

# Scaling Laws (for Language Models)

CSE 493G/599G Recitation

# Course logistics

**A3: Due Sunday May 11**

**Exam: On May 20**

# Today's story, in three parts

1. Why did people start to care about scaling?

# Today's story, in three parts

1. Why did people start to care about scaling?
2. What does a research question about scaling look like?

# Today's story, in three parts

1. Why did people start to care about scaling?
2. What does a research question about scaling look like?
3. How do we answer those questions?

# Today's story, in three parts

1. Why did people start to care about scaling?
2. What does a research question about scaling look like?
3. How do we answer those questions?

 **What does the thought process behind empirical deep learning research look like, and how does this research affect real practice?**

# Part 1: Early Scaling (why do we care?)

# Time travelling back to 2019

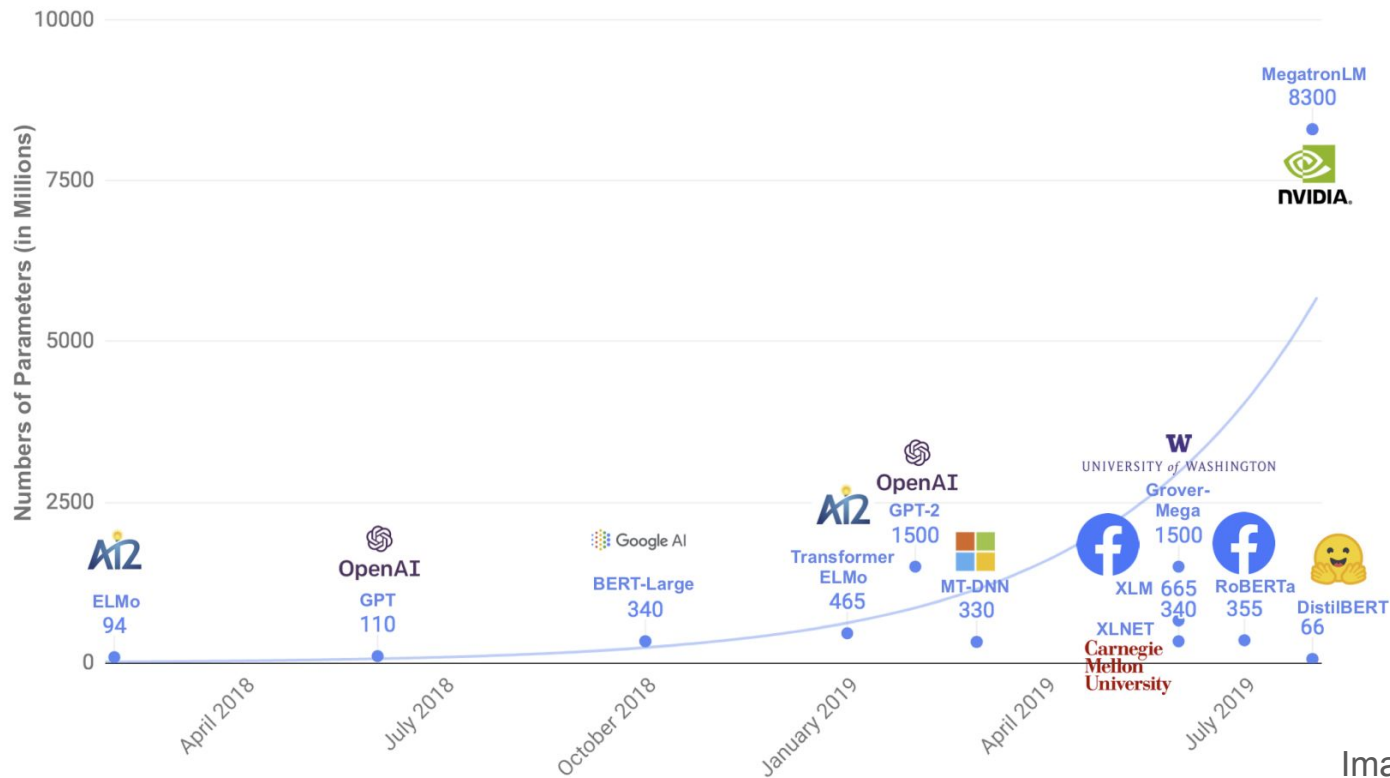
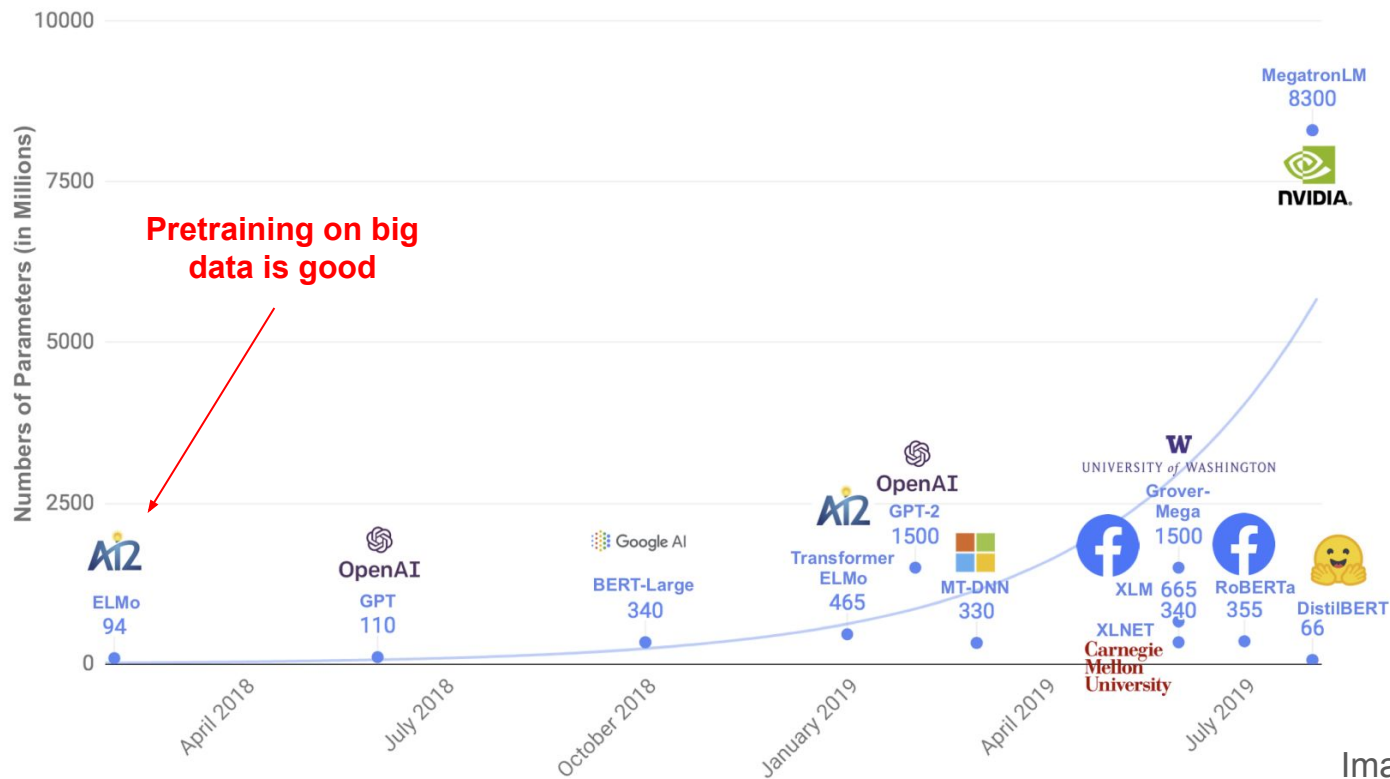


