LLM 101

一起入门大语言模型

https://llm101.top

哔哩哔哩@一万篇论文笔记

一万篇论文笔记

微信扫描二维码，关注我的公众号

2025.02.02

LLM Pre-training and Beyond

- GPT-1 && GPT-2
  - NLP中的预训练-微调范式: CoVe、ELMo、ULMFiT、GPT-1、BERT
  - GPT-1 && GPT-2: Transformer LM + Large scale pre-training ==> zero-shot
  - 编程实践：阅读gpt-1/gpt-2代码；训练124M GPT-2 **llm.c** **Modded-NanoGPT**
- (Train-time Compute) Scaling Laws for LM, Empirically
- GPT-3 and Beyond
  - 涌现、幻觉、位置编码、合成数据、提示工程、SLMs ...

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化
  - 预测不同规模下模型的预期效果
  - 在有限的计算资源(FLOPs)下，如何合理分配模型和训练集大小
- Train-time Compute Scaling Laws, empirically

Scaling Laws

vs Test-time compute scaling laws

LLM reasoning

(Train-time Compute) Scaling Laws for LM, Empirically

- GPT-1(117M/0.1B) → GPT-2(1.5B) —Megatron-LM(8.3B)/T5(11B), Turing-NLG(17B)→ GPT-3(175B) **~$4.3M**

AI index report 2024

(Train-time Compute) Scaling Laws for LM, Empirically

- GPT-1(117M/0.1B) → GPT-2(1.5B) —Megatron-LM(8.3B)/T5(11B), Turing-NLG(17B)→ GPT-3(175B) **~$4.3M** AI index report 2024
- Scaling Laws

(Train-time Compute) Scaling Laws for LM, Empirically

- GPT-1(117M/0.1B) → GPT-2(1.5B) → GPT-3(175B) ~$4.3M

  Megatron-LM(8.3B)/T5(11B)

  Turing-NLG(17B)

  AI index report 2024

- Scaling Laws
- RLHF

(Train-time Compute) **Scaling** Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws

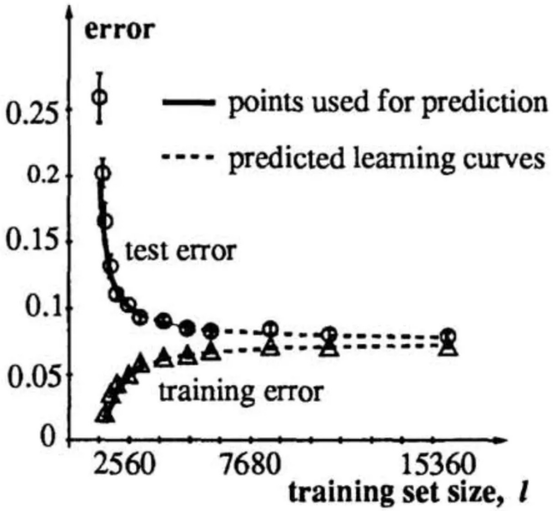
(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化
  - 预测不同(数据/模型/...)规模下模型的预期效果
  - 在有限的计算资源(FLOPs)下，如何合理分配模型和训练集大小

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化
  - 预测不同规模下模型的预期效果
  - 在有限的计算资源(FLOPs)下，如何合理分配模型和训练集大小

模型选择？



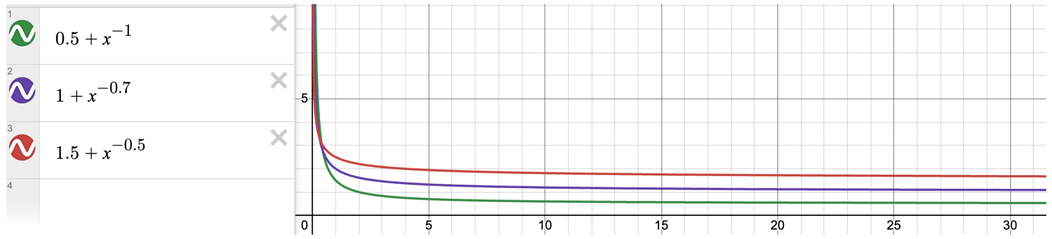
Learning curves: Asymptotic values and rate of convergence, 1993

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化

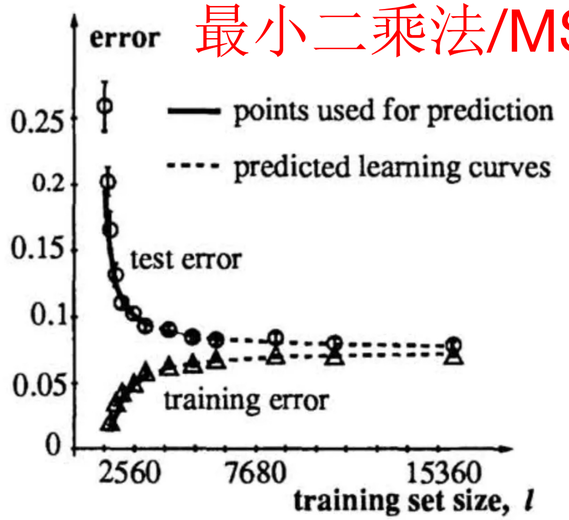
模型选择？

最小二乘法/MSE



幂函数(power func)

$$\varepsilon_{\text{test}} = a + \frac{b}{l^{\alpha}} \quad \text{and} \quad \varepsilon_{\text{train}} = a - \frac{c}{l^{\beta}}$$



