

Lecture 3: Scaling Laws

Allen Liu | CSCI-GA.3033-141

September 22, 2026

1 Pretraining and the decisions before training

Lecture 2 described the architecture of a language model. We now ask *how to make modeling and data decisions before an expensive pretraining run*.

Recall that pretraining fits a model to predict the next token. For tokens $t_1, \dots, t_T$, the training objective is

$$L_{\text{train}}(\theta) = -\frac{1}{T} \sum_{i=1}^T \log p_{\theta}(t_i | t_1, \dots, t_{i-1}).$$

Here θ contains the learned weights, p_{θ} is the predicted probability, and $\log$ is the natural logarithm. Training takes gradient steps on this objective. We evaluate the trained model on new text and write L for its test loss.

Before training, we must choose an architecture, model size, dataset, and training duration. Trying every choice at full scale would be too expensive. Our goal is to predict the results of large runs well enough to make these decisions in advance. For simplicity, we will focus on three quantities:

N = number of model parameters,

D = number of tokens processed during training,

C = training compute, measured in floating-point operations (FLOPs).

A larger model costs more per token, and total compute is roughly proportional to ND . Under a fixed compute budget, increasing N therefore leaves less compute for processing data. Our main question is: *which choices of N and D give the smallest test loss at a fixed compute budget?*

2 Scaling laws as an experimental method

To make these choices in advance, we need to predict the results of larger runs from affordable experiments. *How can measurements at small scales guide decisions at large scales?*

The method is to train models at several scales, measure their losses, and fit a relationship between loss and resources. Such a relationship is called a *scaling law*. When comparing candidate architectures or training recipes, fit a curve for each under comparable data and evaluation conditions. Test the fitted curves on additional runs before using them to select a much larger run.

A common form, when data and optimization are sufficient, is

$$L(N) \approx L_{\infty} + \frac{A}{N^{\alpha}}, \quad A > 0, \quad \alpha > 0.$$

The constant L_{∞} is the loss floor for this setup. The remaining term is the error reduced by increasing model size. The exponent α controls how quickly that error falls: doubling N multiplies the excess loss by $2^{-\alpha}$. On a log-log plot of excess loss against N , this gives a line with slope $-\alpha$.

The resource being varied must actually limit performance. To measure the effect of model size, provide enough data and optimization that they do not hide its benefit. To measure the effect of data, use sufficient model capacity and tune the fitting procedure for the available data. For a compute budget, compare allocations with the same total cost. These distinctions will organize our calculations.

Figure 1 shows this pattern in language-model experiments. Focus on the middle and right panels: the measured losses closely follow the fitted power laws as data or model size increases. The exponents are estimated from the experiments.

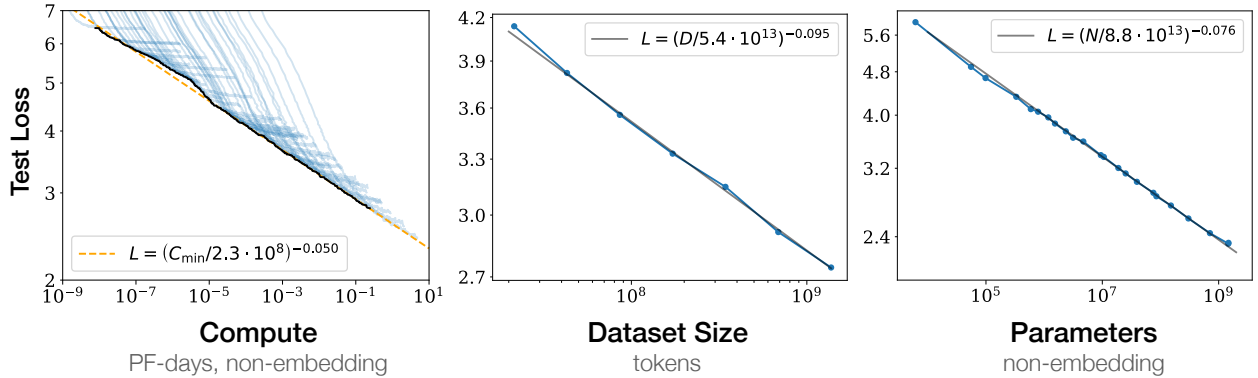


Figure 1: Empirical language-model scaling. Reproduced from Figure 1 of [Kaplan et al., *Scaling Laws for Neural Language Models* \(2020\)](#). The axes are logarithmic. The vertical axis shows test loss L ; these fits use no additive loss floor over the measured range. Here D counts dataset tokens and N excludes embedding parameters. The left panel uses compute estimates adjusted for batch size; PF-days means petaflop-days.

