How to Allocate Your Tokens? Scaling Laws with Training Steps and Batch Size

Anonymous authors
Paper under double-blind review

Abstract

We propose a scaling law that takes into account model size and training data while explicitly splitting the latter into training steps and batch size (called *three-term law*). Fitting the proposed law on a large set of training runs, we find that it correctly recovers the scaling of the optimal batch size. Making use of training runs with suboptimal batch size, our proposed law can be robustly fit with a significantly smaller amount of training runs. We further show that the three-term law can be used to derive scaling laws for suboptimal batch sizes, and that it matches previous empirical findings related to the critical batch size.

1 Introduction

The field of deep learning and specifically large language models (LLMs) has seen an enormous progress over the last few years. Much of this progress has been attributed to "simply" scaling up, both in terms of model size and data used for training. Improvements in model performance often follow predictable trends, called *scaling laws*, which have been found in the context of LLM pre-training (Kaplan et al., 2020; Hoffmann et al., 2022), but also in many other areas of deep learning such as vision (Zhai et al., 2022), weather forecasting (Bodnar et al., 2024), or protein modeling (Lin et al., 2023).

Scaling laws for model size and data. In the context of training large language models, scaling laws classically refer to functional forms that allow to infer the optimal allocation of compute into model size N and training examples D. Seminal works by Kaplan et al. (2020) and Hoffmann et al. (2022) show that the test loss predictably decreases when increasing N or D. A widely-used technique to obtain a scaling law, known as Chinchilla Approach 3, is to model the (final test) loss as a sum of power-laws in N and D, i.e.

$$\mathcal{L}(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}. \tag{1}$$

The parameters (E, A, B, α, β) are fitted from a set of training runs. One can then derive the optimal model size from (1) for a given compute constraint, and potentially extrapolate this to values of (N, D) outside of the experimentally tested ranges. On the other hand, several works show that the fitting procedure itself is delicate, and design choices in both the training procedure as well as the fitting techniques can impact the result (Besiroglu et al., 2024; Li et al., 2025b). Further, it is not guaranteed that the law generalizes well, which however is crucial to derive optimal configurations for large-scale training runs.

Hyperparameter scaling laws. While scaling laws of the Chinchilla form can inform compute-optimal allocation of N and D, they do not explicitly account for training hyperparameters. To address this, several recent works derive scaling laws for the optimal learning rate and/or batch size as a function of (N, D) (DeepSeek-AI et al., 2024; Li et al., 2025a; von Rütte et al., 2026). Typically, these works assume a power-law relationship for the hyperparameter of interest (e.g. optimal batch size as a function of D) and fit the coefficients to data from training runs. However, these approaches are not directly compatible with laws of form (1) as they do not model the loss value.

A different line of work is based on the concept of the *critical batch size* (McCandlish et al., 2018; Shallue et al., 2018): it describes the phenomenon that the number of steps K required to reach a target loss, as a function of the batch size, will at some point decrease much slower than inversely linear. McCandlish et al. (2018) model this with the equation

$$(K/K_{\min} - 1)(D/D_{\min} - 1) = 1. \tag{2}$$

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.

Here, $K_{\min}$ (resp. $D_{\min}$) is the minimum number of steps (resp. number of tokens) required to reach the target loss, and can be fit empirically. The critical batch size is then defined as $D_{\min}/K_{\min}$. Bergsma et al. (2025) establish a connection to weight decay. In an earlier work, Kaplan et al. (2020) use the same model of critical batch size to relate the loss to the number of training steps.

A central issue with (2) is that it implies an *optimal batch size of one* (Bergsma et al., 2025); however, this is in conflict with the empirical situation when using `AdamW`, where the optimal batch size has been found to scale with the available token budget D (Porian et al., 2024; Li et al., 2025a).

Scaling laws and optimization theory. From a theoretical point of view, the relationship between loss, batch size and training steps is inherently related to optimization theory. Recent works by Shulgin et al. (2026); Islamov et al. (2026) derive hyperparameter scaling laws directly from convergence bounds for stochastic conditional gradient methods. Further, it was shown that hyperparameter tuning/transfer for LLM training can be informed by optimization theory, e.g. in the context of learning-rate schedules (Schaipp et al., 2025) or weight decay (Wang & Aitchison, 2025).

2 Overview

Our proposed laws. Here, we propose to model the loss as a power-law function of (N, M, K) where M is the batch size in tokens and K is the number of training steps, that is,

$$\mathcal{L}(N, M, K) = E + \frac{A}{N^\alpha} + \frac{B}{M^\beta} + \frac{C}{K^\gamma}.$$

This has several natural advantages:

(i) Our law brings together the Chinchilla form (1) with scaling laws for the optimal batch size. Under a constrained data budget $D = MK$, our law implies a scaling rule for the optimal batch size with D, while at the same time collapsing to a Chinchilla form when using the optimal batch size.

(ii) The proposed form is also closely connected to and inspired by theoretical results from stochastic optimization, see Shulgin et al. (2026); Kovalev (2025). This allows to bridge from empirical scaling analysis to a more theoretical understanding.

(iii) The proposed law can be fit with runs from suboptimal batch sizes, which (as we will show) drastically reduces the number of training runs needed for fitting.

(iv) While previous scaling laws only model the performance with *optimal* hyperparameters, our formulation describes performance also in the suboptimal batch size regime. This can be important in practice when facing hardware constraints.

Summary of our findings. We fit our proposed laws on two datasets of training runs of (dense) LLMs, from here on referred to as **Li** (Li et al., 2025a) and **OpenEuroLLM** (Ajroldi et al., 2026). Both datasets cover multiple model sizes, token budgets and batch sizes.

(i) Our law results in an implied optimal batch size scaling that is consistent with previous hyperparameter scaling laws that do not model the loss value (see Section 4.1). In particular, we find that with our formulation two batch sizes per (N, D) suffice to robustly find this law (instead of doing a full sweep); this reduces the number of training runs needed to 28% (see Section 4.2).

(ii) By construction, our proposed law results in non-trivial optimal batch sizes (in contrast to previous formulations) and that are independent of model size, matching the empirical results of Li et al. (2025a). At the same time, it describes how the critical batch size scales with N and/or D consistently with findings by Zhang et al. (2025) (see Section 4.4).

(iii) For situations where the optimal batch size might be infeasible due to practical constraints, we derive scaling laws for ε-suboptimal batch sizes that generalize well to out-of-sample token budgets (see Section 4.3).

Notation. The table below summarizes the most important notation used throughout the document. Note that it holds $D = MK$.

Main quantities	
$\mathcal{L}$	Loss value
N	Number of parameters (model size)
D	Size of training data (in tokens)
b	Batch size
s	Sequence length
K	Number of training steps
M	Batch size (in tokens). It holds $M = bs$.
$\mathcal{C}$	Compute (in flops). Usually modeled as $\mathcal{C} = 6ND$.

3 Scaling Laws with Training Steps and Batch Size

Recall the Chinchilla law proposed by Hoffmann et al. (2022): for $A, B, E > 0$ and $\alpha, \beta > 0$ let the loss be parametrized as

$$\mathcal{L}(N, D) = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}.$$

Here, $\mathcal{L}$ usually refers to the test loss at the end of training (or a smoothed version of it). Using the training runs from Hoffmann et al. (2022), but with a more precise fitting procedure, Besiroglu et al. (2024) report the law

$$\mathcal{L}(N, D) = 1.8172 + \frac{482.01}{N^{0.3478}} + \frac{2085.43}{D^{0.3658}}. \tag{EpochAI}$$

The above scaling law does **not** take into account the batch size, which however plays a crucial role for training performance (Bergsma et al., 2025; Zhang et al., 2025). In particular, Hoffmann et al. (2022) report only that they used "well-tested heuristics" for the choice of batch size, but they do not study its effect on the scaling analysis.

Here, we propose to take into account how the token budget D **is allocated into** training steps K and batch size b. We first describe two similar approaches to do so.

Approach I: A three-term law. Following the power-law approach, we propose the form

$$\mathcal{L}(N, M, K) = E + \frac{A}{N^\alpha} + \frac{B}{M^\beta} + \frac{C}{K^\gamma}, \tag{3}$$

where $(E, A, B, C, \alpha, \beta, \gamma)$ are fittable parameters. This has the advantage that for optimal batch size choice it reduces automatically to the original Chinchilla form (1): to see this, minimize (3) with respect to M, subject to $D = MK$. The optimal batch size is

$$M^\star = \left[\frac{\beta B}{\gamma C}\right]^{\frac{1}{\beta+\gamma}} D^{\frac{\gamma}{\beta+\gamma}} =: G D^{\frac{\gamma}{\beta+\gamma}}. \tag{4}$$

In particular, the optimal batch size is independent of the model size N. Plugging back $M^\star$ and $K^\star = D/M^\star$ into (3) gives

$$\mathcal{L}(N, D) = E + \frac{A}{N^\alpha} + \frac{\hat{B}}{D^\tau}, \quad \tau := \frac{\beta\gamma}{\beta+\gamma}, \quad \hat{B} := BG^{-\beta} + CG^\gamma. \tag{5}$$

Hence, under (3), the optimal way to split a fixed token budget D into (K, b) recovers the form (1) by re-parameterizing $(B, \beta) \to (\hat{B}, \tau)$. Equation (3) is structurally very similar to convergence bounds from stochastic optimization. Let $(x_k)_{k\in\mathbb{N}}$ be the iterates of the stochastic conditional gradient method wrt. a general norm. Under suitable assumptions on the loss function (see also Islamov et al. (2026); Kovalev (2025)), for fixed batch size M (or b) and training steps K, under the optimal learning rate $\eta^\star$, Shulgin et al. (2026, Thm. 1) states that

$$\min_{1\le k\le K} \mathbb{E}[\mathcal{L}(x_k)] \lesssim \mathcal{L}_\star + \frac{1}{\sqrt{M}} + \frac{1}{\sqrt{K}},$$

where $\lesssim$ denotes that the bound holds up to some multiplicative constant for each term on the right-hand side. Comparing to (3), the optimal loss is parametrized by $E + \frac{A}{N^\alpha}$, and for the other terms the powers are relaxed from $1/2$ to (β, γ).

