SKALING: CHINCHILLA'S EXPONENTS MEET KAPLAN'S COUPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural scaling laws are foundational for language model development, yet the standard formulation systematically under- and overestimate loss at data-scarce and overtraining extremes. This failure originates in the underlying assumption that model size and training data impact the loss independently. We show that this interaction is directly measurable in the loss surface, and capture it with the Skaling law (/skaːliŋ/), a coupled functional form that extends Chinchilla with a single interaction exponent between model capacity and data. On two large training grids, this one extra parameter reduces the Mean Absolute Percentage Error (MAPE) by roughly 2–3× on interpolation and extrapolation, and by up to an order of magnitude when fitted on sparse grids. When paired with a sparse grid strategy restricted to low-compute regimes, the Skaling law matches or exceeds the accuracy of Chinchilla fitted on the full grid while using up to 10× less compute. By enabling reliable performance prediction from small-scale experiments, the Skaling law provides a more robust and resource-efficient framework for allocating compute budgets in next-generation model training.

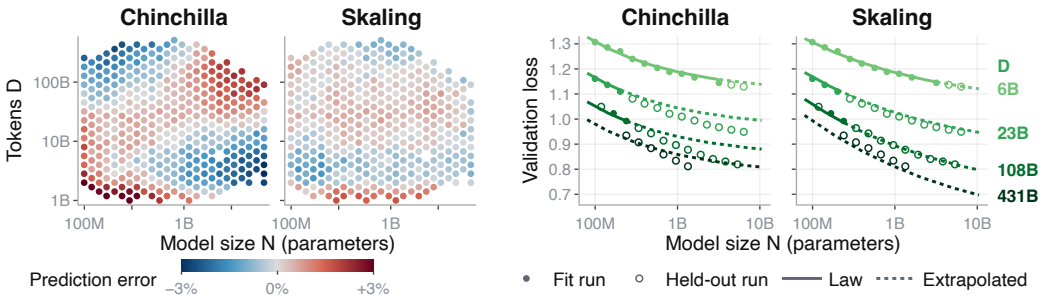


Figure 1: **The additive Chinchilla law leaves a structured prediction bias. Left:** fit on all 404 Farseer runs; each dot is one (N, D) run, coloured by signed error (red: overestimate, blue: underestimate). Chinchilla's residual is saddle-shaped and grows toward the corners; Skaling's is smaller and less structured. **Right:** fit on the cheapest runs (L-shape: smallest 25% of N or D) and evaluated on the remaining (open circles); on four data budgets (dotted where extrapolated). Chinchilla drifts above the held-out runs as D grows, while Skaling stays close (3.2% vs. 1.0%MAPE).

1 INTRODUCTION

The success of modern Large Language Models (LLMs) rests on their predictable and consistent improvement as training data and model size scale. This property is formalized by neural scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; Bi et al., 2024), which predict the performance of high-compute runs using only a handful of low-compute models. Guided by these empirical laws, researchers can confidently optimize critical decisions regarding pretraining budgets, architectural dimensions, and resource allocation.

Despite their widespread adoption, the standard scaling formulation has a structural flaw (Li et al., 2025a; Ardalani et al., 2026). While the original Kaplan form (Kaplan et al., 2020) coupled model

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

size and data, the widely used Chinchilla law (Hoffmann et al., 2022) models the reducible loss as a simple sum of independent terms for model size and training tokens. This mathematical structure implies a strict independence between the two variables, artificially enforcing a cross-derivative of exactly zero. However, early analysis (Figure 1) shows Chinchilla is accurate in the interior of the grid but develops large, oppositely-signed errors toward the corners, reaching several percent where N and D are most imbalanced. This is the saddle-shaped residual expected when the N–D interaction is omitted. Furthermore, our empirical analysis of the loss gradients reveals an interaction between these dimensions (Section 2).

To capture this interaction, we study the Skaling law, a minimal coupled generalization of the Chinchilla form in which a single exponent couples model size and data volume, restoring the interaction of the Kaplan form that the additive Chinchilla law discards. This single extra parameter is enough to correct Chinchilla's boundary bias and substantially lowers its prediction error (Section 4.2). We further show that this improved functional form enables an efficient "L-shape" sampling strategy. By restricting training to the cheaper edges of the compute grid, sweeping data volume for small models (varying training tokens D, referred to as the D-band) and sweeping model size on a small, fixed amount of data (varying model size N, referred to as the N-band), the Skaling law matches the predictive accuracy of Chinchilla fitted on the full grid at a fraction of its compute. Together, these contributions provide a practical, well-grounded framework for scaling decisions under constrained experimental budgets.

Contributions

- **C1.** Evidence for an N–D interaction. We show, from boundary residuals and model-free mixed derivatives of the loss surface, that model size and data do not act independently, and that a coupled form with a single interaction exponent, the Skaling law, corrects the boundary biases of additive laws (Sections 2 and 3.1).
- **C2.** Sparse profiling grids. We show that Skaling's predictive accuracy is preserved under a more effective sampling scheme, enabling "L-shape" grids that reduce profiling compute by up to $10\times$ while maintaining accuracy (Section 3.2).

2 The loss surface couples model size and data

Before committing to a functional form, we ask the data directly whether model size and training data interact. We probe the loss surface through its derivatives, estimated with the Moving least squares (MLS) (Lancaster & Salkauskas, 1981) estimator (described in Section A.1); because the grid is logarithmically spaced, MLS returns log-slopes that we convert to real-space derivatives,

$$\frac{\partial L}{\partial N} = \frac{L}{N}\frac{\partial \ln L}{\partial \ln N}, \qquad \frac{\partial L}{\partial D} = \frac{L}{D}\frac{\partial \ln L}{\partial \ln D},$$

Differentiation removes the irreducible error E and isolates the structure of the reducible loss.

We first summarize the first-order structure with the log-linear diagnostic

$$\ln\left|\frac{\partial L}{\partial N}\right| = \alpha_N \ln N + \gamma_N \ln D + c_N, \qquad \ln\left|\frac{\partial L}{\partial D}\right| = \gamma_D \ln N + \alpha_D \ln D + c_D, \tag{1}$$

where α_N, α_D capture the dominant same-variable decay and γ_N, γ_D the residual dependence on the other axis; we use it only as a diagnostic, not as a scaling law. Figure 2(a,e) shows the same-variable projections are close to linear, with $\alpha_N \approx \alpha_D \approx -1.3$, so the marginal derivatives decay roughly as power laws. The cross-slopes (b,d) are small, $\gamma_N \approx 0.13$ and $\gamma_D \approx 0.07$, so at first order the surface looks nearly separable. But first-order projections alone, however, cannot rule out a weaker interaction.

The mixed derivative is the decisive test. Any additive law $L = f(N) + g(D) + E$ satisfies $\partial^2 L/\partial N \partial D = 0$ identically, whatever power laws are chosen for f and g. Figure 2(c,f) shows the estimated mixed derivative is instead non-zero across the entire grid, with its own power-law decay,

$$\ln\left|\frac{\partial^2 L}{\partial N\, \partial D}\right| = a \ln N + b \ln D + c, \tag{2}$$

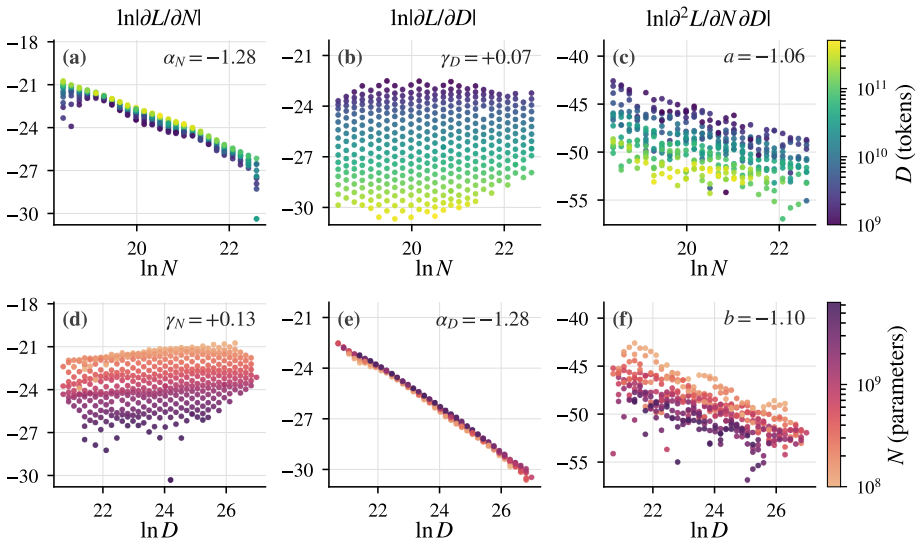


Figure 2: Derivative structure of the Farseer loss surface (MLS estimates). Columns: $\partial L/\partial N$, $\partial L/\partial D$, mixed derivative; rows: against $\ln N$ (coloured by D) and $\ln D$ (coloured by N), with coefficients from Equations (1) and (2). Same-variable projections (a,e) are near-linear, cross-variable slopes (b,d) small. The mixed derivative (c,f), zero under any additive law, is non-zero throughout.

with $a \approx b \approx -1.1$ and a mostly *negative* sign: scaling N and D together lowers the loss more than scaling either alone, a synergy an additive law cannot represent. This motivates replacing Chinchilla's additive sum with a single coupling exponent, the Skaling form introduced next; Section A.2 shows that this form reproduces both the negative mixed derivative and the small, asymmetric first-order slopes, and why a multiplicative coupling is preferable to an additive interaction term.

