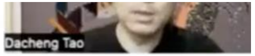
📈 Scaling Language Models

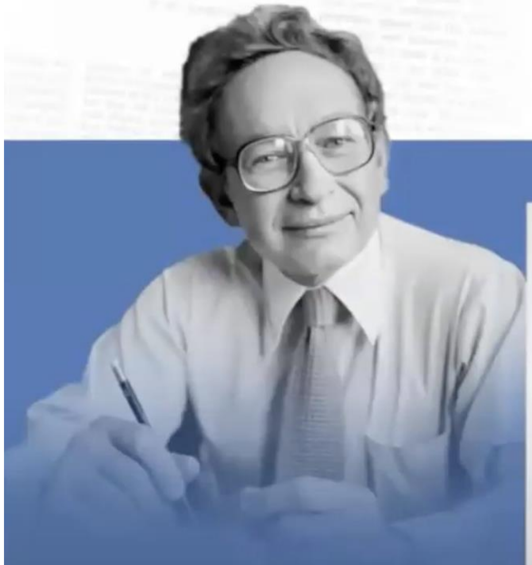
CSCI 601-471/671 (NLP: Self-Supervised Models)

Johns Hopkins
Whiting School of Engineering

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

More is different






SCIENCE

More Is Different

We expect to encounter fascinating and, I believe, very fundamental questions at each stage in fitting together less complicated pieces into the more complicated system and understanding the basically new types of behavior which can result.

—— "More Is Different , SCIENCE"

4 August 1972

Philip W. Anderson

Mov

What is scaling?

α x Model size x Training data = Training compute

	Model size (# parameters)	Training data (# tokens)	Training compute (FLOPs)	Resources
BERT-base (2018)	109M	250B	1.6e20	64 TPU v2 for 4 days (16 V100 GPU for 33 hrs)
GPT-3 (2020)	175B	300B	3.1e23	~1,000x BERT-base
PaLM (2022)	540B	780B	2.5e24	6k TPU v4 for 2 months

SIMONS INSTITUTE

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)
Language Models are Few-Shot Learners (2020)
PaLM: Scaling Language Modeling with Pathways (2022)

https://www.youtube.com/watch?v=Kp5R6GZh8O0&ab_channel=SimonsInstitute

- Nice slides here on efficiency NVlabs/EfficientDL (github.com)
- Tim's slides: https://docs.google.com/presentation/d/1B4md7w09daAs6BRoiDROgBQxO3Osmo33Mvi2ITFXDt0/edit?usp=sharing
- Also, Tianjian's slides:
  - https://docs.google.com/presentation/d/1gRZjh1VIFWx_sh6cleDDpICHXF8nsqjkZD4GESVtxmA/edit?usp=sharing
  - https://docs.google.com/presentation/d/1toRTm2oRUyUB3ozv1c2oHz4hF44JkBWtgzUKyj6gSow/edit?usp=sharing
- CVPR 2023 Tutorial (google.com)

- Good content on scaling data:
- https://huyenchip.com/2023/05/02/rlhf.html

Scaling model size

- LM are getting larger and more expensive

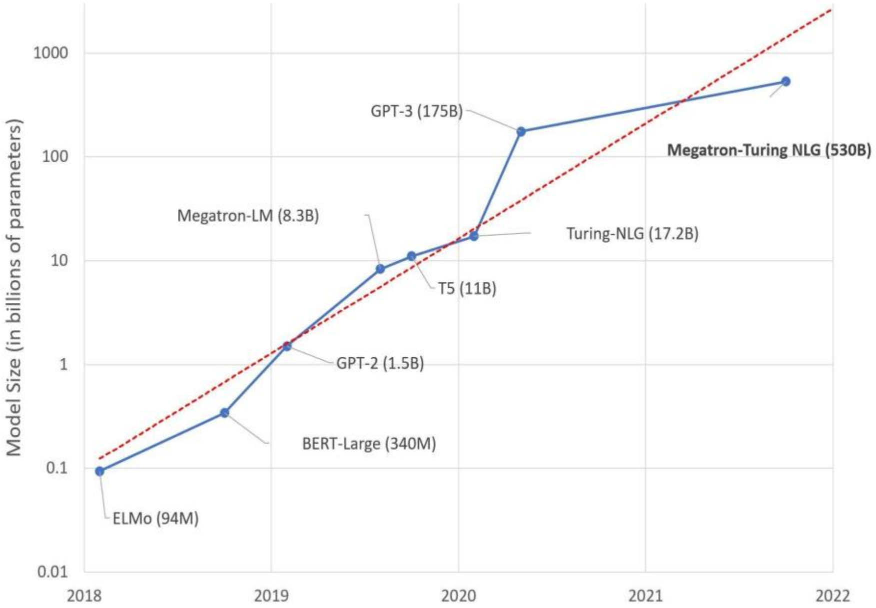
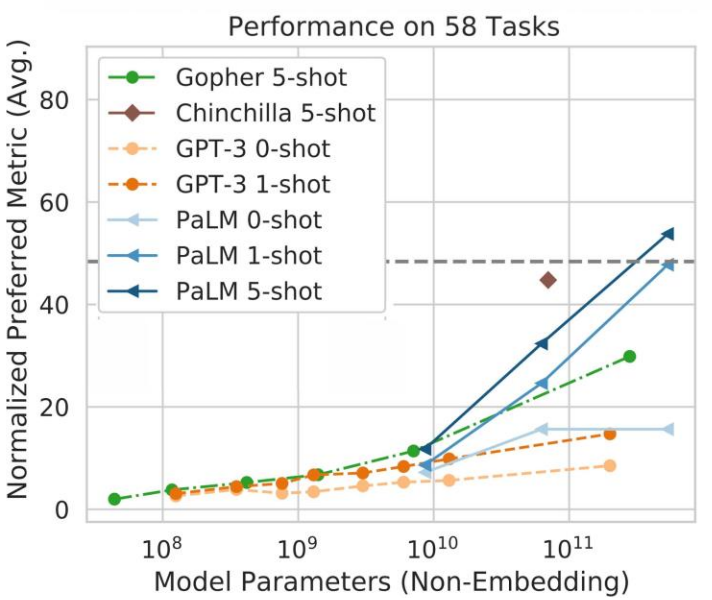
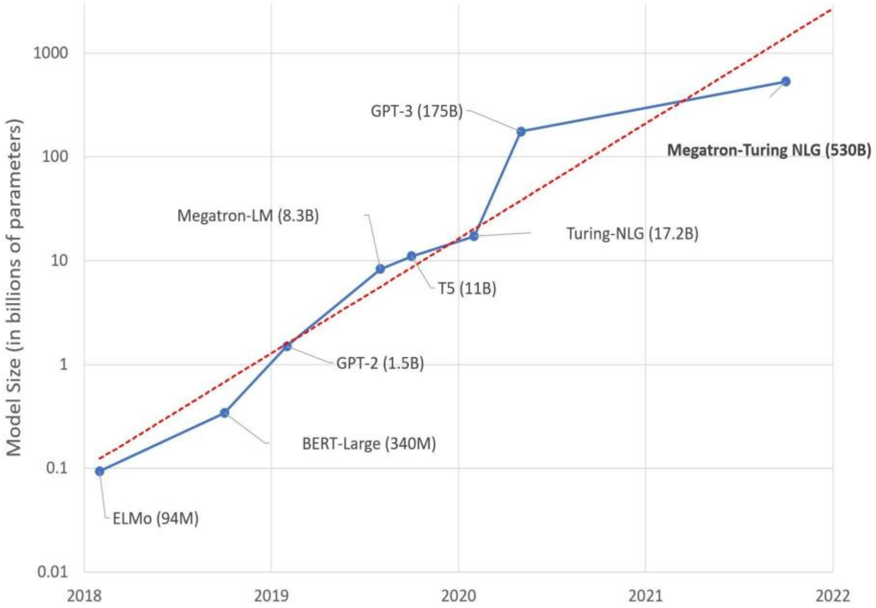