Approach II: Model-specific two-term laws. The functional form (3) implicitly assumes that the model size does not impact the coefficients (B, C) and (β, γ), and therefore by construction $M^\star$ is independent of N. While this implicit assumption is supported by experimental results on LLMs (Li et al., 2025a; Zhang et al., 2025), alternatively we can **fix the model size** and fit the form

$$\mathcal{L}(M, K) = E + \frac{B}{M^\beta} + \frac{C}{K^\gamma}, \tag{6}$$

where again (E, B, C, β, γ) are fittable parameters. We can fit this form (independently) for multiple N, a priori allowing for different values of (B, C, β, γ) across model sizes.

Terminology. From now on, we refer to laws of the form (6) (3) as *two-term law* and *three-term law* respectively. When convenient, we will use the abbreviations `2TL` and `3TL`.

4 Experiments

We fit scaling laws of the form `2TL` and `3TL` on two datasets which contain training logs for multiple combinations of (N, M, K) using `AdamW` (Loshchilov & Hutter, 2019), and which include a learning-rate sweep for each combination. We refer to the two datasets as **Li** and **OpenEuroLLM**; details are described in Section B.1 in the Appendix, in particular see Table 3. Here, we focus on the **Li** dataset; the results for **OpenEuroLLM** are deferred to the Appendix. For each (N, M, K), we choose the smallest final smoothened test loss value across learning rates. We form a validation set by collecting the datapoints from the largest token budget D for each individual N; the remaining datapoints are used as training set for fitting the law. We usually mark points from the validation set with a dashed border in our plots. See Section B.1 for additional details and Figs. 12 and 14 for an overview of the **Li** dataset.

Fitting procedure. If not explicitly mentioned otherwise, we use a standard fitting procedure minimizing the Huber loss with `L-BFGS-B` from multiple initializations, and five-fold cross-validation for each law. For the Huber loss we use $\delta = 10^{-3}$; for a detailed description and additional ablations on these choices, we refer to Section B.1 and references therein. All fitted parameter values can be found in Section B.3 in the Appendix. **Note that** whenever we report a single number for a parameter, it is the average over the five cross-validation runs.

4.1 Comparison of Approach I and II

A priori, it is not clear which of the two approaches described in Section 3 is preferable. Before doing any experimental comparison, we want to mention the main differences of `2TL` and `3TL` that arise from their definition. First, for `3TL` we will usually have more datapoints per (fittable) parameter, as we can use training runs across all model sizes N. For example, for the **Li** dataset, we have 246 samples to fit seven parameters for `3TL`, while for `2TL` we have 13-60 samples to fit five parameters. Consequently, we expect that the in-sample fitting error of `2TL` will be smaller. Second, based on the results of Li et al. (2025b), we would expect that `3TL` is more delicate/unstable to fit, given that it has two more fittable parameters.

We compare both approaches with respect to the following evaluations: (i) quality of fit, (ii) consistency of the scaling coefficients (β, γ) and (iii) consistency of the resulting optimal batch size $M^\star$ from (4) with previously reported results. We fit a `2TL` for each model size N (using only the respective subset of datapoints), and a single `3TL` using the union of all datapoints (across N).

For (i), we compute the mean absolute deviation (MAD) for the `3TL` as well as for the `2TL`, where each `2TL` is evaluated on the subset of datapoints respective to the model size. We then compute the average MAD across all `2TL`, weighted by the sample size, which we call the *two-term law ensemble*. This is done separately for training and validation set (see Fig. 10). For (ii) and (iii), we plot the estimated values of (β, γ) as well as the resulting formula for $M^\star$ from (4). We can directly compare the latter to the scaling law for optimal batch size reported by Li et al. (2025a), which was obtained from the same dataset and which we refer to as **Step-Law** (see Fig. 1).

Discussion: As expected, Fig. 10 shows that the in-sample fit of `2TL` is better; however, the out-of-sample error of `3TL` is slightly smaller. This indicates that despite having two additional parameters in `3TL`, our fitting procedure is sufficiently robust thanks to multiple initializations and bootstrap aggregation. Moreover, Fig. 1 shows that also in terms of consistency of the implied scaling of $M^\star$ the three-term law approach is preferable. In particular, its resulting scaling of $M^\star$ almost perfectly overlaps with the one by Li et al. (2025a).

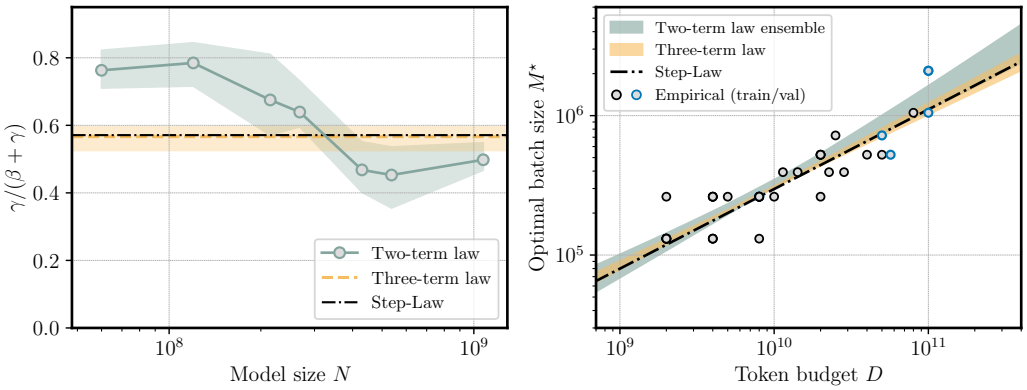


Figure 1: **(Left)** Estimates for $M^\star$-scaling coefficient $\frac{\gamma}{\beta+\gamma}$ for 3TL and each 2TL. Shaded area depicts min and max over five cross-validation fits. **(Right)** Implied scaling of $M^\star$ according to (4). Shaded area depicts min and max over cross-validation. Dots show the empirically best batch size from the train (*black*) and validation split (*blue*).

Takeaway: The three-term law approach achieves an overall slightly better out-of-sample fit than the two-term laws. Its implied scaling for the optimal batch size $M^\star \sim D$ is consistent with previous analyses (see Table 1).

Consistency across datasets. We run the same analysis on the **OpenEuroLLM** dataset, see Section B.4 in the Appendix. On **OpenEuroLLM**, both approaches again lead to a good fit. The scaling of the optimal batch size is relatively similar to the **Li** dataset, albeit slightly steeper. However, we also observe the caveat described below.

Caveat: The fitted parameter values for (E, A, α) are quite different between **Li** and **OpenEuroLLM**; in particular, the fit for **Li** is $E \approx 0$, indicating no irreducible loss which is in conflict to the non-zero entropy of language. This suggests that the effect of the model size is not perfectly reflected in 3TL.

We hypothesize that the reason for scaling inconsistencies between the two datasets is due to different training setups. For example, **OpenEuroLLM** uses different β_2 values in `Adam` for the larger models, which is known to have an impact on scaling laws (Porian et al., 2024). This might also explain that the 2TL has a better out-of-sample error than 3TL for **OpenEuroLLM**, as 2TL has more flexibility with respect to the impact of N. We also remark that imposing small regularization on $\log(E)$ can alleviate the above caveat (see Section B.2).

Table 1: Comparison of 3TL to batch size scaling laws from the literature.

Reference	Scaling	Comment
(Li et al., 2025a)	$M^\star = 0.58 \cdot D^{0.571}$	Referred to as Step-Law
(Bergsma et al., 2025)	$M^\star = (0.0306 \cdot s) \cdot D^{0.383}$	
(DeepSeek-AI et al., 2024)	$M^\star = 0.086 \cdot D^{0.688}$	From $M^\star \propto C^{0.327}$, $D^\star \propto C^{0.475}$
(3TL, **Li** dataset) **(ours)**	$M^\star = 0.667 \cdot D^{0.566}$	From (4)

4.2 Compute Savings Using the Three-term Law

Fitting a scaling law for $M^\star$ with the approach of Li et al. (2025a) imposes massive computational costs, as it requires to obtain the optimal batch size for a set of different token budgets D (and possibly also varying the model size N). Li et al. (2025a) report that producing their entire set of training runs consumed nearly one million NVIDIA H800 GPU hours. Recall that Li et al. (2025a) fit a law of form

$$M^\star = \frac{\tilde{A}}{D^{\tilde{\alpha}}} \tag{7}$$

directly on observations of $(M^\star, D)$. For a single observation $(M^\star, D)$, a full batch size sweep is needed to determine $M^\star$ (in the **Li** dataset, concretely we have 5-10 batch sizes per sweep). In contrast, our three-term law makes explicit use of observations from *suboptimal* batch sizes; we will show that this allows to obtain the same scaling law of $M^\star$ while saving a substantial amount of training runs/compute.

Setup: We mask the original dataset (containing the full batch size sweep), such that for each combination of (N, D) **only {one/two/three} batch sizes are randomly selected** (see Fig. 13 for an illustration). For the **Li** dataset, this shrinks the number of training runs required/available for the fit to {14/28/42} per-cent (see Table 2). We then fit the three-term law on this reduced dataset. As comparison, we fit (7) with $M^\star$ being the batch size (after applying the mask as described above) with the best loss, for each (N, D) separately.[1] We also use five-fold cross-validation to fit (7).

Discussion: We make two main observations: (i) Already with the full batch size sweep, (7) is unstable to removing the validation split. A direct fit on the full dataset (train and val) with our code gives $M^\star = 0.47 \cdot D^{0.584}$, essentially the same as Step-Law. Removing the validation split, we already get a quite different scaling of $M^\star = 6.29 \cdot D^{0.468}$. (ii) The three-term law results in almost identical scaling for $M^\star$, even when reducing the batch size sweep to two values, hence reducing the number of required training runs to 28%. Directly fitting (7) in contrast is highly unstable in this regime, and generalizes badly to higher (unseen) token budgets (see Fig. 2). When masking to only one batch size per sweep, the results of both approaches are very distinct to the original law.