Learning curves: Asymptotic values and rate of convergence, 1993

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化

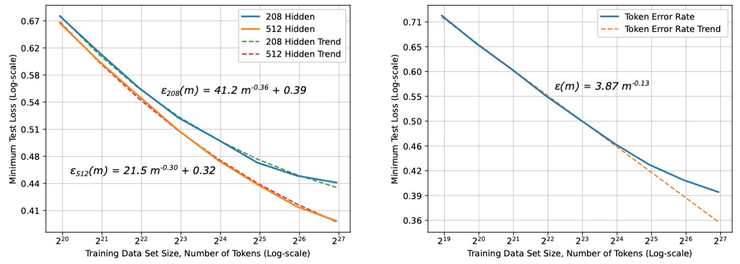


Figure 1: Neural machine translation learning curves. Left: the learning curves for separate models follow $\varepsilon(m) = \alpha m^{\beta_g} + \gamma$. Right: composite learning curve of best-fit model at each data set size.

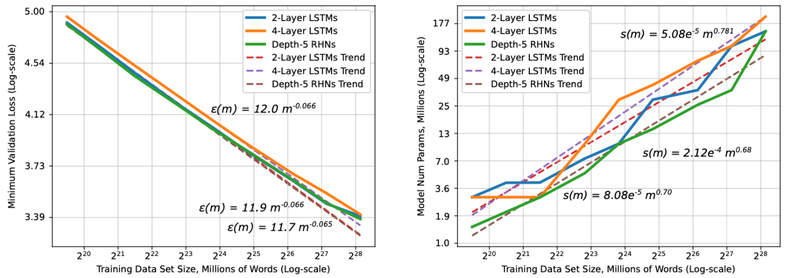


Figure 2: Learning curve and model size results and trends for word language models.

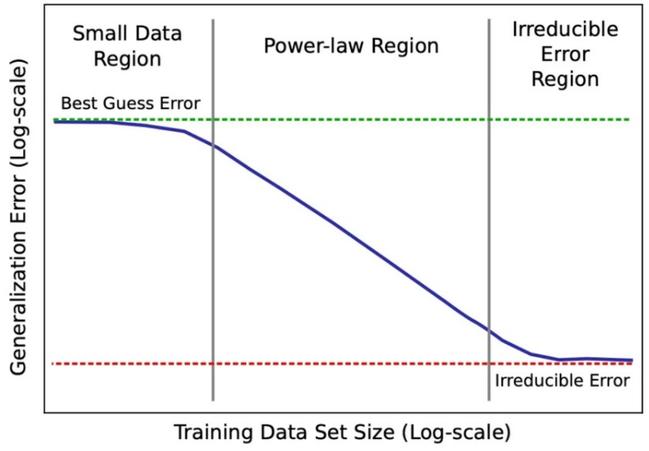


Figure 6: Sketch of power-law learning curves

Deep Learning Scaling is Predictable, Empirically

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化

DEEP LEARNING SCALING IS PREDICTABLE, EMPIRICALLY

SCALING LAWS FOR FINE-GRAINED MIXTURE OF EXPERTS

Scaling Laws For Dense Retrieval

SCALING LAWS FOR PREDICTING DOWNSTREAM PERFORMANCE IN LLMS

Scaling Laws for Neural Language Models

Scaling Laws for Reward Model Overoptimization

Scaling Laws for Linear Complexity Language Models

Are More LM Calls All You Need?
Towards the Scaling Properties of Compound AI Systems

Scaling Scaling Laws with Board Games

.....

(Train-time Compute) Scaling Laws for LM, Empirically

- Scaling/Scale up: bigger 模型 + bigger 训练集 + longer 训练时间 ≈ better 效果
- Laws: 通过研究/实验总结出来的规律/定律/准则
  - Scientific(科学) laws: 圆的周长和半径 $C = 2\pi r$
  - Empirical(经验) laws
- Empirical Scaling Laws: 对scale up进行经验量化
  - 预测不同规模下模型的预期效果
  - 在有限的计算资源(FLOPs)下，如何合理分配模型和训练集大小
- Train-time Compute Scaling Laws, empirically

vs Test-time compute scaling laws

LLM reasoning

(Train-time) Scaling Laws for LM, Empirically

- Kaplan Scaling Laws — Scaling Laws for Neural Language Models
- Chinchilla Scaling Laws — Training Compute-Optimal Large Language Models

Kaplan Scaling Laws

- (GPT) LM在测试集的**cross-entropy loss**和模型大小、训练集大小、训练计算量呈现**幂律**关系
  - 训练前预测大规模模型的测试集CE loss
  - 根据给定的计算量预算，合理分配模型大小和训练集大小

