

# **LLMs as few-shot learners**

**Advanced NLP: Summer 2023**

**Anoop Sarkar**

**"Language provides a natural domain for the study of artificial intelligence, as the vast majority of reasoning tasks can be efficiently expressed and evaluated in language, and the world's text provides a wealth of data for unsupervised learning via generative modeling."**

**- OpenAI**

---

# Improving Language Understanding by Generative Pre-Training

---

GPT1

**Alec Radford**

OpenAI

[alec@openai.com](mailto:alec@openai.com)

**Karthik Narasimhan**

OpenAI

[karthikn@openai.com](mailto:karthikn@openai.com)

**Tim Salimans**

OpenAI

[tim@openai.com](mailto:tim@openai.com)

**Ilya Sutskever**

OpenAI

[ilyasu@openai.com](mailto:ilyasu@openai.com)

# GPT1

## Pre-training an autoregressive language model

- Start with a large amount of unlabeled data  $\mathcal{U} = \{u_1, \dots, u_n\}$
- Pre-training objective: Maximize the likelihood of predicting the next token

$$\bullet \quad L_i(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

$U = (u_{-k}, \dots, u_{-1})$  is the context vector of tokens

- This is equivalent to training a Transformer decoder

$$\bullet \quad h_0 = U \boxed{W_e} + W_p$$

$n$  is the number of Transformer layers

$W_e$  is the token embedding matrix

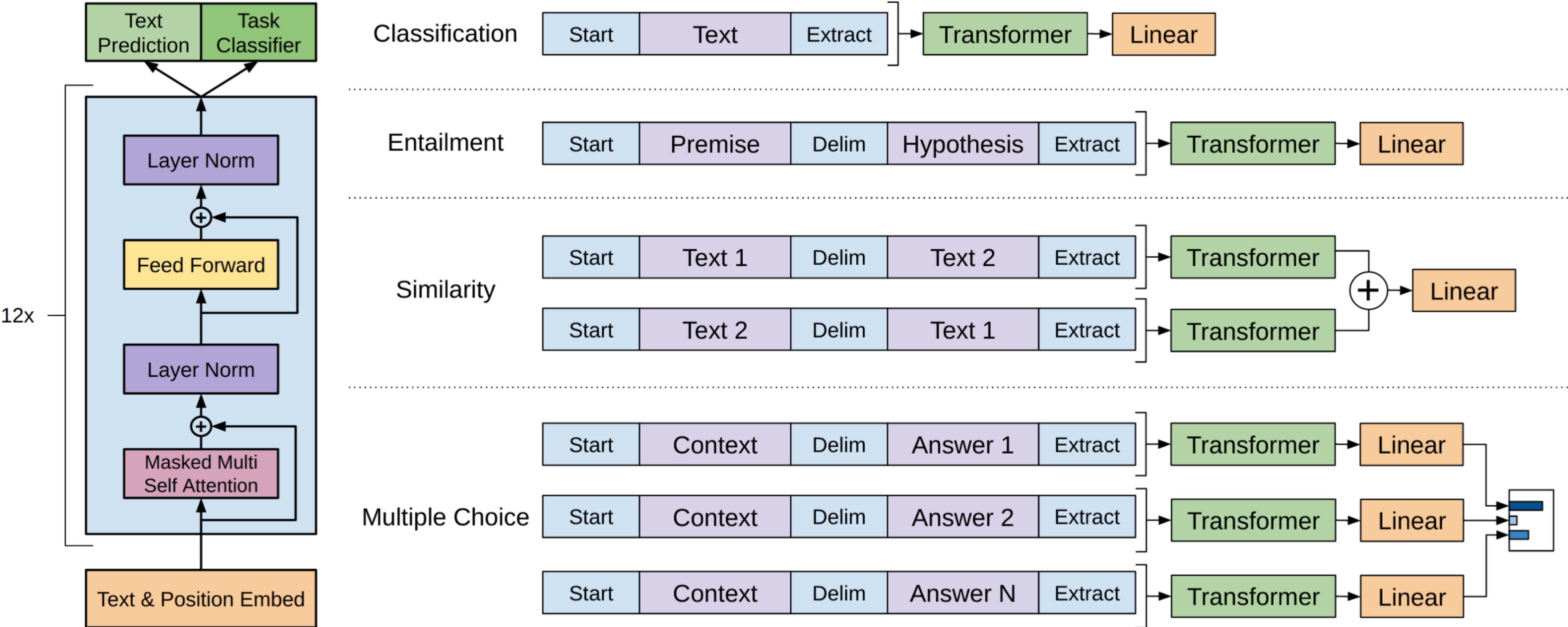
$$\bullet \quad h_\ell = \text{transformer\_block}(h_{\ell-1}) \forall \ell \in [1, n]$$

$W_p$  is the position embedding matrix

$$\bullet \quad P(u) = \text{softmax}(h_n \boxed{W_e^T})$$

- Directionality is needed to generate a well-formed probability distribution

BooksCorpus: 7K unpublished books  
(1B words)