Photo credit: Microsoft Research Blog, Alvi et. al., 2021

Model Size vs. Accuracy

Photo credit: PaLM, Chowdhery et. al., 2022



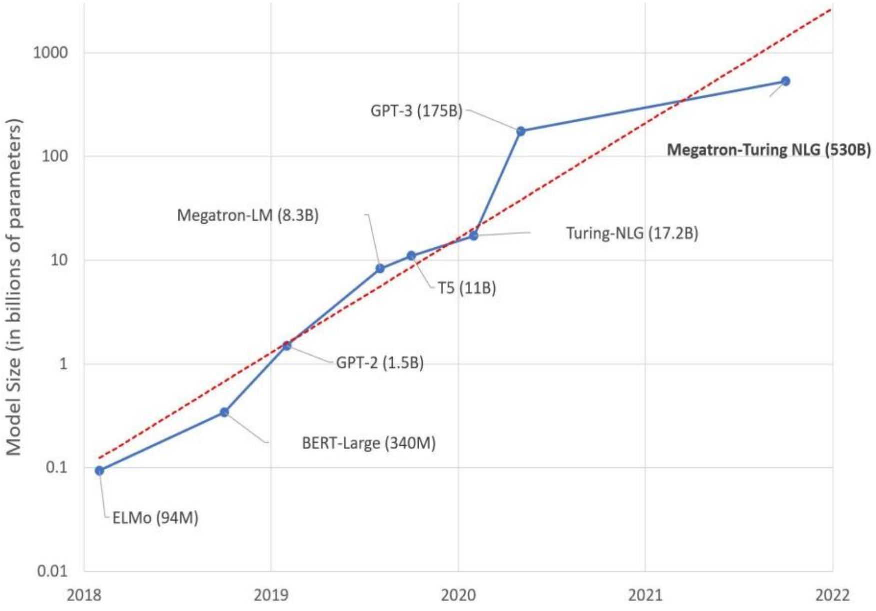


Larger LMs ⇒ better zero/few-shot performance

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. Impact of scaling [TODO: this may need to appear earlier]
4. Scaling and theory
5. Is scale all you need? A discussion.

Chapter goal: TODO

Computation Cost of Models

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

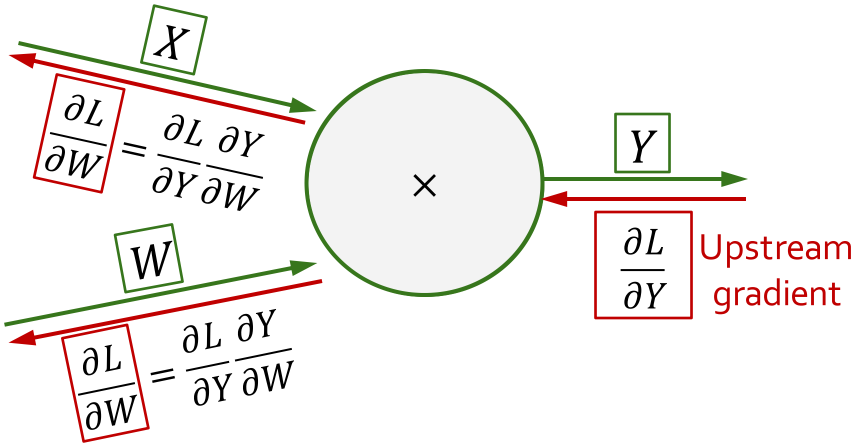
- Now this is just forward propagation in Backprop. What about the **backward** step?

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.

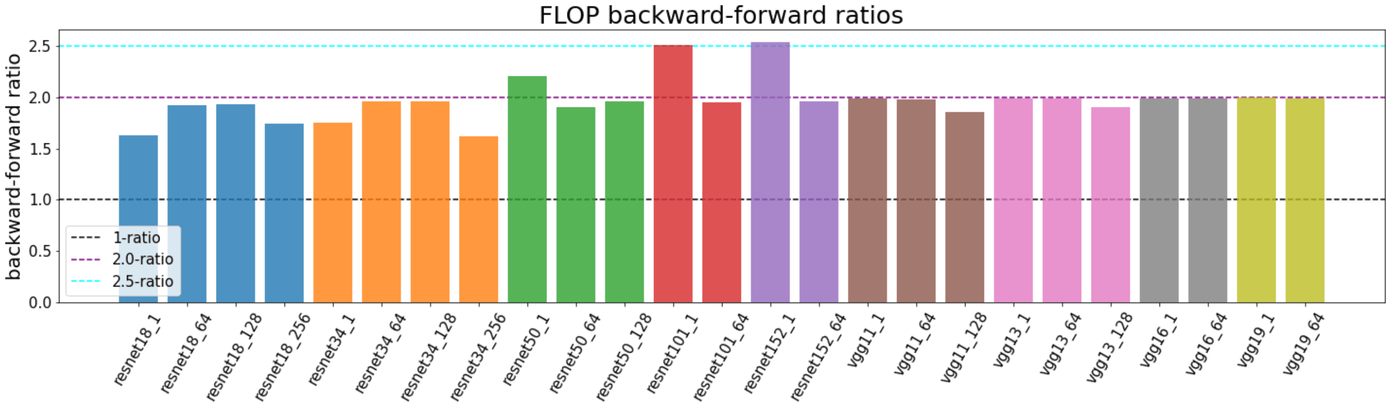
One matrix multiplication for $\frac{\partial L}{\partial W}$



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 x (batch size) x (size of W)

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

- One can show that:

$$\boxed{N = \overbrace{12 \times n_{layers} \times d_{model}^2}^{\text{Non-embedding params}} + \underbrace{(n_{vocab} + n_{pos}) \times d_{model}}_{\text{Embedding params}}}$$

- Assuming:
  - the size of MLP hidden layer to be $4.\, d_{\text{model}}$
  - $n_{\text{heads}} . \, d_{\text{hidden}} = d_{\text{model}}$

- You will prove this in homework assignment! 😊

Most Transformer LMs make these design assumptions.

Transformer Parameter Count

- One can show that:

$$N = \overbrace{12 \times n_{layers} \times d_{model}^2}^{\text{Non-embedding params}} + \underbrace{(n_{vocab} + n_{pos}) \times d_{model}}_{\text{Embedding params}}$$

n_{layer}	d_{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

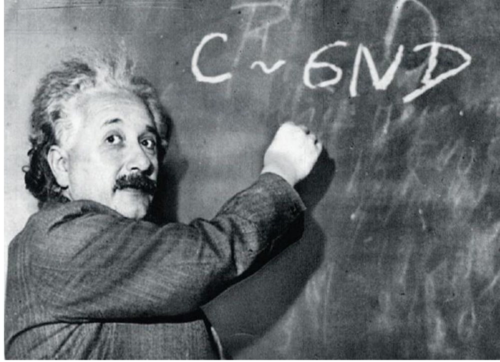
For example, see the models in the following table:

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

Vocab size = 65536
Positional emb size = ?

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

Transformer Parameter Count



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

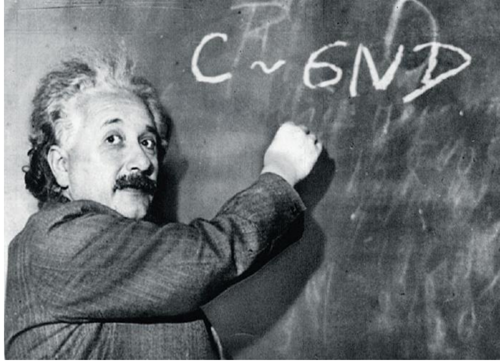
n_{layer}	d_{model}	Parameters (N)	Training FLOPs
4	512	13M	3.0e19
6	768	42M	1.0e20
10	1280	197M	4.7e20
16	2048	810M	1.9e21
24	3072	2.7B	6.5e21
40	5120	13B	3.0e22
64	8192	52B	1.2e23

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

Training cost (FLOPs):

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

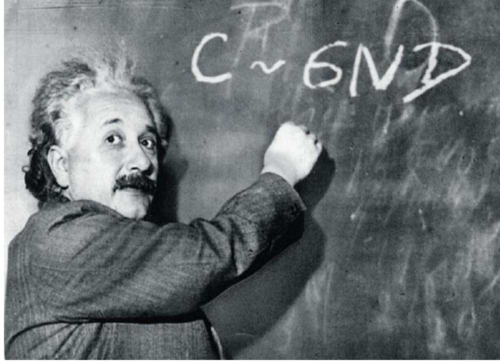
Estimating training time



- 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
https://arxiv.org/pdf/2109.04650.pdf

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):

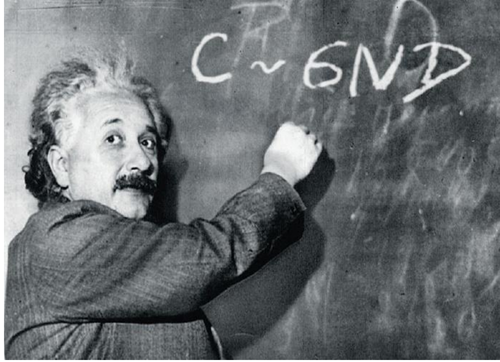
$$C \approx 6ND$$
$$= 6 \times (150 \times 10^9) \times (82 \times 10^9) = 7.3 \times 10^2$$

