A Brief Introduction to Scaling Laws

王榕甄

2025-12-24

Background

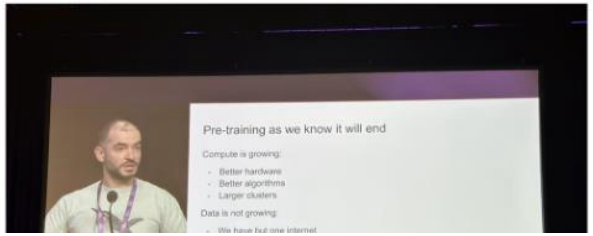
Ilya Sutskever发声：预训练将结束，数据压榨到头了

关注大模型的 Founder Park

2024年12月14日 15:57 北京 97人

Ilya 终于现身了，而且一上来就有惊人之语。

本周五，OpenAI 的前首席科学家 Ilya Sutskever 在全球 AI 顶会上表示：「我们能获得的数据已经到头，不会再有更多了。」



Ilya 小题大作？「预训练终结」≠ Scaling Law 撞墙？

Pro会员通讯 机器之心 2024年12月22日 09:02 北京 41人

本文来自PRO会员通讯内容，文末关注「机器之心PRO会员」，查看更多专题解读。

前 OpenAI 首席科学家 Ilya Sutskever 在 NeurIPS 2024 会议上作主题演讲，分享了他关于人工智能的未来发展方向，特别是围绕数据峰值的问题、预训练模型的局限性、以及下一代 AI 模型的自主性和推理能力等方面的看法。

Scaling时代终结了，Ilya Sutskever刚刚宣布

机器之心 2025年11月26日 09:33 四川 511人

机器之心报道

机器之心编辑部

「Scaling 时代已经终结。」

当这句话出自 Ilya Sutskever 之口时，整个 AI 社区都无法忽视。作为 Safe Superintelligence Inc. 的创始人，他在最新访谈中抛出的这一断言，不仅令业内震惊，更收获了诸多重量级人物的共鸣。

Scaling Law没死！Gemini核心大佬爆料，谷歌已有颠覆性密钥

新智元 新智元 2025年12月20日 09:15 北京 291人

新智元报道

编辑：Aeneas 倾倾

【新智元导读】谷歌大模型将迎颠覆升级！Gemini负责人爆料：长上下文效率与长度双重突破在即，注意力机制迎来惊人发现。Scaling Law未死，正加速演变！

谷歌又要有重大突破了？

最近，Google DeepMind的Gemini预训练负责

Outline

- Preliminaries
- Trend prediction
  - LLM scaling laws: Kaplan et al, 2020
  - Modifications in compute-optimal scenario: Hoffman et al, 2022 and more
- Hyperparameter selection
- Conclusion

Preliminaries

Notations

- L is the test loss (training loss if specified)
- N is the number of parameters, also called the model size
- D is the number of tokens in the training set, also called the dataset size
- S is the number of training steps, B is the batch size
  - When training less than 1 epoch, we take $D = BS$
- C is the number of floating point operations (FLOPs)
  - Training a Transformer of size N with B and S typically needs $C \approx 6NBS$

Calculation of N and C

Operation	Parameters	FLOPs per Token
Embed	$(n_{\text{vocab}} + n_{\text{ctx}})\, d_{\text{model}}$	$4d_{\text{model}}$
Attention: QKV	$n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$	$2n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$
Attention: Mask	—	$2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$
Attention: Project	$n_{\text{layer}} d_{\text{attn}} d_{\text{model}}$	$2n_{\text{layer}} d_{\text{attn}} d_{\text{embd}}$
Feedforward	$n_{\text{layer}} 2d_{\text{model}} d_{\text{ff}}$	$2n_{\text{layer}} 2d_{\text{model}} d_{\text{ff}}$
De-embed	—	$2d_{\text{model}} n_{\text{vocab}}$
Total (Non-Embedding)	$N = 2d_{\text{model}} n_{\text{layer}} (2d_{\text{attn}} + d_{\text{ff}})$	$C_{\text{forward}} = 2N + 2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$

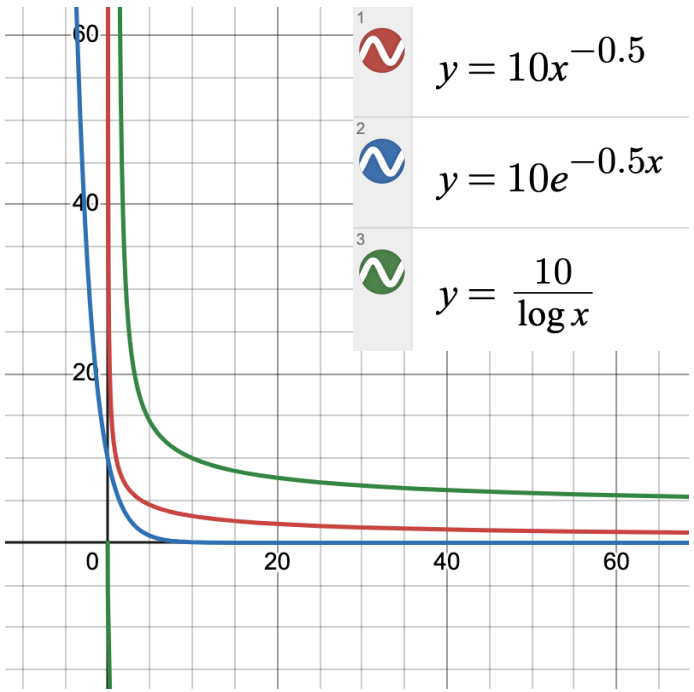
Table 1 Parameter counts and compute (forward pass) estimates for a Transformer model. Sub-leading terms such as nonlinearities, biases, and layer normalization are omitted.

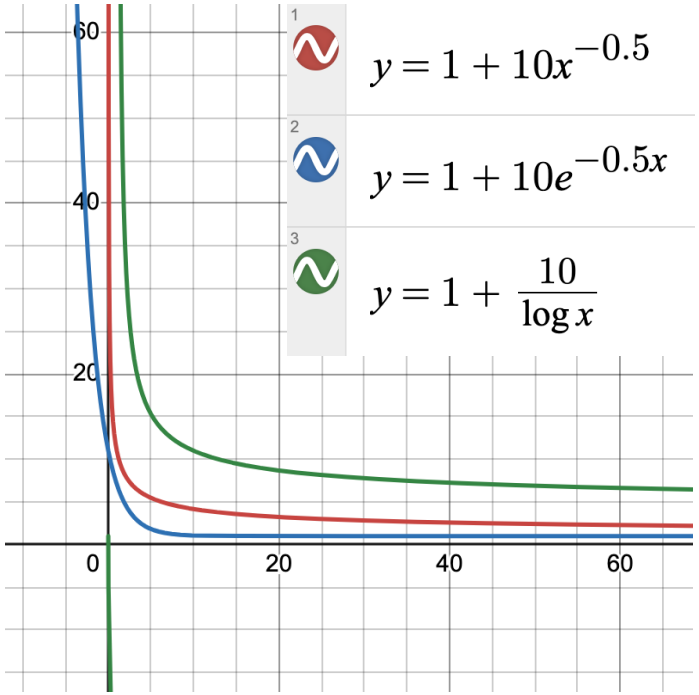
[Scaling Laws for Neural Language Models, Kaplan et al, 2020]

Function forms

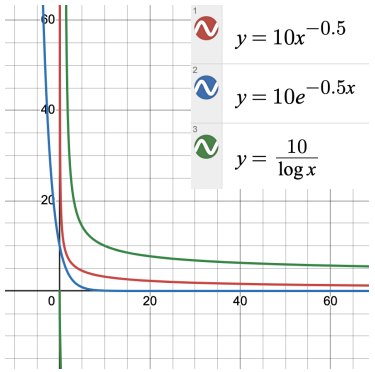
- Goal: Fitting L as a function of N, D, ..., e.g., $L(N)$

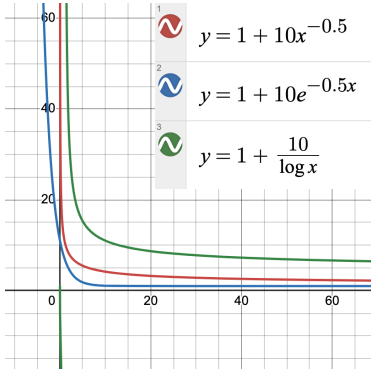
- Power: $y = c + bx^{-a}$
- Exponential: $y = c + be^{-ax}$
- Inverse-log: $y = c + \frac{b}{\log x}$
- And more…