Sweep size for b	`3TL`	Direct fit (7)	Samples/Training runs
Full	$M^\star = 0.67 \cdot D^{0.566}$	$M^\star = 6.29 \cdot D^{0.468}$	246
3 values	$M^\star = 0.48 \cdot D^{0.580}$	$M^\star = 8.59 \cdot D^{0.455}$	102
2 values	$M^\star = 0.84 \cdot D^{0.555}$	$M^\star = 2852.95 \cdot D^{0.210}$	68
1 value	$M^\star = 5.92 \cdot D^{0.475}$	$M^\star = 3.61 \cdot D^{0.514}$	34
Step-Law (Li et al., 2025a): $M^\star = 0.58 \cdot D^{0.571}$			

Table 2: Laws for $M^\star \sim D$ using `3TL` vs. a direct fit, for different masked versions of the **Li** dataset. When having only two or three runs/batch sizes per (N, D), `3TL` still results in essentially the same law, whereas the direct fit deviates. Note that the direct fit only uses the train split, therefore the difference to the Step-Law.

4.3 Performance with Suboptimal Batch Sizes

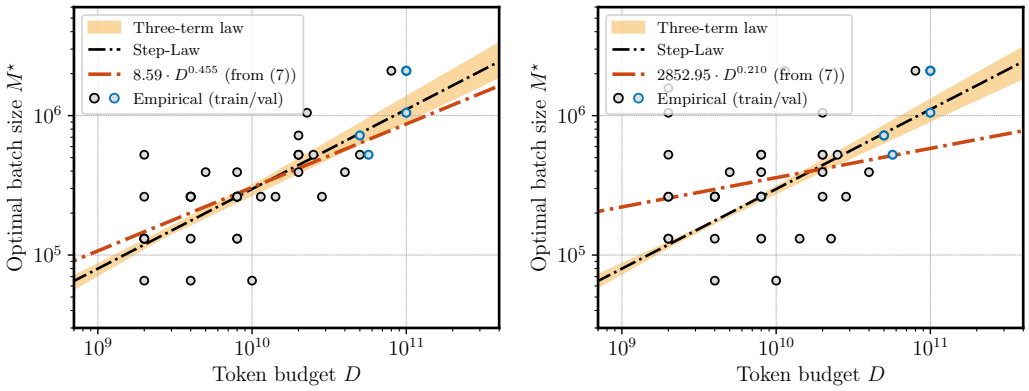
In practice, understanding how the *optimal* batch size scales with N and D might not be enough. In case of hardware constraints, it is mandatory to model the performance of models trained with *suboptimal* batch size. The three-term law form has the evident appeal that it also predicts model performance for suboptimal allocation of tokens into steps and batch size.[2] In short, the goal of this section is to answer the following question:

> *What is the interval of sub-optimal batch sizes $[b_{\min}, b_{\max}]$ such that at most 5% of compute is wasted, and how does it scale with D?*

Limitations of the three-term law. We first evaluate the fitted `3TL` on a range of batch sizes, and compare the predicted loss values to the true ones (see Fig. 11). While the optimal batch size is predicted well across all token budgets (Fig. 11, left), the three-term laws fails to accurately predict the loss value at the boundaries of the empirically covered range of D (Fig. 11, right). This is not surprising: note that for the three-term law, we fit 246 data points with seven parameters; therefore, we can not expect a perfect fit.

[1] This is similar to the procedure described by Li et al. (2025a). Alternatively, one could first fit a quadratic (or similar function) to the batch size sweep, and then read off $M^\star$ as the minimum. However, this is infeasible/unstable with ≤ 3 points available. We try deriving $M^\star$ from a quadratic fit when having 4 points per sweep, but this does not improve the result, rather the opposite (the resulting scaling is $0.02 \cdot D^{0.738}$).

[2] This is similar to how the Chinchilla Approach 3 has the advantage that it describes suboptimal allocation of compute into model size or token budget.



(a) Dataset reduced to 42%

(b) Dataset reduced to 28%

Figure 2: Fitting on a reduced dataset, with only 3 values of b per sweep **(left)** and 2 values **(right)**. Step-Law can be considered the oracle law here, as it was fit on the unreduced dataset (train plus val). In both cases, the implied scaling of $M^\star$ of the three-term law stays close to Step-Law, and generalizes better to large D than the direct fit (in *red*). The gray dots mark the empirically best batch size for each (N, D) on the reduced dataset (for the train split).

Fitting in two stages. To improve the fitting quality, we can fit a functional form $\mathcal{L} \sim b$ only on a subset of the data. We use the three-term law (3) as prior to choose such a functional form. Assume some fixed N and D and fixed sequence length s. Then, (3) simplifies to

$$\mathcal{L} = E + \frac{A}{N^\alpha} + (Bs^{-\beta})b^{-\beta} + (CD^{-\gamma}s^\gamma)b^\gamma.$$

Based on the fitted parameters of the `3TL`, we make the simplifying assumption $\gamma \approx \beta$ (we will see that this is sufficiently expressive to give an accurate fit). Based on this, we then fit the form

$$\mathcal{L}(b) = \tilde{E} + \tilde{A}b^{-\tilde{\alpha}} + \tilde{B}b^{\tilde{\alpha}}. \tag{8}$$

As a first stage, we fit $(\tilde{E}, \tilde{A}, \tilde{B}, \tilde{\alpha})$ from (8) separately for each (N, D).[3] Now, let us define the notion of ε-suboptimal batch size:

Definition 1. *Let $\varepsilon > 0$, and let $b^\star$ be the minimizer of* (8) *(for a fixed D and N). Since* (8) *is unimodal in b, we can define $[b_{\min}, b_{\max}]$ to be the interval of ε-suboptimal batch sizes such that $\mathcal{L}(b_{\min}) = \mathcal{L}(b_{\max}) = \mathcal{L}(b^\star) + \varepsilon$.*

Here, we set ε to the loss difference from the law (EpochAI) evaluated at (N, D) and $(N, 0.95 \cdot D)$, that is, we allow a 5% suboptimality in terms of compute. From (8), we can then easily read off, for each (N, D), the interval of ε-suboptimal batch sizes.

As a second stage, we fit a power-law $b_{\min\,/\,\max} = \Upsilon / D^\nu$, where (Υ, ν) are fitted. Here, for each model size N we keep the largest three token budgets as held-out validation set, and only use values of D where the empirically optimal batch size does not lie on the boundary of the sweep. This two-stage fitting procedure, where we reduced the number of parameters by assuming $\gamma \approx \beta$, has also been recommended in the survey of Li et al. (2025b).

From Fig. 3, we observe that this two-stage procedure – as expected – leads to a better fit across token budgets D. In particular, the fitted power-law on $b_{\min\,/\,\max}$ generalizes well beyond the token budgets used for fitting (Fig. 3, left). See Section B.7.4 for additional model sizes, and for the **OpenEuroLLM** dataset. When averaging the intervals of suboptimal batch sizes $[b_{\min}, b_{\max}]$ across model sizes (Fig. 17), we observe a slightly narrowing trend for the **Li** dataset, and a constant width for **OpenEuroLLM**; overall, the picture is very similar across the two datasets, suggesting that the scaling behavior of suboptimal batch sizes with D is relatively consistent.

[3]Here we perform a simple non-linear least squares fit using `scipy.optimize.curve_fit`.

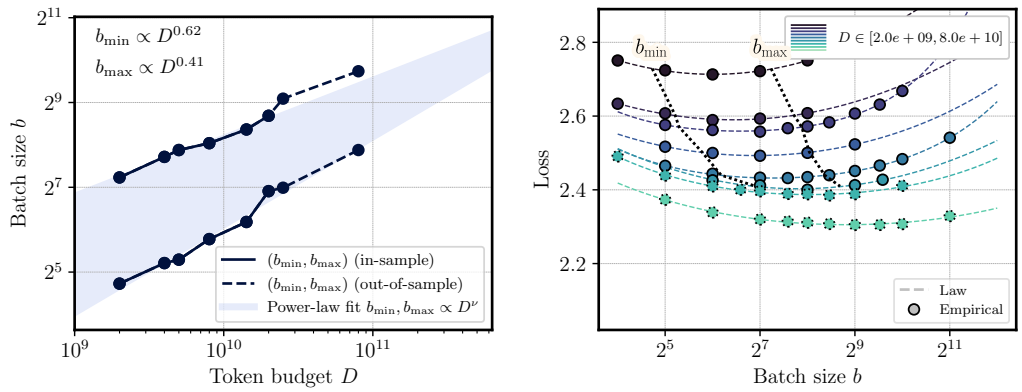


Figure 3: $N = 268$M. **(Left)** Batch size range $[b_{\min}, b_{\max}]$ with ε-suboptimal loss derived from law (8) (with ε such that less than 5% compute is wasted). Shaded area is obtained from fitting a power-law on the values of $b_{\min\,/\,\max}$ in-sample (solid lines). **(Right)** Empirical and predicted loss value across batch size b. Here, the predicted values are from the law (8), fitted separately for each D. Black dotted lines mark $b_{\min\,/\,\max}$ used for fitting the power-law $b_{\min\,/\,\max} \propto D^{\nu}$ on the left. Plots for other model sizes in Section B.7.4.

Takeaway: The interval of ε-suboptimal batch sizes can be modeled with a two-stage fitting procedure based from the three-term law; the scaling behavior generalizes well and is fairly consistent across model sizes and training setups. As rule of thumb, the interval of suboptimal batch sizes that corresponds to wasting at most 5% of compute has a width of 2^2 (Fig. 17).

4.4 Three-term Law and Critical Batch Size

The notion of *critical batch size* can be defined as follows: for a fixed target loss $\bar{\mathcal{L}}$, let $K_{\bar{\mathcal{L}}}(b)$ be the number of steps to reach loss $\bar{\mathcal{L}}$, as a function of the batch size b. As explained in the introduction, McCandlish et al. (2018) show that $K_{\bar{\mathcal{L}}}(b)$ decreases at much slower rate than inverse-linearly beyond a critical value of b, the so-called *critical batch size*. This has an important practical consequence: training at the highest practically feasible batch size can be suboptimal if it exceeds the critical batch size.