3 What simple examples can explain

Fitting a power law tells us how loss changes in an experiment. Our next question is *why a power law might appear, and what determines its exponent*.

We will study simple prediction problems where we can calculate error bounds.¹ Two effects will matter. A limited model cannot represent all the detail in the target; this gives *approximation error*. A finite dataset cannot determine the model’s predictions exactly; this gives *estimation error*.

We will first isolate each limitation, then examine the tradeoff between them. This will produce a model-size scaling law and a data scaling law. Smoothness and input geometry will explain why their exponents change. Finally, we will use an empirical language-model law to solve the compute-allocation problem from Section 1.

4 A common regression setup

We need a prediction problem in which model size and data size have clear meanings. The task is to learn an unknown function f from noisy observations and predict its output at a new input.

¹For a broader discussion of mechanisms behind scaling laws, see Bahri et al., [Explaining Neural Scaling Laws](#) (2024).

The local-averaging approach below is standard in nonparametric regression; see [Krishnamurthy's Lecture 7 \(2017\)](#) for background.

Draw D independent training pairs (x_i, y_i) with

$$x_i \sim \text{Unif}([0, 1]^d), \quad y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, 1).$$

The noises are independent of each other and of the inputs. Here D counts training examples, N counts fitted model parameters, and d is the input dimension. We use D for data size throughout: examples in regression and tokens in language-model training.

Let $\hat{f}$ be the fitted predictor. For a new independent pair (x, y) , its squared prediction loss decomposes as

$$L = \mathbb{E}[(y - \hat{f}(x))^2] = 1 + \mathbb{E}[(f(x) - \hat{f}(x))^2].$$

Fresh observation noise contributes the loss floor $L_\infty = 1$. The excess loss $L - 1$ measures how well we have learned f . We write $L(N, D)$ when tracking its dependence on both model size and sample count, averaging the test loss over the sampled examples.

For the function-learning examples, assume

$$|f(x)| \leq 1, \quad |f(x) - f(z)| \leq \|x - z\|.$$

The second condition is *Lipschitz continuity*: nearby inputs have nearby target values. The norm is Euclidean distance. This assumption lets local information predict nearby outputs; Section 6 will examine what happens when the smoothness or geometry changes.

5 Scaling in different regimes

With the prediction problem fixed, we can ask *which resource is limiting the loss*. In [Scaling Laws for Neural Language Models](#) (2020), Kaplan et al. distinguish limits from model size, data, and compute. We will use regression to understand the first two, and return to compute in Section 7.

5.1 A fixed model with increasing data

Start with a constant target, $f(x) = \mu$, and a model that fits one constant. This is the $N = 1$ setting: the model has just one fitted parameter. The fitted prediction is the sample mean. Since the label noise has variance one,

$$\hat{\mu} = \frac{1}{D} \sum_{i=1}^D y_i, \quad \mathbb{E}[(\hat{\mu} - \mu)^2] = \frac{1}{D}, \quad L(1, D) = 1 + \frac{1}{D}.$$

This familiar averaging result is our baseline. More data estimates a fixed quantity more accurately, giving excess loss proportional to $1/D$. We now examine what changes when the target contains more detail than one constant can represent.

5.2 Limited model size, abundant data

Allow a general target from Section 4, and assume enough data to fit the best model of each size. The question is how much error remains because we have only N parameters.

