Scaling Laws for Neural Language Models

Scaling Laws: Wonach suchen wir?

- **Skaleninvarianz**: Soll breite Gültigkeit haben
- Im Paper: **Power Laws**, also Potenzgesetze: $f: x \mapsto ax^b$

Scaling Laws: Wonach suchen wir?

- **Skaleninvarianz**: Soll breite Gültigkeit haben
- Im Paper: **Power Laws**, also Potenzgesetze: $f: x \mapsto ax^b$
- verwandtes Konzept: **Homogene Funktionen**, erweitern auf $\mathbb{R}^n$, fordern aber gleichzeitige Skalierung aller Parameter: $f(tx_1, \ldots, tx_n) = t^\lambda f(x_1, \ldots, x_n)$
    - Lies: f ist homogen vom Grad λ
    - Rabbit Hole: Scholarpedia, Wikipedia (EN), Wilde Paper

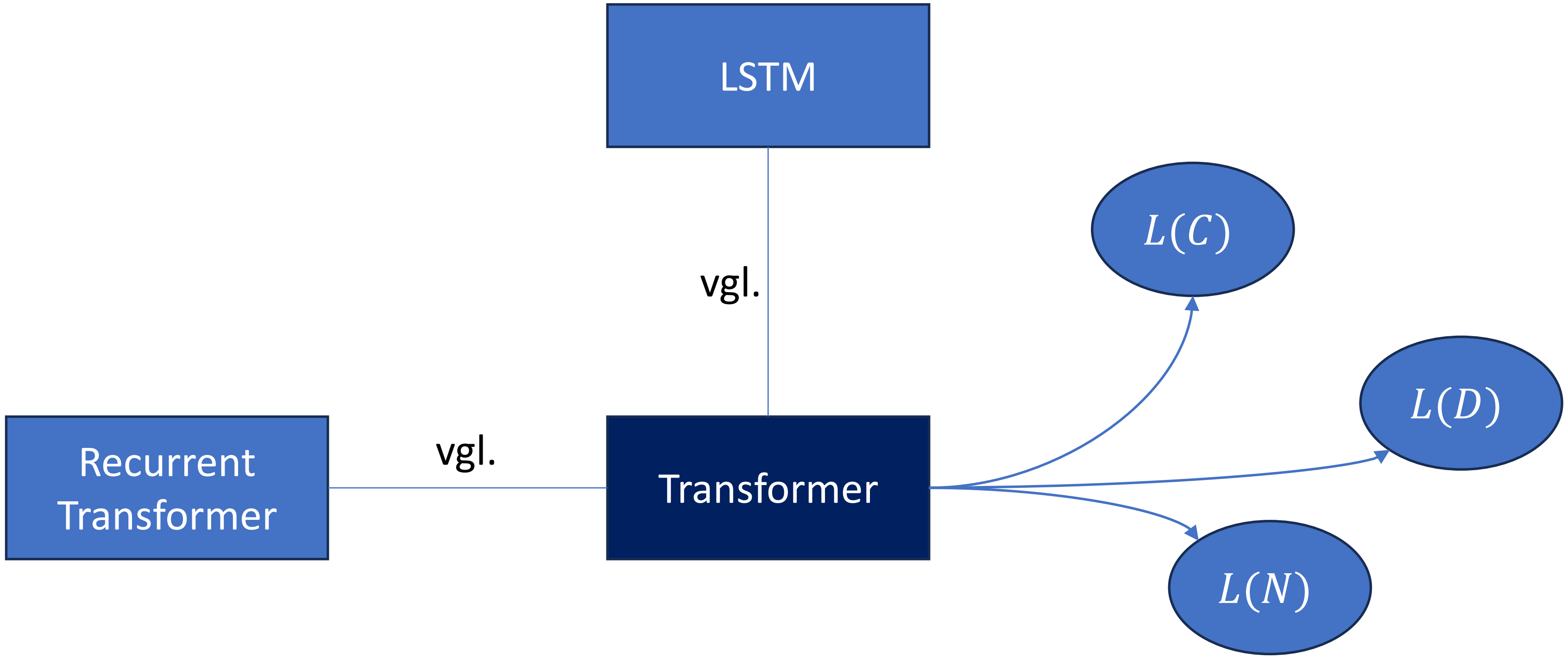
Methodik: Parameter und Messgröße

- Vergleichsgröße: **Cross Entropy Loss** $-\sum_i p_i \cdot ln(q_i)$

- Parameter: Genutzte Rechenleistung, Parameterzahl, Trainingsdaten

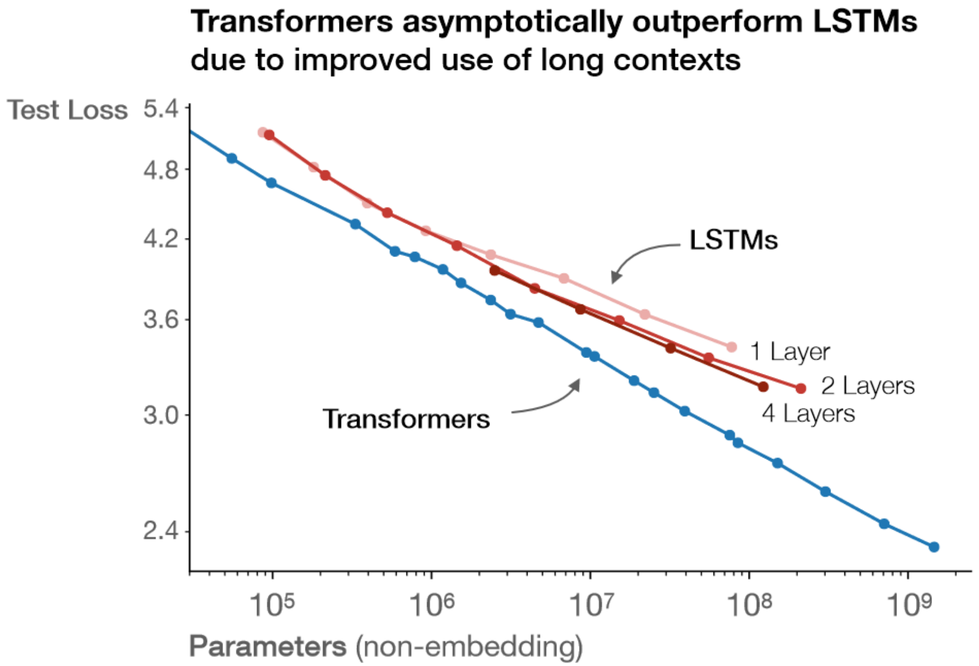
- Trainingsdatensätze:
  - WebText2
  - Englische Wikipedia (Hugging Face, Wikimedia)
  - BookCorpus (Hugging Face)
  - Common Crawl (Dez '19)
  - Gratisbücher