---

# 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**

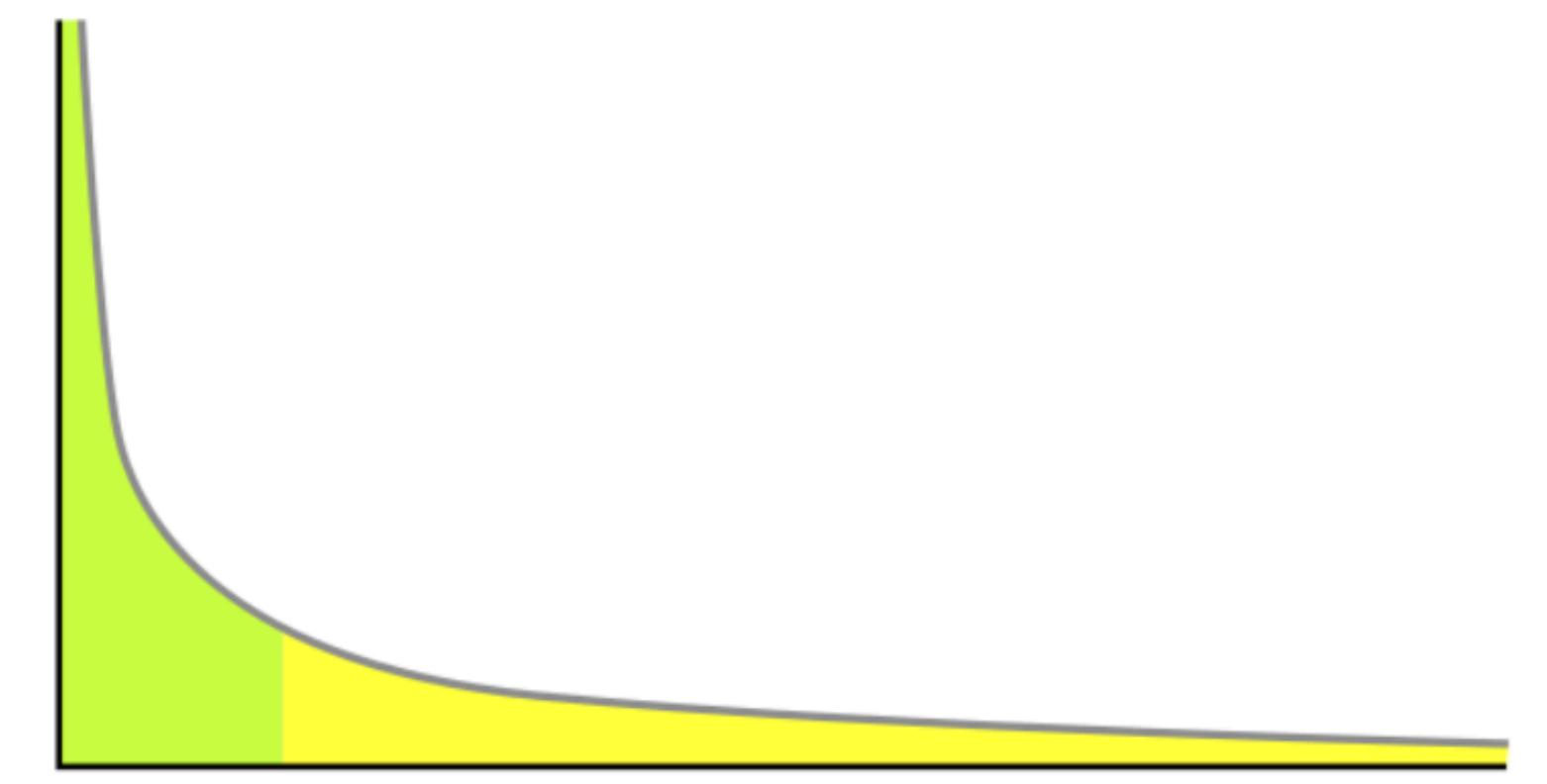
OpenAI  
jeffwu@openai.com

**Dario Amodei**

OpenAI  
damodei@openai.com

# Scaling Laws for LLMs

## Power laws

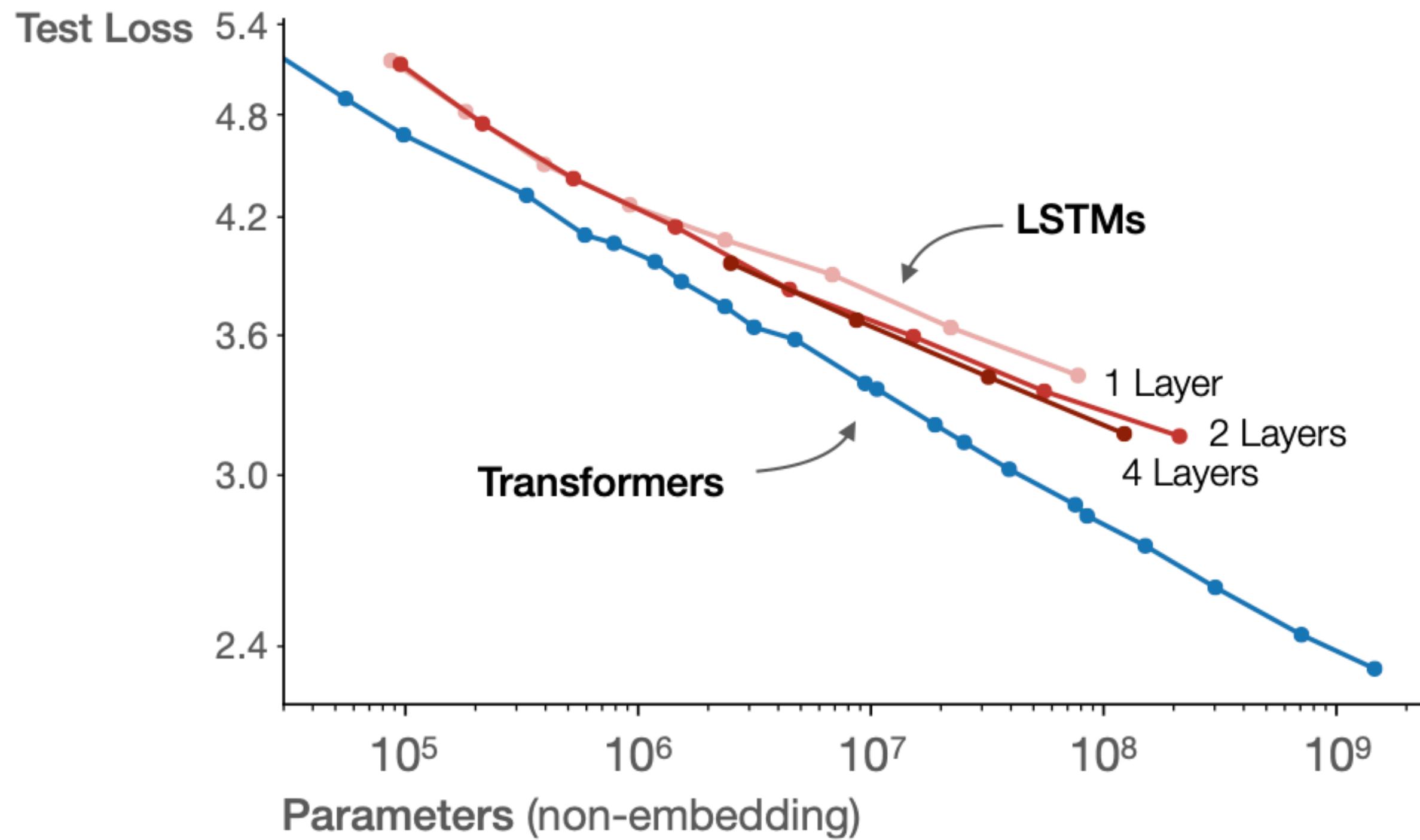


- A power law is a relation between two quantities:  $f(x) = (a/x)^k$  e.g. model performance vs. model size.
- Number of model parameters N (excluding subword embeddings)
- Size of dataset D
- Amount of compute (MFLOPs) C
- N, D, C are dominant. Other choices in hyperparameters like width vs. depth are less relevant
- 1 PetaFLOP-day (PF-day) is  $8.64 \times 10^{19}$  FLOPS

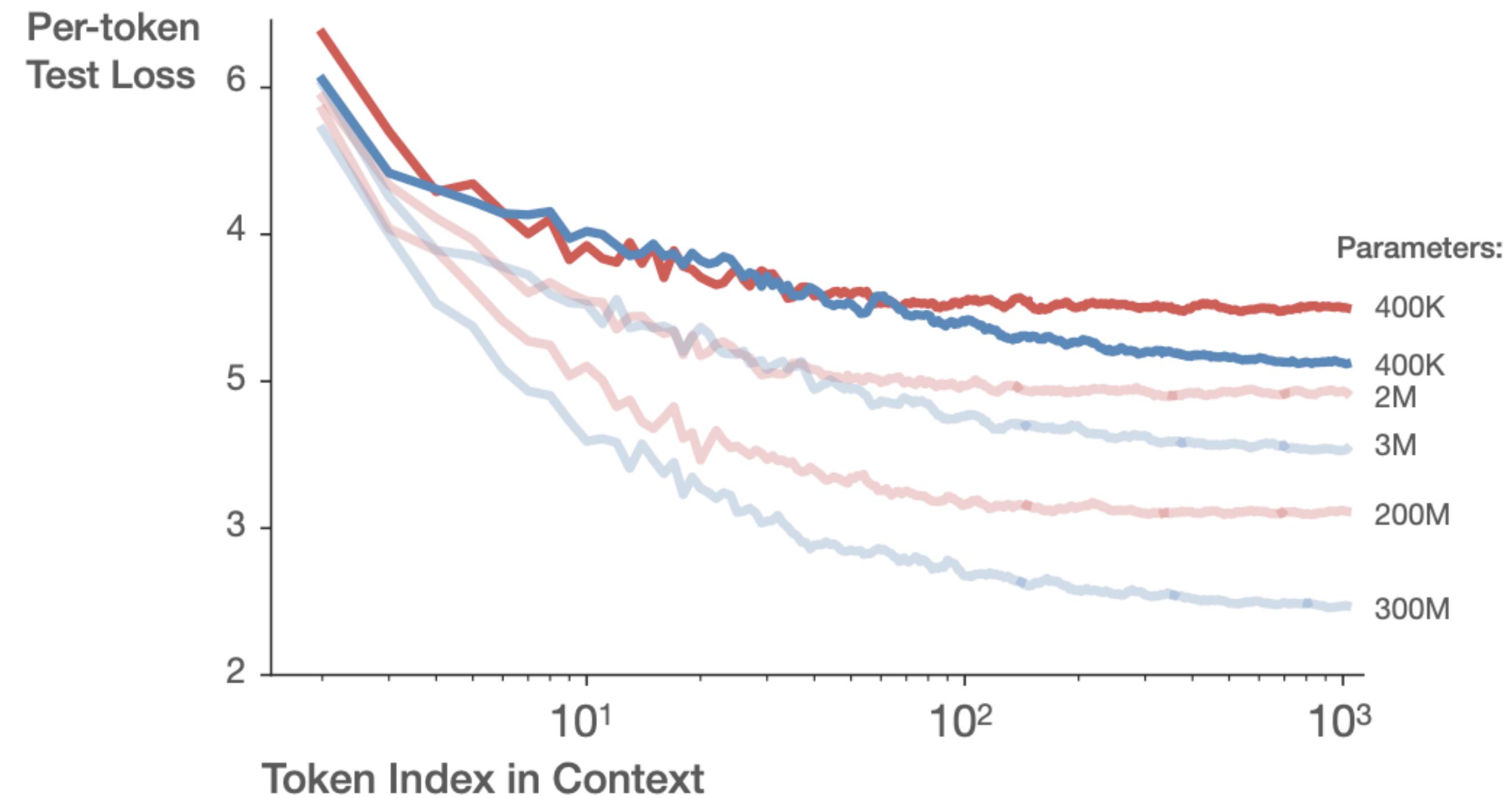
<b>Operation</b>	<b>Parameters</b>	<b>FLOPs per Token</b>
Embed	$(n_{\text{vocab}} + n_{\text{ctx}}) d_{\text{model}}$	$4d_{\text{model}}$
Attention: QKV	$n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$	$2n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$
Attention: Mask	—	$2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$
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	—	$2d_{\text{model}} n_{\text{vocab}}$
<b>Total (Non-Embedding)</b>	$N = 2d_{\text{model}} n_{\text{layer}} (2d_{\text{attn}} + d_{\text{ff}})$	$C_{\text{forward}} = 2N + 2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$