大小很重要

模型结构不那么重要

- **更大的模型具有更高的样本效率**，计算效率最高的训练方法是用相对适量的数据训练非常大的模型，并在还未收敛时就停止训练

大模型 + 相对较小的训练集

Big models may be more important than big data

Kaplan Scaling Laws

- (GPT) LM在测试集的**cross-entropy loss**和模型大小、训练集大小、训练计算量呈现**幂律**关系
- 模型大小：GPT模型参数量
- 训练集大小：token数量
- 计算量：FLOPs

GPT 参数量

这样，我们就得到了Transformer Encoder 中**一层的参数量(MHSA + FFNN + 2 * LayerNorm)**:

$$4d_{model}^2 + 8d_{model}^2 + 5d_{model} + 2 \cdot 2d_{model}$$

$$= 12d_{model}^2 + 9d_{model}$$

同样，可以得到 Transformer Decoder 中**一层的参数量(2*MHSA + FFNN + 3 * LayerNorm)**：

$$2 \cdot 4d_{model}^2 + 8d_{model}^2 + 5d_{model} + 3 \cdot 2d_{model}$$

$$= 16d_{model}^2 + 11d_{model}$$

如果是 6 层 Encoder + 6 层 Decoder 的 Transformer 模型，只需要✖6 就行了，别忘了 Embedding 层的Vd_{model}。

Transformer 101 (一): 参数量和时间空间复杂度

GPT 参数量

这样，我们就得到了Transformer Encoder 中**一层的参数量(MHSA + FFNN + 2 * LayerNorm)**:

$$4d_{model}^2 + 8d_{model}^2 + 5d_{model} + 2 \cdot 2d_{model}$$

$$= 12d_{model}^2 + 9d_{model}$$

同样，可以得到 Transformer Decoder 中**一层的参数量(2*MHSA + FFNN + 3 * LayerNorm)**：

$$2 \cdot 4d_{model}^2 + 8d_{model}^2 + 5d_{model} + 3 \cdot 2d_{model}$$

$$= 16d_{model}^2 + 11d_{model}$$

如果是 6 层 Encoder + 6 层 Decoder 的 Transformer 模型，只需要✖6 就行了，别忘了 Embedding 层的Vd_{model}。

Transformer 101 (一): 参数量和时间空间复杂度

We use N to denote the model size, which we define as the number of *non-embedding* parameters

$$N \approx 2d_{\text{model}}n_{\text{layer}}\left(2d_{\text{attn}} + d_{\text{ff}}\right)$$
$$= 12n_{\text{layer}}d_{\text{model}}^2 \quad \text{with the standard} \quad d_{\text{attn}} = d_{\text{ff}}/4 = d_{\text{model}}$$

Operation	**Parameters**
Embed	$\left(n_{\text{vocab}} + n_{\text{ctx}}\right) d_{\text{model}}$
Attention: QKV	$n_{\text{layer}}d_{\text{model}}3d_{\text{attn}}$
Attention: Mask	—
Attention: Project	$n_{\text{layer}}d_{\text{attn}}d_{\text{model}}$
Feedforward	$n_{\text{layer}}2d_{\text{model}}d_{\text{ff}}$
De-embed	—
Total (Non-Embedding)	$N = 2d_{\text{model}}n_{\text{layer}}\left(2d_{\text{attn}} + d_{\text{ff}}\right)$

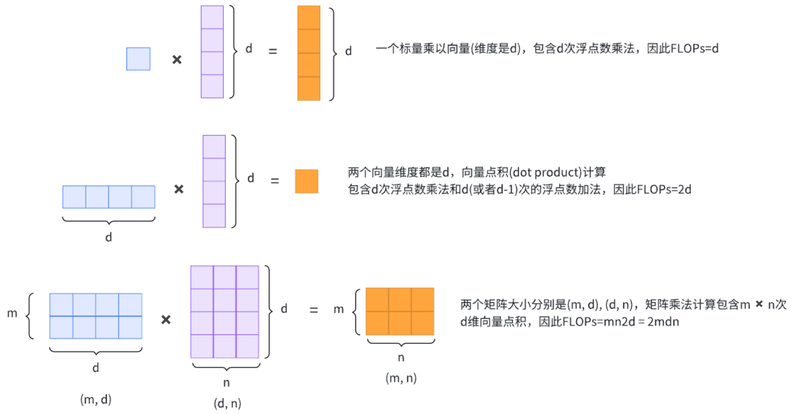
GPT FLOPs per token

- FLOPs: floating point operations
- FLOPS: floating point operations per second
- K: 10^3, M: 10^6, G: 10^9, T: 10^{12}, P: 10^{15}

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.