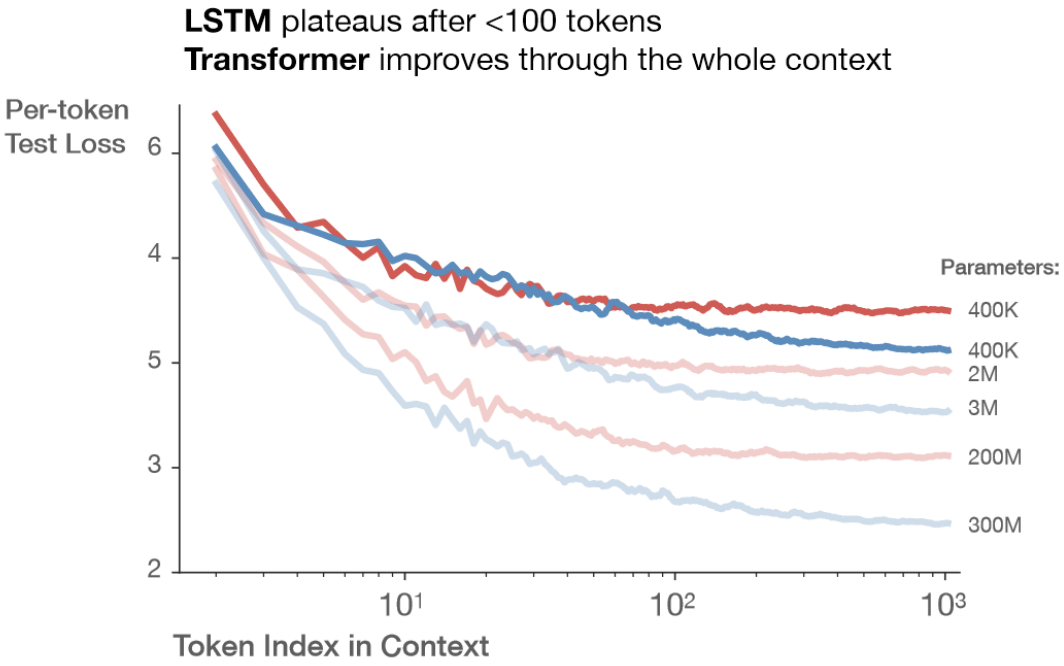
Methodik: Abgrenzung der Arbeit



Vergleich zu LSTM

- LSTM folgen auch Potenzgesetzen
- Lediglich allgemein schlechtere Performance



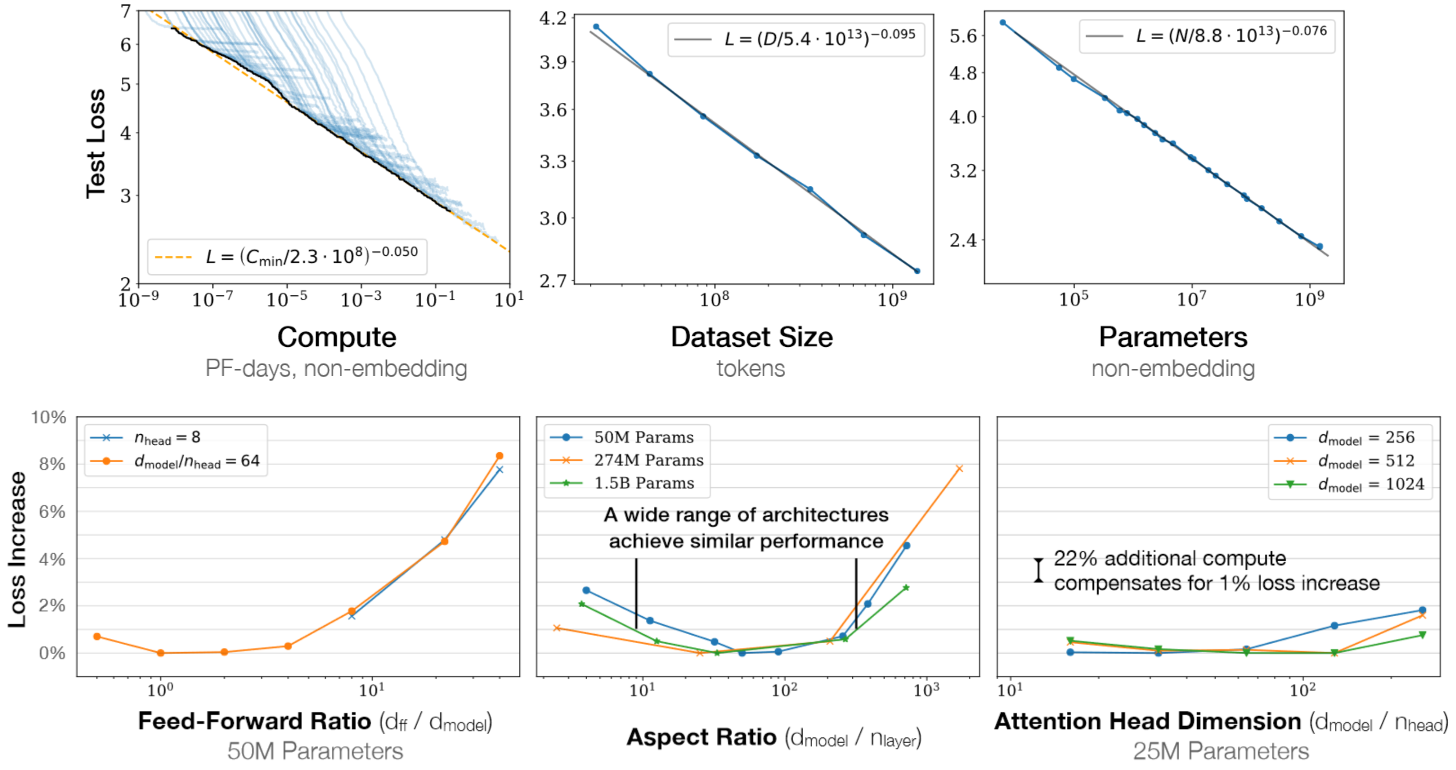


Die Acht Beobachtungen

- Performance depends strongly on scale, weakly on model shape
- Smooth power laws
- Universality of overfitting
- Universality of training
- Transfer improves with test performance
- Sample efficiency
- Convergence is inefficient
- Optimal batch size

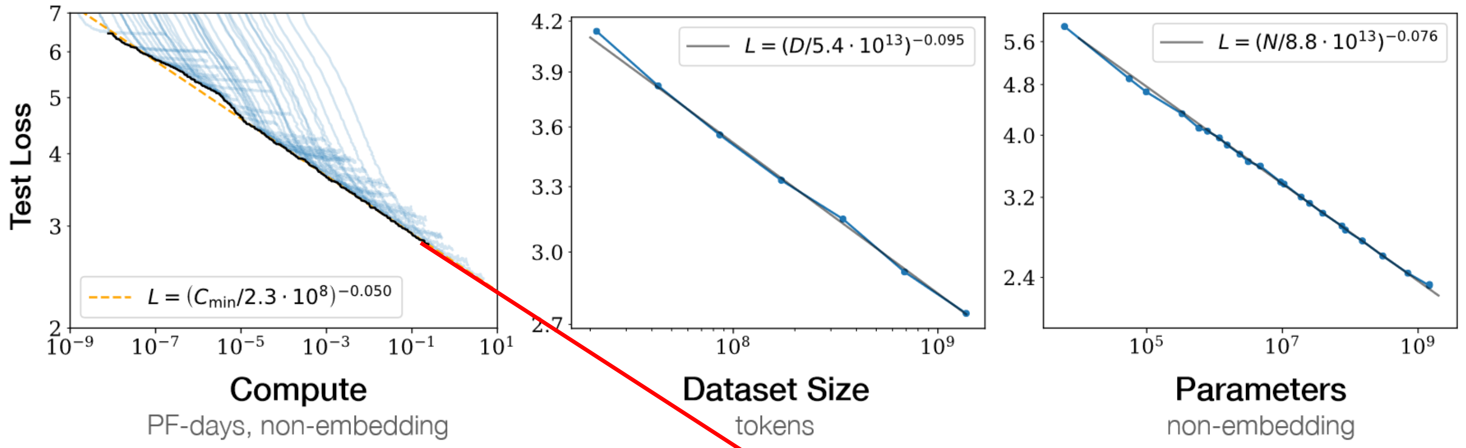
Die Acht Beobachtungen

- Performance depends strongly on scale, weakly on model shape
- Smooth...
- Universality 1
- Universality 2
- Transfer...
- Sample...
- Convergence...
- Batch size...



