北京大学
PEKING UNIVERSITY

Scaling Law

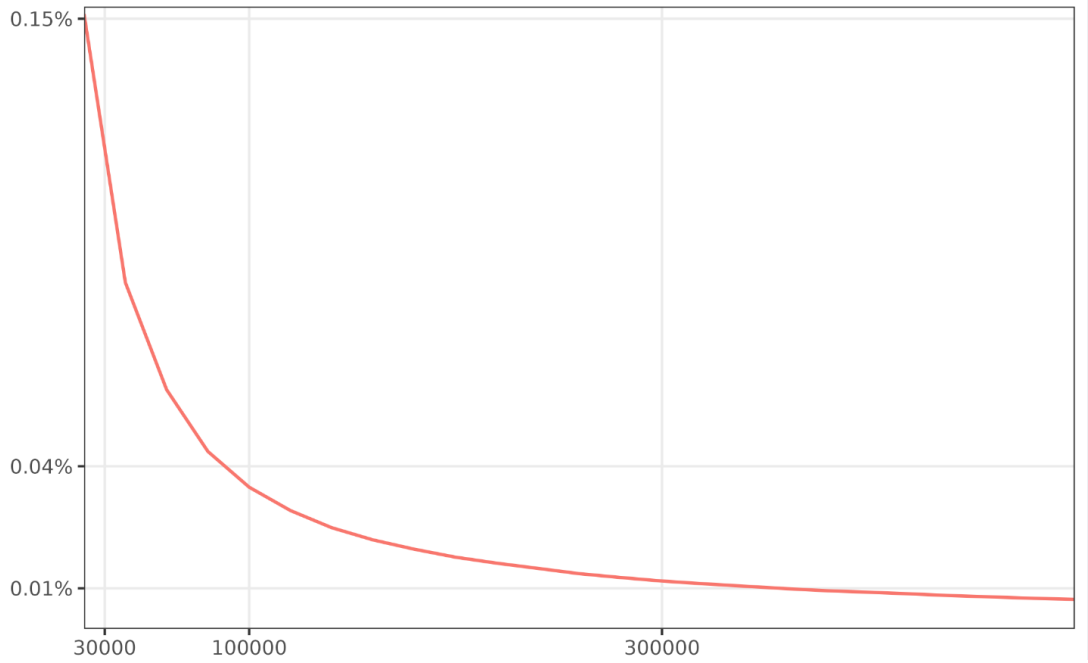
Kun Yuan

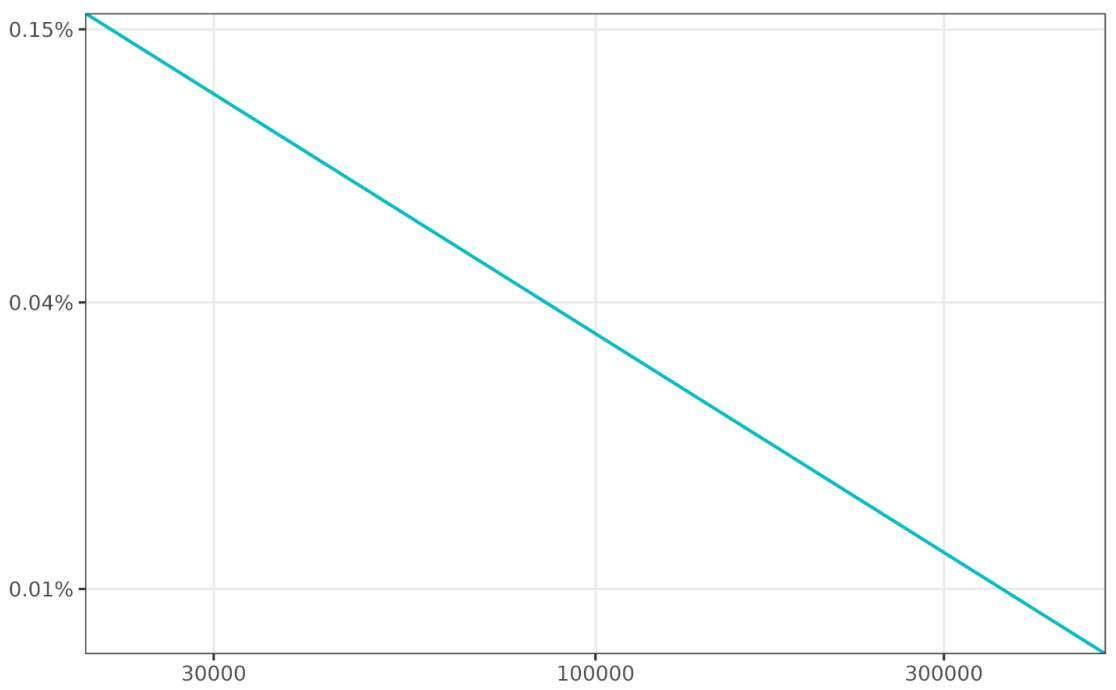
Center for Machine Learning Research @ Peking University

Scaling law (Power law)

