SMALL-SCALE EXPERIMENTS: ARE WE THERE YET?

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling laws promised cost-effective experiments; six years later, they have yet to fully deliver. Instead, researchers have found them unreliable at small scales, starting at 4M parameters, and concluded that sizable models cannot be avoided. We show this is not the case and that the confounding factor is hyperparameter tuning. In fact, small models are highly sensitive, but hyperparameter sensitivity fades with scale. This small-scale sensitivity makes scaling laws easy to miss because they only emerge on the fully tuned frontier, and reaching that frontier requires an extensive search far beyond what most ever run. By ablating the basic scaling law recipe, we show well-tuned hyperparameters matter more than any other ingredient. Further, we show that as scale increases, the hyperparameter loss surface becomes lower dimensional, making it easier to find those hyperparameters. Nevertheless while scaling laws exist in small models, extrapolation hits statistical limitations and a holistic approach becomes necessary. We apply our insights to a question that once took the field years to settle: where to place normalization layers in the transformer architecture. Using a rigorous experimental setup inspired by our findings, we find that pre-normalization works better as models grow in size. With the right tools and a better understanding, small-scale experiments can deliver on scaling laws' long-awaited promise.

1 INTRODUCTION

Foundation models have revolutionized the way we think about computing. They arose as a paradigm only five years ago [7], yet in that time have become the basis for new discoveries across a broad set of scientific fields [40; 42; 53; 75; 36]. As you might expect, those five years saw a flurry of research into how best to build these models, and one key ingredient emerged: scale. Unfortunately, scale makes experiments incredibly expensive.

The natural solution is to experiment at the small scale, then transfer to the large scales. Despite the early promise of scaling laws [41], this solution has not yet materialized after half a decade. Instead, pretraining research grew exponentially in cost, with experiments (not the final training run) dominating the overall expense of model development [16; 19].

Scaling laws failed to deliver because they hit a major roadblock, which is that they are notoriously elusive at small scales. Famously, the two most influential works in the literature reached contradictory conclusions. Kaplan et al. [41] held parameters should scale much faster than data, as a result early language models were under-trained [9; 66; 12]; Hoffmann et al. [33] found parameters and data should scale at the same rate. Porian et al. [65] traced this discrepancy to methodological choices such as whether to count embeddings in the parameters, how long to warm up the learning rate, and scale-dependent hyperparameter tuning; choices that disproportionately affect small-scale models [62]. This early discord foreshadowed later difficulty. Surveying the literature, Li et al. [45] found many scaling laws hard to reproduce and, in particular, that scaling laws based on smaller models with 100M parameters or less are unreliable. Li et al. [43] aptly summarize the situation as the "*scaling gap* where insights from small-scale experiments often fail to transfer to resource-intensive production systems".

How do we close this gap and deliver on scaling laws' original promise? We find that the recipe for a scaling law has one key ingredient that, more than any other, determines final quality: rigorous hyperparameter tuning (§3). By ablating different choices, we show scaling laws are fairly robust to details like how to count parameters or whether to decay the learning rate (§3.1). However, poorly

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.

tuned hyperparameters can be disastrous (§3.2). Whereas common practice tunes over a small grid, you may need to search hundreds of configurations to reveal the scaling law in small models. Such a search sounds expensive, yet we also find scaling laws extend remarkably far down; as little as 4M parameters where models train in under an hour on a single GPU, bringing the cost well within reach for many academic and industry labs.

Hyperparameters unduly affect small-scale laws because small-scale models are unduly difficult to tune. We explain this difficulty in terms of the hyperparameter loss surface (§4). As parameters and data grow together, they drive good configurations to fill a larger fraction of the space and the intrinsic dimension of the surface goes down (§4.1—§4.2). Moreveover, as model size increases, the *effective* number of hyperparameters drops to one. This phenomenon introduces a new way of thinking about scaling, where one should do a thorough job sweeping hyperparameters in the smaller scales to due to their sensitivity whereas larger models are easier to tune (§4.3). Luckily smaller scale experiments are significantly cheaper to run than their larger scale counterparts so proper scaling becomes a matter of proper computational budget allocation.

Building on our insights, we propose a new methodology for proper scaling drawing on three tools from the literature (§5). Namely, the *noisy quadratic limit* reveals if hyperparameters are fully tuned and how hard that is to accomplish [48]; *scaling laws* describe how loss changes as models scale up [67; 41; 33]; and the *perplexity–capability correspondence* answers whether conclusions about pretraining loss have consequences for the capabilities we ultimately care about [51].[1] Small-scale experiments exploit strong assumptions and we argue for an approach where each assumption becomes a diagnostic that improves our qualitative understanding of how the model scales (5.1). This qualitative understanding is important because even though phenomena like scaling laws exist at the small scale, quantitative techniques such as extrapolating the loss compound small errors (5.2). Finally, we apply our takeaways to a question that once took the field years to settle (5.3): where normalization layers should go in the transformer architecture and find that larger scales favor pre-normalization [76; 2; 83].

In the end, these ideas give a practical methodology for model-centric research:[2] build a complete understanding at the small scale by thoroughly exploring the model's hyperparameters, then use simple rules to carry it up. This regimen offers the rare opportunity to cut costs and raise rigor, but it takes the right tools. With them, small-scale experiments can finally deliver on their long-awaited promise.

2 Background and related work

2.1 Scaling Laws

Researchers have long sought to predict the outcomes of large-scale training from small-scale experiments. Early work studied how models improve with more data [15; 31; 74]; later work, with the arrival of web-scale data, turned towards parameters and compute as scaling laws became the primary tool for managing the rising cost of experiments [67; 41]. After Kaplan et al. [41] demonstrated that pretraining loss falls predictably over many orders of magnitude for language models, similar scaling laws were soon discovered in many other domains [30; 39; 11]. Such scaling laws offered an enticing promise: study a model cheaply at the small scale, then extrapolate to the large.

Classically, scaling laws relate parameters (p) and data (d) to pretraining loss ($\mathcal{L}$). Hoffmann et al. [33] popularized the most widely used functional form [67]:

$$\mathcal{L}(p, d) = \epsilon + \frac{\zeta}{p^{\iota}} + \frac{\eta}{d^{\kappa}} \tag{1}$$

During training the network runs on every example, so compute is proportional to parameters times data: $c \propto pd$. With Equation 1, this constaint gives the compute-optimal loss as a power law: $\mathcal{L}(p, d) - \epsilon \propto c^{-\frac{\iota\kappa}{\iota+\kappa}}$ given a fixed compute budget, see Appendix A.1 for details.

[1] Even when we often can not predict these capabilities in advance [49].

[2] *Model-centric* research improves the model holding data fixed; *data-centric* improves the data holding the model fixed. Our discussion focuses on model-centric research while data-centric research likely requires different techniques (§6).

However, the promise of predictable large scale performance for fixed compute budget proved difficult to keep. Pearce & Song [62] found small-scale experiments led to discrepancies across major scaling law studies [41; 33]. More recently, Li et al. [45] surveyed the literature, found scaling law estimates hard to reproduce, and reported those built from small models are unreliable. We revisit this regime and reach a more optimistic conclusion: with rigorous tuning, scaling laws exist even at a few million parameters though we must extrapolate them with care.

2.2 Hyperparameter Optimality

With the evidence presented in the previous section, we might develop the intuition that tuning hyperparamerters is one of the most important problems in estimating accurate scaling laws. If that is true, how do we know if hyperparameters are fully tuned? Lourie et al. [48] found that near the optima, hyperparameter-loss surfaces can be well approximated with a simple low rank model; the noisy quadratic. Empirically, this approximation holds for a large region around the optimum. Since this region determines the practical questions we care about, like tuning difficulty or the best attainable performance, we will use the noisy quadratic throughout this study. For more details see Section A.2.

After drawing conclusions at the small scale, we still need the right hyperparameters to scale up. One strategy parametrizes them so that their optimal values stay *fixed* across scales [84; 85; 21; 55]. A complementary approach fits scaling laws for the optimal values themselves [18; 44; 4]. Many methods work reasonably well. We will later see that is is likely because as models grow, they become less sensitive to their hyperparameters thus a configuration tuned cheaply at the small scale transfers well with just a little adjustment — a central empirical finding in this study we will detail later.

2.3 Perplexity–Capability Correspondence

Scaling laws' successes in pretraining inspired many to seek them for downstream tasks [37; 23; 10; 50; 60]. This stream of work delivered mixed results however as downstream laws inherit and compound the tuning difficulties of upstream ones, spread across multiple training stages and a battery of evaluations. As a consequence, downstream tasks exhibit a wide range of unpredictable behavior, such as emergent [79; 72], inverse [52; 81], or even U-shaped scaling [80]. Lourie et al. [49] found as little as 39% of tasks scale reliably. Carefully chosen metrics can mitigate some of this irregularity [68; 69] but not eliminate it [22; 87].

To sidestep this issue, researchers recast downstream performance as a function of pretraining loss instead of data and compute [82; 71; 35; 22; 23; 10]. Remarkably, Mayilvahanan et al. [51] showed pretraining loss corresponds to capability as long as the pretraining data is held fixed across changes in scale, hyperparameters, architecture, and even the tokenizer. In other words: models of the same perplexity obtain the same capabilities, even when those capabilities are hard to know in advance. We term this phenomenon *perplexity–capability correspondence*. Hence in this study we will rely on and assume this correspondence; a practical illustration is in Appendix A.3.

3 Estimating (Small-Scale) Scaling Laws

In this study, we will use models ranging in size from 4–34M to estimate scaling laws, 67M–134M for validation, and 268M for testing. All details about the experimental design chosen in this study are in Appendix B.

What does it take to estimate scaling laws from small-scale experiments? We find them fairly robust to different decisions (§3.1), but one ingredient makes or breaks the resulting law: hyperparameter tuning (§3.2).

3.1 The Impact of Methodology

Despite the frequent debate over parameter definitions, we find they make little difference. Beyond that, we study three refinements on a basic approach: tuning hyperparameters per token budget,

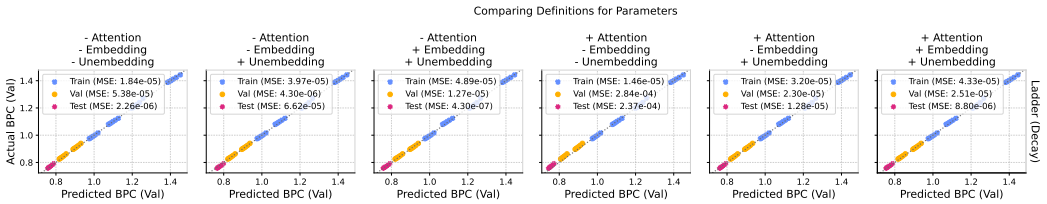


Figure 1: *Even at small scales, how you count parameters has limited impact on the scaling law relative to other factors.* Researchers count parameters in different ways, diverging on whether to (1) account for attention's FLOPs (`Attention`), (2) include the embedding (`Embedding`), and (3) include the unembedding (`Unembedding`). The embedding and unembedding have the same number of parameters, giving six distinct counts. Each one yields a reasonable scaling law.

decaying the learning rate, and tying the scaling exponents. Some help but none, we will see, is what makes the estimate work.

Parameter definitions only make a small difference. When it comes to scaling laws, researchers define parameters many different ways [41; 33; 18]. These definitions differ along three axes: whether to account for attention, the embedding, and the unembedding [65, §B]. The disagreement pivots on the importance of parameters vs. compute: attention uses more FLOPs on longer contexts, while the embedding layer uses no FLOPs at all. Figure 1 reveals these differences in philosophy lead to little difference in outcome. We use the effective parameters (`Attention` and `Unembedding`, no `Embedding`) as it corresponds most directly to FLOPs per token, keeping $c = 6pd$ exact.

Other choices refine the estimate, rather than enable it. Figure 2 compares increasing refinements on a basic approach. Fitting the best run at each scale without learning rate decay, the scaling law is immediately recognizable (if imprecise). Improving the methodology refines its precision: tuning hyperparameters for each parameter–data budget reduces test MSE by 50%, while decaying the learning rate reduces it by a dramatic 98%. In contrast, tying the exponents ($\iota = \kappa$) makes little difference. The improvements are important for obtaining a good law from small-scale experiments, but even without them the law is present.

3.2 The Importance of Hyperparameter Tuning

Figure 3 shows what happens with too little tuning effort. With 4 configurations per scale, the scaling law is completely absent. With 16, the scaling law has yet to fully appear; it cannot be estimated

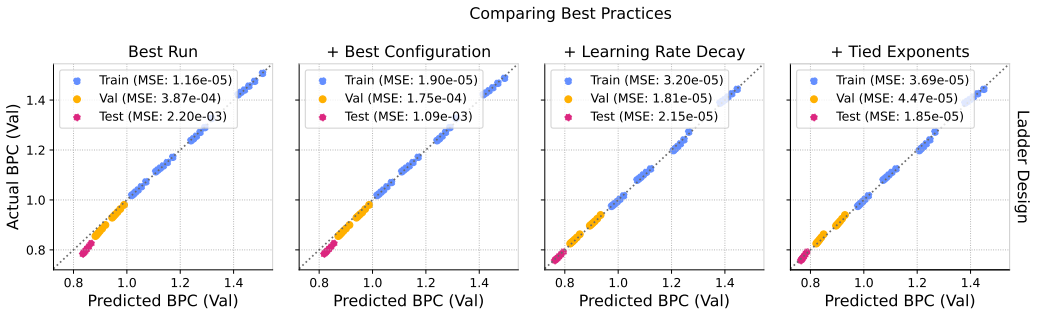


Figure 2: *Better methodology improves the scaling law, but does not make or break it.* A recognizable law emerges from the outset and extrapolation steadily improves as we move from fitting the best run at each scale (left), to the best configuration for each parameter–data budget (middle left), to decaying the learning rate (middle right). Unlike the other refinements, tying the exponents (right) gives inconsistent results: it slightly improves test but hurts validation.