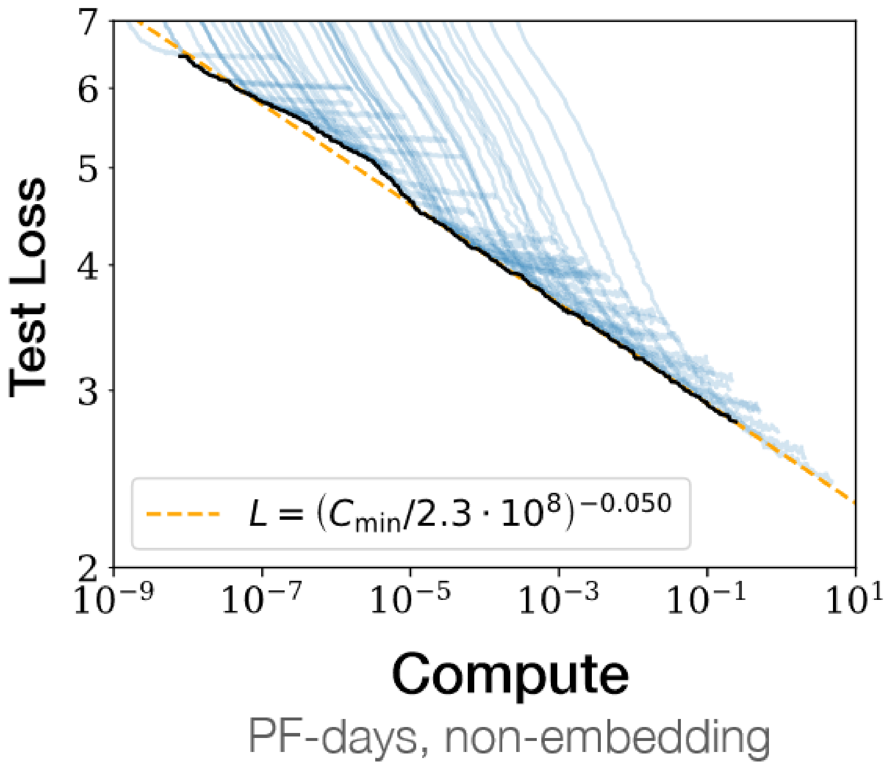
Die Acht Beobachtungen

- Performance...
- Smooth power laws
- Universality 1
- Universality 2
- Transfer...
- Sample...
- Convergence...
- Batch size...



Die Acht Beobachtungen

- Performance...
- **Smooth power laws**
- Universality 1
- Universality 2
- Transfer...
- Sample...
- Convergence...
- Batch size...



- Schwarze Linie?
  - Beste Perf. bei $S = \frac{C}{6BS}$ N?
- Blaue Linien?
  - können nicht einzelne Läufe mit festem C sein: bewegen sich alle entlang x-Achse
  - vllt. Cmin oder B?
- Ist das ein Roofline Model?
  - Bottleneck durch D unterhalb L?

Die Acht Beobachtungen

- Performance...
- Smooth...
- Universality of overfitting
- Universality 2
- Transfer...
- Sample...
- Convergence...
- Batch size...

N und D gleichzeitig erhöhen!

- *N zu klein*: hat das Modell zu wenig Parameter, um ganz D abzubilden?
- *D zu klein*: klassisches Overfitting (?)

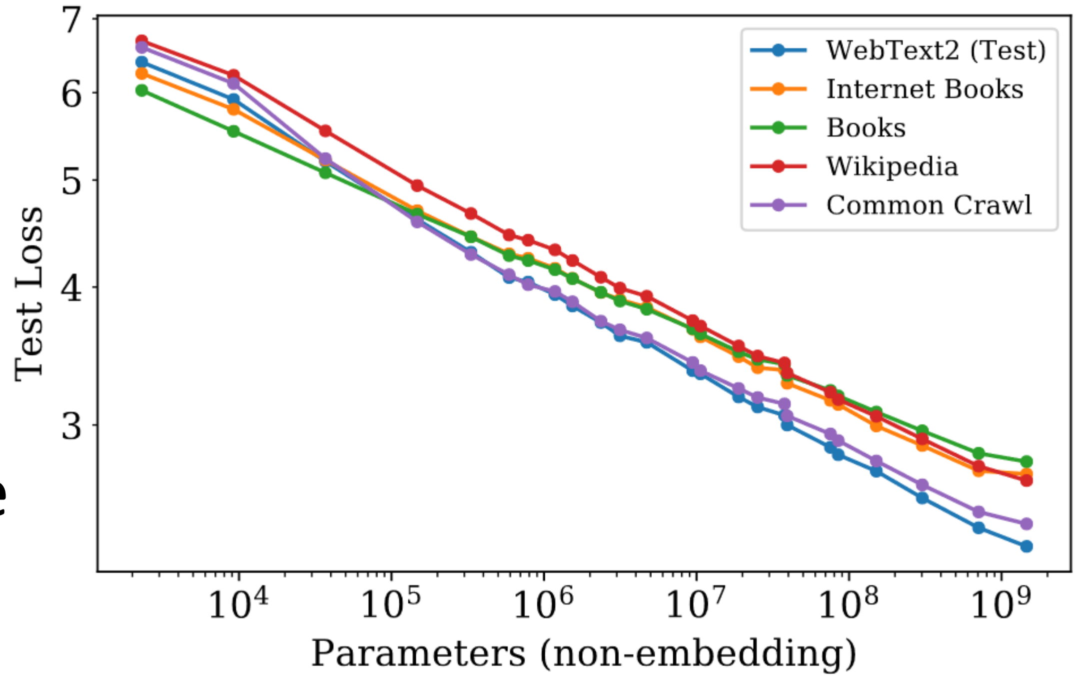
Die Acht Beobachtungen

- Performance...
- Smooth...
- Universality 1
- Universality of training
- Transfer...
- Sample...
- Convergence...
- Batch size...

Die Gestalt der Loss-Kurve ist ax^b, wobei b unabhängig von der Modellgröße ist

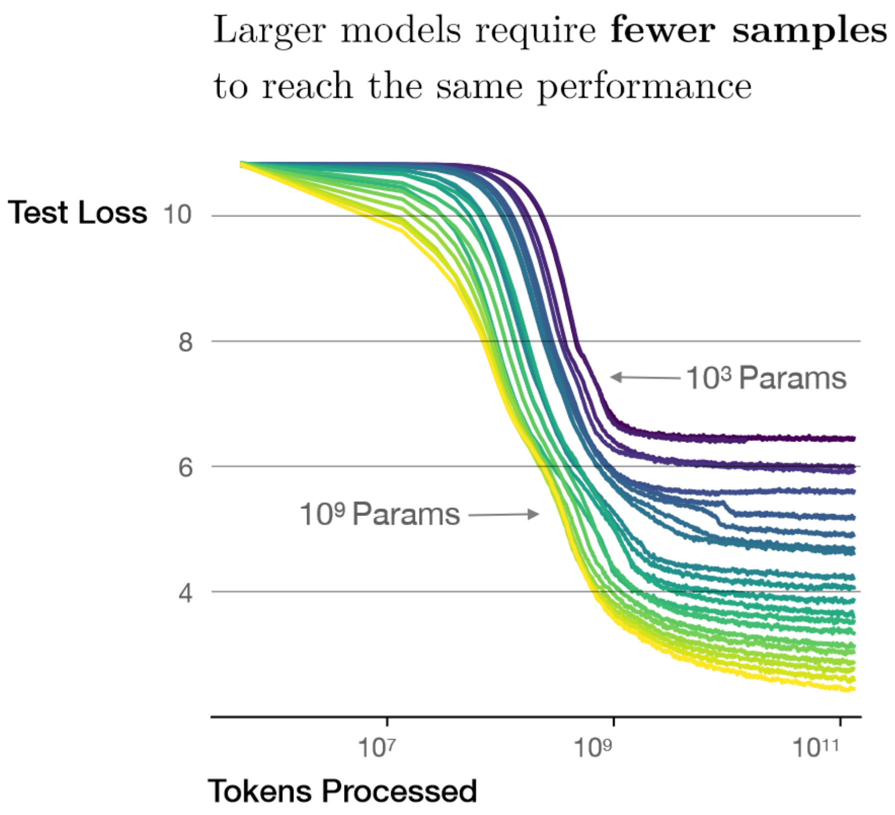
Die Acht Beobachtungen

- Performance...
- Smooth...
- Universality 1
- Universality 2
- **Transfer improves with test performance**
- Sample...
- Convergence...
- Batch size...



Die Acht Beobachtungen

- Performance...
- Smooth...
- Universality 1
- Universality 2
- Transfer...
- Sample efficiency
- Convergence...
- Batch size...



Die Acht Beobachtungen

- Performance...
- Smooth...
- Universality 1
- Universality 2
- Transfer...
- Sample...
- Convergence is inefficient
- Batch size...