Divide each input coordinate into intervals of width h , with $1/h$ an integer. Use one fitted constant per grid cell. The number of cells and parameters is

$$N = h^{-d} \quad \implies \quad h = N^{-1/d}.$$

With unlimited data, the best constant in a cell is the average target value there. Any two points in the cell are at most $\sqrt{d}h$ apart, so the Lipschitz condition limits the variation of f to that amount. Its squared approximation error is at most dh^2 . Hence

$$L(N, \infty) - 1 \leq dh^2 = dN^{-2/d}.$$

The notation $L(N, \infty)$ denotes the loss with the best cell constants, as if they were known from unlimited data.

The exponent $2/d$ comes from two facts: squared error falls as the square of cell width, while the number of cells grows as its inverse d th power. The exponent is 2 in one dimension, 1 in two dimensions, and $2/3$ in three dimensions.

This bound has the right order for the grid model. For example, when $f(x) = x_1$, the optimal cell constant is the midpoint of the first coordinate interval, and its squared error is proportional to h^2 . A linear model could represent that target exactly. Model-size scaling depends on how the model represents the target.

5.3 Limited data, sufficient model capacity

Now fix a finite dataset of D examples. Suppose model capacity is large enough that we can choose the grid resolution freely. The question is how much detail the data support, and how the resulting loss falls as D increases.

Fit each cell's constant by averaging its training labels. Predict zero in an empty cell. There are two costs. Averaging within a cell leaves approximation error of order h^2 . Each cell receives about $Dh^d = D/N$ examples, so estimating its mean from unit-variance noise costs roughly $1/(Dh^d) = N/D$.

Accounting for the random counts and empty cells gives

$$L(N, D) - 1 \lesssim \underbrace{N^{-2/d}}_{\text{approximation error}} + \underbrace{\frac{N}{D}}_{\text{estimation error}}.$$

Here $\lesssim$ hides a constant depending on the fixed dimension d , but not on N or D . Appendix A verifies the sample-count step. Smaller cells capture finer detail but contain fewer observations to average.

Choose the model size by balancing the orders of the two errors:

$$N^{-2/d} = \frac{N}{D} \implies N^{1+2/d} = D \implies N \asymp D^{d/(d+2)}.$$

The symbol $\asymp$ means equality up to fixed positive factors; rounding the grid size changes only these factors. Substituting this choice gives the data scaling law

$$L(N, D) - 1 \lesssim D^{-2/(d+2)}.$$

In one dimension, $N \asymp D^{1/3}$ gives the rate $D^{-2/3}$. Each estimated mean still benefits from averaging, but the number of means grows as we learn finer detail. This is why the overall rate can be slower than $1/D$.

At fixed N , the estimation term falls as $1/D$ and approximation error remains. The fractional data exponent describes learning more detail as data grows. Sufficient capacity lets us choose that detail; it does not require fitting the finest possible grid to every dataset.

For this bounded Lipschitz class and Gaussian observation model, the rate also matches the best guarantee any estimator can provide uniformly over the class (see [Stone, 1982](#)). Individual easier targets can be learned faster.

5.4 What the two laws tell us

With abundant data, approximation limits how much a model of size N can learn. With sufficient capacity, the amount of data limits how much detail we can estimate reliably. The two power laws arise from the same prediction problem, but describe different constraints.

The compute-limited problem couples these resources: improving one leaves less budget for the other. Before solving that allocation for language models, we examine why the exponents themselves can vary.

6 Smoothness, intrinsic dimension, and the exponents

The preceding rates depended on how quickly approximation improves and how much data falls in a small region. We now ask *how changing these properties changes the exponents*, including how noninteger exponents arise.

6.1 Smoothness changes the approximation rate

The Lipschitz calculation used a constant in each cell. A smoother function can be approximated more accurately by a local polynomial. *What does smoothness mean, and how does it control the error?*

First take an integer $s \geq 1$. We assume that f has continuous, bounded derivatives through order s . To see the consequence in one dimension, normalize the highest derivative bound to $|f^{(s)}| \leq 1$. The degree- $(s-1)$ Taylor polynomial around a point z is

$$p_z(x) = \sum_{j=0}^{s-1} \frac{f^{(j)}(z)}{j!} (x-z)^j, \quad f^{(0)} = f.$$

Taylor's theorem bounds the remainder. Within an interval of width h ,

$$|f(x) - p_z(x)| \leq \frac{|x-z|^s}{s!} \leq \frac{h^s}{s!}.$$

Thus the derivative bound gives approximation error of order at most h^s , and squared error of order at most h^{2s} . For $s = 1$, we use a constant; for $s = 2$, we use a line and control its error by the second derivative.