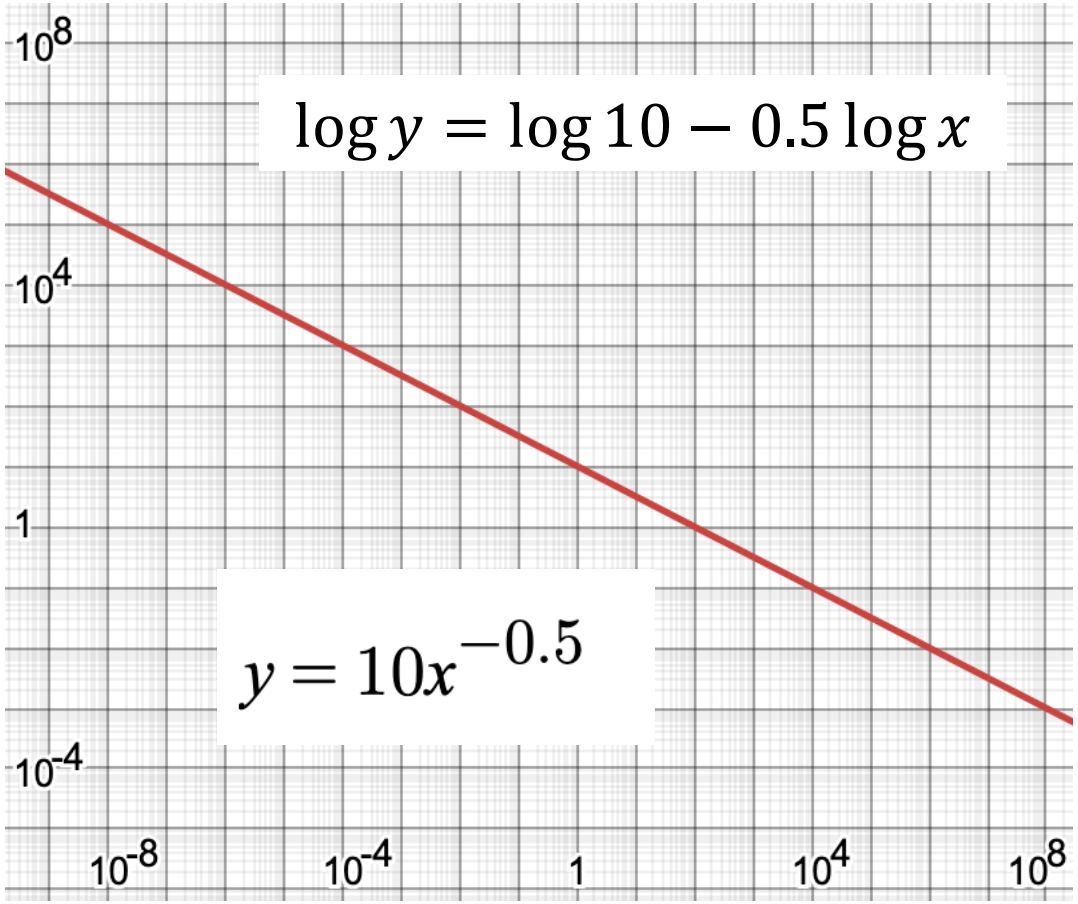


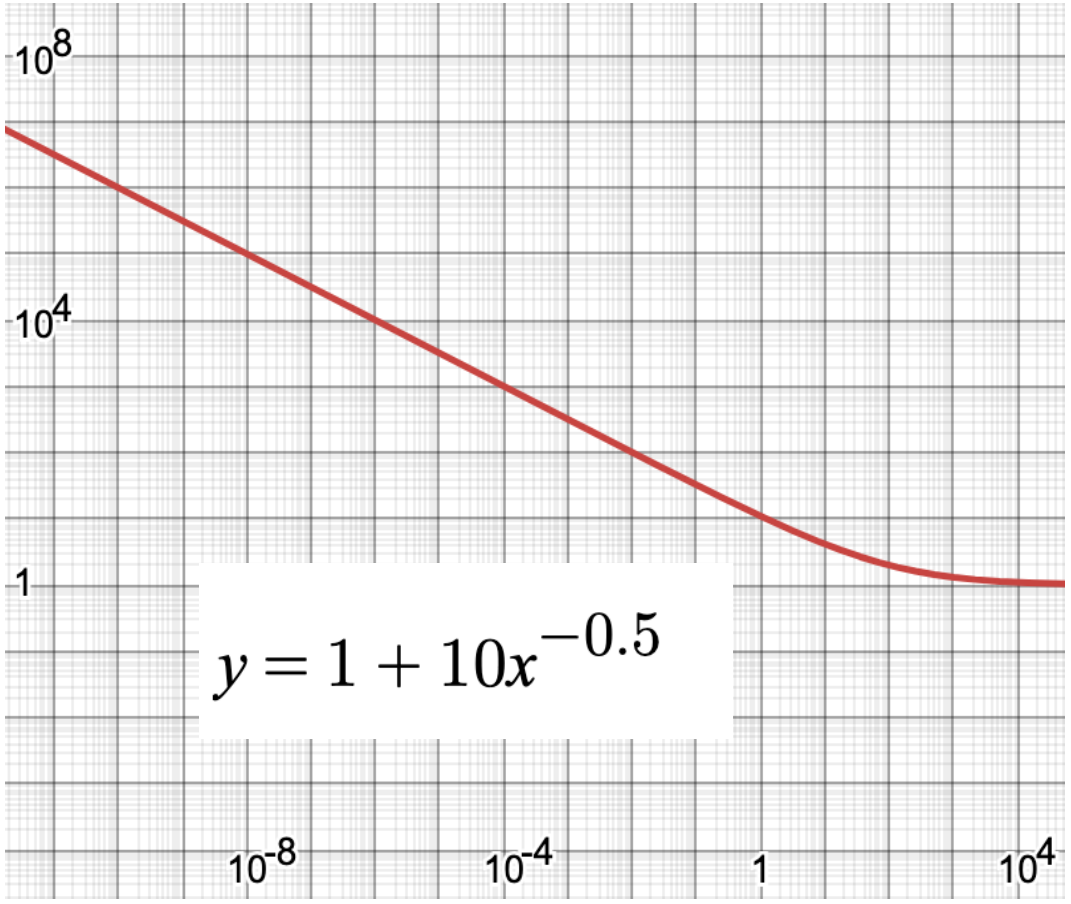


Log-log plots

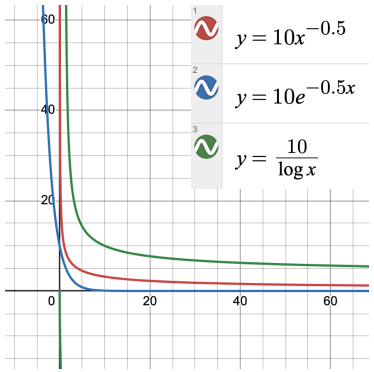


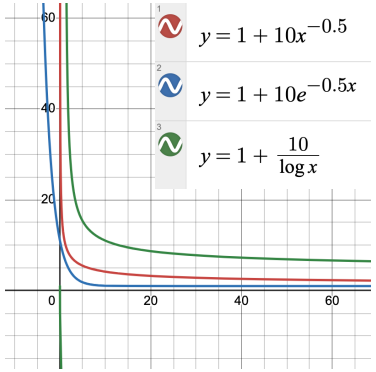


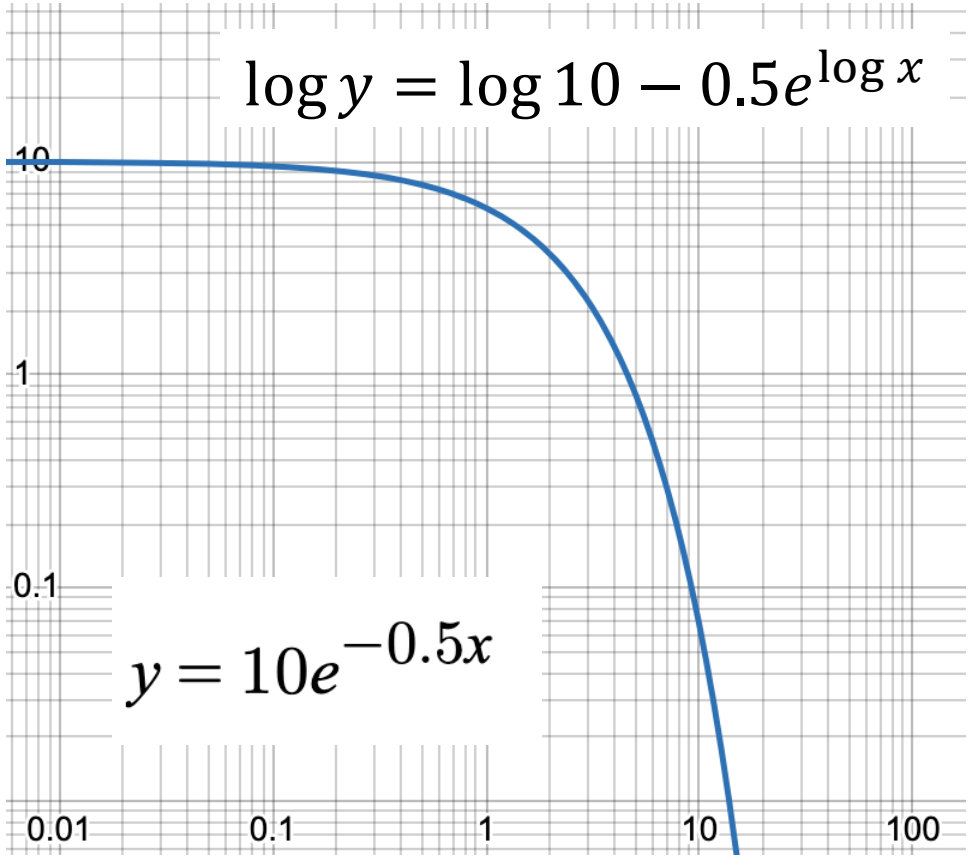


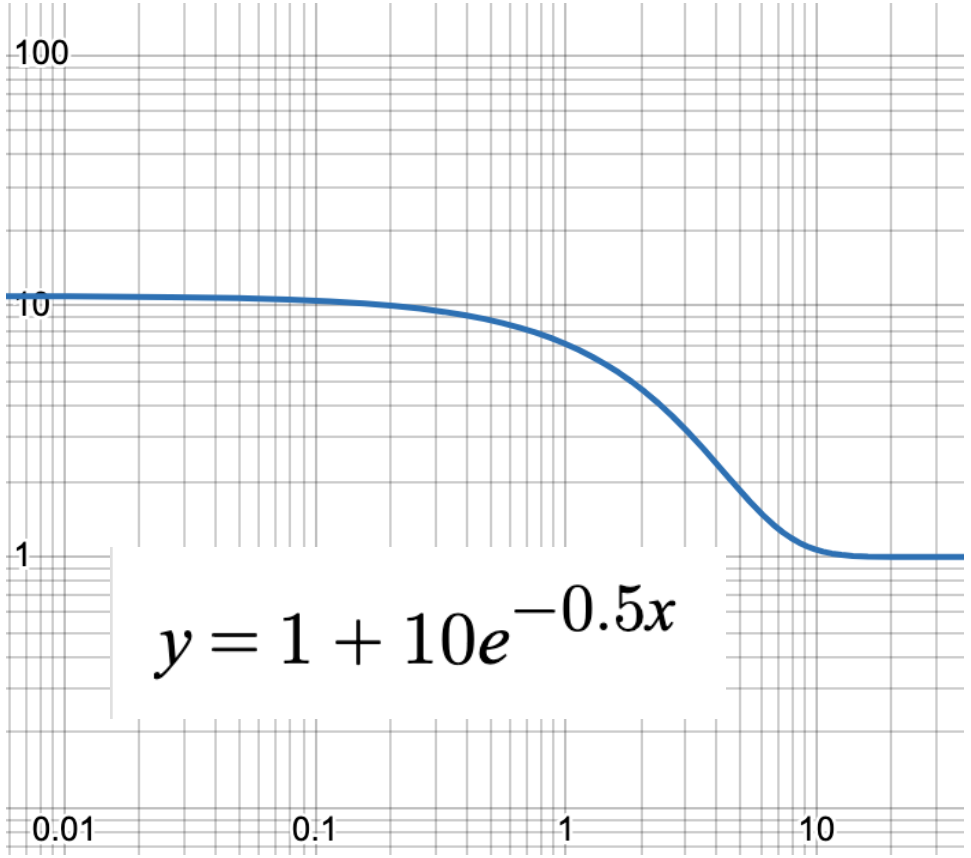


Log-log plots

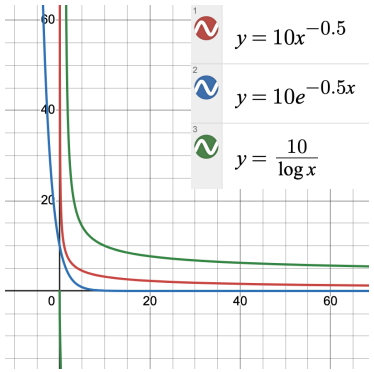


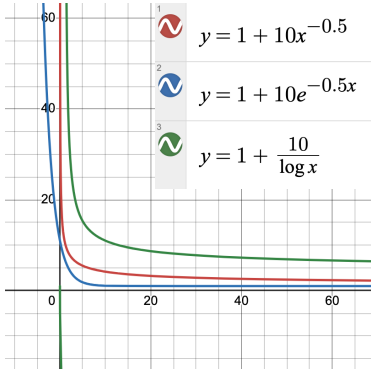


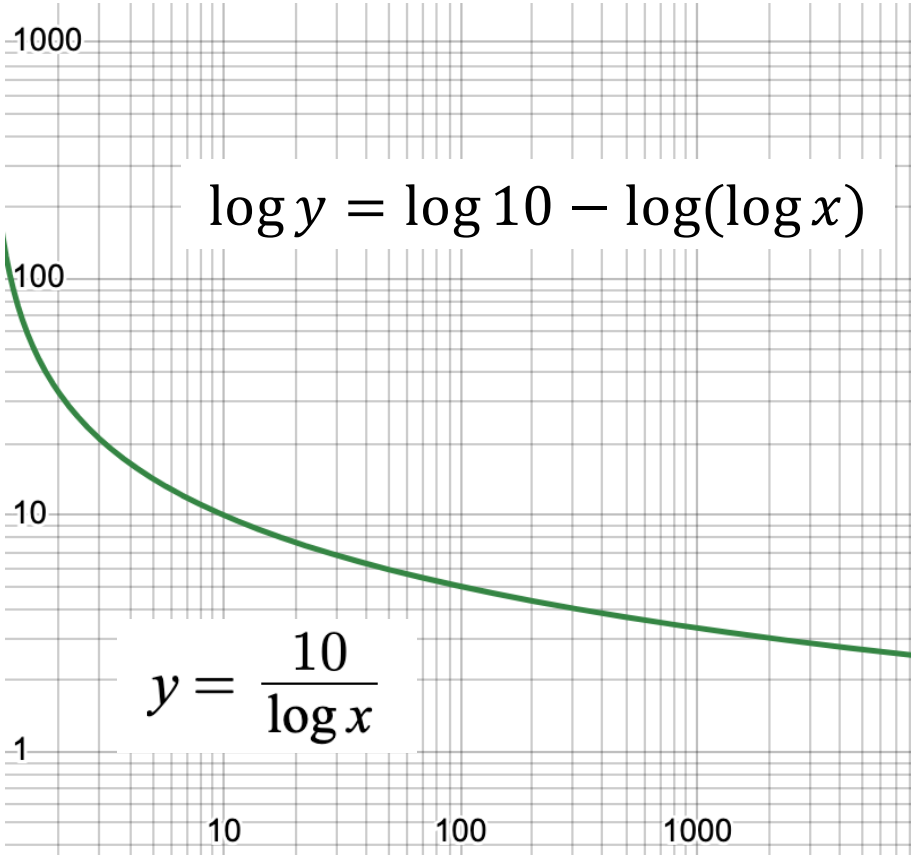


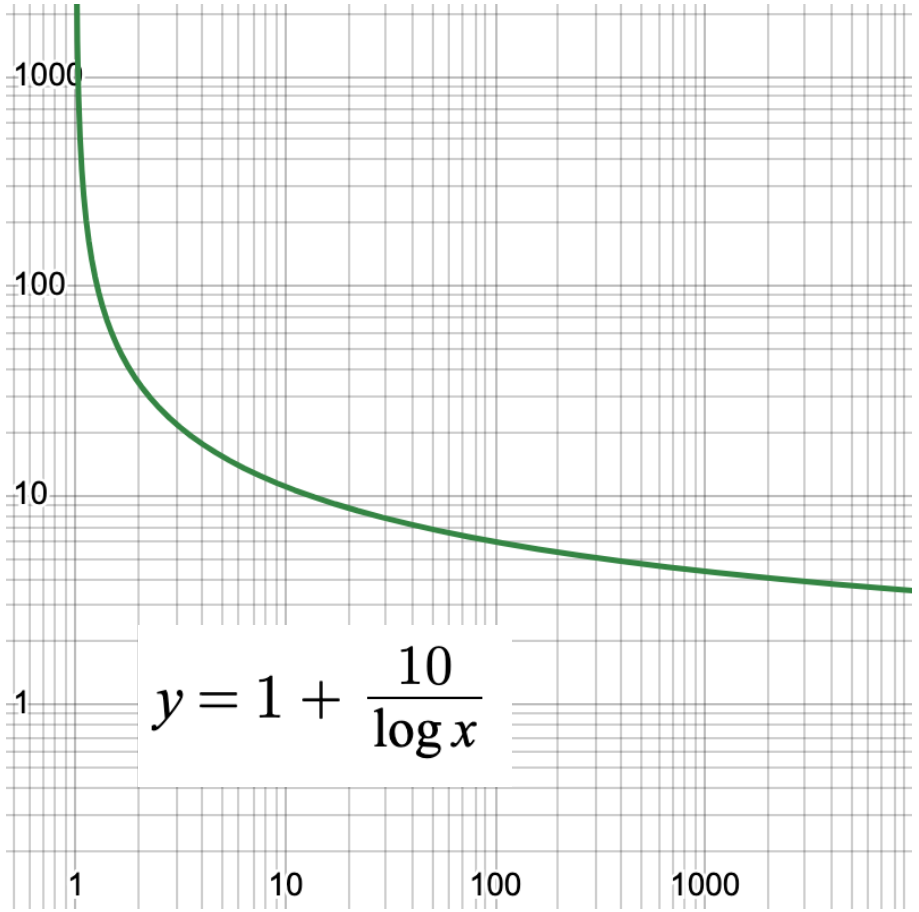


Log-log plots









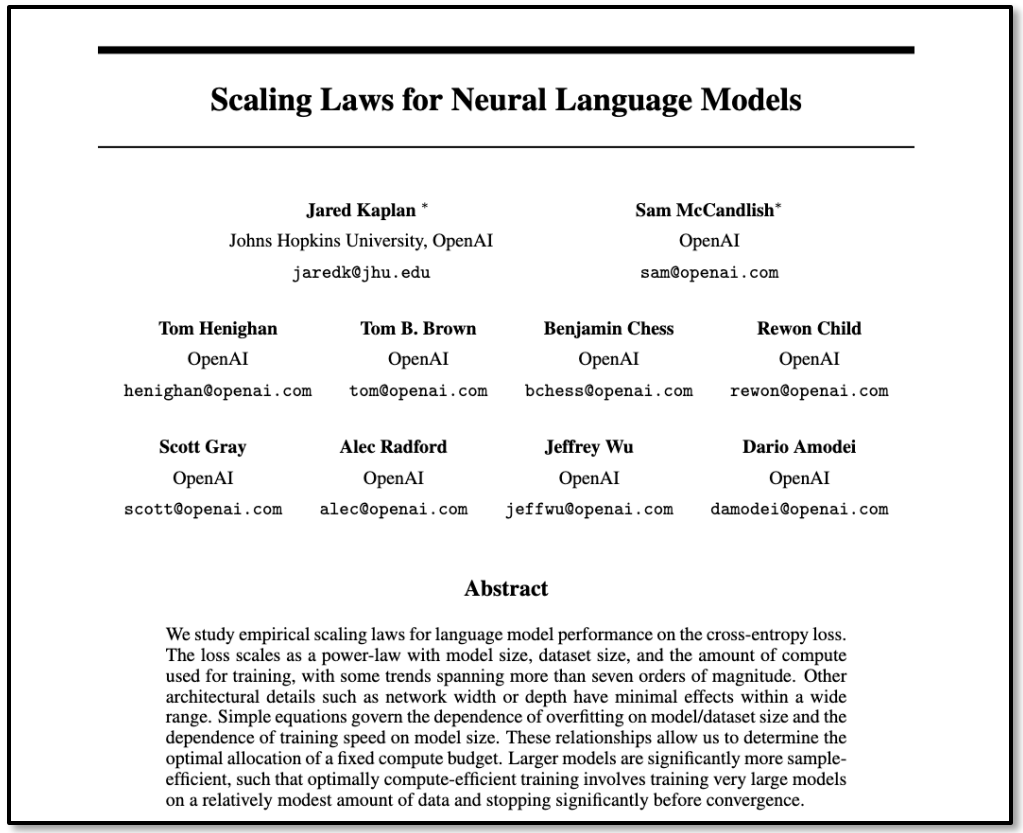
Preview of related works

- 1712, Baidu, Hestness at al, *Deep Learning Scaling is Predictable, Empirically*
- 1909, Rosenfeld et al, *A Constructive Prediction of the Generalization Error Across Scales*, **100M tokens, 41M param**
- 2001, OpenAI, Kaplan et al, *Scaling Laws for Neural Language Models*, **23B tokens, 1.5B param**
- 2005, OpenAI, *GPT-3*, **300B tokens, 175B param**
- 2010, OpenAI, Henighan et al, *Scaling Laws for Autoregressive Generative Modeling* (scaling laws across modalities)
- 2207, Google & Deepmind, Tay et al, *Scaling Laws vs Model Architectures: How does Inductive Bias Influence Scaling?*
- 2203, Deepmind, Hoffmann at al, *Training Compute-Optimal Large Language Models*, **1.5T tokens, 70B param**
- 2305, Hugging Face, Muennighoff et al, *Scaling Data-Constrained Language Models*
- 2307, Meta, *Llama 2*, **2T tokens, 70B param**
- 2403, Gadre et al, *Language models scale reliably with over-training and on downstream tasks*
- 2404, Tsinghua & Modelbest, *MiniCPM*, **1.1T tokens, 2.4B param** (proposing WSD)
- 2404, Epoch AI, Dey et al, *Chinchilla Scaling: A replication attempt*
- 2407, Meta, *Llama 3*, **15.6T tokens, 405B param**

LLM scaling laws: Kaplan et al, 2020

Kaplan et al, 2020

- Empirical scaling laws for **Autoregressive Transformer Language Models**



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

Abstract

We study empirical scaling laws for language model performance on the cross-entropy loss. The loss scales as a power-law with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude. Other architectural details such as network width or depth have minimal effects within a wide range. Simple equations govern the dependence of overfitting on model/dataset size and the dependence of training speed on model size. These relationships allow us to determine the optimal allocation of a fixed compute budget. Larger models are significantly more sample-efficient, such that optimally compute-efficient training involves training very large models on a relatively modest amount of data and stopping significantly before convergence.

For easier reference, we provide a summary below of the key trends described throughout the paper.

Parameters	Data	Compute	Batch Size	Equation
N	∞	∞	Fixed	$L(N) = (N_c/N)^{\alpha_N}$
∞	D	Early Stop	Fixed	$L(D) = (D_c/D)^{\alpha_D}$
Optimal	∞	C	Fixed	$L(C) = (C_c/C)^{\alpha_C}$ (naive)
N_{opt}	D_{opt}	$C_{\min}$	$B \ll B_{\text{crit}}$	$L(C_{\min}) = \left(C_c^{\min}/C_{\min}\right)^{\alpha_C^{\min}}$
N	D	Early Stop	Fixed	$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$
N	∞	S steps	B	$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}(S,B)}\right)^{\alpha_S}$

Table 4

We will look at these results starting from the **experiments**

General training setup

- Decoder-only Transformers, with 768 ~ 1.5B **non-embedding** parameters
- Vocabulary size = 50257, context length = 1024
- Optimizer: Adam
- **Batch size** = 512 sequences of 1024 tokens = 2^{19} tokens
- **#Steps** $= 2.5 \times 10^5$
- **LR schedule**: 3000 step linear warmup followed by a cosine decay with **cycle length** = #Steps $= 2.5 \times 10^5$
- LR: see details in Appendix D.6

$$\mathrm{LR}(N) \approx 0.003239 + -0.0001395 \log(N) \qquad \text{(D.1)}$$

Summary of experiments

For easier reference, we provide a summary below of the key trends described throughout the paper.

	Parameters	Data	Compute	Batch Size	Equation
Set 1	N	∞	∞	Fixed	$L(N) = (N_c/N)^{\alpha_N}$
	∞	D	Early Stop	Fixed	$L(D) = (D_c/D)^{\alpha_D}$
	Optimal	∞	C	Fixed	$L(C) = (C_c/C)^{\alpha_C}$ (naive)
	N_{opt}	D_{opt}	C_{min}	$B \ll B_{\text{crit}}$	$L(C_{\text{min}}) = \left(C_c^{\text{min}}/C_{\text{min}}\right)^{\alpha_C^{\text{min}}}$
Set 2	N	D	Early Stop	Fixed	$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$
	N	∞	S steps	B	$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\text{min}}(S,B)}\right)^{\alpha_S}$

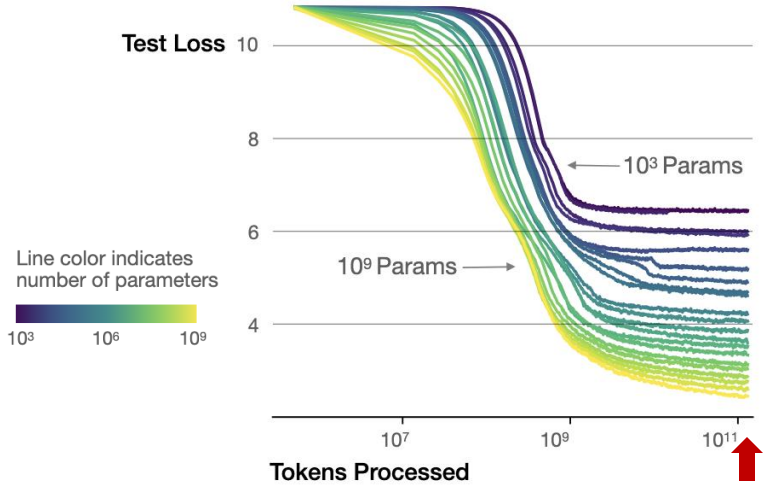
Table 4

Summary of experiments: Set 1

- Train models of different **sizes *N*** on a large dataset (supposed to be infinite)

N	*S*	*D*
768	fixed, 2.5×10^5	fixed, 23 B
3000	fixed, 2.5×10^5	fixed, 23 B
…	…	…
1.5 B	fixed, 2.5×10^5	fixed, 23 B

~39 different *N*, ~39 runs in total



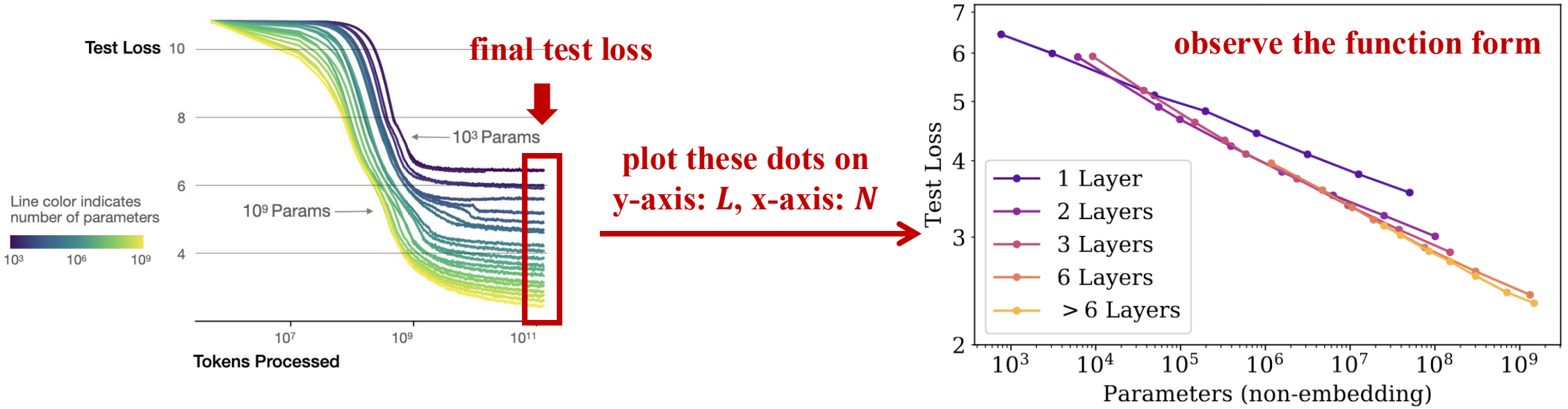
$(2.5 \times 10^5) \times 2^{19}$ = 131B tokens, 5.7 epoch

- Observe "near convergence" on training loss and "no overfitting" on test loss, thus get the (approximately) **lowest possible test loss** for each *N*

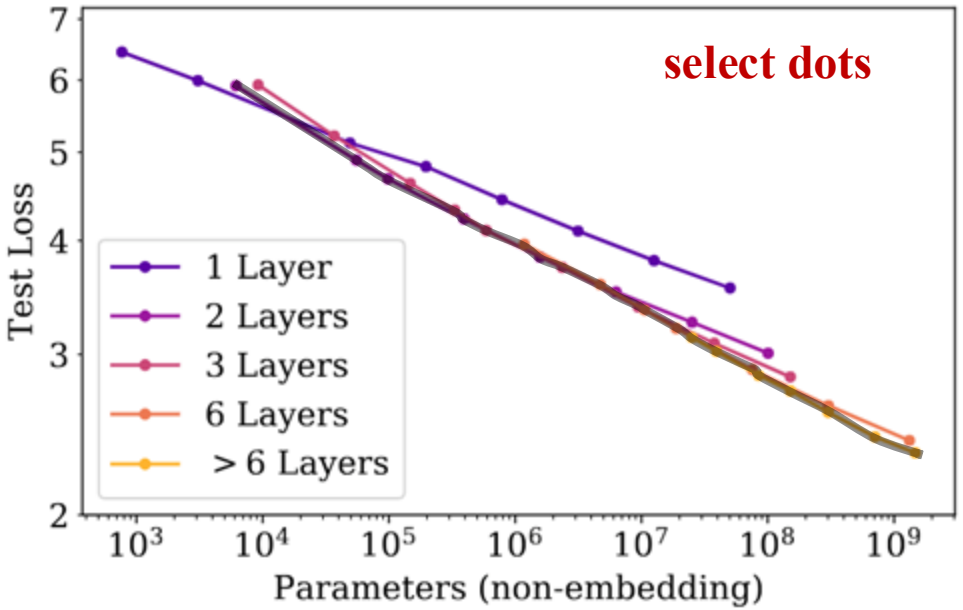
Model-constrained scaling law: $L(N)$

Parameters	Data	Compute	Batch Size	Equation
N	∞	∞	Fixed	$L(N) = (N_c/N)^{\alpha_N}$