- Festes Rechenbudget C wird am besten genutzt, wenn N sehr groß ist
  - konvergiert schneller
  - konvergiert gegen geringeren Loss
  - wird innerhalb von C nicht konvergieren

Die Acht Beobachtungen

- Performance...
- Smooth...
- Universality 1
- Universality 2
- Transfer...
- Sample...
- Convergence...
- Optimal batch size

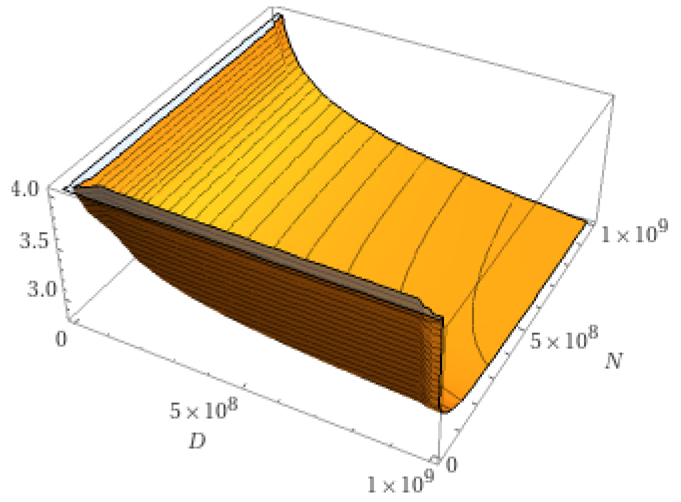
- Kleine Batchsizes B sind **recheneffizient**
- $B \ll B_{crit}$: Minimiert Rechenaufwand C
- $B \gg B_{crit}$: Minimiert Anzahl Trainingsschritte S
- $B <_{bisschen} B_{crit}$: Minimiert C und S fast
- Daher: $\boldsymbol{B_{crit}}$ **(näherungsweise) optimal**

- $B_{crit}(L) \approx \frac{B_*}{L^{\frac{1}{\alpha_B}}}$, also nur von L abhängig
  - Bsp.: $L = 2 \Rightarrow B_{crit} \approx 7 \cdot 10^6$

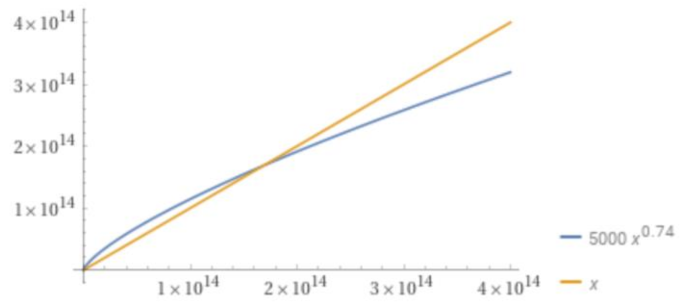
btw. $> 2^{19} \approx 500K$

Die wichtige Formel aus Kapitel 4

- $L(N, D) = \left[\left(\frac{N_C}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_C}{D}\right]^{\alpha_D}$
- Modelliert: Großes N braucht weniger D
  - Also: N (durch Hardware) gegeben. Wie groß muss D sein?
  - Keine Aussage darüber, wann L konvergiert!
  - Keine Abschätzung über Rechenaufwand!
- Overfitting-Grenze: D $\gtrsim (5 \cdot 10^3) N^{0.74}$
  - Bsp. $N = 10^7 \Rightarrow D \gtrsim 8 \cdot 10^8$
  - **Erst ab $\sim\mathbf{2 \cdot 10^{14}}$ sublinear!**



Wolframalpha



Wolframalpha

Compute Abschätzung

Habe 8 A100 für einen Tag, wie D, N, B und S wählen um L zu minimieren?

- Kapitel 5 liefert verlockende Formel: $C_{min}(C) \equiv \frac{C}{1+\frac{B}{B_{crit}(L)}}$
  - $B_{crit}(L)$ gegeben: $\frac{B_*}{L^{\frac{1}{\alpha_B}}}$
  - C gegeben: $6NBS$
  - Nach L umstellen?
- Eigentlich suchen wir aber nicht $C_{min}(C)$ oder $L(C_{\min})$ sondern $L(C)$

Compute Abschätzung

Habe 8 A100 für einen Tag, wie D, N, B und S wählen um L zu minimieren?

- Kapitel 5 liefert verlockende Formel: $C_{min}(C) \equiv \frac{C}{1+\frac{B}{B_{crit}(L)}}$
  - $B_{crit}(L)$ gegeben: $\frac{B_*}{L^{\frac{1}{\alpha_B}}}$
  - C gegeben: $6NBS$
  - Nach L umstellen?
- Eigentlich suchen wir aber nicht $C_{min}(C)$ oder $L(C_{\min})$ sondern $L(C)$

???

Tabellen aus Appendix A

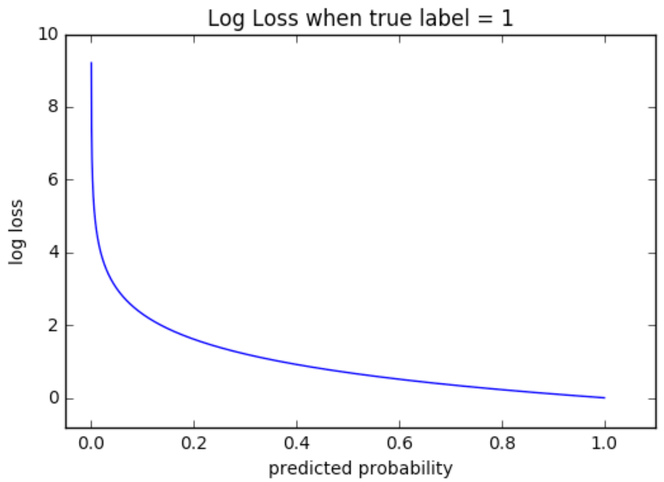
Parameters	Data	Compute	Batch Size	Equation
N	∞	∞	Fixed	$L(N) = (N_c/N)^{\alpha_N}$
∞	D	Early Stop	Fixed	$L(D) = (D_c/D)^{\alpha_D}$
Optimal	∞	C	Fixed	$L(C) = (C_c/C)^{\alpha_C}$ (naive)
N_{opt}	D_{opt}	C_{min}	$B \ll B_{\text{crit}}$	$L(C_{\text{min}}) = \left(C_c^{\text{min}}/C_{\text{min}}\right)^{\alpha_C^{\text{min}}}$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

Power Law	Scale (tokenization-dependent)
$\alpha_N = 0.076$	$N_c = 8.8 \times 10^{13}$ params (non-embed)
$\alpha_D = 0.095$	$D_c = 5.4 \times 10^{13}$ tokens
$\alpha_C = 0.057$	$C_c = 1.6 \times 10^{7}$ PF-days
$\alpha_C^{\text{min}} = 0.050$	$C_c^{\text{min}} = 3.1 \times 10^{8}$ PF-days
$\alpha_B = 0.21$	$B_* = 2.1 \times 10^{8}$ tokens
$\alpha_S = 0.76$	$S_c = 2.1 \times 10^{3}$ steps

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

Zusatz: Notation

- L – Loss in **nats** (Basis e bits)
- N – Parameterzahl ohne Embedding/Position
- D – Tokenzahl im Datensatz
- C – Rechenaufwand für Training in PFLOP-Tagen
- B_{crit} – Kritische Batchgröße. Optimum von Rechenzeit und Effizienz
- C_{min} – Minimaler Rechenaufwand, um gegebenen Loss zu erreichen
- S_{min} – Minimale Schrittzahl
- α_X – Exponenten im Potenzgesetz



Cross Entropy Loss (“Log Loss”). Quelle: [Internet]