For noninteger smoothness, write $s = k + r$, where $k \geq 0$ is an integer and $0 < r < 1$. Assume bounded continuous derivatives through order k , and require

$$|f^{(k)}(x) - f^{(k)}(z)| \leq |x - z|^r.$$

This is called *Hölder continuity* of the k th derivative. It controls how quickly that derivative changes. When $k = 0$, this condition directly bounds the error of a constant approximation by h^s . When $k = 1$, subtracting the local line gives

$$\begin{aligned} |f(x) - f(z) - f'(z)(x - z)| &= \left| \int_z^x (f'(t) - f'(z)) dt \right| \\ &\leq \int_0^{|x-z|} t^r dt = \frac{|x - z|^{1+r}}{1+r}. \end{aligned}$$

For larger k , repeated integration gives error of order at most $h^{k+r} = h^s$ for the degree- k Taylor polynomial.

In d dimensions, impose these bounds on all partial derivatives, including mixed derivatives, using Euclidean distance in the Hölder condition. The multivariable Taylor expansion then gives the same h^s approximation bound on a cell of width h , up to constants depending on d and s .

For fixed d and s , each cell needs a fixed number of polynomial coefficients, so $N \asymp h^{-d}$. We therefore obtain

$$\boxed{L(N, \infty) - 1 \lesssim h^{2s} \asymp N^{-2s/d}.}$$

With finite data, estimating the local polynomial coefficients gives the same approximation–estimation balance as before:

$$L - 1 \lesssim h^{2s} + \frac{1}{Dh^d}.$$

Equating the terms gives

$$h^{2s+d} \asymp \frac{1}{D}, \quad \boxed{L - 1 \lesssim D^{-2s/(2s+d)}}$$

at the chosen resolution. Even with integer input dimension d , the exponent varies continuously with smoothness s .

6.2 Local probability mass determines the relevant dimension

For uniform data on a d -dimensional cube, a neighborhood of radius h contains probability mass of order h^d . This is why the local sample count was of order Dh^d .

Now allow a different input distribution. Suppose, uniformly over its support at small scales,

$$\Pr(\|x_i - x\| \leq h) \asymp h^{d_{\text{eff}}}.$$

The exponent d_{eff} is an *intrinsic dimension*. It describes how probability mass grows with radius and need not count coordinates or be an integer.

For a Lipschitz target, averaging labels within radius h still gives squared approximation error of order h^2 , but the neighborhood now contains about $Dh^{d_{\text{eff}}}$ examples. The same balance yields

$$L - 1 \lesssim h^2 + \frac{1}{Dh^{d_{\text{eff}}}} \implies \boxed{L - 1 \lesssim D^{-2/(d_{\text{eff}}+2)}}.$$

Thus local sampling geometry, as well as target smoothness, changes the data exponent. For a general treatment using nearest neighbors, see [Kpotufe \(2011\)](#).

6.3 The same smoothness constraint in a feature representation

The grid examples represent detail through small regions of input space. We can also represent it through high-frequency features. This gives another way to see how smoothness limits the error as model size grows.