3 Methods

3.1 The Skaling form

Two functional forms anchor the scaling-law literature, differing primarily in how model size and data combine. The widely used Chinchilla law (Hoffmann et al., 2022) decouples them completely: it sums two terms with independent inner exponents α and β, alongside a free irreducible constant E. While mathematically convenient, this purely additive structure forces the cross-derivative $\partial^2 L/\partial N \partial D$ to vanish identically, implicitly assuming that N and D do not interact. The earlier Kaplan form (Kaplan et al., 2020) takes the opposite approach:

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

Here the outer exponent α_D plays the role of Skaling's k, but with the inner exponent of D fixed to 1, it also sets the decay rate in D, so coupling strength and data exponent cannot be fitted separately.

Skaling bridges these two paradigms. It retains Chinchilla's interpretable base terms and independent inner exponents but, following Kaplan, raises their sum to a single free outer exponent k:

$$L(N, D) = \left(\frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}\right)^{k} + E \tag{3}$$

This single parameter interpolates between the two prior forms. At $k = 1$, the Skaling law recovers the purely additive Chinchilla law; with $\beta = 1$ and $E = 0$, it is exactly Kaplan's form. For any $k \neq 1$, it reinstates a Kaplan-style coupling and a non-zero cross-derivative. Unlike Kaplan, the Skaling law achieves this coupling through the outer exponent k while preserving Chinchilla's independent inner exponents. Despite this added flexibility, the Skaling law retains two properties of the additive

law. First, because $k > 0$, the function remains *strictly decreasing* in both N and D: adding model capacity or training data never increases the predicted loss. Second, it preserves Chinchilla's structure. The exponent k dictates how the two source terms *aggregate* into the final loss, so each retains its interpretability even though their joint contribution no longer separates additively.

A practical payoff of this structure concerns compute allocation. Minimizing the loss under a fixed budget C reduces to the same stationarity condition as the additive Chinchilla law, so the Skaling law inherits Chinchilla's closed-form compute-optimal allocation: the optimal token-to-parameter ratio D^*/N^* keeps its closed form and stays constant across scales when $\alpha \approx \beta$. While the formula is identical, the resulting optimal ratio differs in practice because the fitted parameters A, B, α, and β do not transfer between the two forms. We give the full derivation in Section B.

Accurately identifying this optimal ratio is critical because it dictates baseline architectural decisions. Training recipes are typically scaled at a fixed number of tokens per parameter (Section 3.4), so the true optimum must be known to quantify the penalty of suboptimal training. On Farseer, the ratios predicted by Chinchilla and Skaling diverge at large compute, and Skaling's tracks the model-free estimate more closely (Section 4.2, Section A.3).

3.2 Sampling Strategies

To obtain robust parameter estimates without prohibitive computational costs, the design of the experimental training grid plays an important role. Standard full-grid setups typically sample N and D on a square spaced logarithmically to capture behavior across multiple orders of magnitude. Because of this, the total compute is overwhelmingly dominated by the top-right corner of the grid, which contains the largest models trained for the longest horizons. This geometric concentration of compute makes dense sampling particularly wasteful. Rather than expending the vast majority of an experimental budget on a few massive runs, compute could be better allocated to a more discriminative set of points. The mathematical structure of the scaling laws naturally motivates our primary approach: the L-shape sampling strategy (Figure 3a).

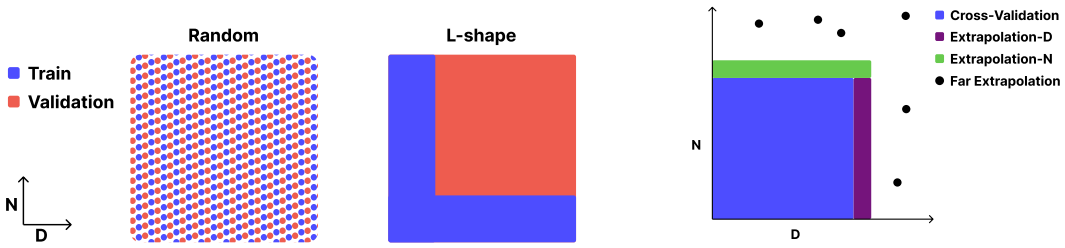
Consider the asymptotic behavior of the loss function. As data volume increases, the estimation error vanishes and the loss approaches the size-dependent approximation error:

$$\lim_{D\to\infty} L(N, D) = \left(\frac{A}{N^{\alpha}}\right)^{k} + E$$

Consequently, increasing D cleanly isolates the N-dependent coefficients. Symmetrically, as model size increases, the approximation error diminishes, cleanly isolating the data-dependent parameters:

$$\lim_{N\to\infty} L(N, D) = \left(\frac{B}{D^{\beta}}\right)^{k} + E$$

In practice, we do not need to reach these theoretical limits. Simply varying one axis while holding the other fixed provides sufficient signal to trace the corresponding decay rate. The L-shape strategy



(a) Sampling strategies. (b) Evaluation regimes.

Figure 3: Partitions of the (N, D) grid. **(a)** Sampling strategies: **Random** spreads held-out points across the grid, whereas the **L-shape** grid restricts training to the low-compute edges; for the L-shape, the interpolation set also contains every grid point outside the L. **(b)** Evaluation regimes used for cross-validation: interpolation, extrapolation in N and in D, and far extrapolation beyond both.

applies this principle at the lowest compute scales. Rather than filling the whole grid, we sweep data volume D only for the smallest models to fit the data parameters (B, β), and sweep model size N only on the shortest training horizons to fit the size parameters (A, α). By anchoring the independent decay rates along the grid boundaries, this sparse geometry lets the Skaling law match or exceed the accuracy of a full-grid Chinchilla fit at a fraction of the compute (Section 4.2).

3.3 Evaluation Protocol

To test the predictive capabilities and algorithmic stability of the fitted methods, we employ a comprehensive cross-validation framework. Rather than relying on a single static train-test split, we repeatedly resample the training data and construct corresponding hold-out sets. This approach allows us to explicitly quantify the uncertainty of our predictions, assess the variance of the fitted parameters, and test the models' ability to extrapolate reliably to unseen scales. For each cross-validation fold, the evaluation sets are partitioned as follows (Figure 3b):

- **Validation (Interpolation):** Randomly held-out points inside the training grid.
- **Extrapolation N:** Points at model sizes beyond the training set, simulating large-scale prediction.
- **Extrapolation D:** Points at larger data volumes, testing predictions for extended training horizons.
- **Far Extrapolation:** The most challenging regime, consisting of the largest models trained on the largest data volumes, lying completely outside both the N and D boundaries of the training grid.

On all cross-validation folds, we report the MAPE on every evaluation set and the coefficient of determination (R^2) on the interpolation set only. For an evaluation set $\mathcal{S}$ in which run i has measured loss L_i and the fitted law predicts $\hat{L}_i$, the MAPE is the mean absolute relative deviation,

$$\mathrm{MAPE}(\mathcal{S}) \;=\; \frac{100}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \frac{\left|\hat{L}_i - L_i\right|}{L_i} \quad [\%]. \tag{4}$$

We restrict R^2 to interpolation because the extrapolation sets contain few points covering a narrow, somewhat arbitrary slice of the grid; since R^2 normalizes error by the variance of the held-out targets, it becomes unstable and uninformative there (often small or strongly negative), whereas MAPE stays directly comparable across all regimes. Aggregating over folds, we report the mean of each metric together with its standard deviation. The interpolation set is resampled in each fold, whereas the extrapolation sets are fixed; their spread therefore measures how stable each law's predictions are under changes of the training subset, not variation in the test points.

3.4 Compute Extrapolation Along Iso-Ratio Slices

Frontier labs commonly use one-dimensional power laws in compute to predict how the loss evolves as pretraining is scaled up (Bi et al., 2024; The Microsoft AI Team, 2026). Because these models are typically scaled at a constant number of tokens per parameter (The Microsoft AI Team, 2026), extrapolating compute along these specific operational rays represents an interesting point of comparison for any scaling formulation. While our earlier cross-validation comprehensively maps the N–D grid, here we address a strictly operational challenge: can a single, globally fitted law reliably predict the most computationally expensive runs using only cheap, low-compute data?

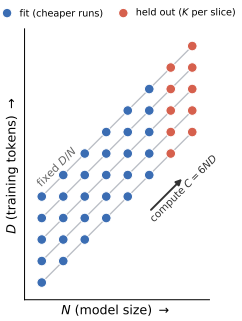


Figure 4: Iso-ratio compute extrapolation.

To mirror this scenario, we group the runs into iso-ratio slices of constant token-to-parameter ratio D/N (Figure 4). Within each slice, we hold out the K highest-compute points (K=8). Every scaling law is refitted *once* on the pooled remainder, the union of the low-compute points across all slices—and then evaluated on the held-out high-compute runs (results in Section 4.3). This ensures each functional form sees the exact same training set, so the global comparison is like-for-like. As a strong baseline, we additionally fit an independent single power law $L = A\,C^a + E$ strictly within each slice (*per-ratio*), following the constant tokens-per-parameter scaling ladders of The Microsoft AI Team (2026), a fixed-ratio variant of the compute-based loss prediction of DeepSeek (Bi et al., 2024). While this per-ratio baseline is heavily tailored to a single recipe and cannot inform joint N–D allocation, it serves as a strong reference for how well a dedicated, one-dimensional compute law can extrapolate along a fixed ray.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Data. We fit and evaluate all scaling laws on two grids of pretraining runs.

Farseer (Li et al., 2025a) records the final validation loss of 404 (N, D) configurations across 25 model sizes (100M to 6.4B parameters) and 55 data budgets (1B to 512B tokens), with compute ranging from 1.6×10^{18} to 4.1×10^{21} FLOPs. All models use a sequence length of 2048. From this grid we hold out three evaluation sets: *Extrapolation* N (the 3 largest model sizes, 4.5B–6.4B, 36 points), *Extrapolation* D (the top-3 data budgets per remaining model size, 66 points), and *Far extrapolation* (7 additional runs at larger scales, 2.3B–25B parameters trained on 126B–453B tokens, well beyond both axes of the training grid). The 302 fitting configurations total $\sim 5.0 \times 10^{22}$ FLOPs.

