Scaling Laws for Neural Language Models

Ruixiao Wang
Jan 31, 2023

A study on language model performance

Cross entropy loss, Transformer architecture

Dataset: WebText2, vocabulary size = 50257

Summary

The test loss scales as a **power-law** with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude.

Experiment

A variety of models have been trained with different factors including:

- Model size (*N*): ranging in size from 768 to 1.5 billion non-embedding parameters.
- Dataset size (*D*): ranging from 22 million to 23 billion tokens.
- Model shape: including depth, width, attention heads, and feed-forward dimension.
- Context length: 1024 for most runs, with some experiments with shorter contexts.
- Batch size: 2^19 for most runs, with some variations to measure the critical batch size. Training at the critical batch size provides a roughly optimal compromise between time and compute efficiency.

Key Findings

- 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

Notations

- L – the cross entropy loss in nats. Typically it will be averaged over the tokens in a context, but in some cases we report the loss for specific tokens within the context.
- N – the number of model parameters, *excluding all vocabulary and positional embeddings*
- $C \approx 6NBS$ – 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). We quote numerical values in PF-days, where one PF-day $= 10^{15} \times 24 \times 3600 = 8.64 \times 10^{19}$ floating point operations.
- D – the dataset size in tokens
- B_{crit} – the critical batch size [MKAT18], defined and discussed in Section 5.1. Training at the critical batch size provides a roughly optimal compromise between time and compute efficiency.
- 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. This is also the number of training steps that would be used if the model were trained at a batch size much greater than the critical batch size.
- 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 $N, D, C, S, B, C^{\text{min}}$.

1. Performance depends strongly on scale, weakly on model shape

Scale:

- the number of model parameters N (excluding embeddings)
- the size of the dataset D
- the amount of compute C used for training

Shape:

- Depth
- width
- number of self-attention heads
- feed-forward dimension

Model Shape

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

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

n_{layer} : number of layers,

d_{model}: dimension of the residual stream,

d_{ff}: dimension of the intermediate feed-forward layer

n_{heads}: number of attention heads per layer

Train models with fixed size while varying a single hyperparameter.

This was simplest for the case of n_{heads}. When varying n_{layer}, we simultaneously varied d_{model} while keeping N fixed. Similarly, to vary d_{ff} at fixed model size we also simultaneously varied the d_{model} parameter,

We parameterize the Transformer architecture using hyperparameters n_{layer} (number of layers), d_{model} (dimension of the residual stream), d_{ff} (dimension of the intermediate feed-forward layer), d_{attn} (dimension of the attention output), and n_{heads} (number of attention heads per layer). We include n_{ctx} tokens in the input context, with $n_{\text{ctx}} = 1024$ except where otherwise noted.

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

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.

Model Shape

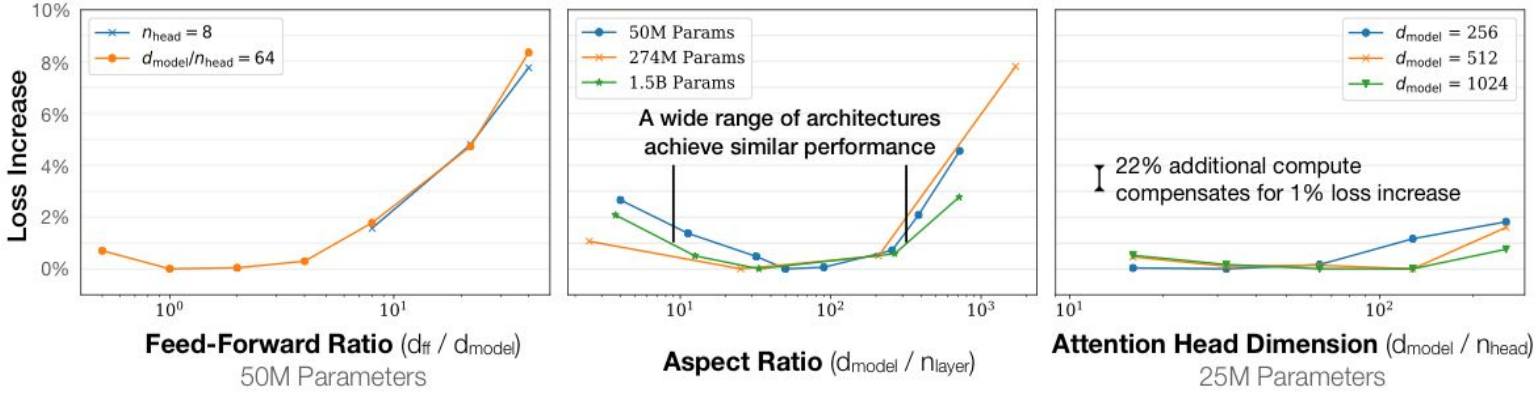


Figure 5 Performance depends very mildly on model shape when the total number of non-embedding parameters N is held fixed. The loss varies only a few percent over a wide range of shapes. Small differences in parameter counts are compensated for by using the fit to $L(N)$ as a baseline. Aspect ratio in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