Return to the one-dimensional, uniform-input setup and the bounded 1-Lipschitz target. Use the orthonormal cosine features

$$\phi_0(x) = 1, \quad \phi_j(x) = \sqrt{2} \cos(\pi j x) \quad (j \geq 1),$$

and expand

$$f(x) = \sum_{j=0}^{\infty} a_j \phi_j(x), \quad a_j = \int_0^1 f(x) \phi_j(x) dx.$$

An N -parameter model retains the first N features and learns their coefficients. With unlimited data, the best coefficients are $a_0, \dots, a_{N-1}$. Orthogonality makes the omitted squared errors add:

$$L(N, \infty) - 1 = \sum_{j=N}^{\infty} a_j^2.$$

Differentiation multiplies a cosine of frequency j by πj , and the derivative waves are orthogonal too. Since $|f'| \leq 1$ almost everywhere,

$$\boxed{\sum_{j=1}^{\infty} (\pi j)^2 a_j^2 = \int_0^1 |f'(x)|^2 dx \leq 1.}$$

The series identity holds in the mean-square sense. High-frequency signal contributes more to the squared derivative, so smoothness limits how much of it the target can contain. Since every omitted frequency has $j \geq N$,

$$L(N, \infty) - 1 = \sum_{j=N}^{\infty} a_j^2 \leq \frac{1}{(\pi N)^2} \sum_{j=N}^{\infty} (\pi j)^2 a_j^2 \leq \boxed{\frac{1}{\pi^2 N^2}}.$$

We recover the one-dimensional approximation rate through a constraint on feature energy. The rate follows from smoothness; we did not need to assume an exact power law for each coefficient.

7 Chinchilla and compute-optimal allocation

The regression calculations give error bounds for specific prediction problems. They explain how smoothness, geometry, and finite data can produce power-law rates. For language models, the evidence comes from experiments such as Figure 1: power laws fit loss well across the measured ranges of model sizes and data. Their exponents are fitted from those measurements; our regression calculations do not determine them.

We now use an empirical language-model fit to answer our opening question: *given a compute budget, how should we choose model size and training duration?*

7.1 Fit the effects of model size and training data

Recall that N counts model parameters and D counts tokens processed during training. A loss model used in the [Chinchilla study](#) (Hoffmann et al., 2022) is

$$L(N, D) = L_\infty + \frac{A}{N^\alpha} + \frac{B}{D^\beta}, \quad A, B, \alpha, \beta > 0.$$

The decreasing terms describe the effects of finite model size and training data. All constants are fitted for a particular dataset, model family, and training procedure. Appendix D.2 of the original study reports the fit

$$L(N, D) \approx 1.69 + \frac{406.4}{N^{0.34}} + \frac{410.7}{D^{0.28}},$$

using raw parameter and token counts and natural-log loss. The exponents $\alpha = 0.34$ and $\beta = 0.28$ are empirical estimates for that setup.

At fixed D , increasing N approaches the floor $L_\infty + B/D^\beta$. This is the floor to subtract when making a log-log plot of the model-size term. Holding N fixed gives the analogous floor for a data-scaling curve.

7.2 Impose the compute budget

For dense models dominated by learned matrix operations, use the training-cost approximation

$$C \approx 6ND.$$

Treating this as the budget constraint gives $D = C/(6N)$. Substitution reduces the allocation problem to one variable:

$$L\left(N, \frac{C}{6N}\right) - L_\infty = AN^{-\alpha} + B\left(\frac{6N}{C}\right)^\beta.$$

The model-size term falls with N . The data term rises because a larger model receives fewer training tokens under the same budget.

7.3 Balance the marginal improvements

Treat N as continuous. A small fractional increase in N forces the same fractional decrease in D . At the optimum, the benefit from model size must balance the cost of reduced training. Setting the derivative with respect to $\log N$ to zero gives

$$-\frac{\alpha A}{N^\alpha} + \frac{\beta B}{D^\beta} = 0 \quad \implies \quad \frac{\alpha A}{N^\alpha} = \frac{\beta B}{D^\beta}.$$

Thus we balance the improvements per fractional increase in each resource. The two loss terms themselves are equal only when $\alpha = \beta$.

Solving this balance together with $6ND = C$ gives

$$N^* = \left(\frac{\alpha A}{\beta B}\right)^{1/(\alpha+\beta)} \left(\frac{C}{6}\right)^{\beta/(\alpha+\beta)}, \quad D^* = \frac{C}{6N^*}.$$

The first marginal term decreases with N , while the second increases, so this is the unique minimum. In particular,

$$N^* \propto C^{\beta/(\alpha+\beta)}, \quad D^* \propto C^{\alpha/(\alpha+\beta)}.$$

The proportionality constants are independent of C . The two exponents sum to one, as required by the compute constraint. If $\alpha = \beta$, model size and token count both grow as $\sqrt{C}$.