The scaling law $$L(x) = (\frac{c}{x})^{\alpha}$$

By taking log, we have $$\log L = \alpha \log c - \alpha \log x$$





A fundamental question

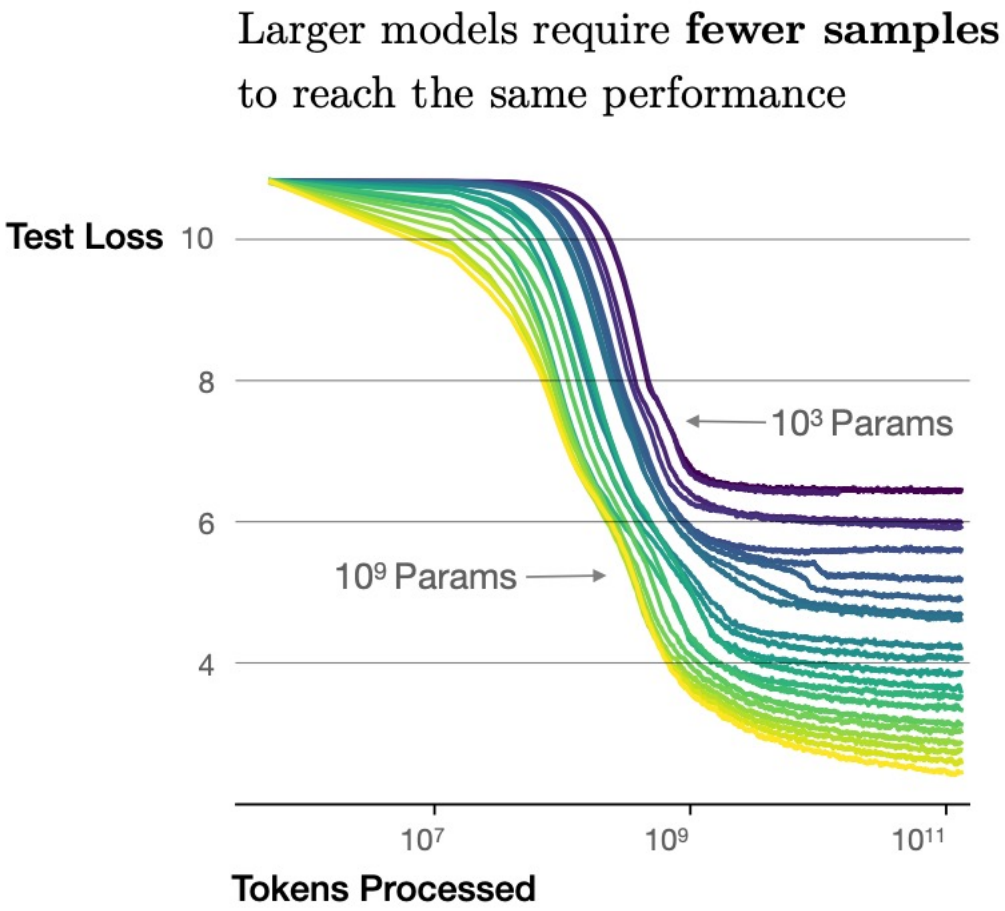
北京大学 PEKING UNIVERSITY

How does the model performance depend on **model shape**, the **size** of neural models, the **computing power** used to train them, and the **data** available for this training process?

Given the computation budget, what model size and data size shall we use?

Scaling law can answer these questions. In this lecture, we will consider the networks based on the transformer decoders

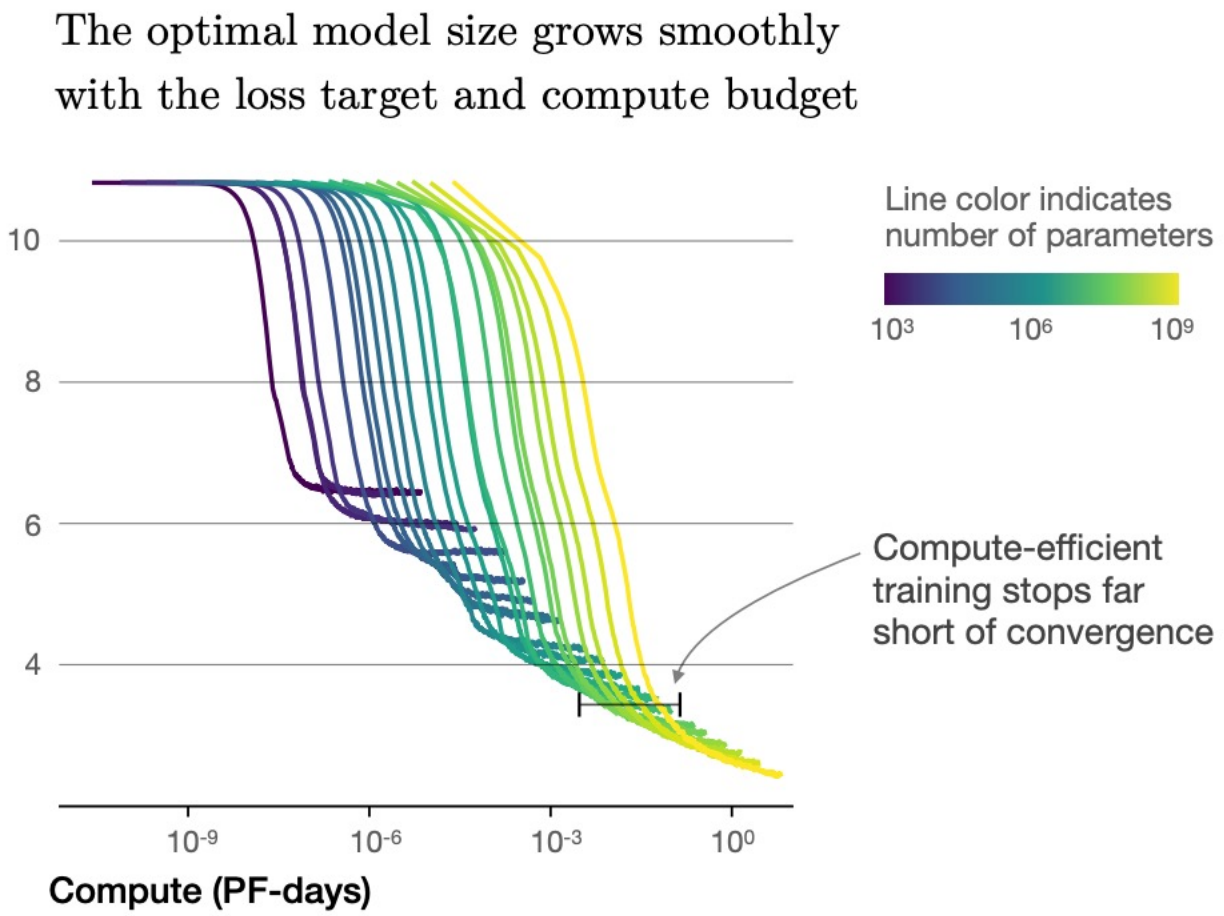
Performance depends strongly on model size and data



[Scaling Laws for Neural Language Models]

- Larger dataset leads to better performance
- Larger model leads to better performance
- Larger model is more data efficient
- This observation is the fundamental pillar for large language model