n_{layer} : number of layers, **d_{model}:** dimension of the residual stream, **d_{ff}**: dimension of the intermediate feed-forward layer

n_{heads}: number of attention heads per layer, **N**: model size

Model Shape

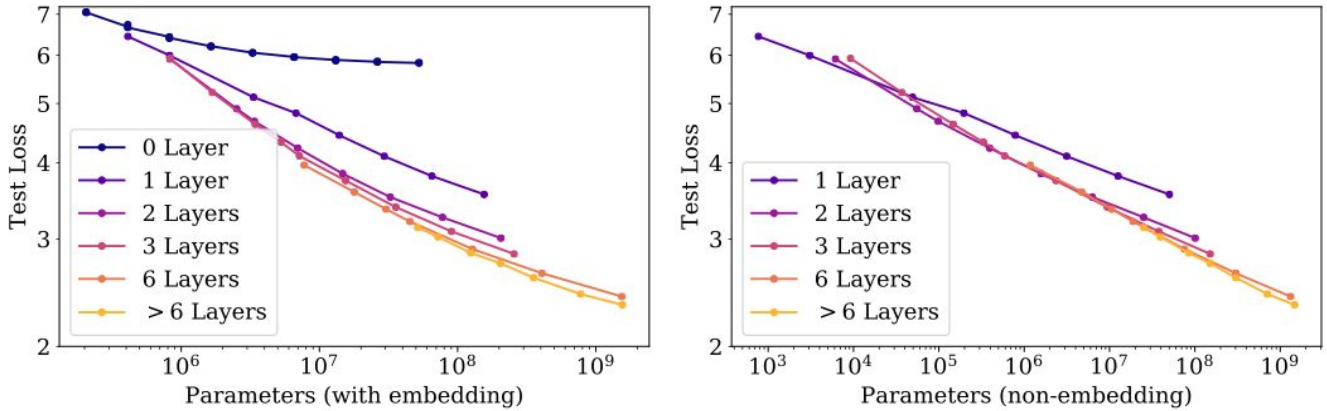


Figure 6 **Left:** When we include embedding parameters, performance appears to depend strongly on the number of layers in addition to the number of parameters. **Right:** When we exclude embedding parameters, the performance of models with different depths converge to a single trend. Only models with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

