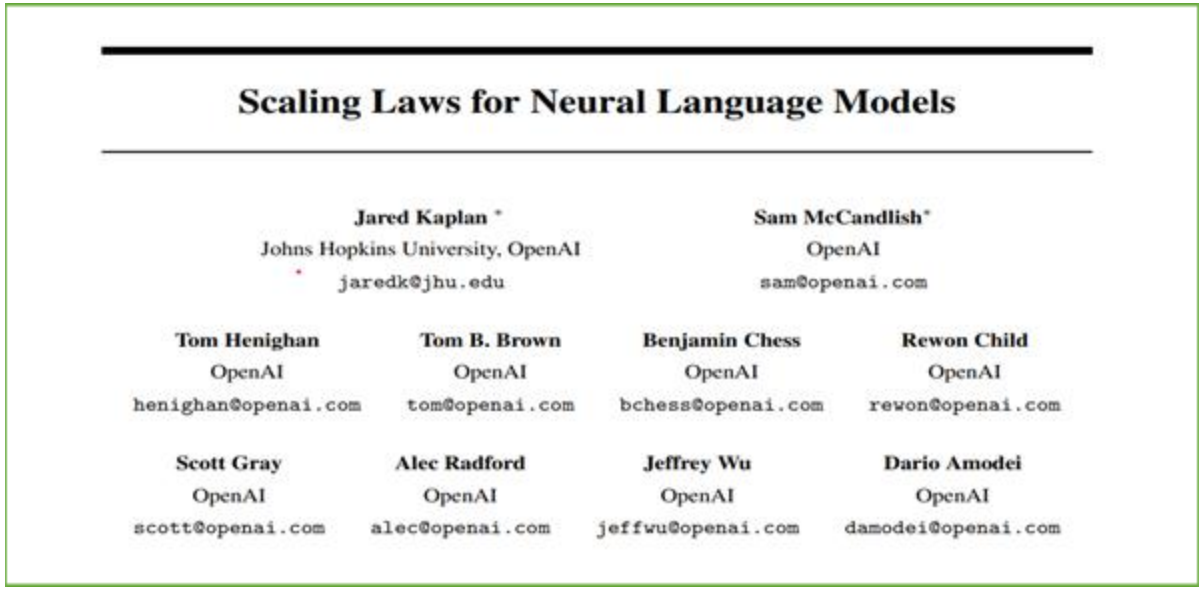
ECE 696B: Spring 2025
Trustworthy Machine Learning

Scaling Laws for Neural Language Models

Luke Dagnillo

Outline



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

- Investigates how the performance of language models depends on their size, training dataset size, and computational resources
- Cited 2,654
  (as of Jan 29, 2025)

Related work

- Hestness et al. (2017)
  - First empirical study of power-law scaling in deep learning
  - Found super-linear scaling of dataset size with model size, whereas this paper found a sub-linear scaling
- EfficientNet
  - Advocates scaling width, depth, and resolution optimally improves vision models
  - For language models this power should be roughly one when scaling up and precise architectural hyperparameters are unimportant compared to the overall scale of the language mode
- Webtext dataset
  - 45 million outgoing links and 40 GB of text

Useful Notation

L – the cross entropy loss in nats.

N – the number of model parameters, excluding all vocabulary and positional embeddings

C ≈ 6*N*B*S – an estimate of the total non-embedding training compute, where B is the batch size, and S is the number of training steps (ie parameter updates). Measured in PF-Days (one PF-Day = 8.64×10^{19} floating operations over 24 hours)

D – the dataset size in tokens

B_{crit} – the critical batch size

C_{min} – an estimate of the minimum amount of non-embedding compute to reach a given value of the loss. This is the training compute that would be used if the model were trained at a batch size much less than the critical batch size.

S_{min} – an estimate of the minimal number of training steps needed to reach a given value of the loss.