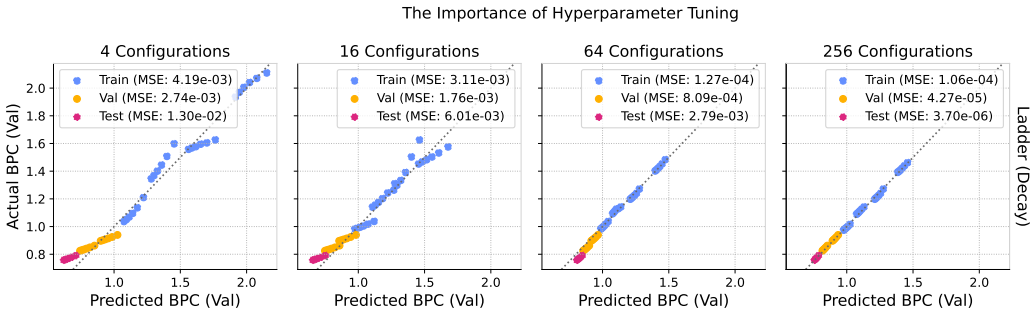


Figure 3: *Hyperparameter tuning is the single most important step in estimating a scaling law.* Each panel fits a scaling law with a more substantial sweep; all panels evaluate (`val` and `test`) on the best configurations from our entire sweep. With 4 (left) or 16 configurations (middle left), the law is not even visible. Increasing to 64 (middle right), it clearly emerges but is imprecise. Only at 256 configurations (right) do we obtain an accurate law.

reliably as the suboptimal configurations inject too much noise. At 64, the law is clearly visible though extrapolation is weak. Only at 256 configurations do we obtain an accurate scaling law. In our previous study, each refinement improved precision on an already recognizable law; here, with too few configurations there is nothing to fit.

Thus one ingredient above all is required to estimate a scaling law, rigorous hyperparameter tuning. This fact explains why small-scale laws are so easy to miss: uncovering them demands a search far more extensive than most ever run. But why are small models so much more sensitive to their hyperparameters? We will show this change in sensitivity comes from how the hyperparameter loss surface evolves as models scale.

4 Understanding the Hyperparameter Loss Surface Across Scales

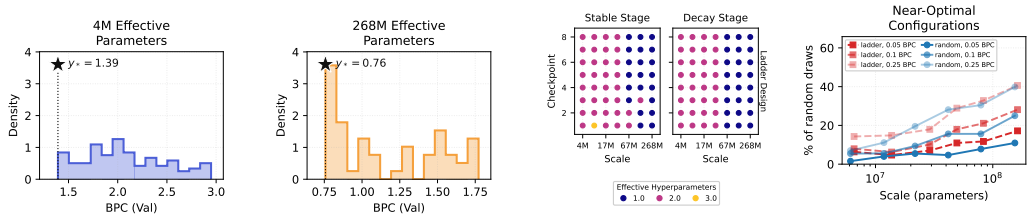
Throughout this section, we will make use of the same experiments we ran in the previous section. Note, that we additionally fit a noisy quadratic distribution to each scale for our analysis. We show that models become less sensitive to hyperparameters as they scale (§4.1). This phenomenon traces back to the hyperparameter loss surface: larger scale makes it lower dimensional (§4.2). These facts influence how we design experiments (§4.3): small scales require extensive search, while large scales are easy to adapt.

4.1 Hyperparameter Tuning Gets Easier with Scale

With scale, good configurations become more common; thus, larger models trained longer are easier to tune.

Near-optimal configurations fill a larger fraction of the space. The effort required to find a good configuration depends on the fraction of the space they take up. We can measure this volume via Monte Carlo integration: sample configurations uniformly and check what percent land near the optimum. This procedure is exactly random search. Figure 4(a,b) shows how the score distribution changes with scale: as models grow, the distribution concentrates on the best achievable loss and the density forms a peak. Thus scale makes models easier to tune.

Hyperparameter sensitivity decreases through both parameters and data. We show in Figure 4(d) that the percentage of configurations within different loss thresholds of the optimal loss increases with scale for both the ladder and random designs. Additionally, neither parameters nor data alone is responsible for this decrease in sensitivity; the effect comes from both. In fact, Figure 16 shows the minimum, 10th, and 25th percentile validation losses from random search and illustrates how sensitivity decreases across each. The gap from the 25th percentile to the minimum



(a) Scores at 4M (b) Scores at 268M (c) Effective hyperparameters (d) Near-optimal configurations

Figure 4: *Smaller scales are more sensitive to hyperparameter tuning.* We visualize the distribution over validation losses at the end of training from random search, in bits per character. **(a)** At 4M the distribution is broad and flat, so fewer configurations land near the best achievable loss, y_*. **(b)** At 268M, the density concentrates into a peak close to y_*, three times higher on the shared axis than the 4M scale. These results demonstrate that more hyperparameter configurations land next to the optimum at larger scales and that smaller scales are more sensitive to hyperparameter tuning. **(c)** The effective hyperparameter count (γ) is the intrinsic dimension of the hyperparameter loss surface. We estimate γ at each parameter–data budget by fitting the noisy quadratic distribution to results from random search. With and without learning rate decay, γ trends lower as parameters increase, while increasing data has much less of an effect. **(d)** The percentage of configurations within 0.1 bits per character of y_* grows from 7.9% to 28.1% between 4M and 134M.

directly measures sensitivity because a small gap means that a quarter of all draws are already close to optimal. Scaling one dimension while the other is held small lowers the loss and slightly moves this gap, but scaling one while the other is large collapses it.

4.2 The Effective Number of Hyperparameters Decreases with Scale

Intuitively, a model with more hyperparameters is harder to tune; in practice, models can have many while their effective number is still low. Formally, the effective number of hyperparameters (γ) is the surface's intrinsic dimension at the optimum, a major determinant of tuning difficulty [48]. We estimate γ at each parameter–data budget by fitting a noisy quadratic to the tail from random search (all losses below a threshold, θ). We show these estimates against model size in Figure 4(d), where **larger models have fewer effective hyperparameters**, dropping to one. Moreover, checkpoints across training largely share the same intrinsic dimension, while increasing parameters clearly reduces γ.

Thus the effective number of hyperparameters shrinks as parameters grow. Data seems to have less effect. Interestingly, prior work found the effective hyperparameters remains constant when using the same search space across different architectures [48, §D]. It seems counter-intuitive that γ depends on scale but not architecture. Moreover, scale's effect might change under an alternative parametrization, such as maximum update [84]. All these factors likely influence γ at some point, our results refine a prior for typical regimes. Better understanding these limits would be a valuable direction for future work.

4.3 Implications for Small-Scale Experiments

The HP loss surface is not a black box; it has rich qualitative structure. We showed empirically that as parameters and data grow, it gets easier to optimize, good configurations fill the space, and the intrinsic dimension goes down. Good reason to hope that there is a simple structure underpinning the HP loss surface, which we may exploit when optimizing at large scales. After all, [48] already showed the noisy quadratic to be a good model for the outcomes of HP optimization at a single scale. Our efforts to model the HP surface across scales however showed no consistent success. We describe several interpretable models in §C (and tried many more). While simple models capture coarse features of the surface, none characterize its finer details. The simplest model, one ignoring how HPs change with scale, is already a strong baseline. This outcome agrees with the qualitative picture: extrapolating to larger scales, HPs have less impact and little variation is left to model. As a result, we treat HP optimization independently per scale.

5 Designing Small-Scale Experiments

We synthesize our discussion into a methodology for small-scale experiments (§5.1). It emphasizes a holistic view because while regularities exist across scales, extrapolation hits statistical limits (§5.2). We demonstrate the approach on a case study from the literature: comparing pre-norm and post-norm transformers (§5.3).

5.1 A Methodology for Small-Scale Experiments

Our approach relies on three facts:

1. with the pretraining data held fixed, capabilities depend on pretraining loss alone (§2.3),
2. with rigorous hyperparameter tuning, scaling laws for pretraining extend to tiny scales (§3),
3. and as models scale, they grow far less sensitive to their hyperparameters (§4).

These principles prescribe a methodology for model-centric research: tune and measure at the small scale, understand how pretraining loss changes, then carry the winner up. The first fact says capability corresponds to perplexity, so we can evaluate models by their cost to obtain it; the second says pretraining laws can be observed at the small scale; and the third says the final scale-up is the easy part.

We expect these assumptions to hold, but should detect when they do not. The shape of small-scale experiments lets us do just that: we can exploit the anticipated structure, check when it fails, and turn those failures into useful diagnostics—for example, if either the noisy quadratic limit or scaling law fails to emerge then there is likely an issue with the hyperparameters, model, or implementation.

More than checking assumptions, however, these diagnostics build a qualitative understanding of how the loss surface scales. Such qualitative understanding is critical because quantitative approaches can compound small errors, even when based on sound structure. More than pure extrapolation, a holistic approach is necessary.

5.2 The Pitfalls of Pure Extrapolation

Though scaling laws exist at small scales, their ability to extrapolate is another matter. We saw small-scale laws successfully extrapolate across an order of magnitude (§3); however, what happens when we go further? Figure 5 illustrates how extrapolation magnifies the sampling error. Comparing independent estimates from different samples, we see the statistical noise: the irreducible error varies wildly across estimates.

If you compare models by extrapolating their scaling laws, at some point you are comparing their estimates of irreducible error. Such estimates are not reliable until the law begins to saturate. Scaling laws exist at small scales, and comparing them is useful; however, you cannot simply pick the model with the lowest final loss. The scaling law is there, but the statistical power is not. Instead, the law is much more reliable near the data.

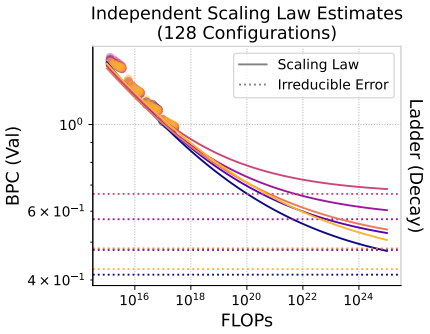


Figure 5: *Despite good agreement near the data, extrapolation magnifies small differences due to sampling error.* We compare independent scaling law estimates, each based on 128 runs at 4M, 8M, 17M, and 34M parameters. Extrapolation eventually reflects the irreducible error, a quantity whose estimates vary greatly across different samples.

5.3 Case Study: Pre-Norm vs. Post-Norm Architectures

Where should normalization layers go in a transformer? The original placed them *after* the residual connection (post-norm) [76]. This made training delicate, needing careful tuning. To stabilize training, Baevski & Auli [2] moved them *before* the sublayer (pre-norm). A year later, an influential empirical study favored pre-norm in most settings yet, against today's practice, found post-norm better for high-resource machine translation [58]. The matter only really settled when Xiong et al.

[83] showed theoretically that pre-norm improves gradients at initialization. Pre-norm's advantage is most pronounced in deeper models, whose experiments are more expensive. Resolving it took nearly three years and a dedicated theoretical analysis. We will recover the answer from straightforward small-scale experiments.

A few well-separated scales should be far more informative than many nearby ones; we use three: two to fit the law, and one to validate it. For post-norm, we sample 511 configurations at 4M, 512 at 34M, and 128 at 134M; for pre-norm, we sample 128 at 4M, 34M, and 134M. We fit on the smaller two then evaluate on the largest one. This search is affordable because when parameters and data scale at the same rate compute is quadratic in model size. A single 1B-parameter run costs as much as 64 runs at 134M, 1,024 runs at 34M, or 65,536 runs at 4M. Thus, our experiment costs as much as a handful of billion parameter runs.

We now deepen our understanding of each architecture through a sequence of qualitative analyses. Each analysis is a diagnostic that asks a question and explores the structure we expect to find.

Diagnostic 1: have we tuned thoroughly? If random search approaches the optimum, then the score distribution converges to a noisy quadratic as discussed in Section 2.2 and established by Lourie et al. [48]. When the limit fails, it is likely the search missed the optimum. We visualize the limit for each architecture in Figure 17. For pre-norm it emerges readily; for post-norm it emerges too, but only deeper into the tail. This smaller asymptotic regime is an early indication of increased hyperparameter sensitivity. We seem to have located the optimum, but the tuning difficulty is itself our first finding: post-norm is sensitive to its hyperparameters, as the literature suggests.

Diagnostic 2: will scaling up be easy? If larger models fail to become less sensitive to their hyperparameters, then transferring from small to large scales might fail as well. Therefore, we examine how sensitivity scales for the architectures at hand. Figure 18 compares the score distribution across scales for each. Both distributions move in the right direction, shifting mass towards the minimum as good configurations fill more of the space. For pre-norm the change is pronounced, growing a distinct peak at the optimum; for post-norm it is only slight, as a second peak lingers at a suboptimal value. Post-norm again exhibits tuning difficulty.

Diagnostic 3: does perplexity still track capabilities? To evaluate by pretraining loss, it must correspond to capabilities (§2.3). This perplexity–capability correspondence should hold when pretraining data is fixed; still, the claim is easy to test as evaluating on a few tasks is much cheaper than pretraining the model. These tasks must show signal at small scales, like AI2 ARC (Easy) [14]. Figure 19 plots downstream accuracy vs. pretraining loss for several such tasks. Regardless of the architecture, all points fall on a common trend. Equal loss implies similar capabilities and a better pretraining law yields a more cost-effective model.