Model Scale

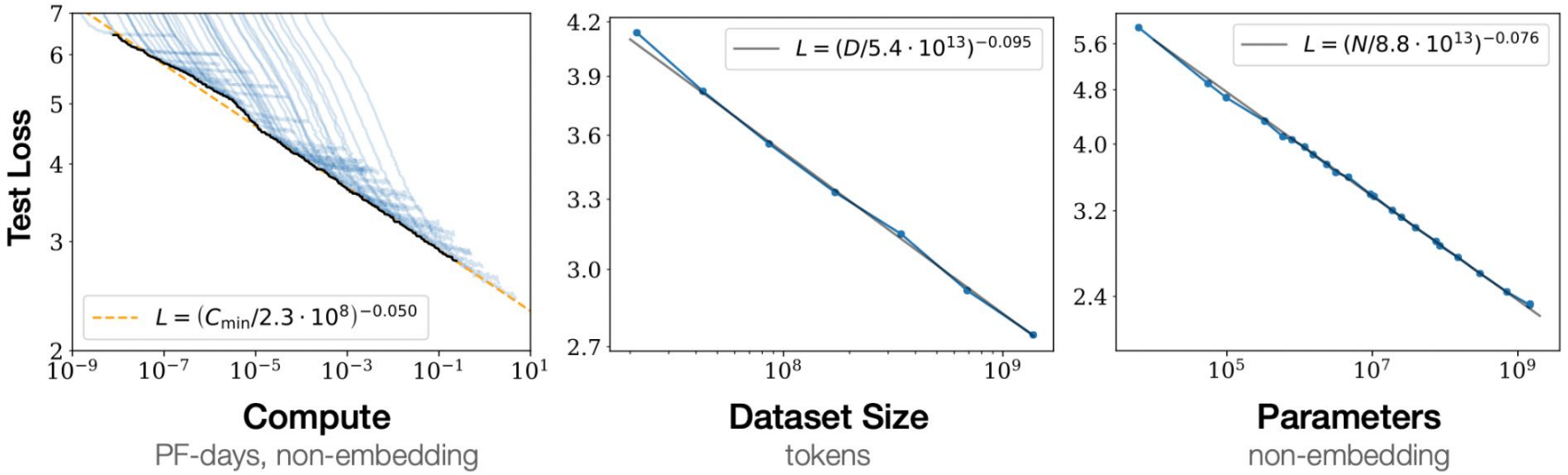


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

2. Smooth power laws

Performance has a power-law relationship with each of the three scale factors N, D, C when not bottlenecked by the other two, with trends spanning more than six orders of magnitude. We observe no signs of deviation from these trends on the upper end, though performance must flatten out eventually before reaching zero loss.

$$L(C) \approx \left(\frac{C_c}{C}\right)^{\alpha_C} \qquad L(D) \approx \left(\frac{D_c}{D}\right)^{\alpha_D} \qquad L(N) \approx \left(\frac{N_c}{N}\right)^{\alpha_N}$$

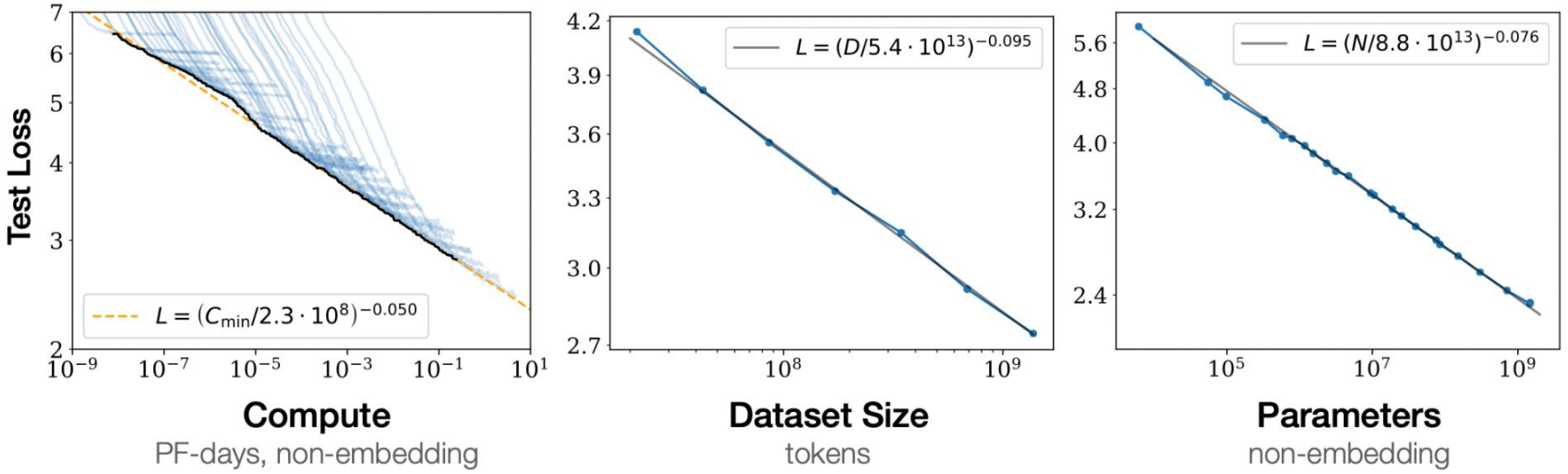


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when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

3. Transfer improves with test performance

When we evaluate models on text with a different distribution than they were trained on, the results are strongly correlated to those on the training validation set with a roughly constant offset in the loss – in other words, transfer to a different distribution incurs a constant penalty but otherwise improves roughly in line with performance on the training set