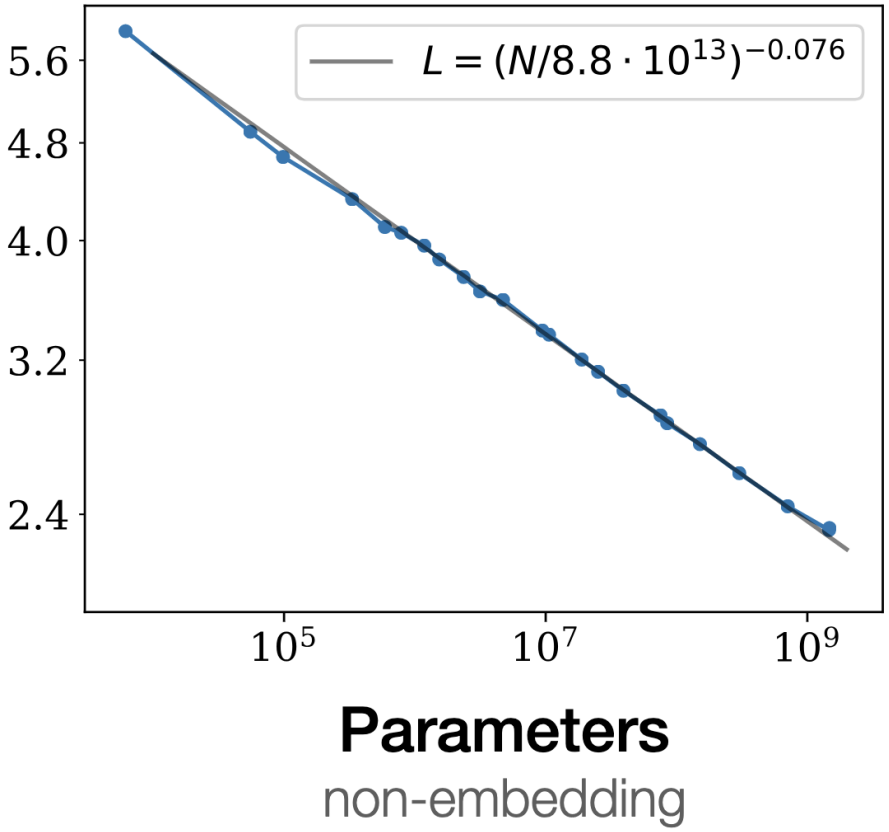
- The lowest loss ***L*** achievable by a model of fixed size ***N***, assuming unlimited data and compute



Model−constrained scaling law: $L(N)$



fit power law



$$L(N) = (N_c/N)^{\alpha_N}\ ;\ \ \alpha_N \sim 0.076,\quad N_c \sim 8.8 \times 10^{13}\ \text{(non-embedding parameters)} \qquad (1.1)$$

Compute−constrained scaling law: $L(C)$

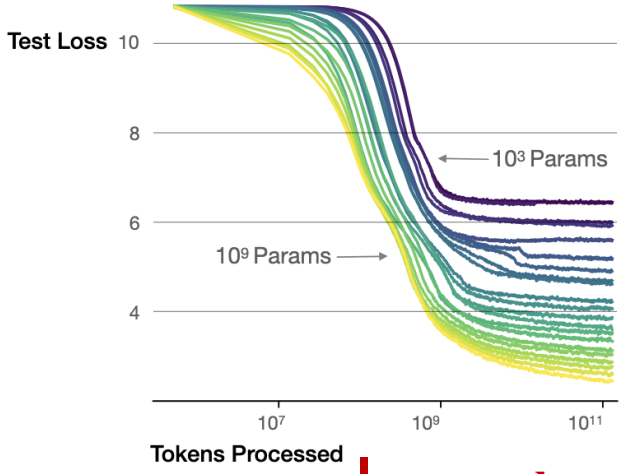
Parameters	Data	Compute	Batch Size	Equation
Optimal	∞	C	Fixed	$L(C) = (C_c/C)^{\alpha_C}$ (naive)

⬇

Recalling $C = 6NBS$, this implicitly requires $\boldsymbol{N^*(C), S^*(C)}$ to achieve $L^*(C)$

- The lowest loss $\boldsymbol{L}$ achievable with a fixed compute budget $\boldsymbol{C}$, determined by **optimally allocating resources between model size and training tokens**

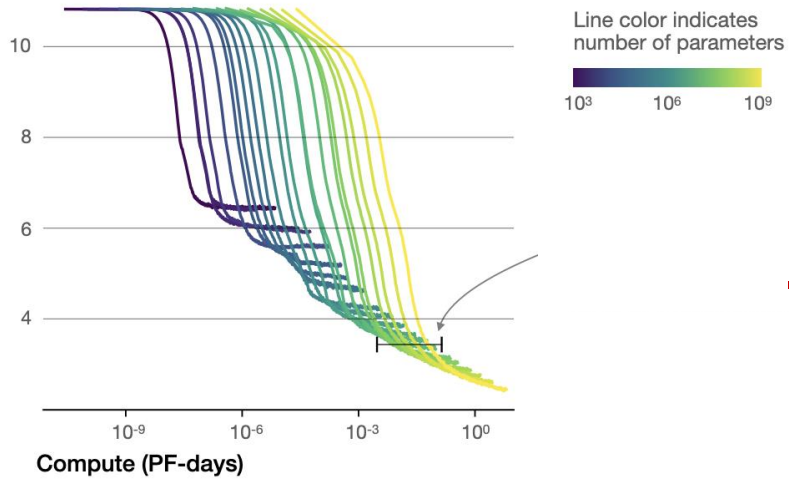
Compute−constrained scaling law: $L(C)$



$$L(C) = (C_c/C)^{\alpha_C}; \qquad \alpha_C \sim 0.057, \qquad C_C \sim 2 \times 10^7$$

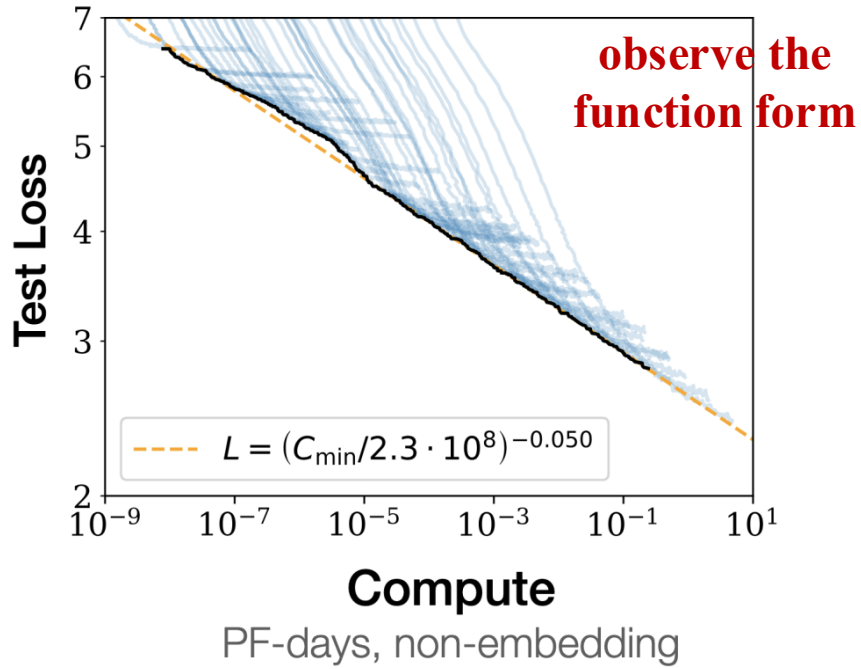
fit power law

change the x-axis
from BS to $C = 6NBS$



extract the envelope

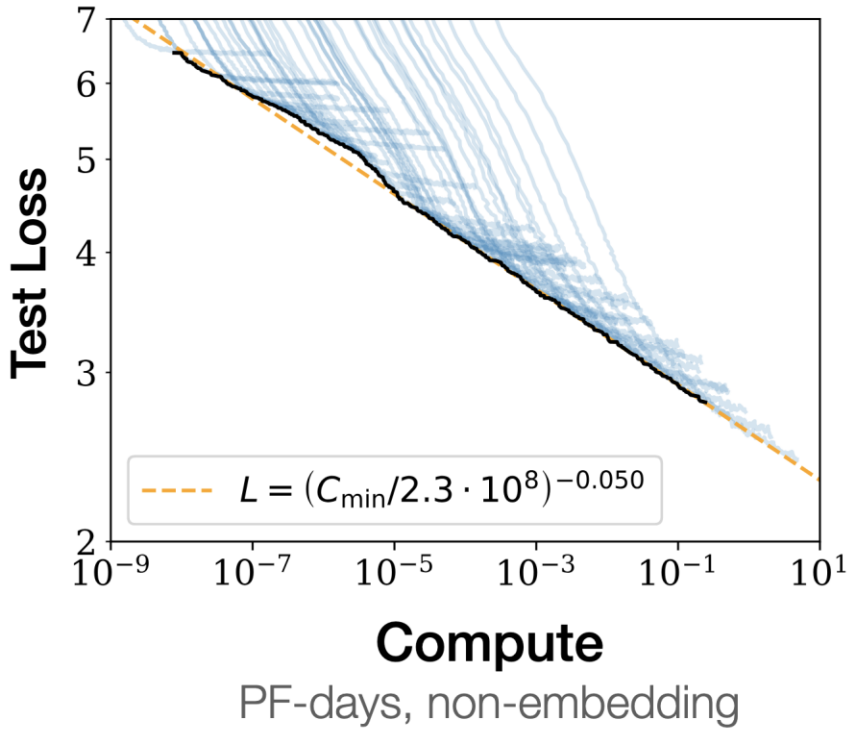
observe the
function form

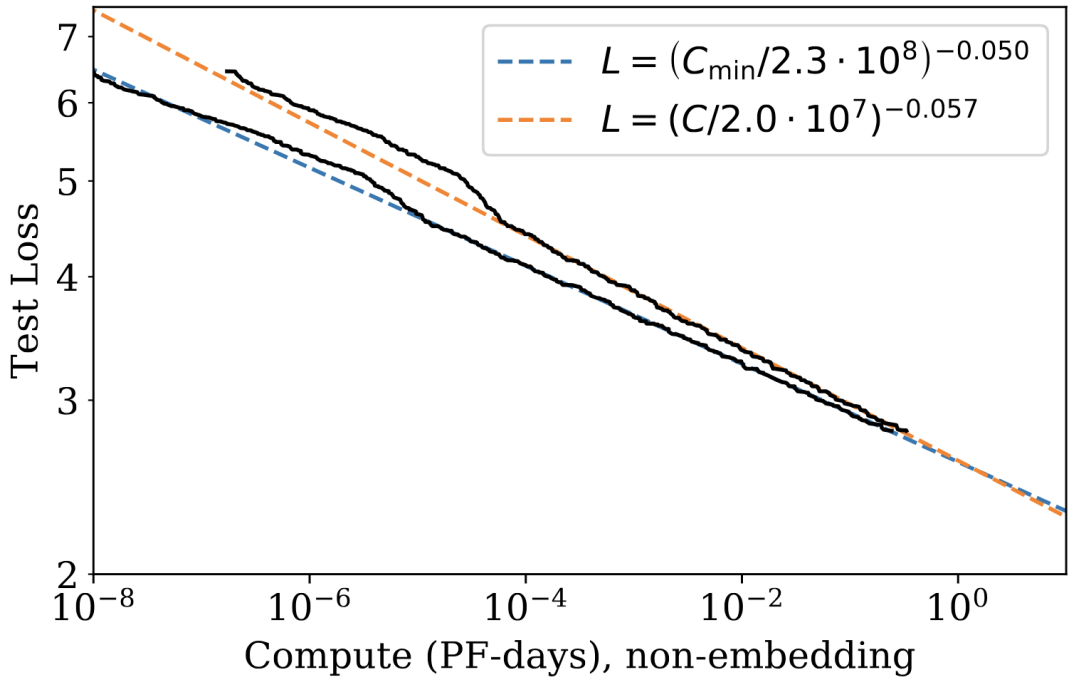


Compute−constrained scaling law: $L(C)$

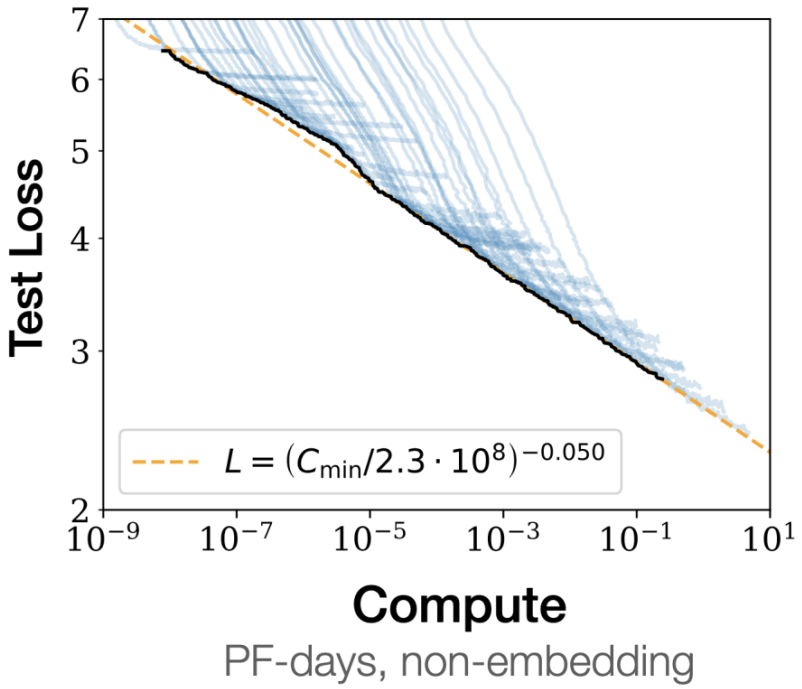
$$C_{\min}(C) \equiv \frac{C}{1 + B/B_{\text{crit}}(L)} \qquad B_{\text{crit}}(L) \approx \frac{B_*}{L^{1/\alpha_B}} \qquad B_* \approx 2 \times 10^8 \text{ and } \alpha_B \approx 0.21$$

(See Section 5 in the original paper for related details)



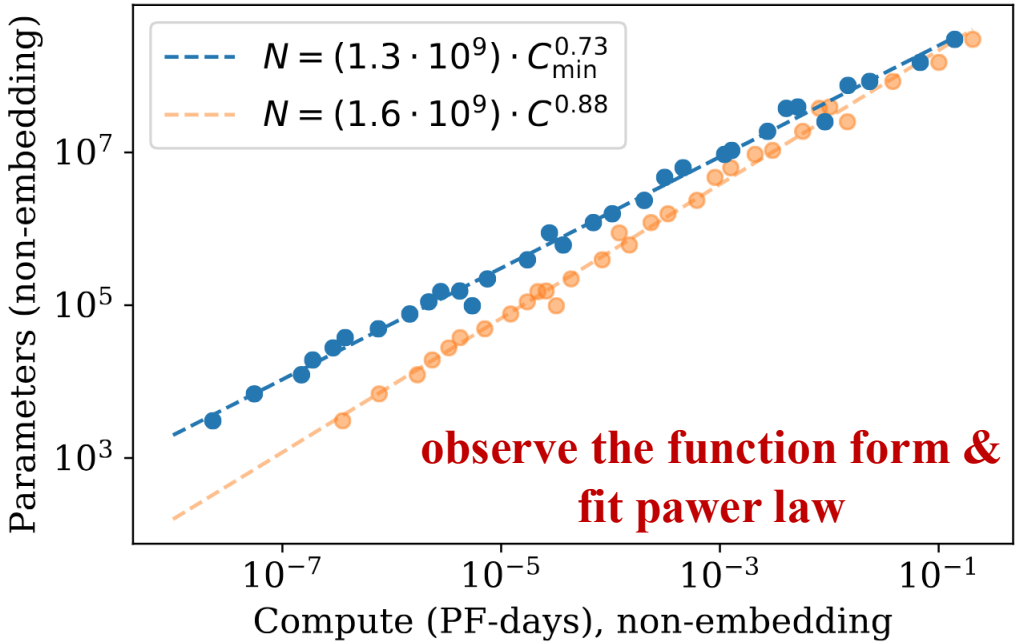


Compute-optimal model size: $N^*(C)$



for each C, record the optimal N

plot these dots on y-axis: N, x-axis: C



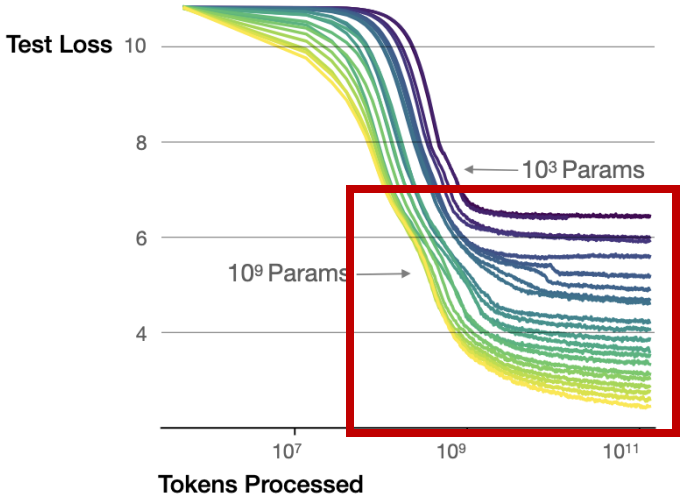
$$N^*(C_{\text{min}}) \propto C^{0.73}, \qquad (BS)^*(C_{\text{min}}) \propto C^{0.27}$$

$L(N, S)$

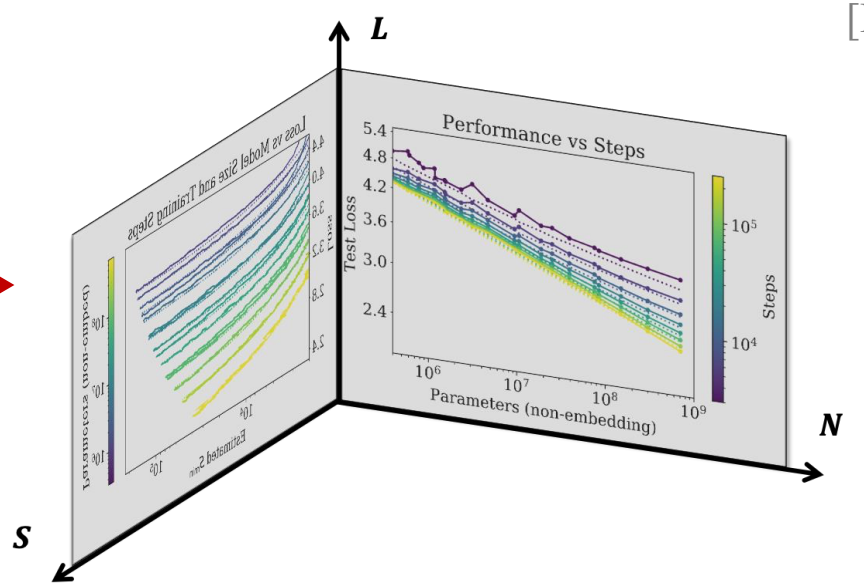
Parameters	Data	Compute	Batch Size	Equation
N	∞	S steps	B	$L(N,S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}(S,B)}\right)^{\alpha_S}$

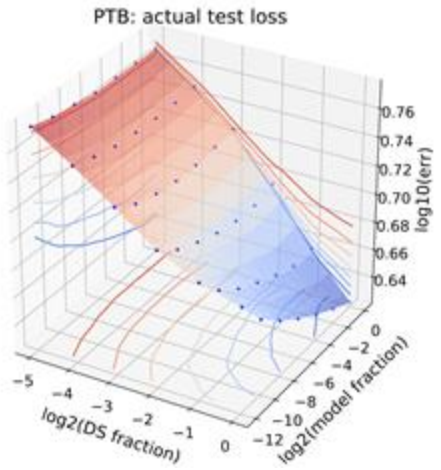
- The loss ***L*** of a model with finite parameters ***N*** updated with finite steps ***S*** on unlimited data

[Figure 13, Rosenfeld et al, 2019]



plot on a 3D graph





(3D view may look like the right graph)

$L(N, S)$

choose a two-variable function form & fit the function

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

Parameter	α_N	α_S	N_c	S_c
Value	0.077	0.76	6.5×10^{13}	2.1×10^{3}

Table 3 Fits to $L(N, S)$

close alignment on α_N

- $L(N,S) \xrightarrow{S \to \infty} L(N) \propto C^{0.077}$

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

- $L(N,S) \xrightarrow{S = C/6NB} N^*(C) \propto C^{\frac{\alpha_S}{\alpha_N+\alpha_S}} = C^{0.91}$

$$N^*(C) \propto C^{0.88}$$

Summary of experiments

For easier reference, we provide a summary below of the key trends described throughout the paper.

