



JOHNS HOPKINS

WHITING SCHOOL  
of ENGINEERING



# Scaling Language Models

CSCI 601-471/671 (NLP: Self-Supervised Models)

<https://self-supervised.cs.jhu.edu/sp2024/>

# Scaling model size

- LM are getting larger and more expensive

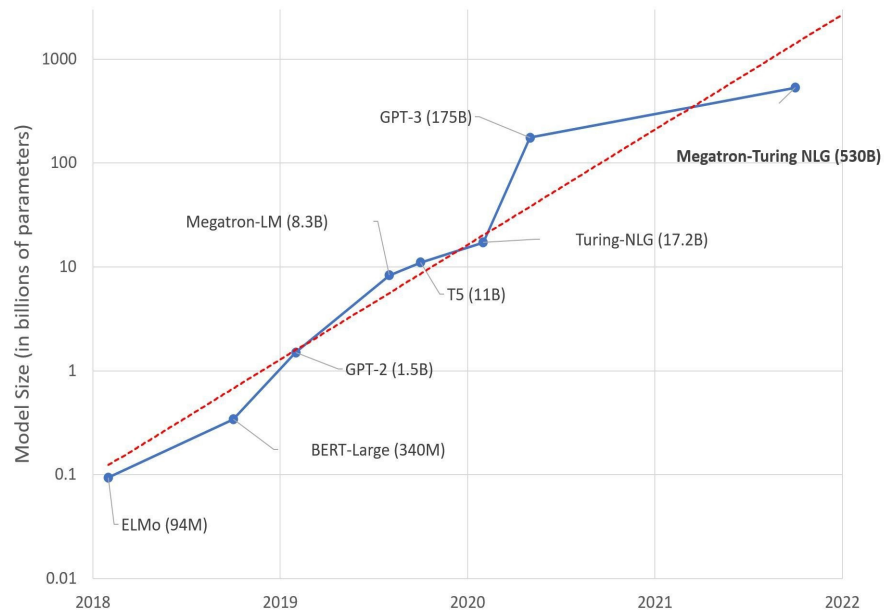
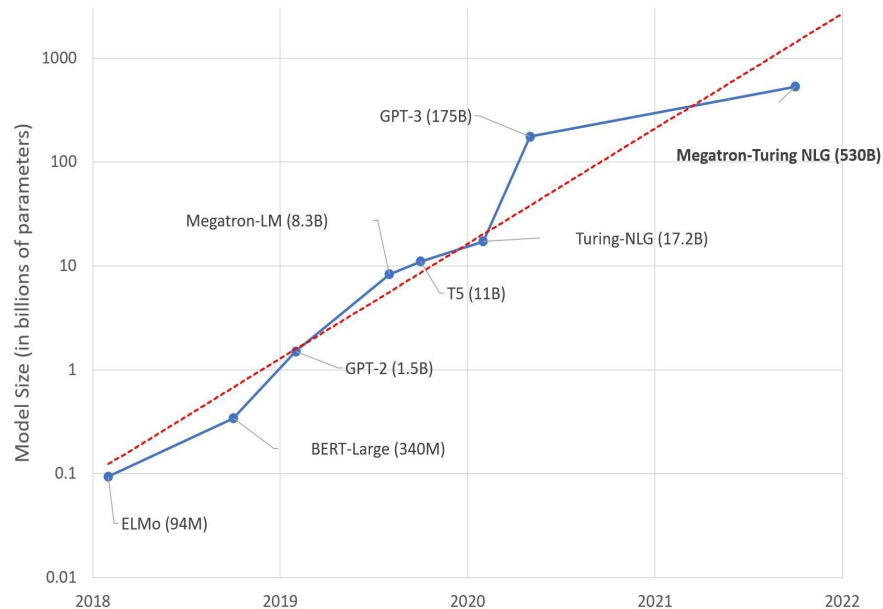
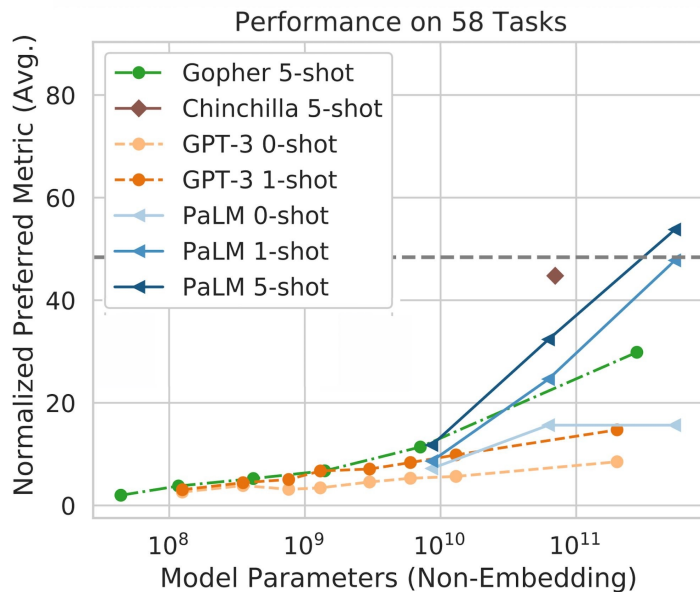


Photo credit: Microsoft Research Blog, Alvi et. al., 2021

# Model Size vs. Accuracy

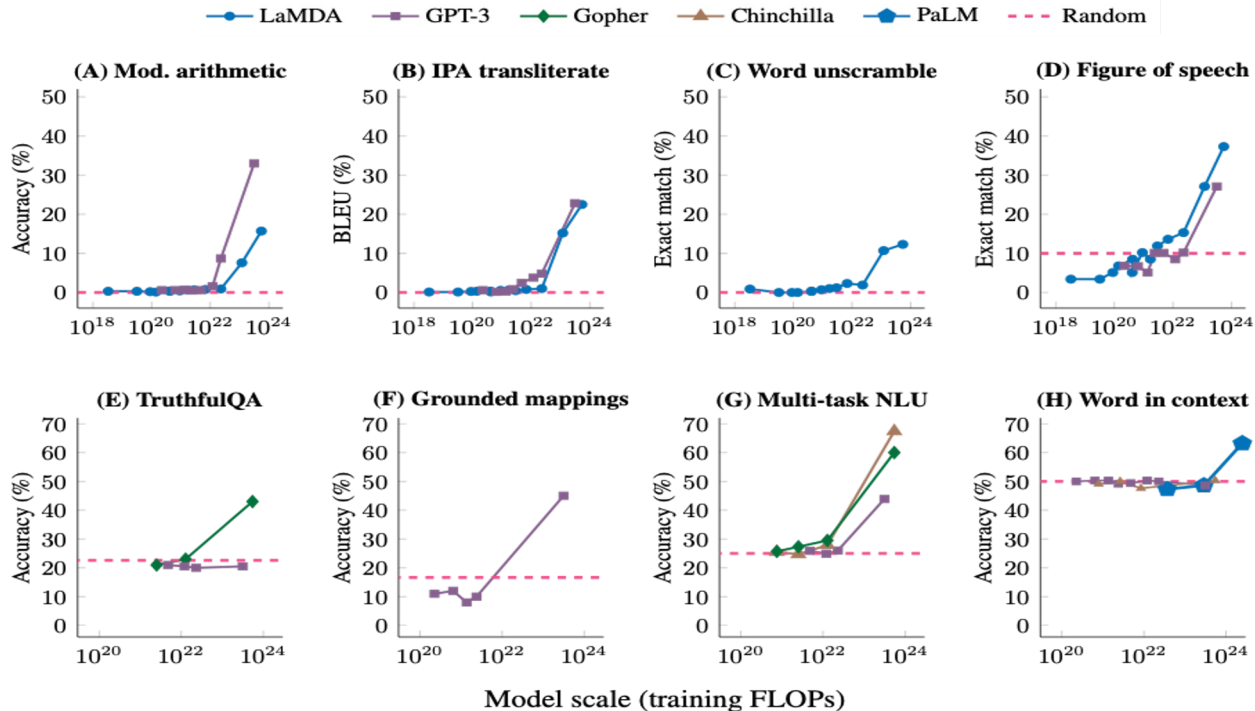
Photo credit: PaLM, Chowdhery et al., 2022



Larger LMs  $\Rightarrow$  better zero/few-shot performance

# Large Language Models Exhibit “Emergent” Abilities

- With scaling models their ICL perf consistency improves.



**Emergence** —qualitative changes in behavior with some “scaling” parameter

# "More is Different"

- The idea that complex physical systems can behave in ways that can't be understood by the laws that govern their microscopic parts.
- Anderson also gives an example of "More is Different" at the molecular level.
  - He describes a peculiar broken symmetry that appears in larger-scale molecules, which seems to go against a law defined at the smaller scale.
  - This broken symmetry is a new effect that appears when the scale changes.
- Anderson argues that new properties appear at each level of complexity.
  - For example, although chemistry is subject to the laws of physics, we can't infer chemistry from our knowledge of physics.

## More Is Different

Broken symmetry and the nature of the hierarchical structure of science.

P. W. Anderson

The reductionist hypothesis may still be a topic for controversy among philosophers, but among the great majority of active scientists I think it is accepted

planation of phenomena in terms of known fundamental laws. As always, distinctions of this kind are not unambiguous, but they are clear in most cases. Solid state physics, plasma physics, and perhaps

less relevance they seem to have to the very real problems of the rest of science, much less to those of society. The constructionist hypothesis breaks down when confronted with the twin difficulties of scale and complexity. The behavior of large and complex aggregates of elementary particles, it turns out, is not to be understood in terms of a simple extrapolation of the properties of a few particles. Instead, at each level of complexity entirely new properties appear, and the understanding of the new behaviors requires research which I think is as fundamental in its nature as any other. That is, it seems to me that one may array the sciences roughly linearly in a hierarchy, according to the idea: The elementary entities of science X obey the laws of science Y.

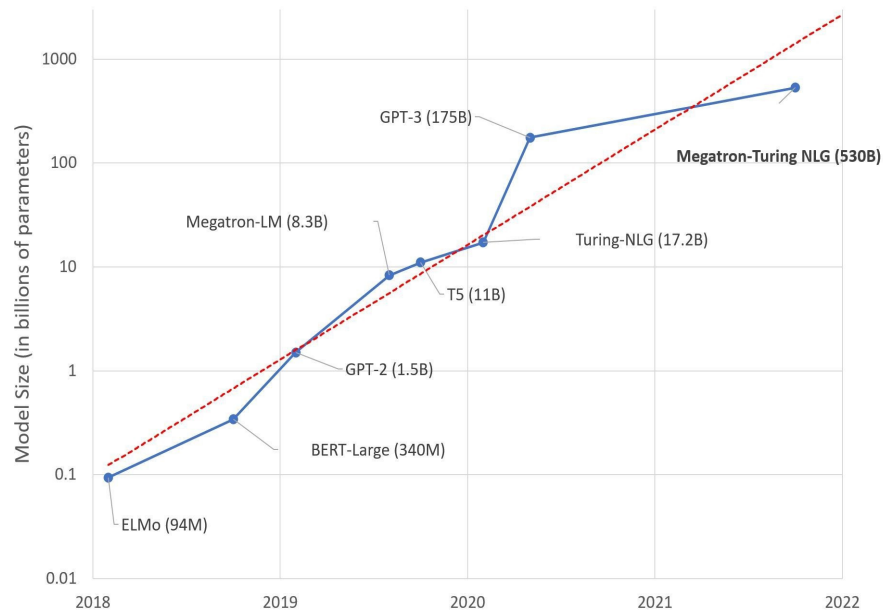


# What is “Scaling”?

- “scaling means **larger model size**”
  - But model parameters may be under-utilized.
- “scaling means **more compute**”
  - But computation may be unnecessarily wasted.
- “scaling means **more data**”
  - But more data might not necessarily contain more information (e.g., duplications)
- Scale means all the above: *effective compression of information*
  - Requires model capacity
  - Requires compute
  - Requires large, rich data

# Constraints of Real World

- Even massive companies have their own constraints.
- Examples of constraints:
  - The total amount of data
  - The total computing budget.
  - Time
  - ....
- **Given a set of constraints, how do you choose which LM to train?**
  - Note, trial and error is wasteful.