Image from DistillBert.



# Time travelling back to 2019



# Time travelling back to 2019



Image from DistillBert.

# Time travelling back to 2019

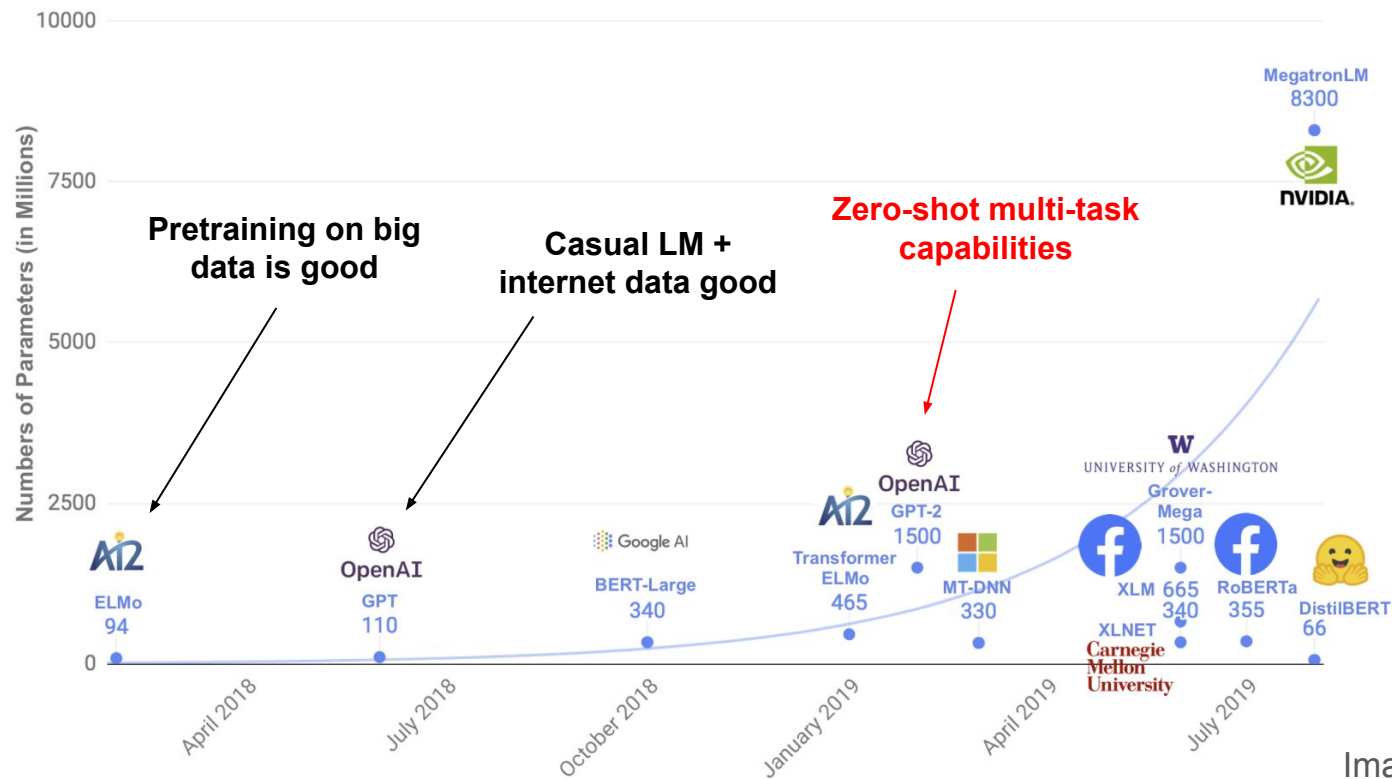
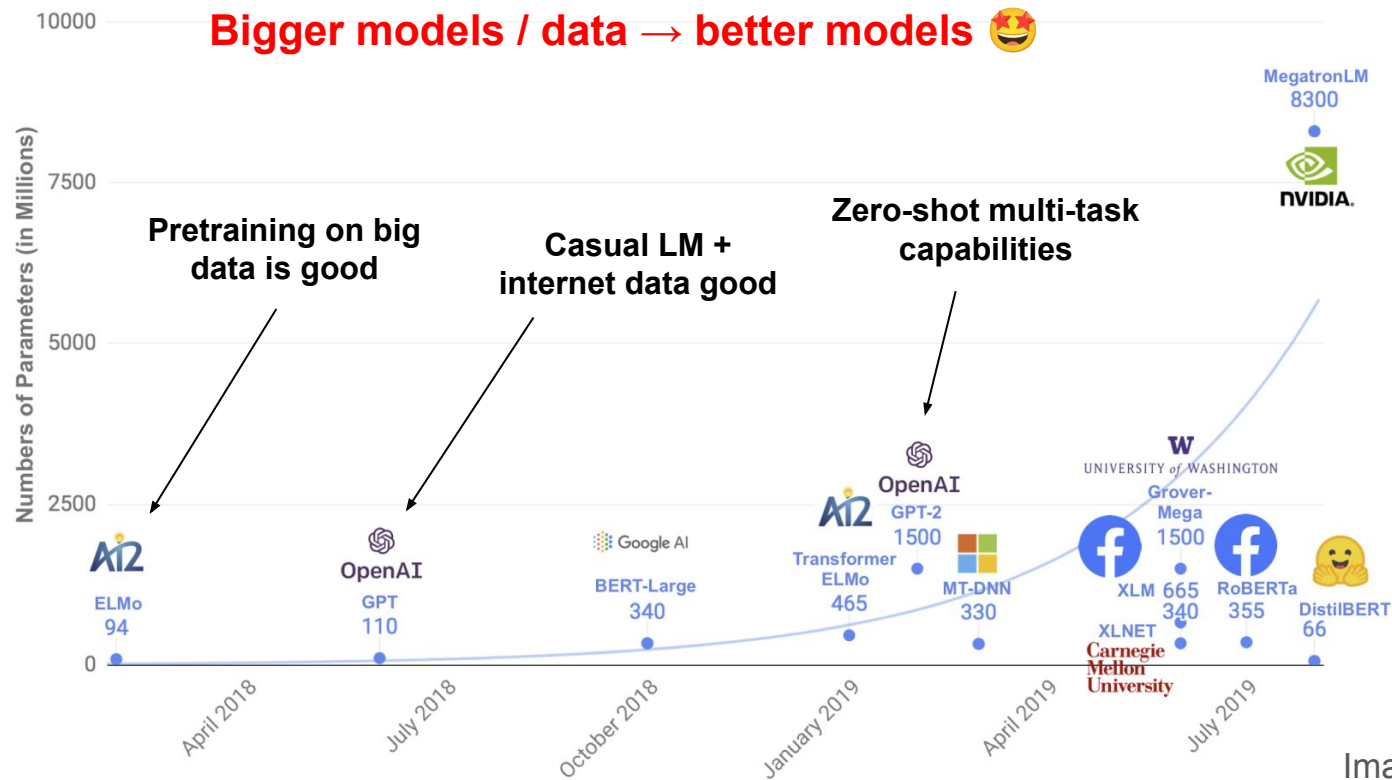