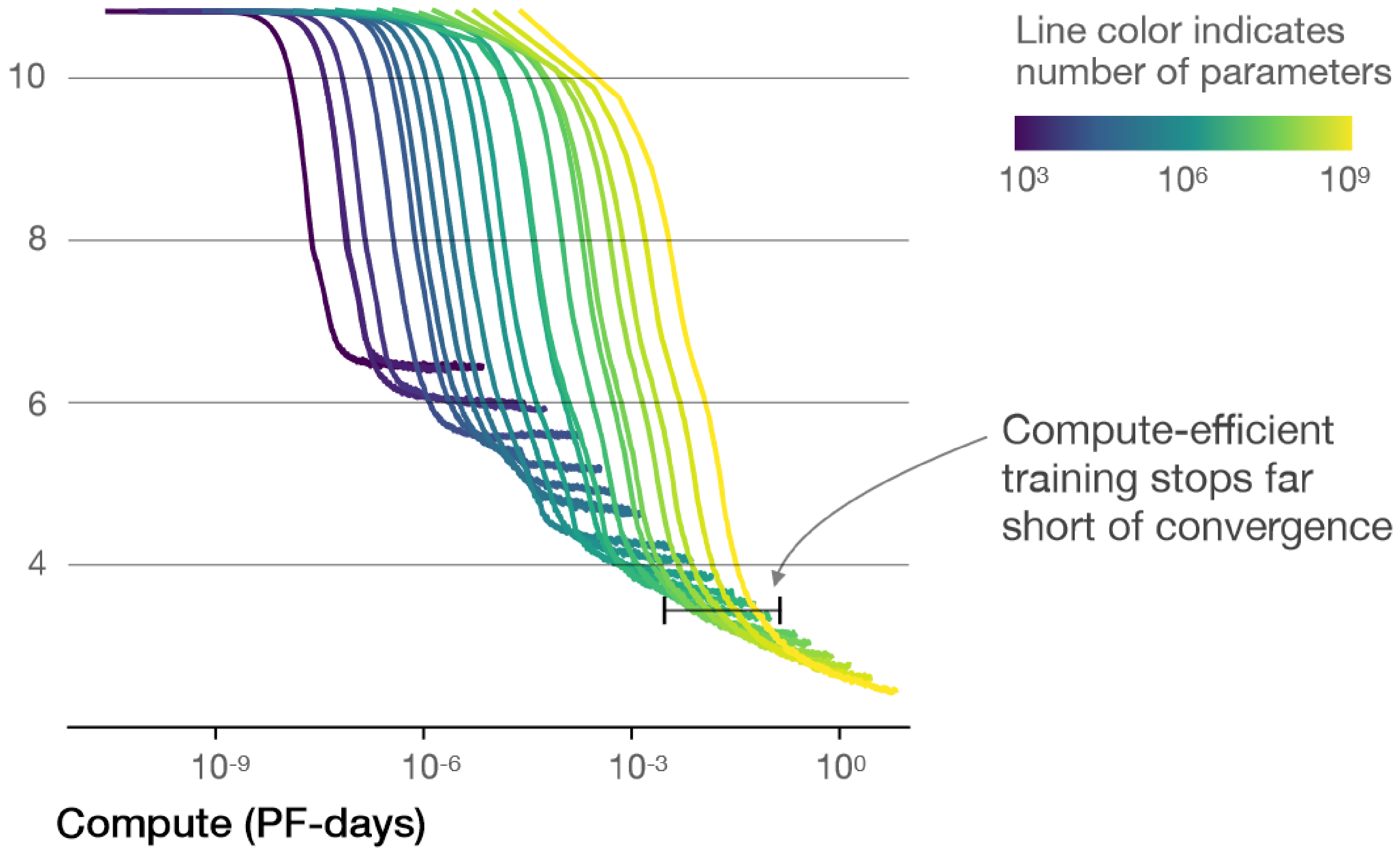
Zusatz: Der 8A100Tag

$$
\begin{aligned}
1\,A100 &= 19.5\,(FP32)\,TFLOPS \\
8\,A100 &= 156\,TFLOPS \qquad |\cdot 86400\,s \\
&= 13\,478\,400 \cdot 10^{12}\,FLOP \\
&\cong 1.34 \cdot 10^{19}\,FLOP = 1\,8A100Tag
\end{aligned}
$$

$\frac{1.34 \cdot 10^{19}}{8.64 \cdot 10^{19}} = 0.156$, was man sich hätte denken können ($156\,TFLOPS$)

Zusatz: Undiskutierte Graphen

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



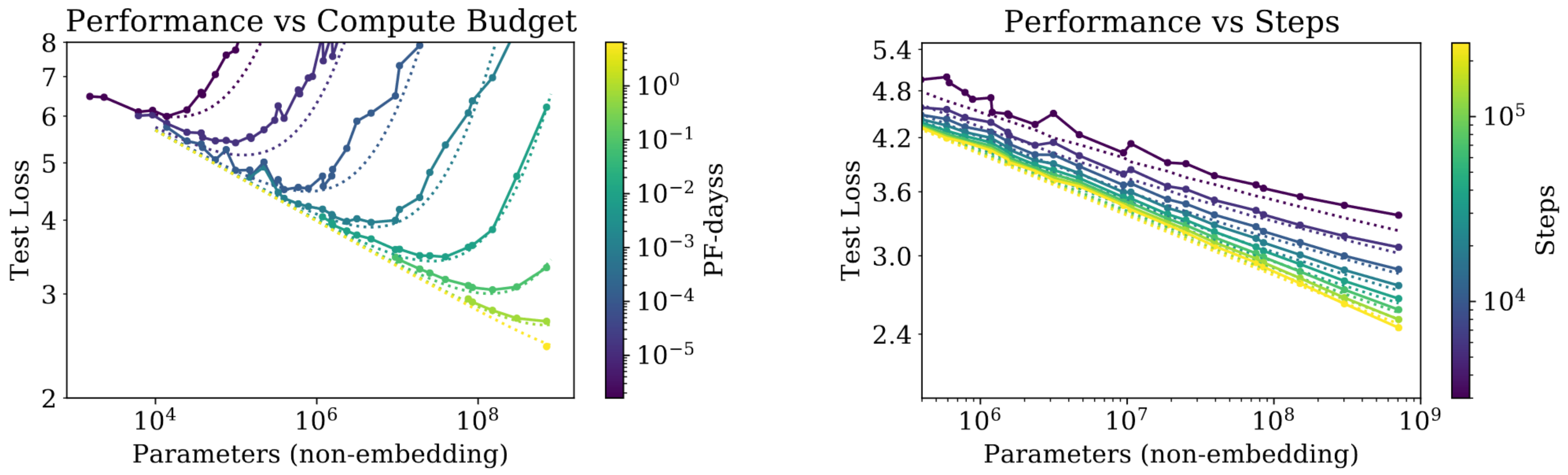


Figure 11 When we hold either total compute or number of training steps fixed, performance follows $L(N, S)$ from Equation (5.6). Each value of compute budget has an associated optimal model size that maximizes performance. Mediocre fits at small S are unsurprising, as the power-law equation for the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