Empirically, Zhang et al. (2025) show that (i) critical batch size scales with compute under Chinchilla-optimal scaling of (N, D), and (ii) this increase comes mostly from scaling up the token budget D. In particular, when D is fixed, the function $K_{\bar{\mathcal{L}}}(b)$ is roughly the same across model sizes N.[4] Furthermore, Zhang et al. (2025) derive a theoretical model of critical batch size that captures the phenomena described above, however only for the very restricted setting of least-squares problems in the infinite-width limit.

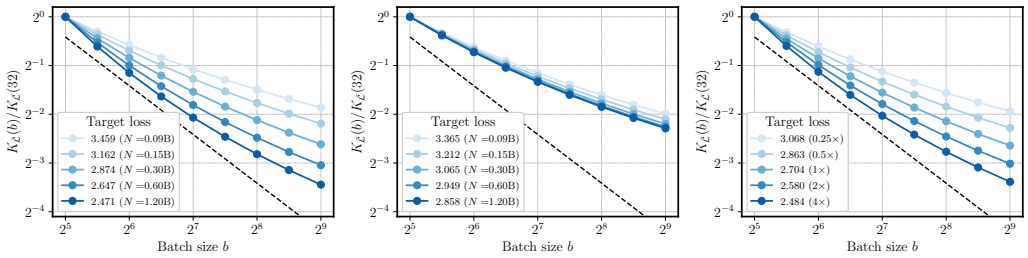
Here, we show that the three-term law (3) can equally well describe the behaviour of $K_{\bar{\mathcal{L}}}(b)$ when scaling D and/or N. For this, fix a target loss $\bar{\mathcal{L}}$ and denote $\tilde{E}(N) := E + \frac{A}{N^{\alpha}}$. From (3), if $\bar{\mathcal{L}} > \tilde{E}(N) + \frac{CB}{M^{\beta}}$, the number of steps to reach $\bar{\mathcal{L}}$ is given by

$$K_{\bar{\mathcal{L}}}(b) = \left[\frac{\bar{\mathcal{L}} - \tilde{E}(N)}{C} - \frac{B}{M^{\beta}}\right]^{-\frac{1}{\gamma}} = \left[\frac{\bar{\mathcal{L}} - \tilde{E}(N)}{C} - \frac{B}{Cs^{\beta}b^{\beta}}\right]^{-\frac{1}{\gamma}}. \tag{9}$$

We replicate the setting of Zhang et al. (2025, Figure 1), but plugging into 3TL: (a) Scale up both N and D in the Chinchilla-optimal setting. (b) Fix $D = 3.07$B and vary N from 85M to 1.2B. (c) Fix $N = 302$M and vary D in $[0.25, 4]$ times the Chinchilla-optimal D.

For all three settings, given (N, D) we compute the target loss $\bar{\mathcal{L}}$ from the law (EpochAI). Then, we compute $K_{\bar{\mathcal{L}}}(b)$ according to (9), using the parameters of 3TL previously fitted on the **Li** dataset. Fig. 4 confirms that the function $K_{\bar{\mathcal{L}}}(b)$ is almost invariant as we scale up model size N, but changes significantly if we scale up the token budget D.

[4]Note that Zhang et al. (2025) operate in a slightly non-standard setup, as they use constant learning-rate schedule with weight averaging (vs. cosine schedule used in Li et al. (2025a)).



(a) Chinchilla-optimal (b) Fixed data $D = 3.07$B (c) Fixed model $N = 302$M

Figure 4: Under the three-term law, critical batch size changes with token budget D **(right)**, but is almost invariant to changes of model size N **(middle)**. This matches empirical findings, cf. Zhang et al. (2025, Figure 1).

Takeaway: Under the three-term law, the number of steps $K_{\bar{\mathcal{L}}}(b)$ to reach a target loss $\bar{\mathcal{L}}$, as a function of the batch size, is mostly invariant to scaling up N, but not to scaling up D. This matches the empirical results of Zhang et al. (2025).

As a consequence, the three-term law is a suitable model for $K_{\bar{\mathcal{L}}}(b)$ at large batch sizes, while *at the same time* allowing for a non-trivial optimal batch size. This is in contrast to the theoretical models by McCandlish et al. (2018); Bergsma et al. (2025) and Zhang et al. (2025), which can describe the critical batch size, but imply that the optimal batch size is one.

5 Limitations

Here, we summarize the main limitations of the presented approach and identify future work:

- While we have shown that the three-term law can be robustly fit for two different datasets, the quantitative results can be inconsistent to a degree which is minor (e.g. for optimal batch size scaling) or moderately high (e.g. impact of model size). It is not clear how well the reported scaling laws generalize to other training setups or tasks. Further, although more sophisticated scaling law formulations can in principle collapse back to the Chinchilla form, the resulting scaling can be quite different (see Section B.5 for details).
- While the three-term law explicitly models the batch size, we still need the optimal learning rate for each single combination of (N, D, b); thus, despite our finding that the required amount of training runs can be reduced, the absolute number is still huge (before selecting the optimal learning rate, the **Li** dataset contains roughly 3000 runs). An interesting direction for future work would be to introduce the learning rate in the three-term law.
- The three-term law is not accurate enough to infer the interval of ε-suboptimal batch sizes. The two-stage procedure we propose instead still requires a relatively fine-grained batch size sweep.
- Optimal batch size scaling might be optimizer-dependent; in particular, it has been shown that the `Muon` optimizer (Jordan et al., 2024) allows for larger batch sizes (Essential AI et al., 2025). Investigating how switching the optimizer affects the fitted three-term law remains future work.

6 Conclusion

We have proposed a three-term scaling law that takes into account model size, training steps and batch size; the latter two explicitly model how the total amount of tokens is allocated. This formulation has natural advantages, bringing together Chinchilla-type and hyperparameter scaling laws, as well as tying it closely to theoretical results in stochastic optimization. On a practical side, we have shown that our proposed law can be robustly fit even with incomplete batch size sweeps, thus largely reducing the number of training runs necessary to obtain scaling laws for the optimal batch size. Our approach naturally allows to model suboptimal batch sizes, and we have derived their scaling with the total data budget. Finally, we have shown that the three-term law, in contrast to previous proposals, correctly describes the phenomenon of critical batch size, while at the same time allowing for non-trivial optimal batch size.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks: design or provide feedback on research methodology or experiments, edit a research paper to improve readability, create or edit uncritical parts of the software code.

We have not used generative AI tools for the following tasks: help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, implement the core methods for fitting, clean and reformat datasets, support qualitative and thematic data analysis, interpret results.

The following tasks with required disclosure are not applicable to this work: Generate synthetic data sets, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our codebase that contains all datasets and scripts to reproduce the experimental results will be made public in case of acceptance.

REFERENCES

Niccolò Ajroldi, Diana Alexandra Onutu, Haider Al-Tahan, Jörg Franke, Sampo Pyysalo, Jenia Jitsev, and Aaron Klein. Deriving scaling laws for openEuroLLM models: Learning rate, batch size and loss. August 2026. doi: 10.48550/ARXIV.2608.28308.

Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in LLM pre-training. In *Advances in Neural Information Processing Systems*, volume 38, pp. 125153–125188, 2025.

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

Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Vaughan, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. A foundation model for the earth system. 2024.

DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. DeepSeek LLM: Scaling open-source language models with longtermism. 2024.

Essential AI, Ishaan Shah, Anthony M. Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, Khoi Nguyen, Kurt Smith, Michael Callahan, Michael Pust, Mohit Parmar, Peter Rushton, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, Somanshu Singla, Tim Romanski, Yash Vanjani, and Ashish Vaswani. Practical efficiency of Muon for pretraining. 2025.

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, 2022.

Rustem Islamov, Roman Machacek, Aurelien Lucchi, Antonio Silveti-Falls, Eduard Gorbunov, and Volkan Cevher. On the role of batch size in stochastic conditional gradient methods. 2026.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. Blog post.

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. 2020.

Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization. 2025.

Houyi Li, Wenzheng Zheng, Jingcheng Hu, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i – optimal hyperparameter scaling law in large language model pretraining. 2025a.

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*, 2025b.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. 2018.

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, 2024.

Fabian Schaipp, Alexander Hägele, Adrien Taylor, Umut Simsekli, and Francis Bach. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training. In *International Conference on Machine Learning*, volume 267, pp. 53267–53294, 2025.

Christopher J. Shallue, Jaehoon Lee, Joseph M. Antognini, Jascha Narain Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *J. Mach. Learn. Res.*, 20:112:1–112:49, 2018.

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

Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. Scaling behavior of discrete diffusion language models. In *International Conference on Learning Representations*, 2026.

Xi Wang and Laurence Aitchison. How to set AdamW's weight decay as you scale model and dataset size. In *International Conference on Machine Learning*, volume 267, pp. 62222–62250, 2025.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12104–12113, 2022.

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean P. Foster, and Sham M. Kakade. How does critical batch size scale in pre-training? In *International Conference on Learning Representations*, 2025.

