Farseer: A Refined Scaling Law in Large Language Models

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Zhenyu Ding, Haoying Wang, Zili Wang, Shijie Xuyang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, Daxin Jiang

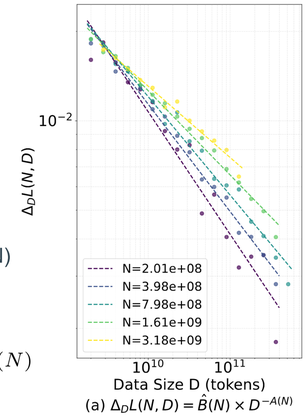
NEURAL INFORMATION PROCESSING SYSTEMS

Key Discovery: N-Dependent Data Scaling

Non-Uniform Scaling: Unlike Chinchilla's assumption of a constant data scaling rate, we discovered that data scaling properties vary significantly with model size (N).

Interaction Term: There is a strong coupling between model size (N) and data size (D). The finite difference of loss $\Delta_D L(N, D)$ follows a power-law where the exponent A(N) and coefficient B(N) are functions of N, not constants.

Equation: $\Delta_D L(N, D) \approx B(N) D^{-A(N)}$



The Farseer Formula

$$L(N, D) = e^{a_3 N^{\gamma} + b_3} + e^{a_2 N^{\beta} + b_2} \cdot D^{-e^{a_1 N^{\alpha} + b_1}}$$

Methodology: Differential Piecewise Fitting

Step 1: Compute finite difference $\Delta_D L$ to isolate data-dependent terms.

Step 2: Fit discrete exponents A_N and coefficients B_N via linear regression.

Step 3: Refine continuous functions $f_A(N)$ and $f_B(N)$ to minimize global error.

Step 4: Model the residual term G(N) to capture pure model-size scaling.

Algorithm 2: Differential Piecewise Fitting (with Stretched-Exponential Forms)

Input: Loss Data points $L(N, D)$, scale factor λ.
Output: Parameters $\theta_A^* = (a_1^*, b_1^*, \alpha^*)$, $\theta_B^* = (a_2^*, b_2^*, \beta^*)$, $\theta_{L_N}^* = (a_3^*, b_3^*, \gamma^*)$.
Output: Final fit $L(N, D) \approx \exp(a_2^* N^{\beta^*} + b_2^*) D^{-\exp(a_1^* N^{\alpha^*} + b_1^*)} + \exp(a_3^* N^{\gamma^*} + b_3^*)$.

// Stage 1: Initial estimation of discrete A_N, B_N
foreach *model size* N **do**
  Compute $R_N(D) \leftarrow L(N, D) - L(N, \lambda D)$
  // From text: $R_N(D) \approx \hat{B}_N D^{-A_N}$
  Estimate $A_N, \hat{B}_N$ via linear fit on
    $\log(R_N(D)) = \log(\hat{B}_N) - A_N \log(D)$
  // Linear fit parameters can be found using the Normal Equation.
  $B_N \leftarrow \hat{B}_N / (1 - \lambda^{-A_N})$
Collect discrete sets $\{A_N\}$ and $\{B_N\}$
// Stage 2: Parameterization and Iterative Refinement of $f_A(N; \theta_A)$ and $f_B(N; \theta_B)$
// Assumed forms: $f_A(N; \theta_A) = \exp(a_1 N^{\alpha} + b_1)$, $f_B(N; \theta_B) = \exp(a_2 N^{\beta} + b_2)$.
Fit $\log(B_N) = a_2 N^{\beta} + b_2$ to $\{(N, B_N)\}$ to find initial a_2, b_2, β
// Similarly, for each β, (a_2, b_2) are found via linear regression (e.g., Normal Equation) minimizing $\sum_N (\log(B_N) - (a_2 N^{\beta} + b_2))^2$.
// Iterative refinement of exponents α, β
Let $\hat{R}_N(D; \theta_A, \theta_B) = f_B(N; \theta_B)(1 - \lambda^{-f_A(N; \theta_A)}) D^{-f_A(N; \theta_A)}$
Let global residual error $\ell_R = \sum_N \sum_D (R_N(D) - \hat{R}_N(D; \theta_A, \theta_B))^2$
repeat
  Fix β, a_2, b_2. Update α (and re-estimate a_1, b_1) to minimize ℓ_R
  // This involves finding α (e.g., via grid search). For each candidate α, (a_1, b_1) are re-calculated by fitting $\log(A_N) = a_1 N^{\alpha} + b_1$. The set (α, a_1, b_1) that minimizes the global residual ℓ_R is chosen.
  Fix updated α, a_1, b_1. Update β (and re-estimate a_2, b_2) to minimize ℓ_R
  // Similarly, this involves finding β. For each candidate β, (a_2, b_2) are re-calculated by fitting $\log(B_N) = a_2 N^{\beta} + b_2$. The set (β, a_2, b_2) that minimizes the global residual ℓ_R is chosen.
until *convergence (e.g., 1-2 iterations or small change in* ℓ_R*)*
Obtain refined parameters $\theta_A^* = (a_1^*, b_1^*, \alpha^*)$ and $\theta_B^* = (a_2^*, b_2^*, \beta^*)$ from the best fit
// Stage 3: Fit model-dependent residual $E + L_N(N)$
// Assumed form: $f_{L_N}(N; \theta_{L_N}) = \exp(a_3 N^{\gamma} + b_3)$.
Compute $O(N, D) \leftarrow L(N, D) - f_B(N; \theta_B^*) D^{-f_A(N; \theta_A^*)}$
$G(N) \leftarrow \text{Avg}_D[O(N, D)]$
Fit $\log(G(N)) = a_3 N^{\gamma} + b_3$ to $\{(N, G(N))\}$ to find a_3, b_3, γ
// This involves finding γ (e.g., grid search); for each γ, (a_3, b_3) are found via linear regression (e.g., Normal Equation) minimizing $\sum_N (\log(G(N)) - (a_3 N^{\gamma} + b_3))^2$.
Obtain final parameters $\theta_{L_N}^* = (a_3^*, b_3^*, \gamma^*)$
// Final fitted Scaling Law
The final scaling law is:

$$L(N, D) \approx \exp(a_2^* N^{\beta^*} + b_2^*) D^{-\exp(a_1^* N^{\alpha^*} + b_1^*)} + \exp(a_3^* N^{\gamma^*} + b_3^*)$$

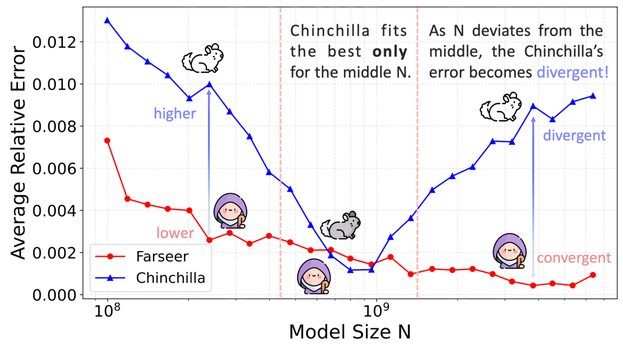
return $\theta_A^*, \theta_B^*, \theta_{L_N}^*$

The Scaling Gap & Motivation

Training LLMs is prohibitively expensive (>10^25 FLOPs). Current scaling laws (e.g., Chinchilla) are fitted on limited ranges and fail to transfer insights from small-scale experiments to production scales.

Chinchilla's Limitation: Chinchilla's law assumes a uniform data scaling rate across all model sizes (N). It fits well only near the middle range but diverges significantly at extremes, leading to high extrapolation errors.

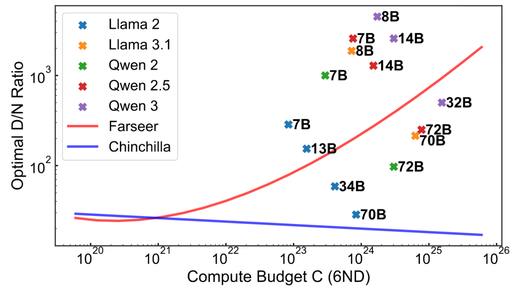
Farseer, a refined scaling law that explicitly models the interaction between model size (N) and data size (D), reducing extrapolation error by 433% compared to Chinchilla.



New Insights on Compute Allocation

Chinchilla's Rule: Suggests a constant Optimal D/N ratio regardless of compute budget. Farseer's Insight: The optimal D/N ratio increases as the compute budget (C) grows. Larger models require proportionally more data than previously thought.

This trend aligns with recent SOTA models (e.g., Llama 3.1, Qwen 2.5) which use much higher D/N ratios.

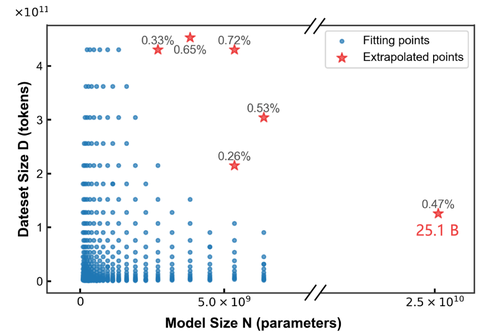


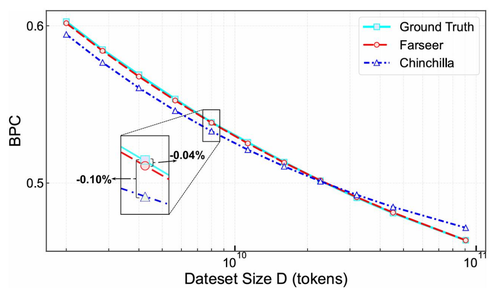
Fitting & Extrapolation Accuracy

Fitting: Farseer achieves significantly lower fitting error than Chinchilla across the entire parameter range.

Extrapolation: Farseer accurately predicts the performance of a 25.1B model (an order of magnitude larger than training data) with a relative error of only 0.47%.

Comparison: Chinchilla's extrapolation error is 433% higher (2.68%).





Open Source

Github: github.com/Farseer-Scaling-Law/Farseer
HuggingFace: huggingface.co/Farseer-Scaling-Law
Home: https://farseer-scaling-law.github.io/
Wandb: https://wandb.ai/billzid/Farseer

References

Jared Kaplan et al. (2020). "Scaling laws for neural language models." In: arXiv preprint arXiv:2001.08361.

Jordan Hoffmann et al. (2022). "Training Compute-Optimal Large Language Models." In: arXiv preprint arXiv:2203.15556.

Hugo Touvron et al. (2023). "Llama: Open and efficient foundation language models." In: arXiv preprint arXiv:2302.13971.

Houyi Li et al. (2025). "Predictable Scale: Part I -- Optimal Hyperparameter Scaling Law in Large Language Model Pretraining." In: arXiv preprint arXiv:2503.04715.