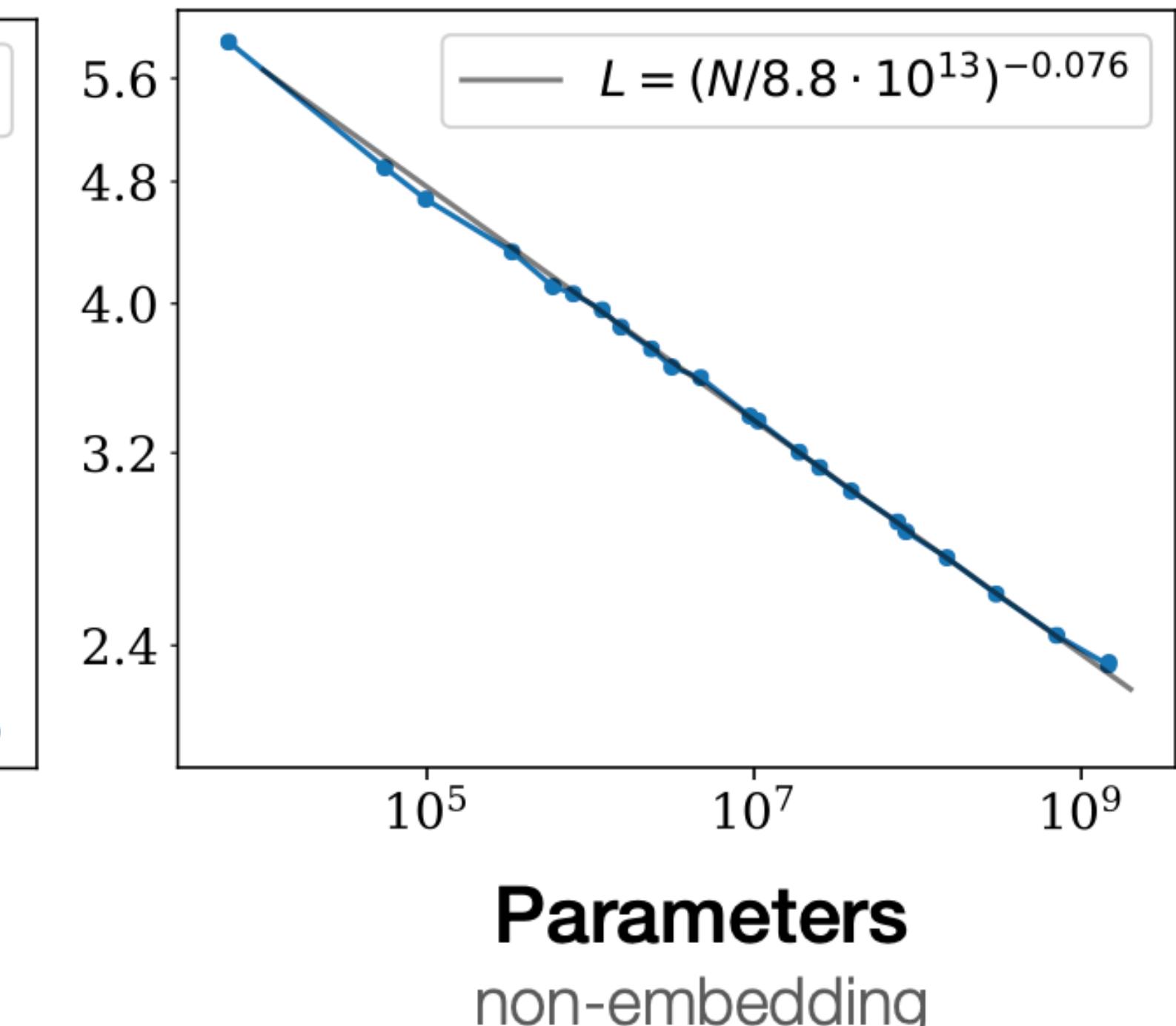
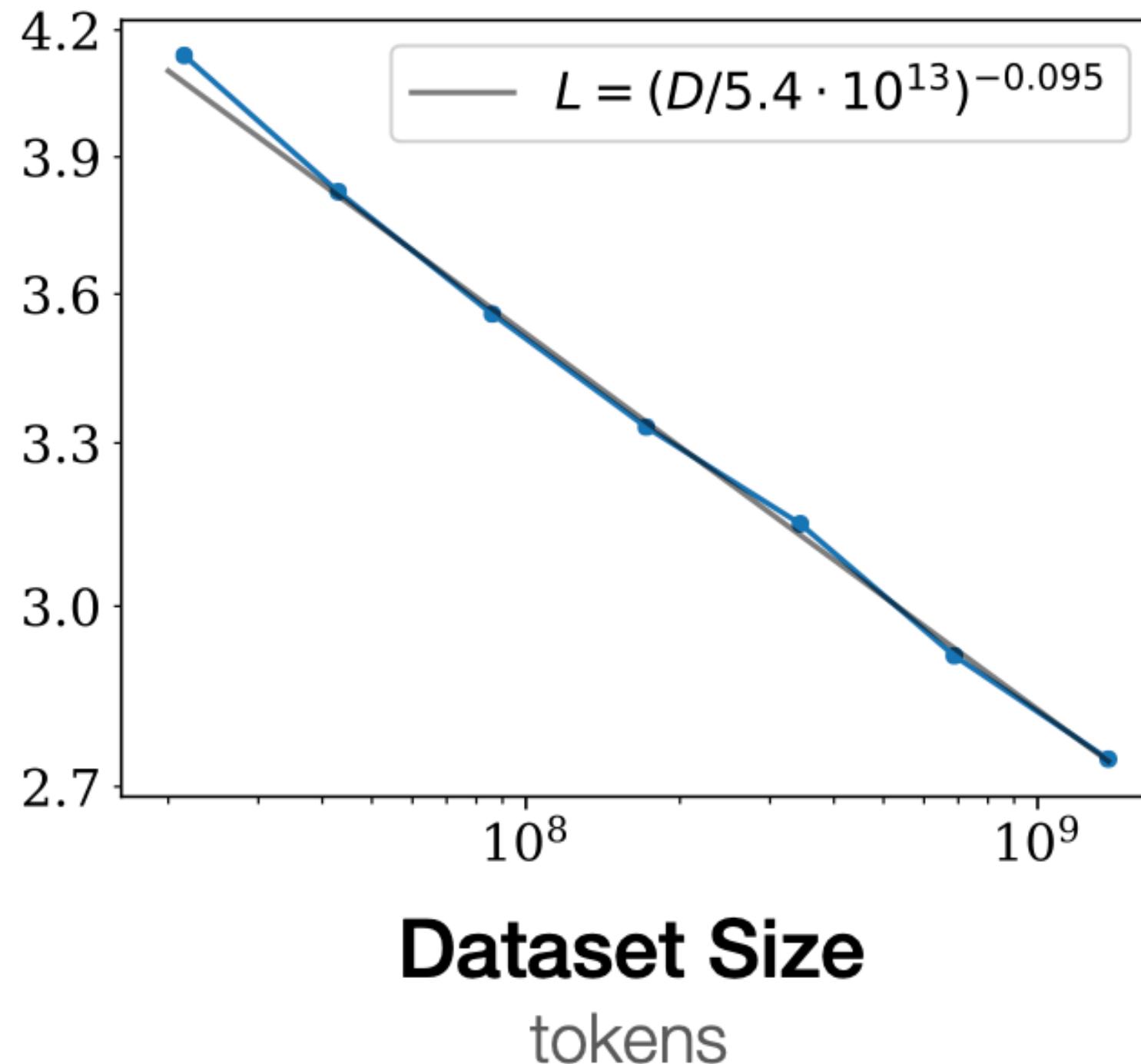
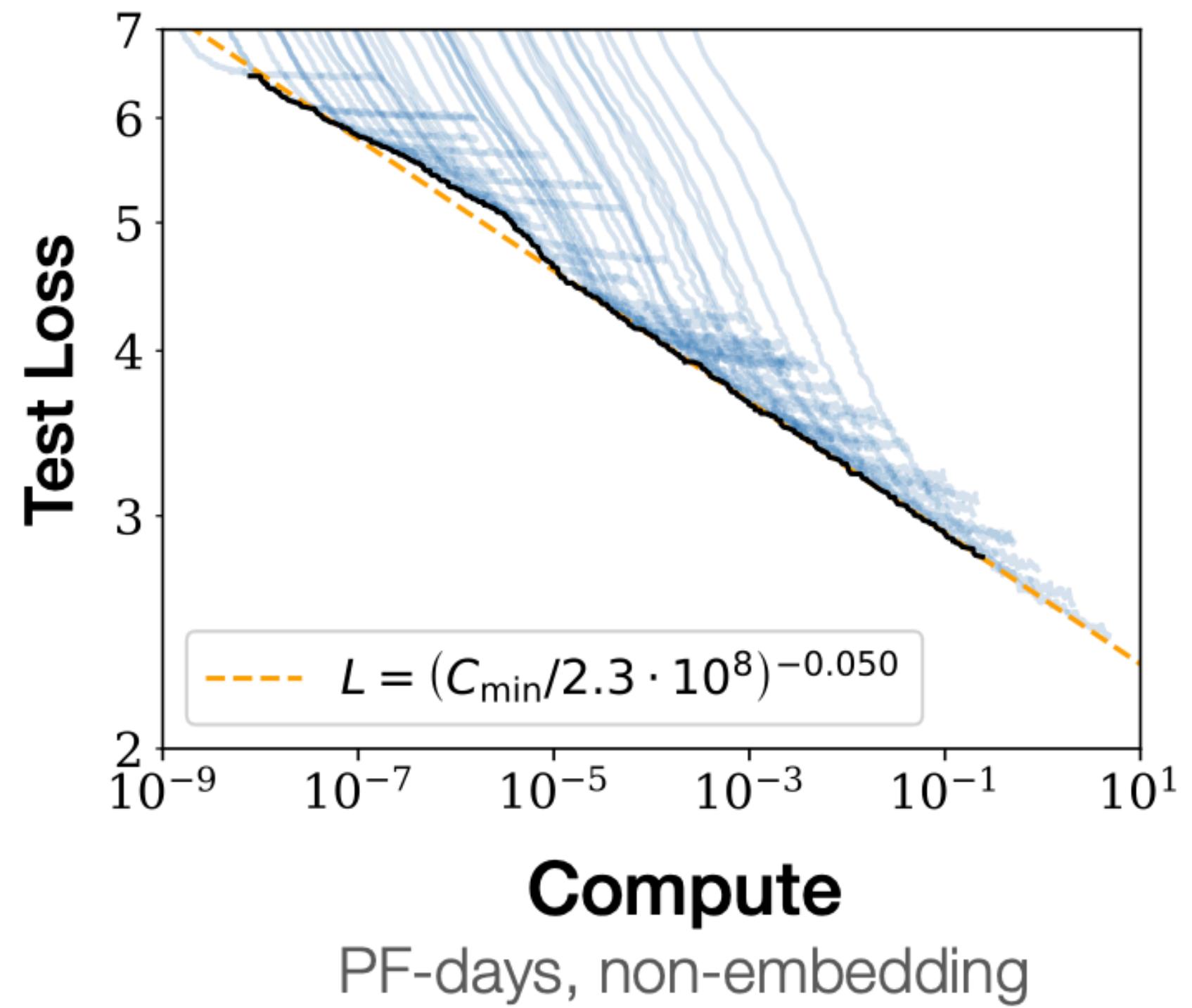
**Table 1** Parameter counts and compute (forward pass) estimates for a Transformer model. Sub-leading terms such as nonlinearities, biases, and layer normalization are omitted.

**Transformers asymptotically outperform LSTMs  
due to improved use of long contexts**



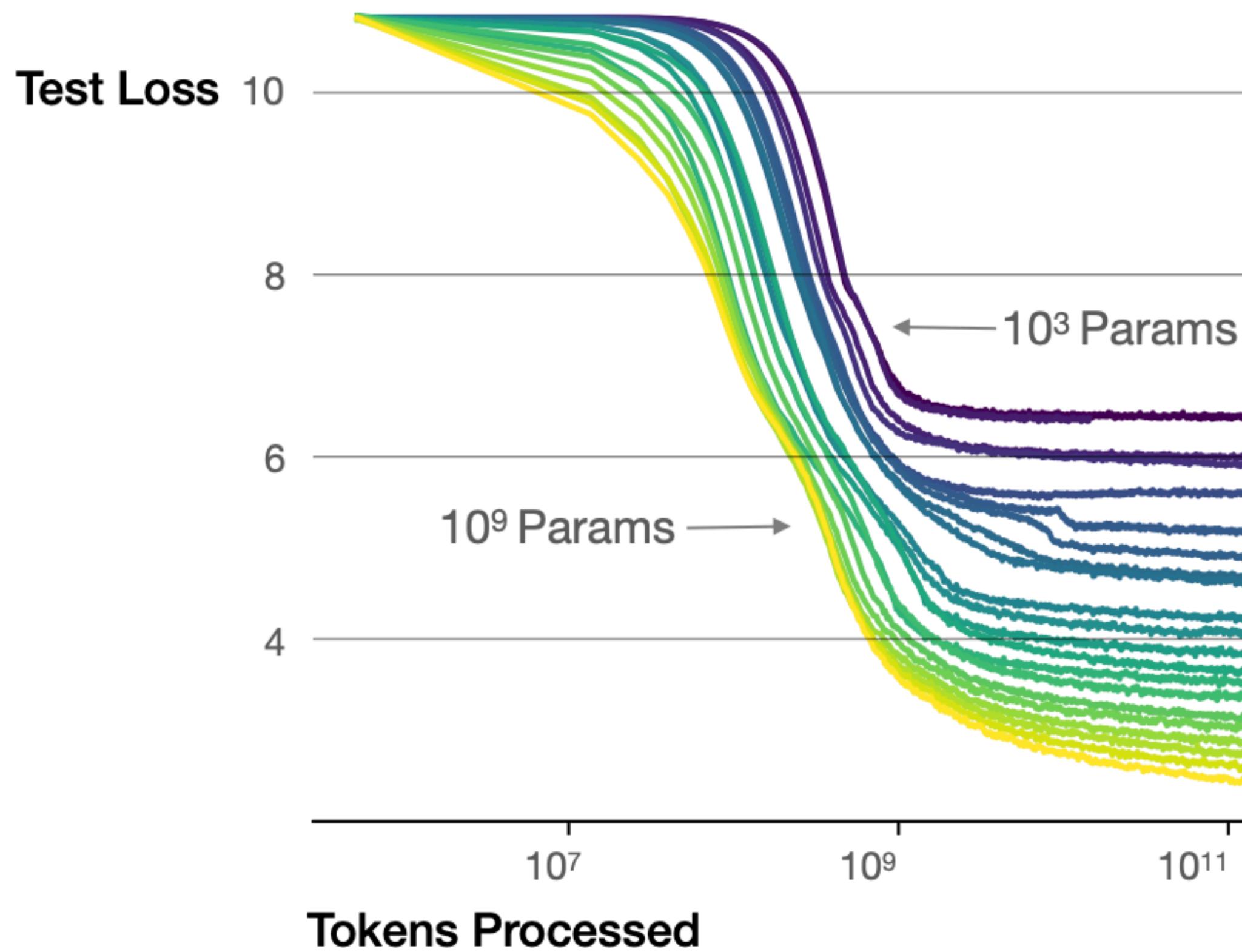
**LSTM plateaus after <100 tokens  
Transformer improves through the whole context**



