on how good our estimate of the margin is: worse estimates eventually go back to power law



Sorscher et al, Beyond neural scaling laws: beating power law scaling via data pruning

From perceptron to neural networks



- Side note: Many pruning strategies possible, not all work equally well (while being efficient)

Sorscher et al, Beyond neural scaling laws: beating power law scaling via data pruning

Summary: Scaling Laws

- Neural scaling laws: predict test loss as a function of resources and model hyperparameters
- Allows to “optimally” allocate compute resources
- Power law scaling in model size, data size across variety of models and tasks
- Actual parameters hard to measure, large extrapolation nontrivial

Massive Text Models

Scaling models also scales author lists

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

Scaling Language Models: Methods, Analysis & Insights from Training *Gopher*

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu and Geoffrey Irving

Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model

Shaden Smith[§,†], Mostofa Patwary[§,‡], Brandon Norick[†], Patrick LeGresley[‡], Samyam Rajbhandari[†], Jared Casper[‡], Zhun Liu[†], Shrimai Prabhumoye[‡], George Zerveas*[†], Vijay Korthikanti[‡], Elton Zhang[†], Rewon Child[‡], Reza Yazdani Aminabadi[†], Julie Bernauer[‡], Xia Song[†], Mohammad Shoeybi[‡], Yuxiong He[†], Michael Houston[‡], Saurabh Tiwary[†], and Bryan Catanzaro[‡]

PaLM: Scaling Language Modeling with Pathways

Aakanksha Chowdhery* Sharan Narang* Jacob Devlin*
Maarten Bosma Gaurav Mishra Adam Roberts Paul Barham
Hyung Won Chung Charles Sutton Sebastian Gehrmann Parker Schuh Kensen Shi
Sasha Tsvyashchenko Joshua Maynez Abhishek Rao† Parker Barnes Yi Tay
Noam Shazeer† Vinodkumar Prabhakaran Emily Reif Nan Du Ben Hutchinson
Reiner Pope James Bradbury Jacob Austin Michael Isard Guy Gur-Ari
Pengcheng Yin Toju Duke Anselm Levskaya Sanjay Ghemawat Sunipa Dev
Henryk Michalewski Xavier Garcia Vedant Misra Kevin Robinson Liam Fedus
Denny Zhou Daphne Ippolito David Luan† Hyeontaek Lim Barret Zoph
Alexander Spiridonov Ryan Sepassi David Dohan Shivani Agrawal Mark Omernick
Andrew M. Dai Thanumalayan Sankaranarayana Pillai Marie Pellat Aitor Lewkowycz
Erica Moreira Rewon Child Oleksandr Polozov† Katherine Lee Zongwei Zhou
Xuezhi Wang Brennan Saeta Mark Diaz Orhan Firat Michele Catasta† Jason Wei
Kathy Meier-Hellstern Douglas Eck Jeff Dean Slav Petrov Noah Fiedel

GPT-3 Brown et al. 2020

- 175B parameter transformer decoder, combined training set is ~500B tokens (though the model is only actually trained for 300B tokens)

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Sma	Dataset	Quantity (tokens)		Weight in training mix		Epochs elapsed when training for 300B tokens	0×10^{-4}
GPT-3 Med							0×10^{-4}
GPT-3 Lar	Common Crawl (filtered)	410 billion		60%		0.44	5×10^{-4}
GPT-3 XL	WebText2	19 billion		22%		2.9	0×10^{-4}
GPT-3 2.7I	Books1	12 billion		8%		1.9	6×10^{-4}
GPT-3 6.7I	Books2	55 billion		8%		0.43	2×10^{-4}
GPT-3 13B	Wikipedia	3 billion		3%		3.4	0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

GPT-3 Brown et al. 2020

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

- GPT-3 demonstrates impressive few-shot learning performance, though there is still room for significant improvement

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Gopher Rae et al, 2021

- Gopher is a 280B parameter transformer decoder
- The training set has over 2T tokens, the model is still only trained for 300B tokens

Model	Layers	Number Heads	Key/Value Size	d_{model}	Max LR	Batch Size
44M	8	16	32	512	6×10^{-4}	0.25M
117M	12	12	64	768	6×10^{-4}	0.25M
417M	12	12	128	1,536	2×10^{-4}	0.25M
1.4B	24	16	128	2,048	2×10^{-4}	0.25M
7.1B	32	32	128	4,096	1.2×10^{-4}	2M
Gopher 280B	80	128	128	16,384	4×10^{-5}	3M $\rightarrow$ 6M