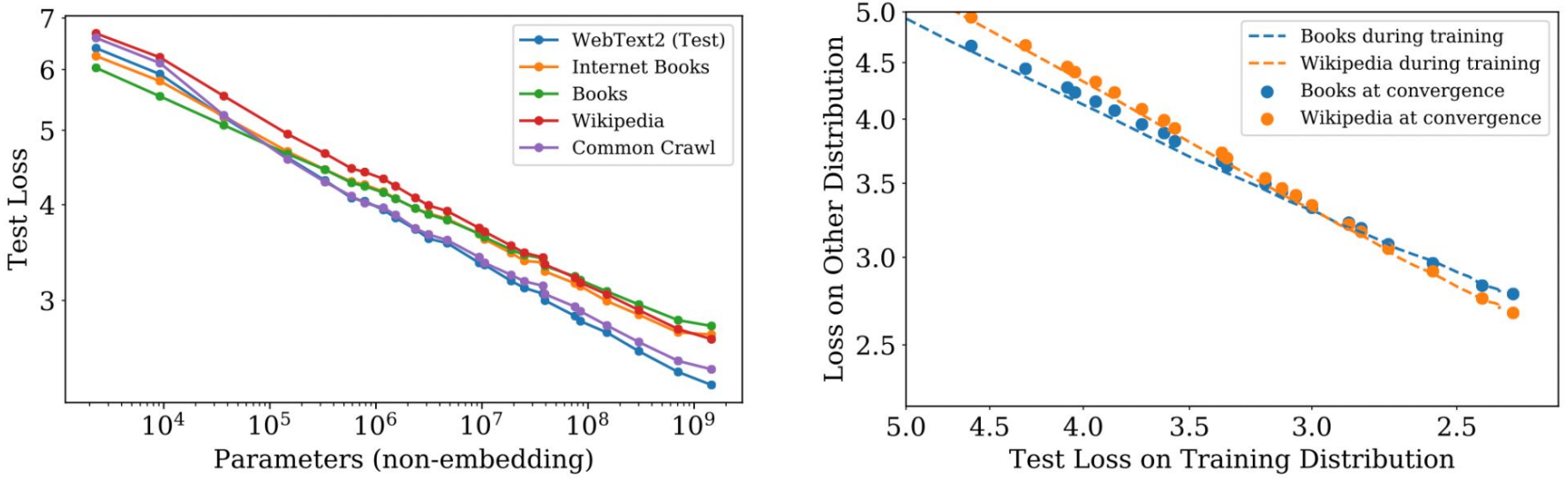


Figure 8 **Left:** Generalization performance to other data distributions improves smoothly with model size, with only a small and very slowly growing offset from the WebText2 training distribution. **Right:** Generalization performance depends only on training distribution performance, and not on the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

4. Universality of overfitting

Performance improves predictably as long as we scale up N and D in tandem, but enters a regime of diminishing returns if either N or D is held fixed while the other increases. The performance penalty depends predictably on the ratio N^0.74/D, meaning that every time we increase the model size 8x, we only need to increase the data by roughly 5x to avoid a penalty.

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

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

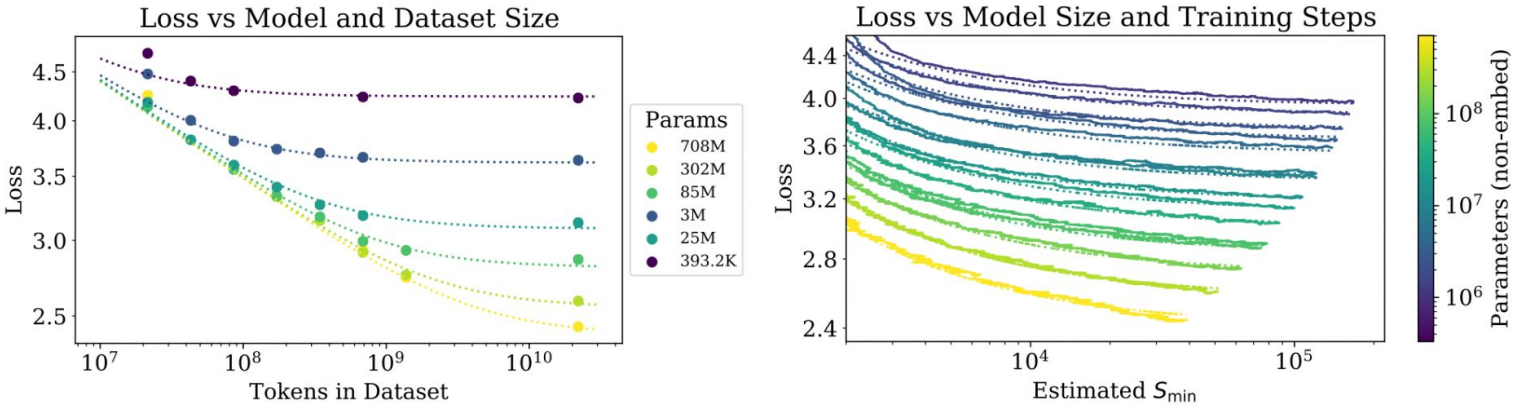


Figure 4 **Left**: The early-stopped test loss $L(N, D)$ varies predictably with the dataset size D and model size N according to Equation (1.5). **Right**: After an initial transient period, learning curves for all model sizes N can be fit with Equation (1.6), which is parameterized in terms of $S_{\min}$,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