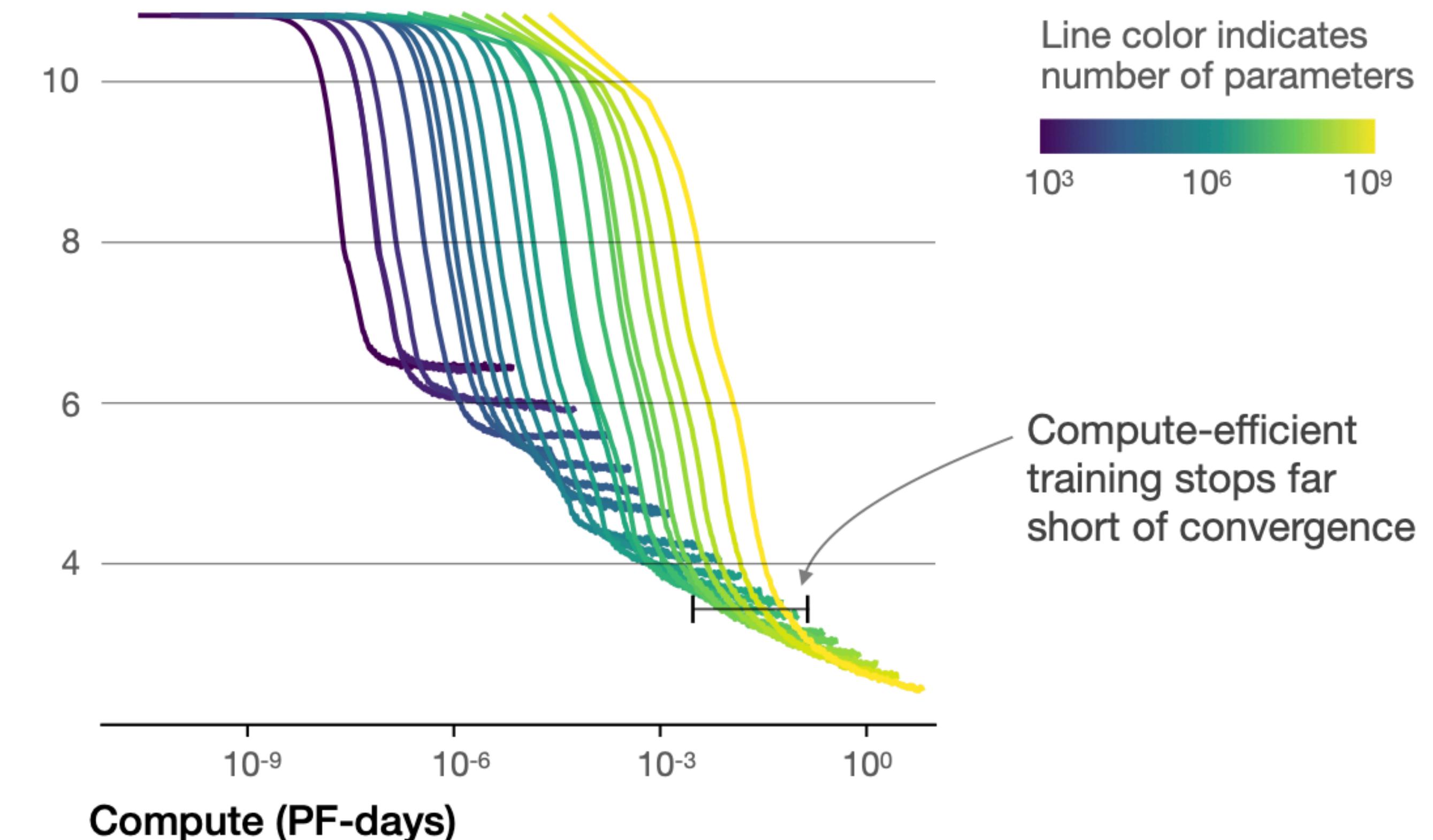


**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual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

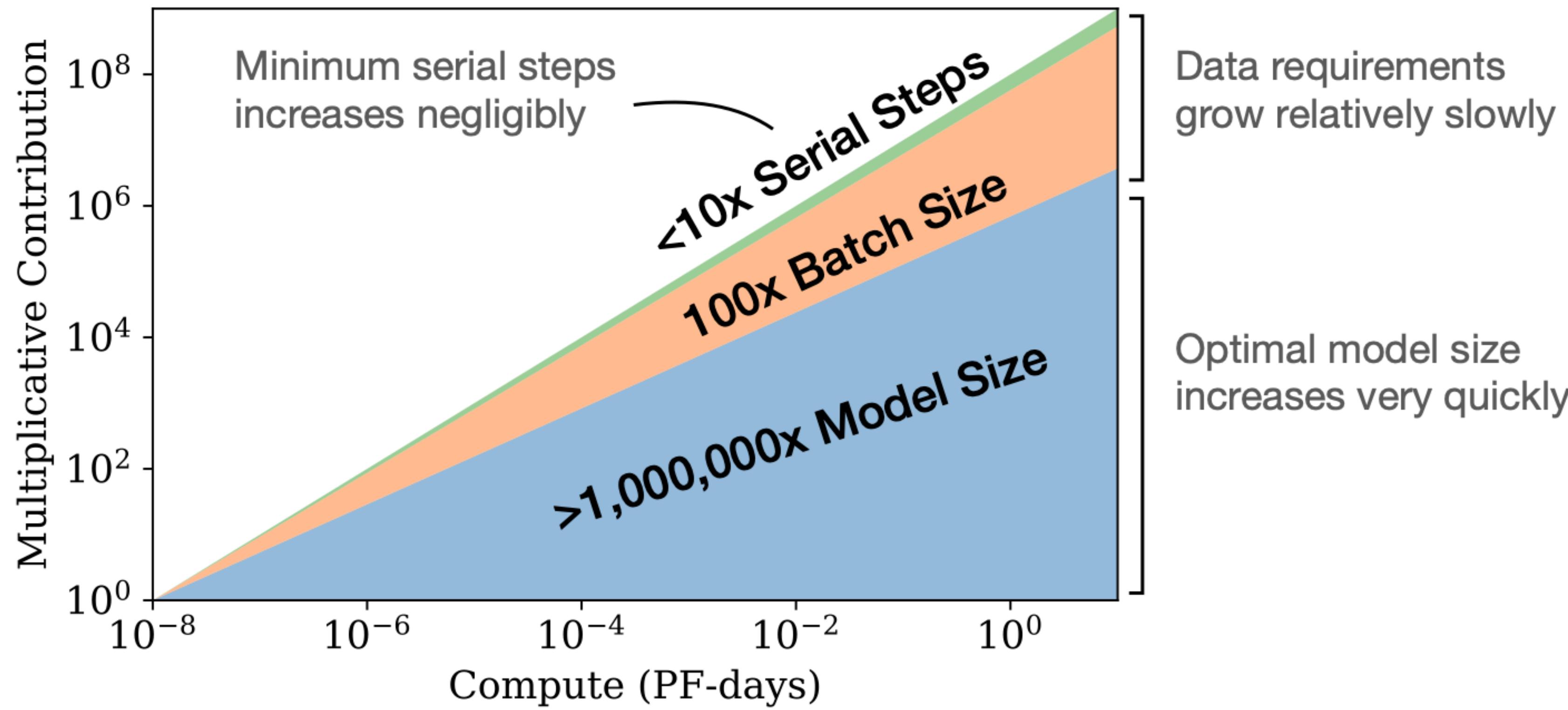
Larger models require **fewer samples** to reach the same performance



The optimal model size grows smoothly with the loss target and compute budget



**Figure 2** We show a series of language model training runs, with models ranging in size from  $10^3$  to  $10^9$  parameters (excluding embeddings).



**Figure 3** As more compute becomes available, we can choose how much to allocate towards training larger models, using larger batches, and training for more steps. We illustrate this for a billion-fold increase in compute. For optimally compute-efficient training, most of the increase should go towards increased model size. A relatively small increase in data is needed to avoid reuse. Of the increase in data, most can be used to increase parallelism through larger batch sizes, with only a very small increase in serial training time required.

# Power laws for test loss

- Let  $L(\cdot)$  represent the test loss dependent on either parameters N, or dataset size D or compute C

- For models with limited number of parameters:

$$L(N) = (N_c/N)^{\alpha_N}; \alpha_N \approx 0.076, N_c \approx 8.8 \times 10^{13} \text{(non-embd params)}$$