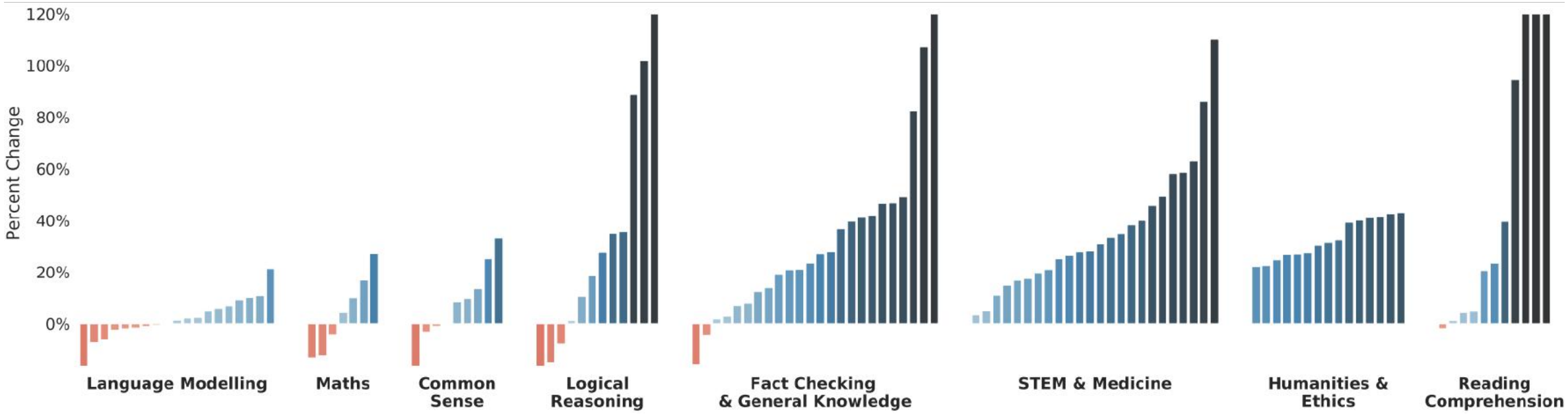
Gopher Rae et al, 2021

- Gopher is a 280B parameter transformer decoder
- The training set has over 2T tokens, the model is still only trained for 300B tokens

Model	Layers	Number Heads	Key/Value Size	d	Max LR	Batch Size
44M						0.25M
117M						0.25M
417M						0.25M
1.4B						0.25M
7.1B						2M
Gopher 28(						3M → 6M

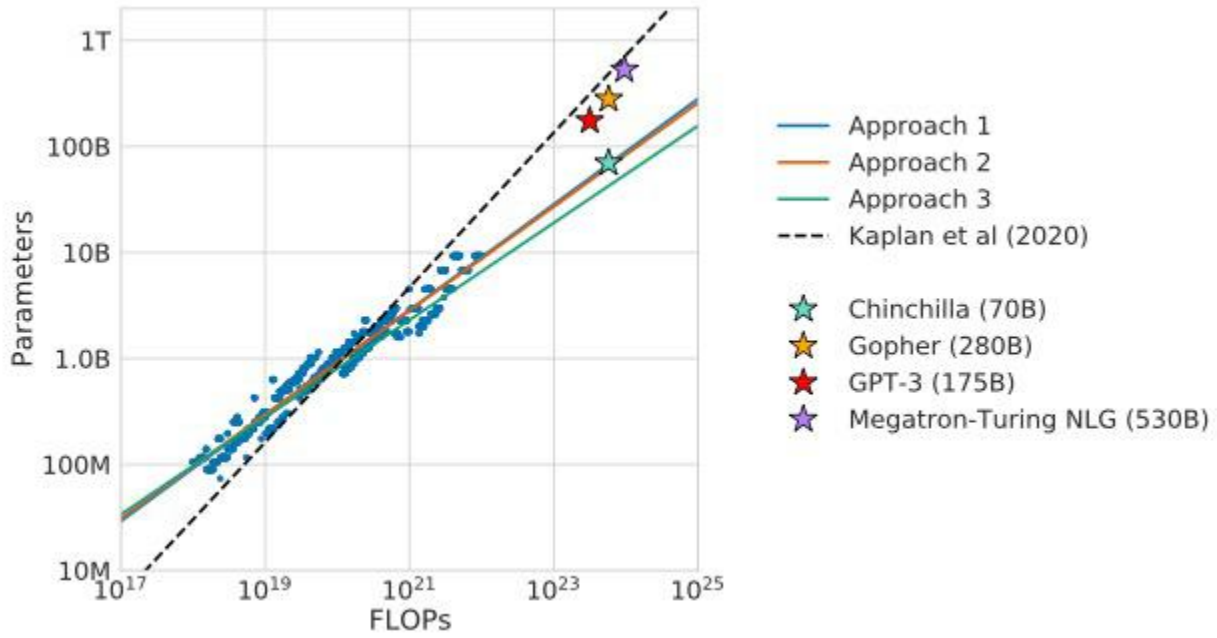
	Disk Size	Documents	Tokens	Sampling proportion
MassiveWeb	1.9 TB	604M	506B	48%
Books	2.1 TB	4M	560B	27%
C4	0.75 TB	361M	182B	10%
News	2.7 TB	1.1B	676B	10%
GitHub	3.1 TB	142M	422B	3%
Wikipedia	0.001 TB	6M	4B	2%