Diagnostic 4: do the scaling laws emerge? To estimate the scaling laws, we adopt improvements from §3.1: tuning the hyperparameters per parameter–data budget and decaying the learning rate. Since extrapolation primarily reflects the noisy irreducible error (§5.2), we try fitting the laws jointly with tied irreducible error, as well as leaving them free to differ. Figure 20 validates these scaling laws on the held-out scale. The free vs. joint laws extrapolate comparably. Pre-norm exhibits an excellent law, while post-norm's law is reasonable but could use improvement from further hyperparameter tuning. Again, hyperparameters present a challenge.

With a better understanding of each model, we now pool all sizes together and compare how the architectures scale. Figure 21 presents the result: pre-norm scales better over the range of sizes we explore. Remembering the statistical difficulty of extrapolating the scaling law (§5.2), we consider two kinds of comparisons. If we let each scaling have its own irreducible error, then the post-norm architecture appears to eventually outperform (beyond the scales we tested); however, if we share the irreducible error between the laws then the pre-norm architecture remains more efficient.

From a holistic viewpoint, post-norm was harder to tune at every turn. Even a better architecture in theory is worse in practice if we cannot find good hyperparameters. Beyond that, pre-norm scales better where the scaling laws are most reliable: near the data. Finally, when we assume both architectures reach the same final loss, pre-norm maintains its scaling advantage. These observations support pre-norm as the better choice.

6 CONCLUSION

Scaling laws promised experiments would stay cheap; however, this promise has gone unfulfilled. Today, many believe regularity only emerges at large scales, and conclusions drawn at small scales will not transfer. This misconception stems from a simple, unappreciated fact: small models are much more sensitive to their hyperparameters. While scaling laws extend to very small sizes, they only emerge on the fully tuned frontier. Reaching it requires an extensive search that researchers rarely conduct. Scaling laws predict pretraining loss, and better loss does not always mean better capability; however, loss also fully captures capability—as long as the data is held fixed. Therefore, a researcher who seeks not to predict capabilities but only to improve a model can rely on loss since a model with a better law buys every loss, and thus any capability, at lower cost.

These insights suggest a new methodology of small-scale experiments, one that exploits the strong assumptions we expect to hold true while checking when they do not. Working at small scales makes this rigor affordable: we can train hundreds of models then use statistical tools, like the noisy quadratic limit, to test our assumptions. Small-scale experiments save compute, but more than that they make our conclusions more robust.

Nevertheless, small-scale experiments have several limitations. While scaling laws exist at small scales, their extrapolation hits statistical challenges. The small scale is informative, but requires judgment. Looking at the bigger picture, is the model hard to tune? Is it getting easier? How does scaling compare where we trust the law most? Instead of a single test, we should explore each of these questions using different diagnostics. Perhaps the most important limitation of all, though, is the restriction to model-centric research. Changing the data breaks the perplexity–capability correspondence which pretraining loss requires to proxy for downstream tasks. With neither pretraining loss as a proxy nor reliable downstream scaling laws, small-scale experiments have limited ability to predict large-scale outcomes. Hopefully future work will unlock small-scale experiments for data-centric research; still, with care we can use small-scale experiments for model-centric research today.

Knowing what we know now, what would we change about how we design the experiments that estimate scaling laws? Appendix D works through this question in full; here we sketch the reasoning.

First, the goal is not necessarily the most accurate scaling law, but the smallest error in the loss estimate $\hat{y}(u_\star)$ at the scale $u_\star$ we extrapolate to. And as we have shown, the two are not necessarily the same (see Figure 2 for empirical and Section D.2 for theoretical evidence). A scaling law fit is similar to a linear regression in log-log space; in the linear case, we can propagate any error ϵ_j made at a small scale u_j to its effect on the extrapolated prediction $\hat{y}(u_\star)$. We find that not all errors are equal, some will have a more severe effect on the prediction than others (see Figure 11). We then tie the error at each scale to the tuning effort spent there. With M_j random-search runs at scale u_j, the expected error shrinks as $\epsilon_j \sim M_j^{-2/\gamma_j}$, where γ_j is the effective number of hyperparameters at that scale (Section D.1).

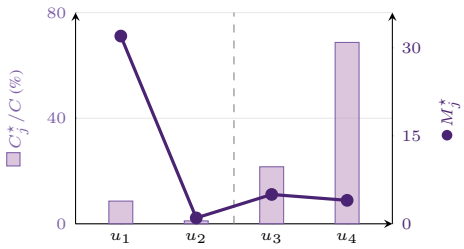


Figure 6: **Getting best scaling law fit requires different number of hyperparameter runs per scale.** The plot illustrates a typical solution, to optimizing the error at a scale $u_\star$ for a given compute budget. Because runs are cheap at u_1, we can run many runs while spending very little of the overall compute. The opposite is the case for the larger scale u_4. Another factor is the proximity to the center of the scales (dotted line). The closer a point is to the center the less influence it is going to have on the scaling law estimate, hence we are willing to make a larger mistake (see u_2). For the full derivation of this picture see Section D.3.

Combining both insights turns experiment design into a constrained optimization problem: given a fixed compute budget, we determine the number of runs at each scale to minimize the expected error at the target scale $u_\star$ (Section D.3). We can solve this problem in closed form. An illustration of what a solution will typically look like is in Figure 6. We develop this argument in full in Appendix D and distill it into a manual for designing small-scale experiments requiring the practitioner to only know their compute budget and the compute it takes to run a model at any given scale. However, executing this protocol we leave for future work.

AI USE STATEMENT

We used AI assistants to help draft the text of this paper. In addition, we initially developed the theoretical models presented in Appendix C without AI assistance, then afterwards discussed these models, their justification, and possible extensions with AI assistants. Claude suggested the insight that any decent hyperparameters should drive a neural networks' loss to the irreducible error in the infinite parameters/data limit. In addition, some of the formulas used in implementing these models were derived and/or checked with AI assistance, in addition to being reviewed by the authors. AI coding assistants were used to help implement and run some of the experiments. Most plots were created with extensive help from AI assistants.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

[1] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023. URL https://arxiv.org/abs/2305.10403.

[2] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling, 2018. URL https://arxiv.org/abs/1809.10853v1.

[3] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024. doi: 10.1073/pnas.2311878121. URL https://www.pnas.org/doi/abs/10.1073/pnas.2311878121.

[4] 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 D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 125153–125188. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b5f78a17a94da3e34c935515d1b6adae-Paper-Conference.pdf.

[5] Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A. Smith, Dirk Groeneveld, Pang Wei Koh, Jesse Dodge, and Hannaneh Hajishirzi. Establishing task scaling laws via compute-efficient model ladders. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=FeAM2RVO8l.

[6] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[7] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, 2021. URL https://crfm.stanford.edu/assets/report.pdf.

[8] Richard J Brook and Gregory C Arnold. *Applied regression analysis and experimental design*. Routledge, 2018.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[10] Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. Scaling laws for predicting downstream performance in LLMs. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=PJUbMDkQVY.

[11] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2829, June 2023.

[12] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee,

Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023. URL http://jmlr.org/papers/v24/22-1144.html.

[13] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

[14] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

[15] Corinna Cortes, L. D. Jackel, Sara Solla, Vladimir Vapnik, and John Denker. Learning curves: Asymptotic values and rate of convergence. In J. Cowan, G. Tesauro, and J. Alspector (eds.), *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper_files/paper/1993/file/1aa48fc4880bb0c9b8a3bf979d3b917e-Paper.pdf.

[16] Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, Tamay Besiroglu, and David Owen. The rising costs of training frontier ai models, 2025. URL https://arxiv.org/abs/2405.21015.

[17] Laurens De Haan and Ana Ferreira. *Extreme value theory: an introduction*. Springer, 2006.

[18] 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. URL https://arxiv.org/abs/2401.02954.

[19] Jean-Stanislas Denain and Cheryl Wu. Final training runs account for a minority of R&D compute spending, 2026. URL https://epoch.ai/gradient-updates/r-and-d-vs-training-compute. Accessed: 2026-06-13.

[20] Singularity Developers. Singularity, 2021. URL https://doi.org/10.5281/zenodo.1310023.

[21] Nolan Dey, Bin Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 137707–137739. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c917d8b9e01427f3184d80ade22f4d1f-Paper-Conference.pdf.

[22] Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 53138–53167. Curran Associates, Inc.,

2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5f1eee2509599faeeb3570a887016a64-Paper-Conference.pdf.

[23] Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Luca Soldaini, Jenia Jitsev, Alex Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, Thomas Kollar, Yair Carmon, Achal Dave, Reinhard Heckel, Niklas Muennighoff, and Ludwig Schmidt. Language models scale reliably with over-training and on downstream tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=iZeQBqJamf.

[24] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

[25] Boris Gnedenko. Sur la distribution limite du terme maximum d'une serie aleatoire. *Annals of mathematics*, 44(3):423–453, 1943.

[26] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide

Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[27] Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 76232–76264. Curran Associates, Inc., 2024. doi: 10.52202/079017-2427. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/8b970e15a89bf5d12542810df8eae8fc-Paper-Conference.pdf.

[28] Trevor Hastie, Robert Tibshirani, Jerome Friedman, et al. The elements of statistical learning, 2009.

[29] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[30] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling, 2020. URL https://arxiv.org/abs/2010.14701.

[31] 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, 2017. URL https://arxiv.org/abs/1712.00409.

[32] Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning, 2023. URL https://arxiv.org/abs/2301.13442.

[33] 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 S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

[34] Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, dahai li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

[35] Yuzhen Huang, Jinghan Zhang, Zifei Shan, and Junxian He. Compression represents intelligence linearly. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=SHMj84U5SH.

[36] Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026. ISSN 1476-4687. doi: 10.1038/s41586-025-09833-y. URL https://doi.org/10.1038/s41586-025-09833-y.

[37] Maor Ivgi, Yair Carmon, and Jonathan Berant. Scaling laws under the microscope: Predicting transformer performance from small scale experiments. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 7354–7371, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.544. URL https://aclanthology.org/2022.findings-emnlp.544/.

[38] Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions, volume 1*, volume 1. John wiley & sons, 1994.

[39] Andy L. Jones. Scaling scaling laws with board games, 2021. URL https://arxiv.org/abs/2104.03113.

[40] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

[41] 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. URL https://arxiv.org/abs/2001.08361.

[42] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336. URL https://www.science.org/doi/abs/10.1126/science.adi2336.

[43] 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 D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 120612–120654. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ae9f1d3b89a81fcb1c45b4fc1afbd4fe-Paper-Conference.pdf.

[44] 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, 2025. URL https://arxiv.org/abs/2503.04715.

[45] Margaret Li, Sneha Kudugunta, and Luke Zettlemoyer. (mis)fitting scaling laws: A survey of scaling law fitting techniques in deep learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xI71dsS3o4.

[46] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[47] Nicholas Lourie, Kyunghyun Cho, and He He. Show your work with confidence: Confidence bands for tuning curves. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3455–3472, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.189. URL https://aclanthology.org/2024.naacl-long.189/.

[48] Nicholas Lourie, He He, and Kyunghyun Cho. Hyperparameter loss surfaces are simple near their optima. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=IAoSG4Q2xC.

[49] Nicholas Lourie, Michael Y. Hu, and Kyunghyun Cho. Scaling laws are unreliable for downstream tasks: A reality check. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 16167–16180, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.877. URL https://aclanthology.org/2025.findings-emnlp.877/.

[50] Rafid Mahmood, James Lucas, David Acuna, Daiqing Li, Jonah Philion, Jose M. Alvarez, Zhiding Yu, Sanja Fidler, and Marc T. Law. How much more data do i need? estimating requirements for downstream tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 275–284, June 2022.

[51] Prasanna Mayilvahanan, Thaddäus Wiedemer, Sayak Mallick, Matthias Bethge, and Wieland Brendel. LLMs on the line: Data determines loss-to-loss scaling laws. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 43302–43330. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/mayilvahanan25a.html.

[52] Ian R. McKenzie, Alexander Lyzhov, Michael Martin Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Xudong Shen, Joe Cavanagh, Andrew George Gritsevskiy, Derik Kauffman, Aaron T. Kirtland, Zhengping Zhou, Yuhui Zhang, Sicong Huang, Daniel Wurgaft, Max Weiss, Alexis Ross, Gabriel Recchia, Alisa Liu, Jiacheng Liu, Tom Tseng, Tomasz Korbak, Najoung Kim, Samuel R. Bowman, and Ethan Perez. Inverse scaling: When bigger isn't better. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=DwgRm72GQF. Featured Certification.

[53] Amil Merchant, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990): 80–85, 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06735-9. URL https://doi.org/10.1038/s41586-023-06735-9.

[54] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

[55] Bruno Kacper Mlodozeniec, Pierre Ablin, Louis Béthune, Dan Busbridge, Michal Klein, Jason Ramapuram, and marco cuturi. Completed hyperparameter transfer across modules, width, depth, batch and duration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=elB9k4nTL1.

[56] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to Linear Regression Analysis, 6e Solutions Manual*. John Wiley & Sons, 2022.

[57] Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. *Journal of Machine Learning Research*, 26(53):1–66, 2025. URL http://jmlr.org/papers/v26/24-1000.html.

[58] Toan Q. Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. In Jan Niehues, Rolando Cattoni, Sebastian Stüker, Matteo Negri, Marco Turchi, Thanh-Le Ha, Elizabeth Salesky, Ramon Sanabria, Loic Barrault, Lucia Specia, and Marcello Federico (eds.), *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, November 2-3 2019. Association for Computational Linguistics. URL https://aclanthology.org/2019.iwslt-1.17/.

[59] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads, 2022. URL https://arxiv.org/abs/2209.11895.