Image from DistillBert.

# Time travelling back to 2019

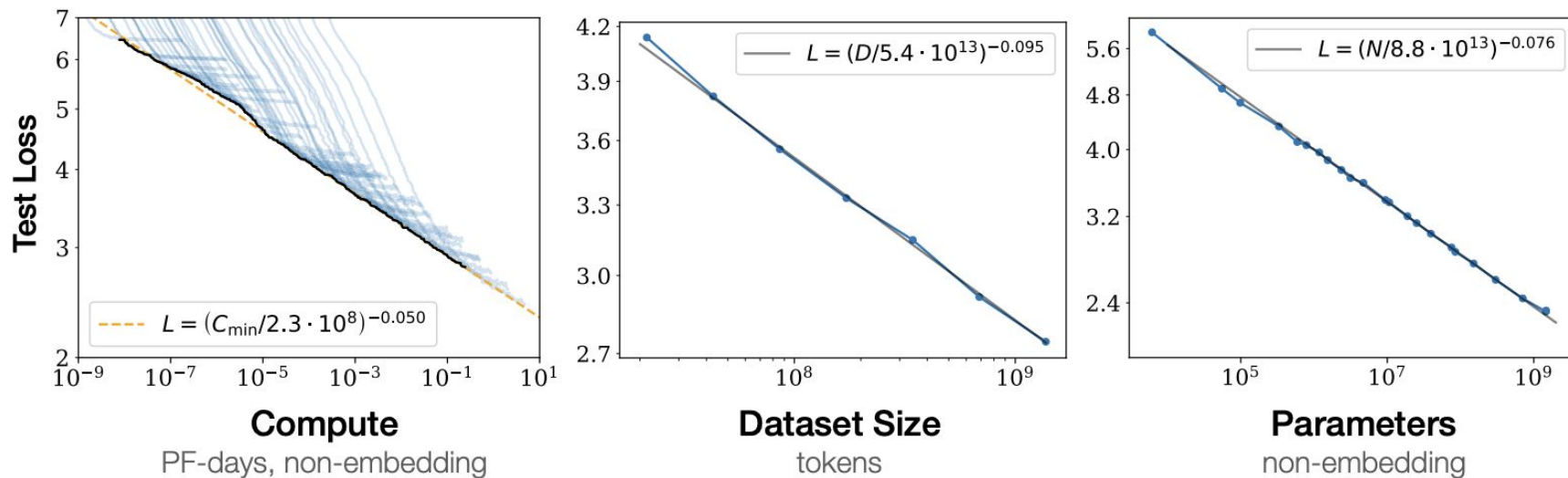


To what degree is ✨**scale**✨ predictive of downstream model performance?