Gopher Rae et al, 2021



Chinchilla: a "smaller Gopher" Hoffman et al, 2022

- Chinchilla considers varying hyperparameters (primarily, learning rate schedule) that Kaplan et al held fixed, which leads to different scaling conclusions
- In particular, they advocate that model and data size scaling should be equal



Chinchilla performance

On Massive Multitask Language Understanding (MMLU)

- The MMLU benchmark contains exam questions from 57 academic subjects, ranging from elementary to professional level difficulty

Task	*Chinchilla*	*Gopher*	Task	*Chinchilla*	*Gopher*
abstract_algebra	31.0	25.0	anatomy	70.4	56.3
astronomy	73.0	65.8	business_ethics	72.0	70.0
clinical_knowledge	75.1	67.2	college_biology	79.9	70.8
college_chemistry	51.0	45.0	college_computer_science	51.0	49.0
college_mathematics	32.0	37.0	college_medicine	66.5	60.1
college_physics	46.1	34.3	computer_security	76.0	65.0
conceptual_physics	67.2	49.4	econometrics	38.6	43.0
electrical_engineering	62.1	60.0	elementary_mathematics	41.5	33.6
formal_logic	33.3	35.7	global_facts	39.0	38.0
high_school_biology	80.3	71.3	high_school_chemistry	58.1	47.8
high_school_computer_science	58.0	54.0	high_school_european_history	78.8	72.1
high_school_geography	86.4	76.8	high_school_gov_and_politics	91.2	83.9
high_school_macroeconomics	70.5	65.1	high_school_mathematics	31.9	23.7
high_school_microeconomics	77.7	66.4	high_school_physics	36.4	33.8
high_school_psychology	86.6	81.8	high_school_statistics	58.8	50.0
high_school_us_history	83.3	78.9	high_school_world_history	85.2	75.1
human_aging	77.6	66.4	human_sexuality	86.3	67.2
international_law	90.9	77.7	jurisprudence	79.6	71.3
logical_fallacies	80.4	72.4	machine_learning	41.1	41.1
management	82.5	77.7	marketing	89.7	83.3
medical_genetics	69.0	69.0	miscellaneous	84.5	75.7
moral_disputes	77.5	66.8	moral_scenarios	36.5	40.2
nutrition	77.1	69.9	philosophy	79.4	68.8
prehistory	81.2	67.6	professional_accounting	52.1	44.3
professional_law	56.5	44.5	professional_medicine	75.4	64.0
professional_psychology	75.7	68.1	public_relations	73.6	71.8
security_studies	75.9	64.9	sociology	91.0	84.1
us_foreign_policy	92.0	81.0	virology	53.6	47.0
world_religions	87.7	84.2			

Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
Gopher 5-shot	60.0%
***Chinchilla* 5-shot**	**67.6%**
Average human expert performance	*89.8%*
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