[60] OpenAI. Gpt-4 technical report, 2023.

[61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[62] Tim Pearce and Jinyeop Song. Reconciling kaplan and chinchilla scaling laws. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=NLoaLyuUUF. Reproducibility Certification.

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[64] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

[65] Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *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/file/b6341525cd84f3be0ef203e4d7cd8556-Paper-Conference.pdf.

[66] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher, 2022. URL https://arxiv.org/abs/2112.11446.

[67] 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.

[68] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ITw9edRDlD.

[69] Rylan Schaeffer, Hailey Schoelkopf, Brando Miranda, Gabriel Mukobi, Varun Madan, Adam Ibrahim, Herbie Bradley, Stella Biderman, and Sanmi Koyejo. Why has predicting downstream capabilities of frontier AI models with scale remained elusive? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=I1NtlLvJal.

[70] Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

[71] Charlie Victor Snell, Eric Wallace, Dan Klein, and Sergey Levine. Predicting emergent capabilities by finetuning. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=vL8BIGuFTF.

[72] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew Kyle Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, Cesar Ferri, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Christopher Waites, Christian Voigt, Christopher D Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, C. Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germàn Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-Lopez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Francis Anthony Shevlin, Hinrich Schuetze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B Simon, James Koppel, James Zheng, James Zou, Jan Kocon, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik

Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh Dhole, Kevin Gimpel, Kevin Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros-Colón, Luke Metz, Lütfi Kerem Senel, Maarten Bosma, Maarten Sap, Maartje Ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramirez-Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael Andrew Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swedrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan Andrew Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter W Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Russ Salakhutdinov, Ryan Andrew Chi, Seungjae Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel Stern Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Shammie Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven Piantadosi, Stuart Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Venkatesh Ramasesh, vinay uday prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Sophie Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj. Featured Certification.

[73] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

[74] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[75] Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. ISSN 1476-4687. doi: 10.1038/s41586-023-06747-5. URL https://doi.org/10.1038/s41586-023-06747-5.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[77] 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.

[78] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[79] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

[80] Jason Wei, Najoung Kim, Yi Tay, and Quoc V. Le. Inverse scaling can become u-shaped, 2023. URL https://arxiv.org/abs/2211.02011.

[81] Ethan G Wilcox, Michael Hu, Aaron Mueller, Tal Linzen, Alex Warstadt, Leshem Choshen, Chengxu Zhuang, Ryan Cotterell, and Adina Williams. Bigger is not always better: The importance of human-scale language modeling for psycholinguistics, Jul 2024. URL osf.io/preprints/psyarxiv/rfwgd_v1.

[82] Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Veselin Stoyanov. Training trajectories of language models across scales. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13711–13738, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.767. URL https://aclanthology.org/2023.acl-long.767/.

[83] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/xiong20b.html.

[84] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

[85] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite depth neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=17pVDnpwwl.

[86] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

[87] Rosie Zhao, Tian Qin, David Alvarez-Melis, Sham Kakade, and Naomi Saphra. Random scaling of emergent capabilities, 2026. URL https://arxiv.org/abs/2502.17356.

APPENDIX

A ADDITIONAL RELATED WORK

A.1 COMPUTE SCALING

Equation 1 omits the hyperparameters, describing loss at the *fully tuned frontier*: the best attainable loss given the parameters and data ; however, parameters and data do not vary independently. The model's gradients must be computed for every example, so their product determines the total compute: $c \propto pd$. Given a budget, you face a trade-off between larger models and longer training. Navigating this trade-off is exactly why people turn to scaling laws. Minimizing Equation 1 with fixed compute yields the condition:

$$\iota \frac{\zeta}{p^{\iota}} = \kappa \frac{\eta}{d^{\kappa}} \tag{2}$$

Hence the optimal parameters and data become power laws in compute:

$$p \propto c^{\frac{\kappa}{\iota+\kappa}} \qquad d \propto c^{\frac{\iota}{\iota+\kappa}} \tag{3}$$

If you plug these values into the joint scaling law then you obtain excess loss as a power law in the compute:

$$\mathcal{L}(p, d) - \epsilon \propto c^{-\frac{\iota\kappa}{\iota+\kappa}} \tag{4}$$

Thus scaling laws reveal the best loss given the compute and how to get it by balancing parameters and data.

Hoffmann et al. [33] found balanced parameters and data grow at the same rate, a conclusion since echoed by many others [1; 18]. This idea justifies the token-to-parameter ratio as a meaningful measure of how well-trained a model is, with the compute-optimal ratio serving as baseline. Under Equation 2, this same-rate growth only occurs when the scaling exponents are equal: $\iota = \kappa$. Some theoretical work predicts this equality [3], and the estimates are often close in practice. Researchers have also explored enforcing it to improve estimation [57; 45].

A.2 THE NOISY QUADRATIC LIMIT

How do we know if the hyperparameters are fully tuned, or how hard that is to accomplish? The *noisy quadratic limit* answers these questions. Lourie et al. [48] found that near the optima, loss surfaces are:

1. **Quadratic**: well approximated by the second-order Taylor expansion;
2. **Low-Rank**: governed by a few directions that explain most of the variation; and
3. **Normal**: distributed about the mean as additive normal noise.

Empirically, this approximation holds for a large region around the optimum, the *asymptotic regime*. This region determines the practical questions we care about, like tuning difficulty or the best attainable performance.

Formally, if $\mathbf{X}$ are the hyperparameters, Y the validation score, and $(\mathbf{x}_*, y_*)$ the optimum, then the conditional distribution $Y \mid \mathbf{X}$ is approximately a quadratic plus additive normal noise, E.[3] Letting $H_{\mathbf{x}_*}$ be the Hessian:

$$\mathcal{L}(\mathbf{X}) \approx y_* + (\mathbf{X} - \mathbf{x}_*)^T H_{\mathbf{x}_*} (\mathbf{X} - \mathbf{x}_*) + E, \qquad E \sim \mathcal{N}(0, \sigma) \tag{5}$$

This conditional view induces a marginal structure that emerges during random search. More precisely, the search induces a distribution over validation scores, and that distribution's tail converges to a *noisy quadratic distribution*, $\mathcal{Q}_{\min}(\alpha, \beta, \gamma, \sigma)$. Writing $F(y; \alpha, \beta, \gamma, \sigma)$ for the CDF of the noisy quadratic distribution, then:

$$\mathbb{P}(Y \leq y) \approx F(y; \alpha, \beta, \gamma, \sigma) \quad \text{as} \quad y \to y_*. \tag{6}$$

[3] Integral hyperparameters (e.g., batch size) can be treated as continuous; discrete ones are handled with separate quadratics.

The distribution's parameters capture properties of the loss surface: α is the best attainable performance (y_*), β measures how tightly the scores concentrate around it, γ gives the *effective* number of hyperparameters (intrinsic dimension at the optimum), and σ represents the noise due to random seeds. In practice, we define the asymptotic regime by a threshold on the expected loss, $\mathbb{E}[Y \mid \mathbf{X}] \leq \theta$, marking where this structure holds.

The **noisy quadratic distribution** arises as the sum $Q + E$, where $Q \sim \mathcal{Q}_{\min}(\alpha, \beta, \gamma)$ is the (noiseless) quadratic distribution and $E \sim \mathcal{N}(0, \sigma)$ is independent Gaussian noise. The noiseless quadratic is supported on $[\alpha, \beta]$ with CDF and density

$$F_Q(y; \alpha, \beta, \gamma) = \left(\frac{y-\alpha}{\beta-\alpha}\right)^{\gamma/2}, \qquad f_Q(y; \alpha, \beta, \gamma) = \frac{\gamma}{2(\beta-\alpha)}\left(\frac{y-\alpha}{\beta-\alpha}\right)^{\frac{\gamma-2}{2}}, \tag{7}$$

for $\alpha \leq y \leq \beta$ (and 0, resp. 1, outside). Its lower tail decays polynomially with exponent $\gamma/2$, encoding the ellipsoidal volume of near-optimal hyperparameters: the rank of the Hessian sets γ, so higher intrinsic dimension makes near-optimal configurations rarer. Convolving with the Gaussian gives the noisy quadratic, supported on all of $\mathbb{R}$, with CDF and density

$$F(y; \alpha, \beta, \gamma, \sigma) = \Phi\left(\frac{y-\beta}{\sigma}\right) + \int_0^1 v^{\gamma/2}\, p_V(v)\, \mathrm{v}, \tag{8}$$

$$f(y; \alpha, \beta, \gamma, \sigma) = \frac{\gamma}{2(\beta-\alpha)} \int_0^1 v^{\frac{\gamma-2}{2}}\, p_V(v)\, \mathrm{v}, \tag{9}$$

where Φ is the standard normal CDF and p_V is the density of $V \sim \mathcal{N}\left(\frac{y-\alpha}{\beta-\alpha}, \frac{\sigma}{\beta-\alpha}\right)$. The Φ term accounts for noise carrying scores past β, while the truncated integral convolves the quadratic tail with the noise over the support $[\alpha, \beta]$ (i.e., $v \in [0, 1]$). As $\sigma \to 0$, the integral concentrates and Equation 8 recovers the noiseless quadratic Equation 7.

A.3 Downstream Scaling laws

The data defines a curve of capabilities; the loss decides a point on that curve; the scaling law determines the cost of that point. Figure 7 illustrates this principle: models trained at different scales or for different lengths fall along a single (possibly noisy) curve. The curve varies wildly across tasks, but exists for all of them. Regardless of whether better capabilities lie down the road, the scaling law sets the toll. In other words a model with a better scaling law achieves every perplexity, and thus any capability, at lower cost.

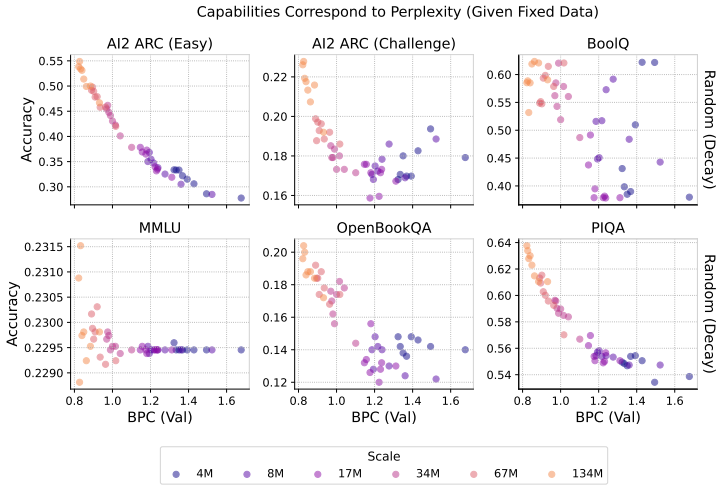


Figure 7: *Given a fixed corpus, perplexity corresponds to downstream capability across a wide variety of tasks.* The relationship between pretraining loss and capability is not always predictable or even monotonic; however, models achieving the same loss have a similar conditional distribution over their capabilities as we increase parameters and data, or even randomize the architecture, as long as the *composition* of pretraining data remains fixed.

B EXPERIMENTAL DESIGN

Our experiments explore the hyperparameter loss surface across scales via random search.

Models. We measure model size by the *effective parameter count*, parameters adjusted for FLOPs, so that compute is $c = 6pd$ for p effective parameters and d tokens. This count excludes the embedding layer (which uses no FLOPs) but includes the unembedding layer and the cost of attention [65, §B]. We target parameter counts at increasing powers of two: $2^{22} \approx 4$M to $2^{28} \approx 268$M, and construct models at each scale with two experimental designs: a hand-crafted model ladder and randomly sampled architectures.

Training and evaluation. We train with a warmup-stable-decay (WSD) schedule [34]. Following Hägele et al. [27], we reuse the stable phase from the main run and branch off decay phases from checkpoints taken $1/8, 2/8, \ldots, 8/8$ through training. We evaluate checkpoints before and after decaying the learning rate, thus a single sweep yields stable and decayed evaluations across eight token budgets at a fraction of the cost.

Analysis. We partition the scales into training (4M–34M), validation (67M–134M), and test (268M), so reported numbers measure extrapolation from small models to larger held-out ones. By default, we fit the joint scaling law of Equation 1 to the best loss attained at each parameter–data budget pair. Like prior work [32; 45], we found shorter training runs fit the scaling law poorly, therefore we discard the two smallest token budgets which noticeably improves performance on our validation set.

B.1 MODELING

The modeling recipe we study is a Llama variant, as implemented in Meta Lingua: `https://github.com/facebookresearch/lingua` (commit: 437d680) [77]. Hyperparameters not specified here were either left to Lingua's defaults or swept over as described in Section B.2.

Data. We pretrain on the 100B token subset of FineWeb-Edu [64]. We use the subset to save storage and preprocessing since the scales we consider require less than 100B tokens. We split this subset into training and validation sets using Lingua's built-in preprocessing pipeline with 48GB of memory and Lingua's suggested seed of 42. We create two such datasets: one shuffling all the data in 1 chunk for single GPU training (4M–134M parameters), and another splitting the data into 8 chunks for 8 GPU training (268M parameters).

Architecture. For the tokenizer, we use `p50k_base` from `tiktoken` (`https://github.com/openai/tiktoken`), the BPE tokenizer from GPT-3.5 (`text-davinci-003`), because it has an appropriately sized vocabulary for our scale. The context length is 1,024 with special tokens to distinguish the beginning and ending of each sequence. For the neural network, we use a decoder-only Llama model [26] available in Lingua (`apps/main/train.py`) [77]. The model uses RMSNorm for normalization [86], SwiGLU for the activation function [70], and RoPE for position embeddings [73]. All runs use a pre-normalization architecture except for those explicitly specified as post-normalization (e.g., in §5), which we implemented as described by Xiong et al. [83]. Aside from Lingua's defaults, we padded the vocabulary size (`model.vocab_size`) to 50,536. In addition, we set `model.multiple_of` to 8 since the default value of 256 is larger than our smaller models' hidden dimensions; 8 compromises between allowing a range of hidden dimensions at small scales while partially exploiting increased GPU efficiency.