Scale  $\approx$  training compute  $\approx$  dataset size \* model size

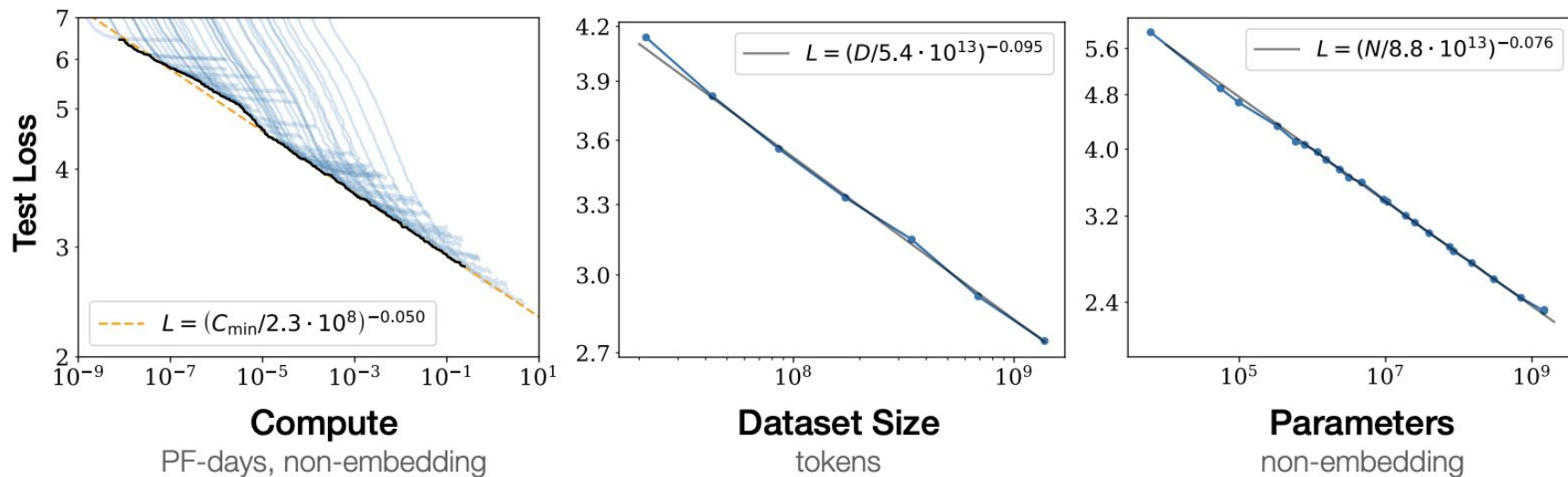
To what degree is ✨scale✨ predictive of downstream model performance?

# Scaling Laws for Neural Language Models



**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

# Scaling Laws for Neural Language Models



**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

**Takeaway: Consistent positive correlation between scale and lower error (improved performance)**



# 2020: now models are really really big

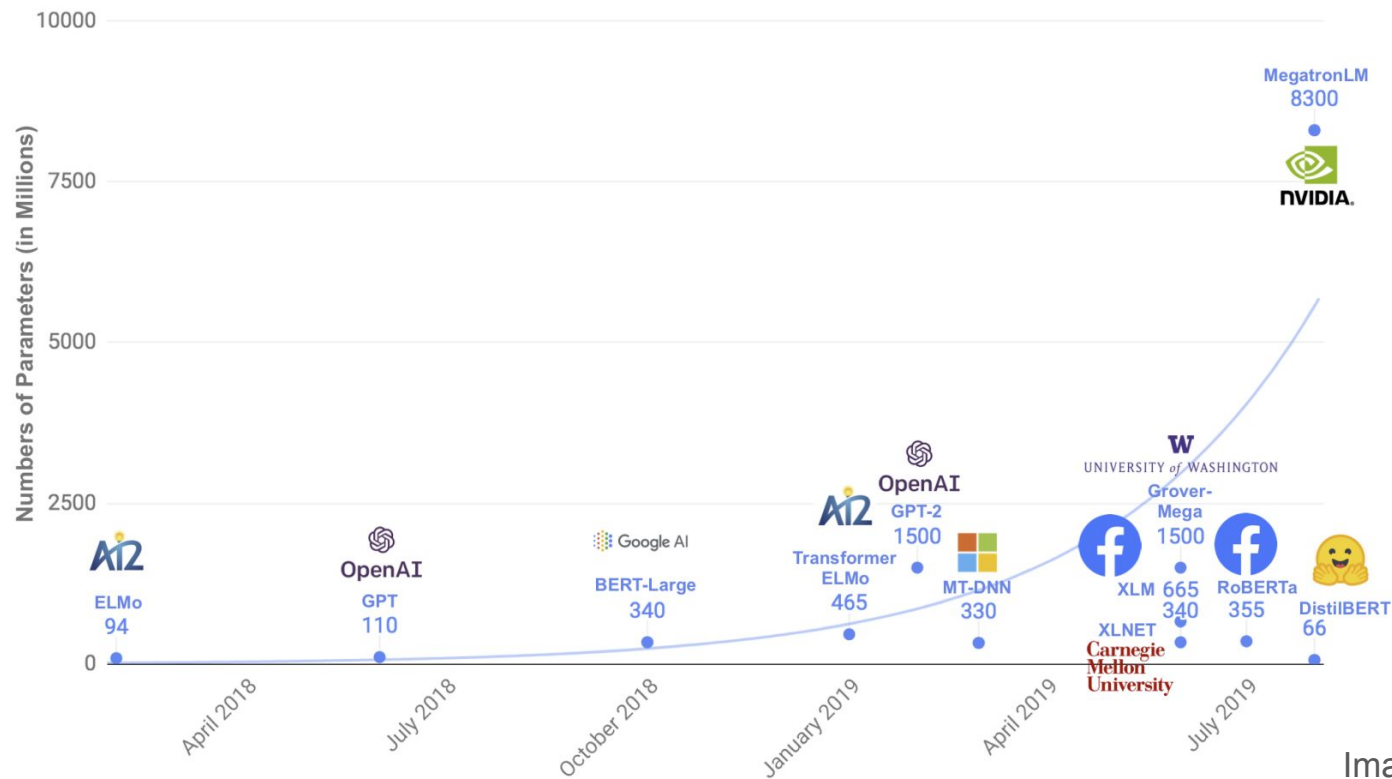
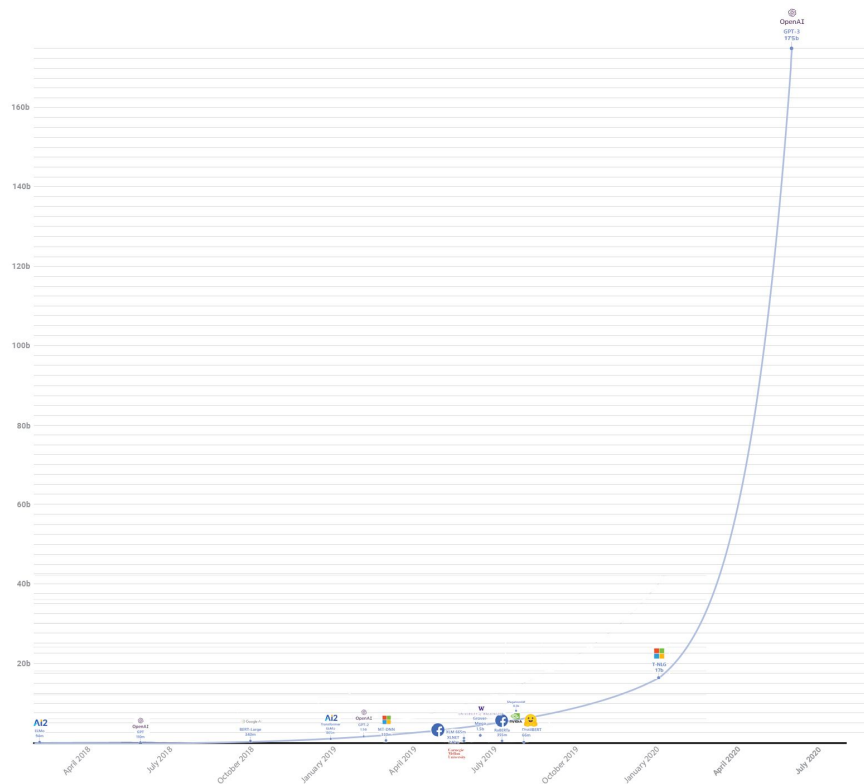