FLOPs (floating point operations)定义：两个浮点数之间的一次加法/乘法/减法/除法是1 FLOP



Per token FLOPs in forward pass

Colab: Transformer 参数量和FLOPs

GPT FLOPs per token

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.

forward pass

$$w@x = z$$

$$(m,n)@(n,p) => (m,p), FLOPs = 2mnp$$

z作为下一层的输入x

backward pass

$$\Delta w = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w} = (m,p)@(n,p) => (m,n), FLOPs = 2mpn$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial x} = (m,p)@(m,n) => (n,p), FLOPs = 2mpn$$

计算$\frac{\partial L}{\partial x}$是为了计算下一层(backward pass)的$\Delta w$

Non-embedding

$$N \approx 12 n_{layer} d_{model}^2$$

$$C \approx 6N$$

Kaplan Scaling Laws

- N固定，调整n_{layer}、head、d_{ff}
- GPT shape不那么重要

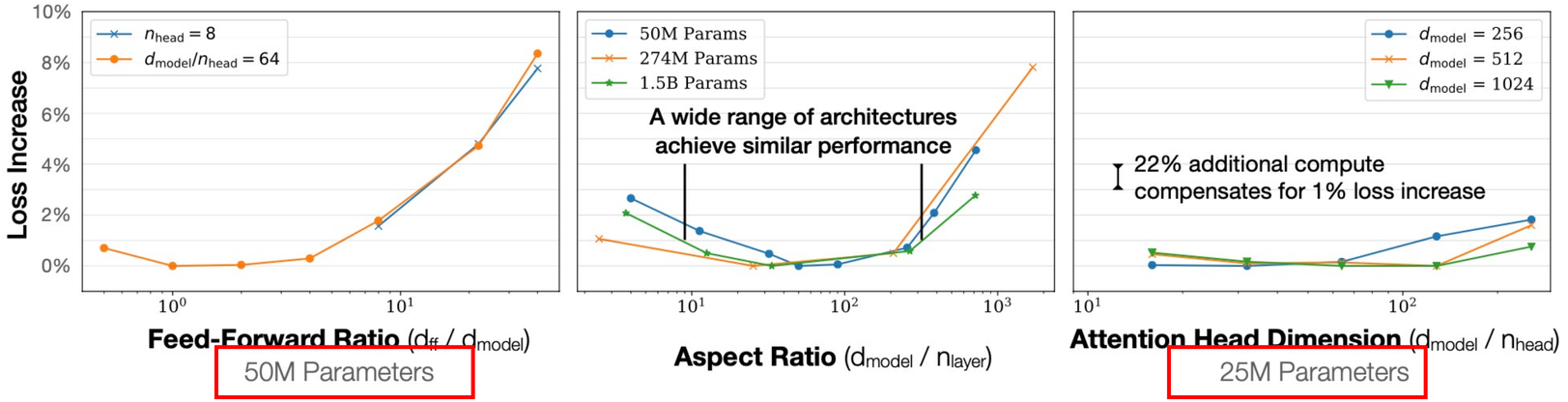
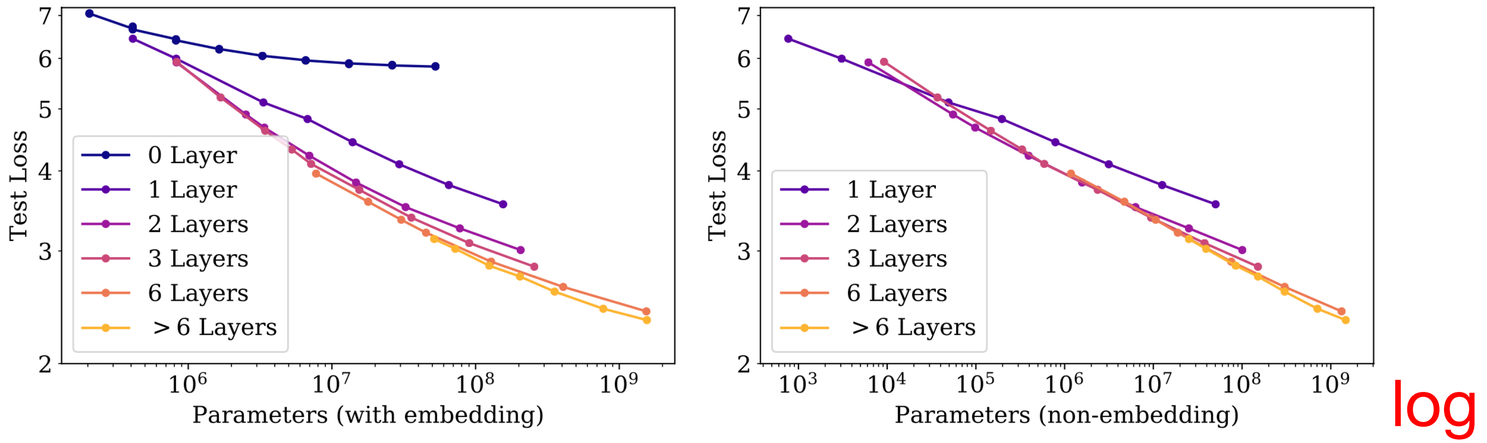


Figure 5 Performance depends very mildly on model shape when the total number of non-embedding parameters N is held fixed. The loss varies only a few percent over a wide range of shapes. Small differences in parameter counts are compensated for by using the fit to $L(N)$ as a baseline. Aspect ratio in particular can vary by a factor of 40 while only slightly impacting performance; an $(n_{\text{layer}}, d_{\text{model}}) = (6, 4288)$ reaches a loss within 3% of the $(48, 1600)$ model used in [RWC+19].