SK-Grid, our own grid of training runs, is a complementary set of 125 configurations across 14 model sizes (134M to 4.9B) and 16 data budgets (316M to 316B tokens), with compute from 9.0×10^{16} to 9.9×10^{20} FLOPs. The same hold-out scheme yields 7 points for Extrapolation N (2.8B–4.9B) and 33 points for Extrapolation D. Far extrapolation uses 3 additional runs at $\sim 10^{22}$ FLOPs (5.8B–10.8B parameters), trained separately from the grid. The fitting grid totals $\sim 3.1 \times 10^{21}$ FLOPs.

Optimizer. All scaling laws are fitted by minimizing a Huber loss in log space using L-BFGS-B with basin-hopping and autograd. Full optimizer settings and per-law bounds are given in Section C.

Baselines. We benchmark the Skaling law against the standard additive Chinchilla law (Hoffmann et al., 2022) and the more heavily parameterized Farseer law (Li et al., 2025a), which also models an N–D interaction. We also tried the Kaplan form (Kaplan et al., 2020) with an added irreducible term, which is Skaling with β fixed to 1. Its fits were far worse than those of every other law on our grids, consistent with the fitted β of Skaling lying far from 1 (0.33–0.63, Table 2), so we do not report it. All laws are fit with the same optimizer and log-space objective (Section C), so that differences in predictive accuracy reflect the functional form rather than the fitting procedure. Predictive error is measured under the cross-validation protocol of Section 3.3.

Training. The models in the grid are trained with hyperparameters set by power laws in the style of Step Law (Li et al., 2025b), which predict the optimal learning rate and batch size from model size and token budget. The optimal learning rate shifts with the training horizon (Bjorck et al., 2025; Shulgin et al., 2026) as well as with model size (Bjorck et al., 2025), so a single setting reused across the grid would be mistuned for some configurations and distort the measured surface. Setting the hyperparameters of every (N, D) near their optimum, as also done for the Farseer grid (Li et al., 2025a), lets the fits target the scaling behaviour of each model under its best setup rather than hyperparameter mistuning. Architectures and hyperparameters are detailed in Section E.

4.2 RESULTS

We first locate where the additive law fails and what the fitted coupling reveals, then test the coupled law under sparse sampling, against a richer law, and on nearly additive data, and end with the consequences for compute allocation. Table 1 reports the predictive errors and Table 2 the fitted coefficients behind them.

Chinchilla fits the interior but mispredicts beyond it. Judged by interpolation alone, the additive law looks excellent: on the full grids it reaches $R^2 = 0.995$ on Farseer and 0.992 on SK-Grid, with MAPE below 1%. Beyond the training region, however, its errors grow, up to 5.17 on the far-extrapolation runs of SK-Grid. This mirrors the saddle-shaped residual of Figure 1: an additive surface can match the interior while bending the wrong way toward the edges, so a high interpolation R^2 says little about how well a law extrapolates. The Skaling law lowers the error in every regime of both grids. On single-axis extrapolation, the reduction is 2–3$\times$ (Farseer: $1.48 \to 0.47$ in N and $1.98 \to 0.88$ in D; SK-Grid: $0.83 \to 0.39$ and $1.44 \to 0.58$), and interpolation improves by a similar factor. On far extrapolation, the error falls $7\times$ on SK-Grid ($5.17 \to 0.70$), whereas on Farseer the two laws are close (2.46 vs. 2.31), the one regime where the coupled form brings little.

Table 1: Fit quality and predictive error on Farseer and SK-Grid, for the full grid and the sparse L-shape (total training FLOPs in the panel headers). Interpolation R^2 (↑) and MAPE (%, ↓; mean ± std over 5 CV folds) on interpolation, larger N, larger D, and far extrapolation beyond both (Section 3.3). Best per column within each panel in **bold**.

	Farseer data					SK-Grid data				
Law	R^2	Interp.	Ext. N	Ext. D	Far	R^2	Interp.	Ext. N	Ext. D	Far
Full grid			5.0×10^{22} FLOPs					3.1×10^{21} FLOPs		
Chinchilla	0.995	0.77±0.04	1.48±0.03	1.98±0.08	2.46±0.19	0.992	0.81±0.14	0.83±0.11	1.44±0.03	5.17±0.28
Farseer	0.982	1.73±0.45	2.37±0.08	4.13±1.36	2.43±1.93	0.967	1.66±0.32	0.90±0.49	4.45±0.21	3.98±1.25
Skaling	**0.998**	**0.41±0.05**	**0.47±0.03**	**0.88±0.06**	**2.31±0.18**	**0.998**	**0.33±0.11**	**0.39±0.05**	**0.58±0.07**	**0.70±0.39**
L-shape			5.1×10^{21} FLOPs					6.5×10^{20} FLOPs		
Chinchilla	0.954	2.51±0.07	4.32±0.13	3.29±0.11	9.82±0.48	0.955	2.19±0.10	6.09±0.24	3.63±0.13	14.63±0.39
Farseer	0.974	1.81±1.23	2.07±1.72	2.52±1.95	2.37±1.33	0.987	0.82±0.52	1.19±0.82	2.66±2.42	4.64±1.31
Skaling	**0.995**	**0.85±0.10**	**0.89±0.23**	**1.35±0.20**	**1.51±0.67**	**0.998**	**0.33±0.03**	**0.77±0.44**	**0.55±0.08**	**1.15±0.53**

Table 2: Fitted coefficients (mean ± std over 5 CV folds) behind Table 1. Chinchilla is the k=1 special case of Skaling; the nine-parameter Farseer law is omitted (see Li et al., 2025a).

Setup	Law	A	B	α	β	k	E
Farseer, full	Chinchilla	$(4.8\pm1.2)\times10^{1}$	$(1.1\pm0.1)\times10^{2}$	0.27±0.01	0.24±0.00	1	0.45±0.01
	Skaling	$(2.9\pm0.2)\times10^{2}$	$(6.0\pm0.3)\times10^{3}$	0.32±0.01	0.39±0.00	0.41±0.01	0.03±0.02
Farseer, L-shape	Chinchilla	$(2.6\pm0.6)\times10^{2}$	$(1.0\pm0.1)\times10^{2}$	0.39±0.02	0.24±0.00	1	0.59±0.01
	Skaling	$(2.5\pm0.7)\times10^{2}$	$(1.7\pm0.4)\times10^{3}$	0.32±0.01	0.33±0.01	0.45±0.03	0.05±0.06
SK-Grid, full	Chinchilla	$(5.0\pm1.2)\times10^{2}$	$(7.1\pm0.8)\times10^{2}$	0.34±0.01	0.31±0.01	1	1.75±0.02
	Skaling	$(5.3\pm1.8)\times10^{6}$	$(7.1\pm2.8)\times10^{6}$	0.73±0.01	0.63±0.01	0.31±0.02	1.14±0.06
SK-Grid, L-shape	Chinchilla	$(1.0\pm0.0)\times10^{4}$	$(1.7\pm0.4)\times10^{3}$	0.52±0.00	0.36±0.01	1	2.16±0.02
	Skaling	$(1.0\pm0.0)\times10^{7}$	$(6.5\pm2.4)\times10^{6}$	0.77±0.02	0.63±0.01	0.31±0.02	1.18±0.12

The fitted coupling is far from additive, and stable. Because Chinchilla is the $k = 1$ special case of Skaling, the fitted k measures how far the data are from additive: the optimizer is free to return $k \approx 1$, and it does not. In all four fits of Table 2, it settles on $k \approx 0.31$–0.45, with a fold-to-fold spread of at most 0.03, so the coupling is a stable property of each grid, not of one training subset. A value $0 < k < 1$ also gives the Skaling surface the negative mixed derivative measured model-free in Section 2 (Section A.2): more data make parameters more useful, and vice versa.

E is poorly constrained at these scales. The two laws also disagree on the irreducible loss: Skaling's E is lower on every grid, and on Farseer it nearly vanishes, close to the lower bound $E = 0$ of the fit ($0.45 \to 0.03$ on the full grid, $0.59 \to 0.05$ on the L-shape; SK-Grid: $1.75 \to 1.14$). We do not read this as a vanishing loss floor: it mainly reflects how hard E is to estimate from small-scale runs. E is the limit of the loss as $N, D \to \infty$, and our grids do not reach the scales at which the loss approaches it, so the data fix the total loss but not its split between a decaying term and a constant floor (Sections C.1 and F). Skaling leans toward a low floor and a more slowly decaying reducible term. In its form, the reducible loss decays asymptotically as $N^{-k\alpha}$ or $D^{-k\beta}$ (the limits of Section 3.2); these effective exponents, rather than the inner α and β, are the ones to compare with Chinchilla's, and they are smaller on every grid (on Farseer, $k\alpha \approx 0.13$ and $k\beta \approx 0.16$ vs. 0.27 and 0.24). Neither law's E should therefore be taken at face value: estimating it reliably would require runs at substantially larger scale than our grids provide.

Sparse L-shape: the cheap edges suffice for the coupled law. The L-shape keeps only the low-compute edges of each grid, cutting fitting compute by about $10\times$ on Farseer and $5\times$ on SK-Grid (lower panel of Table 1), and its interpolation set covers the whole unsampled part of the grid. Chinchilla degrades sharply: its far-extrapolation error quadruples on Farseer ($2.46 \to 9.82$) and reaches 14.63 on SK-Grid, against 1.15 for Skaling. The practically relevant comparison is with Chinchilla fitted on the *full* grid. Skaling fitted on the L-shape is more accurate in every regime on SK-Grid, at a fifth of the compute, and in all three extrapolation regimes on Farseer (1.51 vs. 2.46 far), at a tenth, with comparable interpolation error (0.85 vs. 0.77). The fitted parameters explain why (Table 2). From the full grid to the L-shape, Skaling's exponents move by at most 0.06 and k

by at most 0.04, whereas Chinchilla's size exponent shifts by 0.12–0.18 and its floor E rises. On the imbalanced edges, where one resource is scarce, Chinchilla distorts its exponents to absorb an interaction it cannot represent, so they transfer poorly to the rest of the grid. The edges thus suffice to reconstruct the surface (Section 3.2), but only for a form that represents the interaction.