CONTENTS

1 Introduction **1**

2 Overview **2**

3 Scaling Laws with Training Steps and Batch Size **3**

4 Experiments **4**
4.1 Comparison of Approach I and II 4
4.2 Compute Savings Using the Three-term Law 5
4.3 Performance with Suboptimal Batch Sizes 6
4.4 Three-term Law and Critical Batch Size 8

5 Limitations **9**

6 Conclusion **9**

A Additional Related Work **14**

B Experiments: Supplementary Material **14**
B.1 Details on Experimental Setup 14
B.2 Additional Observations 15
B.3 Scaling Law Coefficients 15
B.4 Fitting 2TL and 3TL on **OpenEuroLLM** Dataset 16
B.5 Back to the Chinchilla Form 16
B.6 Ablation on Value of δ 17
B.7 Additional Plots 18
B.7.1 Miscellaneous 18
B.7.2 **Li** Dataset Overview 19
B.7.3 **OpenEuroLLM** Dataset Overview 21
B.7.4 Suboptimal Batch Size Scaling 22
B.7.5 Reduced Dataset Fit for **OpenEuroLLM** 24

A Additional Related Work

Comparison to other related models. We would like to mention a different approach by von Rütte et al. (2026, Section 4.5), which modifies the model of McCandlish et al. (2018) such that it also allows for optimal batch sizes larger than one. They propose the equation

$$\left(\left[\frac{K}{K_{\min}}\right]^{\alpha} - 1\right)\left(\left[\frac{b}{b_{\min}}\right]^{\alpha} - 1\right) = 1. \tag{10}$$

In comparison to the three-term law, the main conceptual difference is the setup of objective and constraints: von Rütte et al. (2026) fix a target loss $\bar{\mathcal{L}}$, and minimize $D = bsK$ such that $\bar{\mathcal{L}}$ is reached, subject to (10); for `3TL`, we fix D, and minimize the final loss with respect to b subject to $D = bsK$. Given that for a concrete training run it is much easier to fix D than a target loss, the latter seems to be the more practicable approach.

B Experiments: Supplementary Material

B.1 Details on Experimental Setup

Datasets. Below is a short description and source for the two main datasets we use in the analysis. Within each dataset, the sequence length is the same across runs (2048 for **Li**, and 4096 for **OpenEuroLLM**). For the scaling laws, we select for each combination of (N, b, D) (model size, batch size, token budget) the learning rate that obtains smallest final loss.

Li (Li et al., 2025a)

Training logs for different model sizes, token budgets, batch sizes, and learning rates. We use their smoothened loss and filter on dense models (no MoEs).

Source `https://wandb.ai/billzid/predictable-scale`

OpenEuroLLM (Ajroldi et al., 2026)

Dataset of training runs for dense English-language transformer models, executed by the OpenEuroLLM initiative.

Source `https://github.com/OpenEuroLLM/dense_english_scaling_laws`

Table 3: Overview of the datasets used for fitting our scaling laws. Here, we report the ranges of (N, M, K) on the union of train and validation set. See also Figs. 12 and 15.

Name	Li	OpenEuroLLM
Samples (train/val)	(246/54)	(191/21)
Range of N	$[59, 1073]$ $(\cdot 10^6)$	$[50, 1000]$ $(\cdot 10^6)$
Range of M	$[0.032, 4.2]$ $(\cdot 10^6)$	$[0.65, 4.19]$ $(\cdot 10^6)$
Range of K	$[0.9, 1525]$ $(\cdot 10^3)$	$[7.1, 1144.4]$ $(\cdot 10^3)$
Range of $\mathcal{L}$	$[2.12, 3.14]$	$[2.08, 2.93]$

Fitting methodology. We first describe the procedure for fitting the scaling laws. The training set is split into five parts of equal size. For each law, we then fit the same law on each leave-one-out split of datapoints (five-fold cross-validation). This allows for more robustness to outliers and to quantify uncertainty for each fitted parameter.

For the fitting procedure, we use the same Huber loss function as Besiroglu et al. (2024): for true loss values $\mathcal{L}_{\text{true}}$ and predicted loss values $\hat{\mathcal{L}}$, we minimize

$$\sum_i \mathcal{H}_\delta\big(\log(\mathcal{L}^{(i)}_{\text{true}}) - \log(\hat{\mathcal{L}}^{(i)})\big), \quad \mathcal{H}_\delta(z) = \begin{cases} \frac{1}{2}\delta^2, & |z| \le \delta, \\ \delta(|z| - \frac{1}{2}\delta), & \text{else.} \end{cases}$$

We use the `minimize` method from `scipy.optimize` together with the `L-BFGS-B` optimizer.

Choice of δ. For the Huber loss, we set $\delta = 10^{-3}$ for the `2TL` and `3TL` scaling laws, which is the standard value used also by Besiroglu et al. (2024). We run an ablation on this choice, setting $\delta = 10^{-5}$ instead, see Section B.6 for details.

Initialization. For each single fit, we minimize the Huber loss at n_{init} different initializations with `L-BFGS`, and select the solution that results in the smallest objective function. By default, we use a grid of ten values for each of $(\alpha, \beta, \gamma, E)$ and two values for each of (A, B, C). That is, for a `2TL` (see (6)) we have $n_{\text{init}} = 10^3 \cdot 2^2 = 4000$; for a `3TL` (see (3)) this becomes computationally intensive, and we therefore randomly select 5000 starting points from the grid.

Evaluation. After fitting, we predict loss values by averaging over the predictions of each of the five individually fit models (*cross-validation ensemble*). In Tables 4 and 5, we report the in-sample mean-absolute deviation (MAD) of the predicted and true loss values.

B.2 Additional Observations

This section lists changes of the fitting technique that we have tried (usually only on the **Li** dataset), which however do not lead to better or significantly different results.

(I) Due to the almost-zero values of the parameter E in our laws, we try to enforce larger values by adding the regularization term $\frac{\lambda}{2}(\log E)^2$. With $\lambda = 10^{-3}$, this leads to different coefficients, in particular $E = 0.729$; in terms of the other analyses, for example optimal batch size scaling, this has no major impact.

(II) Providing the gradient of (3) to the `L-BFGS` solver has no major impact on results.

(III) Using `BFGS` instead of `L-BFGS` has no major impact on results (while being significantly slower for fitting).

B.3 Scaling Law Coefficients

Table 4: **Li** dataset: Scaling law coefficients for `2TL` (see (6)) across model sizes, as well as for `3TL` (see (3)). `2TL` has no parameters (A, α), see (6). We report average value across subsampled fits, with the standard deviation in brackets. MAD refers to the in-sample mean absolute deviation of the predicted (with cross-validation ensemble) to the true loss values.

Model size N	E	A	B	C	α	β	γ	Samples	MAD
60M	0.705 (0.31)	-	4.56 (0.15)	7.26 (1.4)	-	0.0864 (0.018)	0.278 (0.029)	13	0.00802
120M	0.0395 (0.039)	-	5.24 (0.22)	8.22 (1.7)	-	0.0752 (0.0082)	0.274 (0.044)	13	0.0089
215M	0.481 (0.38)	-	4.66 (0.44)	5.01 (0.6)	-	0.0983 (0.033)	0.204 (0.028)	40	0.0123
268M	0.959 (0.17)	-	4.04 (0.3)	4.33 (0.23)	-	0.113 (0.019)	0.201 (0.011)	60	0.0143
429M	0.763 (0.22)	-	5.21 (0.73)	3.47 (0.25)	-	0.151 (0.027)	0.133 (0.018)	55	0.0115
537M	0.198 (0.1)	-	5.05 (0.55)	3.67 (0.1)	-	0.13 (0.02)	0.108 (0.015)	40	0.00962
1074M	$1.37e-07$ ($2.7e-07$)	-	5.63 (0.35)	4.3 (0.19)	-	0.126 (0.0087)	0.125 (0.008)	25	0.0111
Three-term	$1.08e-11$ ($2.2e-11$)	12.6 (0.59)	4.9 (0.2)	4.27 (0.15)	0.132 (0.0043)	0.139 (0.008)	0.182 (0.0074)	246	0.0159

Table 5: Same as Table 4, but for **OpenEuroLLM** dataset.

Model size N	E	A	B	C	α	β	γ	Samples	MAD
50M	2.3 (0.048)	-	2.26 (0.42)	2.27 (0.38)	-	0.161 (0.026)	0.2 (0.026)	43	0.00654
130M	1.79 (0.11)	-	3.45 (0.76)	2.92 (0.7)	-	0.164 (0.028)	0.184 (0.04)	39	0.00798
300M	1.16 (0.13)	-	3.07 (0.21)	2.51 (0.056)	-	0.113 (0.018)	0.138 (0.007)	38	0.00656
600M	1.09 (0.021)	-	4.41 (0.36)	2.58 (0.057)	-	0.159 (0.011)	0.122 (0.0085)	36	0.00562
1000M	1.02 (0.12)	-	5.17 (2)	2.8 (0.16)	-	0.169 (0.042)	0.128 (0.012)	35	0.00718
Three-term	0.264 (0.23)	180 (24)	2.62 (0.059)	2.73 (0.16)	0.292 (0.0077)	0.0705 (0.017)	0.156 (0.011)	191	0.0128

B.4 Fitting 2TL and 3TL on OpenEuroLLM Dataset

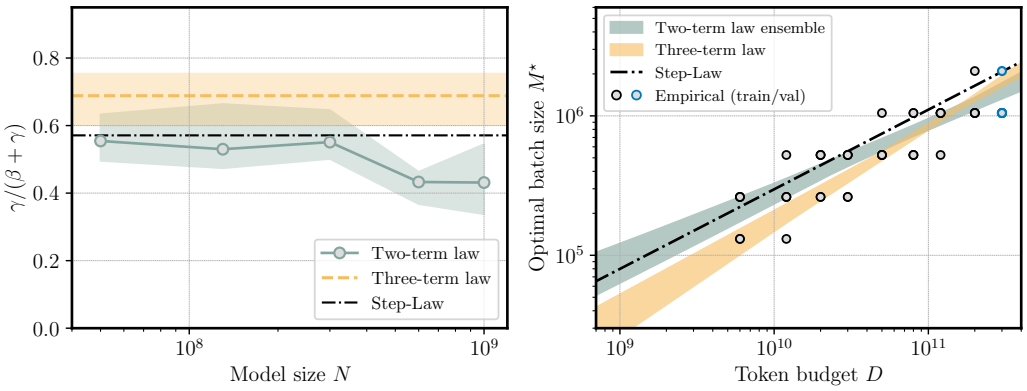


Figure 5: Same as Fig. 1, but for **OpenEuroLLM** dataset.