# Scaling Laws

---

- **Hypothesis:** there are fundamental principles that govern effective scaling
- **Importance:** understanding these “laws” would allow us to find optimal models for a given data/compute budget.
- Think of Newton’s laws
  - Provide the basis for understanding and analyzing the motion of objects in the physical world
  - Can be used to calculate the trajectory of a rocket, the speed of a car, or the motion of a planet.




# Scaling Language Models: Chapter Plan


1. Scaling laws for computational cost of models
2. Optimal scaling of model size and pre-training data
3. Why didn't we scale earlier?
4. Is scale all you need? A discussion.

**Chapter goal:** Getting familiar with various ideas related to “scaling”.

# Computation Cost of Models



How do you compute computational cost of a single-layer NN with one matrix multiplication?



# FLOPS

---

- Floating point operations per second (FLOPS, flops or flop/s)
- Each FLOP can represent an addition, subtraction, multiplication, or division of floating-point numbers,
- The total FLOP of a model (e.g., Transformer) provides a basic approximation of computational costs associated with that model.

# FLOPS: Matrix Multiplication

- Matrix-vector multiplication are common in Self-Attention (e.g., QKV projection)
  - Requires  $2mn$  (2 x matrix size) operations for multiplying  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^n$
  - (2 because 1 for multiplication, 1 for addition)

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n \\ \vdots \\ A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n \end{bmatrix}$$

# FLOPS: Matrix Multiplication

- Matrix-vector multiplication are common in Self-Attention (e.g., QKV projection)
  - Requires  $2mn$  (2 x matrix size) operations for multiplying  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^n$
  - (2 because 1 for multiplication, 1 for addition)
- For multiplying  $A \in \mathbb{R}^{m \times n}$  and  $B \in \mathbb{R}^{n \times p}$ , one needs  $2mnp$  operations.
  - Again, 2 because of 1 for multiplication, 1 for addition
- Now this is just forward propagation in Backprop. What about the **backward** step?

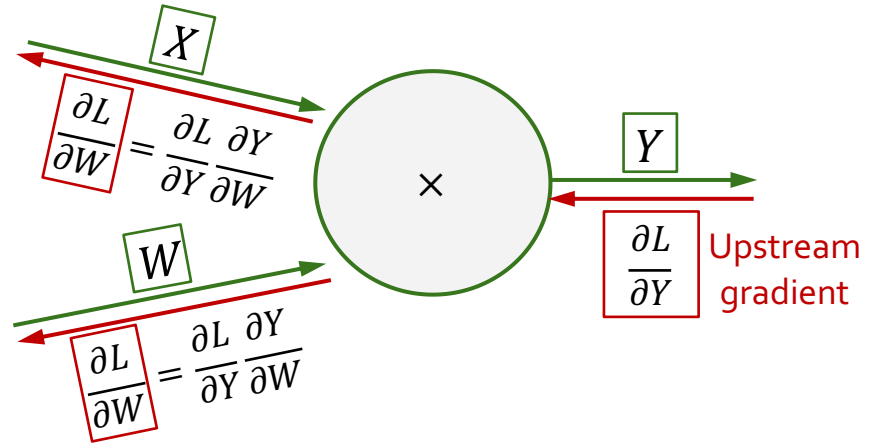
# FLOPS: Matrix Multiplication: Backward

- Backward pass needs to calculate the derivative of loss with respect to each hidden state and for each parameter

We also need  $\frac{\partial L}{\partial X}$  to continue to pass gradient to the previous layers.

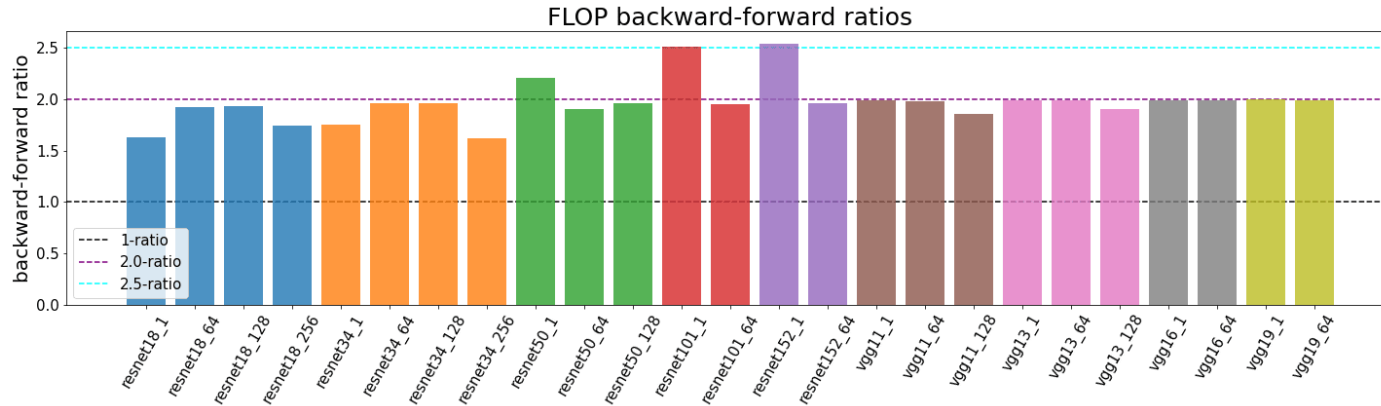
One matrix multiplication for  $\frac{\partial L}{\partial W}$

FLOPs for backward pass is roughly twice of forward pass.



# FLOPS: Matrix Multiplication: Backward

- FLOPs for backward pass is **roughly twice** of forward pass.
- Note that, this ratio depends on various parameters (architecture, batch size, et).



What's the backward-forward FLOP ratio for Neural Networks? LessWrong, 2021  
<https://www.lesswrong.com/posts/fnjKpBoWJXcSDwhZk/what-s-the-backward-forward-flop-ratio-for-neural-networks>



# FLOPS: Matrix Multiplication: Altogether

- Multiplying an input by a weight matrix requires 2x matrix size FLOPS.
- FLOPs for backward pass is **roughly twice** of forward pass.

Training FLOPs for multiplying by a matrix  $W = 6 \times (\text{batch size}) \times (\text{size of } W)$



# Computing the computational cost of Transformer



# Transformer FLOPs: The Quick Estimate

- The Weight FLOPs Assumption
  - The FLOPs that matter the most are weight FLOPs, that is ones performed when intermediate states are multiplied by weight matrices.
  - The weight FLOPs are the majority of Transformer FLOPs
  - We can ignore FLOPs for
    - Bias vector addition
    - layer normalization
    - residual connections
    - non-linearities
    - Softmax

# Transformer FLOPs: The Quick Estimate

- Let  $N$  be number of parameters (the sum of size of all matrices)
- Let  $D$  be the number of tokens in pre-training dataset.
- **Forward pass:**
  - FLOPs for forward pass on a single token is roughly  $2N$
  - FLOPs for forward pass for the entire dataset is roughly  $2ND$
- **Backward pass:**
  - FLOPs for backward pass is roughly twice of forward pass
  - FLOPs for backward pass for the entire dataset is roughly  $4ND$
- What is the total?

# Transformer FLOPs: The Quick Estimate

- Let  $N$  be number of parameters (the sum of size of all matrices)
- Let  $D$  be the number of tokens in pre-training dataset.
- The total cost of pre-training on this dataset is:

$$C \sim 6ND$$

- You can already see how this relates to our constraints:
  - If you have a fixed compute budget  $C$ , increasing  $D$  means decreasing  $N$  (and vice versa).

# Transformer Parameter Count

Non-embedding params

- One can show that:

$$N = 12 \times n_{layers} \times d_{model}^2 + (n_{vocab} + n_{pos}) \times d_{model}$$

Embedding params

- Assuming:
  - the size of MLP hidden layer to be  $4 \cdot d_{model}$
  - $n_{heads} \cdot d_{hidden} = d_{model}$
- You will prove this in homework assignment! 😊

Most Transformer LMs make these design assumptions.

# Transformer Parameter Count

- One can show that:

$n_{\text{layer}}$	$d_{\text{model}}$	Parameters ( $N$ )
4	512	13M
6	768	42M
10	1280	197M
16	2048	810M
24	3072	2.7B
40	5120	13B
64	8192	52B

Table from: A General Language Assistant as a Laboratory for Alignment, 2021

Non-embedding params

$$N = 12 \times n_{\text{layers}} \times d_{\text{model}}^2 + (n_{\text{vocab}} + n_{\text{pos}}) \times d_{\text{model}}$$

Embedding params

For example, see the models in the following table:

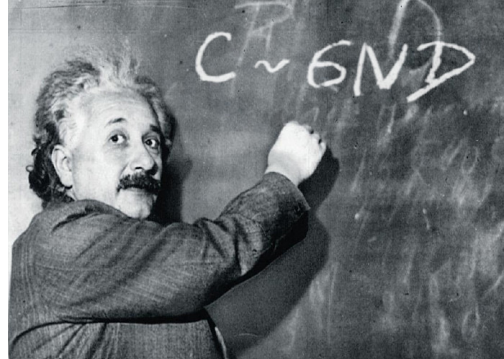
$$N_{\text{non-embedding}} = 12 \times 64 \times 8192^2 = 51.5B$$

Vocab size = 65536

Positional emb size = ?

$$N_{\text{embedding}} = (65536 + ?) \times 8192 = 0.5B + ?$$

# Transformer Parameter Count



- Given the pre-training data with 400B tokens.

$n_{\text{layer}}$	$d_{\text{model}}$	Parameters ( $N$ )	Training FLOPs
4	512	13M	3.0e19
6	768	42M	1.0e20
10	1280	197M	4.7e20
16	2048	810M	1.9e21
24	3072	2.7B	6.5e21
40	5120	13B	3.0e22
64	8192	52B	1.2e23

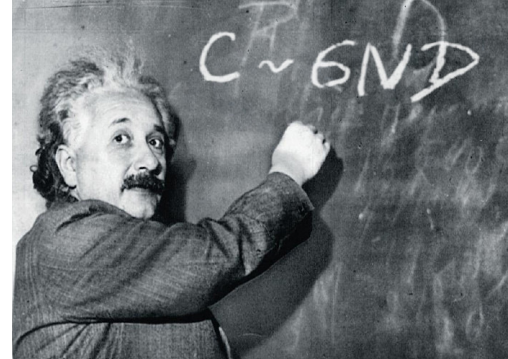
Training cost (FLOPs):

$$\begin{aligned}C &\approx 6ND \\ &= 6 \times (400 \times 10^9) \times (52 \times 10^9) \\ &= 1.24 \times 10^{23}\end{aligned}$$



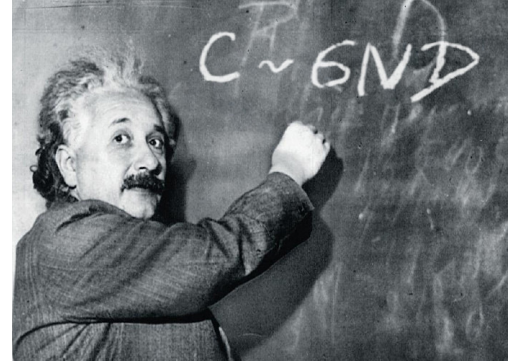
# Estimating training time

- This is a very practical question in real world.
- We will use our formula earlier to estimate training time.
- Consider HyperCLOVA, an 82B parameter model that was pre-trained on 150B tokens, using a cluster of 1024 A100 GPUs.