Kaplan Scaling Laws

- 参数量和测试集CE Loss: $L(N) \approx (\frac{N_c}{N})^{\alpha_N}$
- $\alpha_N \sim 0.076$
- $N_c \sim 8.8 \times 10^{13}$ (non-embedding)

log



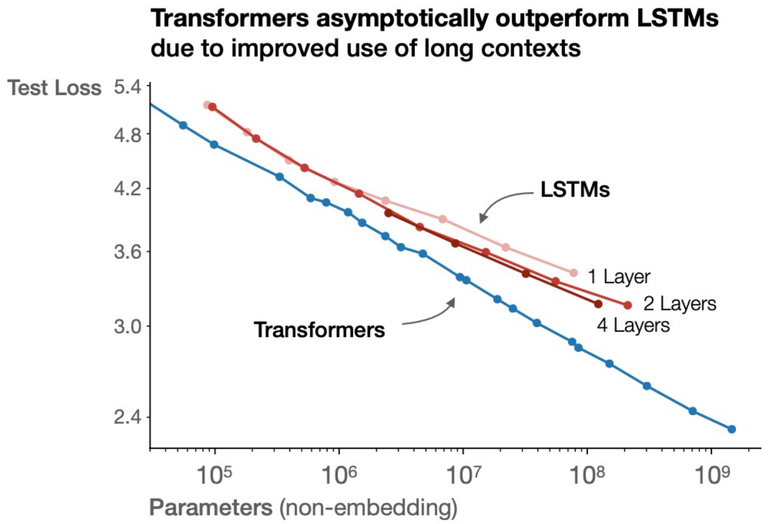
log

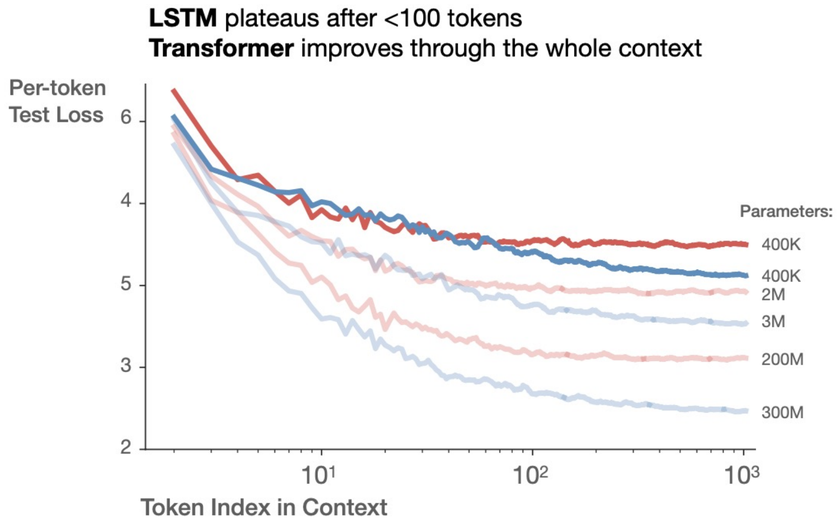
Figure 6 **Left:** When we include embedding parameters, performance appears to depend strongly on the number of layers in addition to the number of parameters. **Right:** When we exclude embedding parameters, the performance of models with different depths converge to a single trend. Only models with fewer than 2 layers or with extreme depth-to-width ratios deviate significantly from the trend.

Kaplan Scaling Laws

- GPT vs LSTM

log





log

Kaplan Scaling Laws

- 训练集大小和测试集CE Loss: $L(\mathrm{D}) \approx \left(\frac{D_c}{D}\right)^{\alpha_D}$
- $\alpha_D \sim 0.095$
- $D_c \sim 5.4 \times 10^{13}$ (tokens)
- 计算量和测试集CE Loss

$$L(C_{\min}) = \left(C_{\mathrm{c}}^{\min}/C_{\min}\right)^{\alpha_C^{\min}}$$

$\alpha_C^{\min} \sim 0.050, \quad C_{\mathrm{c}}^{\min} \sim 3.1 \times 10^8$ (PF-days)