More parameters do not mean better extrapolation. The Farseer law also models an N–D interaction, with nine parameters against Skaling's six, and modeling the interaction does help on the edges: on the L-shape, it beats Chinchilla in every regime. Its extra flexibility, however, does not translate into better predictions. It is less accurate than Skaling in every regime of Table 1, trails even Chinchilla in most full-grid regimes (e.g. 4.13 vs. 1.98 in D on Farseer), and varies far more across folds (up to ± 2.4 points). Parameters that the training data constrain only weakly change from one training subset to the next, and extrapolation amplifies these changes. Part of the gap may also reflect a harder optimization problem, although Farseer's original stage-wise pipeline did not give clearly better predictions where it could be applied (Section C). Good extrapolation thus depends not on the number of parameters but on whether the form reflects the true structure of the loss surface, which Skaling captures with a single extra parameter.

When the coupling is weak, the gains shrink. The nested form also predicts when the coupling should matter little. On Farseer-code and on the original Chinchilla measurements (Section D), the fitted coupling is weaker ($k \approx 0.77$–0.90) and the gains are smaller and less uniform: Skaling still improves on Chinchilla in every regime on Farseer-code, whereas on the Chinchilla measurements the two laws are within 0.12 MAPE points of each other. Since the additive law is the case $k = 1$, adopting Skaling costs little when the coupling is weak, and the fitted k indicates whether it matters for a given setup. Why the coupling strength varies across grids remains open; differences in data and hyperparameter policy are natural candidates (Section E).

Coupling changes the compute-optimal allocation. Both laws share the same closed-form optimal ratio (Section B), whose trend with compute, $D^\star/N^\star \propto C^{(\alpha-\beta)/(\alpha+\beta)}$, is set by the fitted exponents alone. On Farseer (laws fitted once on all runs, as in Figure 5), Chinchilla fits $\alpha \approx \beta$ and predicts a nearly constant ratio (exponent $+0.03$), whereas Skaling fits $\alpha < \beta$ and predicts a ratio that decreases with compute (-0.11). The frontier estimated directly from numerical loss gradients, without any parametric law, agrees with Skaling: it decreases with compute, with fitted exponents -0.14 and -0.15 (Section A.3). Extrapolated to 2×10^{25} FLOPs, nearly four orders of magnitude beyond the data, the two laws recommend ratios that differ by roughly $8\times$ (Figure 5). The direction of the trend is, however, dataset-specific: on SK-Grid, Skaling fits $\alpha > \beta$ on both the full and L-shape grids, so its optimal ratio increases with compute. The robust conclusion is that modeling the coupling changes large-scale allocation, while the direction of the change depends on the data and training recipe.

4.3 Compute extrapolation

The cross-validation above held out the *edges* of the grid; we now turn to the operational question of Section 3.4: predicting the most expensive runs of each training recipe from its cheaper ones. Table 3 reports the error on the 112 held-out high-compute Farseer runs, with every global law fitted once on the pooled low-compute remainder. We group the iso-ratio slices into three regimes named relative to the familiar rule of roughly 20 tokens per parameter (Hoffmann et al., 2022): undertrained, Chinchilla-optimal and overtrained. These names are only a reference point, since the compute-optimal ratio itself varies with compute (Section A.3). Skaling is the most accurate global law in every regime and overall (MAPE 0.60%, $3.9\times$ lower than Chinchilla), and the most uniform, with a mean error below 0.9% in each regime. Chinchilla's error, by contrast, depends strongly on the regime and peaks in the Chinchilla-optimal band (3.47%, $R^2 = 0.47$), which contains this widely used rule; its pooled error understates this failure.

Skaling also beats the per-ratio power law, a dedicated one-dimensional fit for each slice, overall (0.60 vs. 0.99%) and in the undertrained and overtrained regimes. In the Chinchilla-optimal band, whose slices hold the most cheap runs (14–17, against 3–14 elsewhere), the per-ratio law's lower mean (0.77 vs. 0.88%) lies within its much larger slice-to-slice spread (± 0.39 vs. ± 0.15), so the two are comparable. The advantage of the global fit comes from pooling: it is fitted on the cheap runs of all slices at once, whereas a per-ratio law sees only those of its own slice. Pooling helps only if the form describes the whole surface, however: the pooled Chinchilla fit is less accurate than the per-ratio

Table 3: Compute extrapolation on Farseer: undertrained (D/N=1.8–7), Chinchilla-optimal (10–40) and overtrained (56–158) slices. R^2 and MAPE (%) are averaged over the slices of each group (MAPE: mean ± std). *Per-ratio*: one power law per slice. Best per group in **bold**.

	Undertrained		Chinchilla-optimal		Overtrained		All setups	
Law	R^2	MAPE	R^2	MAPE	R^2	MAPE	R^2	MAPE
Power law (per-ratio)	0.94	1.32±0.87	0.96	**0.77±0.39**	0.97	0.86±0.75	0.95	0.99±0.69
Chinchilla	0.89	1.52±0.80	0.47	3.47±0.65	0.87	1.94±0.66	0.74	2.34±1.11
Farseer	**0.99**	0.46±0.20	0.94	1.20±0.20	**0.99**	0.73±0.23	0.97	0.80±0.38
Skaling	**0.99**	**0.45±0.21**	**0.97**	0.88±0.15	**0.99**	**0.42±0.15**	**0.98**	**0.60±0.27**

laws in every regime. Here too, more parameters do not mean better extrapolation: the Farseer law (9 parameters) trails Skaling in every regime, if only marginally on undertrained slices (0.46 vs. 0.45%). Finally, only a global law determines how to trade parameters for data at a given budget.

5 Related Work

Functional forms. Loss falls predictably as a power law in scale (Hestness et al., 2017; Rosenfeld et al., 2020). Kaplan et al. (2020) modeled it jointly in N and D with a coupled form; Hoffmann et al. (2022) replaced it with an additive sum of independent power laws, which became the default behind the compute-optimal rule of roughly 20 tokens per parameter (Besiroglu et al., 2024) and the prediction of large runs from small ones (Bi et al., 2024). Richer laws recover an N–D interaction at a higher cost: Farseer (Li et al., 2025a) makes the data exponent and amplitude depend on N, with nine fitted parameters, while broken neural scaling laws (Caballero et al., 2023) chain power-law segments along a single variable. Other extensions cover repeated data (Muennighoff et al., 2023) and data mixtures (Ye et al., 2025; Shukor et al., 2025).

Fitting and reconciliation. Disagreements between Kaplan- and Chinchilla-style allocations were traced to parameter counting (Pearce & Song, 2024) and to FLOP accounting, warmup, and optimizer tuning (Porian et al., 2024), and fitting choices alone can shift the compute-optimal ratio (Li et al., 2025c); all of these leave the additive form intact. Guidelines for fitting from few runs (Choshen et al., 2025) and methods that select which runs to train (Li et al., 2026) reduce profiling cost; our L-shaped grid is a simple fixed design in this spirit.

Closest work. Busbridge et al. (2025) use the same untied outer exponent, $E + (A/N^\alpha + B/D^\beta)^\gamma$, for the supervised losses inside their distillation scaling law, noting that it fits and extrapolates better than the tied forms. They do not report a controlled comparison of these forms or study the resulting N–D interaction, the central question of our work.

6 Conclusion

We showed that model size and data do not act on the loss independently: the mixed derivative of the loss surface is non-zero across the grid, and the additive Chinchilla law, which forces it to zero, develops oppositely-signed errors at the imbalanced corners of the grid. The Skaling law captures this interaction by combining Chinchilla's separate exponents with Kaplan's coupling through a single outer exponent. This one additional parameter substantially reduces the boundary bias, predicts more accurately than Chinchilla and retains a closed-form compute-optimal allocation. Because the coupling is determined by the edges of the grid, an L-shaped design of inexpensive boundary runs matches the accuracy of a full-grid Chinchilla fit at a fraction of its compute. Extending this analysis to other scaling axes, such as data repetition and data mixtures, is a natural next step.

AI use statement

Generative AI tools assisted implementation, debugging, data reformatting, figures, code, and manuscript editing. The authors reviewed all assisted outputs, checked reported results against the underlying runs and data, and take responsibility for the final content.

REFERENCES

Newsha Ardalani, Feiyang Kang, Michael Kuchnik, Mostafa Elhoushi, Shubho Sengupta, Shang-Wen Li, and Carole-Jean Wu. How text quality interventions reshape neural scaling laws for LLMs: Empirical study. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZC5QBfdOw7.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Agustín Piqueres Lajarín, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Ben Burtenshaw, Clémentine Fourrier, Haojun Zhao, Hugo Larcher, Mathieu Morlon, Cyril Zakka, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=3JiCl2A14H.

Tamay Besiroglu, Ege Erdil, Matthew Barnett, and Josh You. Chinchilla scaling: A replication attempt. *arXiv preprint arXiv:2404.10102*, 2024.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024. URL https://arxiv.org/abs/2401.02954.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal LR across token horizons. In *International Conference on Learning Representations*, pp. 83640–83657, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/cffa22c56c0df3b3edb1df8a9ad67804-Abstract-Conference.html.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russell Webb. Distillation scaling laws. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 5977–6045. PMLR, 2025. URL https://proceedings.mlr.press/v267/busbridge25a.html.

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sckjveqlCZ.

Leshem Choshen, Yang Zhang, and Jacob Andreas. A hitchhiker's guide to scaling law estimation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10683–10699. PMLR, 2025. URL https://proceedings.mlr.press/v267/choshen25a.html.