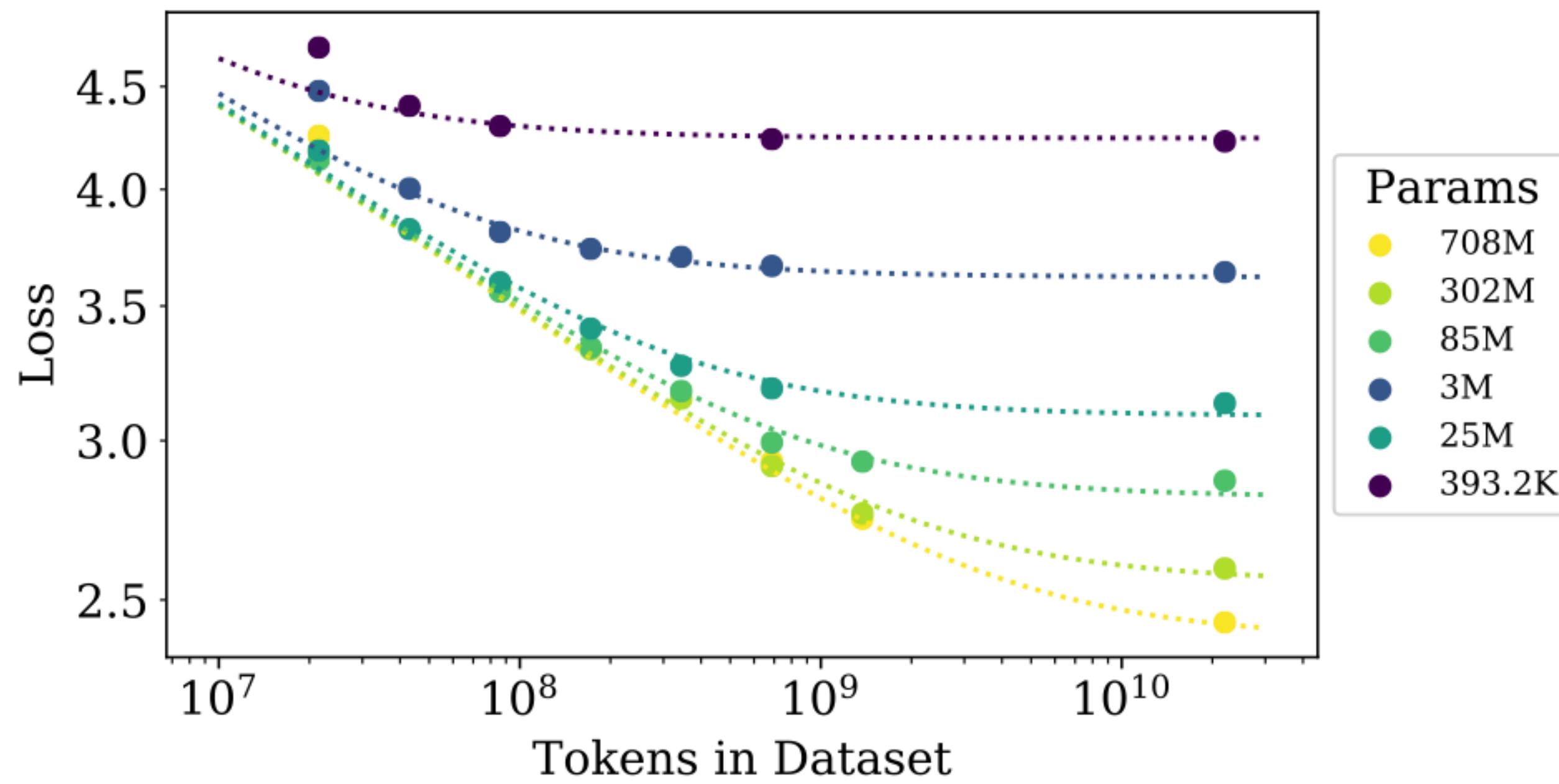
- For models with limited dataset size:

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

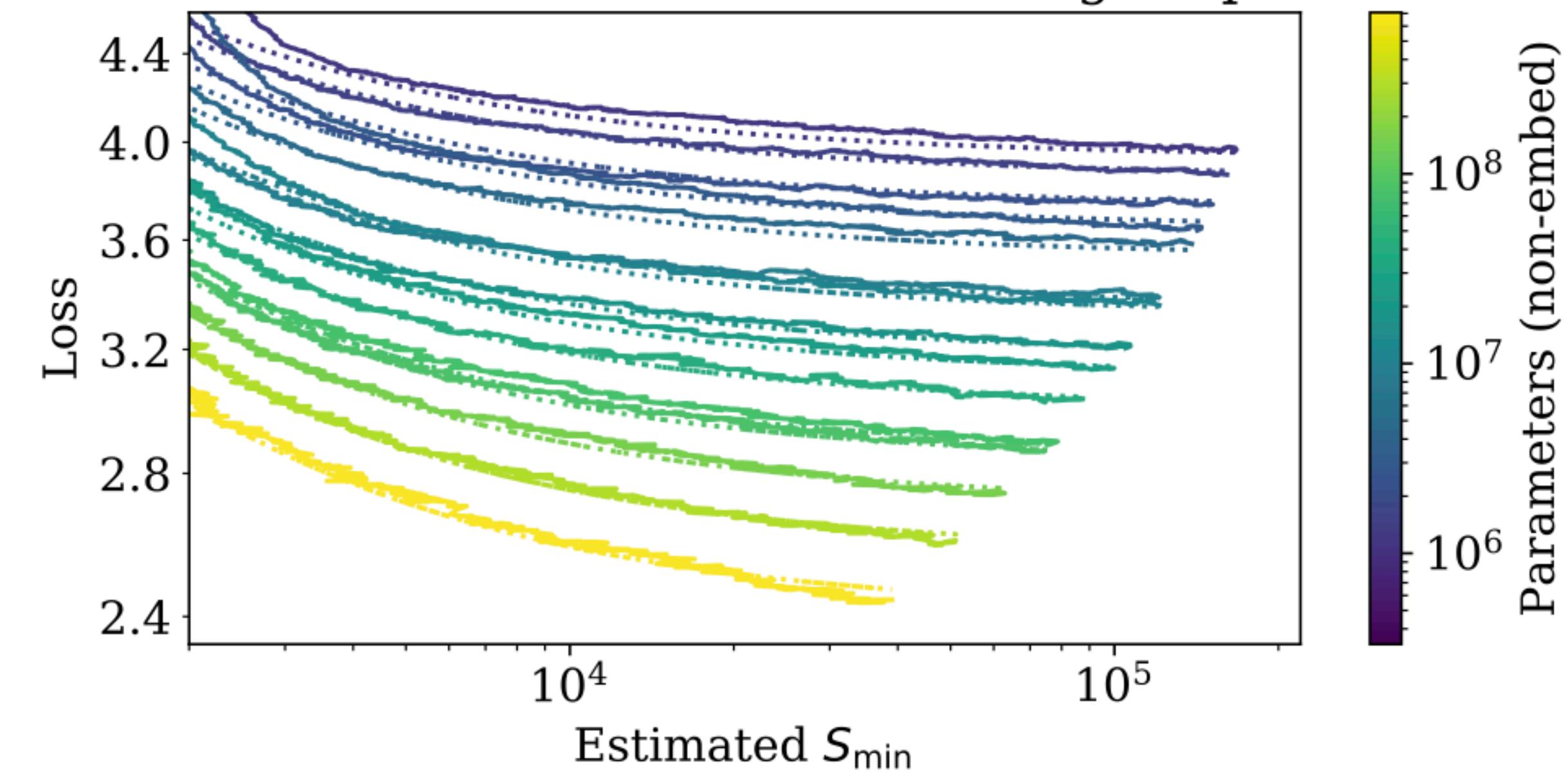
- For models trained with limited compute:

$$L(C) = (C_c^{\min}/C_{\min})^{\alpha_C^{\min}}; \alpha_C^{\min} \approx 0.050, C_c^{\min} \approx 3.1 \times 10^8 \text{(PF-days)}$$

### Loss vs Model and Dataset Size



### Loss vs Model Size and Training Steps

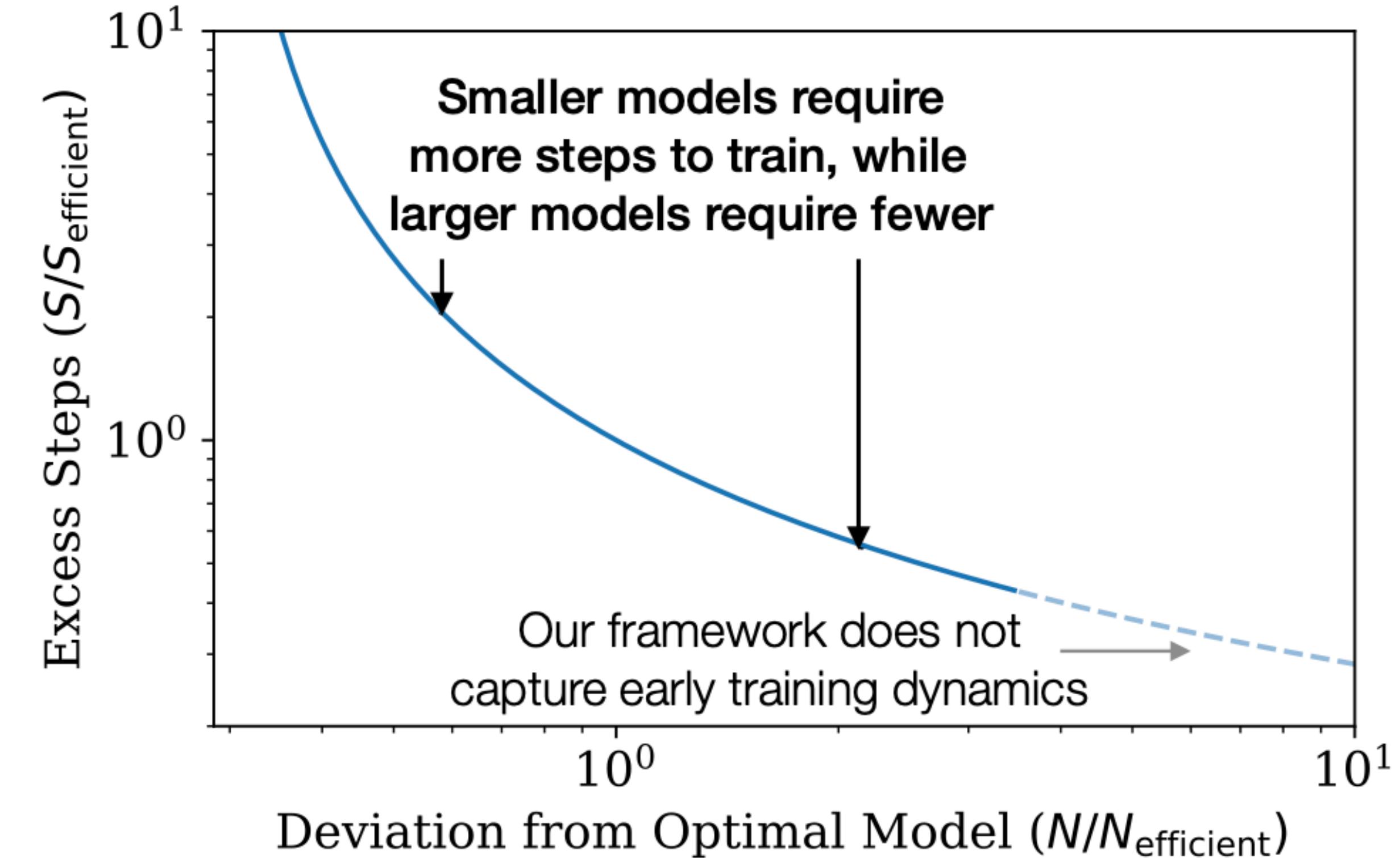
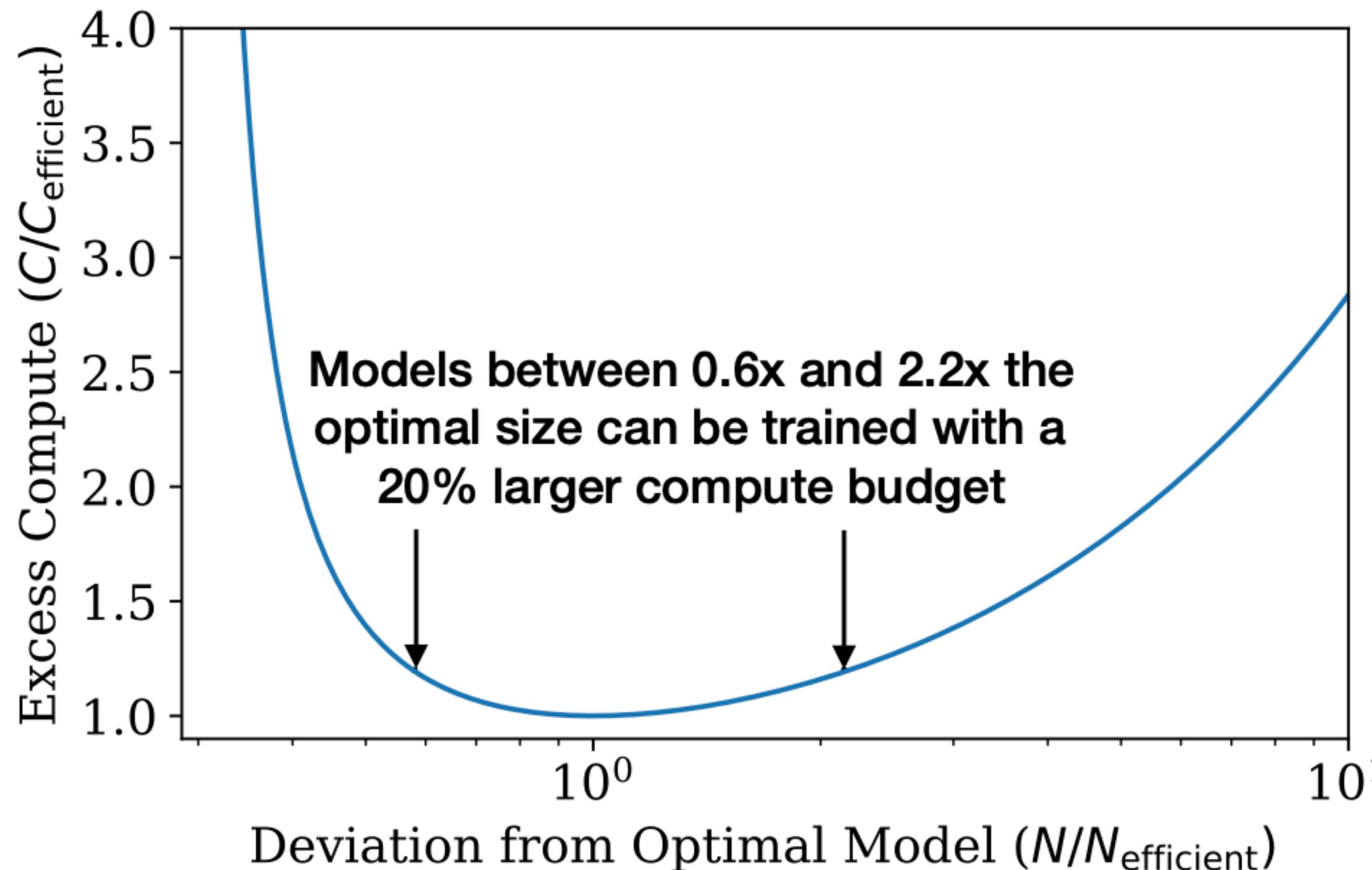