Optimization. For optimization, we use AdamW [46] with a warmup-stable-decay (WSD) learning rate schedule [34]. Unlike Lingua, we use a linear cooldown for the WSD schedule decaying to a `optim.lr_min_ratio` of 1e-6 times the peak learning rate. Each scale we train is defined by a target number of effective parameters (described in Section B.2), ranging from $2^{22} \approx 4M$ to $2^{28} \approx 268M$ in powers of two. Training continues until the number of tokens equals 32 times the target number of effective parameters. The total tokens divided by the tokens per gradient step determines the number of steps. The effective parameter count is essentially the same as FLOPs per step as used in Hoffmann et al. [33] (Approaches 1 & 2), thus we train about 50% further than the

$$
\begin{aligned}
batch_size &\sim DiscreteUniform\left(\{64, 128, 256, \ldots, 4{,}096\}\right) \\
lr &\sim LogUniform(1e{-}5, 1e{-}1) \\
beta1 &\sim LogitUniform(0.7, 0.999) \\
beta2 &\sim LogitUniform(0.9, 0.9999) \\
warmup &= \text{round}\left(steps \times P\right), \quad P \sim LogUniform\left(1e{-}3, 1/8\right) \\
weight_decay &\sim LogUniform(1e{-}4, 1e0) \\
rope_theta &\sim LogUniform\left(2^{10}, 2^{20}\right)
\end{aligned}
$$

Figure 8: *The search distribution for all non-architectural hyperparameters.* The log uniform and logit uniform distributions are obtained by taking the log or logit of the bounds, sampling uniformly on this scale, then inverting the transformation back to the original scale.

Chinchilla-optimal ratio of ∼20 tokens-per-parameter, allowing us to explore both the under- and over-trained regimes. Following Hägele et al. [27], we first train each configuration on a main run without decaying the learning rate, saving checkpoints $1/8, 2/8, \ldots, 8/8$ through training; afterwards we decay each of those checkpoints by extending training with 25% more steps, thus the final results have their learning rates decayed over the last 20% of training. The initial checkpoints provide our constant learning rate results (*stable*), whereas the decayed checkpoints provide our results for the full schedule (*decay*); all results include learning rate warmup.

Evaluation. For pretraining, we evaluate the model via bits-per-character (BPC), which normalizes the pretraining loss for differences in tokenization. To compute BPC, we change the logarithm's base on the negative log-likelihood per character already provided by Lingua. We calculate BPC on the validation set, limiting `eval.validation.max_steps` to 1,000. For capabilities, we evaluate the model using Lingua's integration with the EleutherAI `lm-evaluation-harness` (v0.4.11) [24]. We evaluate on six multiple-choice tasks: AI2 ARC (Easy) and (Challenge) [14], BoolQ [13], MMLU [29], OpenBookQA [54], and PIQA [6]. Our evaluations limit `eval.generator.max_tokens` to 16,384 but otherwise use Lingua's defaults.

Random Seeds. We fix the following random seeds made available by Lingua: `seed` to 0, `data.seed` to 1, `model.seed` to 2, `eval.harness.random_seed` to 3, `eval.harness.numpy_random_seed` to 4, `eval.harness.torch_random_seed` to 5, `eval.harness.fewshot_random_seed` to 6. Accordingly, data order and other details were kept constant while runs differed primarily in their hyperparameters.

B.2 Search Distributions

Our results analyze three different experiments, each defined by a search distribution over the hyperparameters.

All the experiments randomly sample configurations across a range of scales, with each scale defined by the *effective* number of parameters. The effective number of parameters is proportional to the FLOPs per token so that $c = 6pd$ when c is compute, p the effective parameters, and d the number of tokens. Following Porian et al. [65, §B], we compute it as:

$$N_{eff} \coloneqq (3d_{ff} + 4d)\, dl + dv + ndl \tag{10}$$

where d_{ff} is the feedforward dimension, d the hidden dimension, l the number of layers, v the vocabulary size, and n the context length of the transformer. In particular, parameters in the embedding layer are not counted because they do not contribute FLOPs, while parameters in the unembedding layer are counted because they do. See Porian et al. [65, §B] for an overview of different ways to count parameters.

To isolate the effect of tuning the architecture, configurations of the same index share all non-architectural hyperparameters across the different designs. Thus, all differences between the experiments come from the architecture alone. The search distribution over the shared hyperparameters

N_{eff}	dim	n_layers	$head_dim$	n_heads
3.47M	64	2	64	1
7.78M	128	4	64	2
17.1M	256	4	64	4
32.4M	384	6	64	6
62.6M	512	10	64	8
133M	768	12	64	12
270M	1,024	16	64	16

Table 1: *The model ladder used for architectural hyperparameters in the* `ladder` *experiment.* The leftmost column gives the actual effective number of parameters which differs slightly from the targeted number due to architectural constraints.

$$n_layers = \text{round}\left(U^2\right), \quad U \sim Uniform\left(\sqrt{2}, \sqrt{48}\right)$$
$$n_heads \sim DiscreteProportional\left(\{k \mid k \text{ divides } {}^{dim}/_{8},\ 1 \le k < \min\left(24, {}^{dim}/_{8}\right)\}\right)$$

Figure 9: *The search distribution for architectural hyperparameters used in the* `random`*,* `prenorm`*, and* `postnorm` *experiments.* The discrete proportional distribution samples from a finite set of positive numbers with probability proportional to each number. By construction, dim is always divisible by `model.multiple_of` $= 8$.

is given in Figure 8. Typically, researchers search hyperparameters like the learning rate or weight decay on a log scale; using the proper scale for each hyperparameter is also important in efficiently reaching the noisy quadratic limit [48]. In practice, we found the simple heuristic of sampling real values on a linear scale, positive values on a log scale, and bounded values on a logit scale worked quite well. The warm up is sampled as a proportion of the total training steps up to the first ${}^{1}/_{8}$ of training so that the first checkpoint (and evaluation) is always fully warmed up.

Model Ladder (`ladder`). We sample configurations for target scales starting from $N_{eff} = 2^{22} \approx 4M$ up to $2^{28} \approx 268M$ in powers of two. Excluding 2 failed runs, we search 867 configurations at 4M effective parameters, 826 at 8M, 867 at 17M, 934 at 34M, 128 at 67M, 128 at 134M, and 64 at 268M. For the `ladder` experiment, we follow the common practice of setting architectural hyperparameters via a hand-crafted model ladder [27; 5]. The hidden dimension, number of layers, and number of attention heads at each scale were set according to Table 1. We increase the width and depth of successive models in order to steadily increase their parameter count, while keeping other architectural elements (such as the $head_dim$) constant. Following Porian et al. [65], we keep the aspect ratio between 32 and 64. The transformer architecture is subject to a number of constraints, depending on the implementation (e.g., $head_dim$ divides dim, dim is used both for the hidden and the embedding dimension, matrix dimensions should be multiples of 8 when running on a GPU, and so on). These constraints must be considered when designing a ladder.

Random Architecture (`random`). We sample configurations for target scales starting from $N_{eff} = 2^{22} \approx 4M$ up to $2^{27} \approx 134M$ in powers of two. Excluding 2 failed runs, we search 128 configurations at 4M effective parameters, 126 at 8M, 128 at 17M, 128 at 34M, 128 at 67M, and 128 at 134M. In the `random` experiment, we also tune the architectural hyperparameters via random search. The number of layers and attention heads for each configuration are sampled from the distribution described in Figure 9. Given the effective parameters, the number of layers determines the hidden dimension via Equation 10, which is a quadratic in d given the other parameters and the fact that $d_{ff} = {}^{8}/_{3}d$ for our model. We round the solution to the nearest multiple of 8 for GPU efficiency. Once we have the hidden dimension, the number of heads determines the head dimension: $dim = head_dim \times n_heads$. We ensure n_layers is at least 2 because induction heads cannot form with fewer layers [59]; we also use a square root scale because we found it balanced making both the number of layers and the hidden dimension more uniform. n_heads must divide

the hidden dimension, so we sample proper divisors of $^{dim}/_{8}$ to guarantee that $head_dim$ is even (required by RoPE) and greater than 16 (required by the attention implementation). Since small divisors are much more common, we bias towards larger ones by sampling divisors with probability proportional to their size. As previously mentioned, transformers are subject to a number of constraints that must be accounted for when sampling architectural hyperparameters: $head_dim \geq 16$ for common attention implementations, $head_dim$ must be even for RoPE, $head_dim$ divides dim, dim provides both the hidden and the embedding dimension, and matrix dimensions should be multiples of 8 for GPU efficiency. These constraints complicate designing a good search distribution;[4] the distribution in Figure 9 has been constructed to obey them.

Pre-Normalization (`prenorm`) vs. Post-Normalization (`postnorm`). We sample configurations for target scales of $N_{eff} = 2^{22} \approx 4M$, $2^{25} \approx 34M$, and $2^{27} \approx 134M$. For `prenorm`, we reuse all the results at those scales from the `random` experiment. For `postnorm`, we reuse the initial configurations of the `random` experiment (including architectural parameters), but we re-run them after changing the transformer to a post-normalization architecture. In addition, we expand the number of configurations due to the post-normalization architecture's increased tuning difficulty. Excluding 1 failed run, this expansion results in 511 configurations at 4M effective parameters, 512 at 34M, and 128 at 134M.

B.3 Analysis

Noisy Quadratic Limit. We generate confidence bands for the eCDF using the LD Highest Density bands from Lourie et al. [47], and we fit the noisy quadratic distribution as described in Lourie et al. [48], both using `opda` (v0.8.0): `https://github.com/nicholaslourie/opda`. To fit the noisy quadratic limit, we set the threshold θ at various percentiles of the score distribution. We used different percentiles for each parameter–data budget, with the size of the asymptotic regime generally increasing in both. In the `ladder` and `random` / `prenorm`) experiments, they were as follows: for 4M, 8M, and 17M effective parameters, 7% at checkpoints 1–6 and 8% at checkpoints 7–8; for 34M, 7% for checkpoints 1–2, 8% at 3, 10% at 4–6, and 12% at 7–8; for 67M, 12% at 1, 15% at 2, 20% at 3, 25% at 4, 30% at 5, 32% at 6, and 35% at 7–8; for 134M and 268M, 12% at 1, 15% at 2, 20% at 3, 25% at 4, 30% at 5, 35% at 6, 40% at 7, and 50% at 8. In the `postnorm` experiment, they were as follows: for 4M effective parameters, 3% at checkpoint 1, 4% at 2, 5% at 3, 6% at 4, and 7% at 5–8; for 34M and 134M, 7% at 1–2, 8% at 3, and 10% at 4–8. We restricted α to be positive (as bits-per-character always is), and we restricted γ to be 1 to at most the nominal number of hyperparameters (7 for `ladder` and 9 for `random` / `prenorm` and `postnorm`).

Scaling Laws. To fit and evaluate the scaling laws, we partition the scales into training (4M–34M), validation (67M–134M), and testing (268M); thus, we evaluate the scaling laws based on their ability to extrapolate from small to large models. We used training to fit the laws, validation to make decisions during our research, and testing for the final evaluation. For the scaling law, we fit Equation 1 to the best validation loss from each parameter–data budget pair, or (where explicitly stated) to the validation losses from the run achieving the minimum final loss at each scale. In both cases, we drop the first two checkpoints ($^{1}/_{8}$ and $^{2}/_{8}$ of training) as it substantially improved the fit [32]. Where specified, we also tie the scaling exponents together ($\iota = \kappa$). To perform the optimization, we minimized the mean squared error via SciPy's `scipy.optimize.differential_evolution` [78], with bounds of $[0, 1]$ for ϵ, $[0, 10]$ for $\ln \zeta$ and $\ln \eta$ (which we fit on log scales), and $[0, 1]$ for ι and κ. We used a `popsize` of 120 except when fitting laws jointly with a single irreducible error for which we used a `popsize` of 60 due to the higher dimensionality and longer runtime. To evaluate the scaling laws, we report mean squared error between the predicted and observed loss on the held-out scales, with held-out scales using the same preprocessing as those in training (e.g., whether the best run or the best configuration for each parameter–data budget is used).

Interpretable Models of the Hyperparameter Loss Surface. In §C, we compare various models of the hyperparameter loss surface. These models are fit to losses generated via the same methodology used for scaling laws (best loss for each parameter–data budget, dropping the first two checkpoints), except instead of selecting the loss for the single best configuration, we take all

[4] A typical model ladder avoids most of these complications, as they primarily appear as edge cases in small networks.

configurations within the asymptotic regime as defined by the fractions used for our noisy quadratic limits. Before fitting the models, we transform the hyperparameters to put them on an appropriate scale: a log-scale for $n_effective_parameters$, n_tokens, $batch_size$, lr, $weight_decay$, dim, n_layers, n_heads, and $rope_theta$; a log-plus-one-scale for $warmup$ (to avoid log of zero); and a logit-scale for $beta1$ and $beta2$. The interpretable models are fit by first taking a good initialization and then improving it with SciPy's `scipy.optimize.minimize` using the `"Nelder-Mead"`, `"Powell"`, and `"BFGS"` methods in sequence. For the initialization, we fit a quadratic polynomial via ordinary least squares, algebraically solve for the equivalent estimate in the format of Equation 12, and then project the Hessian to a lower rank ($\gamma = 3$) by dropping the eigenvalues of least absolute value. We then use the `static` model to initialize the `dynamic optima` and the `dynamic sensitivity` models, setting the new components to zero since the models are nested. We similarly initialize the `dynamic` model from the `dynamic optima` model. For the `multilayer perceptron` model, we implement the neural network using scikit-learn [63]. The pipeline applies a quadratic basis expansion to the hyperparameters, normalizes the features via `preprocessing.RobustScaler`, and finally predicts via a multilayer perceptron (MLP) with `hidden_layer_sizes=[128]`, `learning_rate="adaptive"`, `batch_size=256`, and `max_iter=4096`. We grid search `alpha` over 1e-4, 1e-3, 1e-2, 1e-1, 1e0 and `learning_rate_init` over 1e-5, 3e-5, 1e-4, 3e-4, 1e-3 using the train-validation split described previously. For the MLP, we fit the scaling law beforehand and then fit the MLP to the percent excess loss, rather than fitting them jointly.