Megatron-Turing NLG Smith et al, 2022

- MT-NLG is a 530B parameter transformer decoder
- Training set is (a puny) 339B tokens, training is done with 270B tokens

Dataset	Tokens (billion)	Weights (%)	Epochs
Books3	25.7	14.3	1.5
OpenWebText2	14.8	19.3	3.6
Stack Exchange	11.6	5.7	1.4
PubMed Abstracts	4.4	2.9	1.8
Wikipedia	4.2	4.8	3.2
Gutenberg (PG-19)	2.7	0.9	0.9
BookCorpus2	1.5	1.0	1.8
NIH ExPorter	0.3	0.2	1.8
ArXiv	20.8	1.4	0.2
GitHub	24.3	1.6	0.2
Pile-CC	49.8	9.4	0.5
CC-2020-50	68.7	13.0	0.5
CC-2021-04	82.6	15.7	0.5
Realnews	21.9	9.0	1.1
CC-Stories	5.3	0.9	0.5

PaLM Chowdhery et al, 2022

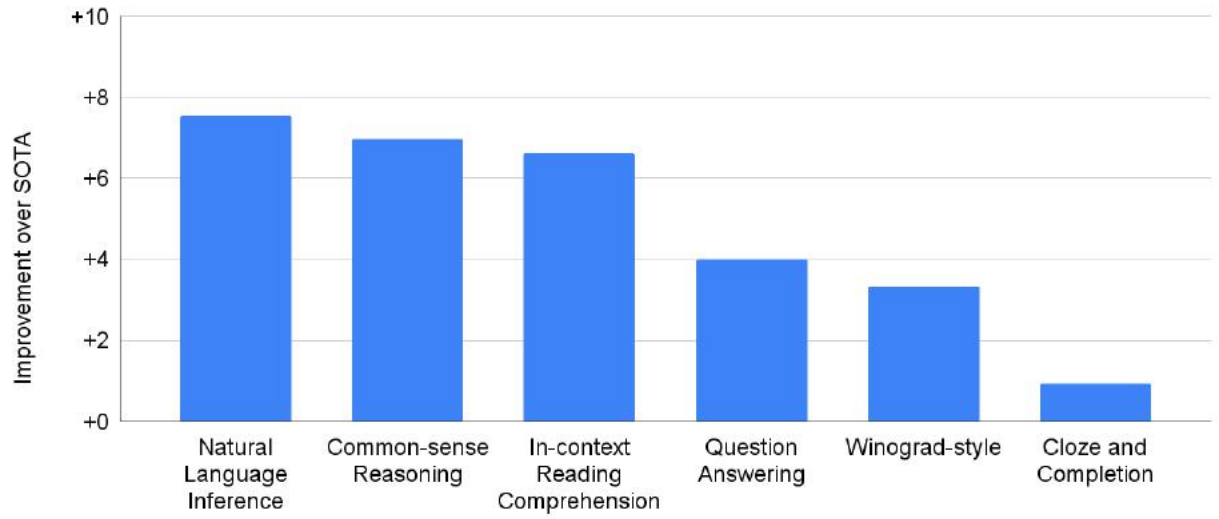
- PaLM is a 540B parameter (yes, you guessed it) transformer decoder
- Training set: 780B tokens; training: one full epoch!