C ≈6NBS

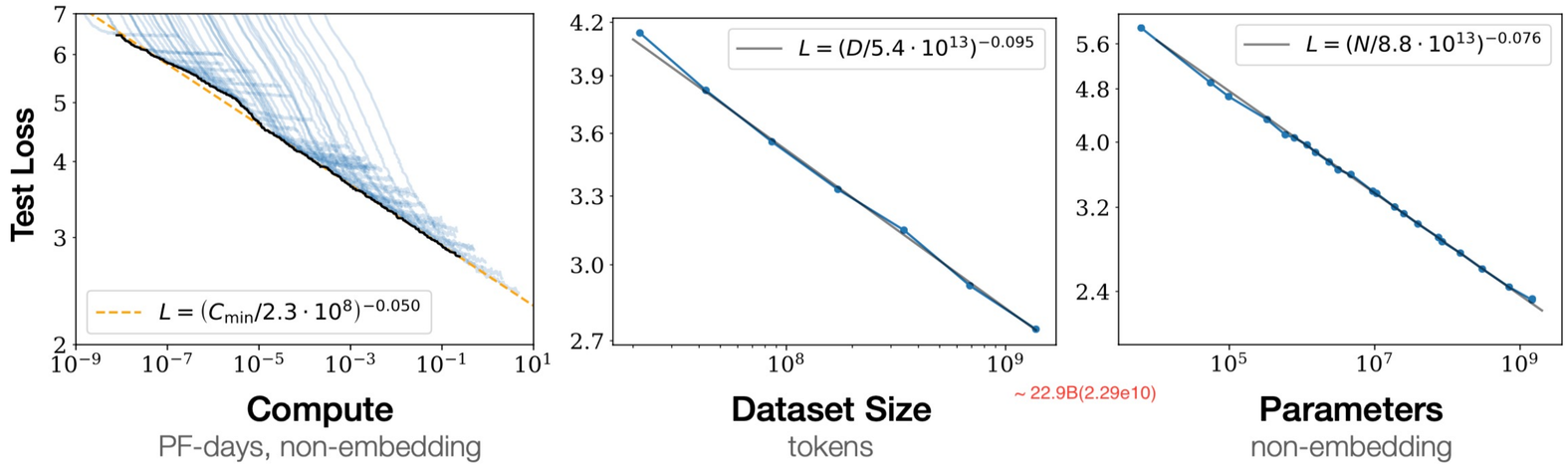


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

Kaplan Scaling Laws

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

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 $L(N, D)$

$$D \propto N^{0.74}$$

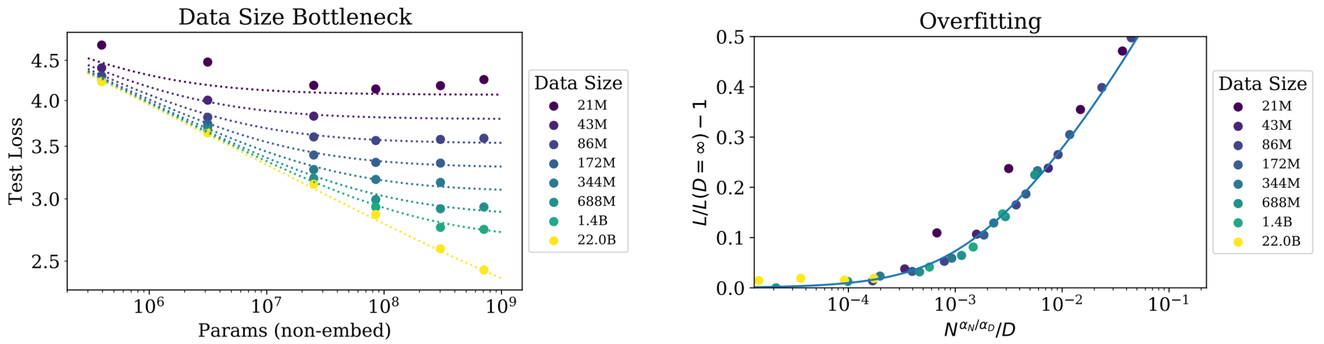
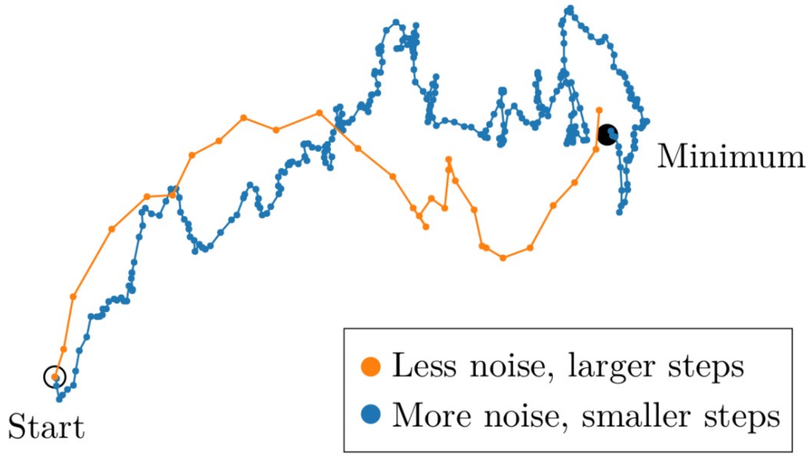


Figure 9 The early-stopped test loss $L(N, D)$ depends predictably on the dataset size D and model size N according to Equation (1.5). **Left**: For large D, performance is a straight power law in N. For a smaller fixed D, performance stops improving as N increases and the model begins to overfit. (The reverse is also true, see Figure 4.) **Right**: The extent of overfitting depends predominantly on the ratio $N^{\frac{\alpha_N}{\alpha_D}}/D$, as predicted in equation (4.3). The line is our fit to that equation.