B.4 Implementation

Our implementation further involves the following details.[5]

Training Pipeline. We use gradient accumulation so that we can tune the batch size despite different scales requiring different amounts of memory and GPU setups (1 vs. 8 GPUs). We set the per-GPU batch size to the largest power of two that safely avoids out-of-memory errors in order to maximize throughput while also evenly dividing the number of tokens (16 sequences for 4M–67M parameters, 8 sequences for 134M–268M parameters). Each scale is assigned a base number of gradient accumulation steps chosen so that after accounting for the per-GPU batch size and the number of GPUs the model will process 64 sequences per update. We then multiply that base by a power of two to achieve the distribution described in Figure 8 for the total batch size. As a result, the batch size sweep is hardware-agnostic and always evenly divides the total number of training tokens. In addition, we set `gc_collect_freq` to 1,024, `data.prefetch_size` to 1,024, `distributed.compile` to `true`, and `distributed.fsdp_type` to `"full_shard"` in order to make training efficient. Finally, we set `logging.freq` to 1 and found it necessary to set `env.ENABLE_INTRA_NODE_COMM` to `"0"` to avoid an unresolved error in the PyTorch version.

Software. We train the models using Meta Lingua (commit: 437d680), as previously described [77]. We modify the code base to support the 100B token subset of FineWeb-Edu, use a linear decay in the WSD schedule, and use PyTorch's Scaled Dot Product Attention (SDPA) instead of Flex Attention as Flex Attention would not successfully generate kernels for some of the transformer shapes that we randomly sample. In addition, we switch the transformer architecture from pre-norm to post-norm for some experiments as previously discussed. We isolate the software environment in an Apptainer (Singularity) container [20]. The container runs Ubuntu 24.04.2 LTS with CUDA 12.8 (V12.8.93) and cuDNN 9.8, and we run Lingua via a Python 3.11.14 conda environment with PyTorch 2.7.0+cu128 [61].

Hardware. Training was done on 80GB NVIDIA A100 GPUs on a SLURM cluster. Model sizes from 4M–134M effective parameters were trained with 1 GPU, while models with 268M effective parameters were trained data-parallel with 8 GPUs.

[5] Preliminary experiments were run on the NYU Torch cluster. We later expanded our random searches using Meta's compute infrastructure. This section describes the compute setup used at NYU, which is sufficient to reproduce our findings.

C Interpretable Models of the Hyperparameter Loss Surface

In §4.3, we discussed developing an interpretable model of the hyperparameter loss surface across scales by combining the noisy quadratic limit with scaling laws. We reported that such models capture the broad shape of the surface, but not precisely how it evolves. Here we provide those explorations in more depth. In particular, we will compare several approaches a researcher might be tempted to try and show that while they have some success in modeling the loss surface, they are far from the clear wins available via scaling laws or the noisy quadratic limit on their own.

Scaling laws model the optimal loss across scales; the noisy quadratic limit models how hyperparameters influence the loss at one scale. To combine them, we must set the quadratic's minimum according to the scaling law. One could accomplish this in several ways;[6] a simple approach with a strong intuition starts from the observation that, given any reasonable hyperparameters, the loss should approach the irreducible error (ϵ) as parameters and data go to infinity. We capture this by modeling the percent increase in *reducible* loss at each scale as a quadratic in the hyperparameters:

$$\mathcal{L}(p, d) \approx \epsilon + \left(\frac{\eta}{d^{\kappa}} + \frac{\zeta}{p^{\iota}}\right)\left(1 + L\mathbf{X} + (\mathbf{X} - \mathbf{x}_*)^T H_{\mathbf{x}_*} (\mathbf{X} - \mathbf{x}_*)\right) \tag{11}$$

Here, L represents residual linear terms necessary to make any quadratic expressible.[7] With this model, the scaling law supplies the optimal loss across scales while the quadratic inflates the reducible part according to how far the hyperparameters sit from the optimum.

Since the Hessian is real and symmetric, we can diagonalize it: $H_{\mathbf{x}_*} = UDU^T$. Let Λ be the diagonal matrix of non-zero eigenvalues and take $M_{\mathbf{x}_*}$ to be the submatrix of U^T that projects onto their corresponding eigenvectors. Letting $_* = M_{\mathbf{x}_*}\mathbf{x}_*$ represent the linear subspace of optimal hyperparameters, we can write this model as a quadratic in the (lower-dimensional) projected space:

$$\mathcal{L}(p, d) \approx \epsilon + \left(\frac{\eta}{d^{\kappa}} + \frac{\zeta}{p^{\iota}}\right)\left(1 + L\mathbf{X} + (M_{\mathbf{x}_*}\mathbf{X} - _*)^T \Lambda (M_{\mathbf{x}_*}\mathbf{X} - _*)\right) \tag{12}$$

This form enables us to fit the model with a low rank constraint.

We explored several variants of this model, in all cases restricting the Hessian's rank to at most $\gamma = 3$. The `static` model fixes both the optimum and the curvature across scales as in Equation 12. The `dynamic optima` model keeps the curvature fixed but makes the (latent) optimal hyperparameters a linear function of the log scale: $_* = _0 + \log(p)_p + \log(d)_d$. The `dynamic sensitivity` model instead fixes the optimum and lets the curvature vary, making the eigenvalues of the Hessian power laws of the scale: $\lambda_i p^{\nu_i} d^{\xi_i}$. The `dynamic` model varies both. For comparison, we include a `polynomial` model which predicts the percent increase in excess loss using a quadratic polynomial (with no constraint on the rank). We also compare against a `multilayer perceptron` model that abandons interpretability altogether and predicts percent increase in excess loss as a nonlinear function (neural network) of the hyperparameters and scale.

Table 2 compares these variants, fit on the small scales and evaluated on the larger held-out ones via mean-squared error (MSE). Ignoring the scale's effect on the hyperparameters (`static`) already performs surprisingly well. The dynamic variants extrapolate with mixed success: varying the optimum appears to help, while varying the curvature seems to hurt. The nonlinear model is the most flexible; here it performs the worst of all models that incorporate the hyperparameters, though it can also perform the best in similar setups depending on optimization details like the initialization. While every variant beats the baseline of ignoring the hyperparameters and predicting from the parameters and tokens alone, none fully captures the loss surface. If one did, it would approach the much smaller error achieved by the scaling law when predicting the optimal hyperparameters.

[6] Beyond the approach discussed here, we also tried using the scaling law for the quadratic's constant term which is perhaps the most obvious approach even if it does not have an a priori justification.

[7] One might expect $L = \mathbf{0}$ at the optimum. While we do not enforce this assumption in our experiments, we explored it and the conclusions are similar.

	Validation (MSE)		Test (MSE)
	67M	134M	268M
Scaling Law	0.000011	0.000025	0.000020
Parameters and Tokens Only	0.011471	0.014014	0.019795
Polynomial	0.004452	0.005217	0.008790
Static	0.004820	0.005900	0.010027
Dynamic Optima	*0.002824*	*0.004401*	*0.006630*
Dynamic Sensitivity	0.005413	0.007185	0.010813
Dynamic	0.003672	0.006806	0.010102
Multilayer Perceptron	0.008866	0.012003	0.015464

Table 2: *The mean-squared error (MSE) of different models for the hyperparameter loss surface extrapolating from small to large scales using our model ladder and decaying the learning rate.* The scaling law predicts loss at the optimal hyperparameters; the rest predict all configurations in the asymptotic regime. Each column's best loss is italicized (excluding the scaling law). The static quadratic that ignores scale is already a strong baseline; varying the optimum appears to help while varying the curvature seems to hurt. The `dynamic optima` model performs best here; however, the models' ranking is sensitive to optimization details like the initialization. Every variant improves over predicting from parameters and tokens alone (ignoring the hyperparameters), but none reaches the error attained by the scaling law when predicting the loss at the optimal hyperparameters; in other words, no model fully captures the surface.

D Additional Theoretical Results

D.1 How many hyperparameter runs are needed?

How many hyperparameter runs are needed to approximate the optimal perplexity at a given scale? Or in other words, how quickly does the sample minimum of the noisy quadratic converge to the best attainable performance α? We will see that the answer depends on the intrinsic dimension γ of the hyperparameter space.

Given the noisy quadratic limit (see Section A.2), whose notation (α, β, γ, σ, and the CDF F) we adopt throughout, we can quantify how close random search gets to the best attainable performance α after M evaluations, i.e. how many hyperparameter runs get us a good estimate for the best attainable performance at a given scale. Let $Y_1, \ldots, Y_M$ be i.i.d. scores drawn within the asymptotic regime so each Y_i follows the noisy quadratic distribution of Equation 6, and let $T_M = \min_{i \le M} Y_i$ be the best score found.

First, we find the CDF of T_M, by reasoning about the complement: the best score exceeds a threshold y only when *every* draw does, and a single draw exceeds y with probability $\mathbb{P}(Y_i > y) = 1 - F(y)$, and because we draw independent samples,

$$\mathbb{P}(T_M > y) = \prod_{i=1}^{M} \mathbb{P}(Y_i > y) = \prod_{i=1}^{M} \underbrace{\big(1 - F(y)\big)}_{=\mathbb{P}(Y_i > y)} = \big(1 - F(y)\big)^M. \tag{13}$$

Taking the complement gives the CDF of the minimum T_M,

$$\mathbb{P}(T_M \le y) = 1 - \big(1 - F(y)\big)^M. \tag{14}$$

Second, we determine how T_M approaches the optimum α as M grows by consulting Extreme Value Theory (EVT)[25; 17] [8]. We will see there are two regimes: the noiseless case ($\sigma = 0$) and the noisy case ($\sigma > 0$) for which we will come to different conclusions. Consider the noiseless case ($\sigma = 0$), where α is the lower endpoint of the support and the tail is the power law of Equation 7. A distribution with a finite endpoint and a polynomially decaying tail lies in the *Weibull domain of attraction* of EVT: its sample minima, once recentered at the endpoint and rescaled, converge to a Weibull law. Concretely, take the natural scale $a_M := (\beta - \alpha)\, M^{-2/\gamma}$, defined implicitly by $F(\alpha + a_M) = 1/M$. Substituting $u = a_M t$ yields $\big(1 - F(\alpha + a_M t)\big)^M \to \exp(-t^{\gamma/2})$, so the rescaled gap converges in distribution to

$$\frac{T_M - \alpha}{a_M} \xrightarrow{d} \mathrm{Weibull}(k), \qquad k = \tfrac{\gamma}{2}, \tag{15}$$

where $\mathrm{Weibull}(k)$ is the Weibull distribution of shape k and unit scale, see Figure 10 for an illustration.

Third, as the limit in Equation 15 is a convergence in distribution, it does not by itself imply convergence of its mean $E[(T_M - \alpha)/a_M]$. For this, we have to additionally show that the rescaled minimum is *uniformly integrable*, which ensures that the mean exists. To see this condition is fulfilled, note that the upper tail is bounded by $\exp(-t^{\gamma/2})$. Thus, the rescaled minimum is uniformly integrable and the mean converges to that of the limiting Weibull distribution;

$$\mathbb{E}\left[\frac{T_M - \alpha}{a_M}\right] \longrightarrow \mathbb{E}\big[\mathrm{Weibull}(k)\big] = \Gamma\big(1 + \tfrac{1}{k}\big)\,, \tag{16}$$

where Γ is the gamma function [38].

Finally, we have all the pieces to show how quickly the gap between the observed and optimal minima, $\mathbb{E}[T_M] - \alpha$, closes as the number of samples M increases. Its dependence on M follows from Equation 15 by multiplying through by the scale $a_M = (\beta - \alpha)\, M^{-2/\gamma}$ and setting $k = \gamma/2$,

$$\mathbb{E}[T_M] - \alpha \;\sim\; a_M\, \Gamma\big(1 + \tfrac{1}{k}\big) \;=\; \underbrace{(\beta - \alpha)\, \Gamma\Big(1 + \tfrac{2}{\gamma}\Big)}_{=:C(\gamma)}\; M^{-2/\gamma}. \tag{17}$$

[8] Extreme-value theory is usually stated for maxima; the result for minima follows by applying it to $-Y$.