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

Three principles:

1. Changes in vocabulary size or tokenization are expected to rescale the loss by an overall factor. The parameterization of L(N, D) (and all models of the loss) must naturally allow for such a rescaling.

2. Fixing D and sending N → ∞, the overall loss should approach L(D). Conversely, fixing N and sending D → ∞ the loss must approach L(N).

3. L(N, D) should be analytic at D = ∞, so that it has a series expansion in 1/D with integer powers. Theoretical support for this principle is significantly weaker than for the first two.

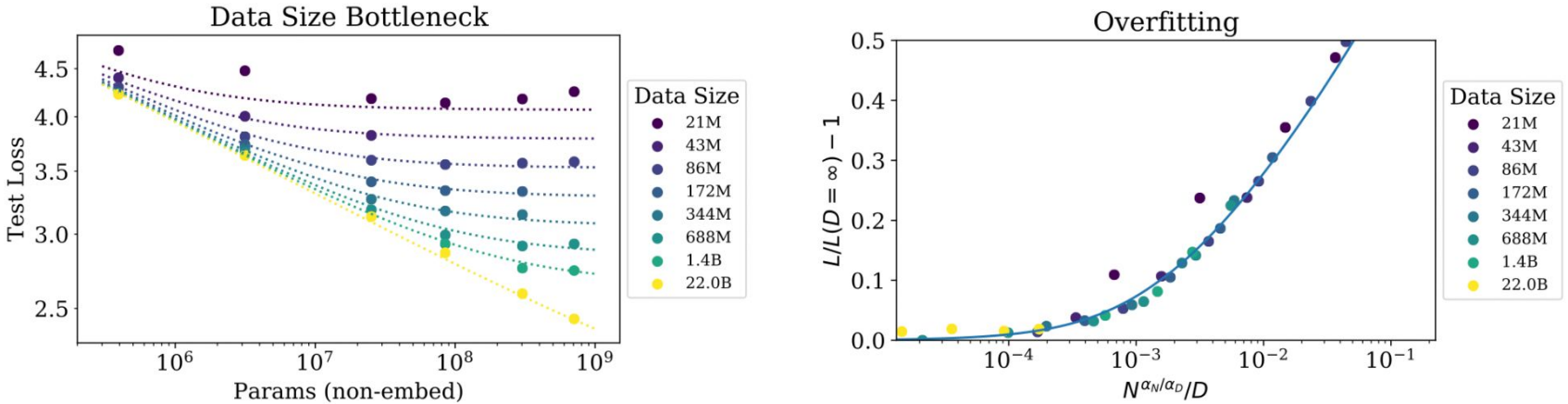


Figure 9 The early-stopped test loss $L(N, D)$ depends predictably on the dataset size D and model size N according to Equation (1.5). **Left**: For large D, performance is a straight power law in N. For a smaller fixed D, performance stops improving as N increases and the model begins to overfit. (The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

We estimate that the variation in the loss with different random seeds is roughly 0.02, which means that to avoid overfitting when training to within that threshold of convergence we require

$$D \gtrsim (5 \times 10^3) N^{0.74} \tag{4.4}$$

5. Universality of Training

Training curves follow predictable power-laws whose parameters are roughly independent of the model size. By extrapolating the early part of a training curve, we can roughly predict the loss that would be achieved if we trained for much longer.