α_X – power-law exponents for the scaling of the loss as $L(X) \propto 1/X^{(\alpha_X)}$ where X can be any of N, D, C, S, B, C^{min}

Defining Scaling Laws

- Power-law relationships are derived between test loss and key scaling factors generally represented as **$L(X) \propto X^{-\alpha X}$**, where X represents model size, dataset size, or compute

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

$$L(D) = (D_c/D)^{\alpha_D}\ ;\ \alpha_D \sim 0.095, \quad D_c \sim 5.4 \times 10^{13}\ \text{(tokens)} \qquad (1.2)$$

$$L(C_{\min}) = \left(C_c^{\min}/C_{\min}\right)^{\alpha_C^{\min}}\ ;\ \alpha_C^{\min} \sim 0.050, \quad C_c^{\min} \sim 3.1 \times 10^8\ \text{(PF-days)} \qquad (1.3)$$

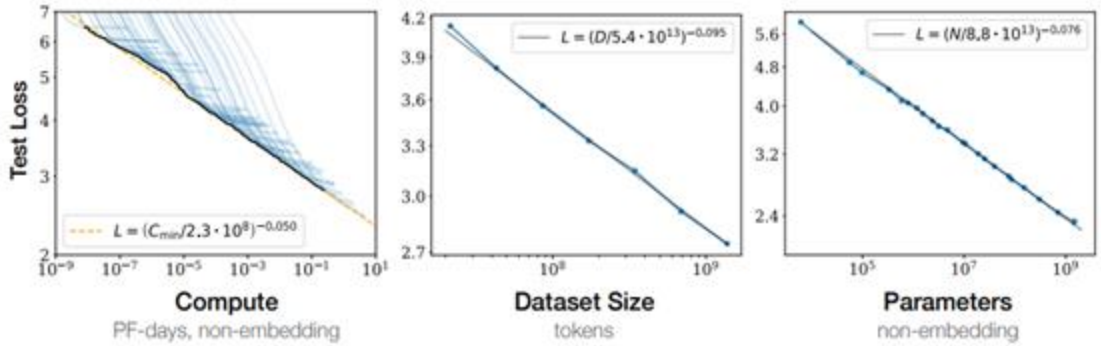


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

Transformer Model Architecture

- Focuses on decoder-only Transformer
  - Stacked self-attention layers so no encoder
  - Autoregressive training
  - Context window of n_{ctx} = 1024 tokens
- Model Hyperparameters: n_{layer} (Number of Layers), d_{model} (Hidden Size), n_{head} (Number of Attention Heads), d_{ff} (Feedforward Network Size)
- Equation for parameter count:
  - $N = 12 * n_{layer} * d^2_{model}$

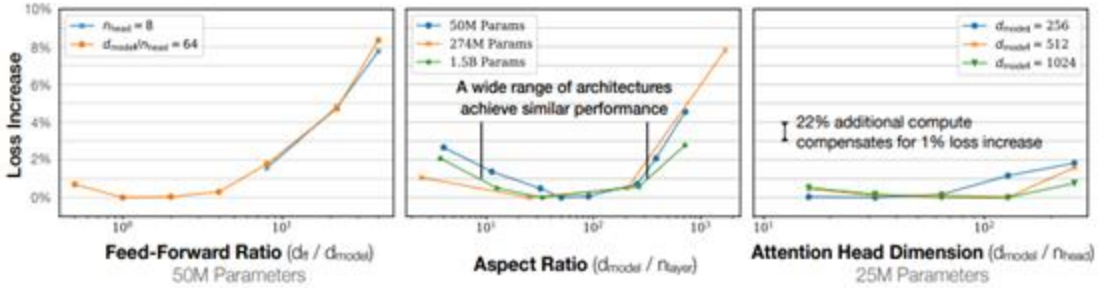


Figure 5 Performance depends very mildly on model shape when the total number of non-embedding parameters N is held fixed. The loss varies only a few percent over a wide range of shapes. Small differences in parameter counts are compensated for by using the fit to $L(N)$ as a baseline. Aspect ratio in 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].

Dataset and Compute Budget

- Dataset: WebText 2 (~22B tokens)
  - Extended from WebText, added “outbound” Reddit links
  - 20.3M documents containing 96 GB of text
  - Tokenized using Byte-Pair Encoding (BPE) (vocabulary size: 50,257)

- Model’s generalization is tested on Books Corpus, Common Crawl, Wikipedia

- Training Compute is measured in petalop-days (PF-days):
  - **C = 6*N*B*S**

Scaling with Model Size (N)