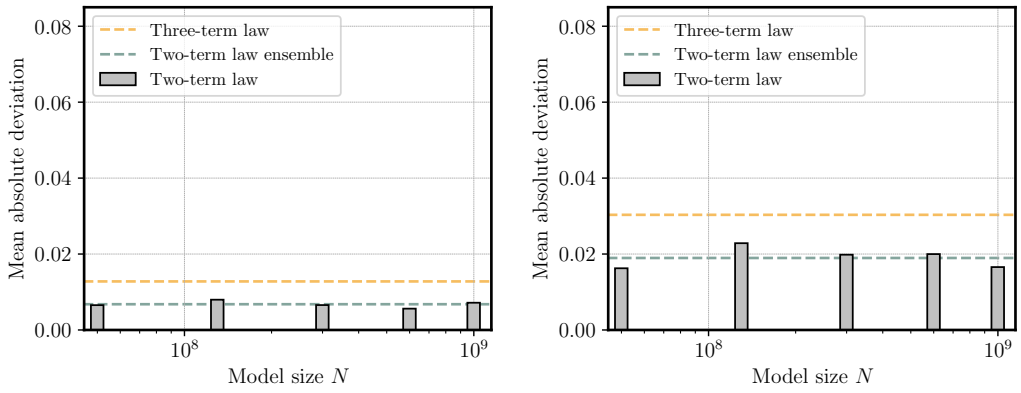


Figure 6: Same as Fig. 10, but for **OpenEuroLLM** dataset. **(Left)** Training set, **(right)** validation set.

B.5 Back to the Chinchilla Form

Having fitted the 3TL, we can compare its batch-size-optimal reduction from (5) to a Chinchilla-type law. To do so, we first fit the form (1) to the runs from Li et al. (2025a) (only using optimal batch sizes); we use the same fitting procedure except that we set $\delta = 10^{-5}$, which leads to a more stable fit. See Section B.1 in the Appendix for more details and ablations of this choice.

Comparing this to the fitted parameters of the 3TL (see Table 6), we can already see a rather big discrepancy; for example, with 3TL we obtain a much smaller value of τ (compared to β) as well as $E \approx 0$. Instead of simply comparing the parameter values, we can also compare the implied scaling behavior of both laws. In particular, the main goal of Chinchilla scaling laws is to determine how the optimal model size scales with compute $\mathcal{C}$. Assuming $\mathcal{C} = 6ND$, from (1) we get $N^\star = \left[\frac{\alpha A}{\beta B}\right]^{\frac{1}{\alpha+\beta}} \left(\frac{\mathcal{C}}{6}\right)^{\frac{\beta}{\alpha+\beta}}$. For the three-term law, using (5), we replace $(\beta, B) \to (\tau, \hat{B})$. Fig. 7 shows that the implied compute-optimal scaling of $N^\star$ overlaps only for a relatively small interval of compute $\mathcal{C}$. This also confirms the finding of Li et al. (2025b), that the exact formulation of the scaling law can already impact the implied optimal model size.

Table 6: Fitted parameter values when fitting a law of form (1) to **Li** dataset, and for 3TL after the reduction (5).

Parameter	Law (1)	3TL
E	1.26	0.0
A	23.82	12.58
$B \mid \hat{B}$	4885.3	9.16
α	0.166	0.132
$\beta \mid \tau$	0.336	0.079

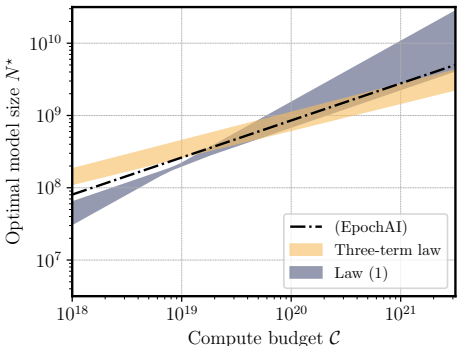


Figure 7: Compute-optimal model size.

B.6 Ablation on Value of δ

When fitting the standard Chinchilla form (1) to the **Li** dataset with $\delta = 10^{-3}$, we observe that the variance in the parameter estimates appears to be unbalanced; this is likely due to the rather small range of N in the dataset, which has been reported to cause issues for fitting scaling laws (Li et al., 2025b). We find that using $\delta = 10^{-5}$ fixes this. Hence, we perform an ablation with $\delta = 10^{-5}$ for the two-term and three-term scaling laws, to check whether the value of δ impacts also the fit of those laws.

Below we show the results of Section 4.1, but using $\delta = 10^{-5}$ instead of $\delta = 10^{-3}$. In short, for 3TL we observe almost identical results, albeit with slightly higher variance for the scaling of $M^\star$. For 2TL, we observe a slightly worse MAD when using $\delta = 10^{-5}$, as well as higher variance for the coefficients (γ, β) as well as the scaling of $M^\star$.

Overall, the choice of δ does not have big impact on the conclusion of Section 4.1, with $\delta = 10^{-3}$ being slightly preferable.

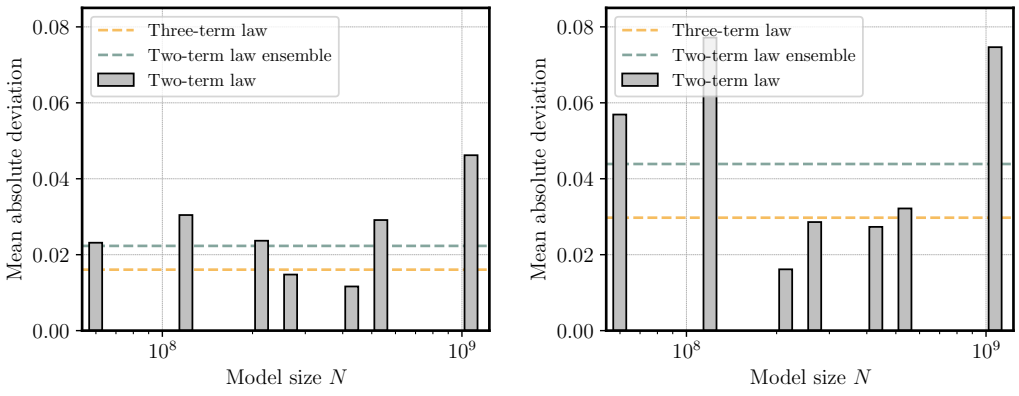


Figure 8: Same as Fig. 10, but with $\delta = 10^{-5}$.

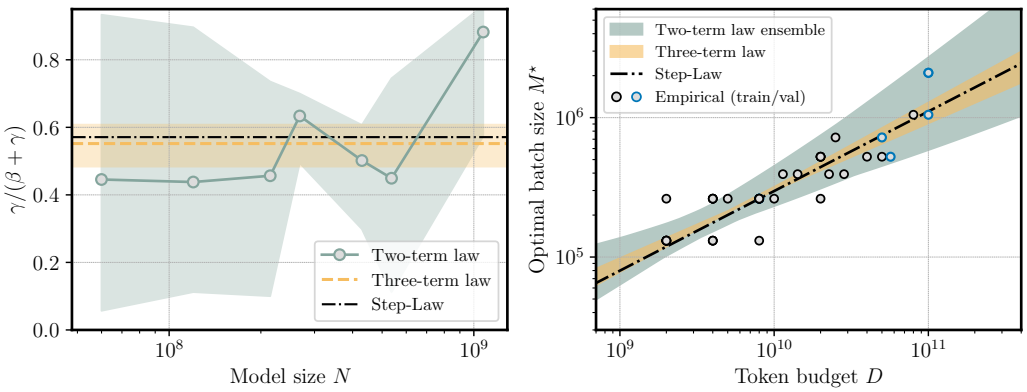


Figure 9: Same as Fig. 1, but with $\delta = 10^{-5}$.

B.7 Additional Plots

B.7.1 Miscellaneous

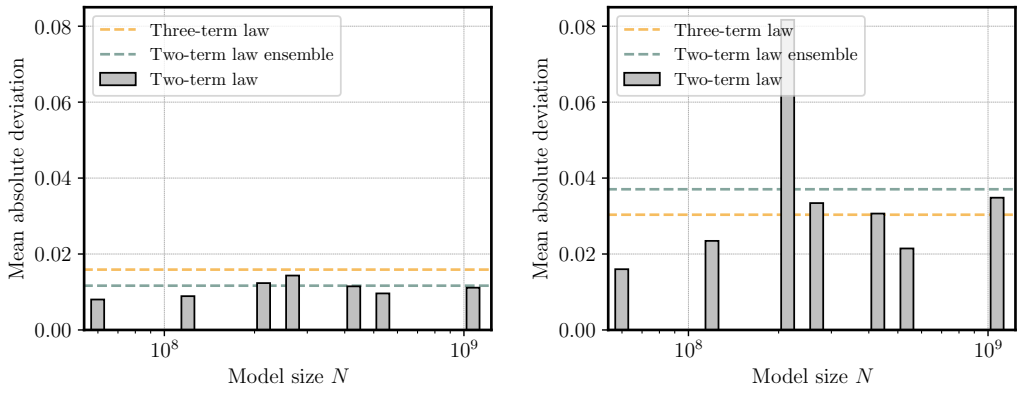


Figure 10: MAD comparison of 2TL and 3TL on train **(left)** and validation **(right)** split.