	Parameters	Data	Compute	Batch Size	Equation
Set 1	N	∞	∞	Fixed	$L(N) = (N_c/N)^{\alpha_N}$
	∞	D	Early Stop	Fixed	$L(D) = (D_c/D)^{\alpha_D}$
	Optimal	∞	C	Fixed	$L(C) = (C_c/C)^{\alpha_C}$ (naive)
	N_{opt}	D_{opt}	C_{min}	$B \ll B_{\text{crit}}$	$L(C_{\text{min}}) = \left(C_c^{\text{min}}/C_{\text{min}}\right)^{\alpha_C^{\text{min}}}$
Set 2	N	D	Early Stop	Fixed	$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$
	N	∞	S steps	B	$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\text{min}}(S,B)}\right)^{\alpha_S}$

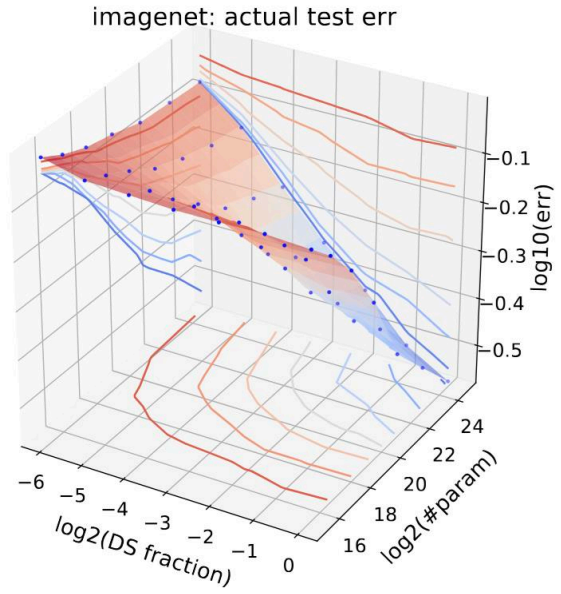
Table 4

Summary of experiments: Set 2

- Train on different combinations of **model sizes *N*** and **dataset sizes *D*** and record the **early-stopping test loss** (stopped training once the test loss ceased to decrease)

N	*S*	*D*
6 levels: 300, 3M, 25M, 85M, 300M, 700M	early stopping	8 levels: 20M, 40M, 80M, 200M, 300M, 700M, 1.4B, 23B

The paper eventually has **< 6*8 = 48 runs in total**



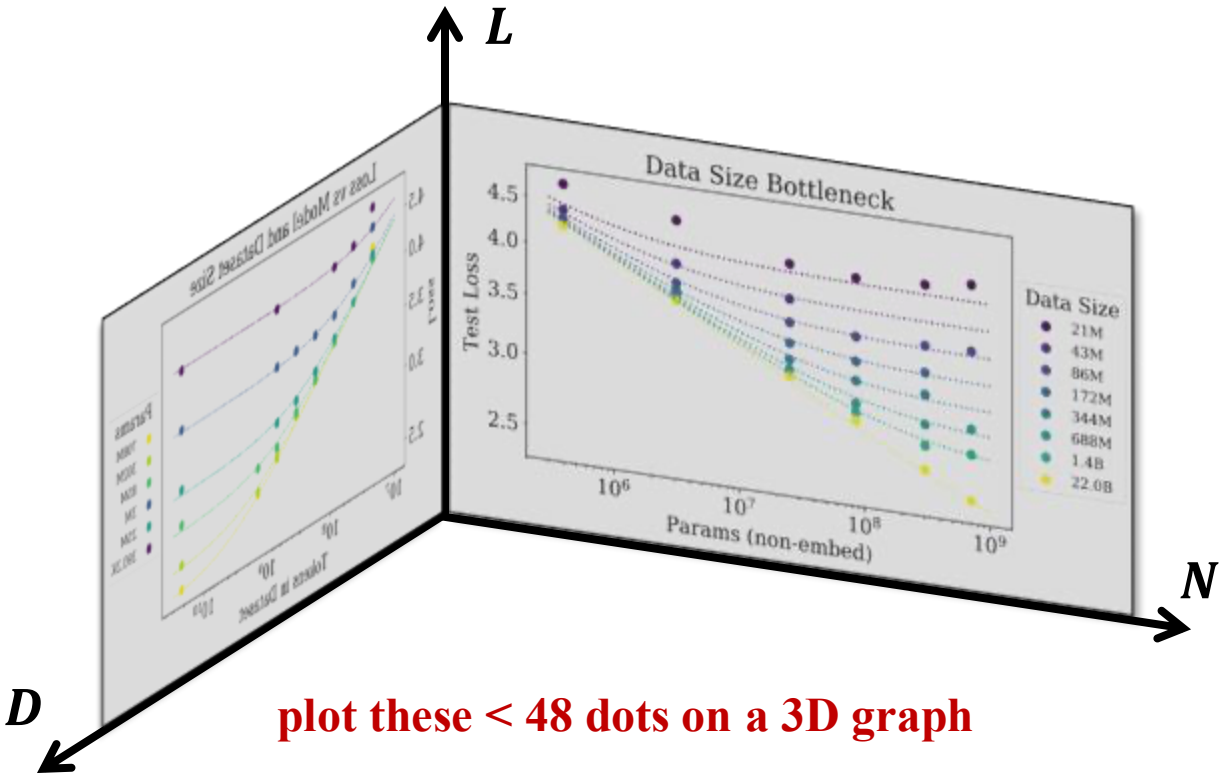
[Figure 9, Rosenfeld et al, 2019]

- Get the (approximately) **best possible test loss** for each *N* and *D*

$L(N, D)$

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

- The lowest test loss ***L*** achievable by a model of a finite size ***N*** on a finite dataset ***D***



plot these < 48 dots on a 3D graph

choose a two-variable function form & fit the function

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

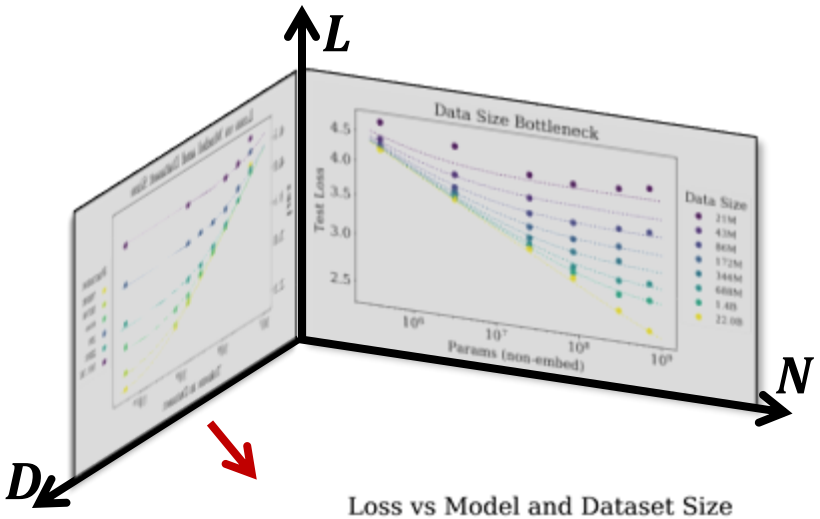
Parameter	α_N	α_D	N_c	D_c
Value	0.076	0.103	6.4×10^{13}	1.8×10^{13}

Table 2 Fits to $L(N, D)$

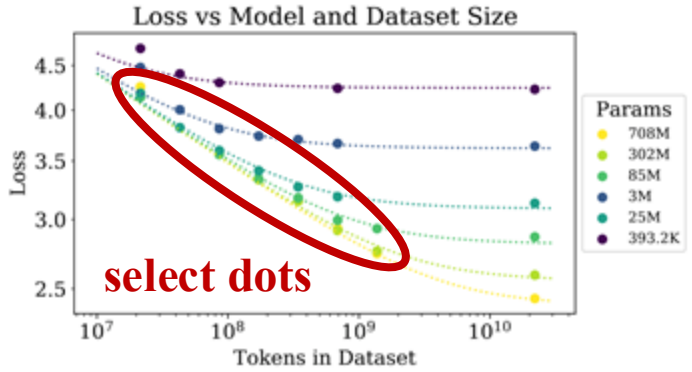
Data−constrained scaling law: $L(D)$

∞	D	Early Stop	Fixed	$L(D) = (D_c/D)^{\alpha_D}$

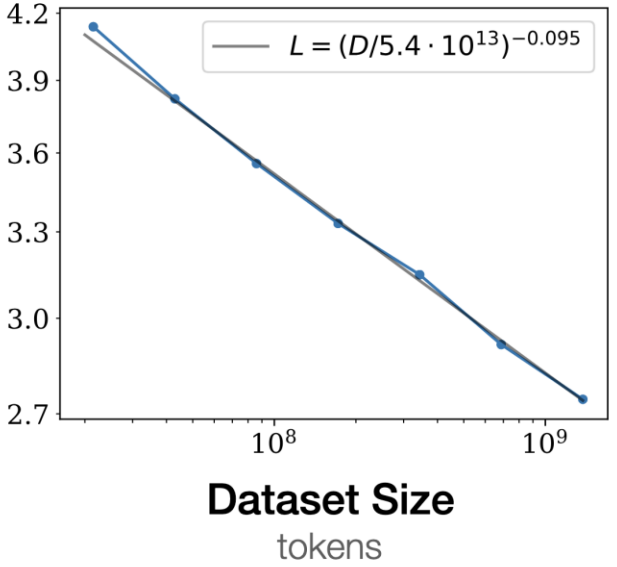
- The lowest test loss ***L*** achievable by a large model on a finite dataset ***D***



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



fit power law



Summary

- 2 sets experiments (~39+48 runs), 5 scaling laws

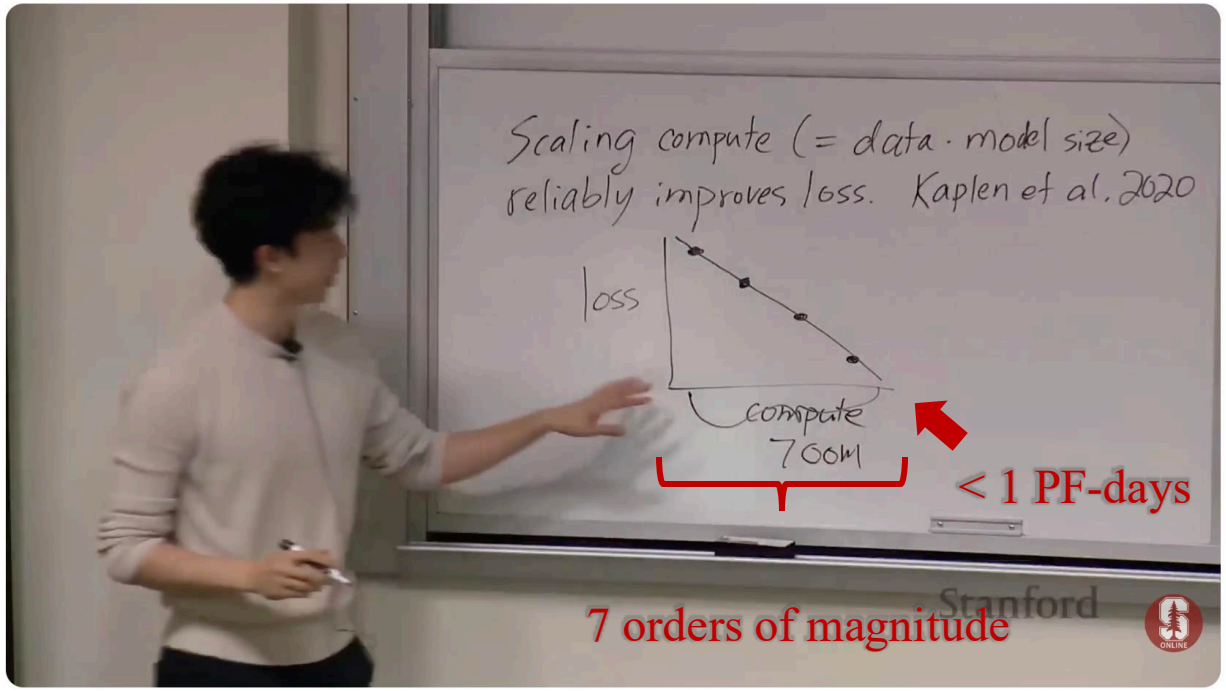
For easier reference, we provide a summary below of the key trends described throughout the paper.

Parameters	Data	Compute	Batch Size	Equation
N	∞	∞	Fixed	$L(N) = (N_c/N)^{\alpha_N}$
∞	D	Early Stop	Fixed	$L(D) = (D_c/D)^{\alpha_D}$
Optimal	∞	C	Fixed	$L(C) = (C_c/C)^{\alpha_C}$ (naive)
N_{opt}	D_{opt}	$C_{\min}$	$B \ll B_{\text{crit}}$	$L(C_{\min}) = \left(C_c^{\min}/C_{\min}\right)^{\alpha_C^{\min}}$
N	D	Early Stop	Fixed	$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$
N	∞	S steps	B	$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}(S,B)}\right)^{\alpha_S}$

Table 4

Scaling laws for trend prediction

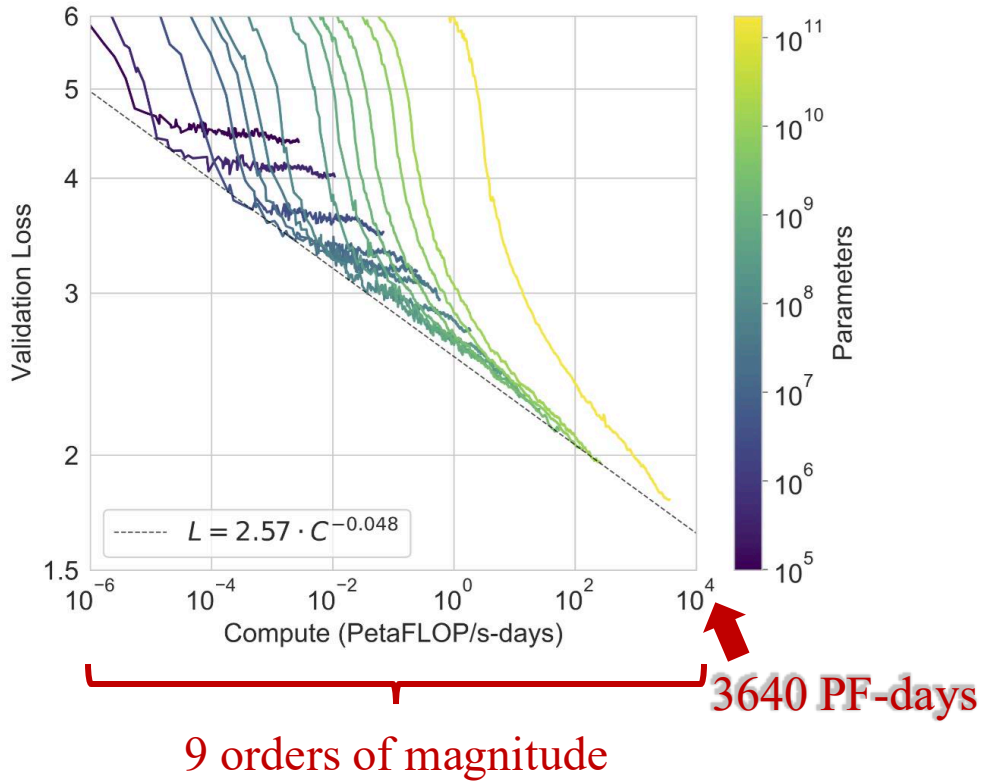
- "*If you have a **large dataset** and you train a very **big neural network**, then success is guaranteed!*" - Ilya Sutskever



Stanford CS25: V4 I Jason Wei & Hyung Won Chung of OpenAI

[https://www.youtube.com/watch?v=3gb-ZkVRemQ]