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

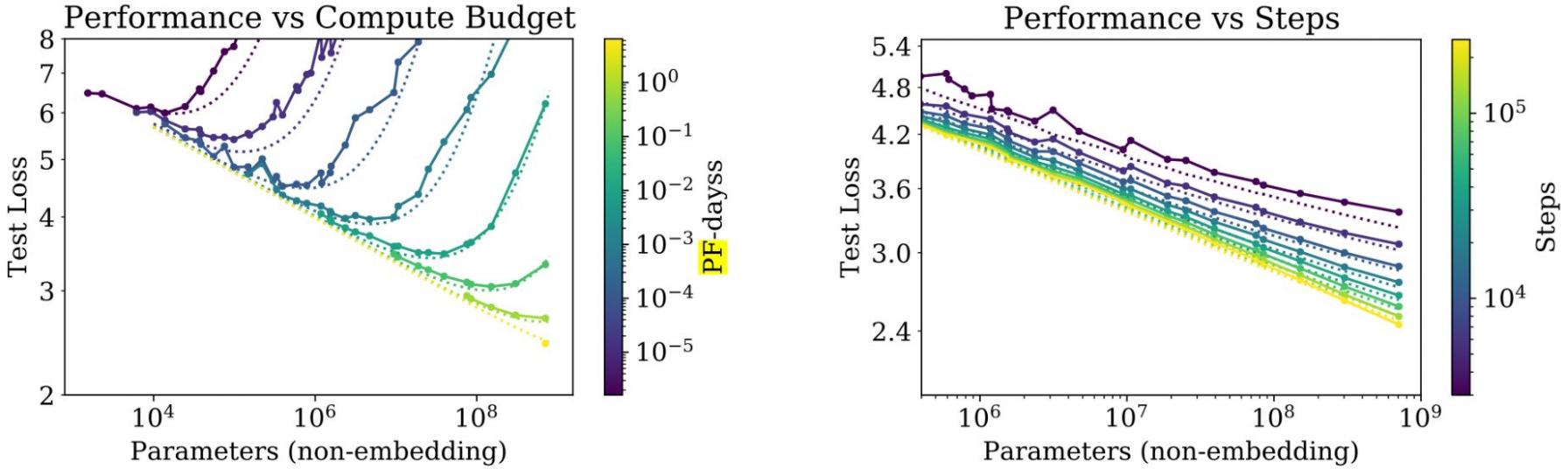
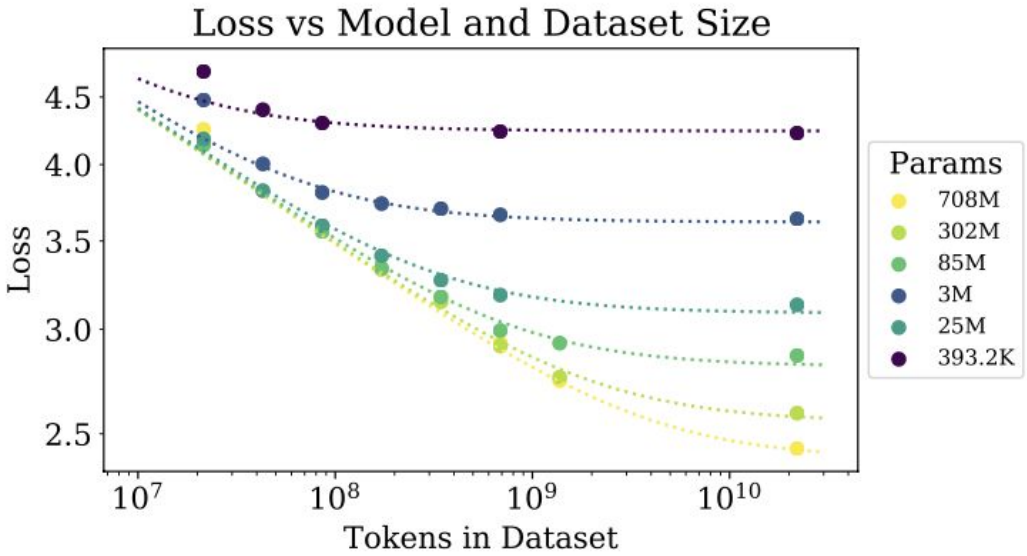


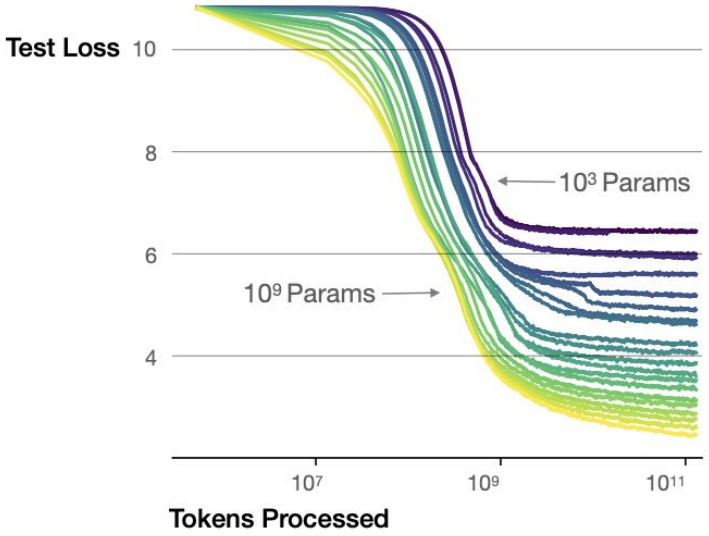
Figure 11 When we hold either total compute or number of training steps fixed, performance follows $L(N, S)$ from Equation (5.6). Each value of compute budget has an associated optimal model size that maximizes performance. Mediocre fits at small S are unsurprising, as the power-law equation for the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