Kaplan Scaling Laws

- Critical batch size

An Empirical Model of Large-Batch Training



$$\mathcal{B}_{\text{simple}} = \frac{\text{tr}(\Sigma)}{|G|^2}, \qquad \left(\frac{S}{S_{\min}} - 1\right)\left(\frac{E}{E_{\min}} - 1\right) = 1$$

$$B_{\text{crit}}(L) \equiv \frac{E_{\min}}{S_{\min}}$$

$$B_{\text{crit}}(L) \approx \frac{B_*}{L^{1/\alpha_B}}$$

$$C_{\min} \equiv 6NB_{\text{crit}}S_{\cdot}$$

Compute-Efficient Value	Power Law	Scale
$N_{\text{opt}} = N_e \cdot C_{\min}^{p_N}$	$p_N = 0.73$	$N_e = 1.3 \cdot 10^9$ params
$B \ll B_{\text{crit}} = \frac{B_*}{L^{1/\alpha_B}} = B_e C_{\min}^{p_B}$	$p_B = 0.24$	$B_e = 2.0 \cdot 10^6$ tokens
$S_{\min} = S_e \cdot C_{\min}^{p_S}$ (lower bound)	$p_S = 0.03$	$S_e = 5.4 \cdot 10^3$ steps
$D_{\text{opt}} = D_e \cdot C_{\min}^{p_D}$ (1 epoch)	$p_D = 0.27$	$D_e = 2 \cdot 10^{10}$ tokens

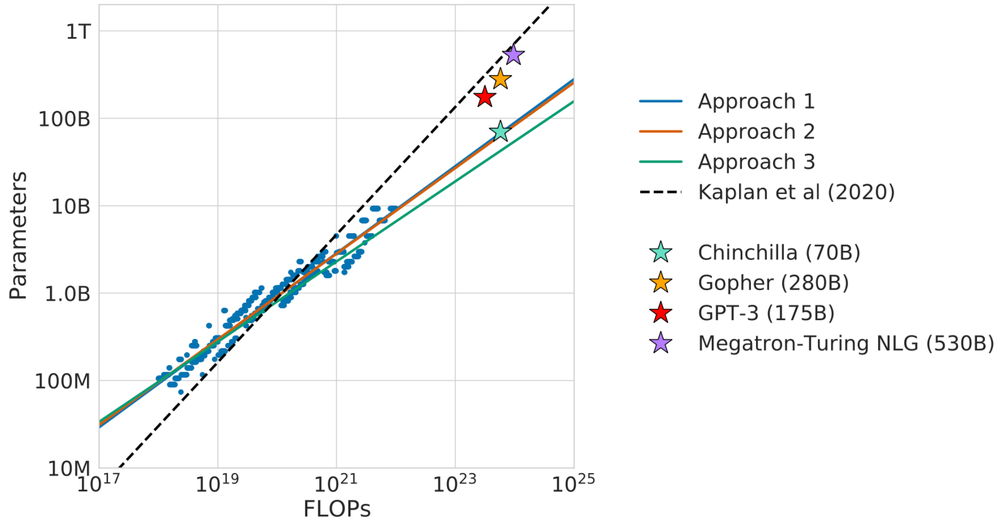
Chinchilla Scaling Laws

- 模型大小和训练集大小同等重要

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \text{FLOPs}(N,D)=C}{\text{argmin}} L(N, D)$$