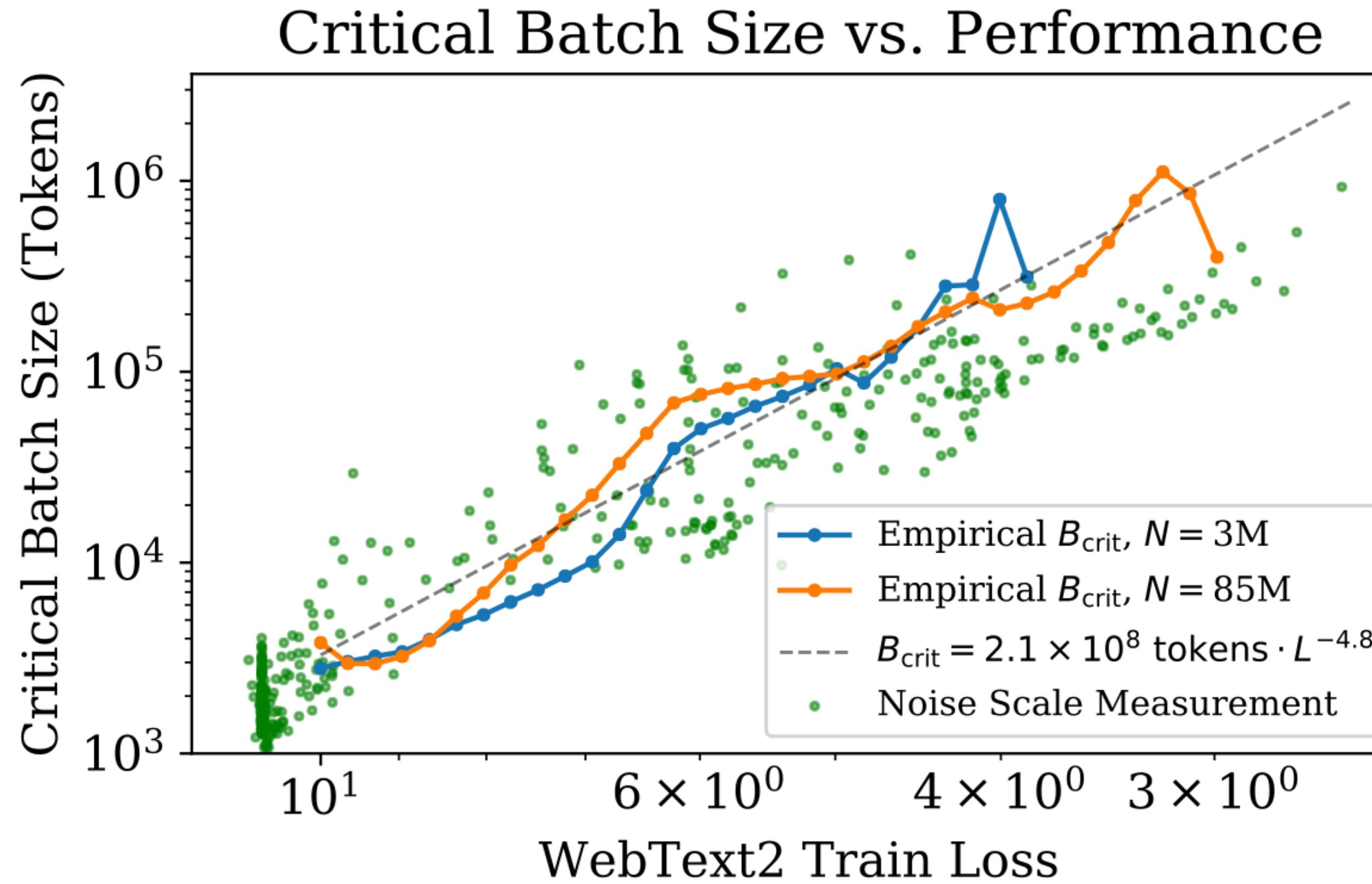
$$L(N, D) = \left[ \left( \frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right]^{\alpha_D}$$

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

$S$  = parameter update steps

# Optimal Allocation of Compute Budget





**Figure 10** The critical batch size  $B_{\text{crit}}$  follows a power law in the loss as performance increase, and does not depend directly on the model size. We find that the critical batch size approximately doubles for every 13% decrease in loss.  $B_{\text{crit}}$  is measured empirically from the data shown in Figure 18, but it is also roughly predicted by the gradient noise scale, as in [MKAT18]. arXiv:1812.06162

# Lessons from scaling LLMs

- Number of model parameters N  
Size of dataset D
- Amount of compute (MFLOPs) C
- Performance depends strongly on scale, weakly on model shape
- Performance has a power-law relationship with each of the three scale factors N, D, C when not bottlenecked by the other two
- Performance improves predictably as long as we scale up N and D in tandem
- Training curves follow predictable power-laws whose parameters are roughly independent of the model size

# Lessons from scaling LLMs

- Transfer to a different distribution incurs a constant penalty but otherwise improves roughly in line with performance on the training set.
- Large models are more sample-efficient than small models, reaching the same level of performance with fewer optimization steps and using fewer data points
- The ideal batch size for training these models is roughly a power of the loss only, and continues to be determinable by measuring the gradient noise scale

The GPT2 paper

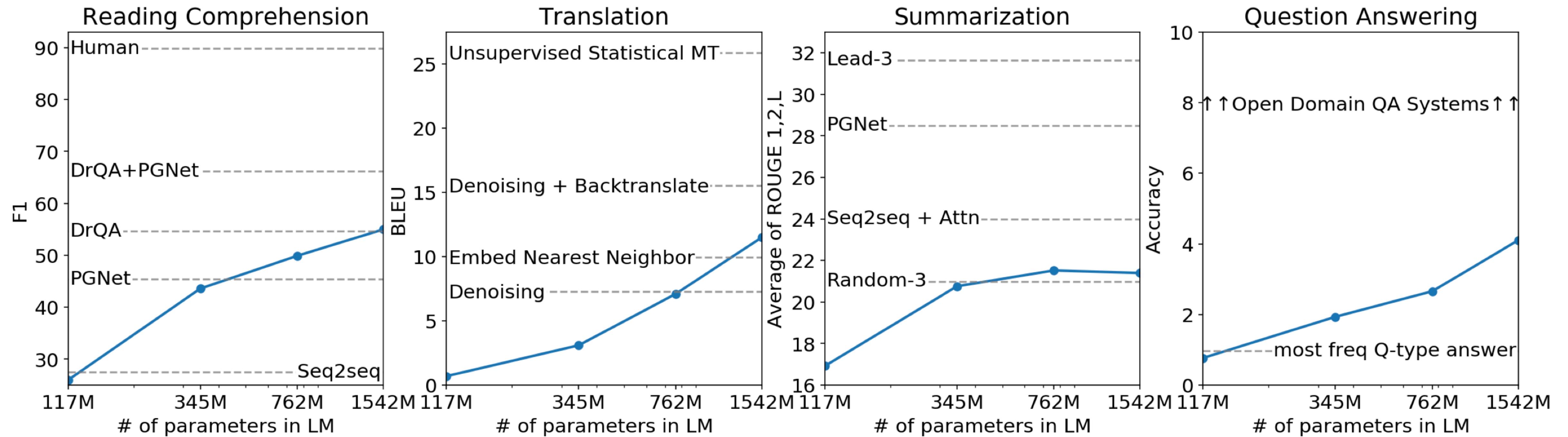
---

# **Language Models are Unsupervised Multitask Learners**

---

**Alec Radford \*<sup>1</sup> Jeffrey Wu \*<sup>1</sup> Rewon Child<sup>1</sup> David Luan<sup>1</sup> Dario Amodei \*\*<sup>1</sup> Ilya Sutskever \*\*<sup>1</sup>**

Feb 2019



# WebText corpus

- Train on web scale corpus but with more reliable data compared to the CommonCrawl.
- English-only, so language detection is used
- Outgoing links from reddit (with at least 3 karma)
- No reddit data was used, instead use the content of the web sites linked on reddit discussions
- 8M documents with 40GB of text

Language detection: <https://github.com/CLD2Owners/cld2>

News site scraping: <https://github.com/codelucas/newspaper>

---

”I’m not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile** [I’m not a fool].