Nikolaus Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pp. 2389–2396. ACM, 2009. doi: 10.1145/1570256.1570333.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016. URL https://arxiv.org/abs/1604.00772.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017. URL https://arxiv.org/abs/1712.00409.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981. doi: 10.1090/S0025-5718-1981-0616367-1.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Zhenyu Ding, Haoying Wang, Zili Wang, Shijie Xuyang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable Scale (Part II)—Farseer: A refined scaling law in LLMs. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025a. doi: 10.52202/085713-4024. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ae9f1d3b89a81fcb1c45b4fc1afbd4fe-Paper-Conference.pdf.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable Scale: Part I, Step Law–optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*, 2025b.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 14200–14282. Curran Associates, Inc., 2024. doi: 10.52202/079017-0455. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/19e4ea30dded58259665db375885e412-Paper-Datasets_and_Benchmarks_Track.pdf.

Margaret Li, Sneha Kudugunta, and Luke Zettlemoyer. (mis)Fitting scaling laws: A survey of scaling law fitting techniques in deep learning. In *International Conference on Learning Representations*, pp. 51134–51165, 2025c. URL https://openreview.net/forum?id=xI71dsS3o4.

Sijie Li, Shanda Li, Haowei Lin, Weiwei Sun, Ameet Talwalkar, and Yiming Yang. Spend less, fit better: Budget-efficient scaling law fitting via active experiment selection. In *Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=M2k8z7RUmL.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 50358–50376. Curran Associates, Inc., 2023. doi: 10.52202/075280-2191. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9d89448b63ce1e2e8dc7af72c984c196-Abstract-Conference.html.

Tim Pearce and Jinyeop Song. Reconciling Kaplan and Chinchilla scaling laws. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=NLoaLyuUUF.

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100535–100570. Curran Associates, Inc., 2024. doi: 10.52202/079017-3189. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b6341525cd84f3be0ef203e4d7cd8556-Abstract-Conference.html.

Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryenvpEKDr.

Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-4314. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/bc1d640f841f752c689aae20b31198c1-Paper-Conference.pdf.

Egor Shulgin, Dimitri von Rütte, Tianyue H. Zhang, Niccolò Ajroldi, Bernhard Schölkopf, and Antonio Orvieto. Deriving hyperparameter scaling laws via modern optimization theory. *arXiv preprint arXiv:2603.15958*, 2026.

The Microsoft AI Team. MAI-Thinking-1: Building a hill-climbing machine. Technical report, Microsoft AI, 2026. URL https://microsoft.ai/pdf/mai-thinking-1.pdf.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. Meta Lingua: A minimal PyTorch LLM training library, 2024. URL https://github.com/facebookresearch/lingua.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *International Conference on Learning Representations*, pp. 82263–82287, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/cc84bfabe6389d8883fc2071c848f62a-Paper-Conference.pdf.

APPENDIX

A DIFFERENTIAL ANALYSES

A.1 NUMERICAL GRADIENTS

We estimate derivatives of the loss surface directly from the measured runs. At each grid point, we estimate $\partial z/\partial x_j$ using two mesh-free procedures: a local moving least-squares estimator (MLS) and a global Gaussian-process estimator (GP).

Moving least squares (MLS). The local estimator fits a truncated Taylor expansion of the surface in a neighbourhood of each query point $x_\star$ (Lancaster & Salkauskas, 1981). To estimate the derivatives at a specific target point $x_\star$, we locally approximate the loss surface using a multivariate polynomial of degree p. This approach comes directly from Taylor expansion: if we fit a local polynomial, its coefficients correspond mathematically to the function's derivatives of various orders.

Let x represent the spatial coordinates, and let $\Delta x_i = x_i - x_\star$ denote the centered distance to a neighboring point. By Taylor's theorem, the local loss value $z(x_i)$ can be expanded as:

$$z(x_i) \approx c_0 + g^\top \Delta x_i + \frac{1}{2}\Delta x_i^\top H \Delta x_i + \dots$$

where the polynomial expansion continues with higher-order basis terms up to degree p. In this formulation, c_0 is the local constant, the vector g is the gradient $\nabla z(x_\star)$, and H is the Hessian matrix containing our target cross-derivative.

To solve for these unknowns simultaneously, we flatten all the coefficients of this expansion into a single vector c. We then determine the optimal coefficient estimate $\hat{c}$ by fitting this polynomial to the k nearest neighbors using distance-weighted ridge regression:

$$\hat{c} = \left(\Phi^\top W \Phi + \lambda I\right)^{-1} \Phi^\top W z$$

The components of this generalized regression map directly to the Taylor expansion:

- Φ is the design matrix. Each row corresponds to a neighbor and contains its evaluated polynomial basis terms up to degree p (for example: 1, $\Delta x_{i,1}$, $\Delta x_{i,2}$, $\Delta x_{i,1}^2$, the interaction term $\Delta x_{i,1}\Delta x_{i,2}$, and higher-order combinations).
- W is a diagonal weight matrix where $w_i = \exp\left(-\|\Delta x_i\|^2/\sigma^2\right)$. This acts as a Gaussian filter, ensuring that points closer to $x_\star$ exert more influence on the local fit. The bandwidth σ scales dynamically with the neighborhood radius.
- λI adds a small ridge regularization penalty to keep the matrix inversion numerically stable, which is crucial if the scattered local data points are poorly distributed.

Once we solve for $\hat{c}$, extracting the necessary derivatives is straightforward. We read them directly from their designated slots in the vector: the first-order block provides the gradient estimate, and the coefficient corresponding to the $\Delta x_{i,1}\Delta x_{i,2}$ basis term provides the exact mixed cross-derivative required for our analysis. Because this Moving Least Squares (MLS) approach relies entirely on local geometry, its accuracy is naturally sensitive to local noise, the chosen neighborhood size k, and the polynomial degree p.

Gaussian process (GP). The global estimator fits a single Gaussian process to all points and differentiates its posterior mean in closed form. We model the empirical loss targets $z \in \mathbb{R}^{n\times 1}$ as a smooth function of the input coordinates, corrupted by some observation noise σ_n^2. In standard GP regression, the predicted log-loss $\hat{z}(x_\star)$ at a target point $x_\star$ is given directly by the posterior mean:

$$\hat{z}(x_\star) = \mu(x_\star) = k(x_\star, X)\alpha$$

Here, $k(x_\star, X) \in \mathbb{R}^{1\times n}$ is a row vector of kernel evaluations between the target point and all n training data points. The learned weight vector is defined as $\alpha = (K + \sigma_n^2 I)^{-1} z \in \mathbb{R}^{n\times 1}$, where $K \in \mathbb{R}^{n\times n}$ is the dense covariance matrix of the training inputs and I is the $n \times n$ identity matrix.

The key advantage of using a GP is that differentiation is a linear operator, meaning the derivative of a GP is simply another GP. We can compute the estimated scalar gradient of the loss surface in closed form by directly differentiating the predicted surface $\hat{z}(x_\star)$ with respect to the input dimensions. Because the weights α are independent of $x_\star$, the derivative simply applies to the $1 \times n$ row vector:

$$\widehat{\frac{\partial z}{\partial x_j}}(x_\star) = \frac{\partial \hat{z}(x_\star)}{\partial x_j} = \frac{\partial k(x_\star, X)}{\partial x_j}\alpha$$

Because standard kernel functions are analytically differentiable, this computation is exact. In this work, we are using a Radial Basis Function (RBF) kernel with length-scale ℓ_j, the derivative is $\partial k(x, x')/\partial x_j = k(x, x')(x'_j - x_j)/\ell_j^2$. Composite kernels can be differentiated using standard sum and product rules, provided that any additive white-noise terms are excluded from the predictive kernel.

Additionally, the GP provides a closed-form predictive variance for this gradient, quantifying our estimation uncertainty. It is calculated by taking the scalar prior variance of the gradient and subtracting the information gained from the observed data:

$$\mathrm{Var}\left[\frac{\partial z}{\partial x_j}(x_\star)\right] = \left.\frac{\partial^2 k(x, x')}{\partial x_j\, \partial x'_j}\right|_{x=x'=x_\star} - \frac{\partial k(x_\star, X)}{\partial x_j}\,(K + \sigma_n^2 I)^{-1}\,\frac{\partial k(X, x_\star)}{\partial x_j}$$

Notice how the matrix dimensions resolve to a scalar: a $1 \times n$ vector multiplied by an $n \times n$ matrix, multiplied by an $n \times 1$ column vector $\frac{\partial k(X, x_\star)}{\partial x_j}$.

Unlike MLS, which requires manual tuning of neighborhood sizes and polynomial degrees, the GP explicitly models observation noise and automatically tunes its length-scales and noise levels by maximizing the marginal likelihood.

Discarded estimates. Because the loss decreases monotonically in both model size and data, a non-negative estimated slope $\partial L/\partial N \geq 0$ or $\partial L/\partial D \geq 0$ can only come from noise. We therefore discard such points from the slope analyses of Section 2. This affects less than 1% of the estimates.

A.2 Form of the coupling

The non-zero mixed derivative of Section 2 constrains the functional form: it should preserve the marginal power-law behaviour visible in the first derivatives without making the two axes additively separable. The Skaling form (Equation (3)) satisfies this. Writing $u = AN^{-\alpha} + BD^{-\beta}$, its mixed derivative is

$$\frac{\partial^2 L}{\partial N\, \partial D} = k(k-1)\, u^{k-2}\, \alpha A\, N^{-\alpha-1}\, \beta B\, D^{-\beta-1}.$$

Thus $k = 1$ recovers the additive Chinchilla law and forces the mixed derivative to vanish, while any $k \neq 1$ introduces an interaction. In the empirically relevant case $0 < k < 1$, the mixed derivative is negative, matching the sign observed in Figure 2(c,f). At the same time, the first derivatives,

$$\frac{\partial L}{\partial N} = -k\alpha A\, N^{-\alpha-1} u^{k-1}, \qquad \frac{\partial L}{\partial D} = -k\beta B\, D^{-\beta-1} u^{k-1},$$

remain dominated by the same-variable power-law factors, with cross-variable dependence entering only through the shared factor u^{k-1}. Consequently, the Skaling law can appear nearly separable at first order while still allowing the non-zero mixed derivative indicated by the data.