Performance depends strongly on computation budget



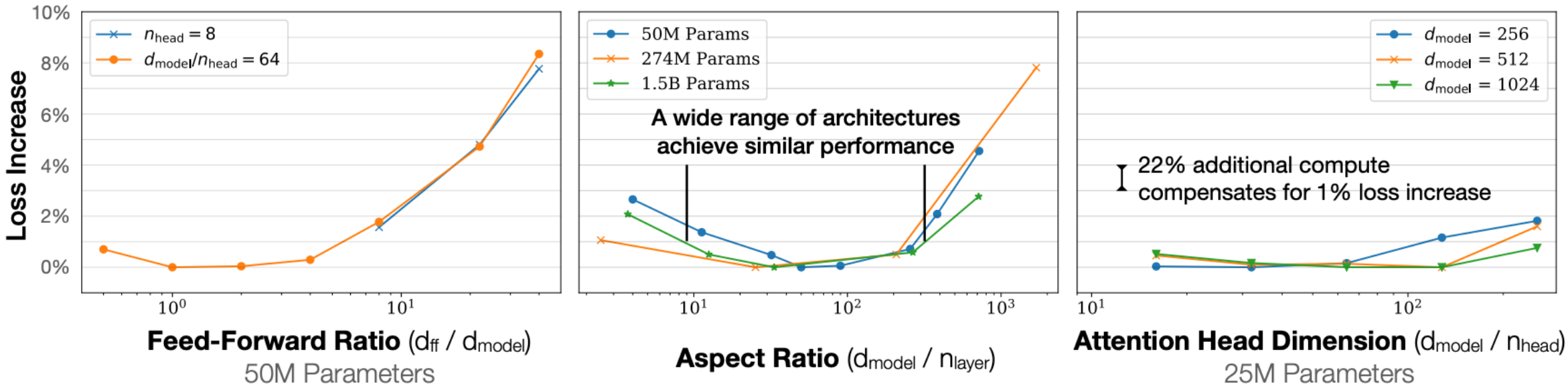
[Scaling Laws for Neural Language Models]

- More computation budget leads to better performance
- Given the computation budget, larger model leads to better performance

Performance depends weakly on model shapes

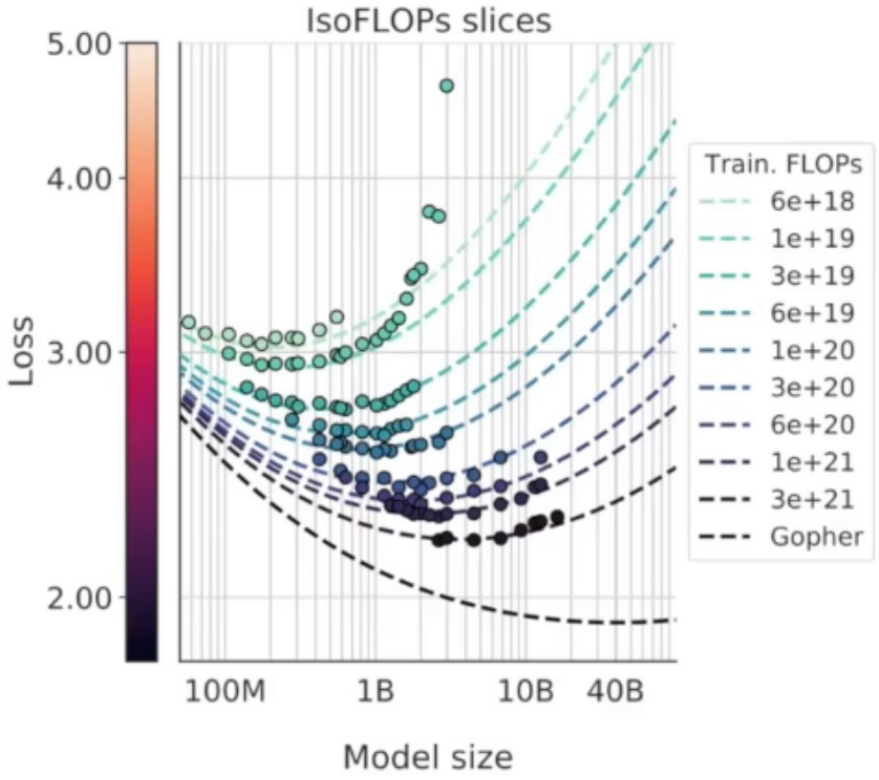
北京大学 PEKING UNIVERSITY

[Scaling Laws for Neural Language Models]



Performance depends strongly on model size and data

[Training Compute-Optimal Large Language Models]



Amazing representation power

Larger dataset + bigger model + longer training

=

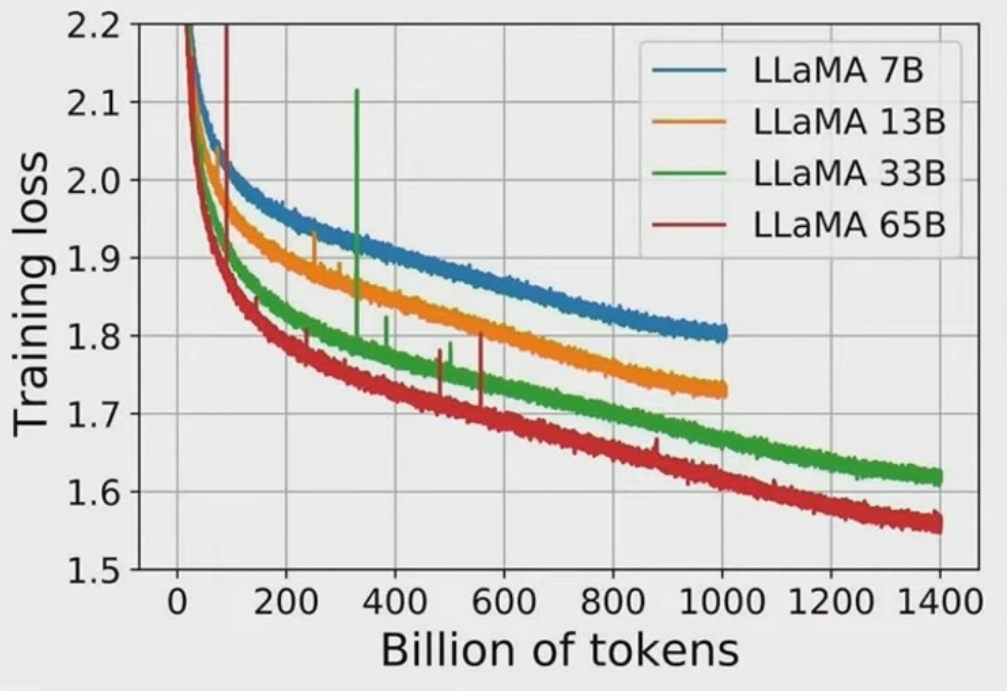
better prediction accuracy

A very straightforward way to achieve good LLM.

All you need is **MONEY**!