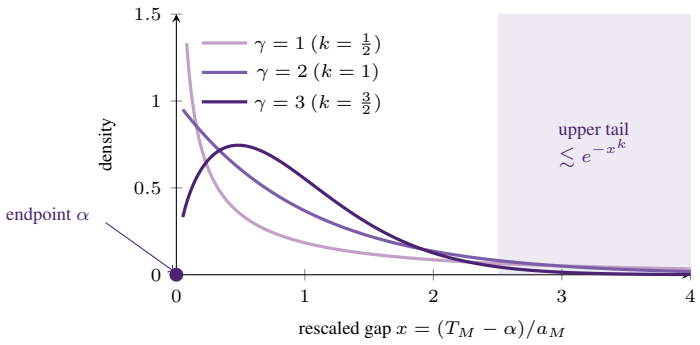


Figure 10: Weibull(k) densities $f(x) = k\,x^{k-1}e^{-x^k}$ (and CDF $F(x) = 1 - e^{-x^k}$). The shape $k = \gamma/2$ governs how quickly probability accumulates away from the endpoint α: a small effective dimension ($\gamma = 1$) piles mass near the optimum, while larger γ pushes the typical gap outward. In the shaded *upper tail* every density decays at least as fast as e^{-x^k}, so a vanishing fraction of both the mass and the mean sits out there, which illustrates why this function is uniformly integrable, meaning that the mean exists.

Here and below, $\sim$ denotes *asymptotic equivalence*: $f(M) \sim g(M)$ means $f(M)/g(M) \to 1$ as $M \to \infty$, not proportionality or "distributed as". For the practically relevant cases $\gamma \in \{1, 2\}$ the mean can even be evaluated exactly,[9] giving the main result of this section: left=8pt, right=8pt, top=4pt, bottom=4pt]

$$\mathbb{E}[T_M] - \alpha \begin{cases} \sim\ C(\gamma)\, M^{-2/\gamma}, & \text{for all } \gamma, \\ = \dfrac{2(\beta-\alpha)}{(M+1)(M+2)}\ \sim\ 2(\beta-\alpha)\, M^{-2}, & \text{for } \gamma = 1, \\ = \dfrac{\beta-\alpha}{M+1}\ \sim\ (\beta-\alpha)\, M^{-1}, & \text{for } \gamma = 2. \end{cases} \tag{19}$$

Thus the best score approaches α as a power law in the number of evaluations, $M^{-2/\gamma}$, with the exponent set by the effective number of hyperparameters. Each additional effective hyperparameter flattens the convergence exponent.

As a concrete example, say we ran 100 hyperparameter evaluations at a scale where $\gamma = 2$. Assuming all other scales have the same α and β, we can compare how many runs would be needed at other scales where $\gamma = 1, 3$ to reach the same quality, i.e. the same gap. Matching the gaps in Equation 19, i.e. $C(\gamma)\, M_\gamma^{-2/\gamma} = C(2)\, 100^{-1}$, gives $M_\gamma = \big[\,100\, C(\gamma)/C(2)\,\big]^{\gamma/2}$. Since $C(1) = 2(\beta - \alpha)$, $C(2) = (\beta - \alpha)$, and $C(3) \approx 0.903\,(\beta - \alpha)$, the constant ratios are $C(1)/C(2) = 2$ and $C(3)/C(2) \approx 0.903$, so

$$M_1 = (100 \cdot 2)^{1/2} \approx 14 \quad \text{and} \quad M_3 = (100 \cdot 0.903)^{3/2} \approx 858.$$

So dropping from two effective hyperparameters to one cuts the tuning cost from 100 runs to about 14 (a 7× saving), while adding one raises it to nearly 900 (a further 8.6×).

Note that we assumed $\sigma = 0$ for our derivation, when $\sigma > 0$, the unbounded Gaussian tail means we can no longer rely on the same analysis as the EVT is now in the heavy tail regime.

[9]The "$\sim$" in Equation 17 is asymptotic: it is the $M \to \infty$ limit of the Weibull law Equation 15, whose mean is the standard $\Gamma(1 + 1/k)$. For the noiseless quadratic we can go further and evaluate the mean exactly at finite M. Using the tail identity $\mathbb{E}[Z] = \int_0^\infty \mathbb{P}(Z > u)\, \mathrm{u}$ for the nonnegative $Z = T_M - \alpha$ together with the survival function Equation 13, the gap is a Beta integral,

$$\mathbb{E}[T_M] - \alpha = \int_0^{\beta-\alpha} \big(1 - F(\alpha + u)\big)^M\, \mathrm{u} = (\beta - \alpha)\, \Gamma\Big(1 + \tfrac{2}{\gamma}\Big)\, \frac{\Gamma(M+1)}{\Gamma(M + 1 + 2/\gamma)}. \tag{18}$$

The ratio of the two gamma functions decays like $M^{-2/\gamma}$ as $M \to \infty$, recovering Equation 17. Since the number of runs M is an integer, $\Gamma(M+1) = M!$, and whenever $2/\gamma$ is a positive integer the ratio collapses to a rational function of M, giving the exact expressions in Equation 19.

D.2 SCALING LAW ERROR PROPAGATION

Scaling laws are fitted to losses measured at a handful of scales and then extrapolated to a larger ones. Each measured loss, however, carries error from a finite tuning budget (section D.1), random seeds, or estimation noise. If the irreducible error were known, this problem would reduce to fitting a power law: essentially a linear regression in log-log space. In practice, the irreducible error is not known, but the fitting procedure can behave quite similarly. We ask a simple question: if the loss at one scale is off by ϵ, how much does that corrupt the loss we *extrapolate* to at a larger scale? For the linear case, the answer is exact, closed form and can be retrieved from standard literature on linear regression.

Throughout, we index the K *scales* by $j = 1, \ldots, K$; each scale contributes one data point to the regression. We reserve n (and M) for the number of *samples* used to *measure* the loss at a given scale, for instance, the best-of-M score T_M from a tuning run (section D.1), so that the measurement error ϵ at a scale can itself be understood through the sampling analysis of the previous section. Figure 12 illustrates the whole construction.

Scaling law fit as log-log regression. Assume a power-law scaling law, $L = A\,x^{-b}$, where x is the scale (compute, parameters, or data) and L the loss. Taking logarithms linearizes it,

$$y = \beta_0 + \beta_1 u, \qquad u := \log x, \quad y := \log L, \tag{20}$$

with $\beta_0 = \log A$ and slope $\beta_1 = -b$. Given measurements $\{(u_j, y_j)\}_{j=1}^{K}$ at the K scales, we estimate (β_0, β_1) by ordinary least squares [8; 56].

Extrapolation as a weighted sum of the data. Ordinary least squares is linear in the responses, so both the fitted coefficients and any prediction are weighted sums of the observed y_j [8; 56]. Writing $\bar{u} = \frac{1}{K}\sum_k u_k$ and $S_{uu} = \sum_k (u_k - \bar{u})^2$, the slope estimate is

$$\hat{\beta}_1 = \sum_{j=1}^{K} w_j\, y_j, \qquad w_j = \frac{u_j - \bar{u}}{S_{uu}}, \tag{21}$$

and the intercept is $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{u}$. Substituting into Equation 20, the predicted log-loss at any scale $u_\star$ — in particular one we extrapolate to — is itself a weighted sum,

$$\hat{y}(u_\star) = \sum_{j=1}^{K} h_j(u_\star)\, y_j, \qquad h_j(u_\star) = \frac{1}{K} + \frac{(u_\star - \bar{u})(u_j - \bar{u})}{S_{uu}}. \tag{22}$$

The weight $h_j(u_\star)$ is the sensitivity of the extrapolated prediction to the j-th measurement, we will refer to it as the *leverage* of point j [28]. $|h_j(u_\star)|$ is large when j is far from the center. We illustrate this in Figure 11. Note that, because $u_\star$ lies outside the data range $|u_\star - \bar{u}|$ is large which is why extrapolated predictions are so sensitive to the measurements with the highest leverage.

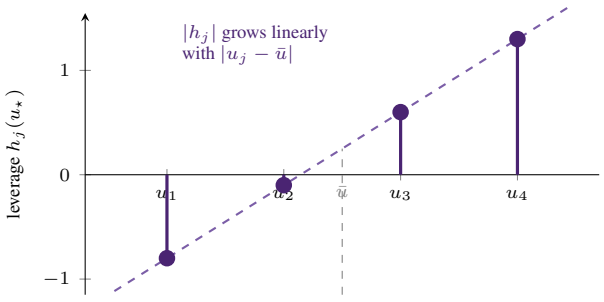


Figure 11: Leverage of each measurement. The leverage $h_j(u_\star)$ (Equation 22) measures how strongly each measurement steers the extrapolated prediction. Its magnitude grows linearly with the distance $|u_j - \bar{u}|$ from the centroid, and its sign flips across $\bar{u}$, so the endpoints carry the most weight.

Propagating errors. Suppose the measurement at scale j is perturbed, $y_j \mapsto y_j + \epsilon$, while the others are exact. Because Equation 22 is linear, the prediction shifts by exactly

$$\delta\hat{y}(u_\star) = h_j(u_\star)\,\epsilon. \tag{23}$$

As is standard for scaling laws, take the scales equally spaced in log space, $u_j = u_1 + (j-1)\Delta$. Then $u_j - \bar{u} = (j - \bar{\jmath})\Delta$ with center $\bar{\jmath} = \frac{K+1}{2}$, and $S_{uu} = \Delta^2 \sum_k (k - \bar{\jmath})^2 = \Delta^2 \frac{K(K^2-1)}{12}$.

Extrapolating from the corrupted scale j to a scale i steps further, $u_\star = u_{j+i}$, the spacing Δ cancels and Equation 23 becomes

$$\delta\hat{y}_{j+i} = \epsilon \left[\frac{1}{K} + \frac{12\,(j-\bar{j})(j+i-\bar{j})}{K(K^2-1)} \right]. \tag{24}$$

Note that, an error ϵ at scale j shifts the extrapolated log-loss at scale $j+i$ by a factor that depends on how many scales j is from the center $\bar{j}$ and how far the extrapolation i is. Errors at the endpoints of the fitting range are the most damaging, while an error at the center ($j = \bar{j}$) contributes only the $1/K$ intercept shift and is not amplified by extrapolation.

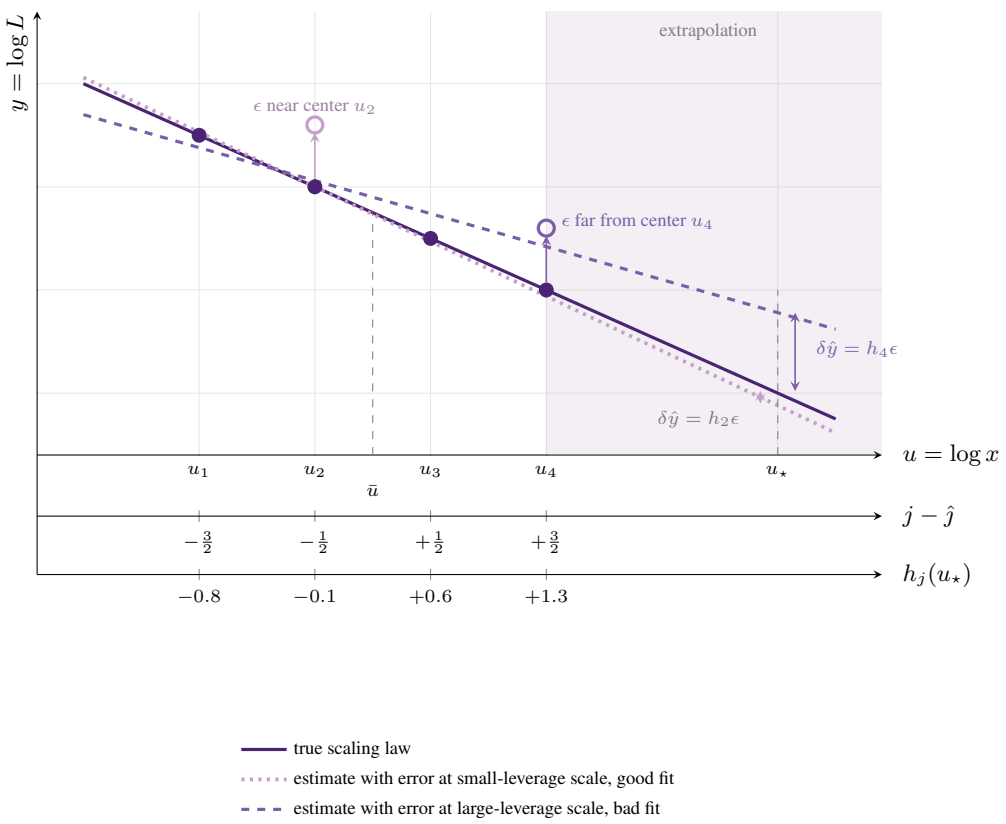


Figure 12: Consequence for extrapolation. Losses are measured at K equally spaced scales $u_1, \ldots, u_K$ (centroid $\bar{u}$) and fit by ordinary least squares (solid; Equation (20) and 21). Perturbing a measurement by the same ϵ shifts the extrapolated prediction at $u_\star$ by $\delta\hat{y} = h_j(u_\star)\,\epsilon$ (Equation 23): a high-leverage point far from the center (u_4, dashed) *amplifies* the error, while a low-leverage point near the center (u_2, dotted) barely moves it. Because $u_\star$ lies far outside the data range (shaded), it amplifies the effect of the leverage further ($h_4(u_\star) > 1$ here). The lower axes read each measurement's centered index $j - \hat{j}$ and its leverage $h_j(u_\star)$.

D.3 Compute-optimal tuning across scales

In this section we will combine the results from the previous two sections to answer the question: Given a fixed compute budget, how should we split it across scales to minimize the error at the scale we extrapolate to?

For our analysis, we make the following assumptions,

1. **Tuning error per scale.** At scale j we run random search with M_j configurations and record the best. Its expected gap to the optimum is $\epsilon_j = A_j\, M_j^{-2/\gamma_j}$ with $A_j = (\beta_j - \alpha_j)\,\Gamma(1+2/\gamma_j)$ (Equation 17), where γ_j is the effective number of hyperparameters at that scale.