Intensive Study on HyperCLOVA: Billions-scale Korean Generative Pretrained Transformers, 2021

# Estimating training time



- Consider HyperCLOVA, an 82B parameter model that was pre-trained on 150B tokens, using a cluster of 1024 A100 GPUs.
- Training cost (FLOPs):

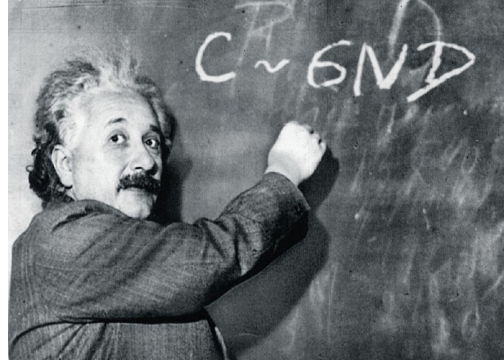
$$\begin{aligned}C &\approx 6ND \\ &= 6 \times (150 \times 10^9) \times (82 \times 10^9) = 7.3 \times 10^{22}\end{aligned}$$

- The peak throughput of [A100 GPUs](#) is 312 teraFLOPS or  $3.12 \times 10^{14}$ .
- **How long would this take?**

$$\text{Duration} = \frac{\text{model compute cost}}{\text{cluster throughput}} = \frac{7.3 \times 10^{22}}{3.12 \times 10^{14} \times 1024} = 2.7 \text{ days}$$

Intensive Study on HyperCLOVA: Billions-scale Korean Generative Pretrained Transformers, 2021

# Estimating training time



- How long would this take?

$$\text{Duration} = \frac{\text{model compute cost}}{\text{cluster throughput}} = \frac{7.3 \times 10^{22}}{3.12 \times 10^{14} \times 1024} = 2.7 \text{ days}$$

- According to the white paper, training took 13.4 days. Our estimate is 5 times off (why?), but we did get the order of magnitude right! 🙌

Intensive Study on HyperCLOVA: Billions-scale Korean Generative Pretrained Transformers, 2021

# Factors We Did Not Consider

- Note that these estimates can be slightly off in practice
  - Theoretical peak throughput is not achievable with distributed training. (unless your model only does large matrix multiplications).
  - We ignored many additional operations like softmax, ReLU/GeLU activations, self-attention, Layer Norm etc.
  - Training divergence and restarting from earlier checkpoints are not uncommon.
- There are various factors that contribute to computation latency
  - Communication latency, memory bandwidth, caching, etc.
  - See <https://kipp.ly/transformer-inference-arithmetic/> for an excellent discussion.

# Measuring FLOPS Empirically

- There are libraries for computing FLOPS
  - Example: <https://github.com/MrYxJ/calculate-flops.pytorch>

```
from calflops import calculate_flops_hf

batch_size, max_seq_length = 1, 128
model_name = "meta-llama/Llama-2-7b"
access_token = "" # your application for using llama

flops, macs, params = calculate_flops_hf(model_name=model_name,
                                       access_token=access_token,
                                       input_shape=(batch_size, max_seq_length))
print("%s FLOPs:%s MACs:%s Params:%s \n" %(model_name, flops, macs, params))
```

# Measuring FLOPS Empirically

- There are libraries for computing FLOPS
  - Example: <https://github.com/MrYxJ/calculate-flops.pytorch>

Model	Input Shape	Params(B)	Params(Total)	fwd FLOPs(G)	fwd MACs(G)	fwd + bwd FLOPs(G)
bloom-1b7	(1,128)	1.72B	1722408960	310.92	155.42	932.76
bloom-7b1	(1,128)	7.07B	7069016064	1550.39	775.11	4651.18
bloomz-1b7	(1,128)	1.72B	1722408960	310.92	155.42	932.76
baichuan-7B	(1,128)	7B	7000559616	1733.62	866.78	5200.85
chatglm-6b	(1,128)	6.17B	6173286400	1587.66	793.75	4762.97
chatglm2-6b	(1,128)	6.24B	6243584000	1537.68	768.8	4613.03
Qwen-7B	(1,128)	7.72B	7721324544	1825.83	912.88	5477.48
llama-7b	(1,128)	6.74B	6738415616	1700.06	850	5100.19
llama2-7b	(1,128)	6.74B	6738415616	1700.06	850	5100.19

# Summary

---

- One can measure the computational cost of training neural networks in terms of FLOPS.
- Such estimates allow you to estimate the training time of your model, given your GPU specs.
- What else can we do?

# Optimal Scaling



# Optimal Scaling

- **A real problem:** Your boss gives you a compute budget \$\$\$\$. What is the best model you can build with this budget?
- We know from the literature that **larger** models generally lead to **better** models.
  - Does that mean that you should aim to build the largest model possible?
- Intuitively, if you choose a model that is **too large** for your budget, you need to **cut your training** cycles that **may reduce its quality**.
- **This chapter:** principled approach to selecting optimal data/model scaling.

# Scaling

## Experimental Setup:

- Pre-train various models of different sizes
- Plot their validation loss throughout training

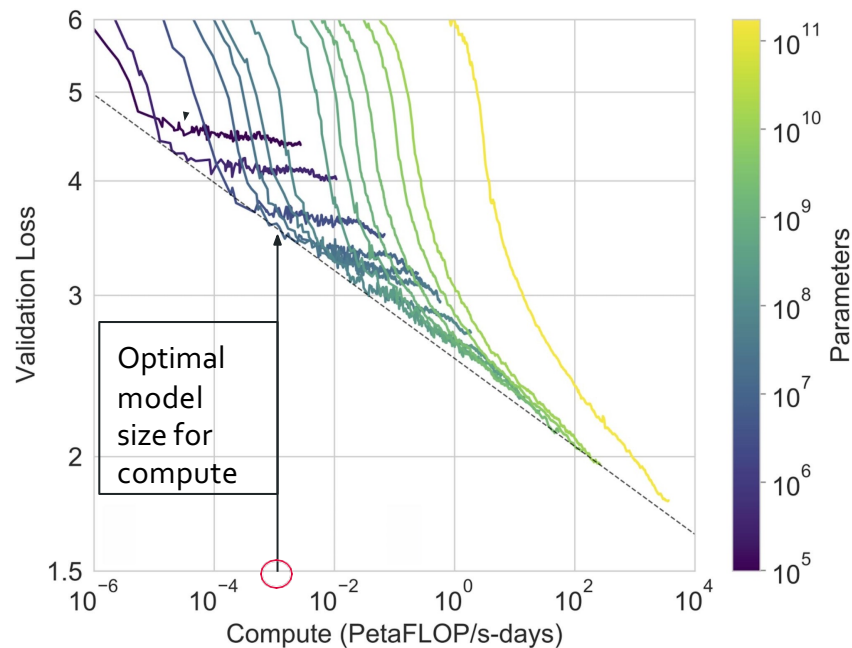


Photo credit: GPT3, Brown et. al., 2020

# Scaling

- **Smaller** models don't have enough capacity to utilize the extra compute. They plateau early.
- **Larger** models are initially slower to train, but with more compute they reach lower losses.

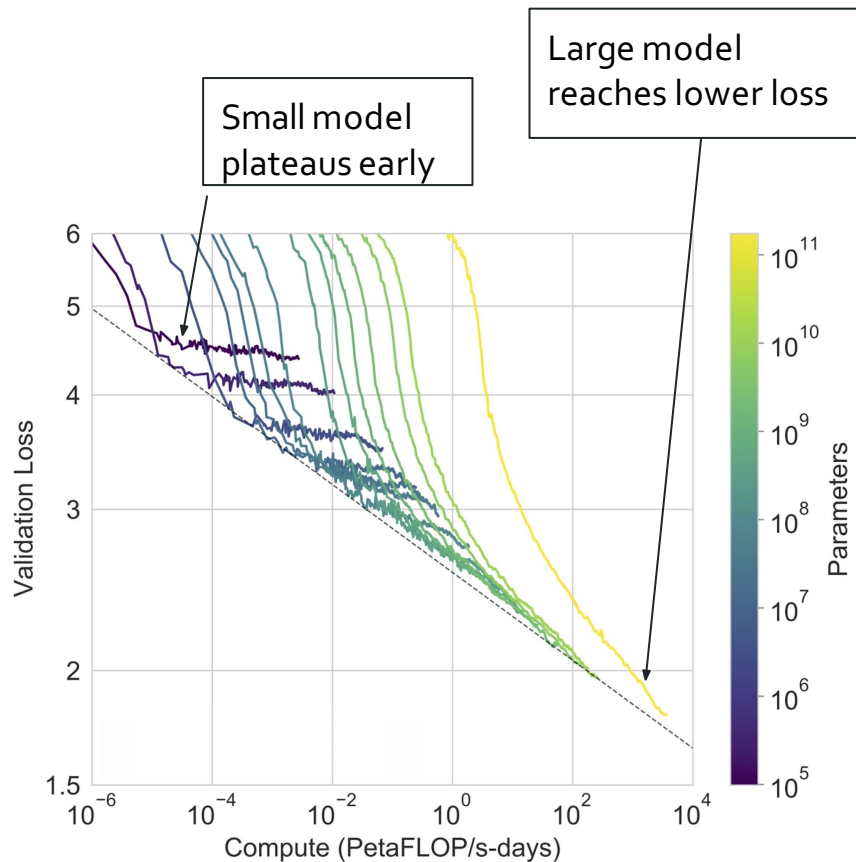


Photo credit: GPT3, Brown et. al., 2020

# Scaling - Optimal Model Size

- Let's say our compute budget is  $C = 10^{-2}$  PetaFLOPs-days.
- The **optimal model** is the one that plateaus at exactly  $C$ .
- If we train a **larger** model than optimality point, we won't reach the best performance.
- If we train a **smaller** model the performance wouldn't be optimal

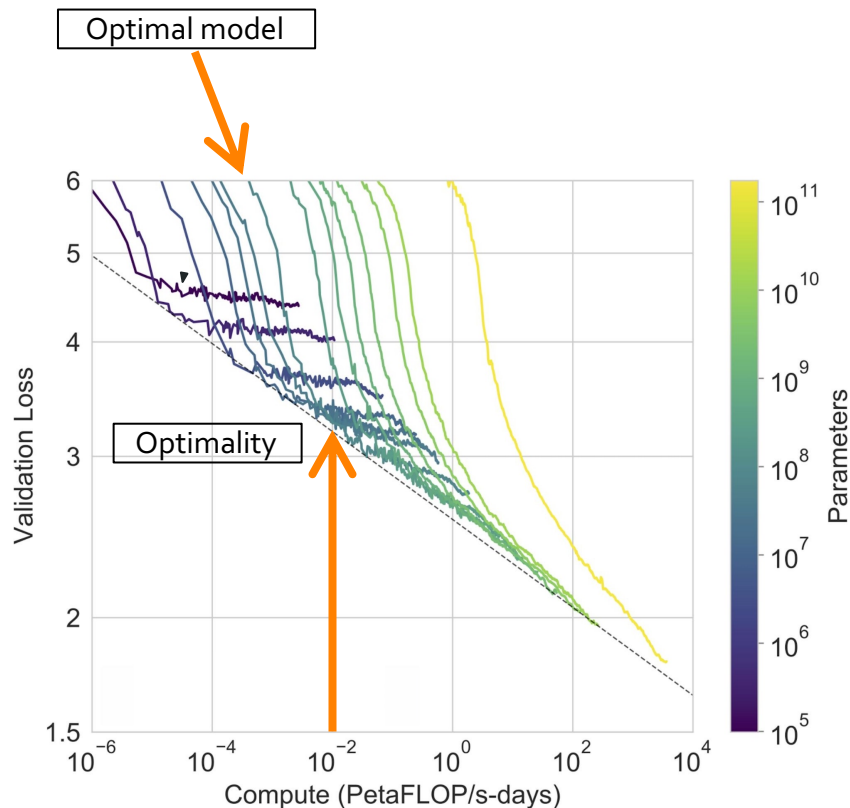


Photo credit: GPT3, Brown et. al., 2020

# Scaling - Optimal Model Size

- The idea of “optimal model size for given compute” was introduced by Kaplan et. al.
- In ideal world, we are given lots of compute to train many models to find the optimality.
- Alas not feasible when you have budget to train a single model.
- If we have the equations (“laws”) describing the behavior, we can compute it **analytically**.