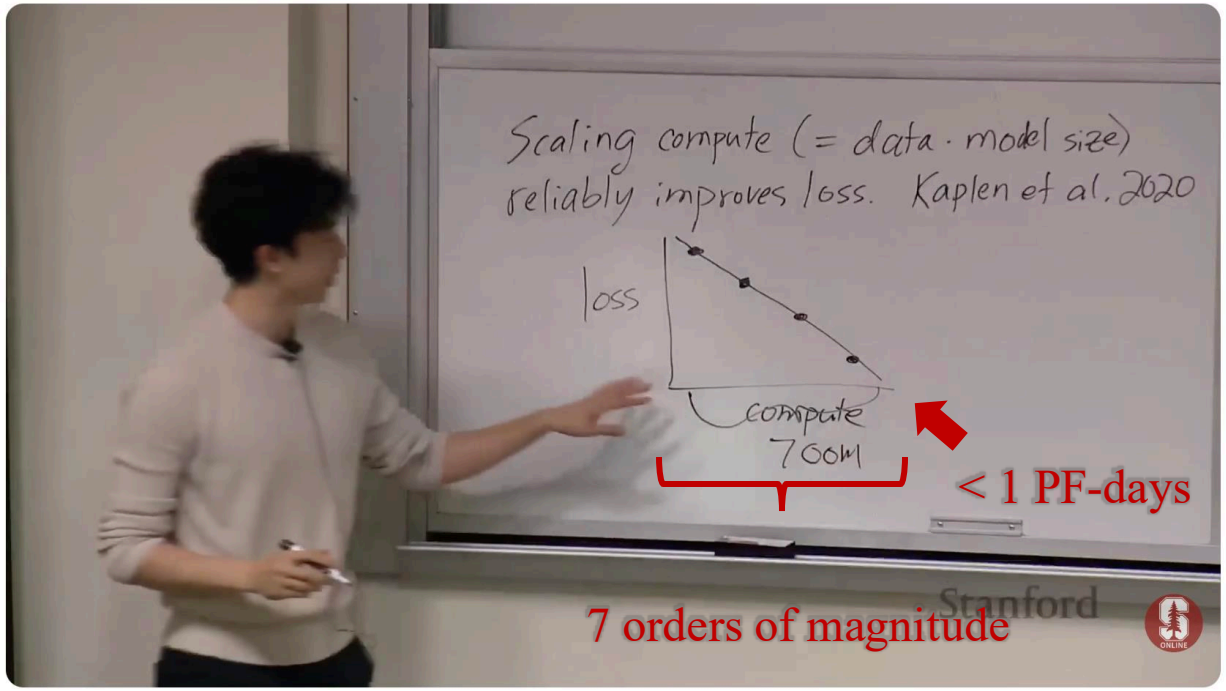
[Figure 3.1, GPT-3, May, 2020]



Scaling laws for trend prediction

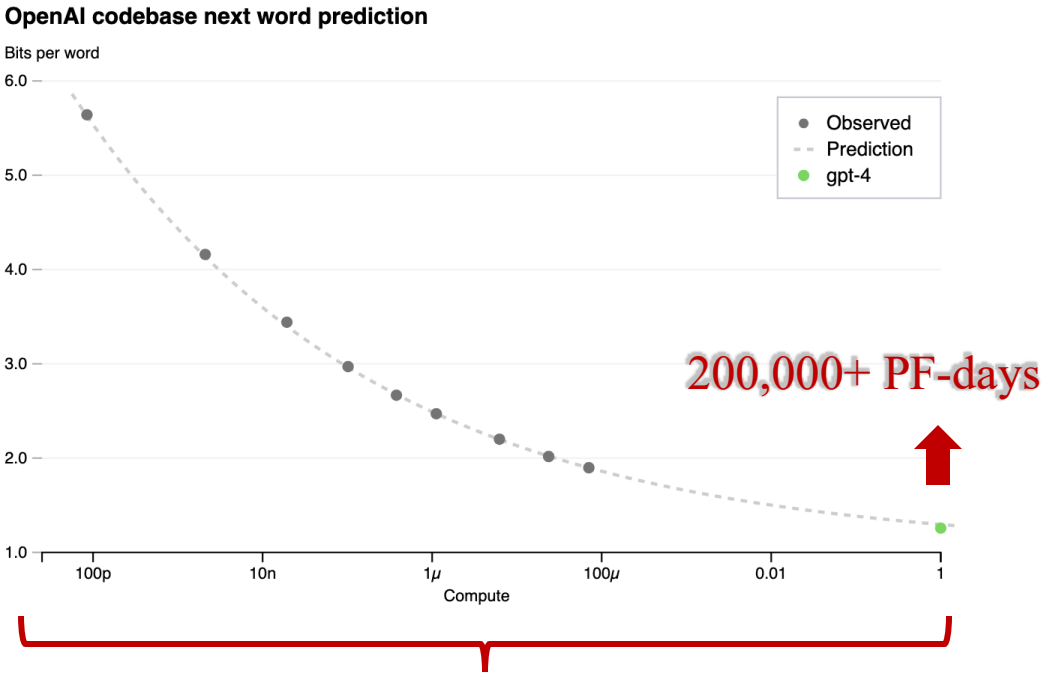
- "*If you have a **large dataset** and you train a very **big neural network**, then success is guaranteed!*" - Ilya Sutskever

[Figure 1, GPT-4, March, 2023]



Stanford CS25: V4 I Jason Wei & Hyung Won Chung of OpenAI

[https://www.youtube.com/watch?v=3gb-ZkVRemQ]



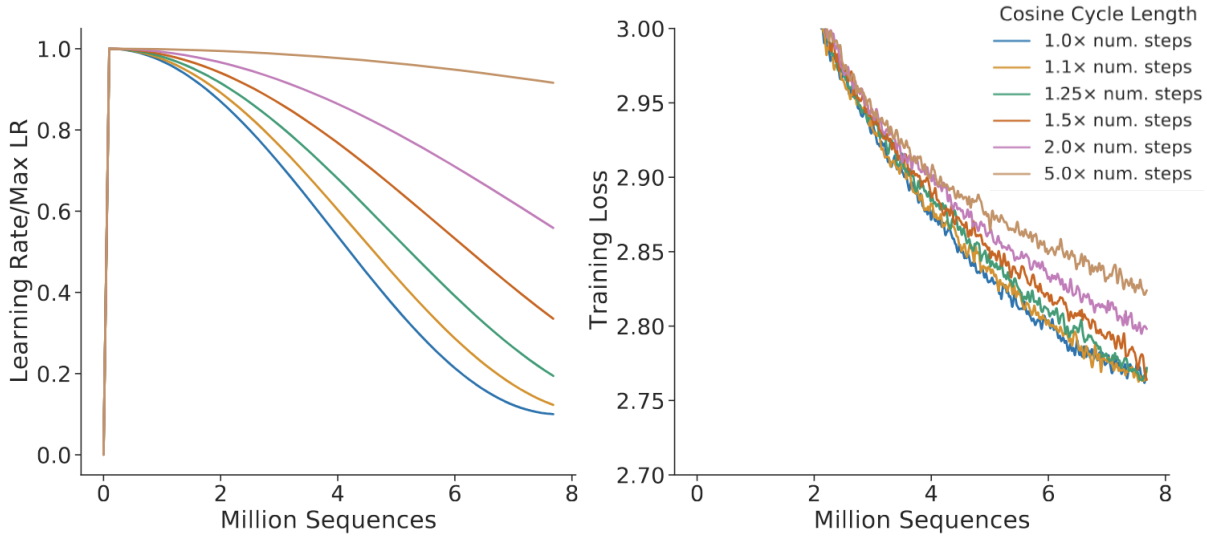
Summary

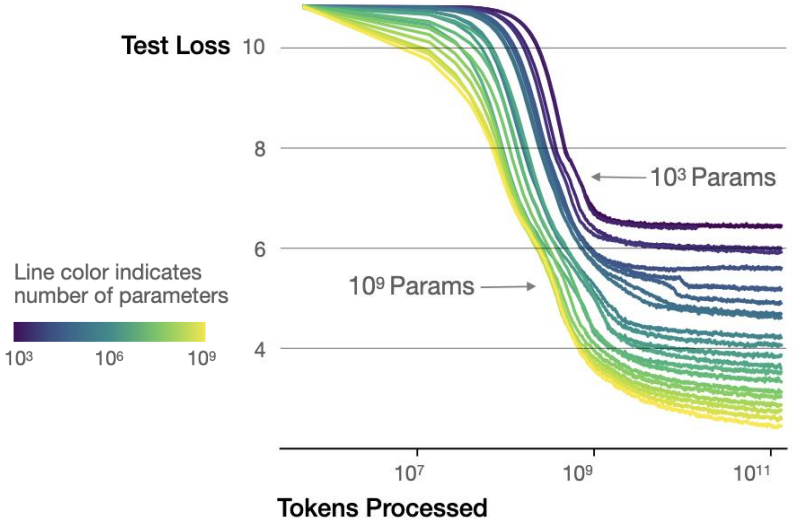
- Fixed **batch size** and **LR schedule cycle length** for all runs with **epoch** larger than one

not optimal for most runs

not optimal for intermediate steps

the loss decay might slow down after 1 epoch





[Figure A1, Hoffman et al, 2022]

The scaling laws might not be accurate!

Modifications in compute-optimal scenario:

Hoffman et al, 2022 and more

Hoffman et al, 2022, “Chinchilla Scaling Law”

- New recipe for **compute-optimal allocation**

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

We investigate the optimal model size and number of tokens for training a transformer language model 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 facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

$$N^*(C_{\min}) \propto C^{0.73}, \qquad (BS)^*(C_{\min}) \propto C^{0.27}$$

[Kaplan et al, 2022]

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

Compute-optimal model size: $N^*(C)$

- Goal: Fit $N^*(C)$
- Focus on the scenario where epoch ≤ 1, thus $\boldsymbol{D = BS}$ and measure the training loss
  - When epoch ≤ 1, for relatively large dataset, it is reasonable to regard training loss and test loss as the same
- Tune LR schedule cycle lengths to match S

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

We will look at these results starting from the **experiments**

Summary of experiments: Set 1

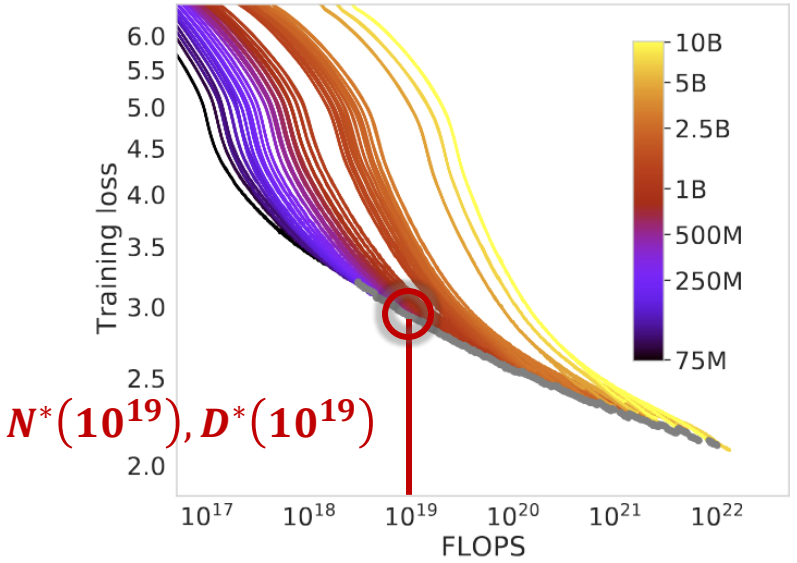
- Train models of different **sizes *N*** with different numbers of **steps *S***

N	*S*	*B*	LR Schedule
73M	4 levels, ranging by a factor of 16 ×	tuned	4 levels, setting cosine cycle length equal to *S*
…	…	…	…
10B	4 levels, ranging by a factor of 16 ×	tuned	4 levels, setting cosine cycle length equal to *S*

~ 40 different models, ~ 40*4=160 runs in total

Approach 1: Minimum over training curves

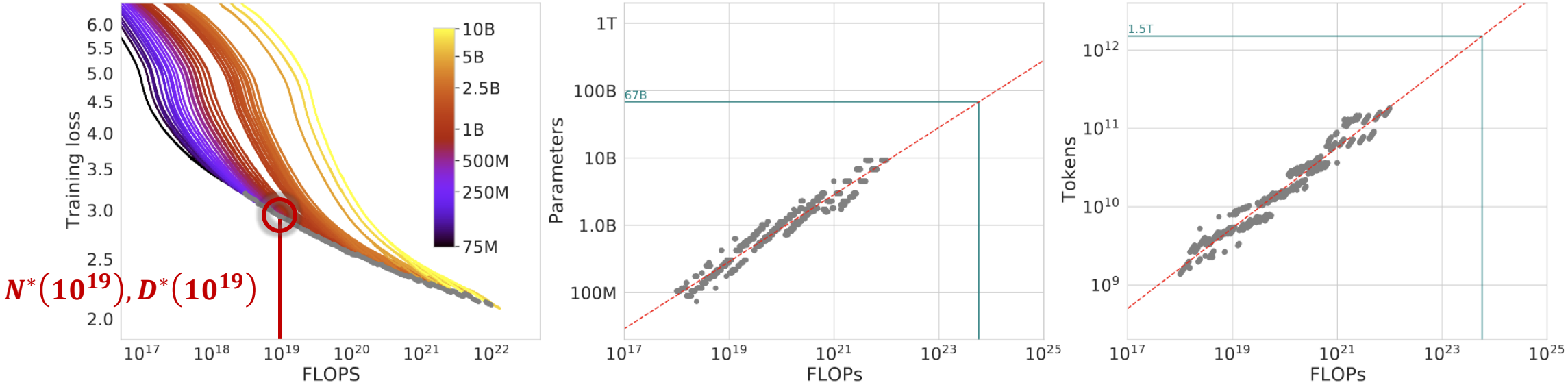
- **Training curve**: for each run, smooth and interpolate the training loss curve
- **Envelope**: for each FLOP count C, determine which run achieves the lowest loss; such run represents the compute-optimal $N^*(C)$ and $D^*(C)$



$N_{opt} \propto C^a$ and $D_{opt} \propto C^b$ $\quad a = 0.50$ and $b = 0.50$

Approach 1: Minimum over training curves

- **Training curve**: for each run, smooth and interpolate the training loss curve
- **Envelope**: for each FLOP count C, determine which run achieves the lowest loss; such run represents the compute-optimal $N^*(C)$ and $D^*(C)$



$$N_{opt} \propto C^a \text{ and } D_{opt} \propto C^b \qquad a = 0.50 \text{ and } b = 0.50$$

Summary of experiments: Set 2

- Under a fixed **compute C**, vary the model **sizes N**

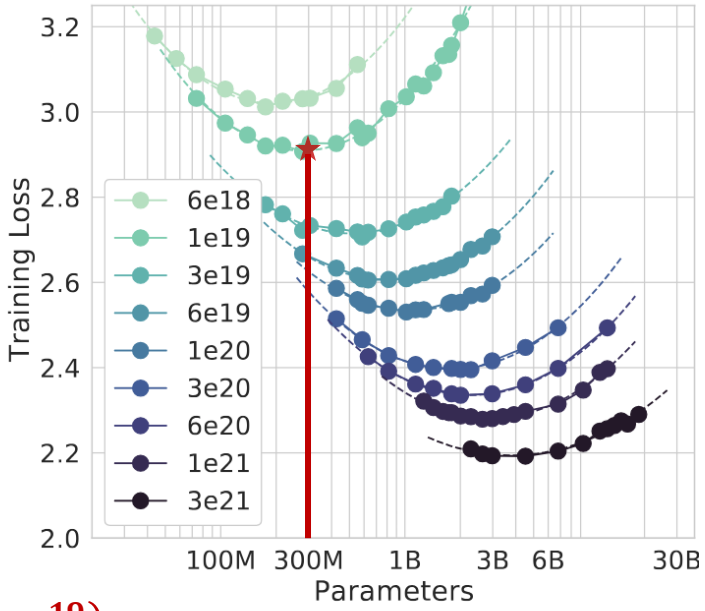
9 levels of compute

C	N	$S = C/6NB$	B	LR Schedule
6×10^{18}	11 levels, in 70M-16B	determined by $C/6NB$	tuned	setting cosine cycle length equal to S
…	…	…	…	…
3×10^{21}	12 levels, in 70M-16B	determined by $C/6NB$	tuned	setting cosine cycle length equal to S

~130 runs in total

Approach 2: IsoFLOP profiles

- **IsoFLOP curve**: for each FLOP count C, fit a parabola curve with N as the x-axis
- **Vertex**: for each parabola, determine the minimum point $N^*(C)$

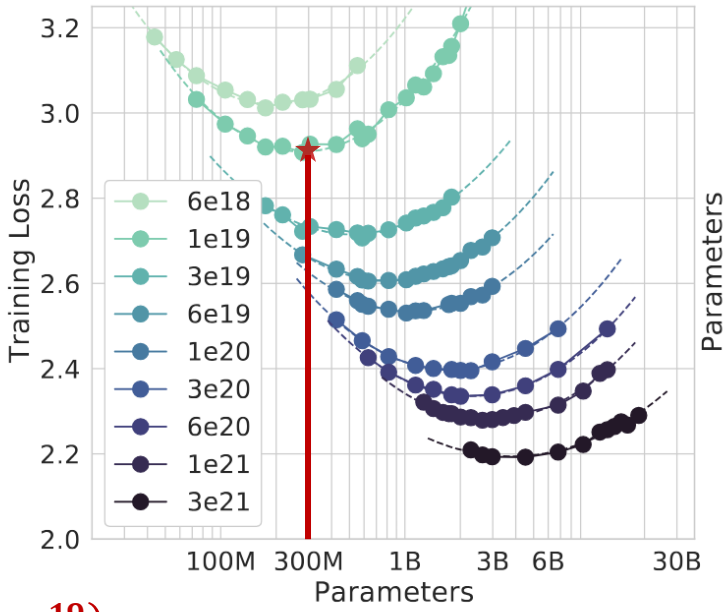


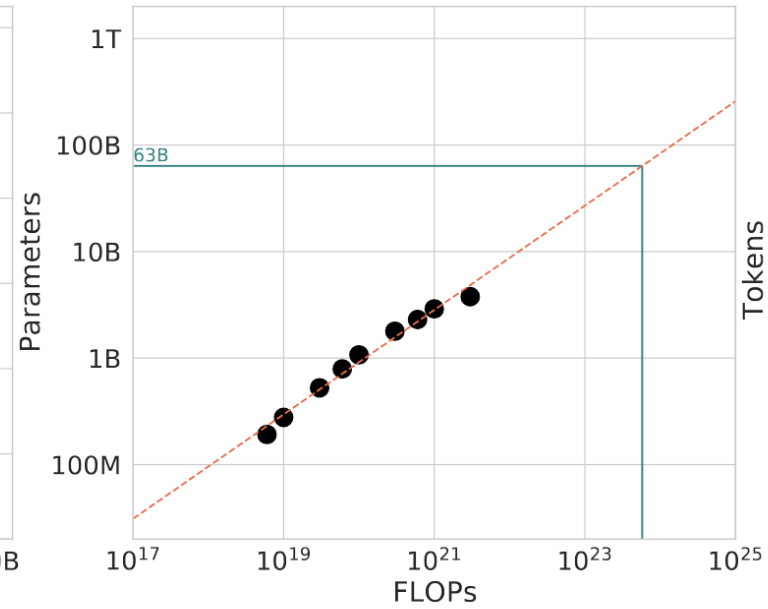
$N^*(10^{19}) = 300\text{M}$

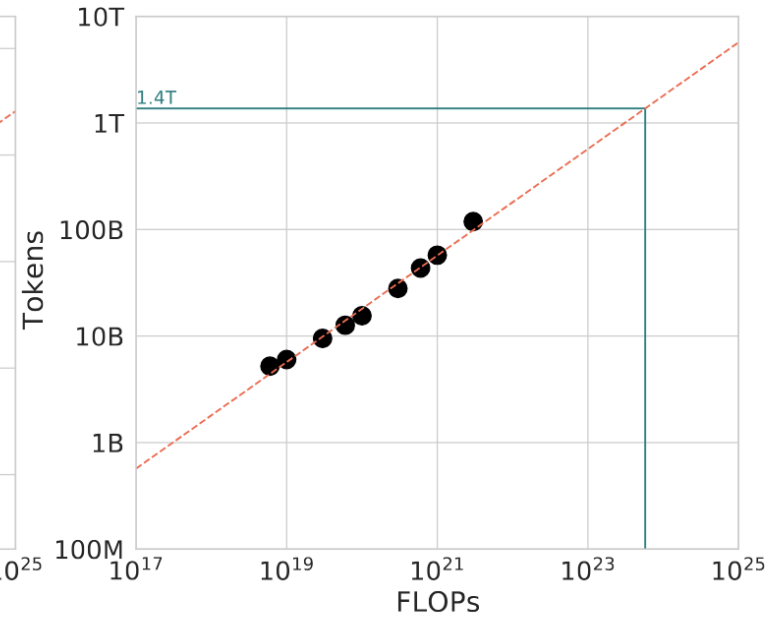
$$N_{opt} \propto C^a \text{ and } D_{opt} \propto C^b \quad a = 0.49 \text{ and } b = 0.51$$