Image from DistillBert.

# 2020: now models are really really big



**GPT-3, 175B**

## Part 2: Increasing Scaling Efficiency (what questions do we ask?)

# Models are now reallllly expensive



\$10M+ for a single training job 🤪💰



100+ lifetime CO2 for one model 🔥

# Models are now reallllly expensive



\$10M+ for a single training job 💰



100+ lifetime CO2 for one model 🔥

**How can we scale up our training as efficiently as possible?**

# Deep dive: compute-optimal scaling laws

$$\mathbf{C} \sim 6\mathbf{ND}$$

**C = Compute (Floating Point Operations)**

**N = Dataset Size (Training Tokens)**

**D = Model Size (Parameter Count)**

# Side note: where does 6ND come from?

- See slides 13-16 of

<https://www.cs.princeton.edu/courses/archive/fall22/cos597G/lectures/lec12.pdf>

# Deep dive: compute-optimal scaling laws

$$C \sim 6 N^{\frac{1}{3}} D^{\frac{2}{3}}$$

**C = Compute (Floating Point Operations)**

**N = Dataset Size (Training Tokens)**

**D = Model Size (Parameter Count)**



# Deep dive: compute-optimal scaling laws

$$C \sim 6ND$$

**C = Compute (Floating Point Operations)**

**N = Dataset Size (Training Tokens)**

**D = Model Size (Parameter Count)**

# Deep dive: compute-optimal scaling laws

$$C \sim 6ND$$

**Given a fixed compute budget  $C$ , how much should we allocate towards  $N$  and  $D$ ?**

**$C$  = Compute (Floating Point Ops)  
 $N$  = Dataset Size (Training Tokens)  
 $D$  = Model Size (Parameter Count)**

## Part 3: Computing scaling laws (how do we answer our research questions?)

# Approach 1: Loss Envelope

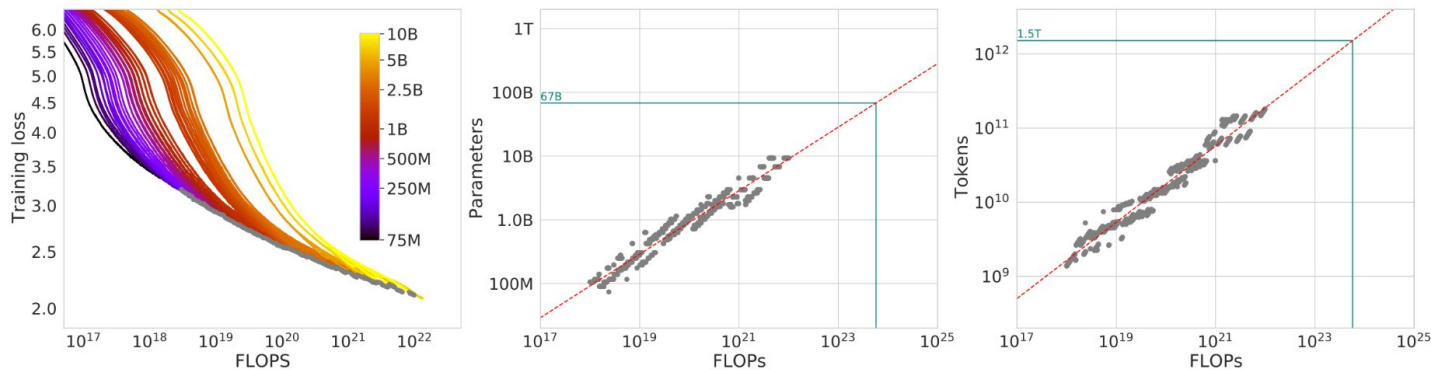


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* ( $5.76 \times 10^{23}$ ).

# Approach 1: Loss Envelope

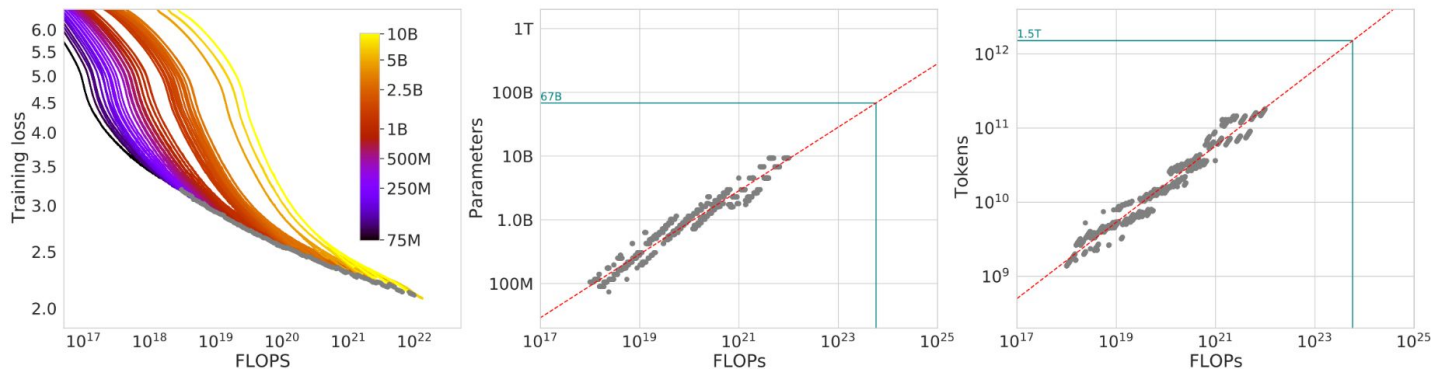


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* ( $5.76 \times 10^{23}$ ).