Performance depends strongly on model size and data

[LLaMA: Open and Efficient Foundation Language Models,2023]



According to this observation:

- LLM model gets increasingly bigger
- Dataset gets increasingly larger
- Training gets increasingly important

Smooth power law

N: the number of model parameters

D: the number of tokens

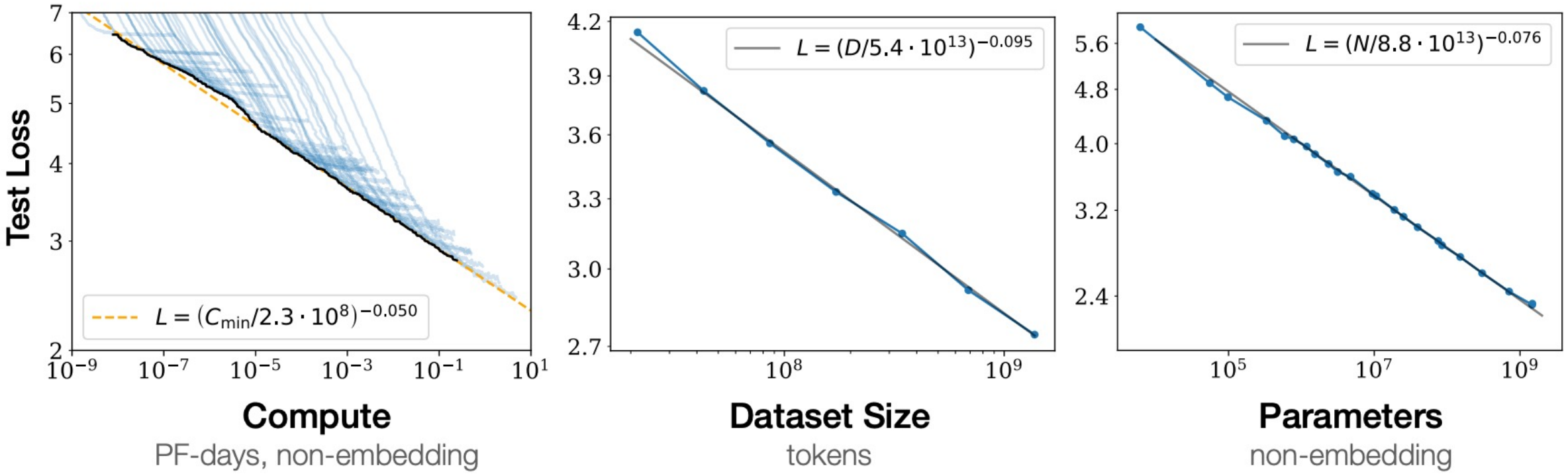
C: the computation budget

$C \approx 6\,N\,D$ (see slides on transformer computations)

Performance has a power-law relationship with each of the three scale factors N, D, C when not bottlenecked by the other two

Smooth power law

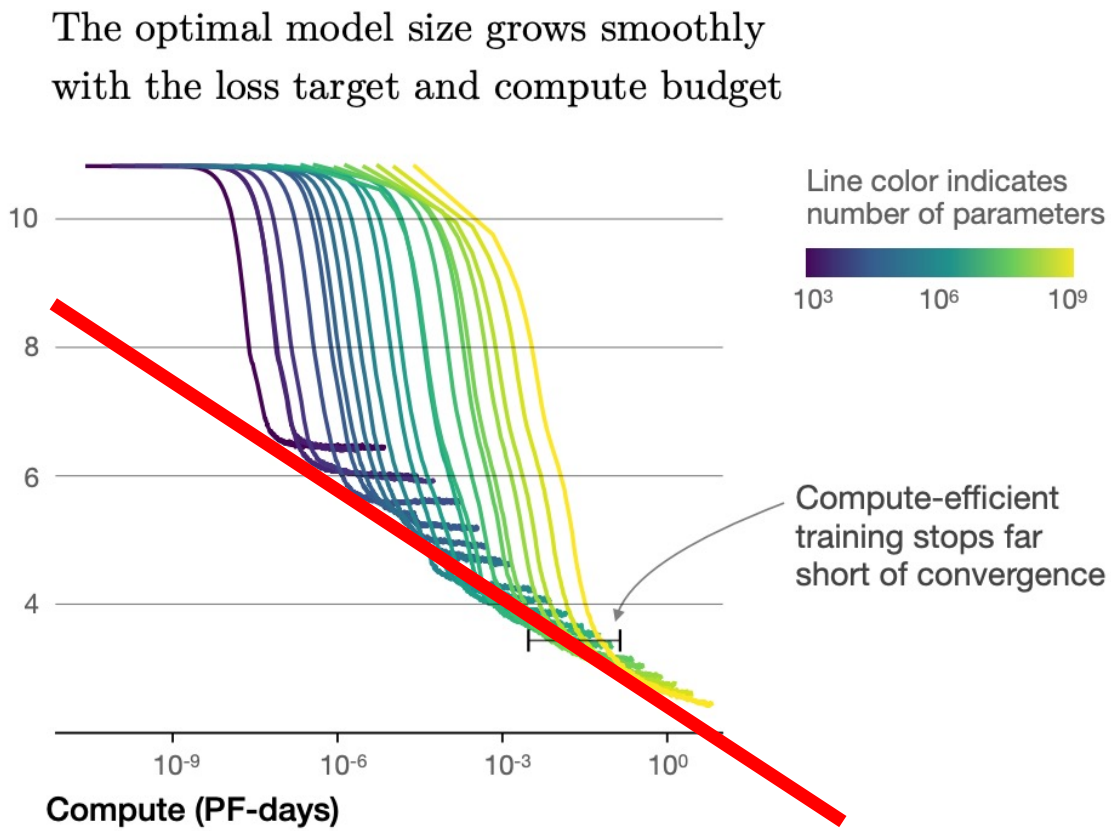
[Scaling Laws for Neural Language Models]



As data or parameters increase, the benefits they bring per unit becomes smaller

Compute-optimal scaling law

[Scaling Laws for Neural Language Models]