6. Sample efficiency

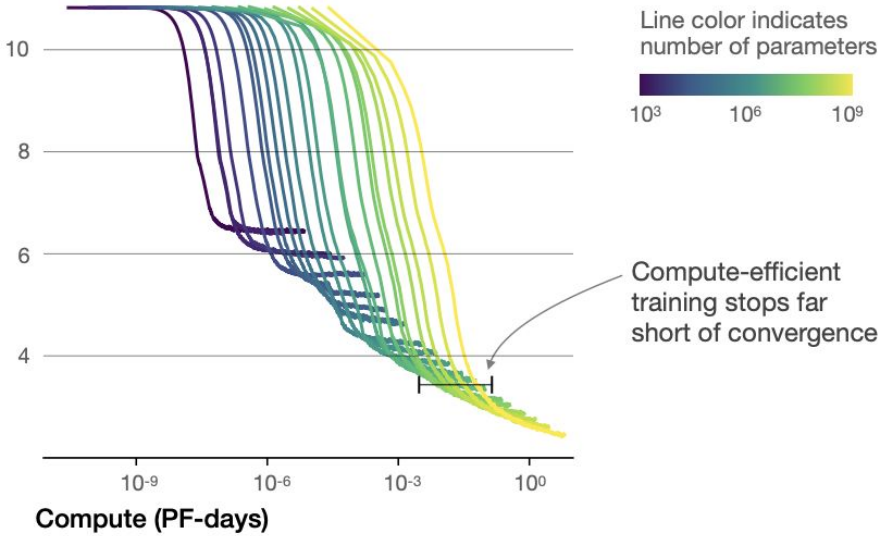
Large models are more sample-efficient than small models, reaching the same level of performance with fewer optimization steps and using fewer data points.



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



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



7. Convergence is inefficient

When working within a fixed compute budget *C* but without any other restrictions on the model size *N* or available data *D*, we attain optimal performance by training very large models and stopping significantly short of convergence.

$$N(C_{\min}) \propto (C_{\min})^{0.73}$$

$$S_{\min} \propto (C_{\min})^{0.03}$$

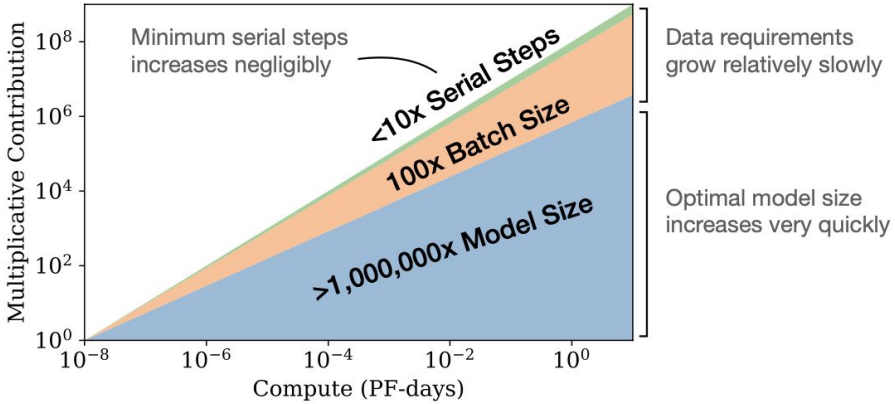


Figure 3 As more compute becomes available, we can choose how much to allocate towards training larger models, using larger batches, and training for more steps. We illustrate this for a billion-fold increase in compute. For optimally compute-efficient training, most of the increase should go towards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

As we scale up language modeling with an optimal allocation of computation, we should predominantly increase the model size N, while simultaneously scaling up the batch size via B $\propto$ Bcrit with negligible increase in the number of serial steps.

8. Optimal batch size

The ideal batch size for training these models is roughly a power of the loss only; It is roughly 1-2 million tokens at convergence for the largest models we can train.