- The peak throughput of [A100 GPUs] if 312 teraFLOPS or 3.12×10^{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
https://arxiv.org/pdf/2109.04650.pdf

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
https://arxiv.org/pdf/2109.04650.pdf

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

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

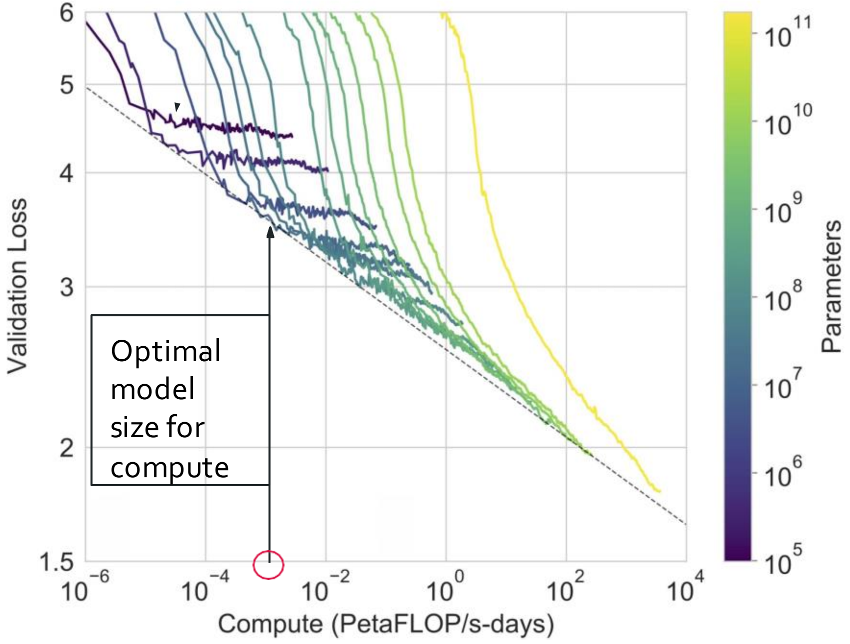


Photo credit: GPT3, Brown et. al., 2020

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.

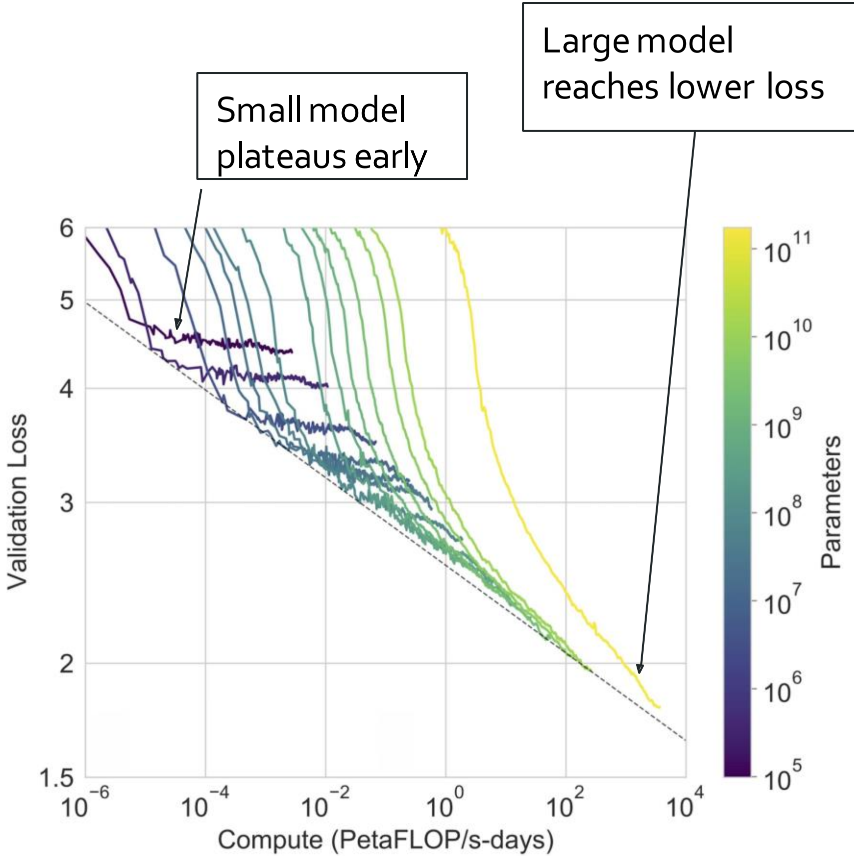


Photo credit: GPT3, Brown et. al., 2020

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

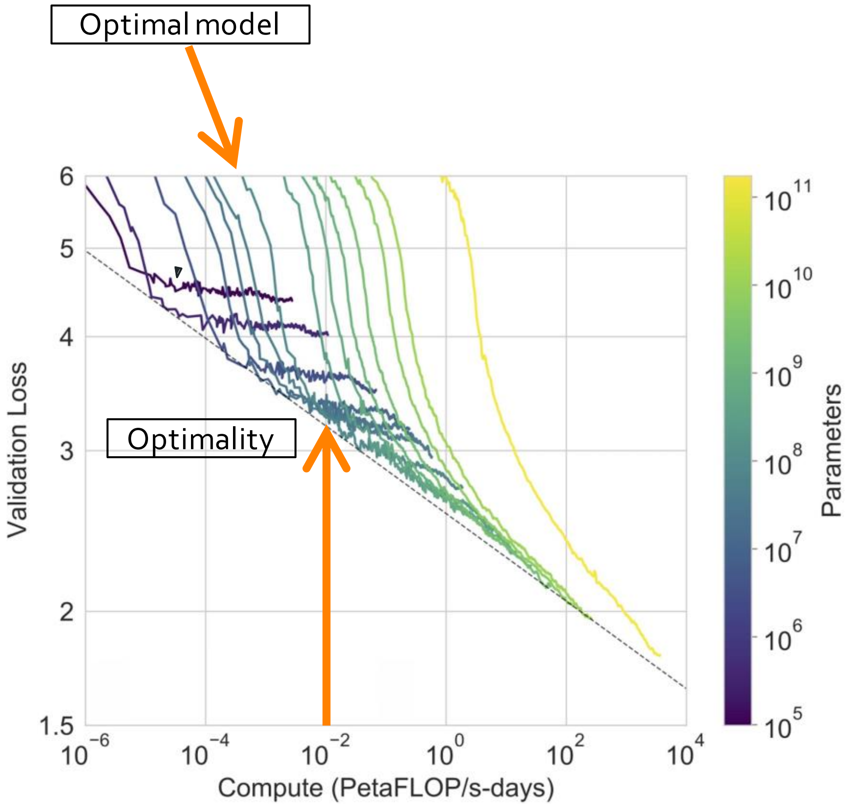


Photo credit: GPT3, Brown et. al., 2020

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.

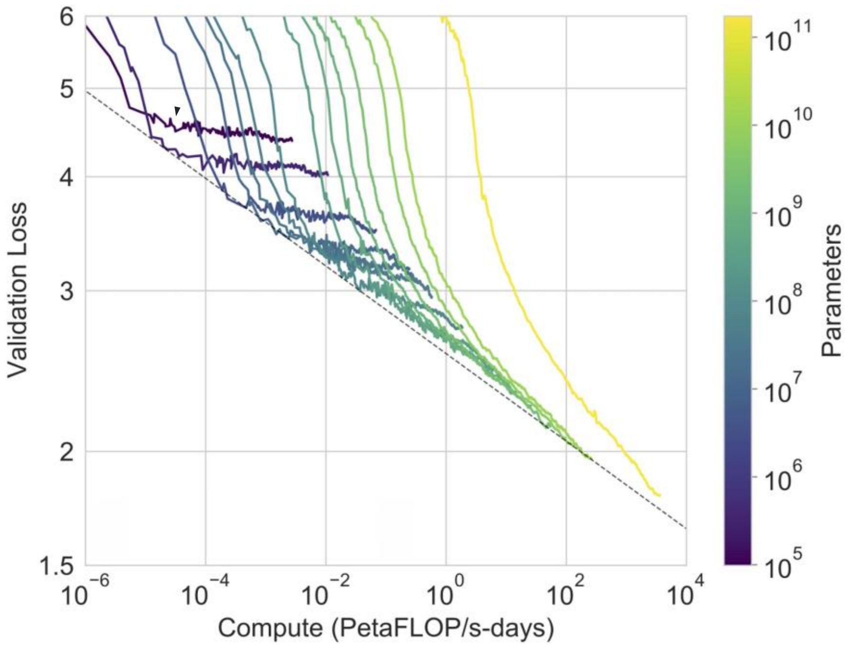


Photo credit: GPT3, Brown et. al., 2020

Scaling - Optimal Model Size

- What is the function that describes this optimality line?