- Increasing model size improves performance
  - As the number of non-embedding parameters increases, test loss predictably decreases
- Mathematical Formula:

$$L(N) \approx \left(\frac{N_c}{N}\right)^{\alpha_N} \qquad (3.1)$$

- $\alpha_N \approx 0.076$, N_c = empirical constant

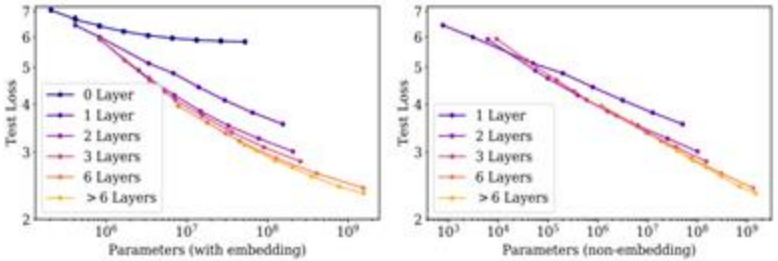


Figure 6 **Left:** When we include embedding parameters, performance appears to depend strongly on the number of layers in addition to the number of parameters. **Right:** When we exclude embedding parameters, the performance of models with different depths converge to a single trend. Only models with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

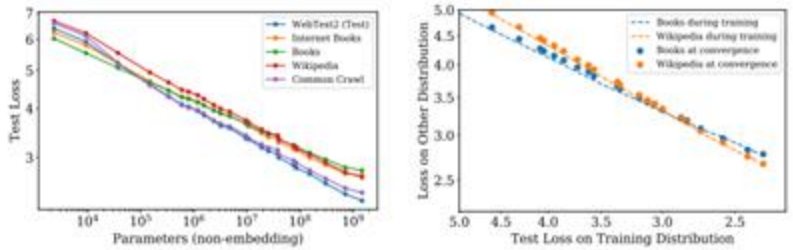


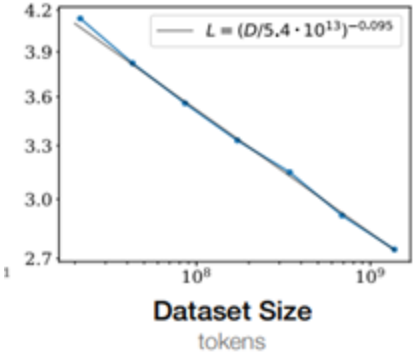
Figure 8 **Left:** Generalization performance to other data distributions improves smoothly with model size, with only a small and very slowly growing offset from the WebText2 training distribution. **Right:** Generalization performance depends only on training distribution performance, and not on the phase of training. We compare generalization of converged models (points) to that of a single large model (dashed curves) as it trains.

Scaling with Dataset Size (D)

- More data leads to lower loss (but with diminishing returns)
  - Larger datasets improve performance, but loss reduction slows down as data scales
- Mathematical Formula:

$$L(D) \approx \left(\frac{D_c}{D}\right)^{\alpha_D} \tag{3.2}$$

- $\alpha_D \approx 0.095$, D_c = empirical normalization constant
- Model was trained with (n_{layer}, n_{embd}) = (36, 1280)



rmance improves smoothly as we in
or training. For optimal performanc
e has a power-law relationship with

Scaling with Compute Budget (C)

- More Compute Improves Performance (but is the least efficient factor)
  - Increasing compute reduces loss, but at a slower rate than the model size or dataset size
- Mathematical Formula:

$$L(C) \approx \left(\frac{C_c}{C}\right)^{\alpha_C} \tag{3.3}$$

- $\alpha_C \approx 0.050$, C_c = A critical compute budget

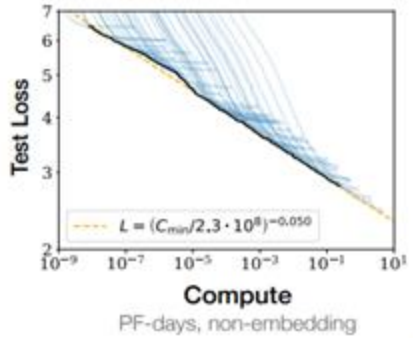


Figure 1 Language modeling perforr
size, and amount of compute[2] used for
up in tandem. Empirical performance
bottlenecked by the other two.