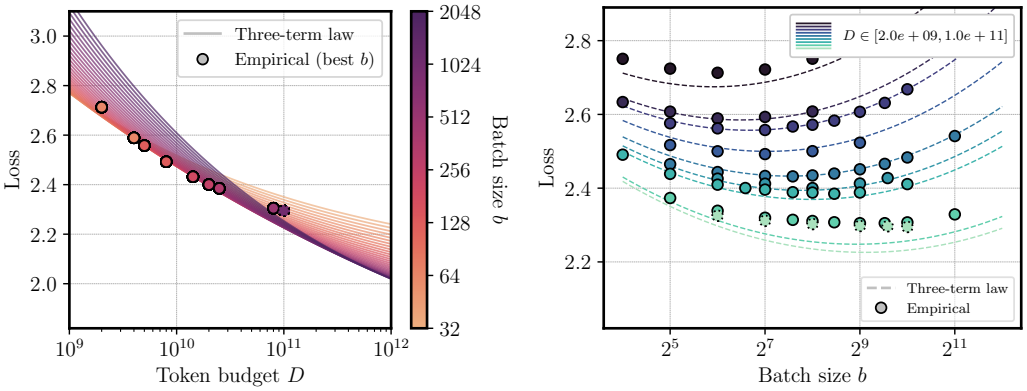


Figure 11: $N = 268$M. **(Left)** While the three-term law (3) accurately predicts optimal batch size, its predicted *loss value* for very large/small token budgets deviates from the empirical value. **(Right)** Empirical and predicted loss value across batch size b. Again, for very large/small token budgets the accuracy of the three-term law degrades. Dashed border marks datapoints not used for fitting 3TL.

B.7.2 **Li** DATASET OVERVIEW

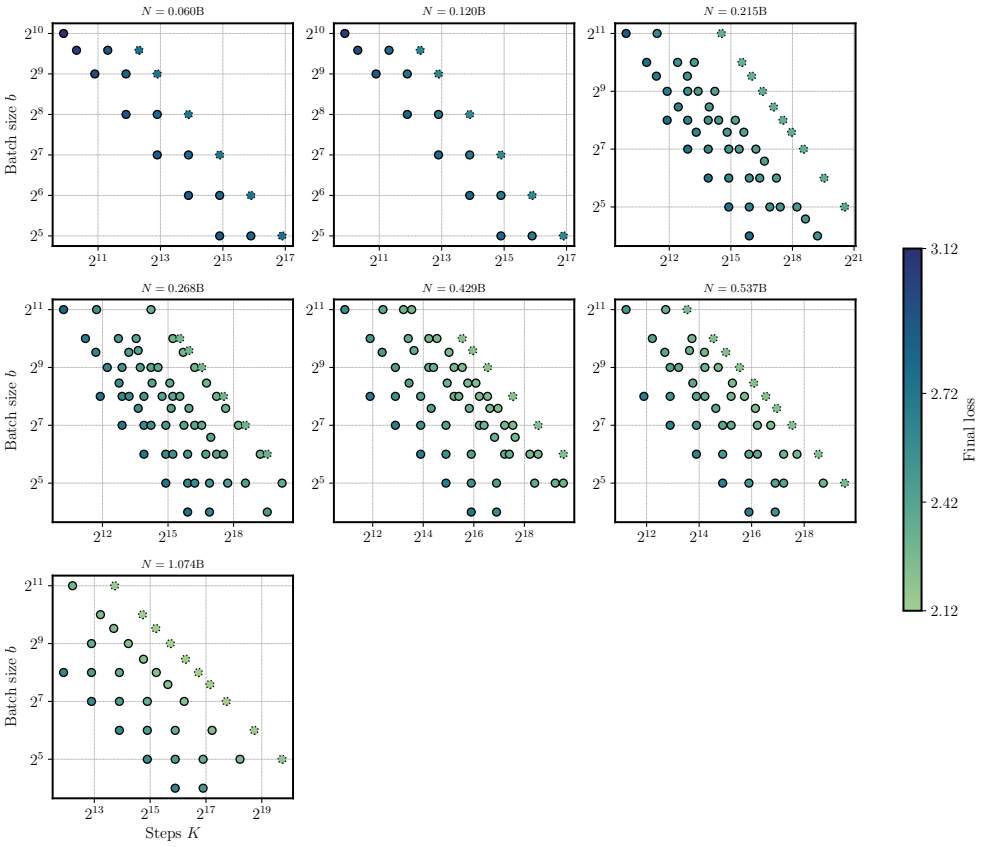


Figure 12: Overview of the **Li** dataset used for fitting scaling laws. Dots with dashed border are part of the validation set.

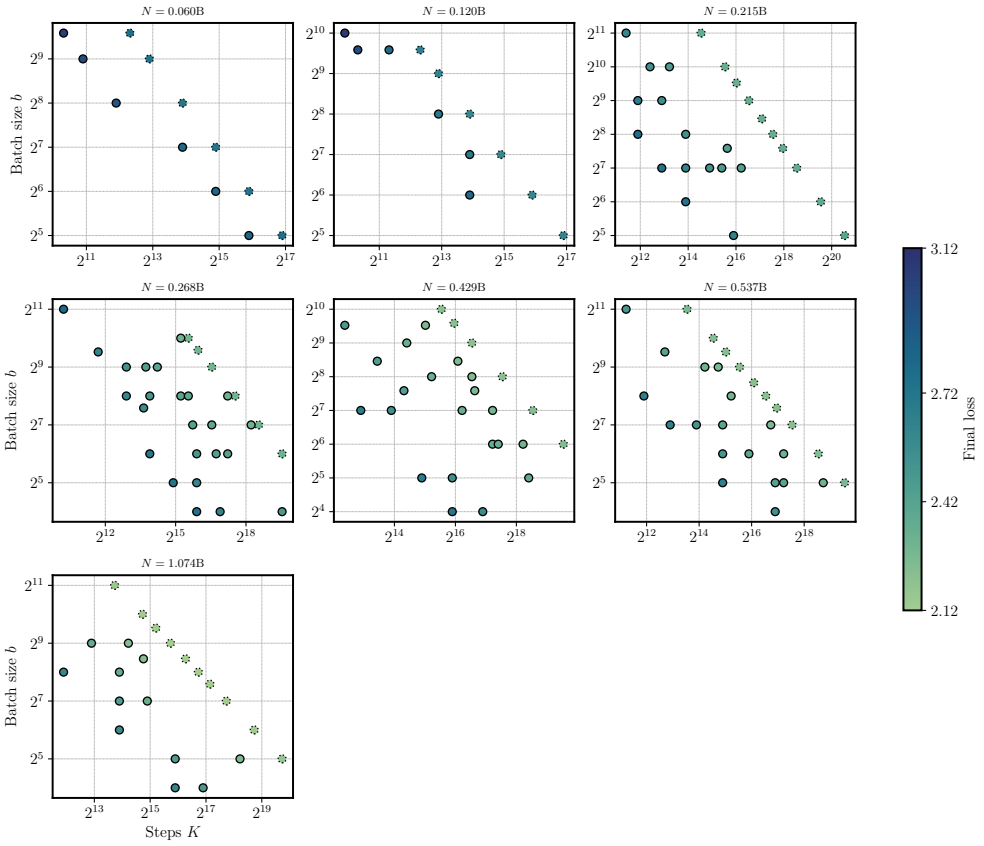


Figure 13: Illustration of the reduced dataset used in Section 4.2, with 3 batch size values b per sweep. See also caption of Fig. 12; here, the difference is that for each value of (N, D) we use only three different batch sizes (lying on a diagonal) for fitting, while the validation set remains the same as before.



Figure 14: Overview of the full **Li** dataset (before learning-rate selection). Each heatmap represents the final loss over a grid of batch size b (y-axis) and learning rate η (x-axis) for a single combination of (N, D). Blue squares mark the optimal combination of (η, b), gray squares mark optimal learning rate for the given row of batch size. Note that most marked squares do not lie on the border, therefore indicating that the sweep is sufficiently extensive.

B.7.3 OPENEUROLLM DATASET OVERVIEW

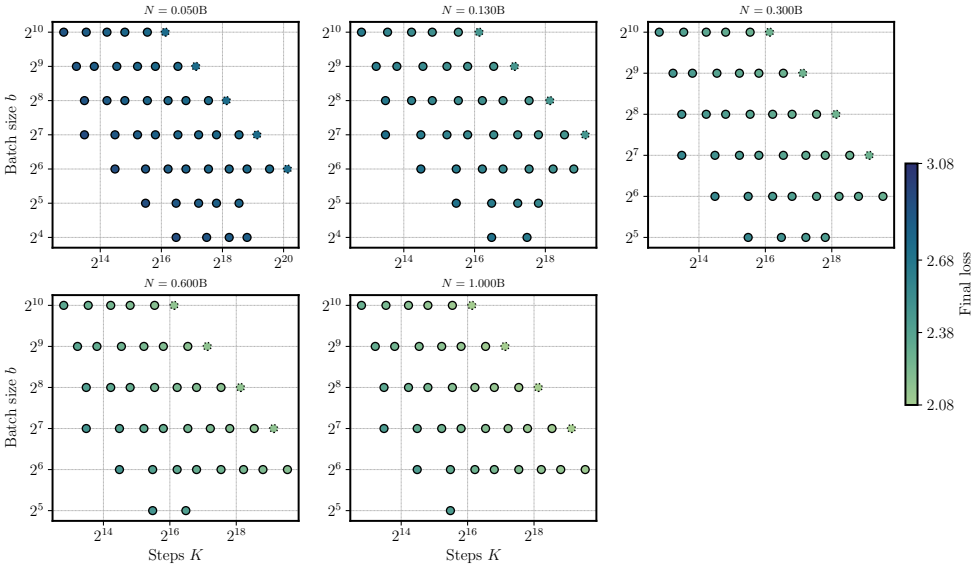


Figure 15: Same as Fig. 12, but for the **OpenEuroLLM** dataset.