Why a multiplicative coupling, not an additive interaction term. A natural alternative to the Skaling law would be to keep the Chinchilla law additive and append a separate product term,

$$L = AN^{-\alpha} + BD^{-\beta} + G\, N^{-\mu} D^{-\nu} + E.$$

This term can create a non-zero mixed derivative, but its sign creates an immediate constraint. For this model, the mixed derivative and the contribution of the interaction term to the size derivative are

$$\frac{\partial^2 L}{\partial N\, \partial D} = \mu\nu\, G\, N^{-\mu-1} D^{-\nu-1}, \qquad \frac{\partial L}{\partial N}\left(GN^{-\mu} D^{-\nu}\right) = -\,\mu\, G\, N^{-\mu-1} D^{-\nu}.$$

Thus the sign of G controls two quantities in opposite ways. Matching the observed negative mixed derivative requires $G < 0$. But with $G < 0$, the interaction term contributes positively to $\partial L/\partial N$, opposing the desired monotonic decrease with model size. This is not just a local inconvenience: the fitted mixed-derivative decay implies $\mu \approx 0.1$, smaller than the main size exponent α, so this positive contribution decays more slowly in N than the leading negative term. At large N and small D, it can therefore dominate and make $\partial L/\partial N > 0$, predicting that loss increases when the model becomes larger. Choosing $G > 0$ avoids this monotonicity failure, but then the mixed derivative becomes positive and the desired synergy disappears. Thus a single additive product term cannot simultaneously preserve monotonicity and match the observed negative interaction.

Skaling avoids this sign conflict because the interaction is introduced by a positive multiplicative factor rather than by a separate signed term. Its size gradient is

$$\frac{\partial L}{\partial N} = -k\,\alpha A\,N^{-\alpha-1}u^{k-1},$$

which is negative for every $k > 0$. The dependence on data enters only through the positive factor u^{k-1}. When $0 < k < 1$, increasing D decreases u, which increases u^{k-1} and therefore increases the magnitude of the already-negative size gradient. This yields $\partial^2 L/\partial N\partial D < 0$ without ever changing the sign of $\partial L/\partial N$. In this parameterization, monotonicity and synergy are compatible by construction.

The same factor also explains the asymmetric first-order slopes in Figure 2(b,d). Let $w_D = BD^{-\beta}/u$ and $w_N = AN^{-\alpha}/u$ denote the data and size shares of the inner sum. Differentiating $\ln|\partial L/\partial N|$ and $\ln|\partial L/\partial D|$ gives $\gamma_N = (1-k)\,\beta\,w_D$ and $\gamma_D = (1-k)\,\alpha\,w_N$, so $\gamma_N/\gamma_D = (\beta/\alpha)\,(w_D/w_N)$. This ratio exceeds one whenever $\beta > \alpha$ and the inner sum is data-leaning, reproducing the measured $\gamma_N \approx 0.13 > \gamma_D \approx 0.07$ through the single coupling exponent rather than through a skewed interaction term.

A.3 Empirical optimal token to parameter ratio

The numerical gradients also provide an empirical estimate of the compute-optimal allocation, without fitting a parametric scaling law. For a fixed training budget $C = 6ND$, feasible configurations lie on a single iso-compute curve: increasing model size requires decreasing the number of training tokens, and conversely. The empirical optimum for that budget is the lowest-loss point on this curve,

$$\min_{N,D}\ L(N,D) \quad \text{s.t.} \quad 6ND = C.$$

For an interior optimum on a differentiable loss surface, the Lagrangian

$$\mathcal{J}(N,D,\lambda) = L(N,D) + \lambda(6ND - C)$$

has stationarity conditions

$$\frac{\partial L}{\partial N} = -6\lambda D, \qquad \frac{\partial L}{\partial D} = -6\lambda N.$$

Dividing the two equations cancels the multiplier and leaves a condition on the loss gradients alone:

$$N\,\frac{\partial L}{\partial N} = D\,\frac{\partial L}{\partial D} \quad\Longleftrightarrow\quad \frac{\partial \ln L}{\partial \ln N} = \frac{\partial \ln L}{\partial \ln D}. \tag{5}$$

Thus, at a compute-optimal point, a 1% increase in model size and a 1% increase in data have the same signed effect on the loss. Equivalently, the directional derivative along the tangent to the iso-compute curve must vanish. This is a constrained stationarity condition: the full gradient is not required to vanish, since increasing the total compute would still reduce loss.

To solve this condition non-parametrically, we write the loss in log coordinates,

$$x = \ln N, \qquad y = \ln D, \qquad z(x,y) = \ln L(N,D).$$

The optimality condition becomes $\partial z/\partial x = \partial z/\partial y$.

Tracing the frontier. The measured runs rarely sit exactly at the optimum of their budget, so we cannot read the frontier off the grid directly. Both estimators of Section A.1, however, define the log-slopes as a continuous field that can be evaluated anywhere inside the data range: the GP by differentiating its posterior mean, and MLS by fitting its local polynomial around any query point. We use this field to solve the optimality condition one budget at a time.

For a budget C, we parameterize the iso-compute line by the log-ratio $s = \ln(D/N)$. With $S = \ln(C/6)$, the line is $x = (S - s)/2$, $y = (S + s)/2$, and moving along it changes the log-loss at rate

$$\frac{dz}{ds} = \frac{1}{2}\, g(x, y), \qquad g = \frac{\partial z}{\partial y} - \frac{\partial z}{\partial x}.$$

The sign of g therefore says which way to move: where $g < 0$, shifting compute from parameters to data still lowers the loss; where $g > 0$, it raises it. The compute-optimal ratio is the point where g changes sign from negative to positive, which is a minimum of the loss along the line (a positive-to-negative change would be a maximum). We evaluate g on a fine grid of s, restricted to the part of the line covered by the runs, and locate the sign change by linear interpolation. If noise produces more than one such crossing, we keep the first along increasing D/N. Tracing this crossing as C varies gives the empirical frontier $D^\star/N^\star$.

The GP and MLS frontiers are computed from the same runs with the same procedure, but rest on different assumptions (a single smooth global surface versus local polynomial fits), so their agreement is a check that the frontier is not an artifact of either estimator.

In both cases, tracking $D^\star/N^\star$ as C varies gives an empirical token-to-parameter ratio, which we compare with the closed-form R_{opt} of Section B on the Farseer data (Figure 5). The two independent estimates agree qualitatively: both recover a ratio that decreases with compute, closely following Skaling's R_{opt}, whereas Chinchilla predicts a nearly flat ratio. Fitting each empirical frontier as a power law $D^\star/N^\star \propto C^m$ gives $m \approx -0.14$ for the GP and $m \approx -0.15$ for MLS. At 2×10^{25} FLOPs, Chinchilla approaches $\sim$380 tokens per parameter, while the Skaling law falls to $\sim$45 and the empirical fits to $\sim$25.

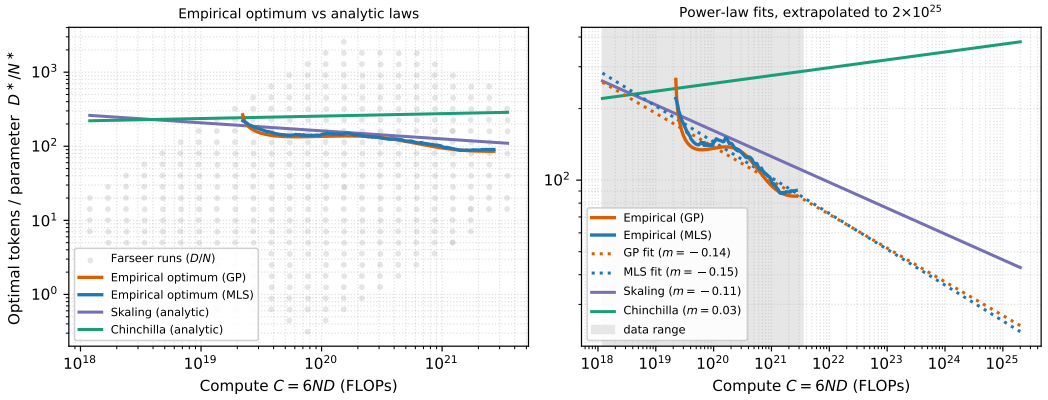


Figure 5: **Empirical compute-optimal token-to-parameter ratio on Farseer, recovered without a parametric fit.** The optimum $D^\star/N^\star$ is located where the log-slopes balance ($\partial \ln L/\partial \ln N = \partial \ln L/\partial \ln D$), using a global GP surrogate and a local MLS surrogate (both evaluated as continuous mesh-free fields), and compared with the closed-form R_{opt} of Skaling and Chinchilla fitted once on all Farseer runs. *Left:* both empirical optima, over the run cloud, follow Skaling more closely than the nearly flat Chinchilla prediction. *Right:* power-law fits of each empirical frontier (dotted) against the analytic laws, extrapolated to 2×10^{25} FLOPs (shaded = data range); the empirical exponents (-0.14, -0.15) are close to Skaling (-0.11) and have the opposite sign from Chinchilla ($+0.03$).

B COMPUTE-OPTIMAL TRAINING

The primary utility of a scaling law is to determine the optimal allocation of a given compute budget C between model size N and data volume D. We formally define the training compute budget as $C = 6ND$. To identify the compute-optimal configuration, we minimize the Skaling loss of Equation (3) subject to this budget constraint.