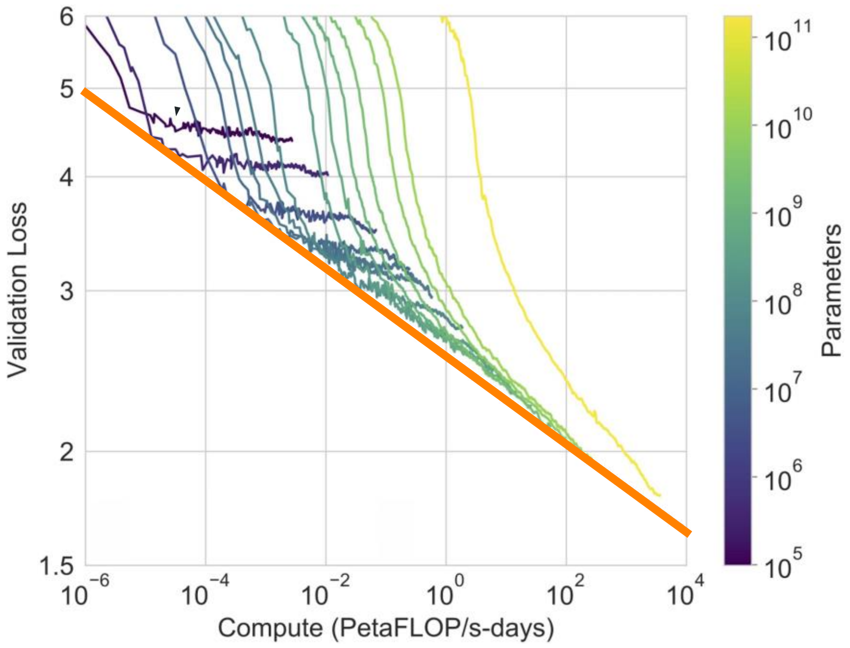


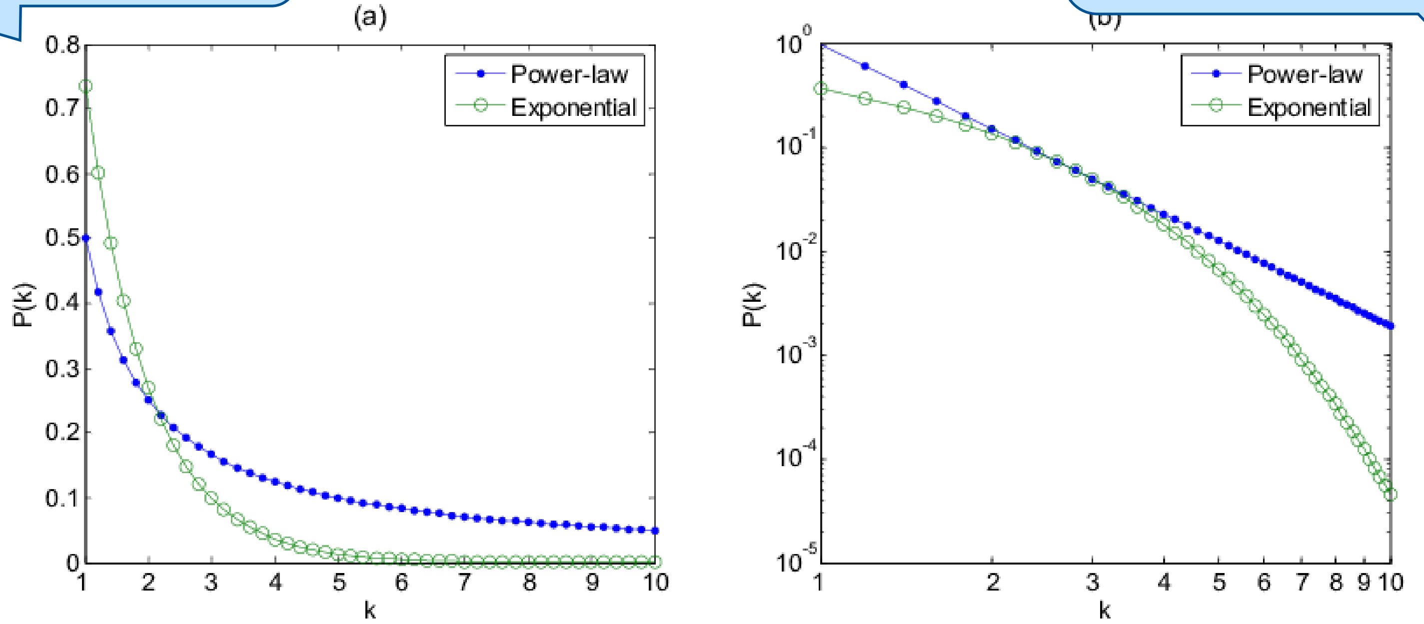
Photo credit: GPT3, Brown et. al., 2020

Terminology: Power law vs exponential

Exponential goes to zero at a faster rate.

$$\text{Power law} = \text{variable}^{(\text{constant})}$$
$$\text{Exponential} = (\text{constant})^{\text{variable}}$$

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.

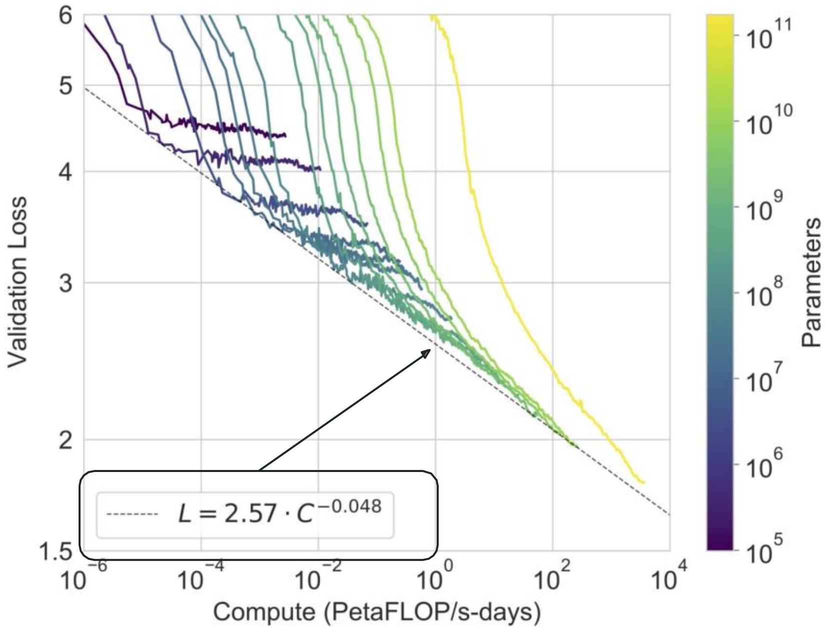


Photo credit: GPT3, Brown et. al., 2020

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

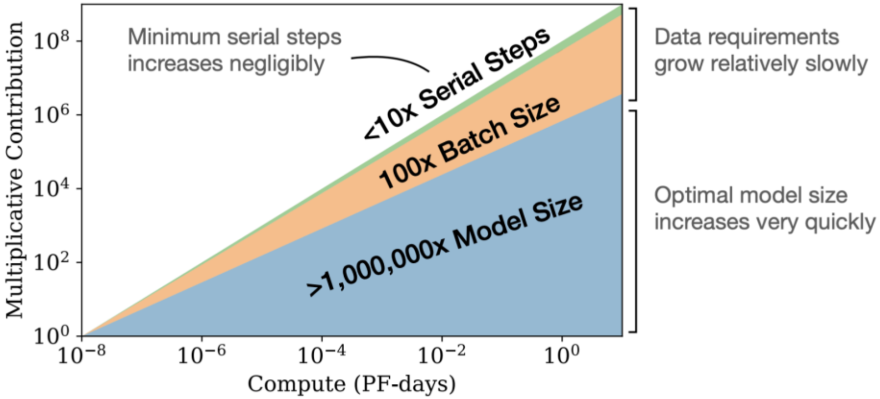
- **Takeaway:** grow the model size faster than growing the number of tokens.
  - **Example:** Given 10x compute, increase N by [MASK] and D by [MASK]
- GPT3 (and many other followed this recipe) training a 175B model on 300B tokens

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

[ACL 2022 Tutorial Beltagy, Cohan, Logan IV, Min and Singh]

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.)