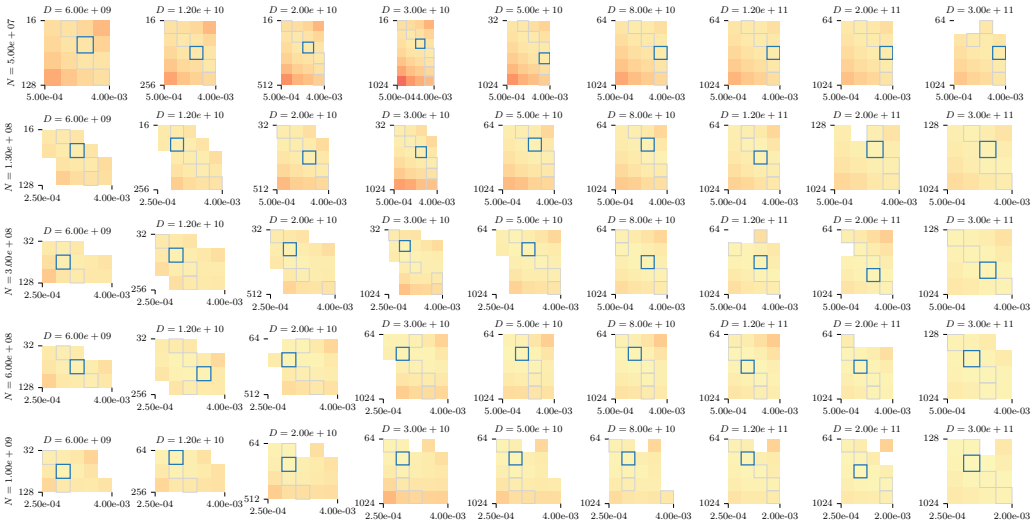
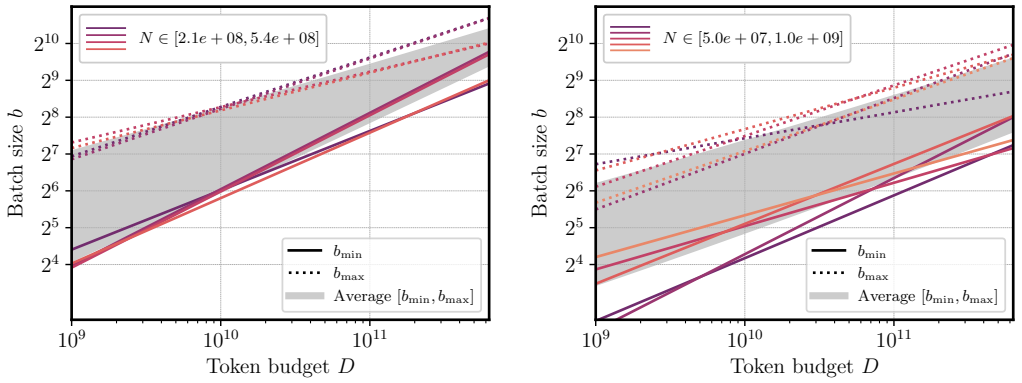


Figure 16: Same as Fig. 14, but for the **OpenEuroLLM** dataset. For each heatmap, x-axis marks learning rate and y-axis marks batch size.

B.7.4 Suboptimal Batch Size Scaling



(a) **Li** (sequence length 2048) (b) **OpenEuroLLM** (sequence length 4096)

Figure 17: Scaling of ε-suboptimal batch size across model sizes, for **Li** dataset **(left)** and **OpenEuroLLM** dataset **(right)**. The scaling of suboptimal batch sizes $[b_{\min}, b_{\max}]$ (grey area) is relatively consistent across the two datasets, after accounting for a factor of two due to the different sequence length.

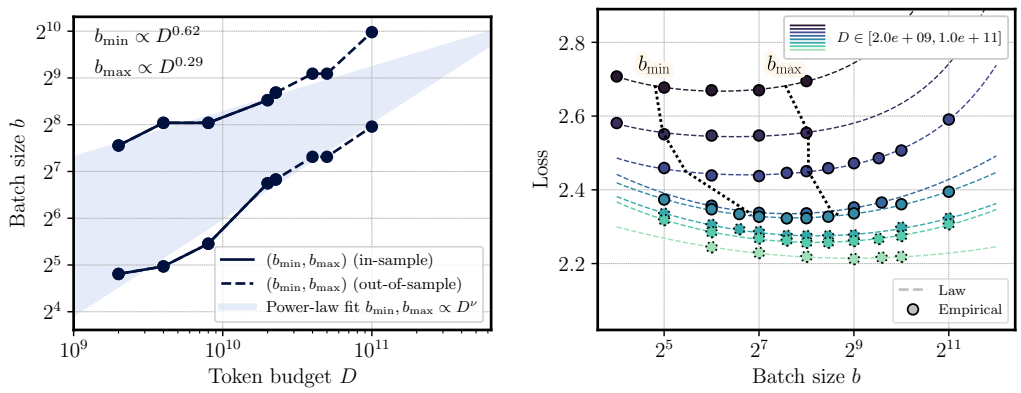


Figure 18: Same as Fig. 3, but for $N = 429$M.

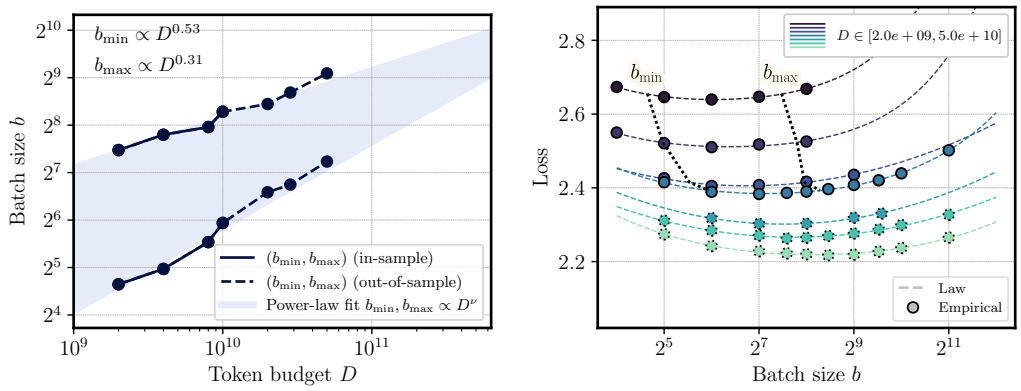


Figure 19: Same as Fig. 3, but for $N = 537$M.

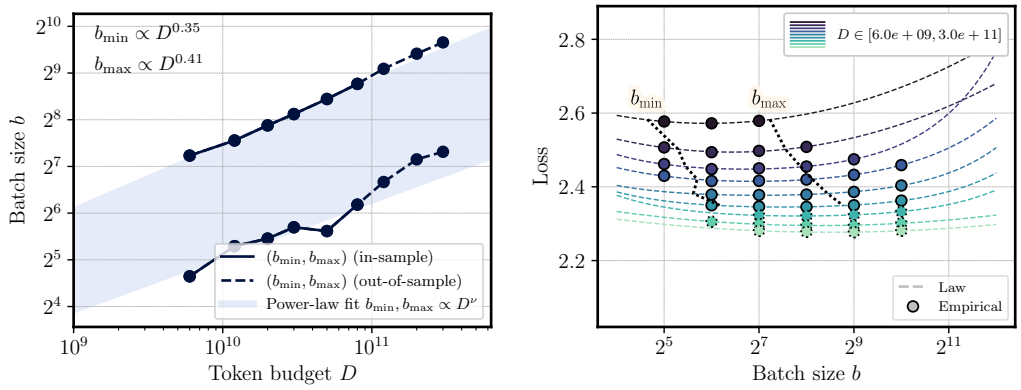


Figure 20: Same as Fig. 3, but for **OpenEuroLLM** dataset with $N = 300$M.

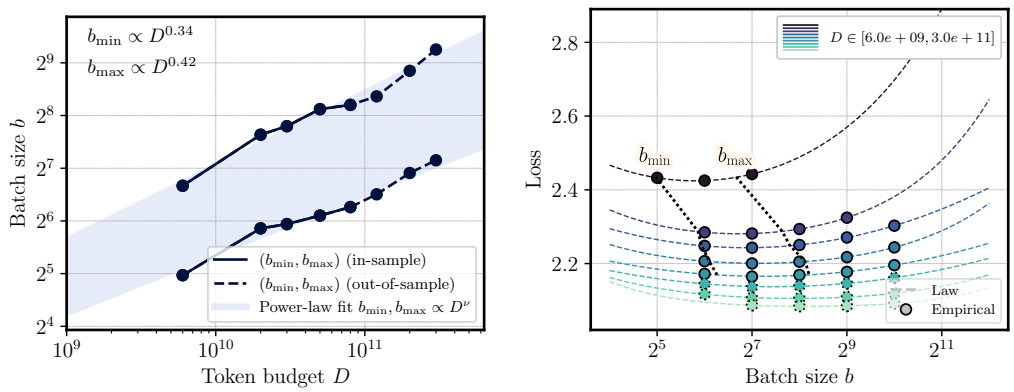
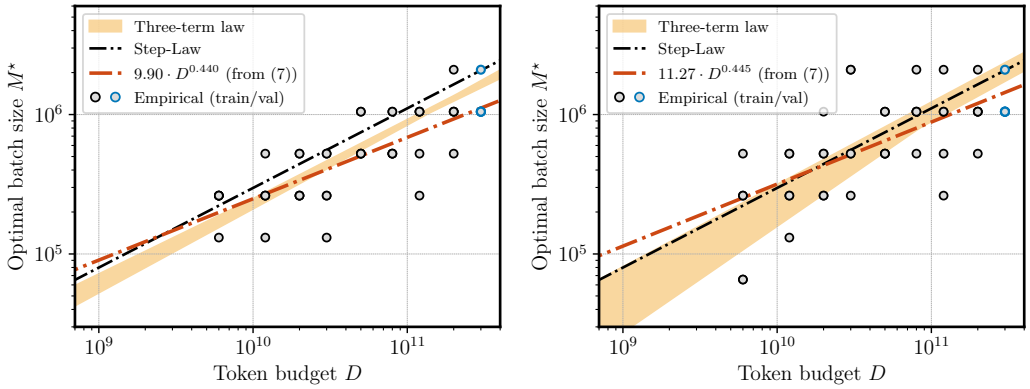


Figure 21: Same as Fig. 3, but for **OpenEuroLLM** dataset with $N = 1$B.

B.7.5 Reduced Dataset Fit for OpenEuroLLM



(a) Dataset reduced to 63%

(b) Dataset reduced to 42%

Figure 22: Same as Fig. 2, but for **OpenEuroLLM** dataset. Fitting on a reduced dataset, with only 3 values of b per sweep **(left)** and 2 values **(right)**.