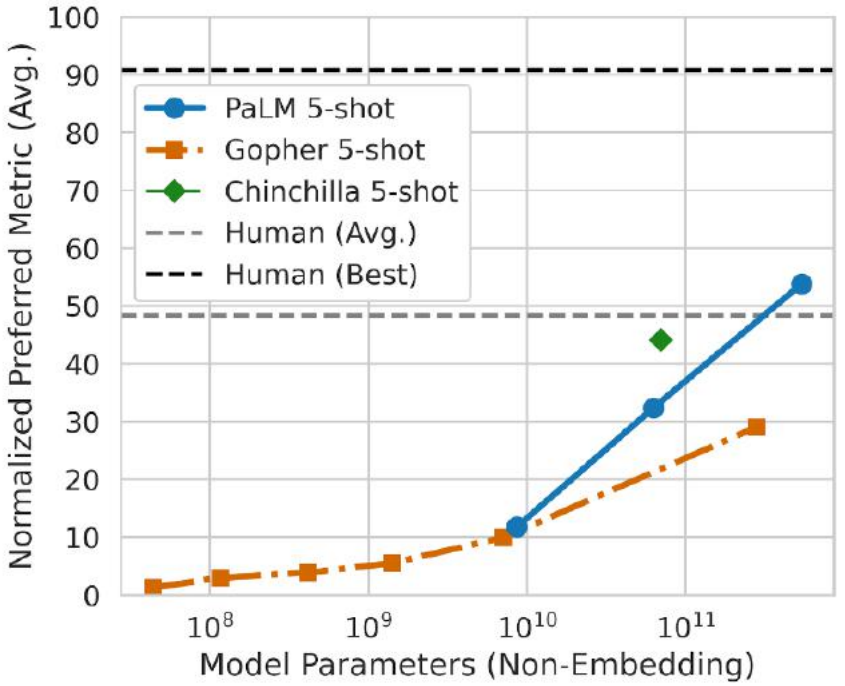
Model	Layers	# of Heads	d_{model}	# of Parameters (in billions)	Batch Size
PaLM 8B	32	16	4096	8.63	$256 \rightarrow 512$
PaLM 62B	64	32	8192	62.50	$512 \rightarrow 1024$
PaLM 540B	118	48	18432	540.35	$512 \rightarrow 1024 \rightarrow 2048$

Total dataset size = 780 billion tokens	
Data source	Proportion of data
Social media conversations (multilingual)	50%
Filtered webpages (multilingual)	27%
Books (English)	13%
GitHub (code)	5%
Wikipedia (multilingual)	4%
News (English)	1%

PaLM Chowdhery et al, 2022

- PaLM improves across a number of natural language tasks, including the recently proposed **BIG-Bench** (right), a recently proposed benchmark of >200 tasks designed for evaluating large language models