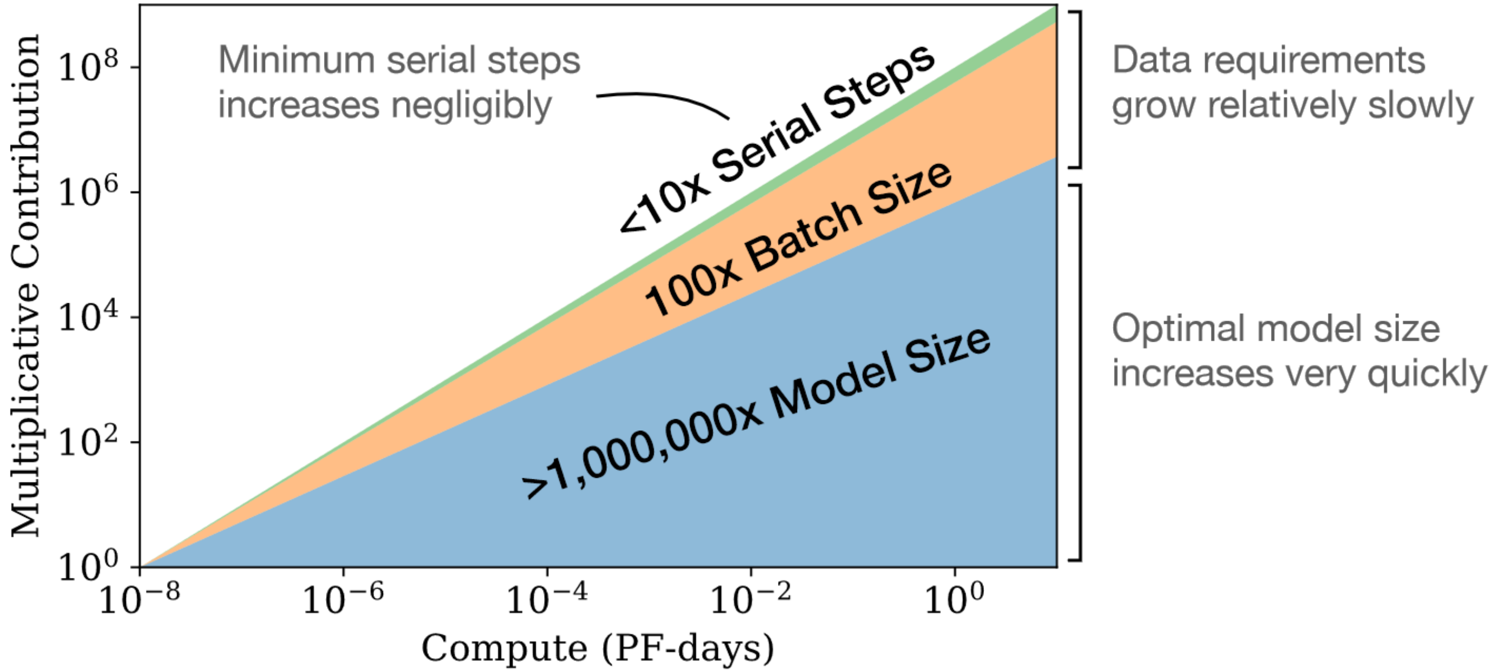


Figure 3 As more compute becomes available, we can choose how much to allocate towards training larger models, using larger batches, and training for more steps. We illustrate this for a billion-fold increase in compute. For optimally compute-efficient training, most of the increase should go towards 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.

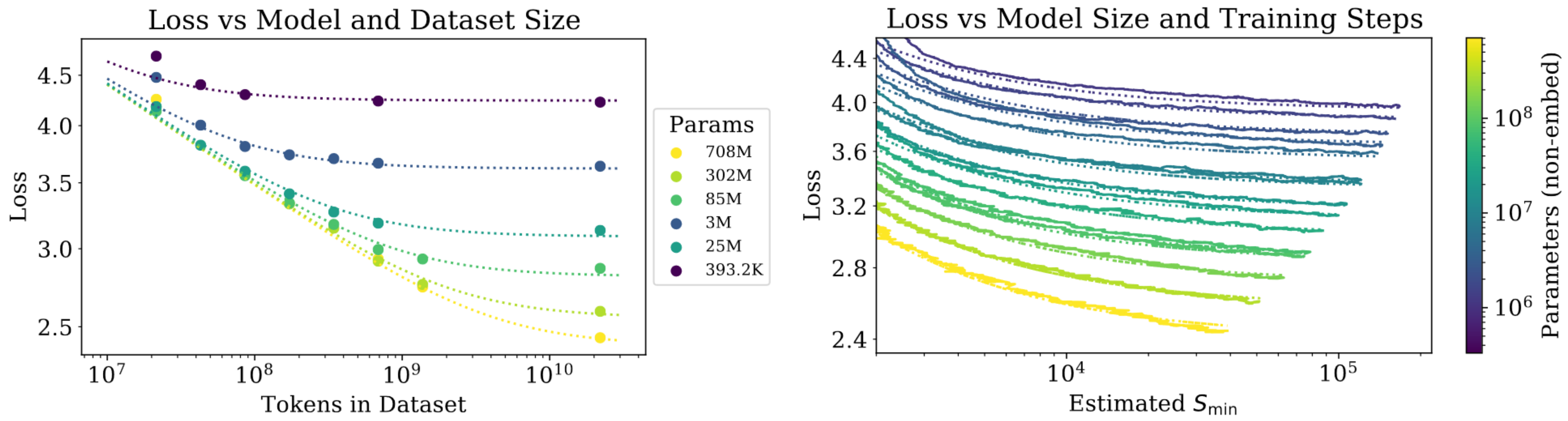


Figure 4 **Left**: The early-stopped test loss $L(N, D)$ varies predictably with the dataset size D and model size N according to Equation (1.5). **Right**: After an initial transient period, learning curves for all model sizes N can be fit with Equation (1.6), which is parameterized in terms of S_{min}, the number of steps when training at large batch size (details in Section 5.1).

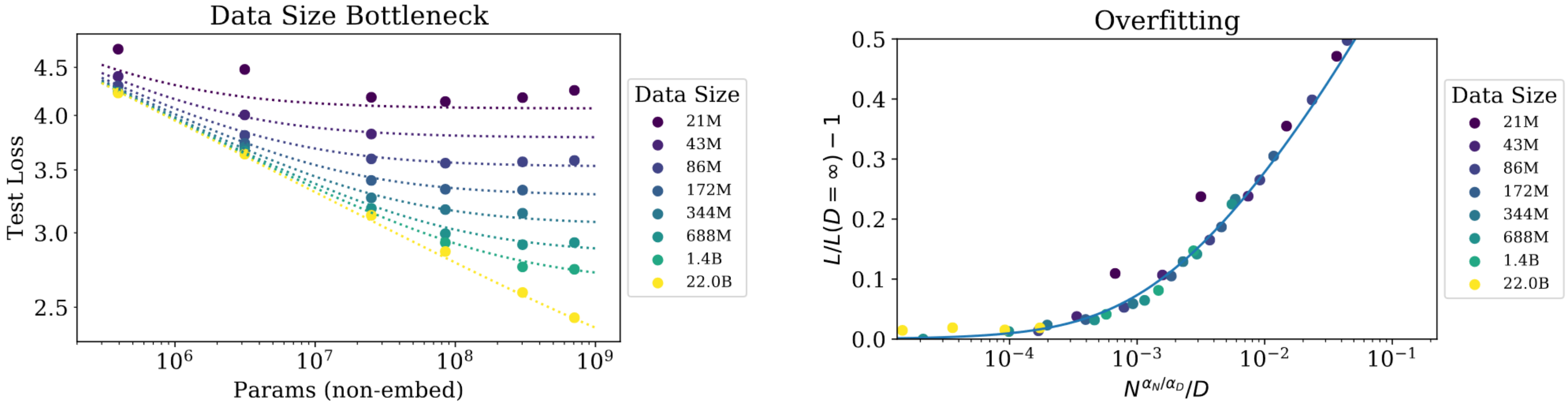


Figure 9 The early-stopped test loss $L(N, D)$ depends predictably on the dataset size D and model size N according to Equation (1.5). **Left**: For large D, performance is a straight power law in N. For a smaller fixed D, performance stops improving as N increases and the model begins to overfit. (The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

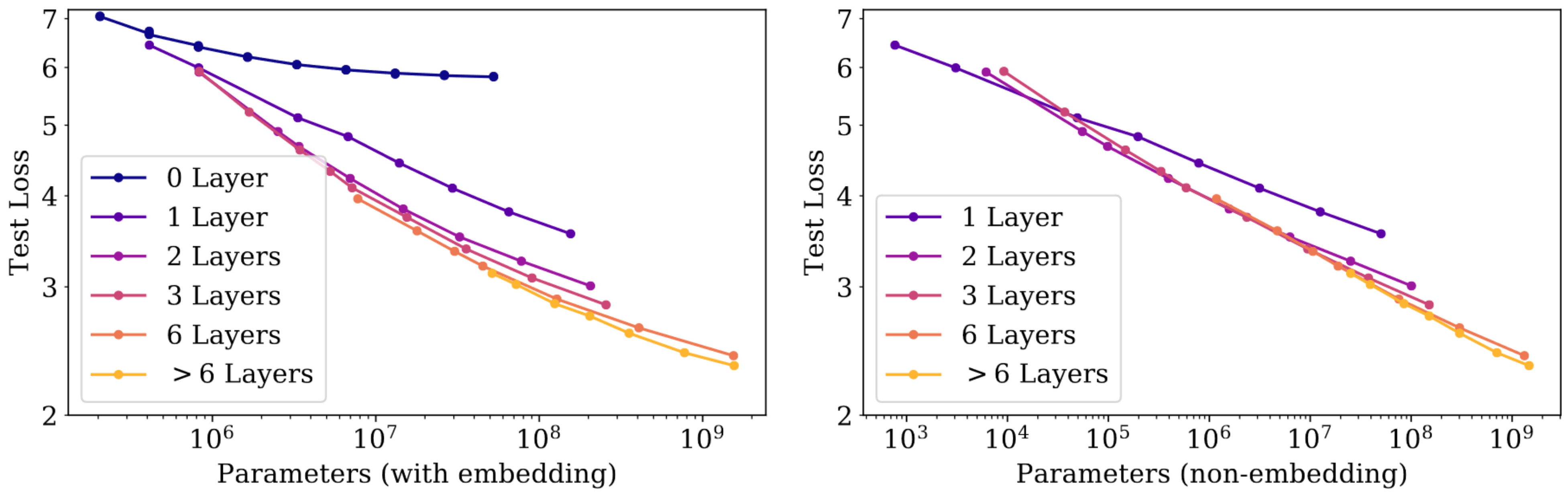


Figure 6 **Left:** When we include embedding parameters, performance appears to depend strongly on the number of layers in addition to the number of parameters. **Right:** When we exclude embedding parameters, the performance of models with different depths converge to a single trend. Only models with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