Performance of Model Size (N) on Dataset (D)

- When training models of increasing size, test loss improves as long as the dataset size is scaled appropriately
- If D remains fixed while N increases, we get diminishing returns and overfitting
- To formalize this relationship the authors introduce the following:

$$L(N, D) = \left[\left(\frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right]^{\alpha_D} \tag{4.1}$$

- Key principles:
  - Changes in vocabulary size or tokenization are expected to rescale the loss by an overall factor
  - Fixing D and sending N → ∞, the overall loss should approach L(D). Conversely, fixing N and sending D → ∞ the loss must approach L(N)
  - L(N, D) should be analytic at D = ∞, so that it has a series expansion in 1/D with integer powers

Overfitting

- The dataset must scale with the model size
  - If model size grows without increasing dataset size, test loss stops improving and overfitting begins
- Mathematical Relationship:
  - Optimal dataset size should scale sublinearly with model size:

$$D \propto N^{0.74} \propto C_{\min}^{0.54} \tag{6.6}$$

  - Overfitting Ratio:

$$\delta L(N, D) \equiv \frac{L(N, D)}{L(N, \infty)} - 1 \tag{4.2}$$

$$\delta L \approx \left(1 + \left(\frac{N}{N_c}\right)^{\frac{\alpha_N}{\alpha_D}} \frac{D_c}{D}\right)^{\alpha_D} - 1 \tag{4.3}$$

Data Size Bottleneck and Overfitting

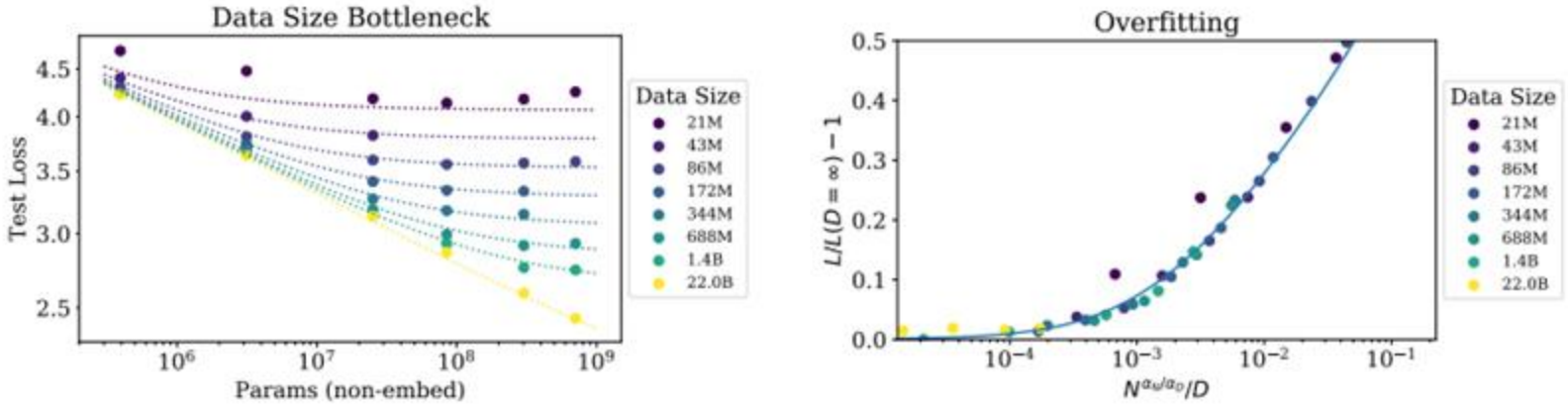


Figure 9 The early-stopped test loss $L(N, D)$ depends predictably on the dataset size D and model size N according to Equation (1.5). **Left**: For large D, performance is a straight power law in N. For a smaller fixed D, performance stops improving as N increases and the model begins to overfit. (The 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.

Batch Size & Critical Batch Size (B_{crit})

- Critical batch size B_{crit} defines the optimal tradeoff between training time and compute efficiency
- Mathematical Formula:

$$B_{\text{crit}}(L) \approx \frac{B_*}{L^{1/\alpha_B}} \tag{5.3}$$

- α_B = Empirical scaling exponent (~0.21), B_{crit} = Ideal batch size for a given loss L, B_* ≈ 2 × 108