**Train varying sized models on varying amounts of data, track lowest loss achieved across all configs at distinct amounts of compute (measured in FLOPs) used. This is called the envelope. Fit a trend line from “compute used” to “optimal data” / “optimal model” sizes.**

# Approach 1: Loss Envelope

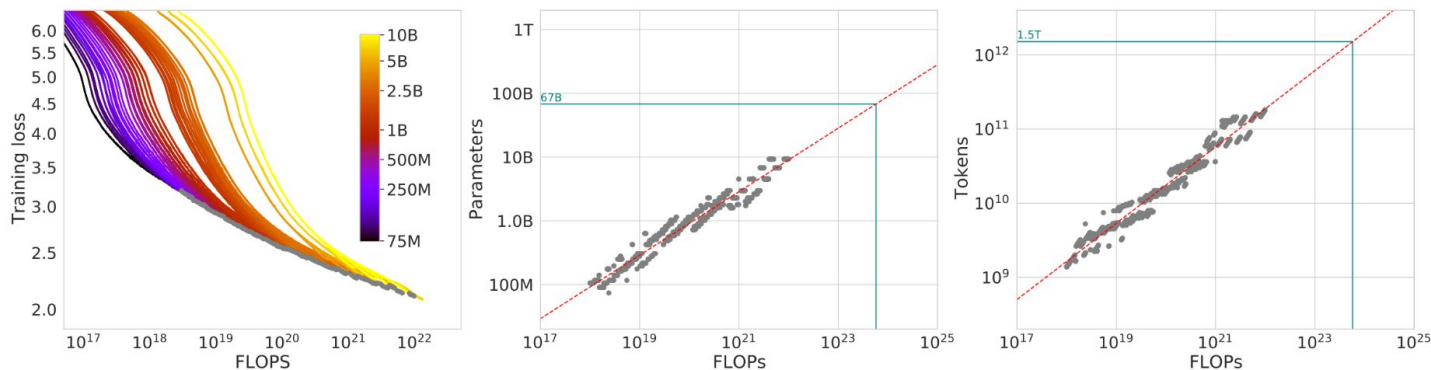


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* ( $5.76 \times 10^{23}$ ).

**Takeaway: dataset size and model size scale equally with additional compute – both trend lines (middle and left figure) have the same slope.**

# Approach 2: IsoFLOP curves

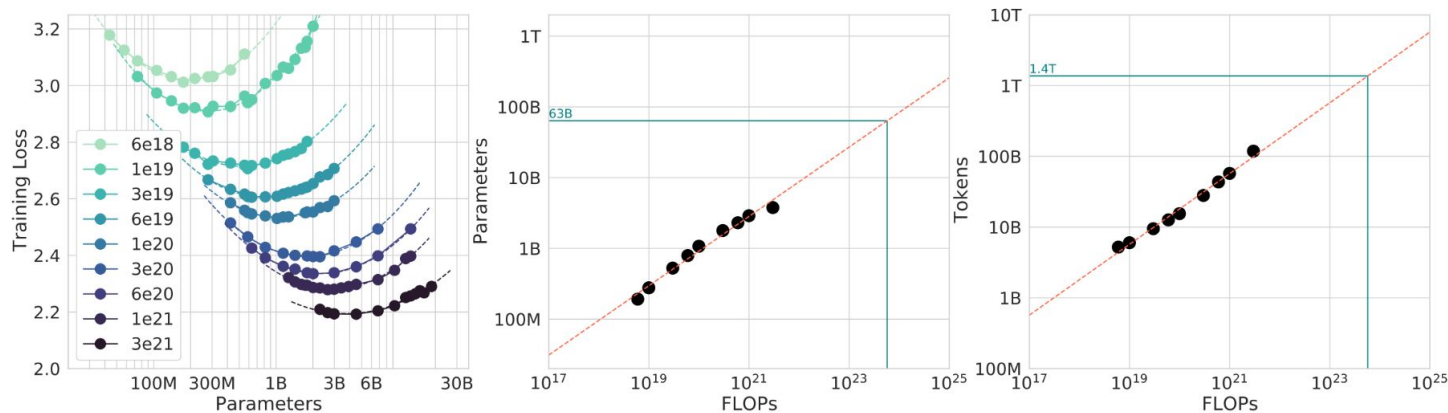


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

For a few different model sizes, try different training dataset sizes so that the total compute used is fixed. Plot a trend line from “FLOPs used” to “optimal model” / “optimal data” size.

# Approach 2: IsoFLOP curves

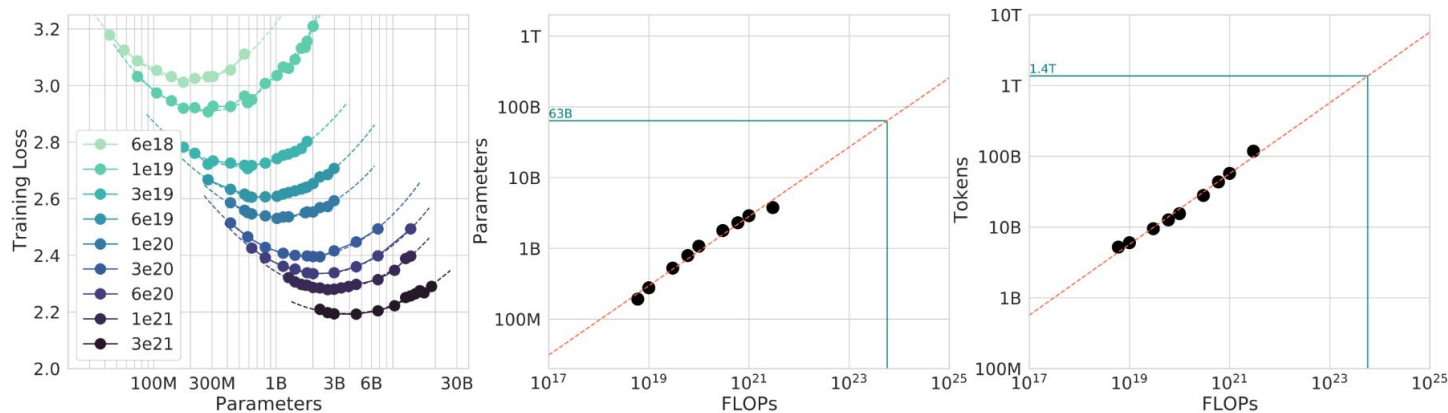


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

**Takeaway: dataset size and model size scale equally with additional compute – both trend lines (middle and left figure) have the same slope.**