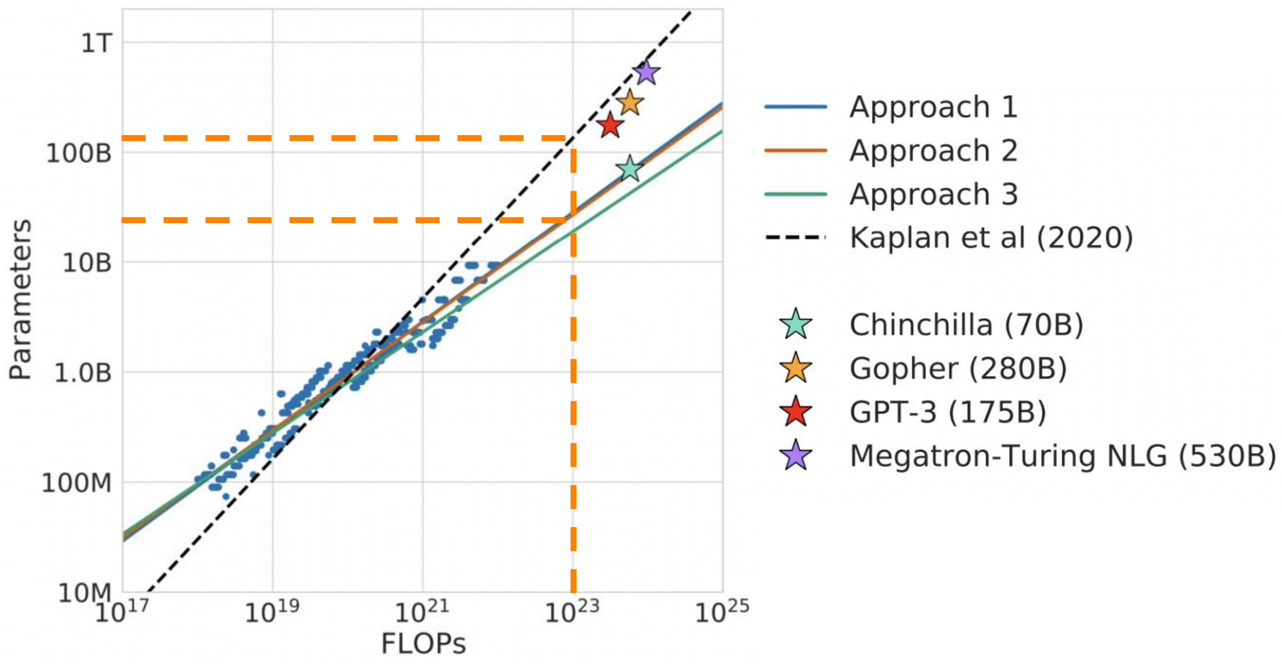
- Subsequent papers tried to engineer larger and larger models

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
Gopher (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 Laws: Hoffmann et al.

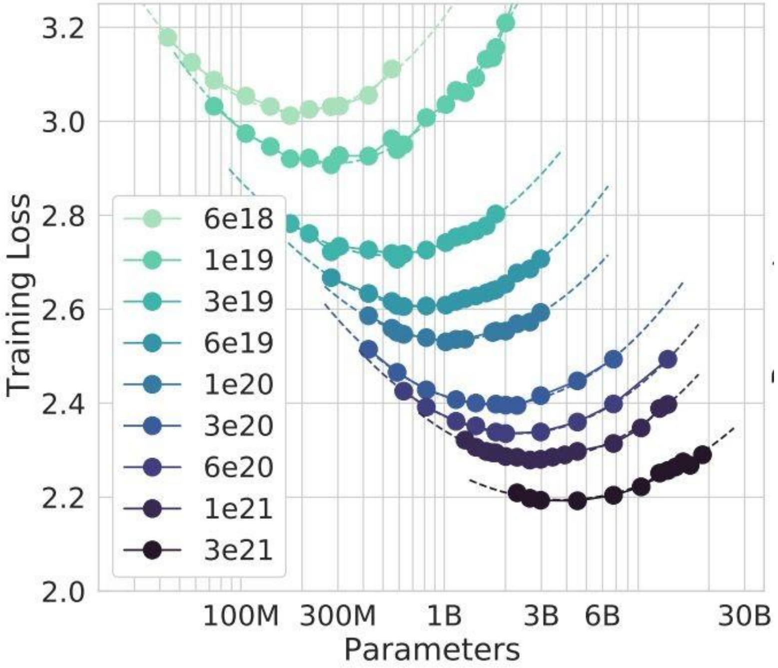


Training Compute-Optimal Large Language Models (2022)
https://arxiv.org/pdf/2203.15556.pdf

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



Training Compute-Optimal Large Language Models (2022)
https://arxiv.org/pdf/2203.15556.pdf

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 [MASK] and D by [MASK]
  - **Example 2:** Given 100x compute, grow N by [MASK] and D by [MASK]

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

[ACL 2022 Tutorial Beltagy, Cohan, Logan IV, Min and Singh]

Recap

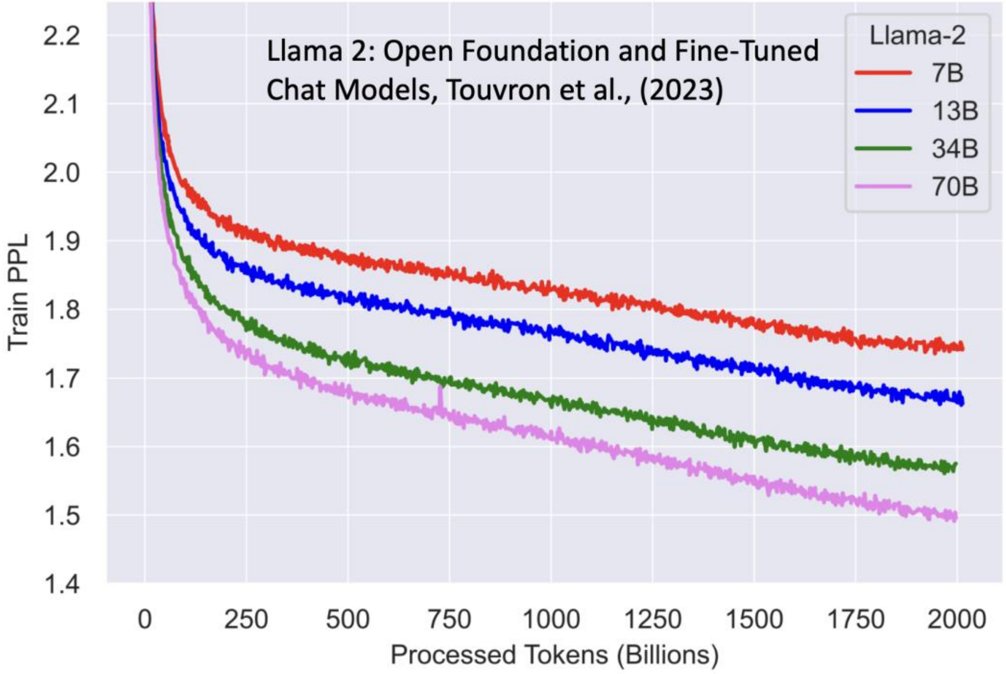
- We used to train “oversized” and “under-trained” models.
- You should scale your model at the same rate as your data.
- For example, if you get a 100x increase in compute,
  - you should make your model 10x bigger and your data 10x bigger.

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)

Data-Constrained Scaling Laws

- Here I can discuss Sasha Rush’s recent work.