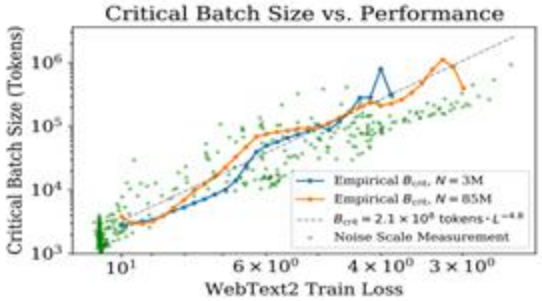


Figure 10 The critical batch size B_{crit} follows a power law in the loss as performance increase, and does not depend directly on the model size. We find that the critical batch size approximately doubles for every 13% decrease in loss. B_{crit} is measured empirically from the data shown in Figure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

Optimal Compute Allocation

- Given a fixed compute budget C, the **optimal strategy** is to:
  - **Increase model size** aggressively (N).
  - **Increase batch size** moderately (B).
  - **Increase training steps only slightly (S)**

$$N \propto C_{\min}^{0.73},\ B \propto C_{\min}^{0.24},\ \text{and}\ S \propto C_{\min}^{0.03}$$

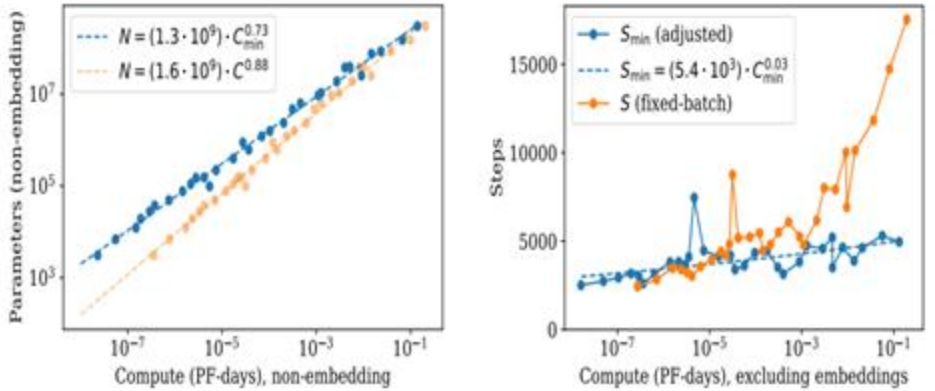


Figure 14 **Left:** Each value of the compute budget $C_{\min}$ has an associated optimal model size N. Optimal model size grows very rapidly with $C_{\min}$, increasing by 5x for each 10x increase in compute. The number of data examples processed makes up the remainder of the increase, growing relatively modestly by only 2x. **Right:** The batch-adjusted number of optimization steps also grows very slowly, if at all, meaning that most of the growth in data examples processed can be used for increased batch sizes.

Performance of Model Size (N) with Training Steps (S)

- Training curves follow predictable power laws similar to model and dataset scaling
  - If a model is trained for too few steps, it won't reach optimal performance
  - If a model is trained for too many steps we waste compute with minimal performance gains
  - The authors derive equations to allow for **predicting model loss at any point in training**, without needing to fully train the model!

$$L(N, S_{\min}) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}}\right)^{\alpha_S} \qquad (5.6)$$

- S_c = a characteristic number of training steps required for convergence, S_{min} = training steps adjusted for batch size