# Approach 3: Parametric model

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

The first term captures the loss for an ideal generative process on the data distribution, and should correspond to the entropy of natural text. The second term captures the fact that a perfectly trained transformer with  $N$  parameters underperforms the ideal generative process. The final term captures the fact that the transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution.

# Approach 3: Parametric model

Test loss: how uncertain is my model at modeling text? Empirically measured.


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

The first term captures the loss for an ideal generative process on the data distribution, and should correspond to the entropy of natural text. The second term captures the fact that a perfectly trained transformer with  $N$  parameters underperforms the ideal generative process. The final term captures the fact that the transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution.

# Approach 3: Parametric model

$E, A, B, \alpha, \beta$  are parameters we will fit to observational data.

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

The first term captures the loss for an ideal generative process on the data distribution, and should correspond to the entropy of natural text. The second term captures the fact that a perfectly trained transformer with  $N$  parameters underperforms the ideal generative process. The final term captures the fact that the transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution.

# Approach 3: Parametric model

Intrinsic (irreducible) uncertainty of natural text

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

The first term captures the loss for an ideal generative process on the data distribution, and should correspond to the entropy of natural text. The second term captures the fact that a perfectly trained transformer with  $N$  parameters underperforms the ideal generative process. The final term captures the fact that the transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution.

# Approach 3: Parametric model

How much error do we incur from using a model with only finite parameters ( $N$ )?

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

The first term captures the loss for an ideal generative process on the data distribution, and should correspond to the entropy of natural text. The second term captures the fact that a perfectly trained transformer with  $N$  parameters underperforms the ideal generative process. The final term captures the fact that the transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution.

# Approach 3: Parametric model

How much error do we incur from training for only a finite number of steps ( $D$ )?

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

The first term captures the loss for an ideal generative process on the data distribution, and should correspond to the entropy of natural text. The second term captures the fact that a perfectly trained transformer with  $N$  parameters underperforms the ideal generative process. The final term captures the fact that the transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution.

# Approach 3: Parametric model

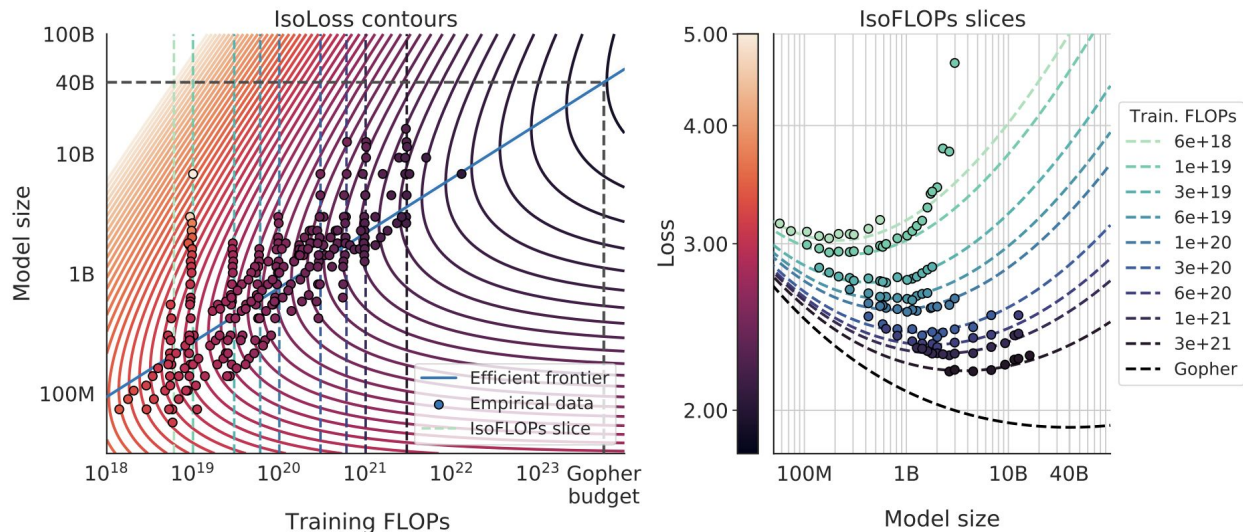


Figure 4 | **Parametric fit.** We fit a parametric modelling of the loss  $\hat{L}(N, D)$  and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a line in log-log space. Specifically, the curve goes through each iso-loss contour at the point with the fewest FLOPs. We project the optimal model size given the *Gopher* FLOP budget to be 40B parameters.