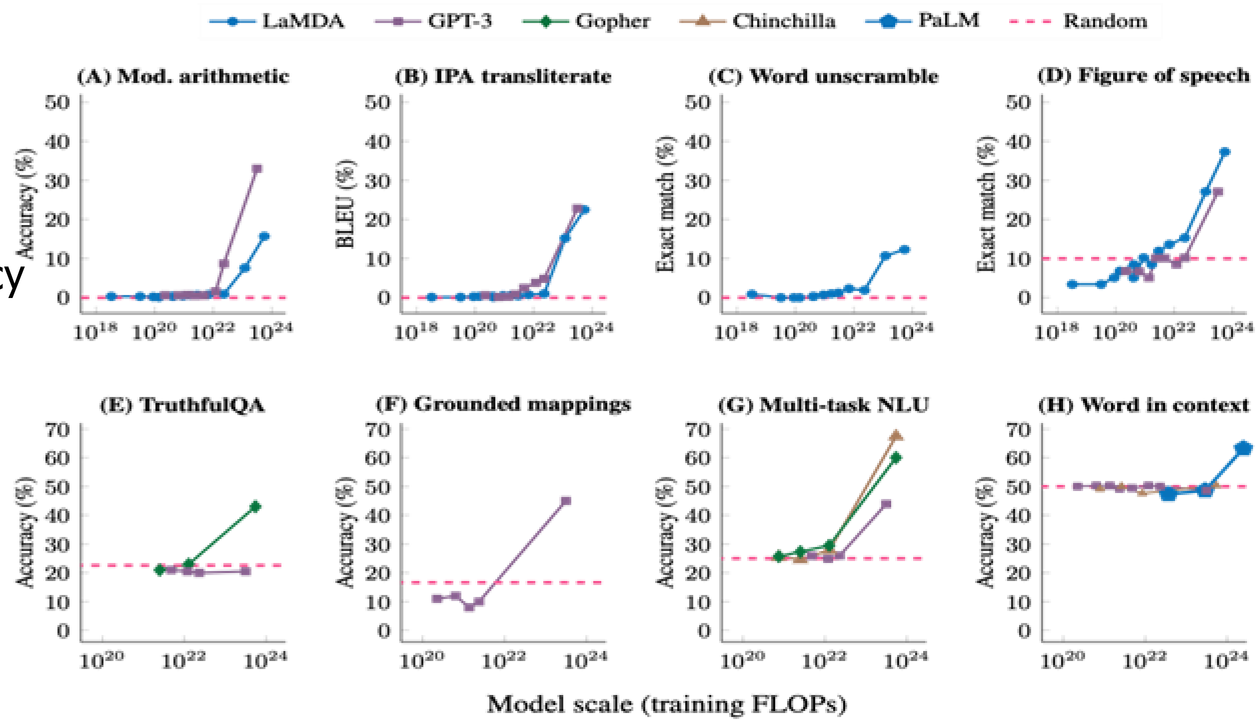
What emerges out of pre-training language models?

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

Large language models exhibit emergent abilities

- With scaling models their ICL perf consistency improves.

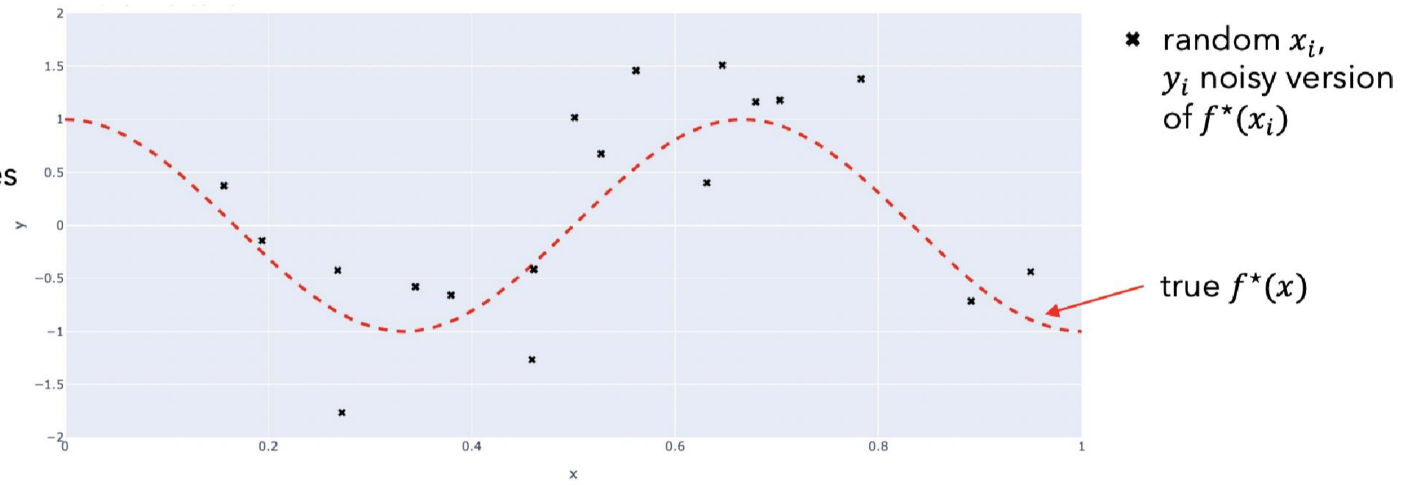


Emergence —qualitative changes in behavior with some scaling parameter

Why didn’t we scale earlier??

The Old Wisdom: Optimizing Model Capacity

n = 20 samples

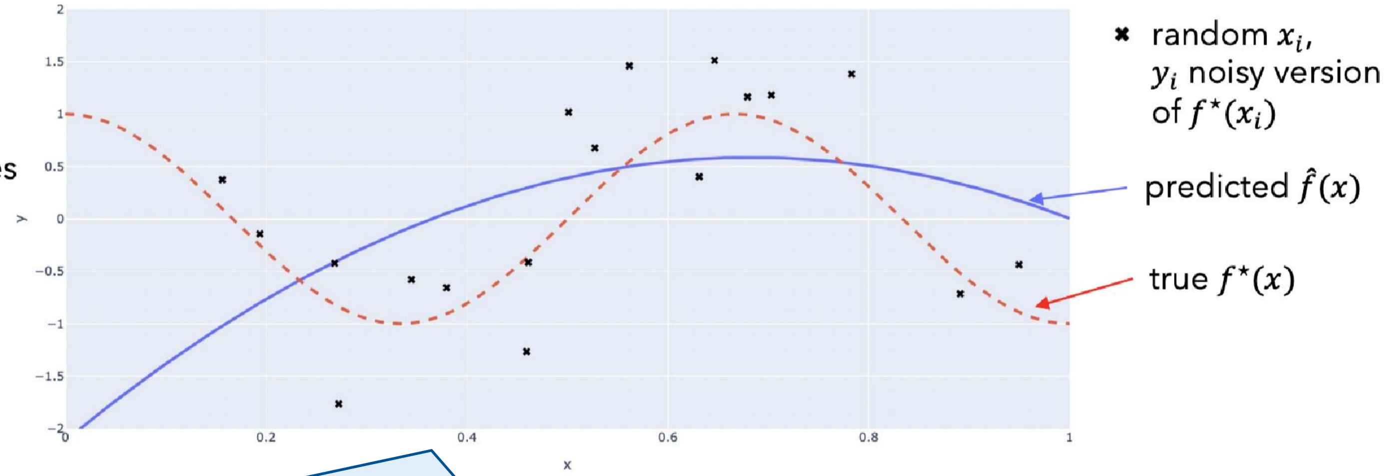


[Figure credit: NeurIPS '23 Tutorial: Reconsidering Overfitting in the Age of Overparameterized Models]

The Old Wisdom: Optimizing Model Capacity

n = 20 samples

polynomial fit
degree d = 2



Small models cannot fit perfectly.