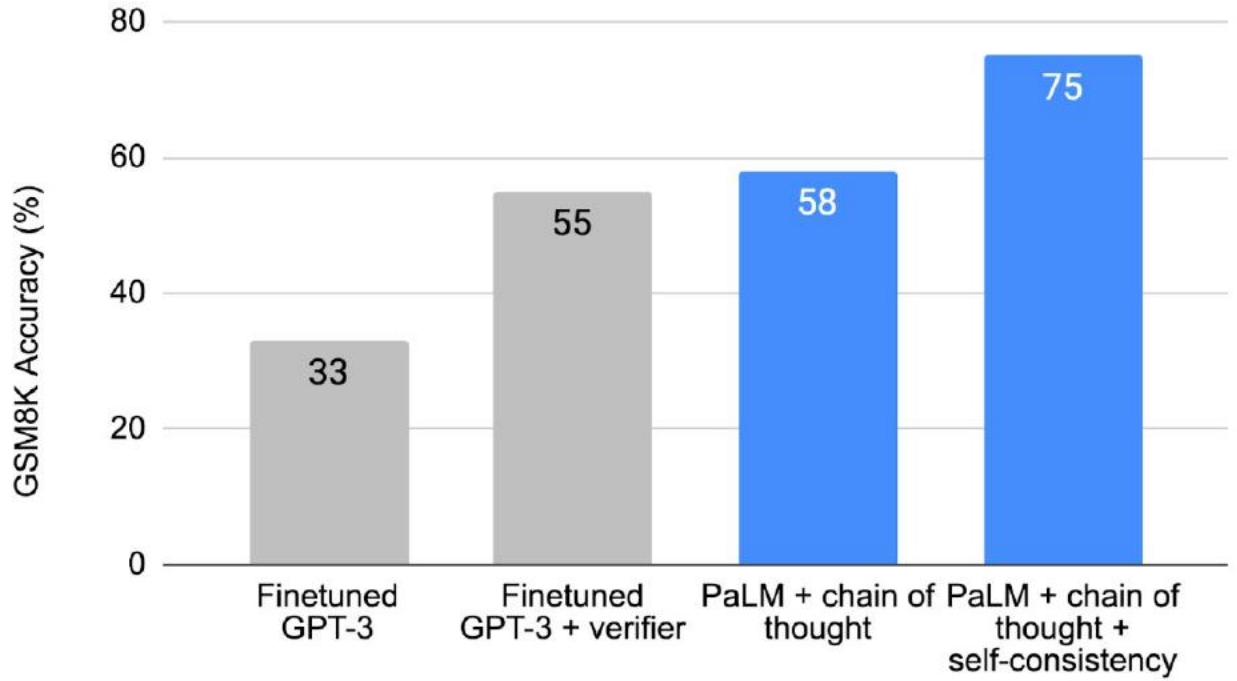
Applications of massive models

Few-shot learning via prompting

- GPT-3's **prompting** strategy has also been adopted by other large models: provide some number of examples of the task within the model's input itself
  - Often, the number of examples is less than 10
- **Prompt engineering** is now an important part of getting large models to perform their best, and this often requires some fiddling to get right
- PaLM sometimes uses **chain-of-thought prompting**, which not only prompts with the correct answer but also the process by which that answer is reached
  - This significantly improves performance for some more complex tasks

PaLM + chain-of-thought + self-consistency

- Further combined with **self-consistency** (sampling multiple answers and picking the most consistent answer), PaLM + chain-of-thought results in substantial improvements on eighth grade arithmetic word problems (GSM8K)



PaLM can explain jokes with chain-of-thought prompting

Explaining a joke

Prompt

```
Explain this joke:

Joke: Did you see that Google just hired an eloquent whale for
their TPU team? It showed them how to communicate between two
different pods!
```

Model Response

```
Prediction: TPUs are a type of computer chip that Google uses
for deep learning. A "pod" is a group of TPUs. A "pod" is also a
group of whales. The joke is that the whale is able to
communicate between two groups of whales, but the speaker is
pretending that the whale is able to communicate between two
groups of TPUs.
```

"Medium-shot" learning with fine tuning

- **Fine tuning** usually refers to updating the model via gradient based optimization with a small dataset (hundreds or thousands of data points)
- With the size of these models, even this can be impractical or even infeasible
  - GPT-3 offers fine tuning as part of the OpenAI API
- When possible, fine tuning still outperforms prompting significantly

Model	BoolQ	CB	CoPA	MultiRC	Record	RTE	WiC	WSC
Few-shot	89.1	89.3	95	86.3/-	92.9/-	81.2	64.6	89.5
Finetuned	92.2	100/100	100	90.1/69.2	94.0/94.6	95.7	78.8	100

Table 7: Results on SuperGLUE dev set comparing PaLM-540B few-shot and finetuned.

Specializing to code: Codex and AlphaCode

- Codex is a 12B parameter model that starts from a (smaller) GPT-3 and finetunes on 159GB of code from Github
- This model is what powers Github Copilot: https://copilot.github.com/
- AlphaCode scales up capabilities to competition level coding, achieving performance comparable to the median competitor
- Scaling up model size (41B) and dataset size (715GB), changing the model architecture to be an encoder-decoder, fine tuning on competition code, and sampling/filtering many candidate solutions all help in scaling to this level

Limitations of (current) massive models

Challenge tasks

- A number of tasks still elude the largest models and may be beyond the reach of simply making the models even bigger
  - Challenge sets designed to “stress test” models, e.g., ANLI, still have significant room for improvement, and scaling up is making slow progress
  - Hard tasks, such as generating solutions for high school math competition problems, also have very low accuracy even for the largest models
- This is even after these models are trained / fine tuned with more text/code/math than a human will ever see in their lifetime, so it seems like something is missing

Potential harms and biases

- Papers about large models now typically come with a model card describing its details and intended uses, along with some analysis about potential harms
- For example, GPT-3 was analyzed for gender, race, and religion biases, and this shed light on its predispositions that (unfortunately) seem in line with its training
  - This analysis has also been carried out for the three other models mentioned
- Analysis on Gopher (and, to an extent, PaLM) demonstrates that large models, when given a toxic prompt, are more likely to generate toxic continuations
- These concerns, and more, have to be carefully studied and mitigated before deploying such models into sensitive applications

Summary: Massive Models

- Massive models represent another potential paradigm shift within machine learning: pretraining + fine tuning was one such shift over the last ~10 years, but now perhaps we don't even need to do fine tuning anymore!
- In some ways, this may be more accessible (fewer data/expertise requirements); in other ways, this may be less accessible (compute requirements, privatization)
- Massive models still have problems in which they struggle, and they are primarily language models at the moment, but this may all change over the next few years
- As these models continue to proliferate, careful auditing of the potential benefits vs. potential harms will be needed to truly understand their full impact