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: **”Mentez mentez, il en restera toujours quelque chose,”** which translates as, **”Lie lie and something will always remain.”**

“I hate the word ‘perfume,’” Burr says. ‘It’s somewhat better in French: ‘parfum.’

If listened carefully at 29:55, a conversation can be heard between two guys in French: **“-Comment on fait pour aller de l’autre côté? -Quel autre côté?”**, which means **“- How do you get to the other side? - What side?”**.

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

**“Brevet Sans Garantie Du Gouvernement”,** translated to English: **“Patented without government warranty”.**

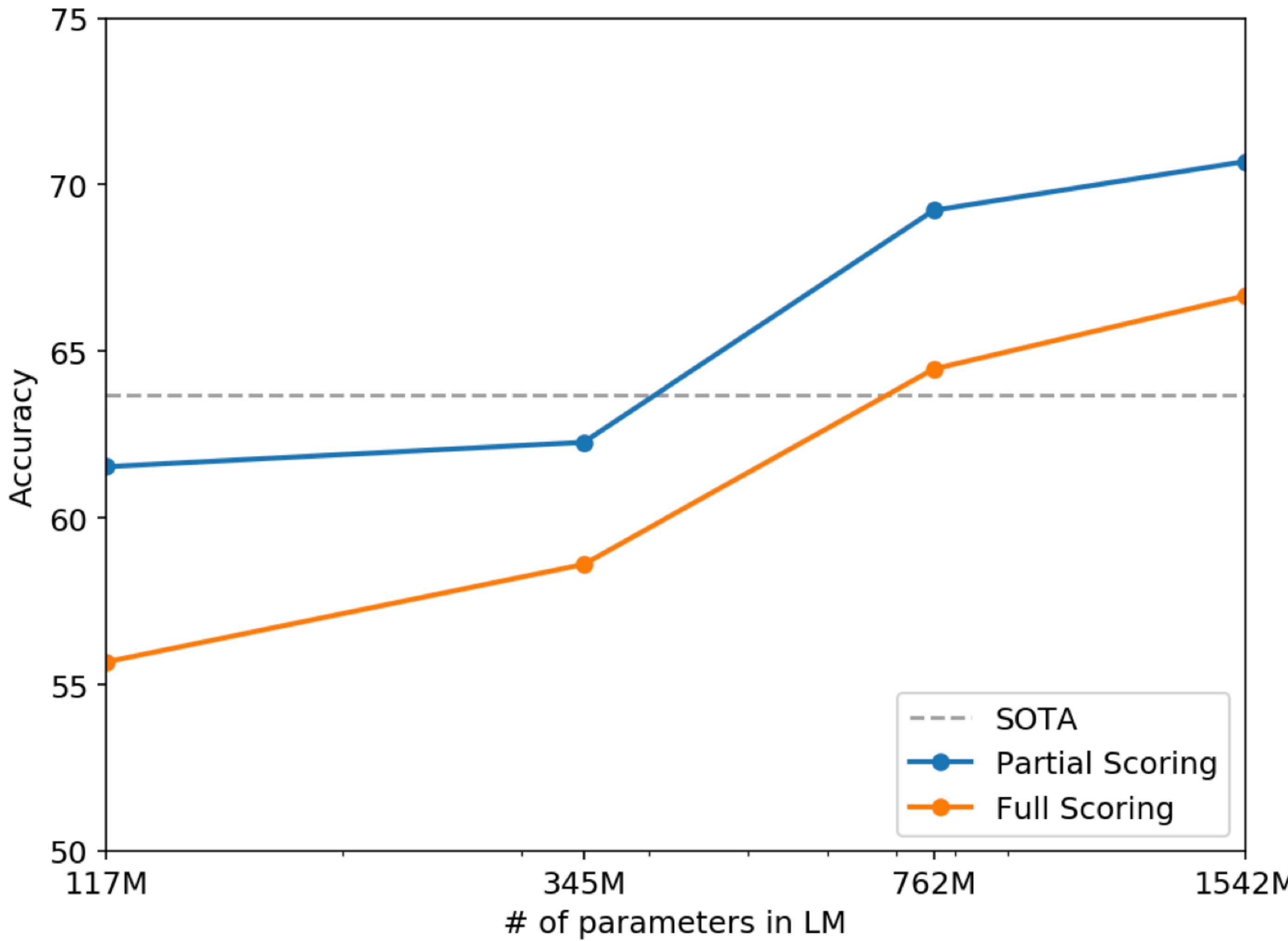
---

*Table 1.* Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

Parameters	Layers	$d_{model}$
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

*Table 2.* Architecture hyperparameters for the 4 model sizes.

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	<b>21.8</b>
117M	<b>35.13</b>	45.99	<b>87.65</b>	<b>83.4</b>	<b>29.41</b>	65.85	1.16	1.17	37.50	75.20
345M	<b>15.60</b>	55.48	<b>92.35</b>	<b>87.1</b>	<b>22.76</b>	47.33	1.01	<b>1.06</b>	26.37	55.72
762M	<b>10.87</b>	<b>60.12</b>	<b>93.45</b>	<b>88.0</b>	<b>19.93</b>	<b>40.31</b>	<b>0.97</b>	<b>1.02</b>	22.05	44.575
1542M	<b>8.63</b>	<b>63.24</b>	<b>93.30</b>	<b>89.05</b>	<b>18.34</b>	<b>35.76</b>	<b>0.93</b>	<b>0.98</b>	<b>17.48</b>	42.16



*Figure 3. Performance on the Winograd Schema Challenge as a function of model capacity.*