By substituting $D = \frac{C}{6N}$, we can isolate the inner additive term as a function of N, defining $Z(N) = \frac{A}{N^\alpha} + B\left(\frac{C}{6N}\right)^{-\beta}$. Our loss objective then simplifies to:

$$L(N) = Z(N)^k + E \tag{6}$$

Taking the derivative with respect to N yields:

$$\frac{dL}{dN} = k \cdot Z(N)^{k-1} \cdot Z'(N) \tag{7}$$

Since $Z(N)$ is a sum of strictly positive terms and our empirical fits consistently yield $k > 0$, the scaling factor $k \cdot Z(N)^{k-1}$ is non-zero. This means minimizing the loss strictly requires that $Z'(N) = 0$, precisely the stationarity condition of the additive Chinchilla law (Hoffmann et al., 2022); the Skaling law therefore inherits Chinchilla's compute-optimal allocation unchanged. Solving $Z'(N) = -\alpha A\, N^{-\alpha-1} + \beta B\,(6/C)^\beta N^{\beta-1} = 0$ gives $N^{\alpha+\beta} = \frac{\alpha A}{\beta B}(C/6)^\beta$; substituting $D^* = C/(6N^*)$ then yields the optimal token-to-parameter ratio (i.e., D^*/N^*):

$$R_{opt} = 6^{\frac{\beta-\alpha}{\alpha+\beta}} \left(\frac{\beta B}{\alpha A}\right)^{\frac{2}{\alpha+\beta}} C^{\frac{\alpha-\beta}{\alpha+\beta}} \tag{8}$$

This reveals that if $\alpha \approx \beta$ (as observed in Chinchilla), the optimal ratio D/N remains constant across scales. More broadly, the Skaling law cleanly separates the shape of the loss landscape (controlled by k) from the location of the optimal allocation ($Z'(N) = 0$): the monotone outer map $x \mapsto x^k + E$ rescales the loss but leaves its minimizer unchanged, so the Skaling law gains expressivity without sacrificing tractability.

C FITTING DETAILS

C.1 CHALLENGES

Despite the small number of parameters, fitting scaling laws to empirical loss data is a surprisingly fragile non-convex optimization problem. Several structural factors contribute to this instability. First, the parameters operate on vastly different numerical scales. For instance, the large magnitudes of coefficients A and B contrast sharply with the fractional exponents α and β, ill-conditioning the problem for standard gradient-based optimizers like L-BFGS. Second, the optimization landscape is notoriously flat. Because the loss decreases as a slow power law, optimizers are prone to halting prematurely, a vulnerability that is worsened when relying on finite-difference gradient approximations. Finally, the parameters exhibit strong compensatory behaviors; the irreducible loss E, in particular, is difficult to estimate accurately when, as in our grids, the runs stop well short of the scales at which the loss approaches it, because other coefficients can easily shift to offset its value. This creates wide valleys of distinct local minima that yield nearly identical overall surface fits. Section F evaluates a fitting variant that removes E from the objective altogether.

We fit every law by minimizing the objective in log space, which (as in Chinchilla) absorbs much of this scale disparity. Within this setup, a carefully configured L-BFGS (Hoffmann et al., 2022), paired with basin-hopping (Shukor et al., 2025) and automatic differentiation, and the scale-independent, gradient-free CMA-ES (Hansen, 2016) reach equally good fits. The difference we observe is practical rather than in final quality: L-BFGS must be carefully tuned (initialization, restarts, and the log-space objective) to reach them reliably, whereas CMA-ES attains the same solutions out of the box, without any such tweaks.

C.2 HYPERPARAMETERS

We implement two global optimization strategies. The first, L-BFGS-B with basin-hopping, pairs a gradient-based local optimizer (L-BFGS-B with analytical gradients via autograd) with multi-start basin-hopping for global search; starting points are drawn from a Sobol quasi-random sequence over the bounded parameter space. The second, BIPOP-CMA-ES (Hansen, 2016; 2009), is a gradient-free evolutionary strategy; we use doubled population size, active CMA, and 9 BIPOP restarts. As discussed in Section C.1, both reach equally good fits; all results reported in this paper use L-BFGS-B with basin-hopping, as quasi-Newton optimizers are the standard choice in the literature (L-BFGS in Hoffmann et al., 2022; BFGS in Besiroglu et al., 2024) and L-BFGS has been combined with basin-hopping for scaling-law fitting (Shukor et al., 2025). Every law minimizes a Huber loss ($\delta = 0.05$) in log space; coefficients A and B are optimized in log scale to absorb their large dynamic range. The per-law configurations are summarized in Table 4.

Table 4: Per-law parameter bounds and fitting configuration. All laws use 2000 basin-hopping restarts, a log-space Huber objective, and analytic (autograd) gradients; *Params* is the number of free parameters and *Log params* those optimized in log scale. Farseer uses the stretched-exponential forms of Li et al. (2025a) (their Eq. 3).

Law	Params	Log params	Bounds
Chinchilla	5	A, B	$A \in [10^{-6}, 10^{4}]$, $B \in [10^{-6}, 5{\times}10^{4}]$, $\alpha, \beta \in [0, 1]$, $E \in [0, 3]$
Skaling	6	A, B	$A, B \in [10^{-6}, 10^{7}]$, $\alpha, \beta, k \in [0.01, 2]$, $E \in [0, 3]$
Farseer	9	—	$E \in [0, 5]$, $s \in [-10, 10]$, $q \in [-0.5, 0.5]$, $S \in [-30, 10]$, $B_c \in [-10, 500]$, $b \in [-2, 0.5]$, $Q \in [-10, 15]$, $A_c \in [-25, 5]$, $a \in [-0.5, 0.2]$

Fitting Farseer. We fit Farseer, like every other law, with this common direct optimizer. Beyond keeping the comparison controlled, this is a practical necessity: Farseer's original pipeline estimates its parameters stage by stage from loss differences taken along the axes of a complete grid (consecutive D values at fixed N). Cross-validation hold-outs and the sparse L-shape remove points and break this grid structure, so the pipeline cannot be applied uniformly across our settings, whereas direct optimization can. Where we could run the original pipeline under the same cross-validation, it did not give clearly better predictions: its errors lay within the fold-to-fold standard deviation of the direct fit in nearly every regime.

D ADDITIONAL DATASETS

We repeat the cross-validation protocol of Section 3.3 on two further datasets (Table 5). *Farseer-code* is the code-domain counterpart of the Farseer grid (Li et al., 2025a): 117 runs over 9 model sizes (201M–3.18B) and 20 token budgets (2B–128B), spanning compute $C{=}6ND$ from 2.4×10^{18} to 2.4×10^{21} FLOPs. Being grid-structured, it admits both the full-grid and L-shape splits; its larger-N and larger-D hold-outs are the largest model sizes and longest token horizons at the top edge of the grid. The *Chinchilla* loss measurements (Besiroglu et al., 2024) are 245 scattered (N, D) points covering 57M–16.2B parameters and 245M–318B tokens (1.4×10^{18}–1.3×10^{22} FLOPs); they do not lie on a regular grid, so only the full-grid (random) split applies (no L-shape), with the largest-N (16.2B) and largest-D (318B-token) points forming the two extrapolation sets.

Table 6 lists the corresponding coefficients for Farseer-code and the Chinchilla data. The same pattern holds, with the Skaling law fitting $k < 1$ and a smaller E than Chinchilla, though the coupling is weaker here ($k \approx 0.77$–0.90), in line with the more mixed accuracy gains in Table 5.

E PRE-TRAINING DETAILS

E.1 MODEL

The configurations in this section describe the SK-Grid, an internal set of pretraining runs; for the Farseer grid we use the released runs and refer the reader to Li et al. (2025a) for details. SK-Grid

Table 5: Fit quality and predictive error on two additional datasets, following the protocol of Table 1: interpolation R^2 (↑) and MAPE (%, ↓; mean ± std over 5 CV folds) on interpolation and two extrapolation regimes (larger N, larger D). Farseer-code is grid-structured (full and L-shape splits); the Chinchilla measurements are not gridded, so only a full-grid split applies. Best per column within each block in **bold**.

	Farseer-code (full)				Farseer-code (L-shape)				Chinchilla (full)			
Law	R^2	Interp.	Ext. N	Ext. D	R^2	Interp.	Ext. N	Ext. D	R^2	Interp.	Ext. N	Ext. D
Chinchilla	0.998	0.28±0.07	0.93±0.15	0.60±0.04	0.991	0.59±0.06	1.93±0.18	1.60±0.12	**0.993**	0.63±0.15	**1.16±0.33**	0.63±0.06
Farseer	0.983	0.89±0.12	1.01±0.04	3.74±0.23	0.993	0.53±0.16	**0.96±0.27**	**0.91±0.68**	0.955	1.27±0.55	1.23±1.10	1.12±0.21
Skaling	**0.999**	**0.24±0.09**	**0.67±0.19**	**0.26±0.06**	**0.995**	**0.45±0.12**	1.39±0.49	1.14±0.41	**0.993**	**0.61±0.14**	1.28±0.29	**0.51±0.03**

Table 6: Fitted coefficients (mean ± std over 5 CV folds) behind Table 5. Chinchilla is the k=1 special case of Skaling.

Setup	Law	A	B	α	β	k	E
Farseer-code, full	Chinchilla	$(1.8\pm1.1)\times10^3$	$(5.6\pm0.8)\times10^2$	0.48±0.03	0.34±0.01	1	0.65±0.01
	Skaling	$(4.2\pm2.8)\times10^3$	$(2.0\pm0.2)\times10^3$	0.52±0.03	0.40±0.01	0.77±0.03	0.60±0.02
Farseer-code, L-shape	Chinchilla	$(5.4\pm1.2)\times10^3$	$(1.5\pm0.1)\times10^3$	0.55±0.01	0.39±0.00	1	0.71±0.00
	Skaling	$(4.1\pm1.4)\times10^3$	$(2.1\pm0.5)\times10^3$	0.53±0.02	0.40±0.01	0.90±0.06	0.67±0.03
Chinchilla, full	Chinchilla	$(7.0\pm1.5)\times10^2$	$(1.3\pm0.3)\times10^4$	0.37±0.02	0.45±0.01	1	1.91±0.02
	Skaling	$(4.9\pm3.0)\times10^3$	$(1.1\pm0.7)\times10^5$	0.45±0.04	0.53±0.03	0.77±0.06	1.85±0.01

comprises 125 runs across 14 model sizes (Table 7). The 3 far-extrapolation runs at $\sim10^{22}$ FLOPs (Section 4) are trained with the same recipe but are not part of the grid; they are listed separately at the bottom of the table.