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

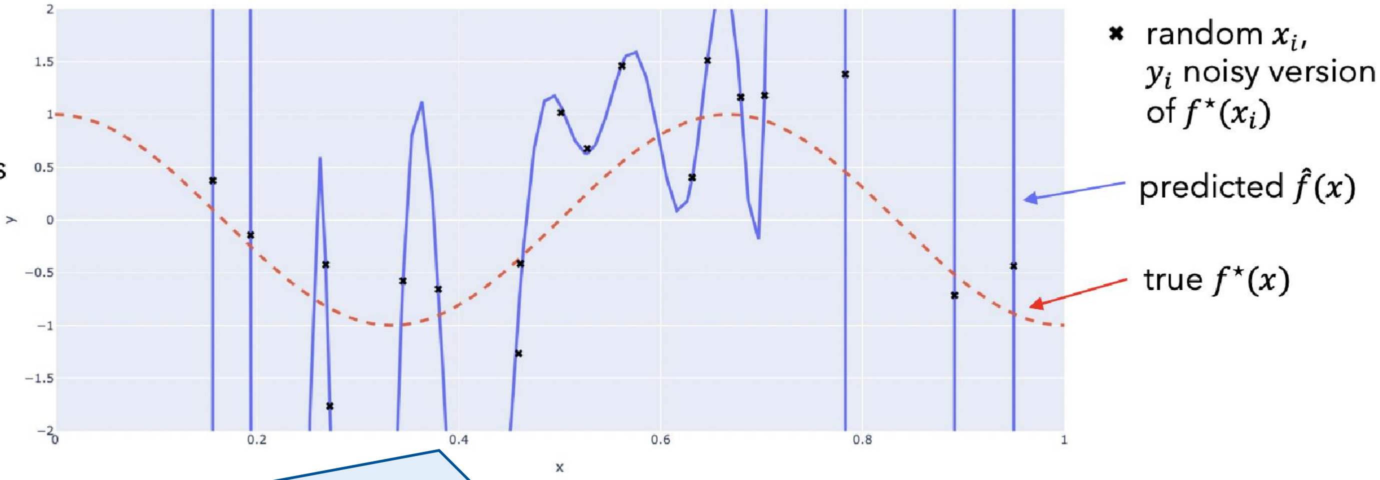
Johns Hopkins Whiting School of Engineering

[Figure credit: NeurIPS '23 Tutorial: Reconsidering Overfitting in the Age of Overparameterized Models]

The Old Wisdom: Optimizing Model Capacity

n = 20 samples

polynomial fit
degree d = 20



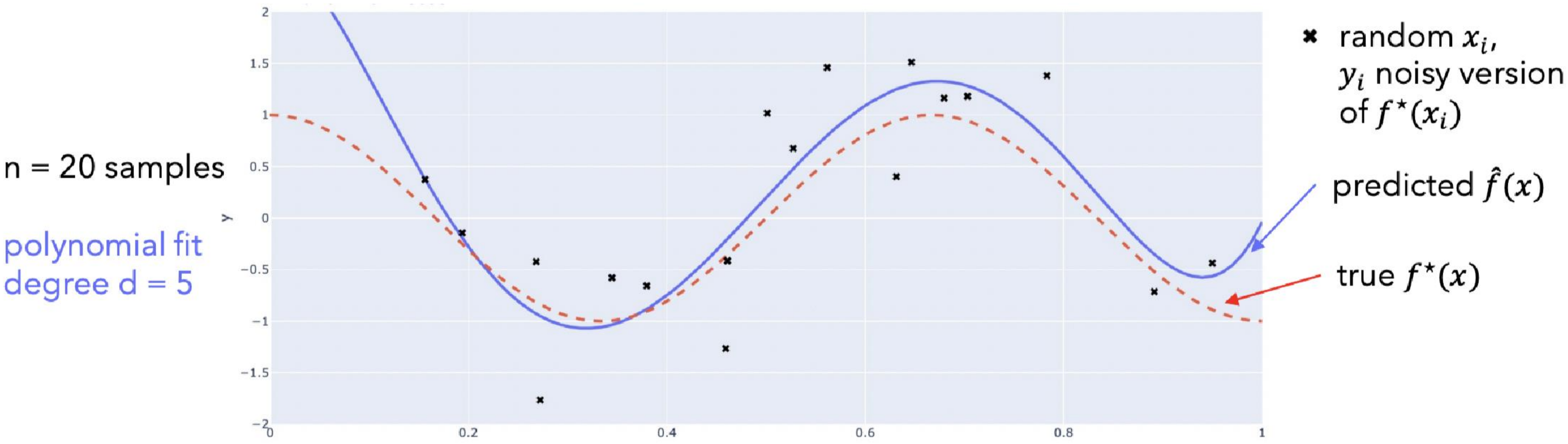
Large models fit perfectly (overfit)

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

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

[Figure credit: NeurIPS '23 Tutorial: Reconsidering Overfitting in the Age of Overparameterized Models]

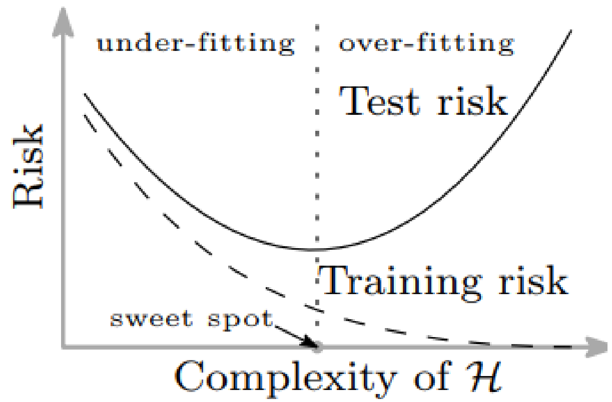
The Old Wisdom: Optimizing Model Capacity



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

[Figure credit: NeurIPS '23 Tutorial: Reconsidering Overfitting in the Age of Overparameterized Models]

The Old Wisdom: Optimizing Model Capacity



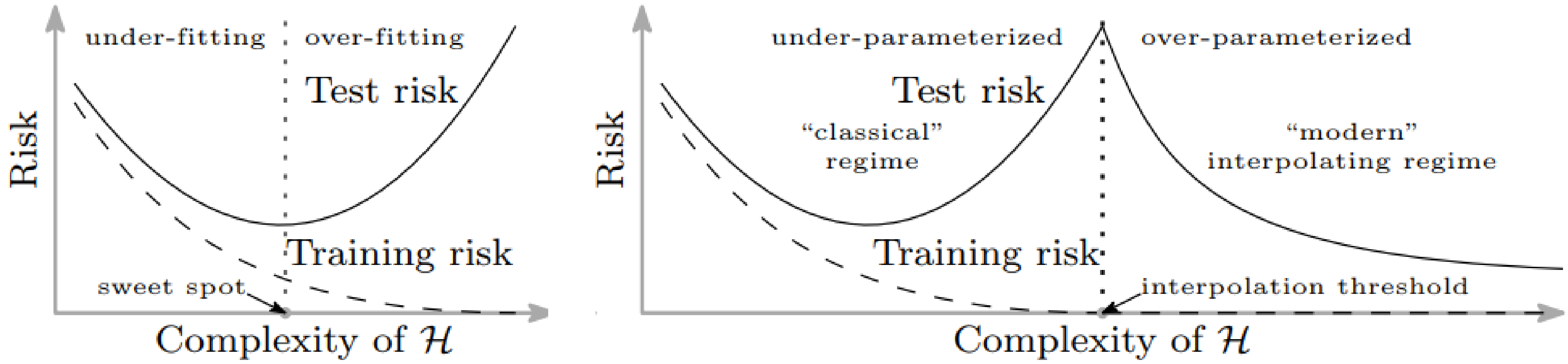
(a) U-shaped "bias-variance" risk curve

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

[Reconciling modern machine learning practice and the bias-variance trade-off, Belkin et al. 2019]

Harmless Interpolation for Large Models

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

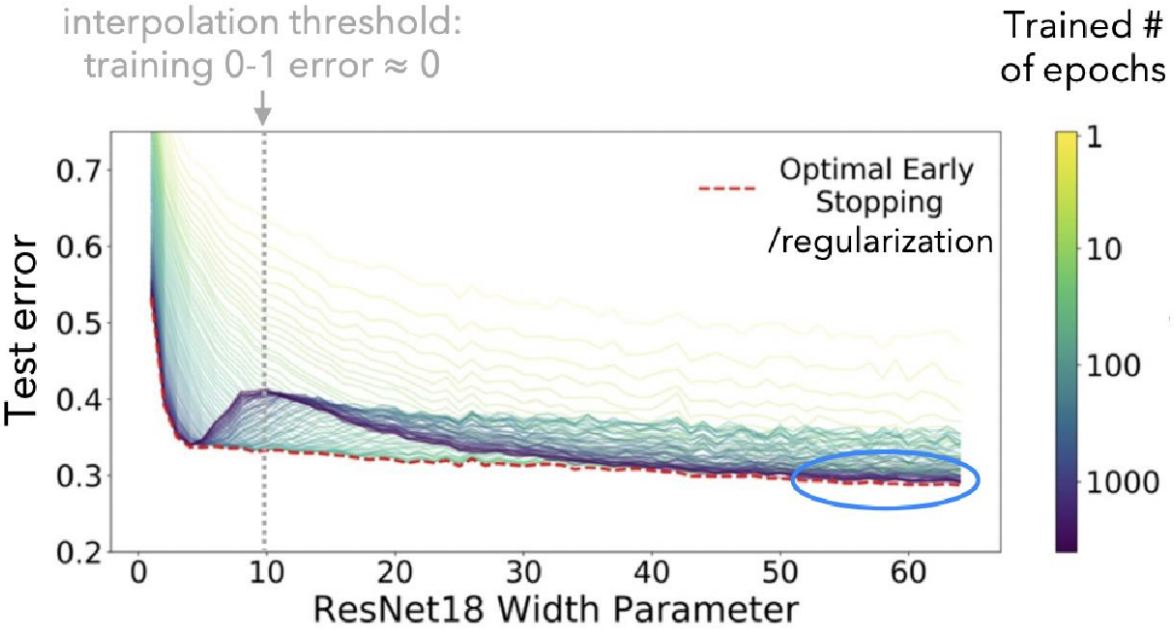


(a) U-shaped "bias-variance" risk curve

(b) "double descent" risk curve

[Reconciling modern machine learning practice and the bias-variance trade-off, Belkin et al. 2019]