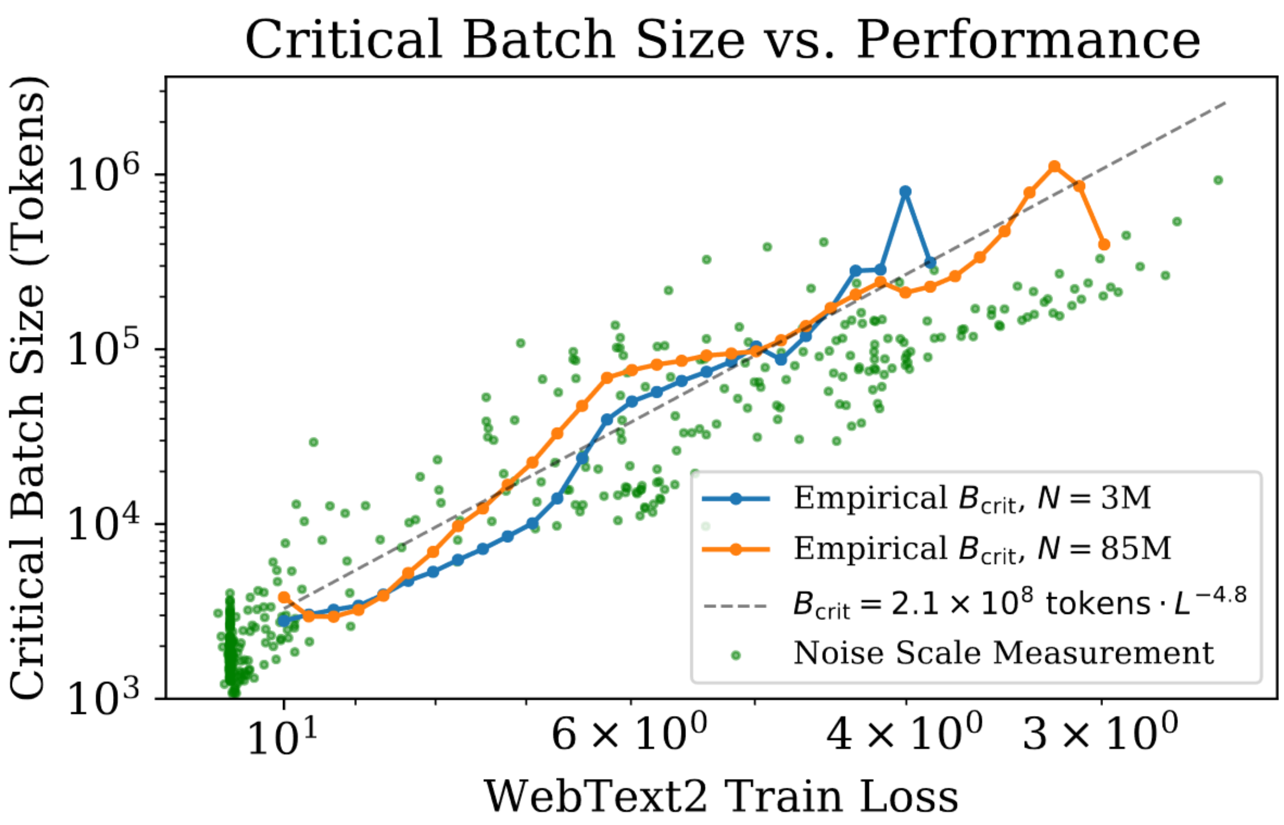


Figure 10 The critical batch size $B_{\rm crit}$ follows a power law in the loss as performance increase, and does not depend directly on the model size. We find that the critical batch size approximately doubles for every 13% decrease in loss. $B_{\rm crit}$ is measured empirically from the data shown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

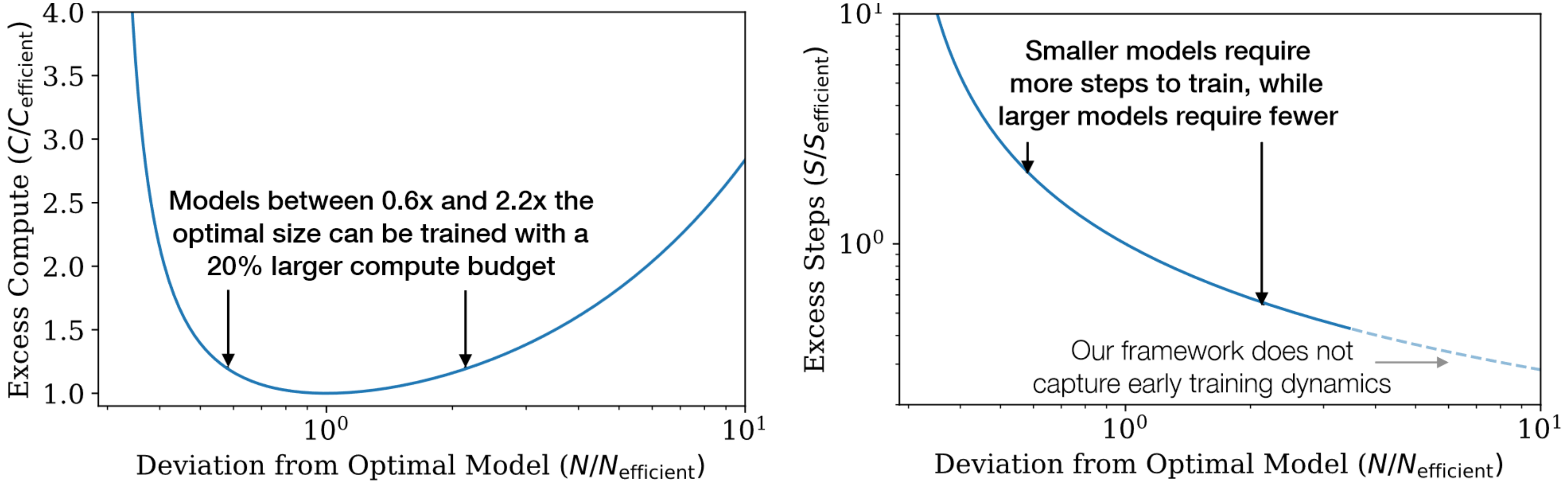


Figure 12 **Left:** Given a fixed compute budget, a particular model size is optimal, though somewhat larger or smaller models can be trained with minimal additional compute. **Right:** Models larger than the compute-efficient size require fewer steps to train, allowing for potentially faster training 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.