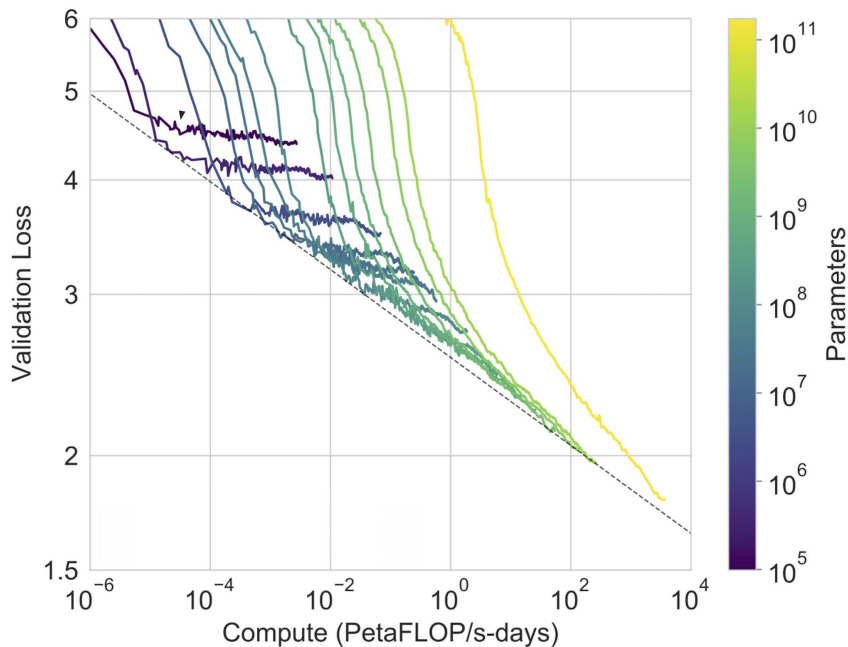


Photo credit: GPT3, Brown et. al., 2020

# Scaling - Optimal Model Size

- What is the function that describes this **optimality line**?

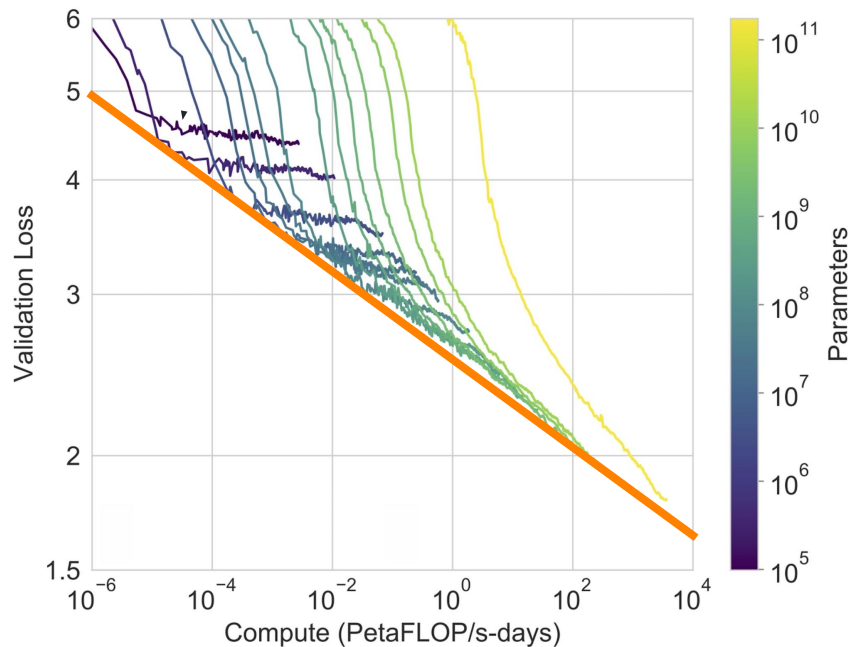


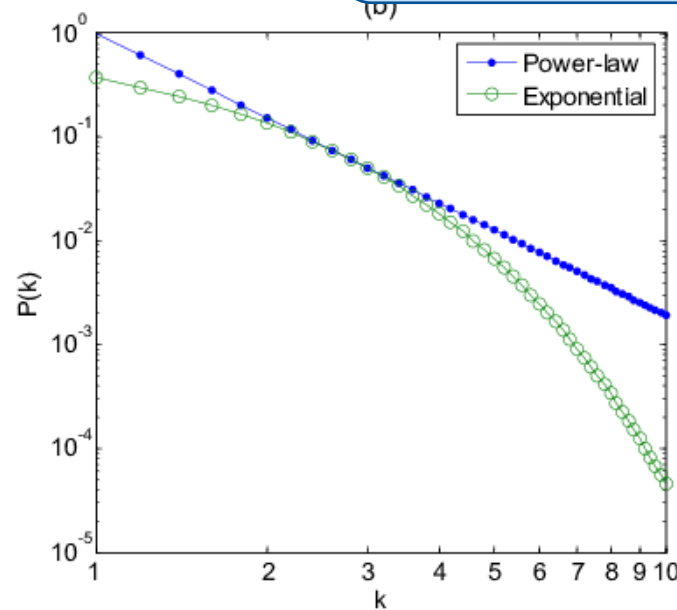
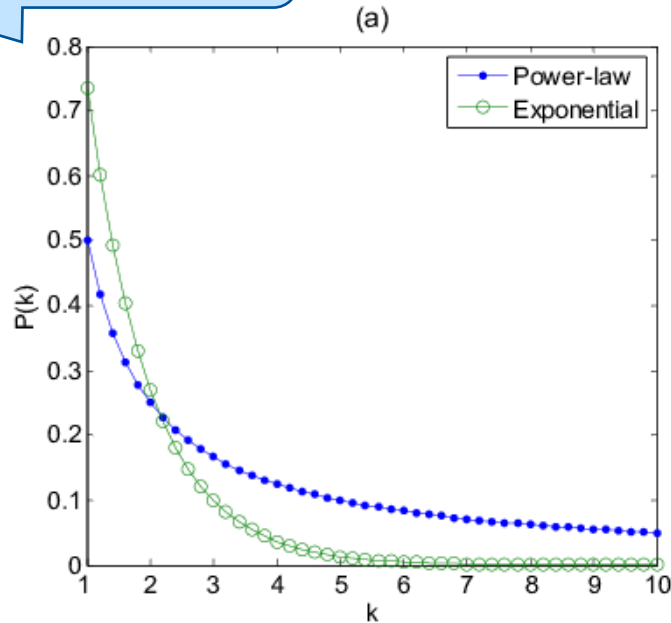
Photo credit: GPT3, Brown et. al., 2020

# Terminology: Power law vs exponential

Exponential goes to zero at a faster rate.

Power law = variable<sup>(constant)</sup>  
Exponential = (constant)<sup>variable</sup>

Power law trend looks linear, when the variable is shown in logarithmic scale.



# Scaling Laws of Kaplan et. al., 2020

- A power-law function predicts the “compute efficient” frontier:

$$L \propto C^{-0.48}$$

- Using this (and some other analysis not shown here) we can **analytically** predict the optimal model and data size, for a given amount of compute.

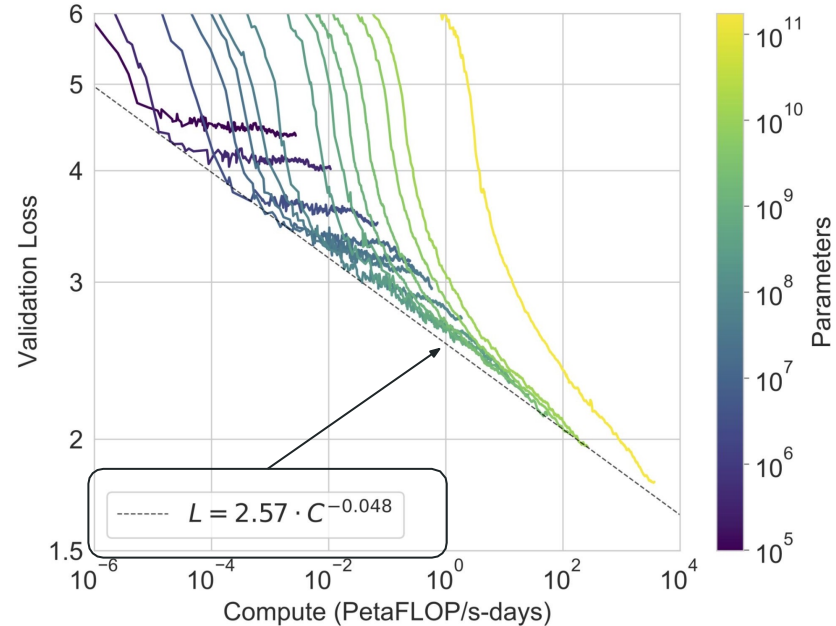


Photo credit: GPT3, Brown et. al., 2020



# Scaling Laws: Kaplan et al.

N: number of model parameters  
C: compute  
D: dataset size

- Optimal model size and optimal number of tokens, for a given compute budget

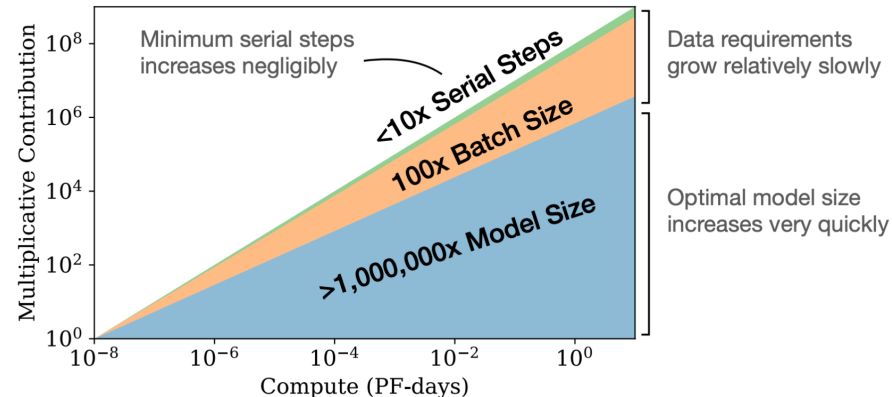
Kaplan et. al. 2020	$N_{\text{opt}} \propto C^{0.73}$	$D_{\text{opt}} \propto C^{0.27}$
---------------------	-----------------------------------	-----------------------------------

$N_{\text{opt}}$  exponent  $\gg$   $D_{\text{opt}}$  exponent

- Takeaway:** grow the model size faster than growing the number of tokens.
  - Example:** Given 10x compute, increase N by [MASK], and D by [MASK]
- GPT3 (and many other followed this recipe) training a 175B model on 300B tokens

# Recap: Scaling Laws (Kaplan et al.)

- It appears that there are Precise scaling laws predicting the performance of AI models based on
  - Model size: Number of params
  - Dataset size
  - Total compute used for training
- Scaling Laws: scale model size at a faster rate.
- Given a 10x increase in compute budget,
  - increase the size of the model by 5.5x, and training data size by 1.8x.