Harmless Interpolation for Large Models

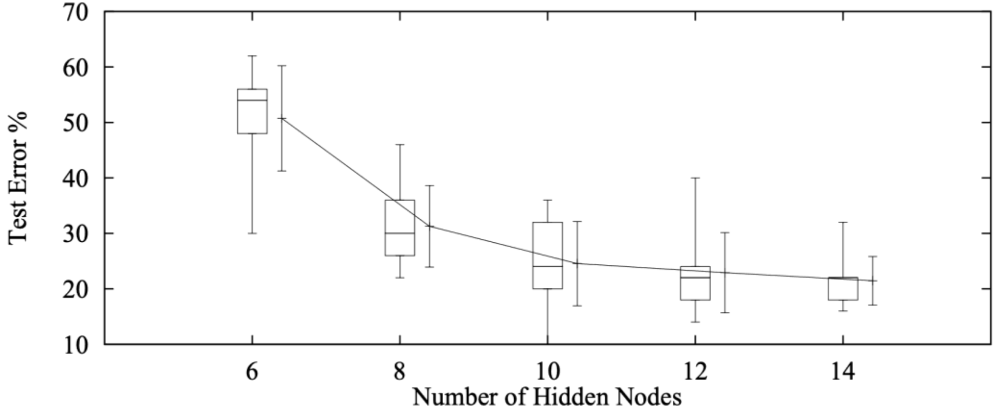


compare dark blue
(at convergence)
with red dashed
(best stopping time)

[Nakkiran, Kaplun, Bansal, Yang, Barak, Sutskever '20]

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

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

[Lawrence, Giles, and Tsoi 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)

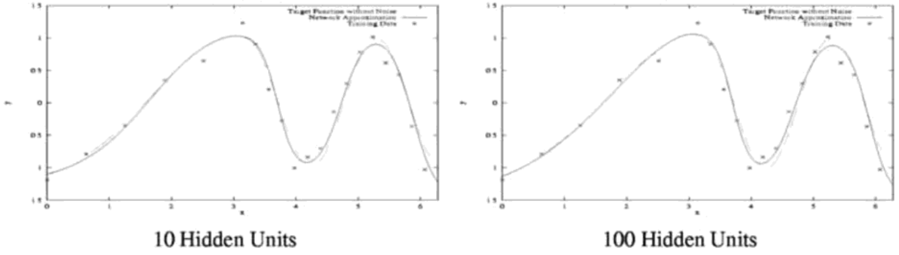


Figure 3: MLP approximation using backpropagation (BP) training of data from Equation 1 as the number of hidden units is increased. No significant overfitting can be seen.

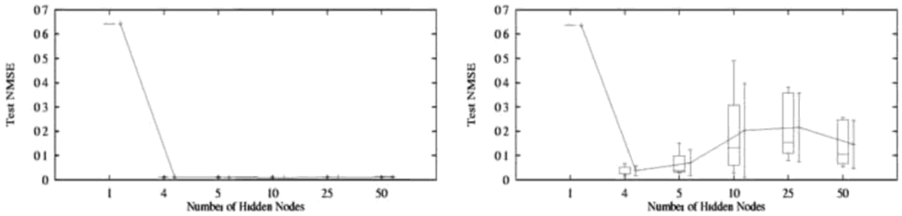


Figure 4: Test Normalized Mean Squared Error for MLPs trained with BP (left) and CG (right). Results are shown with both box-whiskers plots and the mean plus and minus one standard deviation.

[Caruana, Lawrence, and Giles 2000]

- Overfitting not bad: double descent phenomenon

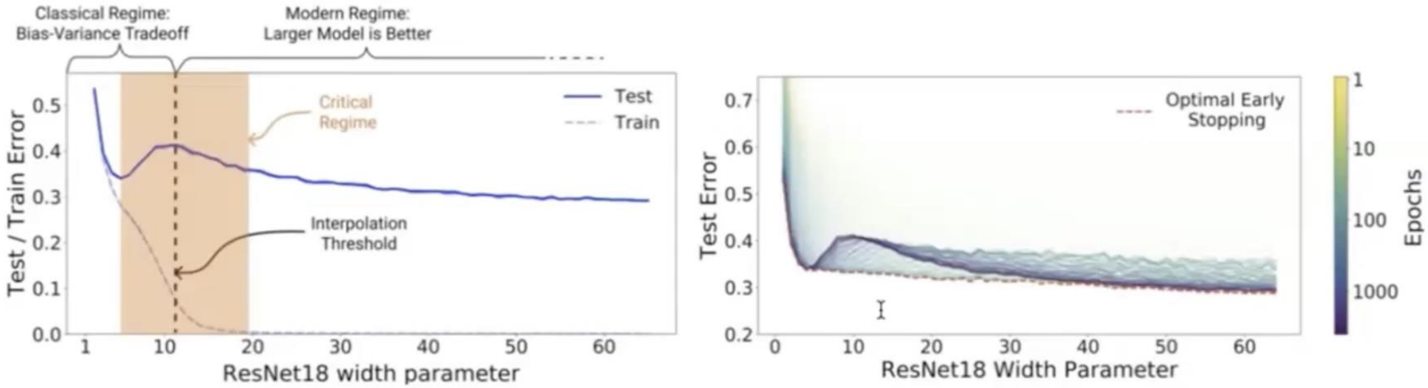


Figure 1: **Left:** Train and test error as a function of model size, for ResNet18s of varying width on CIFAR-10 with 15% label noise. **Right:** Test error, shown for varying train epochs. All models trained using Adam for 4K epochs. The largest model (width 64) corresponds to standard ResNet18.

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

When Scale Does Not Help …

When Scaling Does NOT Help

- https://arxiv.org/pdf/2306.09479.pdf

Is Scale All You Need?

Getting the most out of your GPUs

- Someone (perhaps earlier when discuss training on GPUs via PyTorch), I need to discuss various issues like
  - A brief introduction into how GPUs work.
  - GPU throughput
  - GPU utilization, which relates to FLOPS
  - GPU memory bandwidth

- https://www.cs.princeton.edu/courses/archive/fall22/cos597G/lectures/lec12.pdf
- https://arxiv.org/pdf/2109.05472.pdf
- https://www.deep-kondah.com/scaling-large-language-models/
- https://arxiv.org/pdf/1712.00409.pdf
- https://www.cirm-math.fr/RepOrga/2551/Slides/Mensch.pdf
- https://ai.google/static/documents/palm2techreport.pdf
- https://arxiv.org/pdf/2205.05198.pdf
- https://www.nextbigfuture.com/2023/05/more-ai-breakthroughs-and-reaching-agi.html
- https://members.loria.fr/CGardent/publis/thesis-lescao-2023.pdf
- https://arxiv.org/pdf/2309.08520.pdf
- https://www.lesswrong.com/posts/fnjKpBoWJXcSDwhZk/what-s-the-backward-

Scaling costs

Scaling Language Models: LLMs

What does scaling mean and how is it measured?

α x Model size x Training tokens = Training compute

	Model size (# parameters)	Training data (# tokens)	Training compute (FLOPs)	Resources
BERT-base (2018)	109M	250B	1.6e20	64 TPU v2 for 4 days (16 V100 GPU for 33 hrs)
GPT-3 (2020)	175B	300B	3.1e23	~1,000x BERT-base
Gopher (2021)	280B	300B	5.8e23	NA
PaLM (2022)	540B	780B	2.5e24	6k TPU v4 for 2 months
LLaMA2 (2023)	7B-70B	2T	NA	A100 (184K – 1720K GPU hours)