Approach 2: IsoFLOP profiles

- **IsoFLOP curve**: for each FLOP count C, fit a parabola curve with N as the x-axis
- **Vertex**: for each parabola, determine the minimum point $N^*(C)$







$N^*(10^{19}) = 300\text{M}$

$$N_{opt} \propto C^a \text{ and } D_{opt} \propto C^b \quad a = 0.49 \text{ and } b = 0.51$$

Approach 3: Parametric modelling of the loss

- Function form:

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

- Put together all results (final training loss, ~ 290 dots in total) in Set 1 and Set 2 to fit this two-variable function:

$$L(N, D) = E + \frac{A}{N^{0.34}} + \frac{B}{D^{0.28}}, \quad E = 1.69, A = 406.4, B = 410.7$$

- Derived optimal allocation under $C = 6ND$:

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

$N_{opt} \propto C^{a}$ and $D_{opt} \propto C^{b}$ $\quad a = 0.46$ and $b = 0.54$

Approach 3: Parametric modelling of the loss

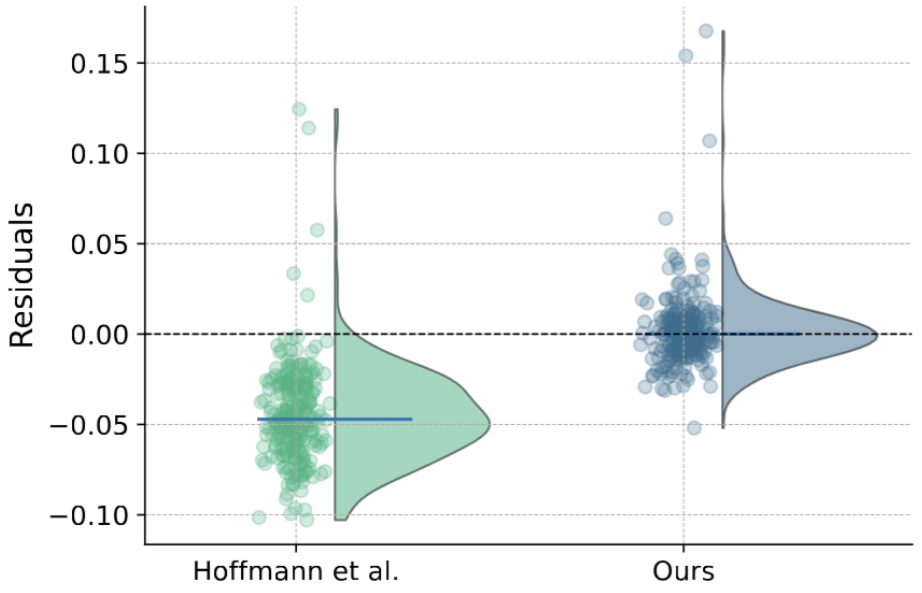


Figure 3: Plot of residuals of Hoffmann et al.'s estimated model with rounded parameter values and our estimated model.)

Our estimated model

$$L(N, D) = 1.8172 + \frac{482.01}{N^{0.3478}} + \frac{2085.43}{D^{0.3658}} \quad (3)$$

Hoffmann et al.'s estimated model

$$L(N, D) = 1.6934 + \frac{406.4}{N^{0.3392}} + \frac{410.7}{D^{0.2849}} \quad (4)$$

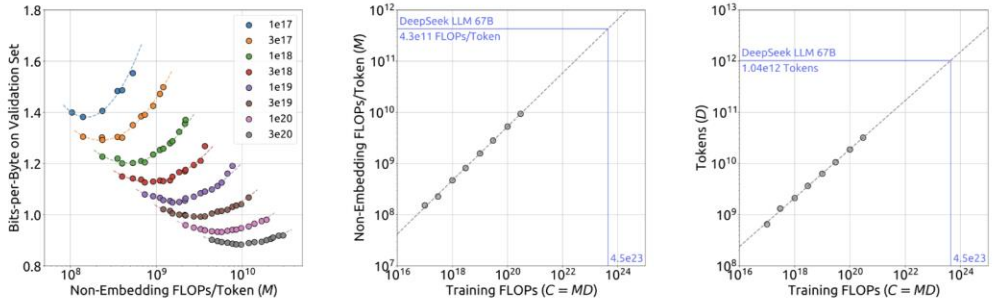
Parameter	Our estimate	Hoffmann et al's estimate
A	482.01 (124.58)	406.4
B	2085.43 (1293.23)	410.7
E	1.8172 (0.03)	1.6934
α	0.3478 (0.02)	0.3392
β	0.3658 (0.02)	0.2849
$a = \beta/(\alpha+\beta)$	0.5126 (0.02)	0.454
Data points	240	> 400

Table 1: Our parameter estimates and their standard errors. The standard errors are shown in parentheses and are obtained by bootstrapping. We show the estimate from Hoffmann et al. along with our estimates for comparison.[1]

[Epoch AI, Chinchilla Scaling: A replication attempt, 2024]

Near-proportional scaling of model and dataset size—quite stable!

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27



(a) IsoFLOP curve (b) Optimal model scaling (c) Optimal data scaling

Figure 4 | IsoFLOP curve and optimal model/data allocation. The metric in IsoFLOP curve is bits-per-byte on the validation set. The dotted lines in optimal model/data scaling curves represent the power law fitting the smaller model (grey circles).

$$M_{\text{opt}} = M_{\text{base}} \cdot C^a, \quad M_{\text{base}} = 0.1715, \quad a = 0.5243$$
$$D_{\text{opt}} = D_{\text{base}} \cdot C^b, \quad D_{\text{base}} = 5.8316, \quad b = 0.4757$$

[Deepseek LLM, 2024]

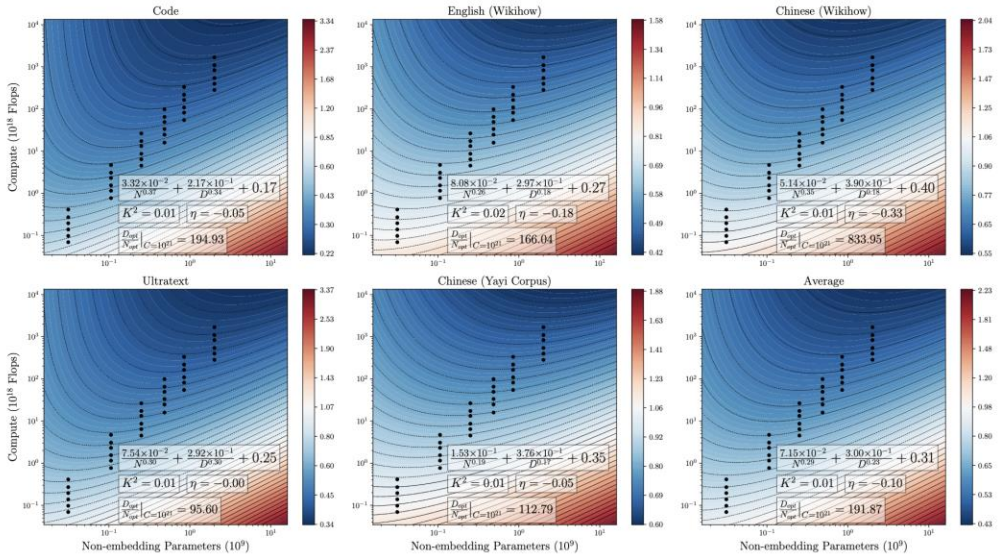


Figure 10: The fit result of the scaling experiment with WSD Scheduler. The black dots in a horizontal line denote the decayed checkpoints in different compute within the same model size.

$$a = 0.45, b = 0.55$$

[MiniCPM, 2024]

Tokens / param—not that stable!

Table 3 | **Estimated optimal training FLOPs and training tokens for various model sizes.** For various model sizes, we show the projections from Approach 1 of how many FLOPs and training tokens would be needed to train compute-optimal models. The estimates for Approach 2 & 3 are similar (shown in Section D.3)

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

~ 20 tokens / param

As for the concrete data-to-model ratio $\frac{D_{opt}}{N_{opt}}$, we notice that there is a huge gap in compute optimal regime between ours and Hoffmann et al. (2022) despite that the trend of $\frac{D_{opt}}{N_{opt}}$, with compute C is aligned between ours and theirs. Specifically, the data size should be 192 times larger than the model size on average, as opposed to 20 times in Hoffmann et al. (2022). We note that this aligns with the observation in Section 4.3 and Figure 6.

MiniCPM, 2024: ~ 192 tokens / param

We use the compute-optimal models we identified this way to predict the optimal number of training tokens for a specific compute budget. To do so, we assume a power-law relation between compute budget, C, and the optimal number of training tokens, $N^\star(C)$:

$$N^\star(C) = AC^\alpha.$$

We fit A and α using the data from Figure 2. We find that $(\alpha, A) = (0.53, 0.29)$; the corresponding fit is shown in Figure 3. Extrapolation of the resulting scaling law to 3.8×10^{25} FLOPs suggests training a 402B parameter model on 16.55T tokens.

Llama 3 (405B), 2024: ~ 40 tokens / param

Tokens / param

- If inference compute is taken into account…

Important note – train-optimal may not be what you want

Chinchilla aims to tell you what gives the best model for fixed training compute..
But most of the compute in a real deployment is inference.. So we should 'over' train

- **GPT3** – 2 tokens / param
- **Chinchilla** – 20 tokens / param
- **LLaMA65B** – 22 tokens / param
- **Llama 2 70B** – 29 tokens / param
- **Mistral 7B** – 110 tokens / param
- **Llama 3 70B** – 215 tokens / param

The more usage we expect, the more it becomes worth it to pay the upfront cost

[Stanford CS336, Lecture 9]

Scaling laws in over-trained regime

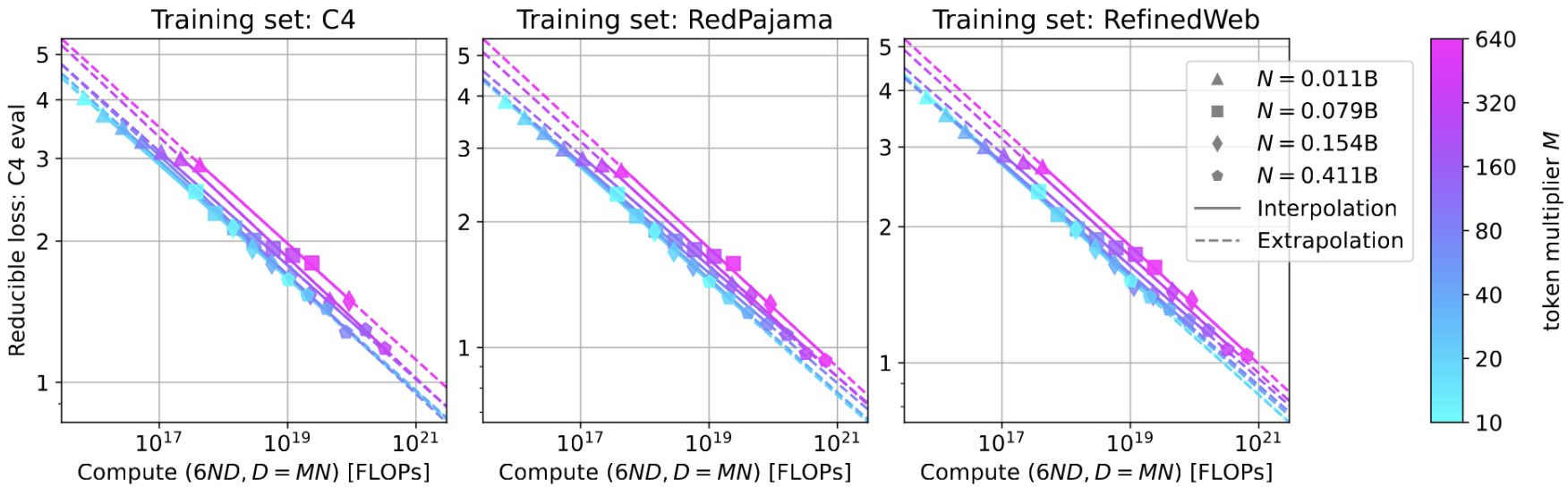
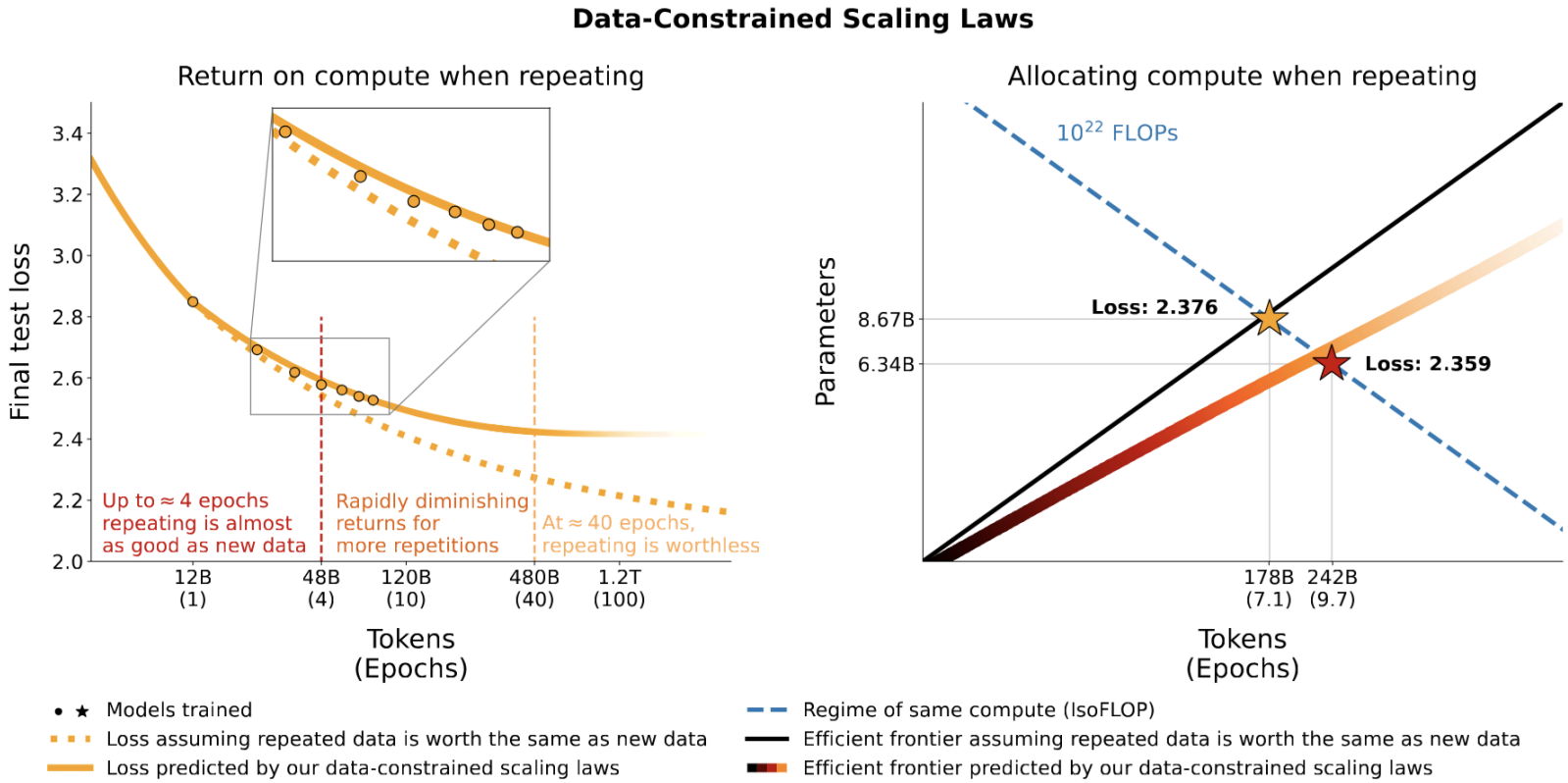


Figure 2: **Scaling in the over-trained regime follows consistent power law exponents.** We notice parallel lines in the log-log plots of reducible loss vs. training compute for a range of token multipliers M, which give the ratio of training tokens to model parameters. Larger M corresponds to more over-training. For a power law giving reducible loss as a function of compute: $L'(C) = \lambda \cdot C^{-\eta}$, the exponent η remains relatively constant resulting in lines with approximately fixed slope (Figure 17). The scalar λ that determines the y-intercept, however, shifts with different token multipliers. This suggests λ is a function of the token multiplier, while η is not.