# Implications of Scaling Laws (Kaplan et al.)

- Subsequent papers tried to engineer larger and larger models

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
<i>Gopher</i> (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion

# However ...

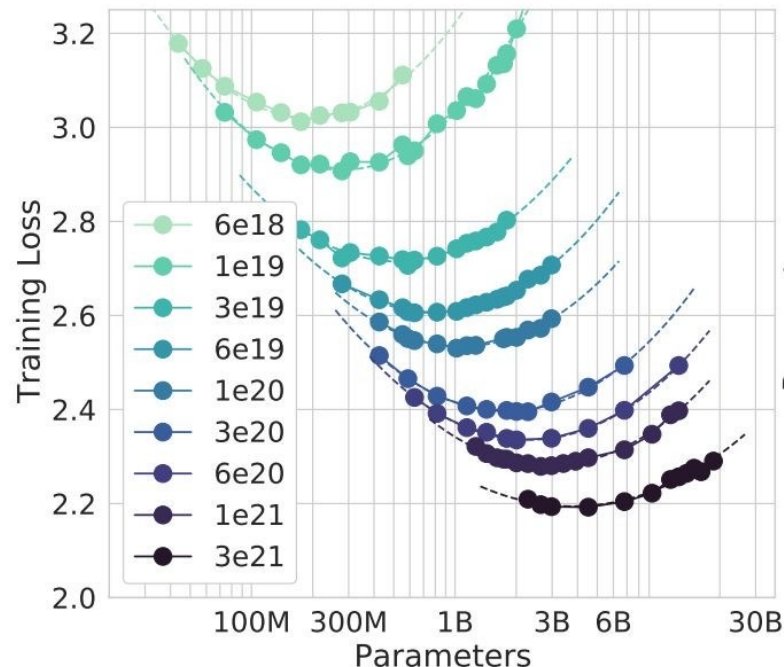
---

- In 2022 a Hoffmann et al. from DeepMind showed a different set of scaling laws.

# Scaling - Kaplan et al. vs. Hoffmann et al.

## Experimental setup:

- They chose different compute budgets.
- For each compute budget, train different sized models (varying data or model size)
- They find a clear valley like shape
- For each compute budget there is an optimal model to train



# Scaling Laws: Hoffmann et al.

N: number of model parameters  
C: compute  
D: dataset size

- Optimal model size and optimal number of tokens, for a given compute budget

Kaplan et. al. 2020	$N_{\text{opt}} \propto C^{0.73}$	$D_{\text{opt}} \propto C^{0.27}$
Hoffmann et. al., 2021	$N_{\text{opt}} \propto C^{0.5}$	$D_{\text{opt}} \propto C^{0.5}$

$$N_{\text{opt}} \text{ exponent} \cong D_{\text{opt}} \text{ exponent}$$

- Compute and tokens should increase **at the same rate**.
  - Example 1:** Given 10x compute, grow N by 3.2x and D by 3.2x
  - Example 2:** Given 100x compute, grow N by [MASK] and D by [MASK]

# Recap

- We used to train “oversized” and “under-trained” models.
- You should scale your model at the same rate as your data.
- For example, if you get a 100x increase in compute,
  - you should make your model 10x bigger and your data 10x bigger.

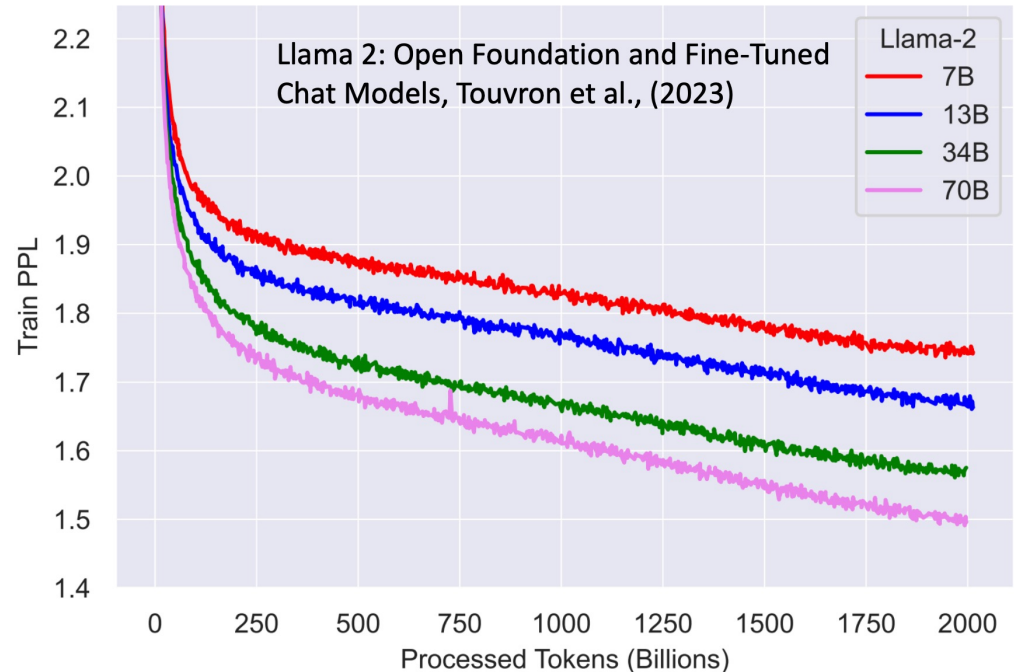
# A Word of Caution

- While we kept referring to these as “law”, one should take them with grain of salt.
- There are various confounding factors here:
  - Different optimizer: AdamW vs. Adam vs. others
  - Different tokenizers
  - Different numerical representation (e.g., bfloat16 vs float32)
  - ....



# More Recently ...

- Training Loss for Llama 2 models.
- After pretraining on 2T Tokens, the models still did not show any sign of saturation. Why? 🤔
- The scaling laws are usually derived on much smaller scales. Behavior might be different at larger scales.



# Data Quality Matters

- There is increasing evidence that data more than just token count.
  - Previously we saw that data duplications and noise hurts LLM performance.
  - There is also evidence that's one can be selective about data diversity.
- These are topics of the ongoing research and there will be more discuss here in coming years.

Beyond neural scaling laws: beating power law scaling via data pruning (Sorscher et al., 2022)

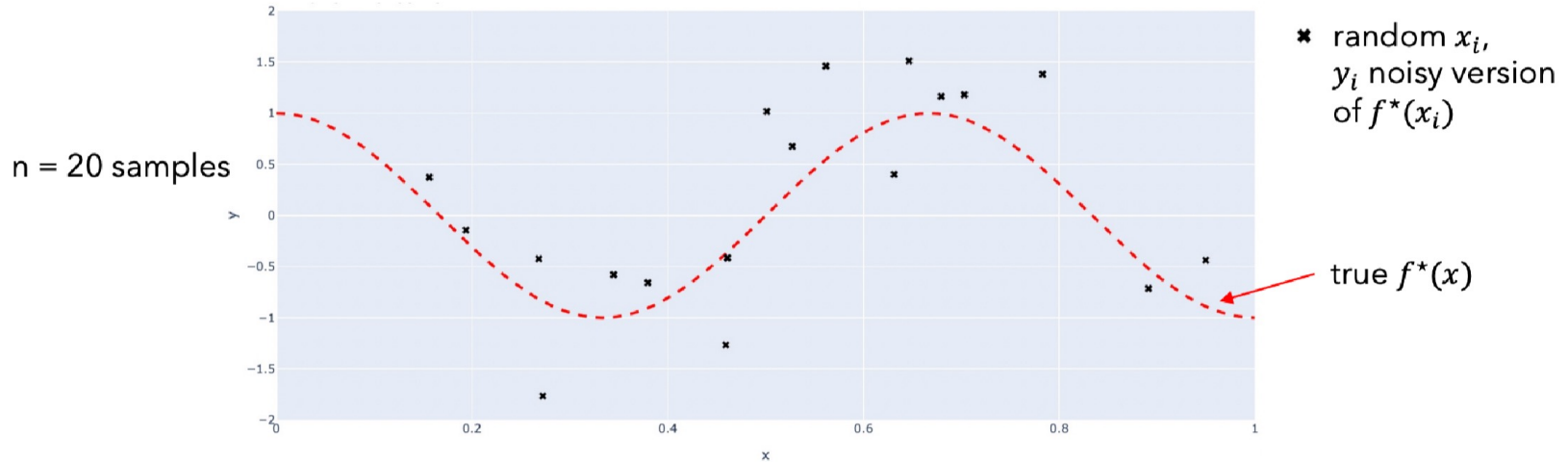
# Summary

---

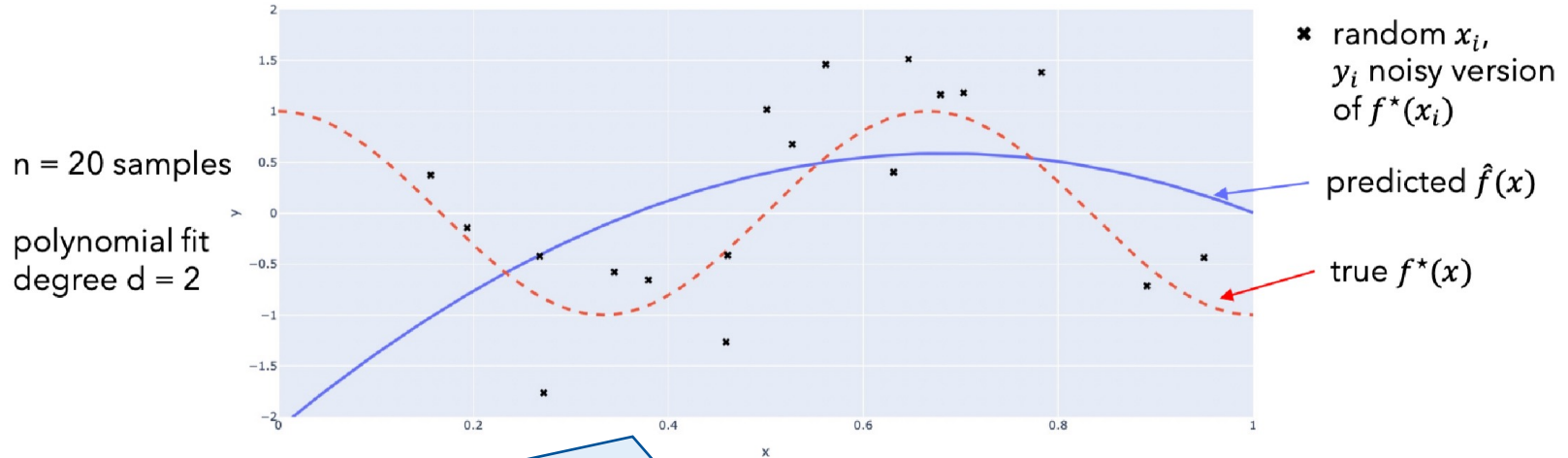
- Optimality conditions: given a limited budget (compute, data, size) what is the best model you can train.
- For now: maintain similar ratio for model size and data size.
- Next: why didn't we scale earlier?