Substituting the optimal sizes into either loss term gives a compute exponent of $\alpha\beta/(\alpha + \beta)$. Adding the terms preserves it:

$$L(N^*, D^*) - L_\infty \propto C^{-\alpha\beta/(\alpha+\beta)}.$$

The fitted law predicts both how to allocate additional compute and how much that compute can reduce the loss.

7.4 Check the prediction with equal-compute runs

Choose a budget and train several model sizes, each on $D = C/(6N)$ tokens. Plot test loss against N . This is an *isoFLOP curve*: every run has the same compute cost. Its minimum estimates the best model size for that budget. Repeat at several budgets to test how the best size grows.

Keep the tokenizer, data distribution, and evaluation procedure comparable, and tune each learning-rate schedule to its intended duration. Check the fitted law on additional runs before using it to choose the model size and token count for a much larger run.

7.5 Why many models are trained beyond the Chinchilla optimum

The Chinchilla calculation minimizes pretraining loss for a fixed *pretraining* compute budget. Many modern models are deliberately *overtrained* relative to this rule: they use a smaller N and a larger D than the pretraining-optimal allocation for the same budget. Their test loss can still improve with additional training.

Why can this be a good use of compute? Pretraining is followed by post-training and deployment. Within a model family, a smaller model costs less per generated token. Spending more on pretraining to reach a desired quality with fewer parameters can therefore save compute across many future requests. The [Llama 3 report](#) (Dubey et al., 2024) explicitly motivates training its smaller models longer by their better performance at a fixed inference budget. This tradeoff is analyzed in [Beyond Chinchilla-Optimal](#) (Sardana et al., 2024).

The same cost argument can apply during post-training. For example, reinforcement learning generates candidate responses, scores them, and updates the model. A smaller model makes both generation and updates cheaper, potentially allowing more learning within the same budget. At deployment, long reasoning responses or repeated attempts further increase inference costs.

The practical objective is therefore *final performance for the total cost of pretraining, post-training, and inference*. The best allocation depends on both post-training gains and expected use.

A Accounting for random cell counts

Section 5 used the typical cell count Dh^d to explain the estimation term. Here we verify that the bound also holds when counts are random and some cells are empty.

Fix a test input x . Its cell has probability h^d , so its training count is $M \sim \text{Binomial}(D, h^d)$. Conditional on the training inputs and $M > 0$, the error in estimating $f(x)$ is an average of $f(x_i) - f(x)$ plus an average of independent noises. The first has magnitude at most $\sqrt{d}h$, and the second has mean zero and variance $1/M$. Their expected squared sum is at most $dh^2 + 1/M$.

If $M = 0$, we predict zero, so $|f| \leq 1$ bounds the squared error by one. The empty-cell probability is at most e^{-Dh^d} . For nonempty cells, $1/m \leq 2/(m+1)$ gives

$$\sum_{m=1}^D \frac{\Pr(M=m)}{m} \leq 2\mathbb{E}\left[\frac{1}{M+1}\right].$$

Using $1/(m+1) = \int_0^1 t^m dt$ and the binomial generating function,

$$\begin{aligned} \mathbb{E}\left[\frac{1}{M+1}\right] &= \int_0^1 (1 - h^d + h^d t)^D dt \\ &= \frac{1 - (1 - h^d)^{D+1}}{(D+1)h^d} \leq \frac{1}{(D+1)h^d}. \end{aligned}$$

Combining the cases and averaging over the test input yields

$$L(N, D) - 1 \leq dh^2 + \frac{2}{(D+1)h^d} + e^{-Dh^d} \lesssim h^2 + \frac{1}{Dh^d} = N^{-2/d} + \frac{N}{D}.$$

The last inequality uses $e^{-u} \leq 1/u$ for $u > 0$ and holds with d fixed. This proves the bound used in the main argument.