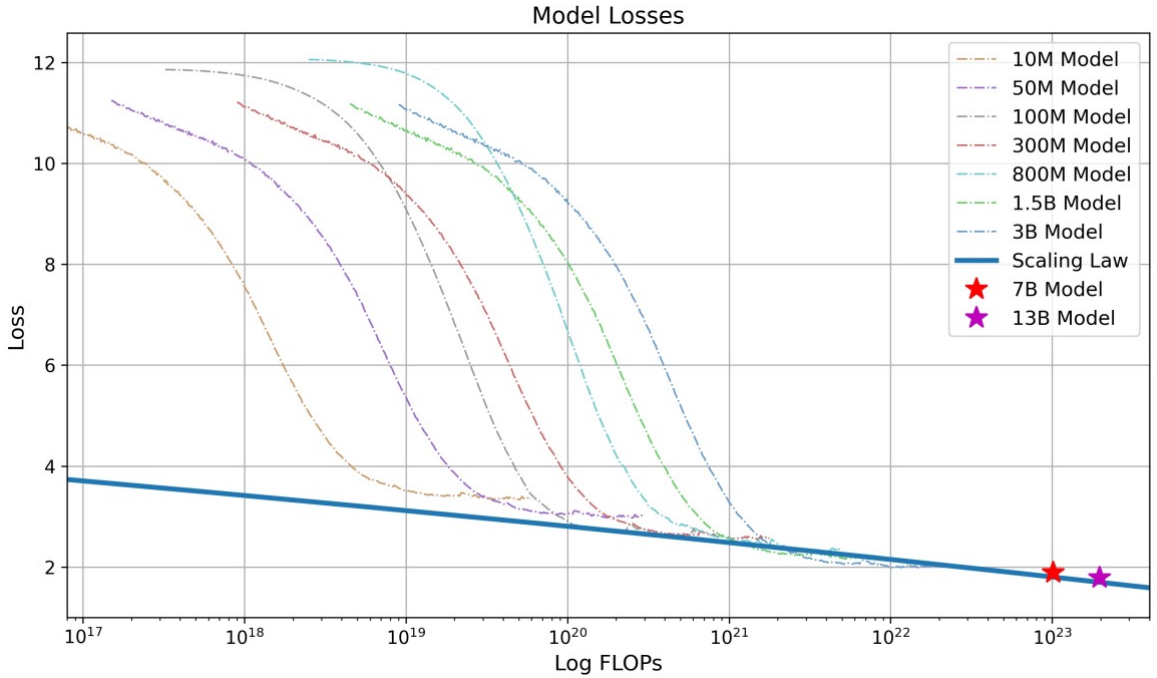
Compute-optimal pattern

- Given a computation budget, increasing the data size without increasing the model size will finally saturate the performance
- Model size and data size should be increased simultaneously
- Compute-optimal model size and data size follows **predictable** patterns

Compute-optimal scaling law

北京大学 PEKING UNIVERSITY

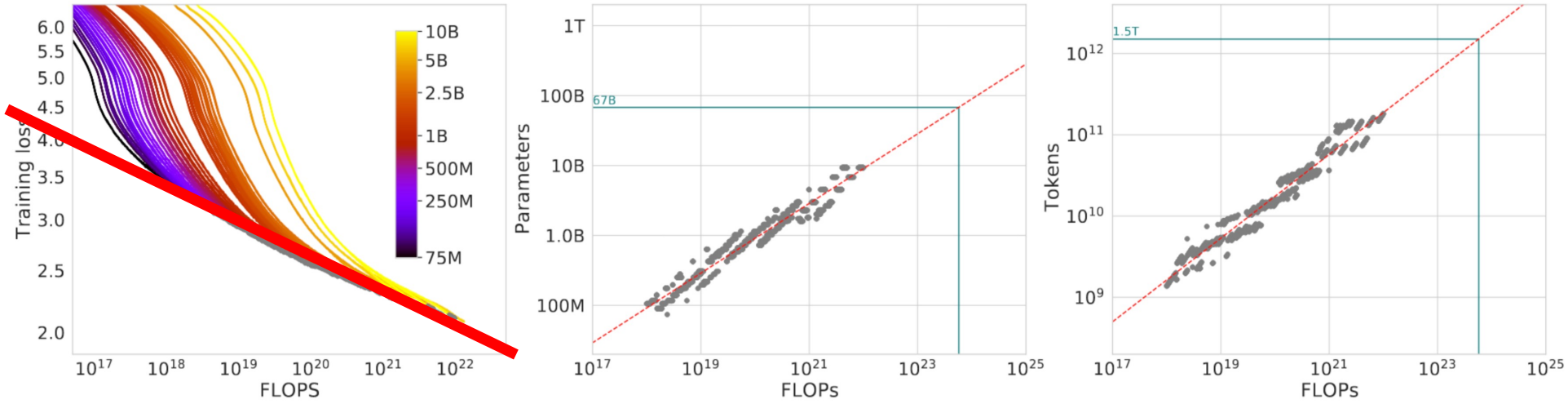
[Baichuan 2: Open Large-scale Language Models]



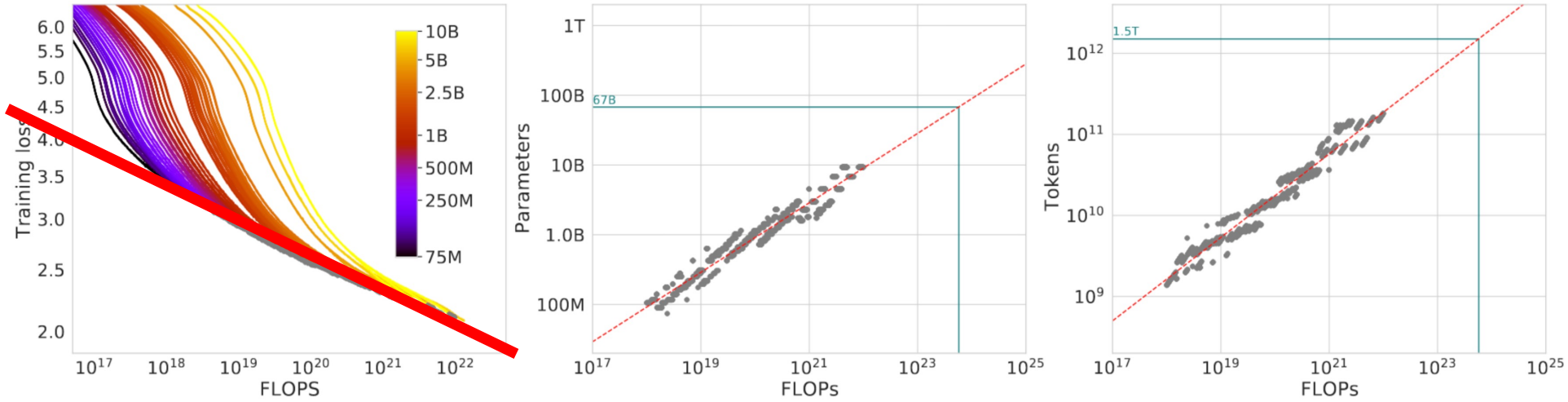
- Baichuan also endows with such a compute-optimal scaling law
- With such scaling law, we can predict the optimal model size and token number given the computation budget

How to achieve and use the compute-optimal scaling law?