[Gadre at al, 2024]

Data-constrained scenario



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

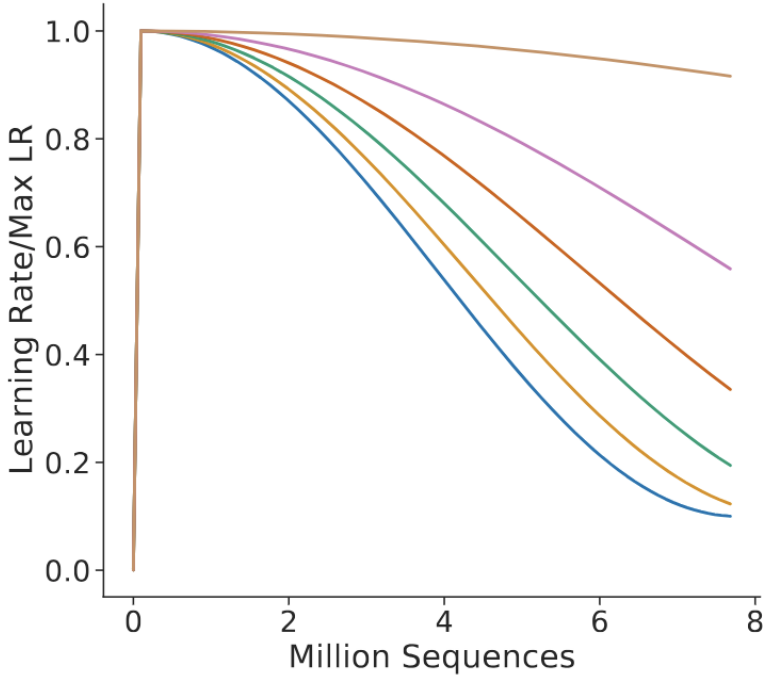
$$D' = U_D + U_D R_D^* (1 - e^{\frac{-R_D}{R_D^*}})$$

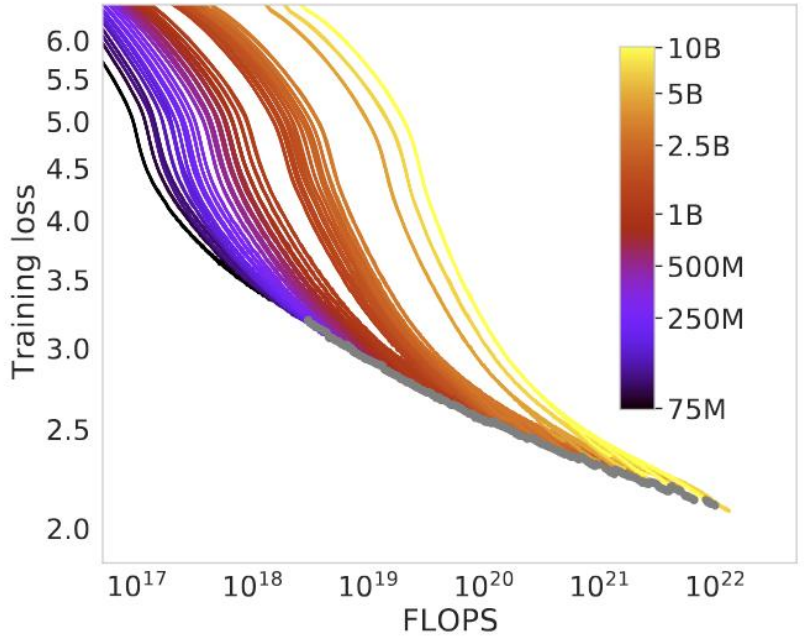
$$N' = U_N + U_N R_N^* (1 - e^{\frac{-R_N}{R_N^*}})$$

[Muennighoff et al, 2023]

Limitations of cosine schedule

- Not globally optimal
- P model sizes and Q steps need $\boldsymbol{P \times Q}$ runs in total





Warmup-Stable-Decay (WSD) schedule

- Mostly optimal
- P model sizes and Q steps need only ~ $\boldsymbol{P}$ runs in total

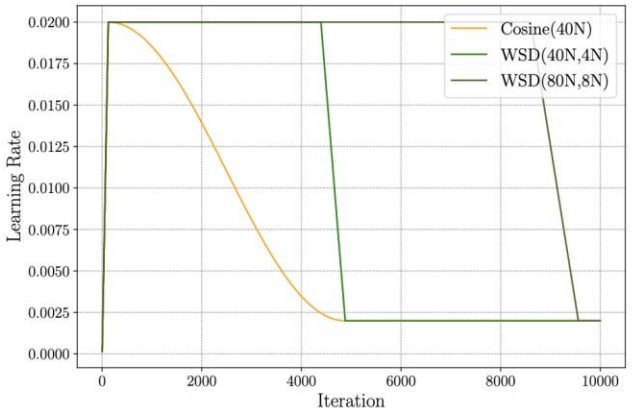
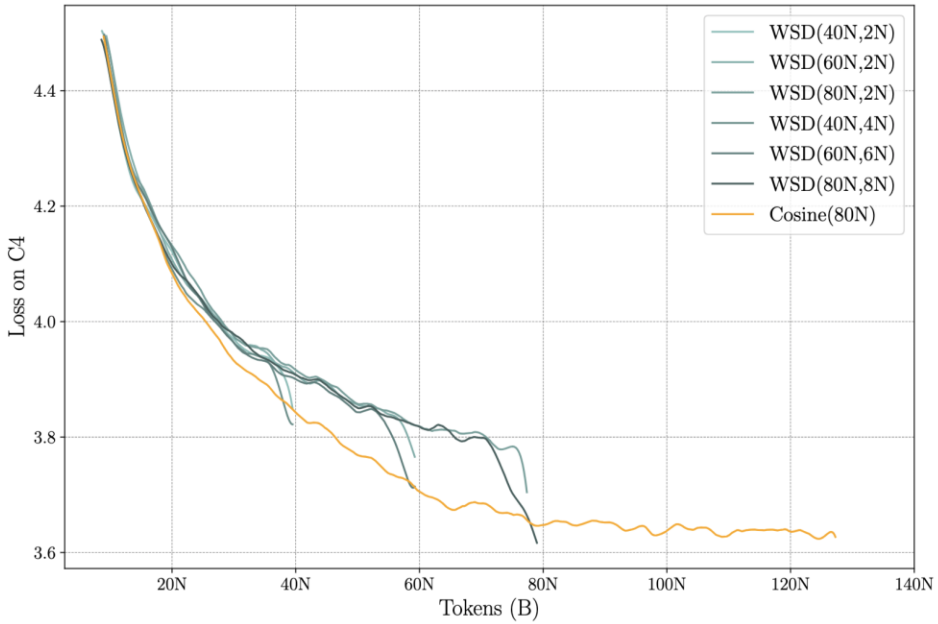


Figure 15: Illustrative comparison between Cosine LRS and WSD LRS. The WSD LRS with different end steps share the same stable training stage.



[MiniCPM, 2024]

Hyperparameter selection

Hyperparameter selection

- Architecture
- Shape
- Optimizer
- Learning rate
- Batch size
- …

Architecture

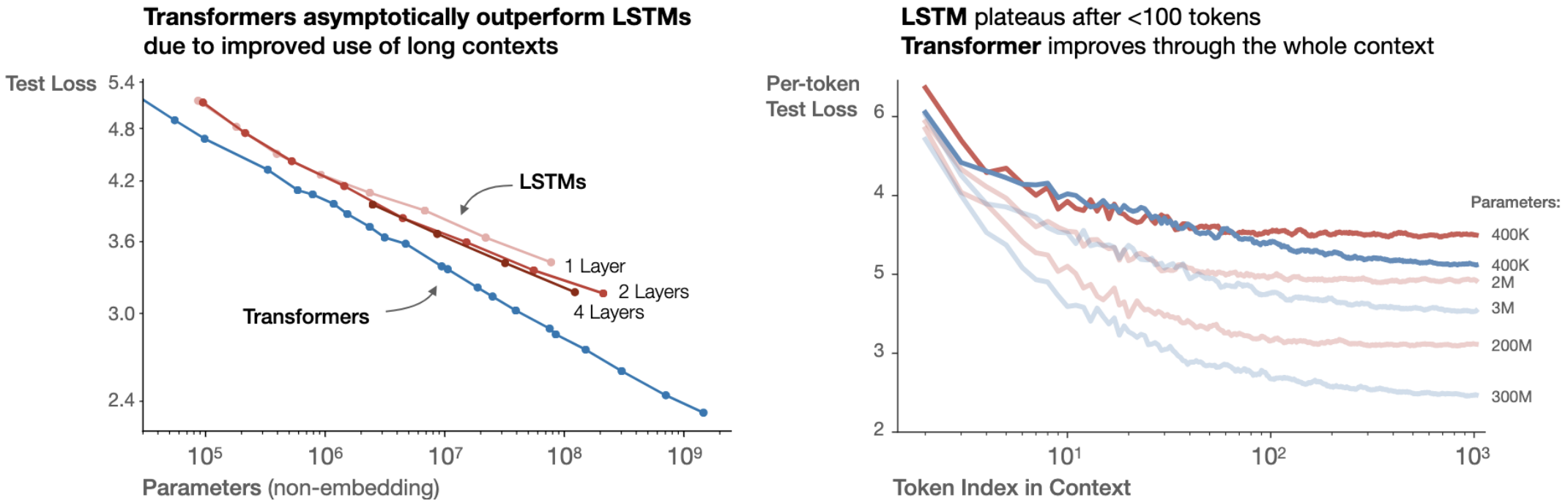


Figure 7

[Kaplan et al, 2020]

Architecture

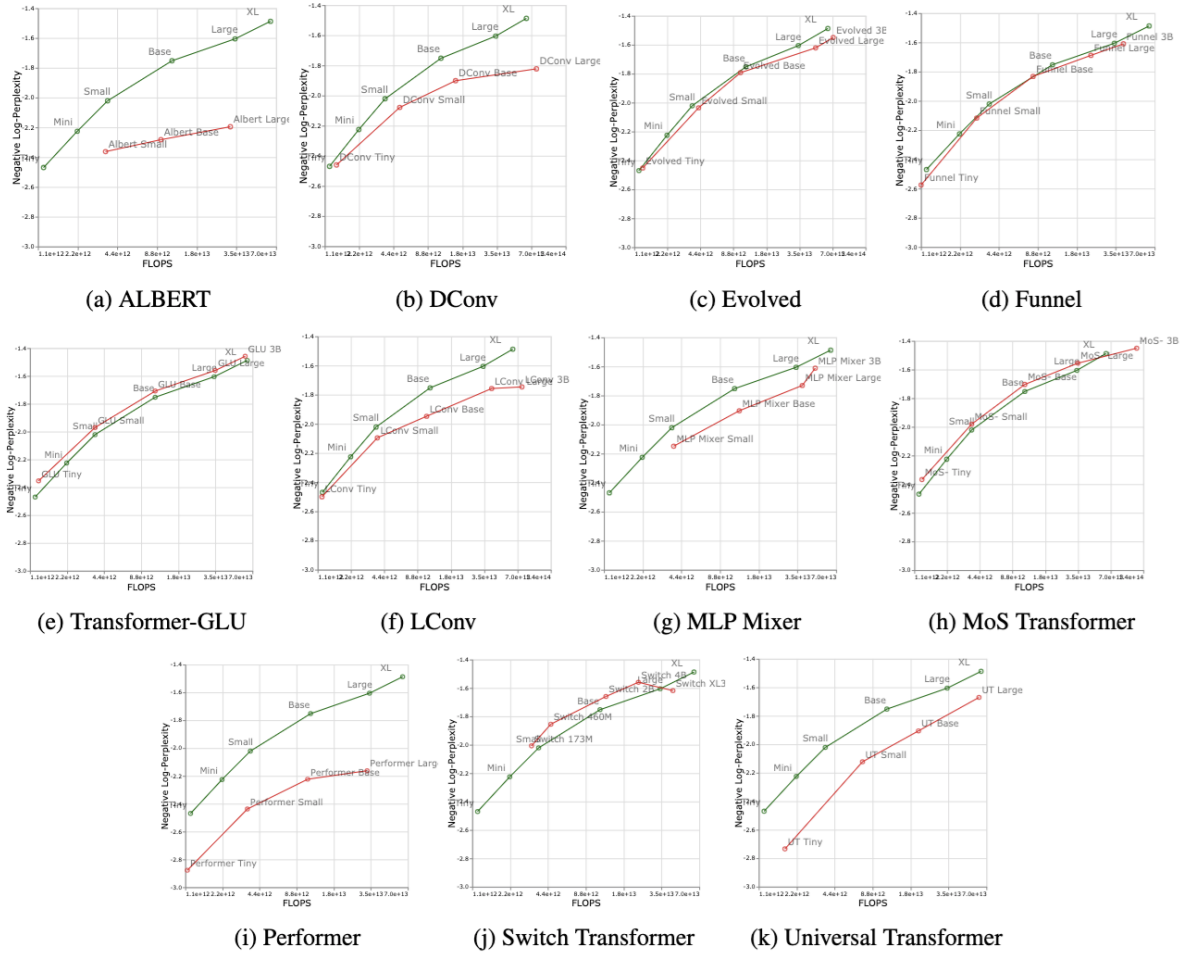


Figure 2: Upstream Negative Log-Perplexity of vanilla Transformer compared to other models.

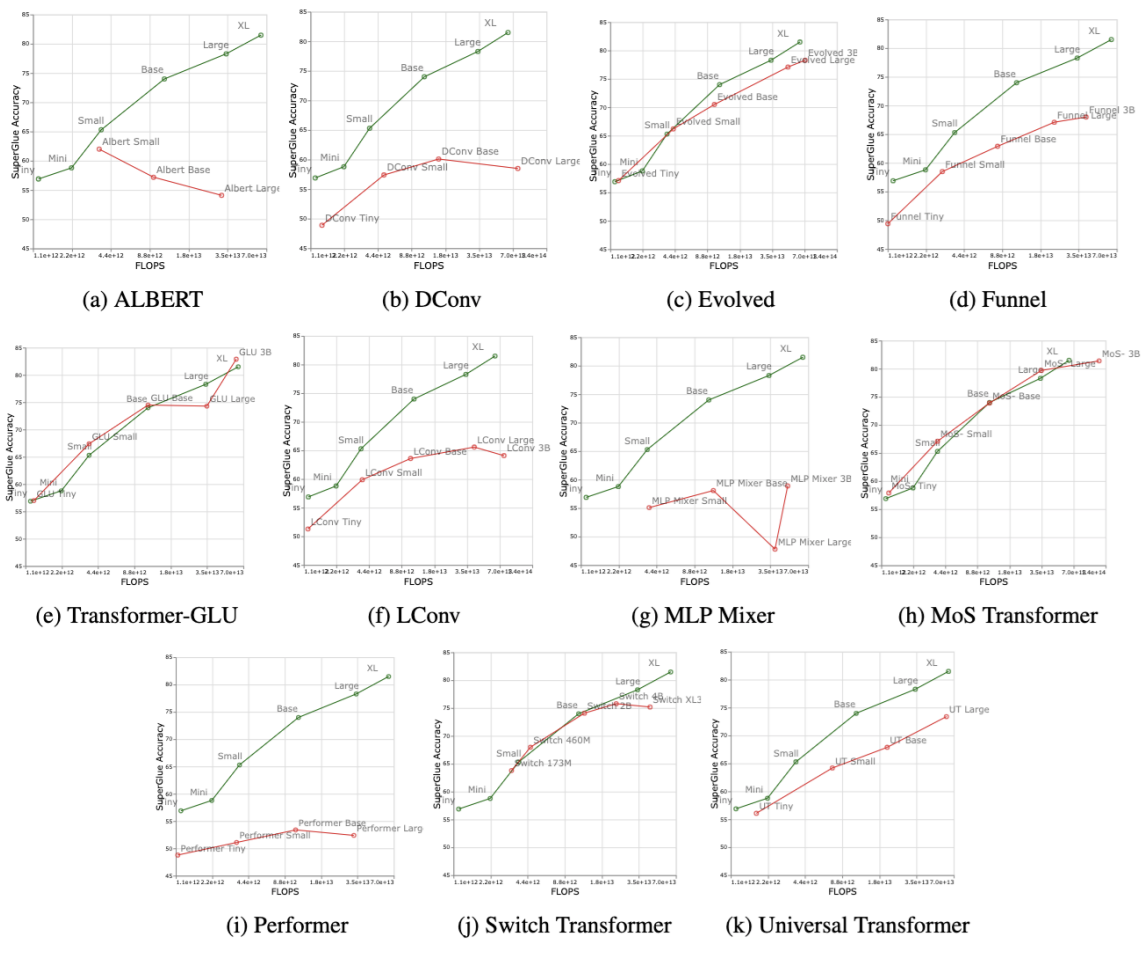
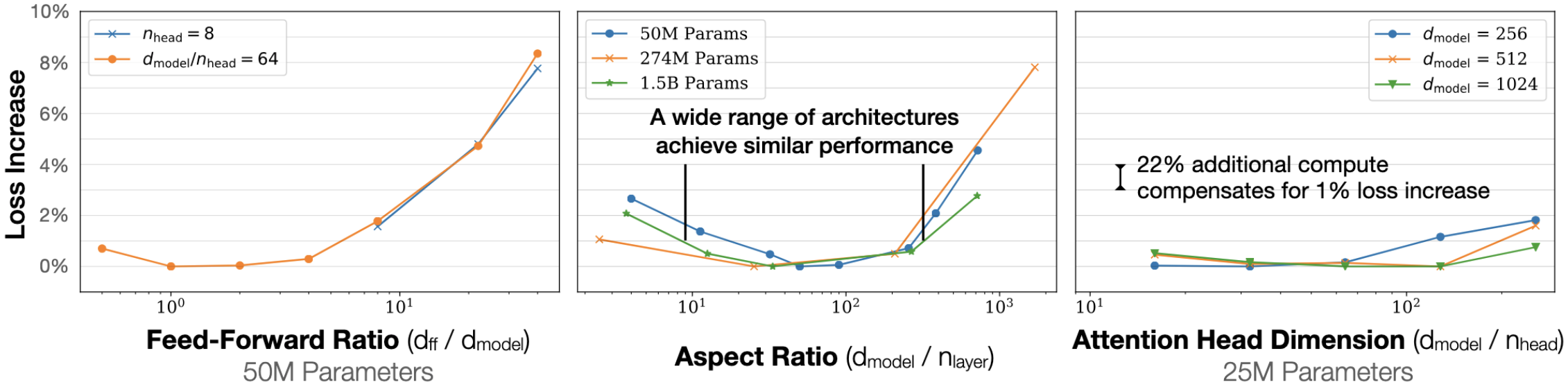


Figure 3: Downstream accuracy of vanilla Transformer compared to other models.

[Tay et al, 2022]

Shape

[Kaplan et al, 2020]

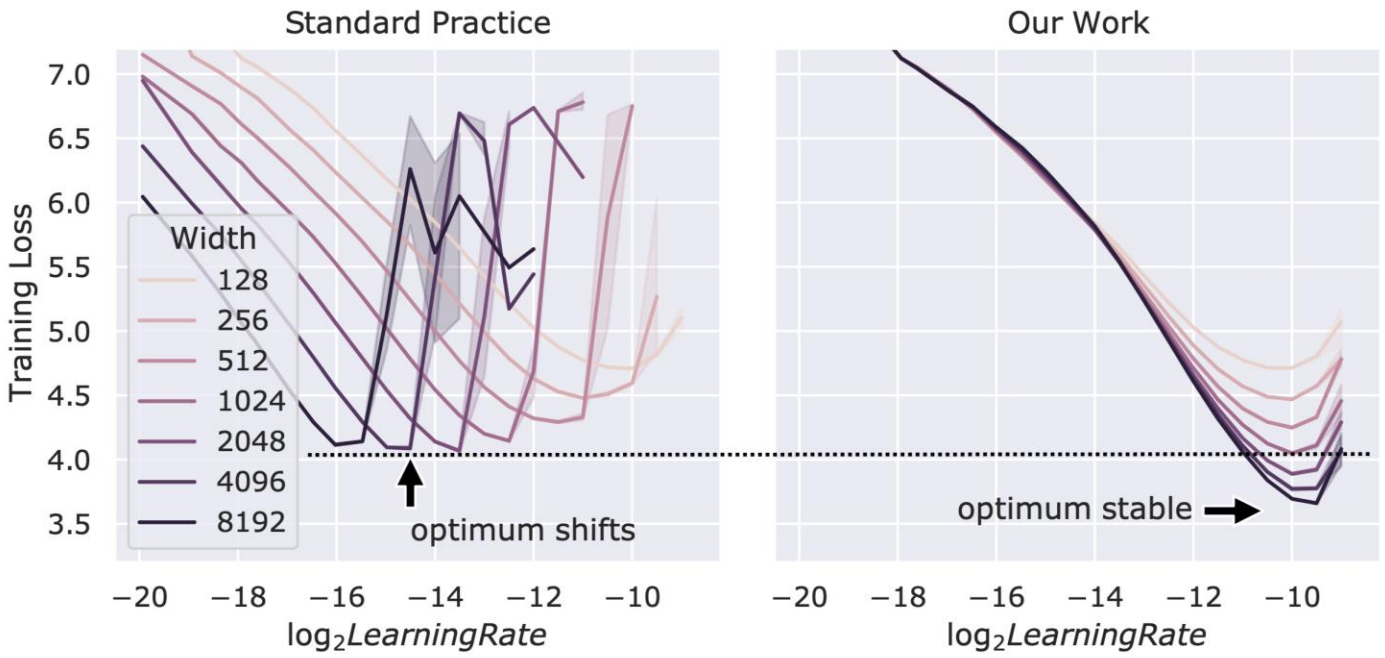


Sweet spot?