# Approach 3: Parametric model

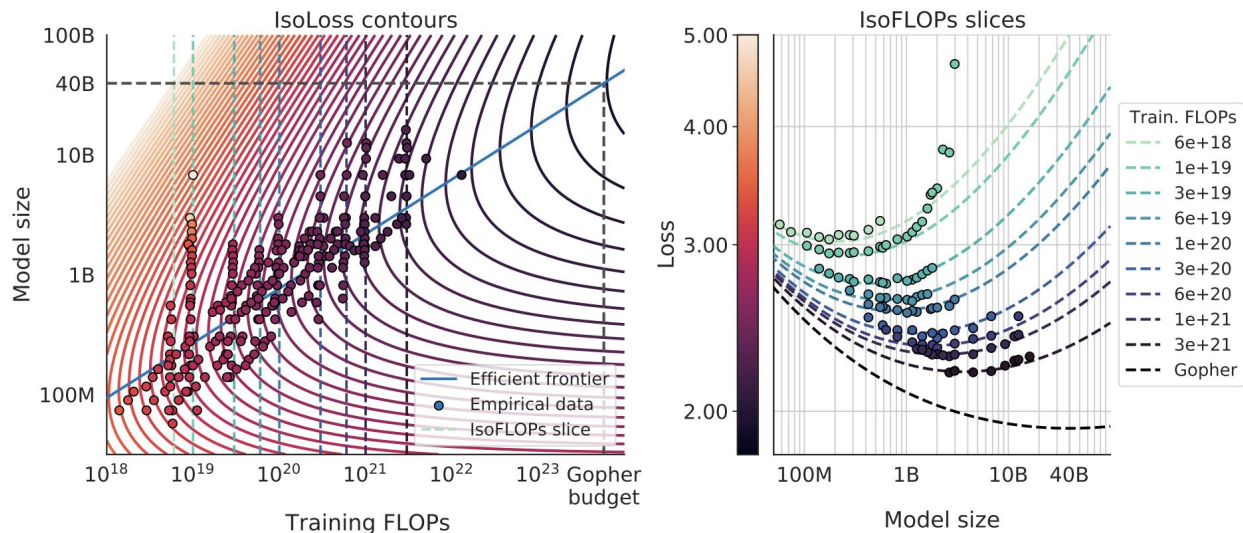
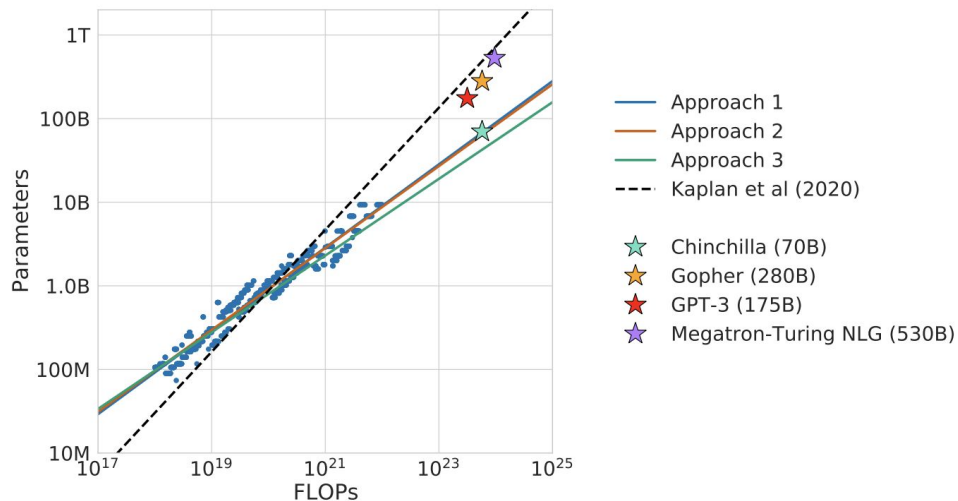


Figure 4 | **Parametric fit.** We fit a parametric modelling of the loss  $\hat{L}(N, D)$  and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a line in log-log space. Specifically, the curve goes through each iso-loss contour at the point with the fewest FLOPs. We project the optimal model size given the *Gopher* FLOP budget to be 40B parameters.

**Takeaway: dataset size and model size scale equally with additional compute**

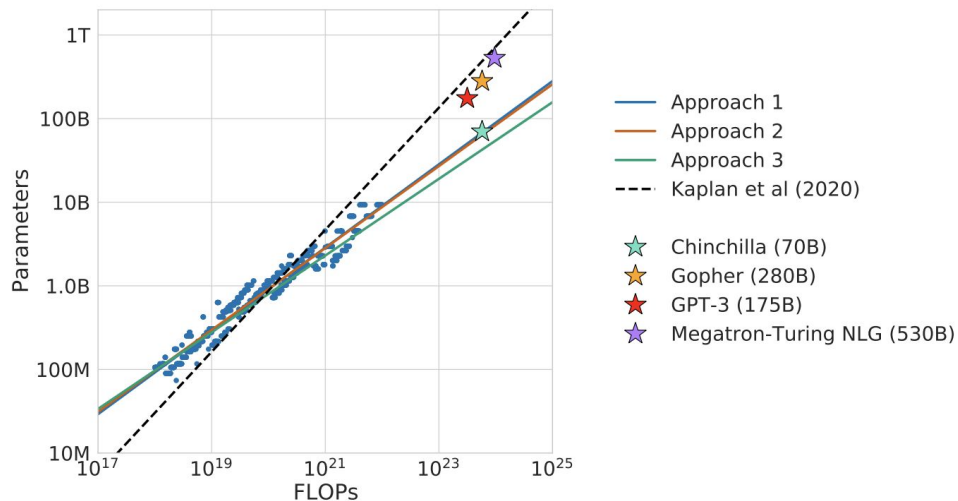


# Compute optimal scaling.



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

# Compute optimal scaling.



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

**Train smaller models for longer → “Chinchilla optimal”**

# Compute optimal scaling.

Parameters	FLOPs	FLOPs (in <i>Gopher</i> unit)	Tokens
400 Million	$1.92\text{e}+19$	1/29,968	8.0 Billion
1 Billion	$1.21\text{e}+20$	1/4,761	20.2 Billion
10 Billion	$1.23\text{e}+22$	1/46	205.1 Billion

Optimal scaling predicts that a 10B model should be trained on 205B tokens of natural language text

# Modern practice differs...

performance, a smaller one trained longer will ultimately be cheaper at inference. For instance, although [Hoffmann et al. \(2022\)](#) recommends training a 10B model on 200B tokens, we find that the performance of a 7B model continues to improve even after 1T tokens.

Llama trains with much more data than the “Chinchilla-optimal” amount

# Modern practice differs...

performance, a smaller one trained longer will ultimately be cheaper at inference. For instance, although [Hoffmann et al. \(2022\)](#) recommends training a 10B model on 200B tokens, we find that the performance of a 7B model continues to improve even after 1T tokens.

Llama trains with much more data than the “Chinchilla-optimal” amount

**Lifetime Compute  $\sim$  <sub>6ND</sub> + Inference Costs**

# Filling in gaps: language model scale reliably with overtraining and on downstream tasks.