It was argued that there is a critical batch size B_{crit} for training; for B up to B_{crit} the batch size can be increased with very minimal degradation in compute-efficiency, whereas for $B > B_{crit}$ increases in B result in diminishing returns. It was also argued that the gradient noise scale provides a simple prediction for B_{crit}, and that neither depends directly on model size except through the value of the loss that has been attained. These results can be used to predict how training time and compute will vary with the batch size.

$$\left(\frac{S}{S_{\min}}-1\right)\left(\frac{E}{E_{\min}}-1\right)=1$$

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

We will use $B_{\rm crit}(L)$ to estimate the relation between the number of training steps S while training at batch size $B=2^{19}$ tokens and the number of training steps while training at $B\gg B_{\rm crit}$. This is simply

$$S_{\min}(S)\equiv\frac{S}{1+B_{\rm crit}(L)/B}\qquad\text{(minimum steps, at } B\gg B_{\rm crit}\text{)}\tag{5.4}$$

for any given target value L for the loss. This also defines a critical value of the compute needed to train to L with a model of size N if we were to train at $B\ll B_{\rm crit}(L)$. This is

$$C_{\min}(C)\equiv\frac{C}{1+B/B_{\rm crit}(L)}\qquad\text{(minimum compute, at } B\ll B_{\rm crit}\text{)}\tag{5.5}$$

where $C=6NBS$ estimates the (non-embedding) compute used at batch size B.

The results for $L(N, S_{\min})$ can be used to derive a lower-bound (and rough estimate) of the step at which early stopping should occur when training is data limited. It is motivated by the idea that finite and infinite D learning curves for a given model will be very similar until we reach $S_{\min} \approx S_{\text{stop}}$. Thus overfitting should be proportional to the correction from simply ending training at S_{stop}. This will underestimate S_{stop}, because in reality the test loss will decrease more slowly when we have a finite D, and therefore we will require more training steps to reach the optimal test loss at finite D. This line of reasoning leads to the inequality

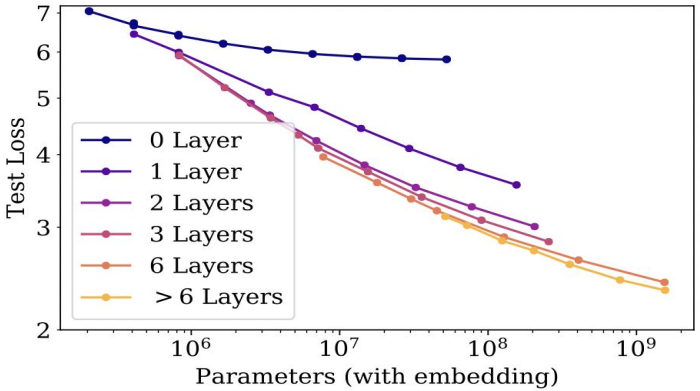
$$S_{\text{stop}}(N, D) \gtrsim \frac{S_c}{[L(N, D) - L(N, \infty)]^{1/\alpha_S}} \tag{5.7}$$

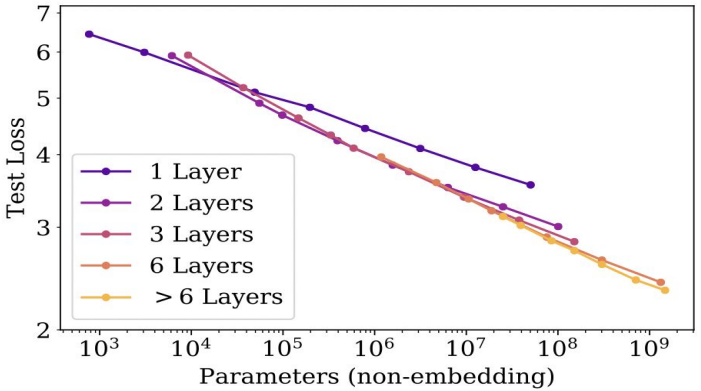
Summary

These results show that language modeling performance improves smoothly and predictably as we appropriately scale up model size, data, and compute. We expect that larger language models will perform better and be more sample efficient than current models.

Discussion

In experiments with large empirical datasets, the power law becomes obscure if we consider all the parameters (including embedding).





- Why not taking all the parameters (including embedding) into consideration?

Discussion

- Is it possible to generate the power law relation to other models? What other models you want to test on?

Discussion

- What are some caveats/suspicious parts in the power law conclusion in this paper?