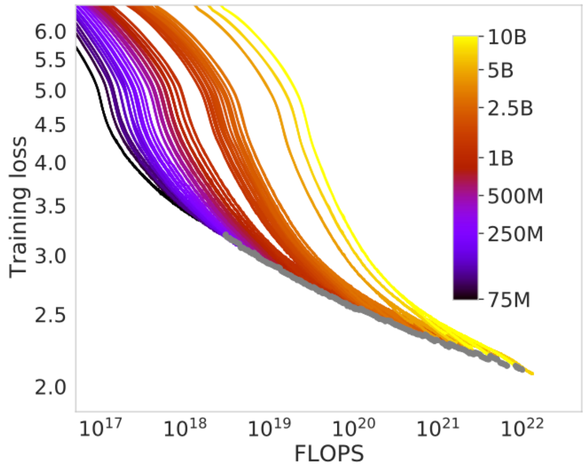
N和C都包含embedding

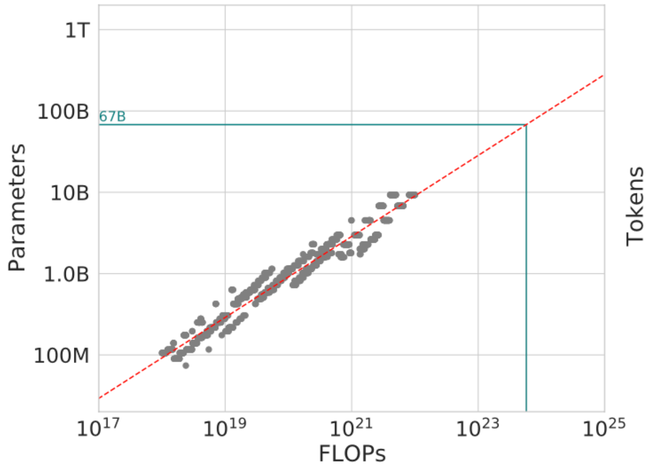
L是训练集损失

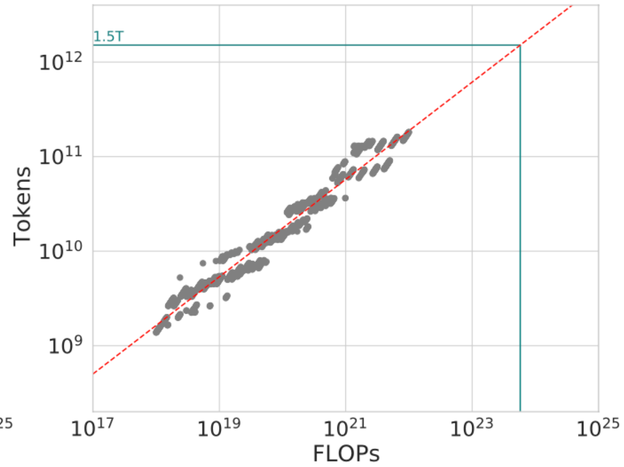


Chinchilla Scaling Laws

- 方法1：Fix model sizes and vary number of training tokens





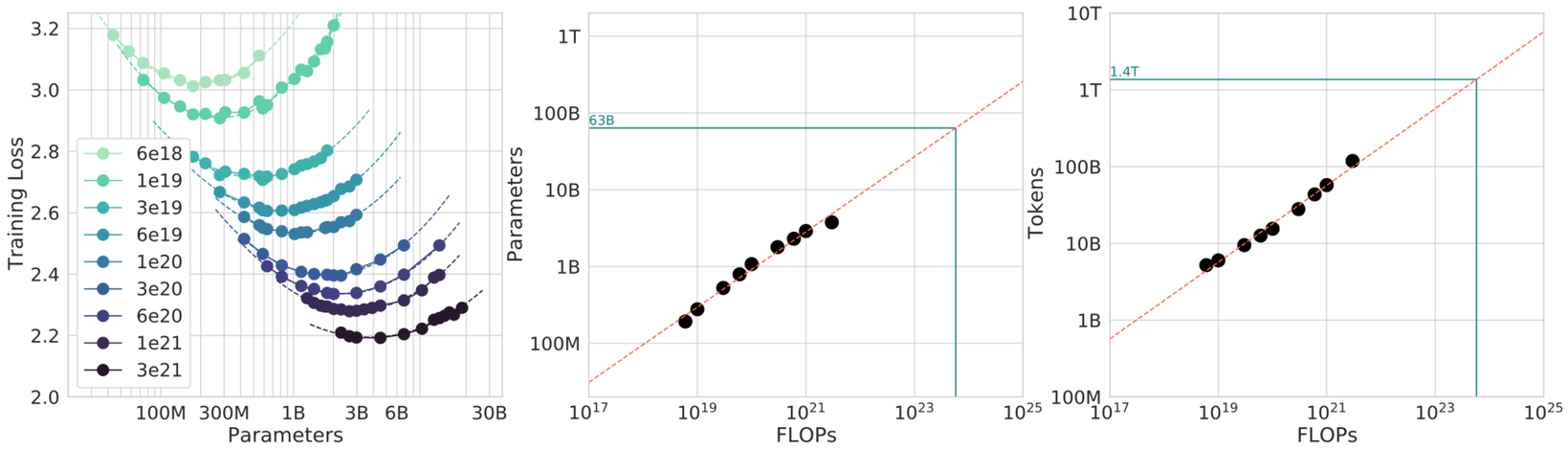


$N_{opt} \propto C^a$ and $D_{opt} \propto C^b$

a=b=0.5

Chinchilla Scaling Laws

- 方法2：Fix FLOPs and vary model size (IsoFLOP profiles)



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

a=0.49

b=0.51

Chinchilla Scaling Laws

- 方法3：Fitting a parametric loss function

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

$$\min_{A,B,E,\alpha,\beta} \sum_{\text{Runs } i} \text{Huber}_{\delta}\left(\log \hat{L}(N_i, D_i) - \log L_i\right)$$

约束条件

$$\widehat{\text{FLOPs}}(N, D) \approx 6ND$$

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

a=0.46

b=0.55

Chinchilla Scaling Laws

- 方法1：Fix model sizes and vary number of training tokens
- 方法2：Fix FLOPs and vary model size (IsoFLOP profiles)
- 方法3：Fitting a parametric loss function

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

Scaling Laws

- Scaling Laws is everywhere
- 考虑Vocab size、data repeat、data complexity、new model arch …

colab scaling laws mnist