# Why didn't we scale earlier??

# The Old Wisdom: Optimizing Model Capacity



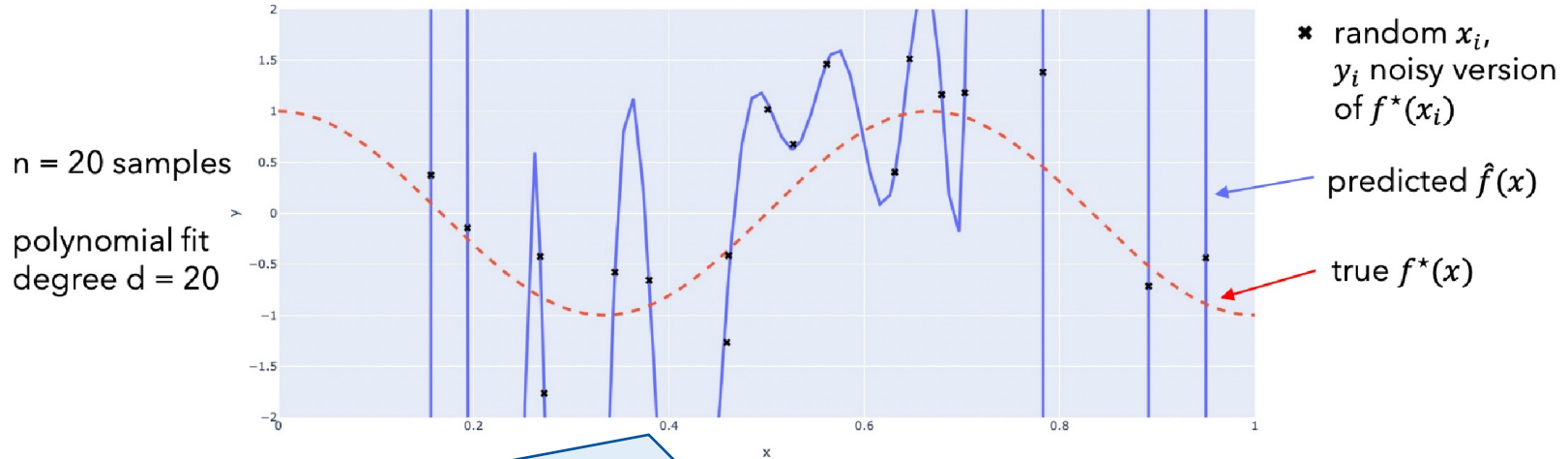
# The Old Wisdom: Optimizing Model Capacity



Small models cannot fit perfectly.

- they cannot express complex functions  $\rightarrow$  high statistical bias.
- largely ignores noise  $\rightarrow$  does not fluctuate a lot (small variance)

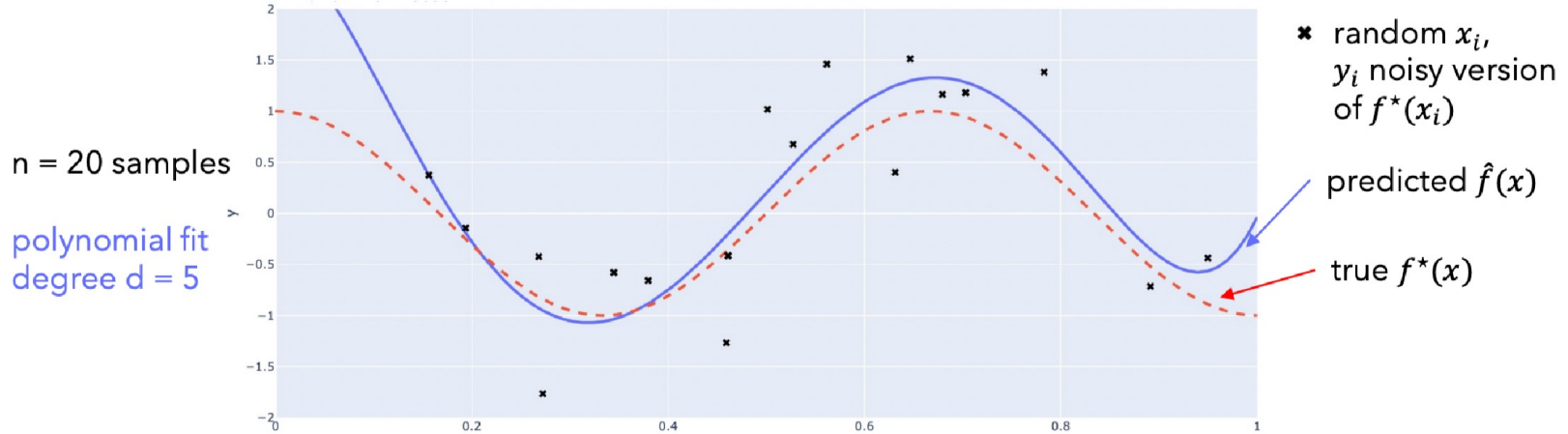
# The Old Wisdom: Optimizing Model Capacity



Large models fit perfectly (overfit)

- Can express function of interest  $\rightarrow$  small statistical bias.
- Fits too much of the noise (overfit)  $\rightarrow$  fluctuates a lot (high variance)

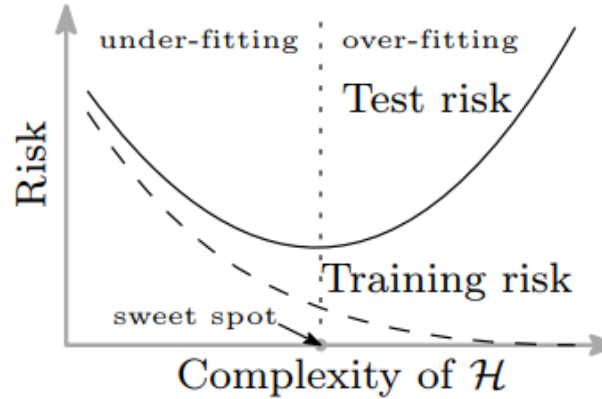
# The Old Wisdom: Optimizing Model Capacity



**Classical generalization theory** — one can get generalization by optimizing for capacity (expressivity) — equivalent to balancing the bias-variance trade-off.



# The Old Wisdom: Optimizing Model Capacity

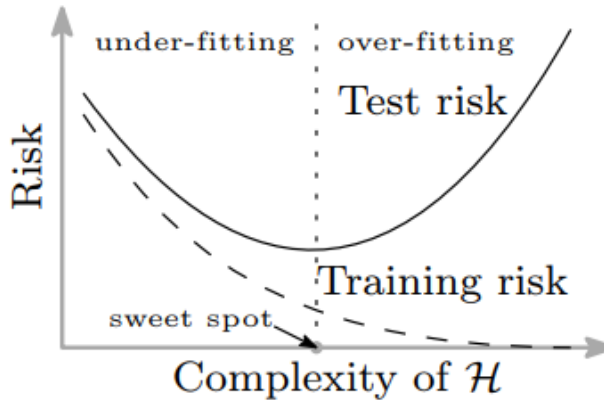


(a) U-shaped “bias-variance” risk curve

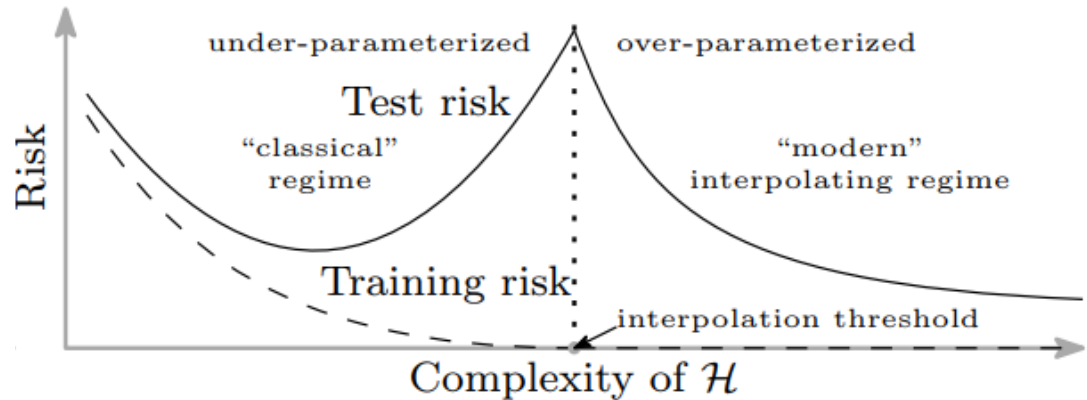
**Classical generalization theory** — one can get generalization by optimizing for capacity (expressivity) — equivalent to balancing the bias-variance trade-off.

# Harmless Interpolation for Large Models

- Learning theory made us allergic to over-parametrized models.

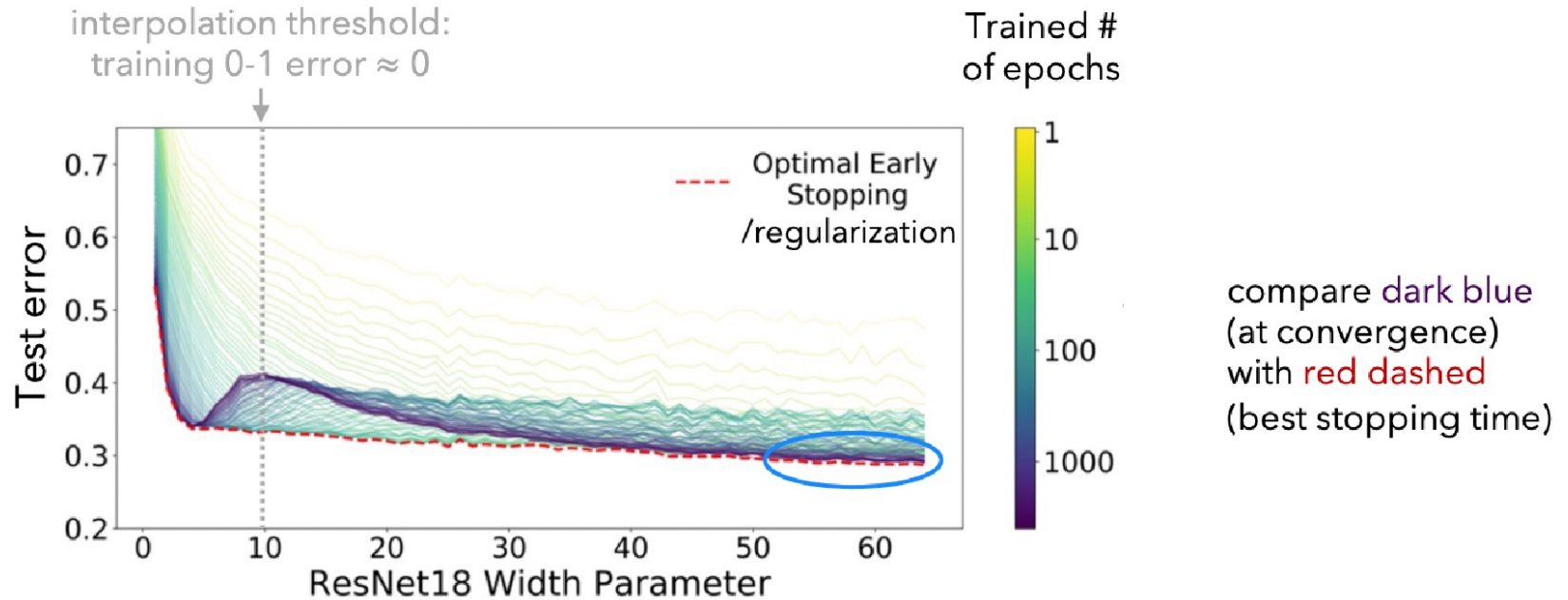


(a) U-shaped "bias-variance" risk curve



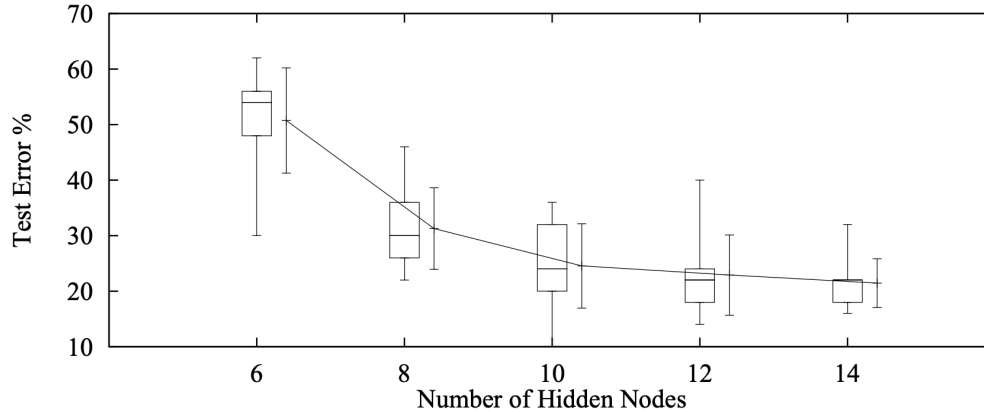
(b) "double descent" risk curve

# Harmless Interpolation for Large Models



# There Were Empirical Evidence

- Even in mid-90's there were evidence supporting the benefit of larger models
- Although they were ignored ....