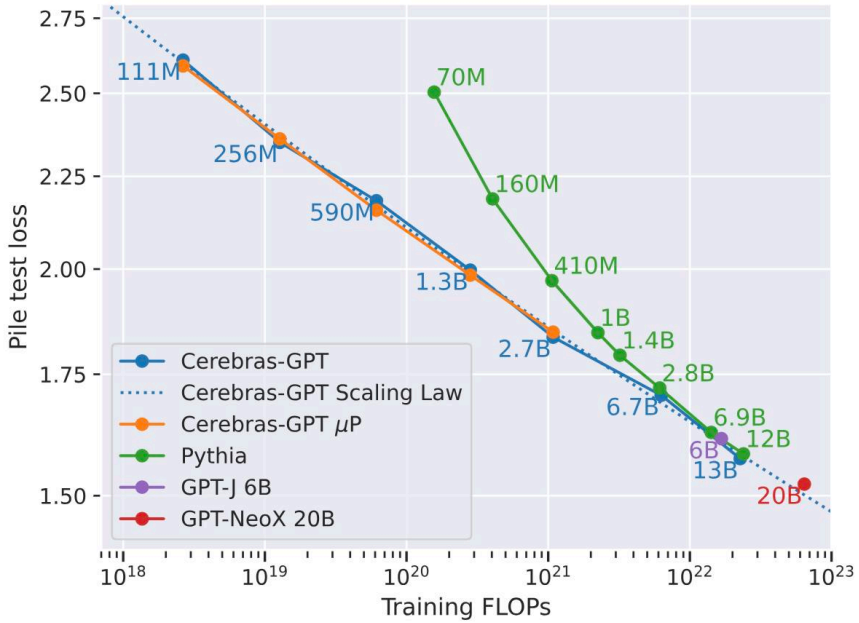
Model	d_{model}/n_{layer}
BLOOM	205
T5 v1.1	171
PaLM (540B)	156
GPT3/OPT/Mistral/Qwen	128
LLaMA / LLaMA2 / Chinchila	102
T5 (11B)	43
GPT2	33

[Stanford CS336, Lecture 3]

Learning rate

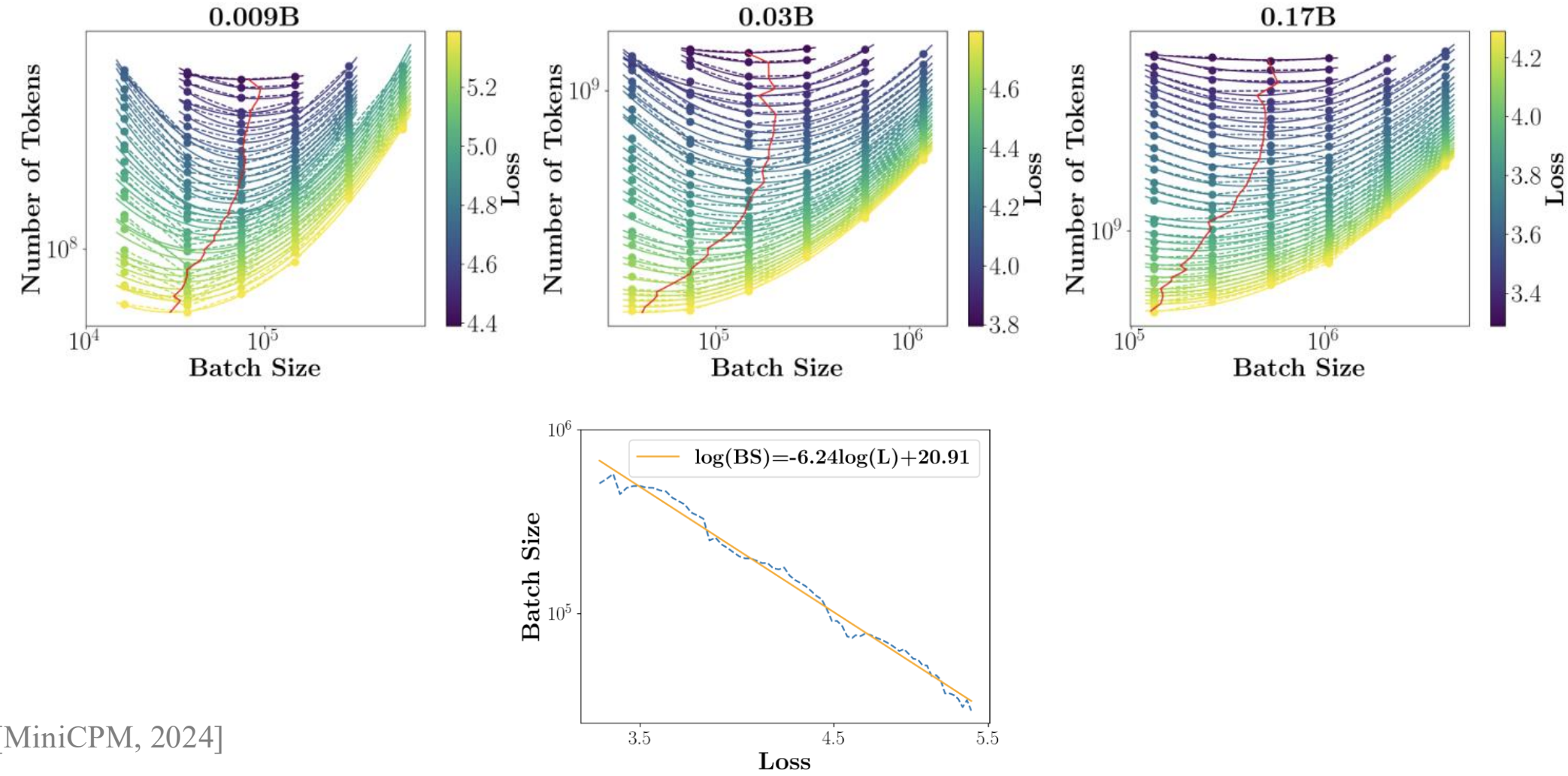


[Tensor Programs V, Yang et al, 2022]



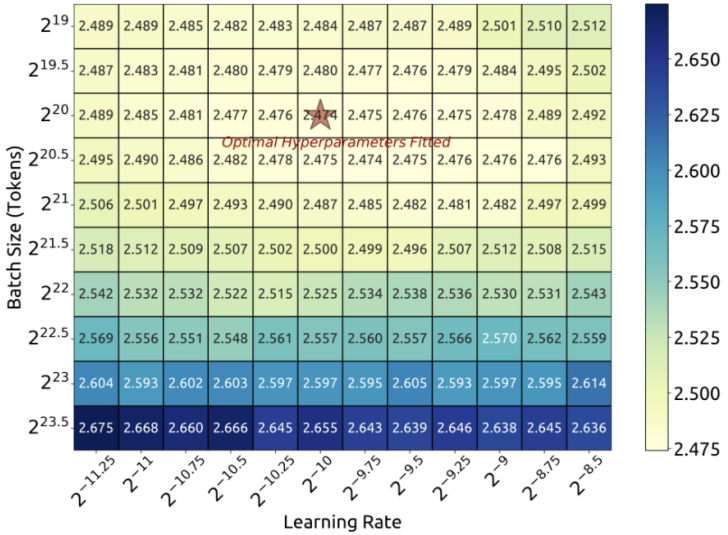
[Cerebras-GPT, Dey et al, 2023]

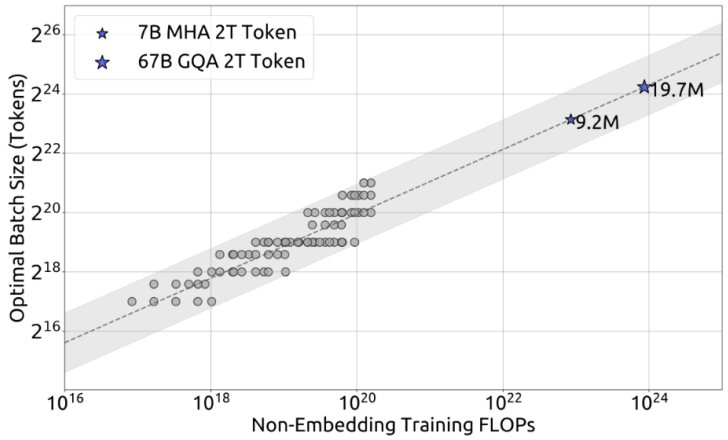
Batch size



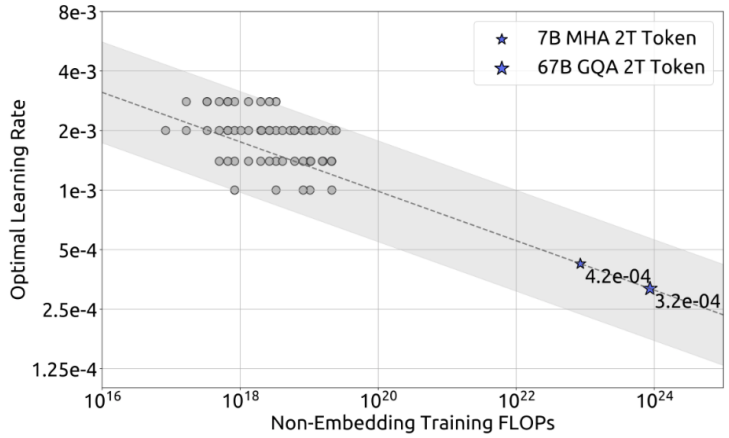
[MiniCPM, 2024]

Batch size





(a) Batch size scaling curve



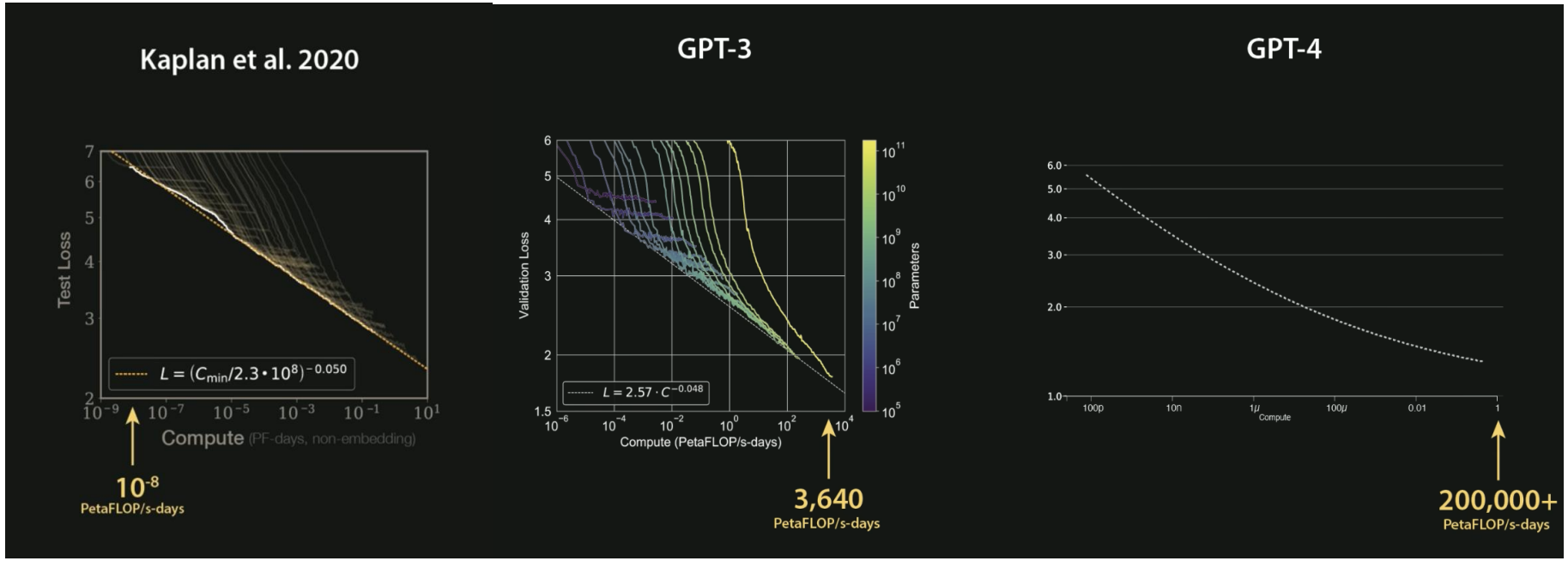
(b) Learning rate scaling curve

[Deepseek LLM, 2024]

Conclusion

Conclusion

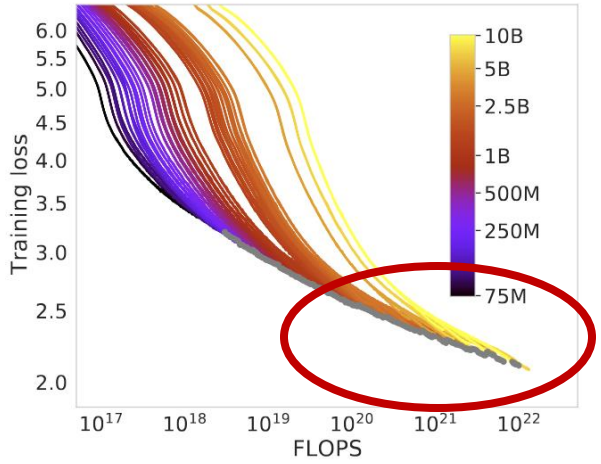
- Scaling law is an effective approach for **predicting trends** and selecting **hyperparameters**.
  - This approach remains valuable as long as we are at the computational frontier.

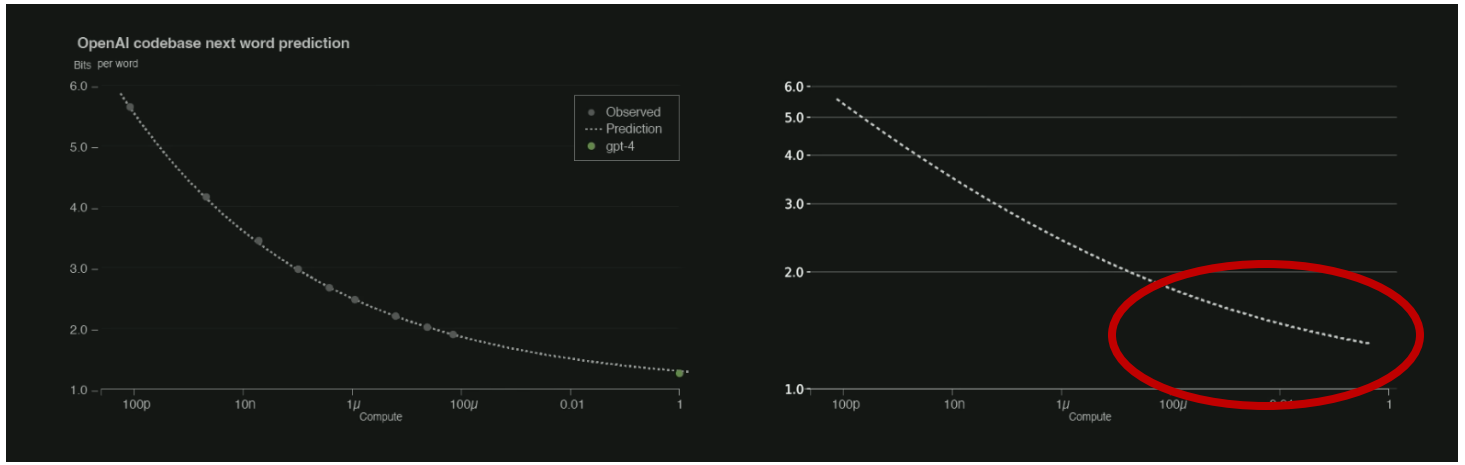


[AI can't cross this line and we don't know why. https://www.youtube.com/watch?v=5eqRuVp65eY&t=609s]

Conclusion

- Scaling law is an effective approach for **predicting trends** and selecting **hyperparameters**.
  - This approach remains valuable as long as we are at the computational frontier.





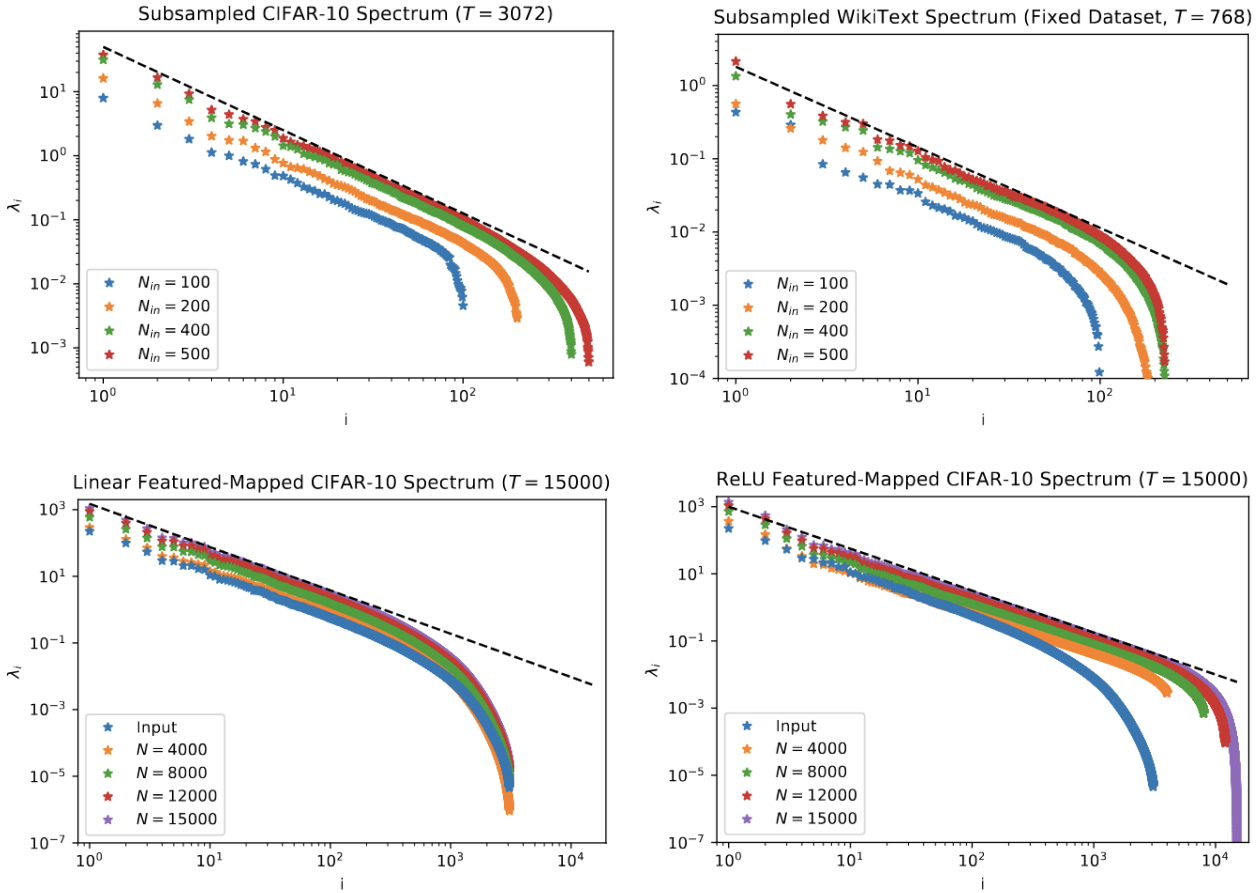
$$L(N, D) = E + \frac{A}{N^{0.34}} + \frac{B}{D^{0.28}},$$

irreducible error

Given finite data and compute, **the scaling law will eventually plateau**,
… but improving it to reduce cost is still important,
… and understanding what the final point will look like is important.

Conclusion

- Scaling law is a well-established phenomenon that theory needs to take seriously.



[A Solvable Model of Neural Scaling Laws, Maloney et al, 2022]

Thank you for listening!