- Prepare enough data (e.g., 1T)
- Train models of different size ranging from 1M ~ 1B
- Collect all convergence curves and draw the loss-computation plot
- Estimate the scaling law and use it to help construct your model and dataset

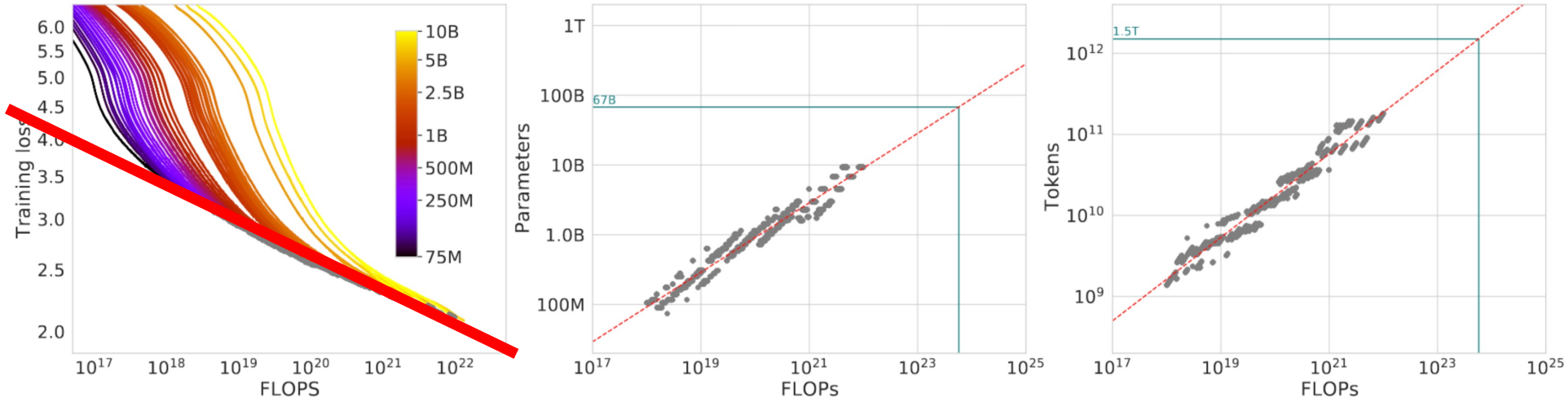


How to achieve and use the compute-optimal scaling law?



- According to the figure, it holds that $N_{opt} \propto C^a, D_{opt} \propto C^b$,
- Since C = 6 N D, we have a + b = 1
- OpenAI believes a = 0.73 and b = 0.27; while Google believes a = 0.5 and b = 0.5

How to achieve and use the compute-optimal scaling law?



- OpenAI believes a = 0.73 and b = 0.27; while Google believes a = 0.5 and b = 0.5
- Suppose the budget of computation increases by 10 times, OpenAI will increase the model by $10^{0.73} = 5.32$ times and increase the data size by $10^{0.27} = 1.86$
- In contrast, Google will increase both the model and data by $10^{0.5} = 3.16$ times

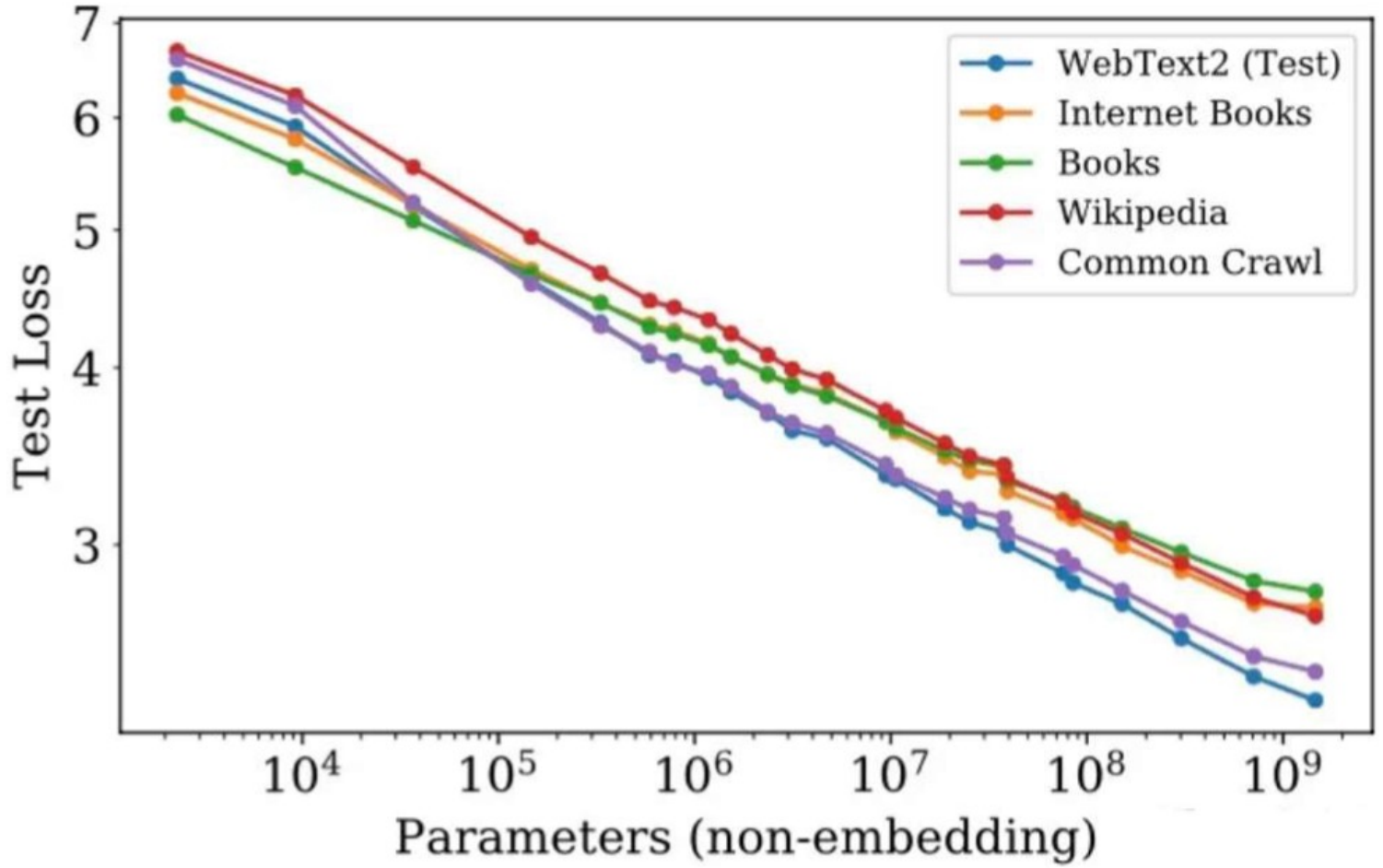
How to achieve and use the compute-optimal scaling law?