“ .... larger networks may generalize well and better generalization is possible from larger networks if they can be trained more successfully than the smaller networks” -- Lawrence, Giles, and Tsoi in 1997

# There Were Empirical Evidence

- Even in mid-90's there were evidence supporting the benefit of larger models
- Although they were ignored ....

“ .... "Nets of all sizes overfit some problems. But generalization is surprisingly insensitive to excess capacity if the net is trained with backprop.”

-- Caruana, Lawrence, and Giles (2000)

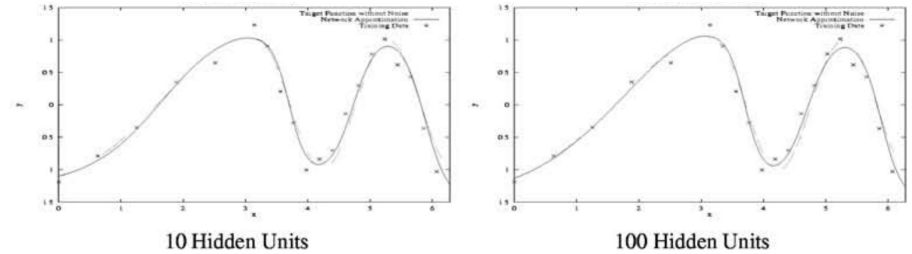


Figure 3: MLP approximation using backpropagation (BP) training of data from Equation 1 as the number of hidden units is increased. No significant overfitting can be seen.

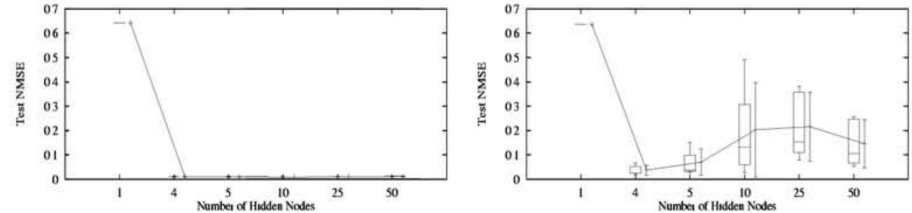


Figure 4: Test Normalized Mean Squared Error for MLPs trained with BP (left) and CG (right). Results are shown with both box-whiskers plots and the mean plus and minus one standard deviation.

- Overfitting not bad: double descent phenomenon

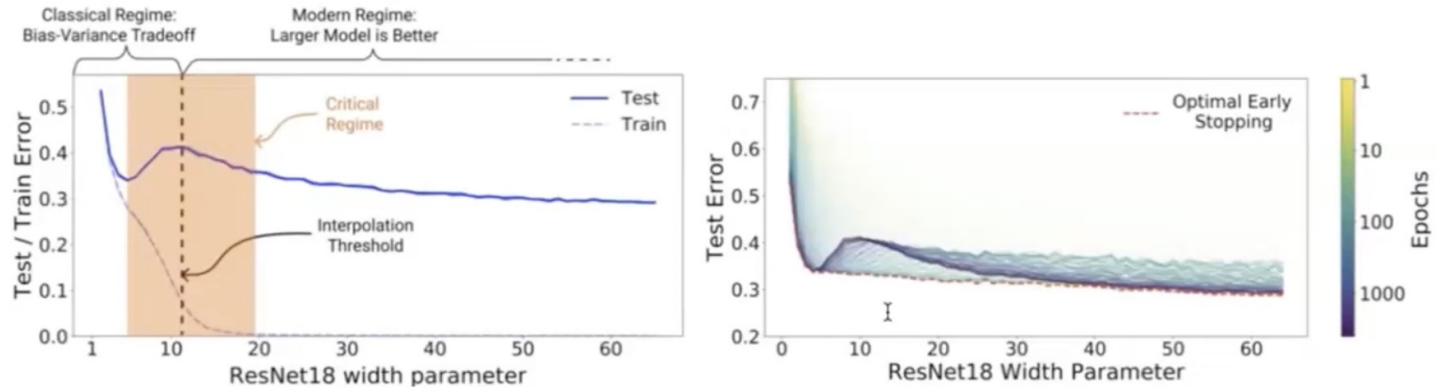


Figure 1: **Left:** Train and test error as a function of model size, for ResNet18s of varying width on CIFAR-10 with 15% label noise. **Right:** Test error, shown for varying train epochs. All models trained using Adam for 4K epochs. The largest model (width 64) corresponds to standard ResNet18.

Nakkiran, et al. Deep double descent: Where bigger models and more data hurt. *JSTAT*, 2021.  
 Nakkiran, P., Venkat, P., Kakade, S., & Ma, T. (2020). Optimal Regularization Can Mitigate Double

# Summary

---

- We some had evidence for impact of scaling.
- Took us some time to trust them.

# Is Scale All You Need?



# Is Scale All We Need?

---

- For what purpose?
  - For building useful applications (answering simple questions, translating simple sentences) we already have good models. Not our focus.
  - General intelligence: think of an assistant that is always with you, knows what you want, assists you with anything you need.

# Is Scale All We Need?

- For what purpose?
  - For building useful applications (answering simple questions, translating simple sentences) we already have good models. Not our focus.
  - General intelligence: think of an assistant that is always with you, knows what you want, assists you with anything you need.



Nando de Freitas 🇵🇷  
@NandoDF



Solving these scaling challenges is what will deliver AGI. Research focused on these problems, eg S4 for greater memory, is needed. Philosophy about symbols isn't. Symbols are tools in the world and big nets have no issue creating them and manipulating them 2/n

4:50 AM · May 14, 2022 · Twitter for iPhone

23 Retweets 5 Quote Tweets 153 Likes

Do you agree with Nando?

# Is Scale All We Need?

---

1. Is scale the/a right “hill to climb”?
2. Even if it is a right “hill” is it feasible/practical to climb this hill? (there might be other “hills” too).

# Argument: Not Enough Compute

Limitations regarding compute:

- There is simply not enough compute available.
  - Models have been increasing 10x every year
  - Moore's law: # of transistors on an IC doubles about every two years.
  - There are physical limits to how much faster computers can get.
- Even if we have the compute, scaling the compute will be quite costly.
- Scaling compute is simply infeasible. [QED]

Are you convinced?

# Rebutting “Not Enough Compute”

- On insufficiency of compute resource:
  - Hardware technologies continue to progress at a rapid pace.
  - Huang’s law: advancements in GPUs happen at much faster rate than what Moore predicted.
  - So much potentials in parallel computing.
- On cost-[in]efficiency of scaling:
  - While models like GPT3 cost a lot (monetary or otherwise), their availability prevent training MANY smaller, mediocre models.
  - Therefore, it might be that the net cost of scaling large models is negative.
    - It is the case within Microsoft according to its CTO, Kevin Scott.

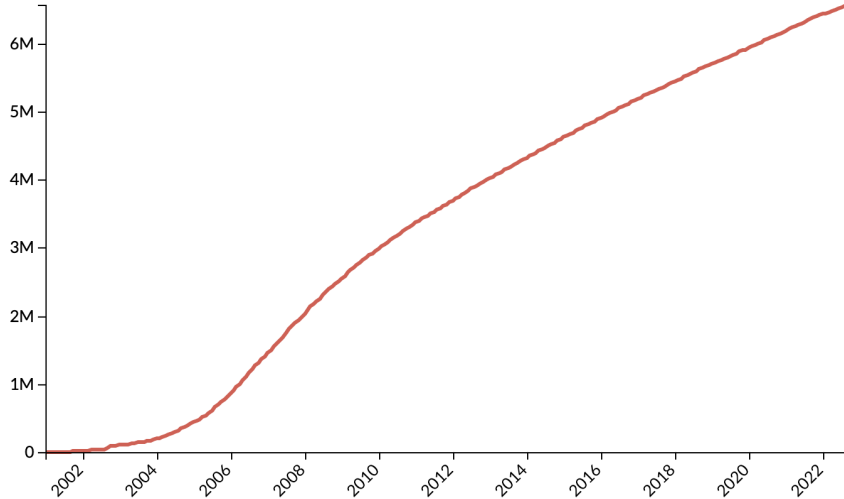
# Argument: Not Enough Data

- Hoffmann et al showed that, to be compute-optimal, **model size and training data must be scaled equally**.
- It shows that existing LLMs are severely data-starved and under-trained.
- Given the new scaling law, even if you pump a billions of params into a model, the gains will **not** compensate for more training tokens.
- There is simply not enough [language] data. [QED]

Are you convinced?

# Rebutting “Not Enough Data”

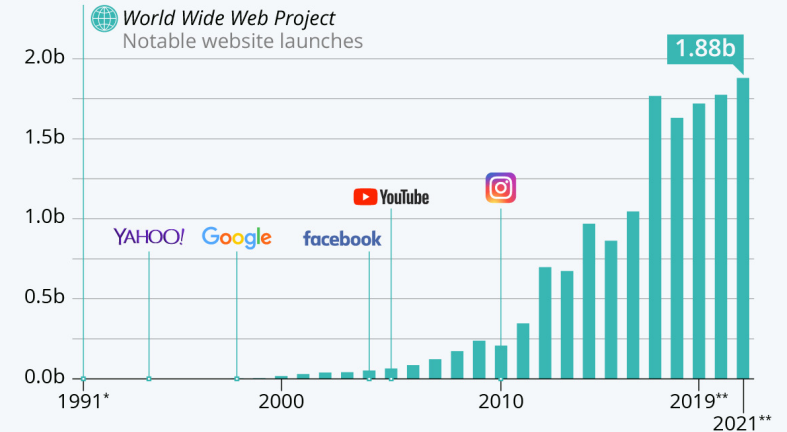
- Data is growing exponentially (?)



Wikipedia size

## How Many Websites Are There?

Number of websites online from 1991 to 2021



\* As of August 1, 1991.

\*\* Latest available data for 2019: October 28, for 2020: June 2, for 2021: August 6.

Source: Internet Live Stats

# Rebutting “Not Enough Data” (2)

- You can harness data from other modalities.
  - For example, to get more text data we can build a solid speech processor model that converts speech to text.
  - (aside: more than 80% of internet traffic is video)

## **SKYQUEST**

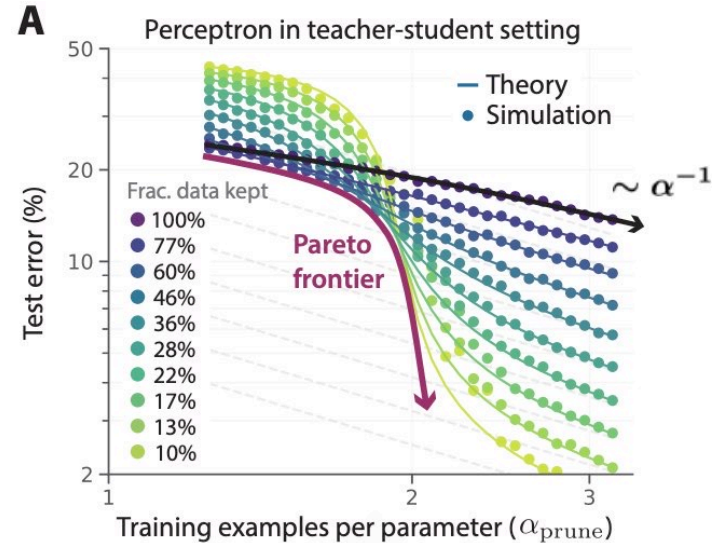
Global Online Video Platforms Market Drives over 80% of Total Internet Traffic | Skyquest Technology

- (aside2: is that why OpenAI built Whisper?!)