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 $L(N, S)$

Performance of Model Size (N) with Training Steps (S) cont'd

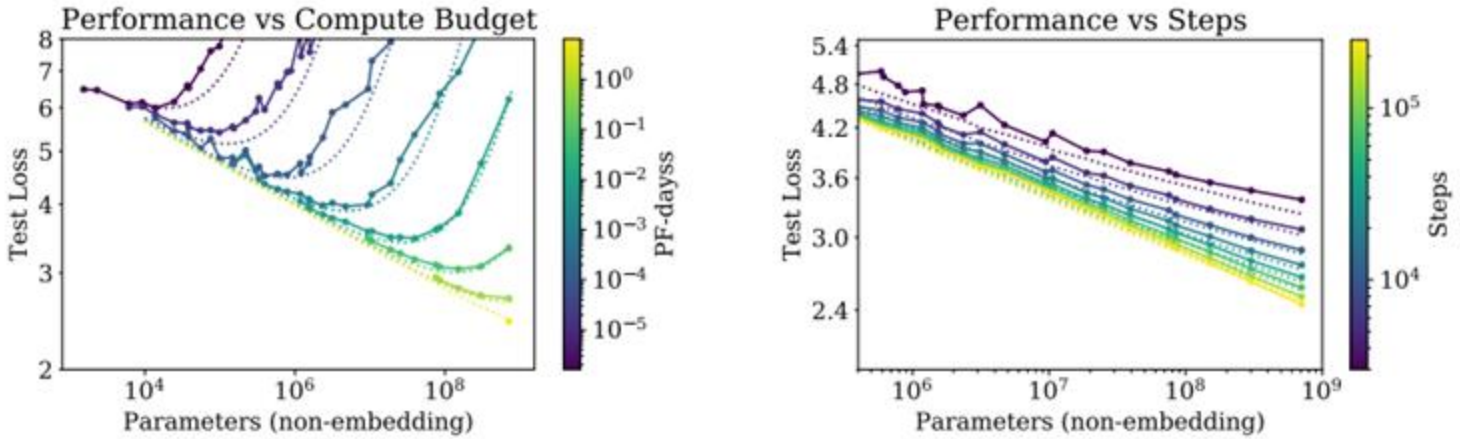


Figure 11 When we hold either total compute or number of training steps fixed, performance follows $L(N, S)$ from Equation (5.6). Each value of compute budget has an associated optimal model size that maximizes performance. Mediocre fits at small S are unsurprising, as the power-law equation for the learning curves breaks down very early in training.

Why Stopping Training Early is Optimal

- The most compute-efficient training strategy is to stop training early, rather than training to full convergence.
  - Empirical scaling shows that early stopping can reduce compute needs by 50% with minimal loss increase
  - Loss vs. training steps follows a power-law, meaning most improvements happen early

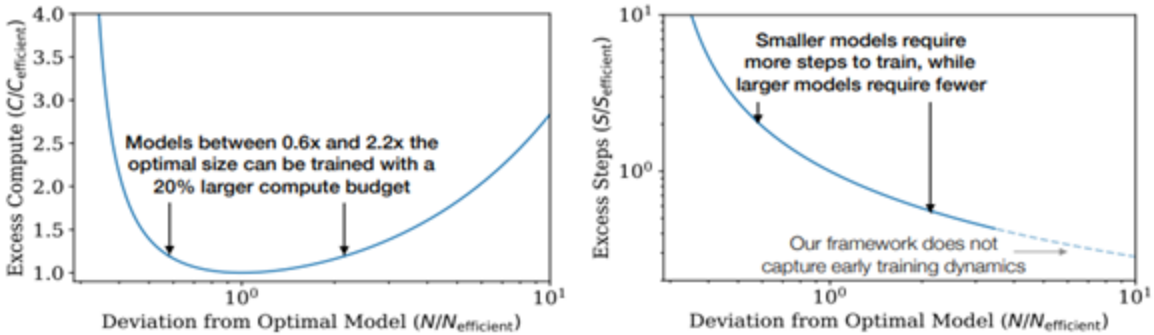


Figure 12 **Left:** Given a fixed compute budget, a particular model size is optimal, though somewhat larger or smaller models can be trained with minimal additional compute. **Right:** Models larger than the compute-efficient size require fewer steps to train, allowing for potentially faster training if sufficient additional parallelism is possible. Note that this equation should not be trusted for very large models, as it is only valid in the power-law region of the learning curve, after initial transient effects.

Limits of Scaling – Data & Compute Constraints

- The Scaling Laws predict at some extreme point $L(C_{min})$ will become lower than what is possible with finite data
  - The contradiction is that infinite compute alone does not guarantee infinite improvement because data becomes the limiting factor
- The authors propose a fundamental limit where at some point, we will extract all possible learnable patterns from text data
  - Beyond this point, loss will not improve, even with more model parameters or compute
- This breakdown would occur around:

$$C^* \sim 10^4 \text{ PF-Days} \quad N^* \sim 10^{12} \text{ parameters}, \quad D^* \sim 10^{12} \text{ tokens}, \quad L^* \sim 1.7 \text{ nats/token} \qquad (6.8)$$

Implications of Scaling Laws

- Bigger models generalize better
  - Large models trained on enough data show better sample efficiency than smaller models trained longer
- Compute-efficient training is crucial
  - Development is now limited more by compute and data efficiency than by algorithmic improvements
- Diminishing returns in scaling
  - As models approach massive scales (~10^{12} parameters), marginal gains shrink—alternative strategies needed.