FLOPs	Total Parameters	Loss	Tokens	Chinchilla Optimal Params	PaLM 2 Optimal Params
1×10^{22}	3.86×10^9	2.488	4.32×10^{11}	$\sim$ 10B	10.7B
	7.08×10^9	2.404	2.36×10^{11}		
	9.50×10^9	**2.400**	1.75×10^{11}		
	1.61×10^{10}	2.406	1.04×10^{11}		
1×10^{21}	1.23×10^9	2.716	1.36×10^{11}	$\sim$ 3B	3.35B
	3.01×10^9	2.642	5.53×10^{10}		
	3.86×10^9	**2.627**	4.32×10^{10}		
	9.50×10^9	2.669	1.75×10^{10}		
1×10^{20}	7.41×10^8	2.949	2.25×10^{10}	$\sim$ 1B	1.04B
	1.46×10^9	**2.896**	1.14×10^{10}		
	1.98×10^9	2.908	8.43×10^9		
	4.44×10^9	2.977	3.75×10^9		

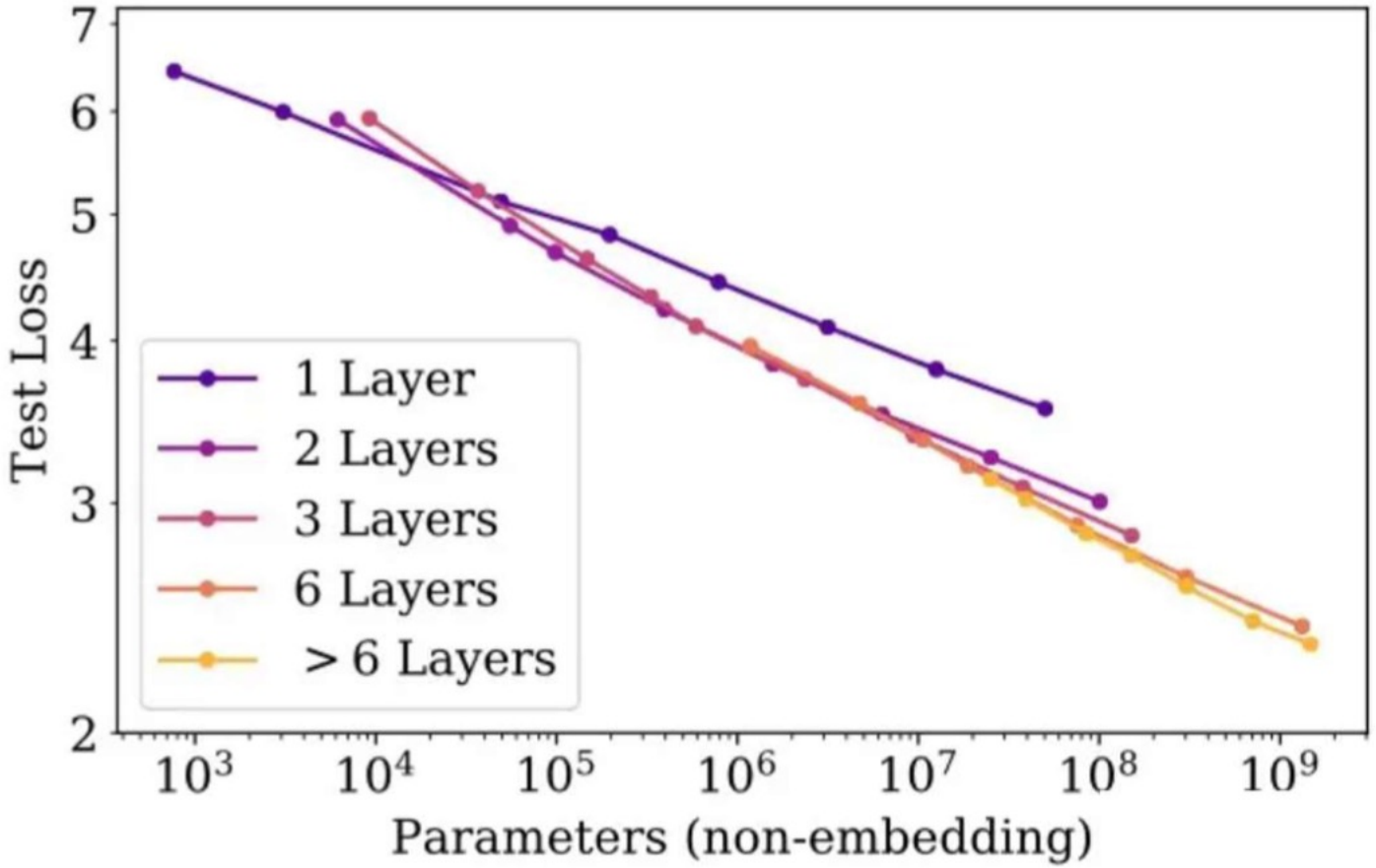
[PaLM 2 Technical Report]