# Language Models are Few-Shot Learners

---

**Tom B. Brown\***

**Benjamin Mann\***

**Nick Ryder\***

**Melanie Subbiah\***

**Jared Kaplan<sup>†</sup>**

**Prafulla Dhariwal**

**Arvind Neelakantan**

**Pranav Shyam**

**Girish Sastry**

**Amanda Askell**

**Sandhini Agarwal**

**Ariel Herbert-Voss**

**Gretchen Krueger**

**Tom Henighan**

**Rewon Child**

**Aditya Ramesh**

**Daniel M. Ziegler**

**Jeffrey Wu**

**Clemens Winter**

**Christopher Hesse**

**Mark Chen**

**Eric Sigler**

**Mateusz Litwin**

**Scott Gray**

**Benjamin Chess**

**Jack Clark**

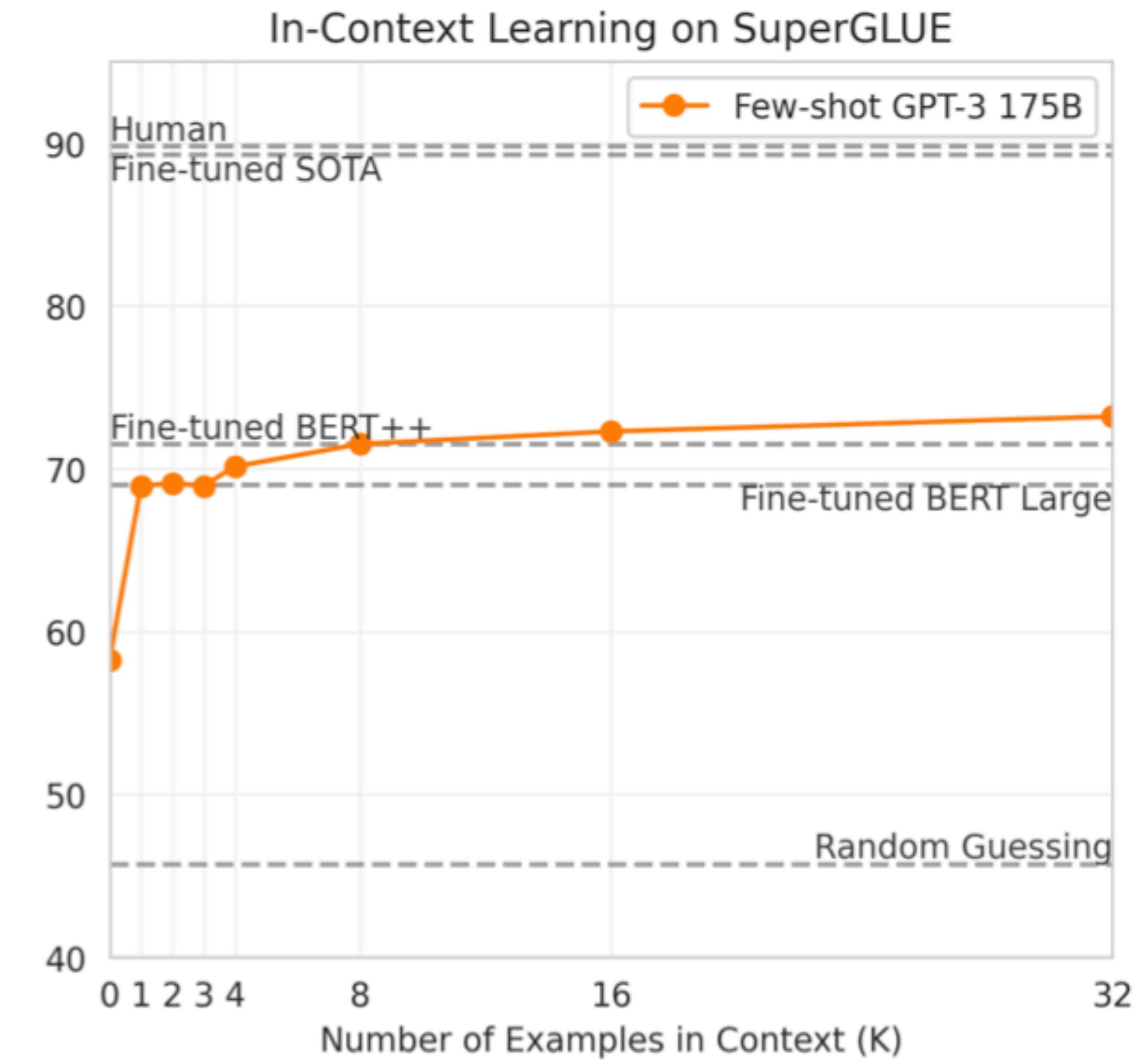
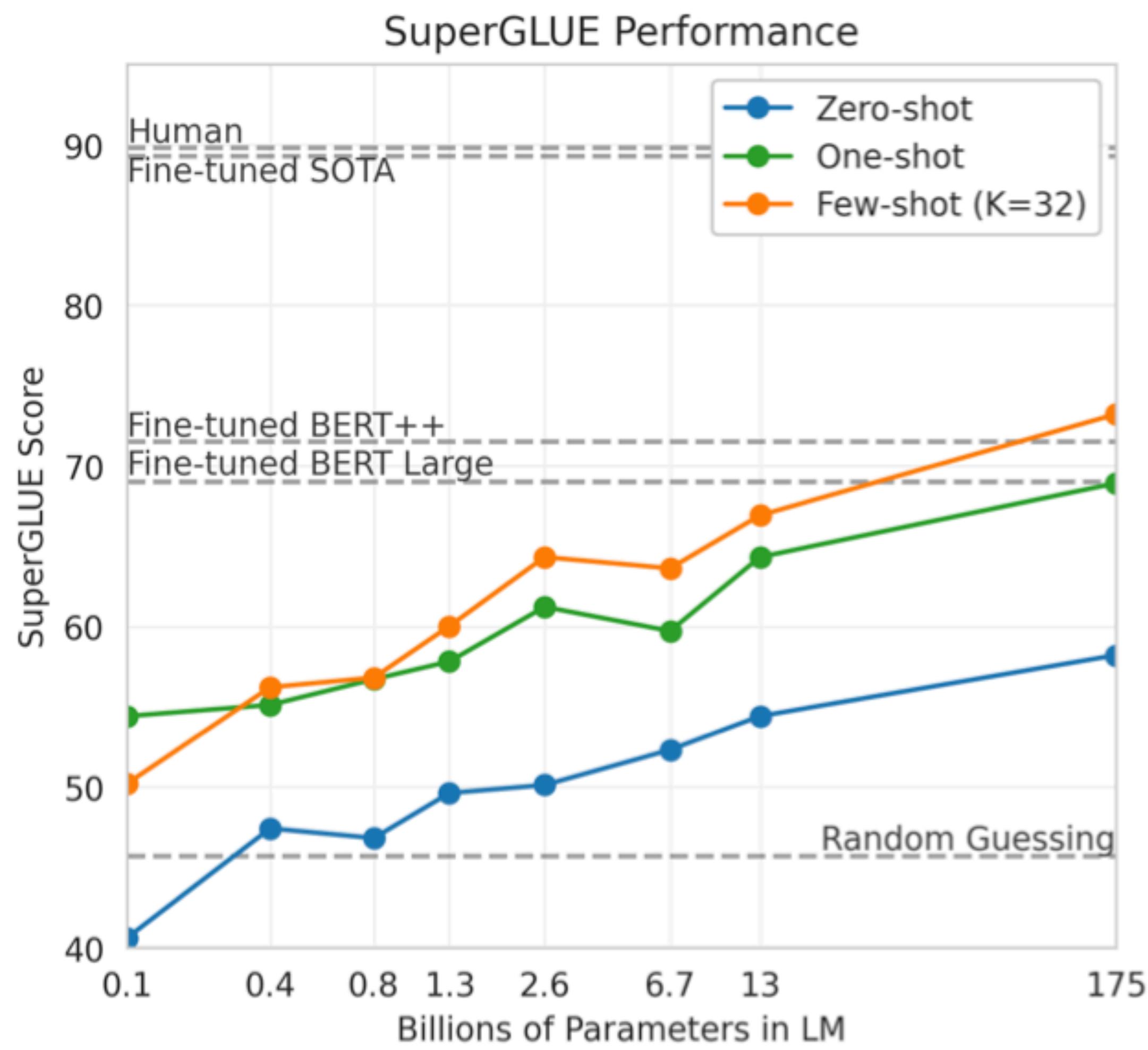
**Christopher Berner**

**Sam McCandlish**

**Alec Radford**

**Ilya Sutskever**

**Dario Amodei**



**Performance on SuperGLUE increases with number of examples in context.** We find the difference in performance between the BERT-Large and BERT++ to be roughly equivalent to the difference between GPT-3 with one example per context versus eight examples per context.

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	<b>89.0</b>	<b>91.0</b>	<b>96.9</b>	<b>93.9</b>	<b>94.8</b>	<b>92.5</b>
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	<b>76.1</b>	<b>93.8</b>	<b>62.3</b>	<b>88.2</b>	<b>92.5</b>	<b>93.3</b>
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

**Table 3.5:** Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

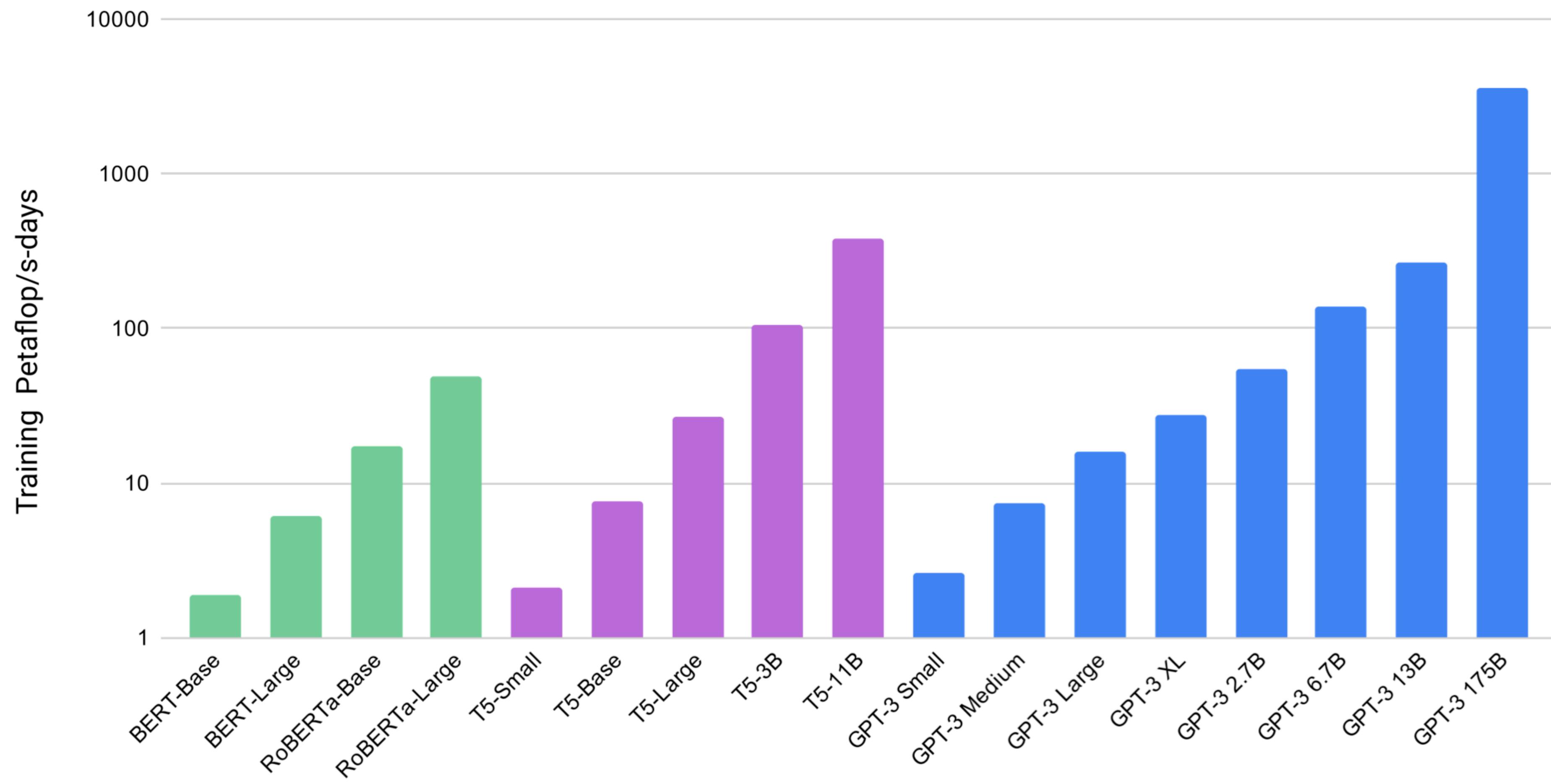
Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0 <sup>a</sup>	8.63 <sup>b</sup>	<b>91.8<sup>c</sup></b>	<b>85.6<sup>d</sup></b>
GPT-3 Zero-Shot	<b>76.2</b>	<b>3.00</b>	83.2	78.9
GPT-3 One-Shot	<b>72.5</b>	<b>3.35</b>	84.7	78.1
GPT-3 Few-Shot	<b>86.4</b>	<b>1.92</b>	87.7	79.3

Setting		NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP <sup>+</sup> 20]		<b>44.5</b>	<b>45.5</b>	<b>68.0</b>
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]		36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)		34.5	37.4	50.1
GPT-3 Zero-Shot		14.6	14.4	64.3
GPT-3 One-Shot		23.0	25.3	<b>68.0</b>
GPT-3 Few-Shot		29.9	41.5	<b>71.2</b>

Setting	ARC (Easy)	ARC (Challenge)	CoQA	DROP
Fine-tuned SOTA	<b>92.0<sup>a</sup></b>	<b>78.5<sup>b</sup></b>	<b>90.7<sup>c</sup></b>	<b>89.1<sup>d</sup></b>
GPT-3 Zero-Shot	68.8	51.4	81.5	23.6
GPT-3 One-Shot	71.2	53.2	84.0	34.3
GPT-3 Few-Shot	70.1	51.5	85.0	36.5

WMT 2014

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	<b>45.6<sup>a</sup></b>	35.0 <sup>b</sup>	<b>41.2<sup>c</sup></b>	40.2 <sup>d</sup>	<b>38.5<sup>e</sup></b>	<b>39.9<sup>e</sup></b>
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ <sup>+</sup> 19]	<u>37.5</u>	34.9	28.3	35.2	<u>35.2</u>	33.1
mBART [LGG <sup>+</sup> 20]	-	-	<u>29.8</u>	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	<u>39.2</u>	29.7	<u>40.6</u>	21.0	<u>39.5</u>



**Figure 7.2: Total compute used during training.** Based on the analysis in Scaling Laws For Neural Language Models [KMH<sup>+</sup>20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in the Appendix.