Architecture. The models span 14 sizes from 134M to 4.9B parameters, scaled by growing width and depth together: the model dimension d_{model} increases from 672 to 3264 and the depth from 7 to 34 layers. The far-extrapolation models (5.8B–10.8B) continue this scaling with a head dimension of 128. All models use the Llama 3 tokenizer, with a vocabulary of 128,256.

Token budgets. Each size is trained on a geometric ladder of token budgets spanning 316M to 316B tokens, five budgets per decade. Per-run compute is capped, so larger models are trained on fewer budgets: the smallest models cover all 16 horizons while the largest is trained on a single budget, producing the staircase grid whose row counts appear in the last column of Table 7.

E.2 Hyperparameters

The two run-specific hyperparameters, the global batch size B (tokens) and the peak learning rate η, follow power laws in the style of Step Law (Li et al., 2025b), refit to our own setup. Following Bi et al. (2024), we characterize model scale by the per-token compute F (FLOPs per token) rather than by the parameter count; the batch size depends on F and the learning rate on F and the token budget D:

$$B = 896.07\, F^{0.231}, \qquad \eta = 0.0709\, F^{-0.4303}\, D^{0.2785}. \tag{9}$$

Because the shape of the loss surface is highly sensitive to these tuning choices, different parameterizations of hyperparameters in the training recipe can alter the apparent N–D interaction and shift the inferred compute-optimal ratio. For instance, a broadly mistuned grid might artificially dampen the measured interaction or skew the optimal ratio; this is why we tune each configuration near its optimum rather than reuse a single setting. This sensitivity also implies that differences in hyperparameter policies, such as the original Chinchilla recipe compared to our Step Law-tuned SK-Grid, may partly explain why different datasets exhibit varying degrees of N–D coupling. Consequently, cross-dataset comparisons inherently reflect the specific training recipes and data used to generate each grid.

All remaining settings are fixed across the grid (Table 8), based on the defaults of the Meta Lingua framework (Videau et al., 2024). Every run uses the same data mixture: 60% DCLM-Edu web text (Li et al., 2024; Ben Allal et al., 2025), 30% code, and 10% math. The loss we fit is the validation loss

on a held-out split of this mixture. For the other datasets, refer to their respective papers: Farseer (Li et al., 2025a) and Chinchilla (Besiroglu et al., 2024; Hoffmann et al., 2022).

Table 7: SK-Grid model configurations. *Budgets* is the number of token budgets per size; each far-extrapolation run is trained on a single budget, whose token count is given instead.

N	d_{model}	Layers	Heads	d_{head}	Budgets
134M	672	7	14	48	16
177M	864	9	9	96	16
234M	960	10	10	96	15
308M	1056	11	11	96	14
407M	1248	13	13	96	11
537M	1440	15	15	96	11
708M	1632	17	17	96	9
935M	1824	19	19	96	9
1.23B	2016	21	21	96	7
1.63B	2208	23	23	96	5
2.15B	2496	26	26	96	5
2.83B	2688	28	28	96	4
3.74B	2976	31	31	96	2
4.93B	3264	34	34	96	1
Far extrapolation					
5.83B	3328	34	26	128	344B
7.91B	3712	39	29	128	244B
10.8B	4224	43	33	128	172B

Table 8: Fixed configuration shared by all SK-Grid runs; per-run LR and batch size follow Equation (9).

Architecture	
Position enc.	RoPE (θ=10^4)
Vocabulary	128,256
Sequence len.	2048
Optimization	
Optimizer	AdamW
(β_1, β_2)	$(0.9, 0.95)$
Weight decay	0.1
Grad. clip	0.1
LR schedule	cosine
Warmup	10%
Final LR	1×10^{-6}

F DOMINATED-PAIR FITTING

A recurring difficulty above is that the irreducible loss E is only weakly identified: interior points constrain it poorly and the other coefficients shift to absorb it. The *dominated-pair* fit removes E from the objective entirely. For any ordered pair in which configuration i dominates j ($N_i \geq N_j$, $D_i \geq D_j$ and $L_i < L_j$), the additive floor cancels in the loss difference:

$$L_j - L_i = \left(A\,N_j^{-\alpha} + B\,D_j^{-\beta}\right)^k - \left(A\,N_i^{-\alpha} + B\,D_i^{-\beta}\right)^k, \tag{10}$$

which no longer involves E (Chinchilla is the k=1 case). We fit the shape parameters (A, B, α, β, k) on all such pairwise differences, then recover the floor as $E = \text{median}_m \left(L_m - (A\,N_m^{-\alpha} + B\,D_m^{-\beta})^k\right)$, decoupling the reducible shape from the constant offset.

Table 9 shows that dominated-pair fitting serves primarily as a correction for the additive Chinchilla law. On the two main datasets, it reduces most of Chinchilla's extrapolation errors, yielding the clearest gains in boundary regimes where accurately fitting the irreducible loss floor matters most. For example, on the full grids, Chinchilla's far-extrapolation error drops from 2.46% to 0.79% on Farseer and from 5.17% to 3.67% on SK-Grid. On their L-shape grids, Chinchilla improves on every reported metric.

Part of Chinchilla's extrapolation error therefore stems from weak identification of the floor E; once E is removed from the objective, the constant offset no longer distorts the fit of the reducible component. The pairwise objective also acts as an implicit reweighting toward the endpoints of the grid. This lets Chinchilla discount the curvature at the beginning of the loss curves, which its additive form cannot represent, and fit the high-compute end more closely, hence the large gain in far extrapolation. This is consistent with our main finding rather than at odds with it: the additive law extrapolates well only when it gives up fitting the region where the N–D interaction is visible. Skaling, which represents that curvature directly, gains little from the reweighting and does not consistently improve.

Table 9: Dominated-pair ("+dom") fitting versus the default joint L-BFGS fit, for Chinchilla and Skaling on every dataset and grid. Each "+dom" row is the dominated-pair fit (Equation (10)) of the law above it. Columns: interpolation R^2 (↑) and MAPE (%, ↓; mean ± std over 5 CV folds) on interpolation and the larger-N, larger-D, and far regimes. The Chinchilla measurements are not gridded, so they have no L-shape and no far set; the Farseer-code far set is a single held-out run and is omitted (–). Both variants come from one fitting run, so the baselines can differ marginally from Table 1. Best per column within each panel in **bold** (far excluded where not comparable).

Law	R^2	Interp.	Ext. N	Ext. D	Far	R^2	Interp.	Ext. N	Ext. D	Far
	Farseer (full)					*Farseer (L-shape)*				
Chinchilla	0.995	0.77±0.04	1.48±0.03	1.98±0.08	2.46±0.19	0.954	2.51±0.07	4.32±0.13	3.29±0.11	9.82±0.48
+dom	0.995	0.82±0.06	1.40±0.06	2.61±0.22	**0.79±0.08**	0.959	2.41±0.17	3.82±0.18	2.89±0.13	7.84±0.87
Skaling	**0.998**	**0.41±0.05**	0.47±0.03	0.88±0.06	2.32±0.18	0.995	0.84±0.10	0.87±0.23	**1.35±0.20**	**1.57±0.62**
+dom	**0.998**	**0.41±0.04**	**0.39±0.01**	**0.71±0.07**	1.55±0.17	**0.996**	**0.67±0.09**	**0.80±0.25**	2.29±0.79	5.87±1.79
	SK-Grid (full)					*SK-Grid (L-shape)*				
Chinchilla	0.992	0.81±0.14	0.83±0.11	1.44±0.03	5.17±0.28	0.955	2.19±0.10	6.09±0.24	3.63±0.13	14.63±0.39
+dom	0.991	0.80±0.19	0.59±0.05	1.38±0.13	3.67±0.20	0.971	1.70±0.15	4.96±0.39	3.07±0.20	13.07±0.60
Skaling	**0.998**	0.33±0.11	**0.39±0.05**	0.58±0.07	0.70±0.39	**0.998**	**0.33±0.03**	**0.77±0.44**	**0.55±0.08**	**1.16±0.53**
+dom	**0.998**	**0.32±0.12**	0.46±0.05	**0.45±0.05**	**0.41±0.21**	0.996	0.54±0.19	2.08±0.94	0.66±0.27	2.99±2.05
	Farseer-code (full)					*Farseer-code (L-shape)*				
Chinchilla	0.998	0.28±0.07	0.93±0.16	0.60±0.04	–	0.991	0.59±0.06	1.93±0.18	1.60±0.12	–
+dom	0.998	0.29±0.06	0.80±0.06	0.65±0.05	–	0.991	0.61±0.06	1.99±0.09	1.60±0.11	–
Skaling	**0.999**	0.24±0.09	0.67±0.19	0.26±0.06	–	0.993	0.51±0.11	**1.51±0.59**	**1.23±0.50**	–
+dom	**0.999**	**0.23±0.09**	**0.60±0.12**	**0.24±0.06**	–	**0.994**	**0.50±0.15**	1.57±0.43	1.27±0.41	–
	Chinchilla (full)									
Chinchilla	**0.993**	0.63±0.15	**1.16±0.33**	0.63±0.06	–					
+dom	0.991	**0.59±0.26**	1.52±0.14	0.47±0.02	–					
Skaling	**0.993**	0.61±0.14	1.28±0.29	0.51±0.03	–					
+dom	0.990	0.64±0.25	1.43±0.24	**0.44±0.03**	–					