In practice, you can compute your own a and b

Scaling law varies a little bit on different dataset

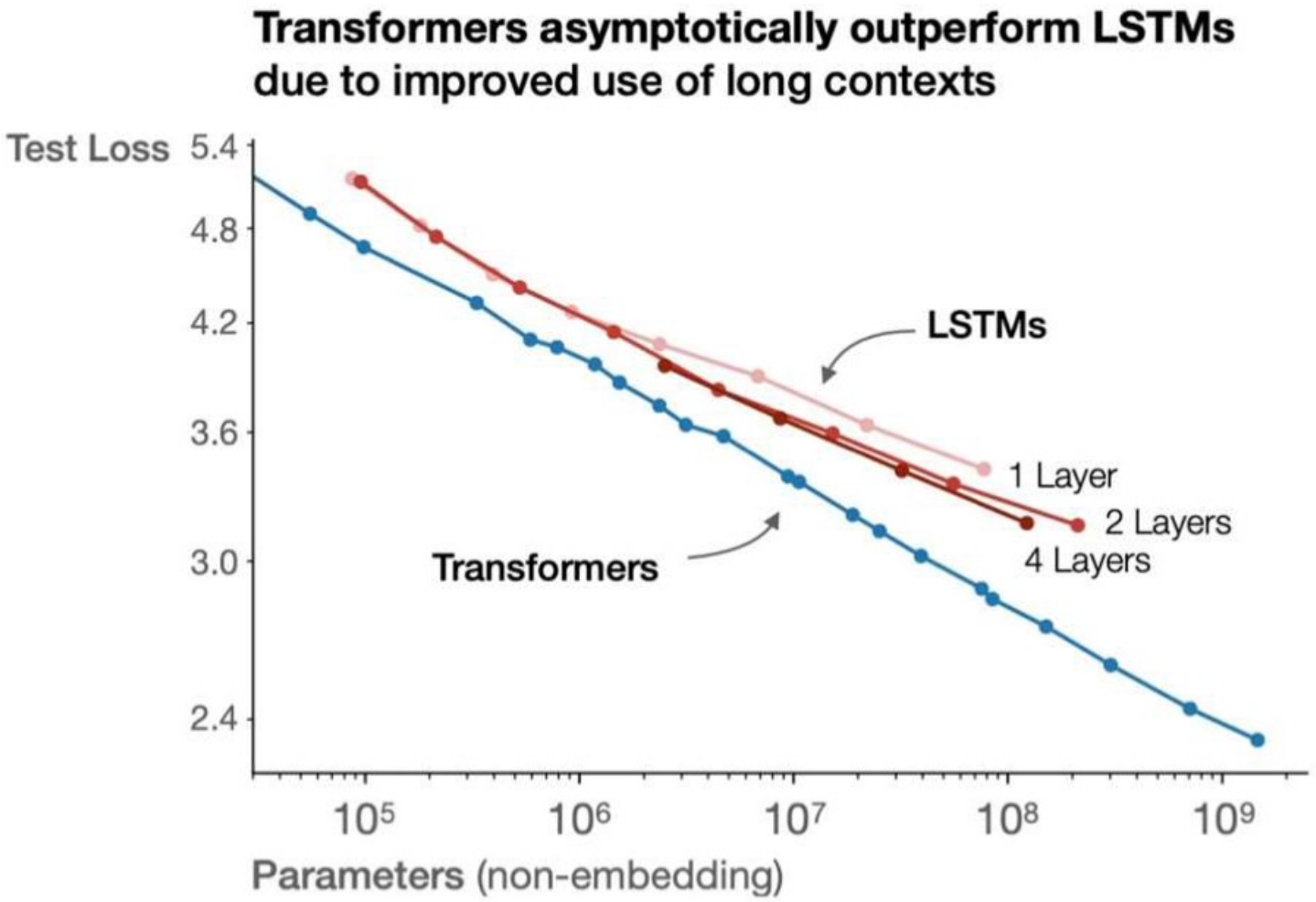


Scaling law varies a little bit on different shapes



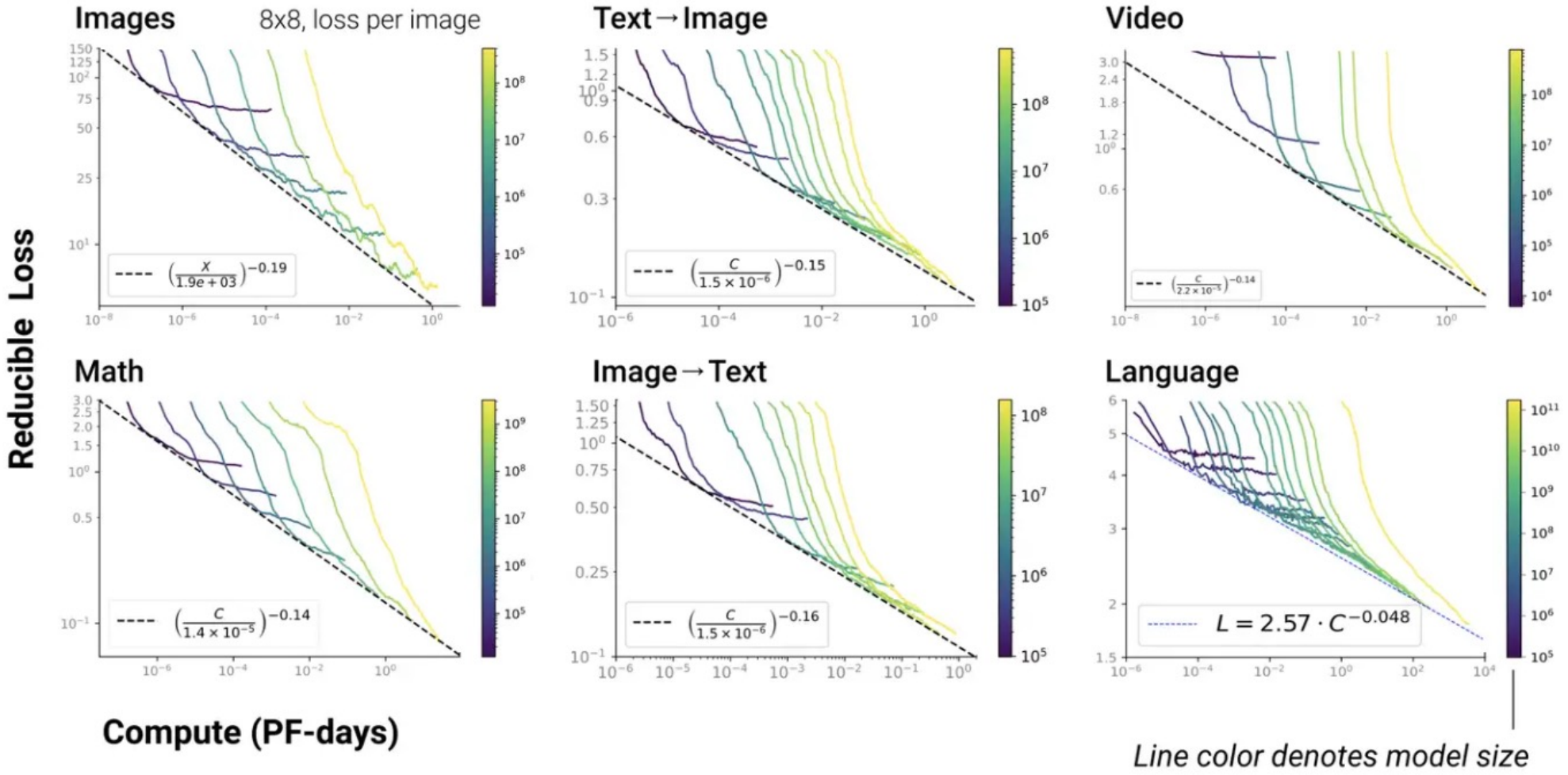
Deep is better

Scaling law varies on different models



Transformer is better

Scaling law on multi-modal models



[Scaling Laws for Autoregressive Generative Modeling]

Shall we always pursue compute-optimal?

- While compute-optimal scaling law saves computation during the training phrase, it pays the cost of a larger model
- It is expensive to use a larger model during the inference stage
- Meta believes that it is worth paying extra computations to train a small model, in order to save computations in inference
- In the plot, Meta prefers to use more data and computing resources to train LLaMA 13B rather than use less data to train LLaMA 65B