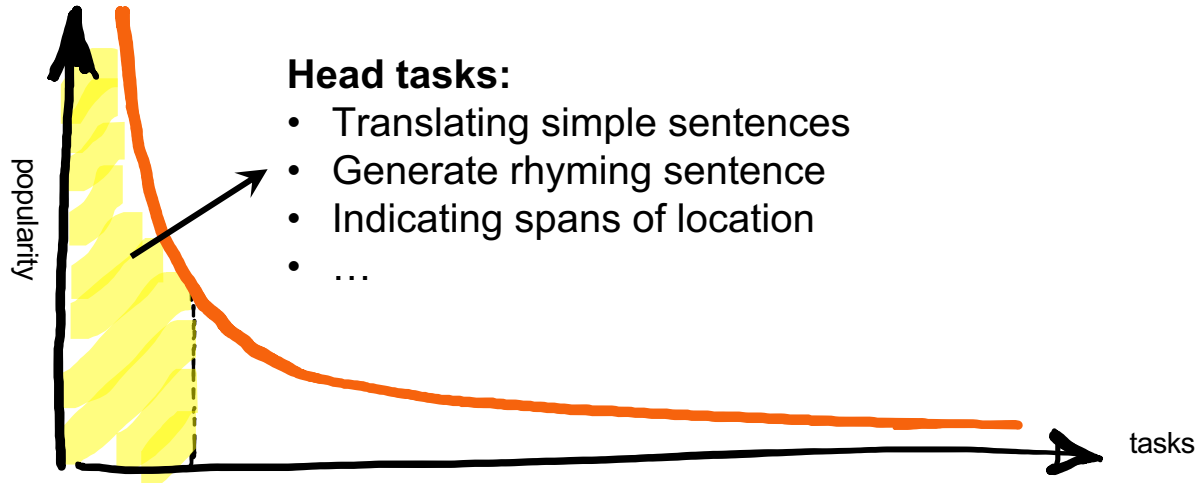
# Rebutting “Not Enough Data” (3)

- You can use data more effectively.
- Sorscher et al. lays out recipes to achieve \*exponential\* scaling instead through statistical mechanics theory.
- Carefully curating a small subset goes a long way!



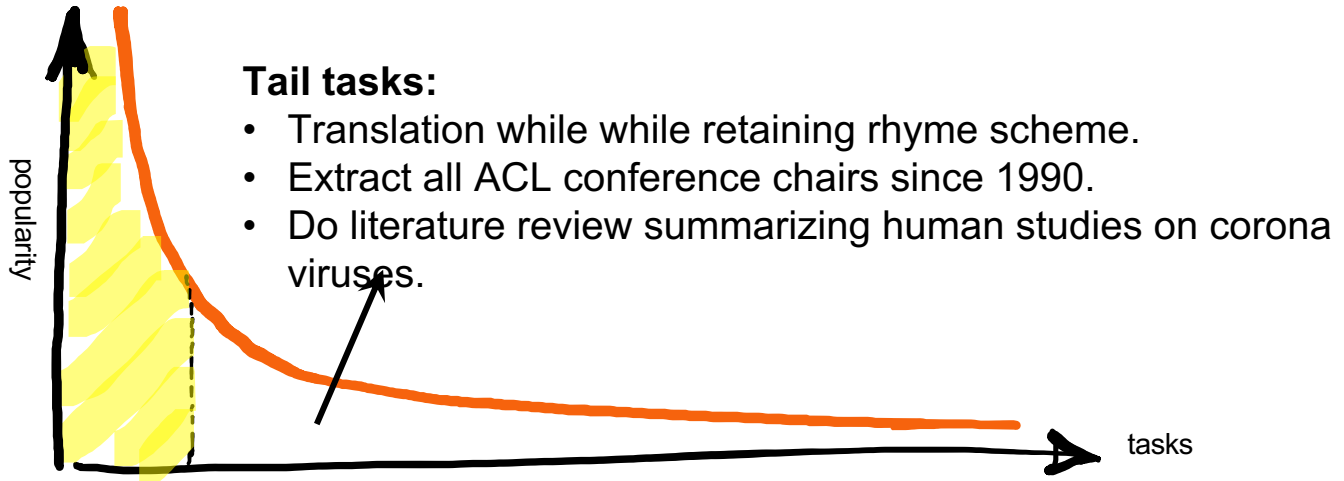
# Argument: Scale is Not all You Need Because of Tail Phenomena

- Tail phenomena will never go away!



# Argument: Scale is Not all You Need Because of Tail Phenomena

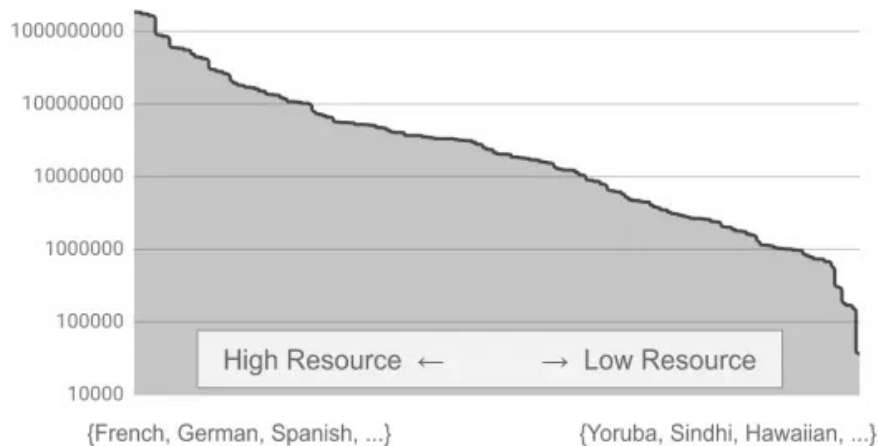
- Tail phenomena will never go away!



# Argument: Scale is Not all You Need Because of Tail Phenomena

- Tail phenomena will never go away!

Data distribution over language pairs



## Example: Google Translate

*“The number of parallel sentences [...] ranges from around tens of thousands to almost 2 billion.”*

# Argument: Scale is Not all You Need Because of Tail Phenomena

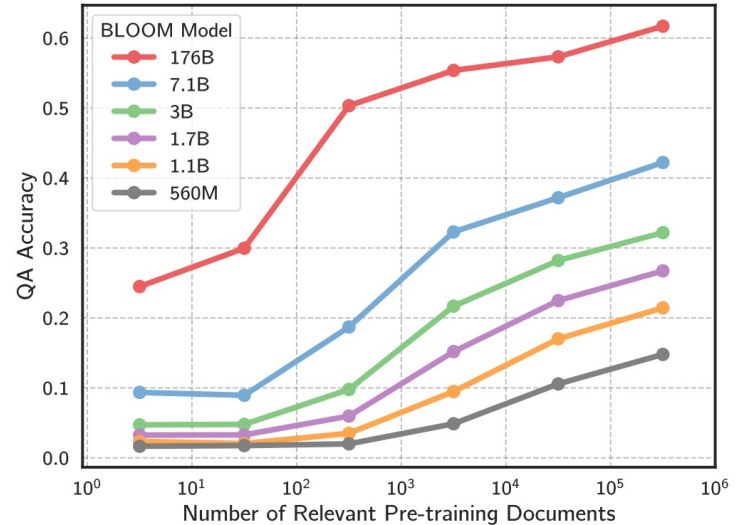


Figure 1: Language models struggle to capture the long-tail of information on the web. Above, we plot accuracy for the BLOOM model family on TriviaQA as a function of how many documents in the model’s pre-training data are relevant to each question.

# Argument: Scale is Not all You Need Because of Tail Phenomena

- Tail phenomena will never go away!
- Will result in brittleness to small changes

“Tesla's Autopilot system confusing horse-drawn carriage for truck”



Y

which number is bigger, 999934 or 32323232332?



The number 32323232332 is larger than 999934. To compare the sizes of two numbers, we need to look at the number of digits in each number. In this case, the number 32323232332 has 10 digits, while the number 999934 has 6 digits. Because the number with more digits is always larger, we can conclude that 32323232332 is the larger number in this case.



Y

which number is bigger, -999934 or -32323232332?



The number -32323232332 is larger than -999934. When comparing negative numbers, we follow the same rules as when comparing positive numbers. In this case, the number -32323232332 has 10 digits, while the number -999934 has 6 digits. Because the number with more digits is always larger, we can conclude that -32323232332 is the larger number in this case.

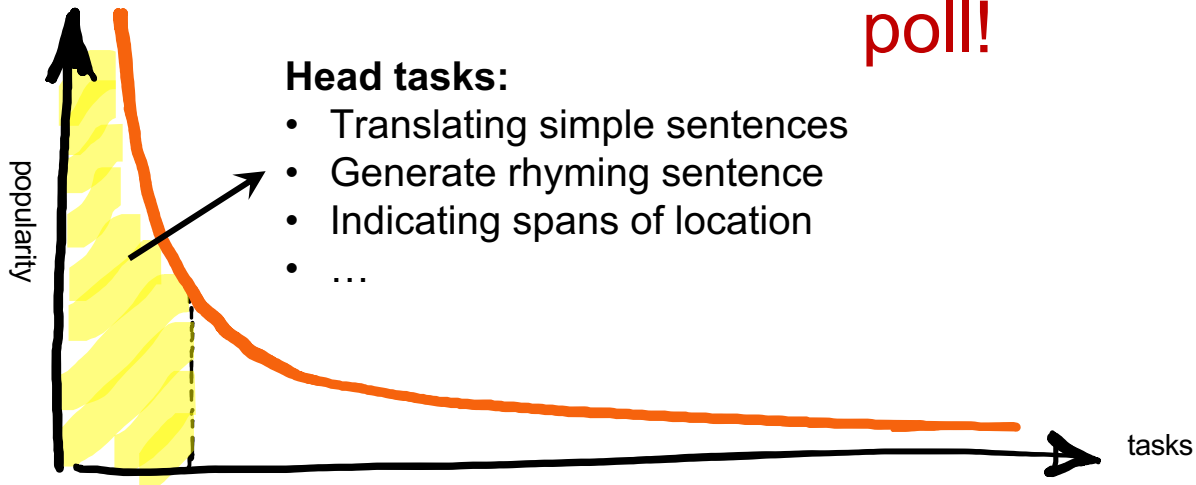


It is worth noting that negative numbers are always less than positive numbers, so even though -32323232332 is larger than -999934, it is still smaller than 999934. This is because a negative number represents a value that is less than zero, while a positive number represents a value that is greater than zero.

# Argument: Scale is Not all You Need Because of Tail Phenomena

- Hence, scale won't solve the tail phenomena. [QED]

Let's do a poll!





# Argument: Scale is Not all You Need Because of Tail Phenomena

---

- How do you rebut this??

# Summary

---

- “Scaling” is here is here to stay for now.
  - There is plenty of data out there that we haven’t been able to use.
  - There is no sign of development in hardware technology.
- The long tail poses a serious challenge:
  - It’s possible that scaling will continue to yield gains in the long-tail, but unlike to solve it.
  - It is unclear if there are better ways of solving long tail phenomena.
- Unclear “scale” can say about other aspect of intelligence
  - coordination, cooperation, communication, etc.