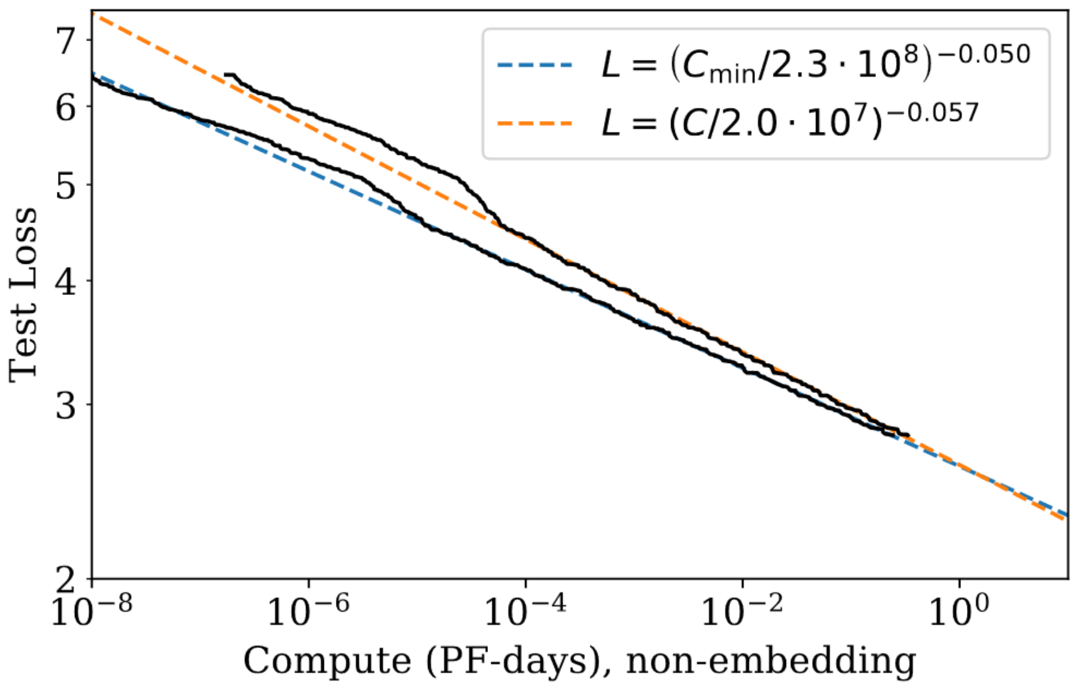


Figure 13 When adjusting performance to simulate training far below the critical batch size, we find a somewhat altered power law for $L(C_{\mathrm{min}})$ when compared with the fully empirical results. The conspicuous lump at 10^{-5} PF-days marks the transition from 1-layer to 2-layer networks; we exclude 1-layer networks in the power-law fits. It is the $L(C_{\mathrm{min}})$ trend that we expect to provide a reliable extrapolation for larger compute.

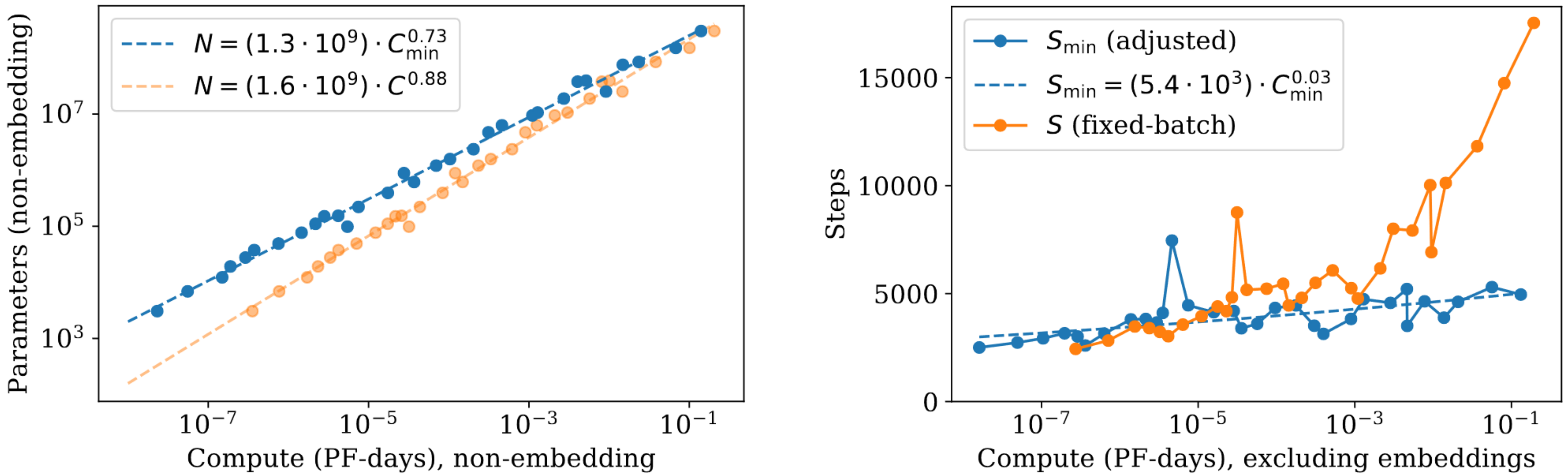


Figure 14 **Left:** Each value of the compute budget $C_{\min}$ has an associated optimal model size N. Optimal model size grows very rapidly with $C_{\min}$, increasing by 5x for each 10x increase in compute. The number of data examples processed makes up the remainder of the increase, growing relatively 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.

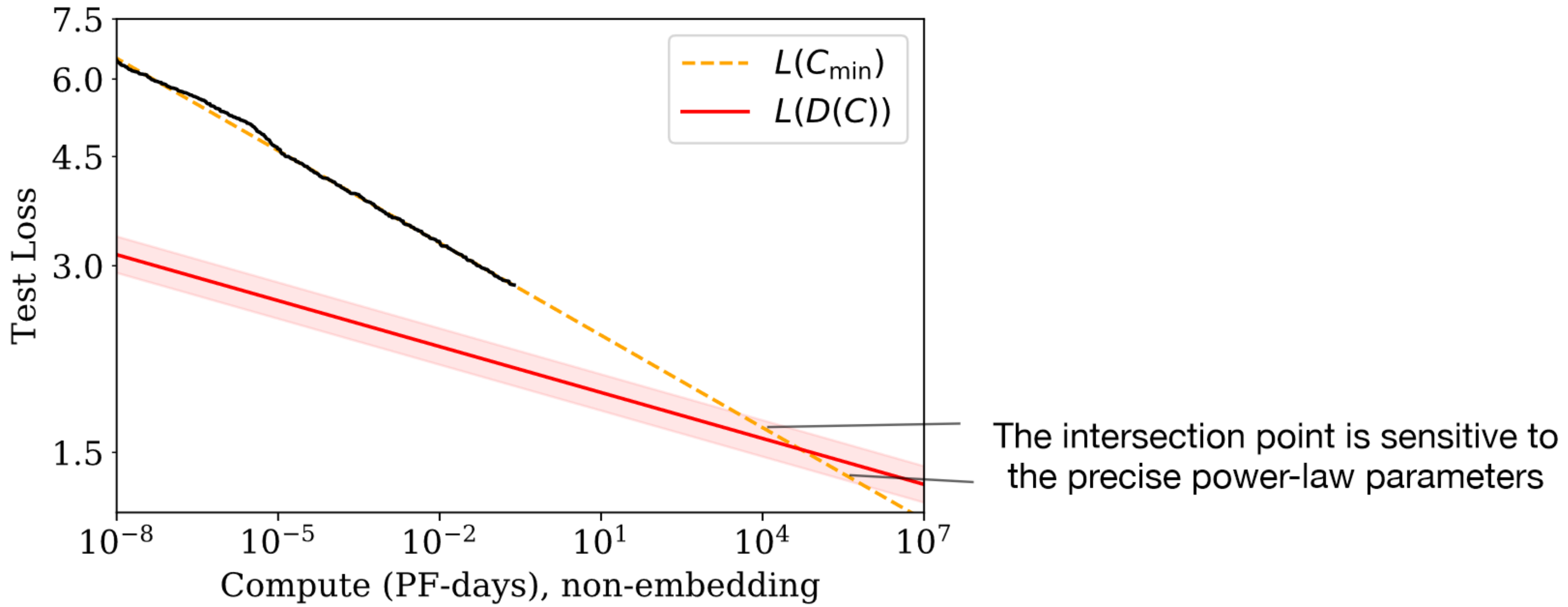


Figure 15 Far beyond the model sizes we study empirically, we find a contradiction between our equations for $L(C_{\min})$ and $L(D)$ due to the slow growth of data needed for compute-efficient training. The intersection marks the point before which we expect our predictions to break down. The location of this point is highly sensitive to the precise exponents from our power-law fits.