2. **Cost.** Compute per run scales linearly with the parameter count, so one run at scale j costs $\propto N_j$. The scales are geometric, $N_j = N_1 \rho^{j-1}$ (equidistant in $\log N$), and the total budget is $C = \sum_{j=1}^{K} M_j N_j$.
3. **Propagation.** An error ϵ_j shifts the extrapolated prediction by $h_j\,\epsilon_j$ (Equation 23), where $h_j = h_j(u_\star)$ is the leverage of scale j at the target $u_\star$. We minimize the propagated-error bound $E_\star = \sum_j |h_j|\,\epsilon_j$.[10]
4. **Effective dimension.** Empirically $\gamma_j \in \{1, 2\}$, and it *decreases* with scale: the low-compute scales have $\gamma_j = 2$, and beyond some scale the effective dimension drops to $\gamma_j = 1$.

The optimization problem. Substituting the tuning error $\epsilon_j = A_j M_j^{-2/\gamma_j}$ into the propagated-error bound, the quantity we want to minimize with respect to the number of runs M_j is

$$E_\star(M_1, \ldots, M_K) = \sum_{j=1}^{K} |h_j|\,\epsilon_j = \sum_{j=1}^{K} |h_j|\,A_j\,M_j^{-2/\gamma_j}, \tag{25}$$

however this error minimization is subject to the compute budget constraint Equation 26;

$$\sum_{j=1}^{K} M_j N_j = C. \tag{26}$$

We attach the constraint with a Lagrangian multiplier λ and minimize the loss

$$\mathcal{L}(M_1, \ldots, M_K, \lambda) = \sum_{j=1}^{K} |h_j|\,A_j\,M_j^{-2/\gamma_j} + \lambda \left(\sum_{j=1}^{K} M_j N_j - C \right). \tag{27}$$

Differentiating with respect to a single M_j gives ,

$$\frac{\partial \mathcal{L}}{\partial M_j} = -\frac{2}{\gamma_j}\,|h_j|\,A_j\,M_j^{-2/\gamma_j - 1} + \lambda N_j = 0. \tag{28}$$

Rearranged for λ;

$$\underbrace{\frac{1}{N_j} \cdot \frac{2}{\gamma_j}\,|h_j|\,A_j\,M_j^{-2/\gamma_j - 1}}_{\text{error reduction per unit compute at scale } j} = \lambda \quad \text{for every active scale } j. \tag{29}$$

Solve for M_j in Equation 28,

$$M_j^{2/\gamma_j + 1} = \frac{2\,|h_j|\,A_j}{\gamma_j\,\lambda\,N_j}. \tag{30}$$

The exponent on the left combines as $\dfrac{2}{\gamma_j} + 1 = \dfrac{\gamma_j + 2}{\gamma_j}$, so raising both sides to the reciprocal power $\dfrac{\gamma_j}{\gamma_j + 2}$ gives the optimal number of runs,

$$M_j^\star = \left(\frac{2\,|h_j|\,A_j}{\gamma_j\,\lambda\,N_j} \right)^{\frac{\gamma_j}{\gamma_j + 2}} \propto \left(\frac{|h_j|}{\gamma_j\,N_j} \right)^{\frac{\gamma_j}{\gamma_j + 2}}. \tag{31}$$

Example. To make the allocation concrete, take $K = 4$ equally spaced scales and extrapolate one step beyond the fitted range, $u_\star = u_{K+1}$ (i.e. $i = 1$ in Equation 24). With centre $\bar{\jmath} = \frac{K+1}{2} = \frac{5}{2}$, the target sits at $u_\star - \bar{u} = \bar{\jmath}\,\Delta$ and the index form of the leverage collapses to a single line,

$$h_j = \frac{1}{K} + \frac{12\,\bar{\jmath}\,(j - \bar{\jmath})}{K(K^2 - 1)} = \frac{j - 2}{2}, \qquad \text{so} \qquad \big(|h_1|, |h_2|, |h_3|, |h_4|\big) = \big(\tfrac{1}{2},\ 0,\ \tfrac{1}{2},\ 1\big). \tag{32}$$

[10] Using the triangle inequality on $\sum_j h_j \epsilon_j$ gives a sign-robust objective. Minimizing the variance proxy $\sum_j h_j^2 \epsilon_j^2$ instead replaces the exponent $\gamma_j/(\gamma_j+2)$ below by $\gamma_j/(\gamma_j+4)$ and leaves the qualitative allocation unchanged.

The endpoint nearest the target ($j = 4$) carries the most leverage, while the near-central scale ($j = 2$) carries essentially none. Following the fourth assumption, the effective dimension drops across the centroid: $\gamma_j = 2$ for the small scales below the centre ($j = 1, 2$) and $\gamma_j = 1$ for the large scales above it ($j = 3, 4$). The per-run costs are geometric in the index, $N_j = N_1\,\rho^{j-1}$, i.e. linear in log space, $\log N_j = \log N_1 + (j-1)\log\rho$.

Substituting into Equation 31 and reading off the exponent $\frac{\gamma_j}{\gamma_j+2}$ ($= \frac{1}{2}$ where $\gamma_j = 2$, $= \frac{1}{3}$ where $\gamma_j = 1$), the optimal number of runs at each scale is

$$M_j^\star \;\propto\; \left(\frac{|h_j|}{\gamma_j\,N_j}\right)^{\frac{\gamma_j}{\gamma_j+2}} = \begin{cases} \big(\,|h_j|/(2\,N_1\rho^{j-1})\,\big)^{1/2}, & j \le 2 \quad (\gamma_j = 2), \\ \big(\,|h_j|/(N_1\rho^{j-1})\,\big)^{1/3}, & j \ge 3 \quad (\gamma_j = 1). \end{cases} \tag{33}$$

Multiplying by N_j gives the compute placed at each scale,

$$C_j^\star = N_j\,M_j^\star \;\propto\; \left(\frac{|h_j|}{\gamma_j}\right)^{\frac{\gamma_j}{\gamma_j+2}} N_j^{\frac{2}{\gamma_j+2}}, \tag{34}$$

whose exponent on N_j ($\frac{1}{2}$ then $\frac{2}{3}$) is below one, so compute grows *sublinearly* with a scale's per-run cost. Within the large-scale regime the two upper scales are tied together cleanly,

$$\frac{M_4^\star}{M_3^\star} = \left(\frac{|h_4|}{|h_3|}\cdot\frac{N_3}{N_4}\right)^{1/3} = \left(\frac{2}{\rho}\right)^{1/3}, \qquad \frac{C_4^\star}{C_3^\star} = 2^{1/3}\,\rho^{2/3}. \tag{35}$$

For any $\rho > 2$ the larger scale receives *fewer* runs yet *more* compute than its neighbour: a handful of expensive runs at the top outweighs many cheap ones below. Figure 13 plots the allocation for $\rho = 4$ ($N_j = 1, 4, 16, 64$): the number of runs is largest at the cheap bottom scale and tapers off, while the compute share climbs monotonically and piles up at the top scale. The near-central scale $j = 2$, with $h_2 \approx 0$, drops out entirely and is kept only by its single anchoring run.

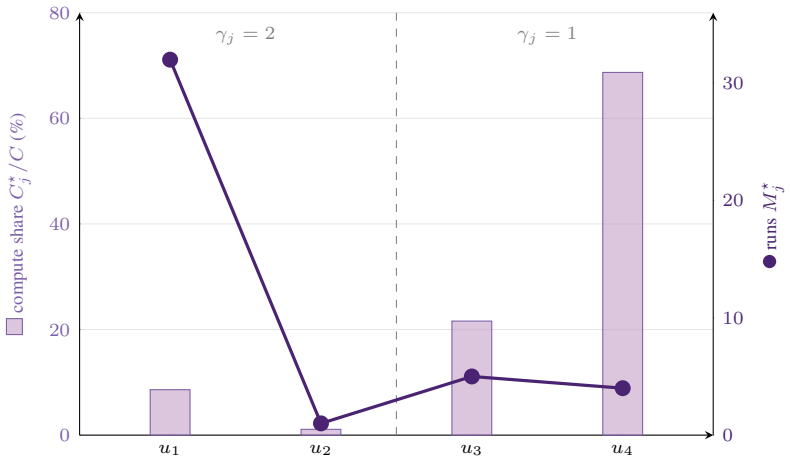


Figure 13: Compute-optimal allocation across $K = 4$ scales for the example of Equation 32 and 33 ($\rho = 4$, extrapolating one step past the range). Bars (left axis) show the share of the budget spent at each scale $C_j^\star/C$; the line (right axis) shows the number of runs $M_j^\star$. Cheap low scales absorb many runs but little compute; the budget concentrates on the high-leverage, low-γ scale nearest the target (u_4), which is bought with only a few expensive runs. The near-central scale u_2 has $h_2 \approx 0$ and is kept only to anchor the fit.

A manual for designing a compute-optimal scaling law estimate. Any practitioner should roughly follow the following steps in designing their experiment:

1. *Estimate (or set)* ρ, the relative cost of a scale $N_j = N_1\,\rho^{j-1}$. For transformers this cost is well approximated by the standard $6ND$ estimate (roughly six FLOPs per parameter per

token over the forward and backward pass) [41; 33], so once a practitioner chooses a model ladder and its token counts, every N_j, and hence ρ, is fixed. (Anyone building their own ladder has to do this FLOP accounting up front to know what a run at each scale will cost.)

2. *Assume the effective dimensions γ_j.* We take these as known. This is not a perfect assumption, but it is broadly understood where along a model ladder tuning becomes reliable and γ drops, and the choice can be checked after the fact from the realized slope of error against compute at each scale.
3. *Compute the total budget C.* This is normally known outright usually in GPU-hours, converted to FLOPs through the hardware's sustained throughput. The only remaining factor, the per-scale error amplitude A_j, varies slowly and is typically taken comparable across scales (or read off the loss spread if not).

With ρ, the γ_j, and C in hand, the number of runs per scale can be computed. The multiplier λ is computed through Equation 31, and it is set by substituting the $M_j^\star$ into the budget constraint $\sum_j M_j N_j = C$ (Equation 26) and solving the resulting equation for λ. Reading the $M_j^\star$ back out of Equation 31 then gives the number of tuning runs to launch at each scale.

E ADDITIONAL EMPIRICAL RESULTS

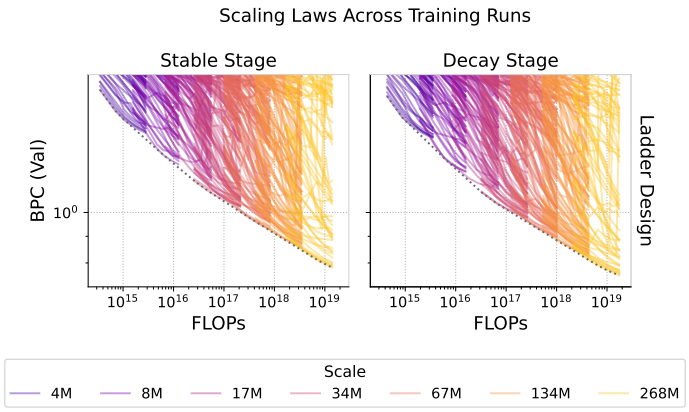


Figure 14: *Scaling laws emerge at very small scales, smaller than widely believed; however, hyperparameter tuning becomes critical.* Predictable scaling appears only at the fully tuned frontier. Smaller models are harder to tune, making their scaling laws harder to observe though no less present.

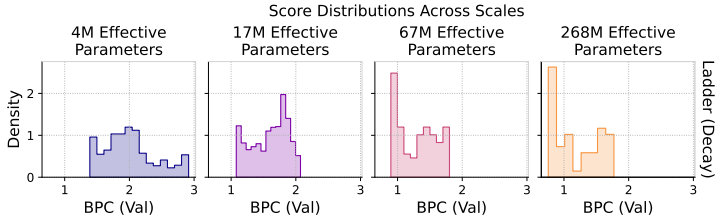


Figure 15: *As scale increases, random configurations are more likely to land near the optimum.* We measure the volume of near-optimal configurations by visualizing the distribution over validation scores at the end of training from random search (the best 85% to avoid divergent runs). As scale grows, the distribution concentrates near the best achievable loss at each scale, with probability mass

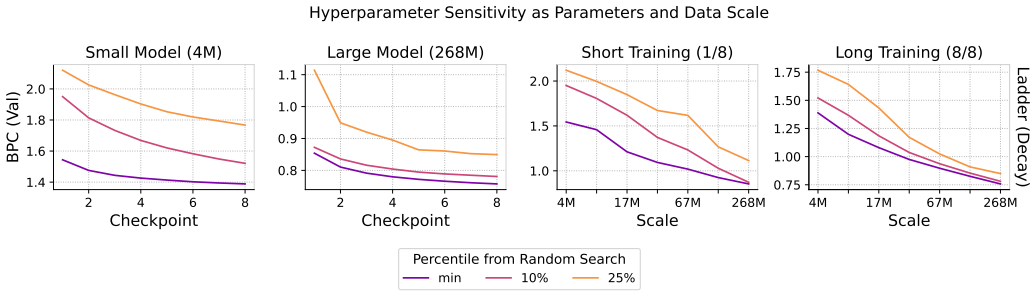


Figure 16: *Hyperparameter sensitivity diminishes through increasing parameters and data together.* We plot the minimum, 10th, and 25th percentile validation losses from random search. The gap from the minimum measures sensitivity. Scaling data with a small model (left) or parameters with short training (middle right) lowers the loss but only slightly cuts the gap. Scaling data with a large model (middle left) or parameters with longer training (right) collapses the gap.

F CASE STUDY DIAGNOSTICS

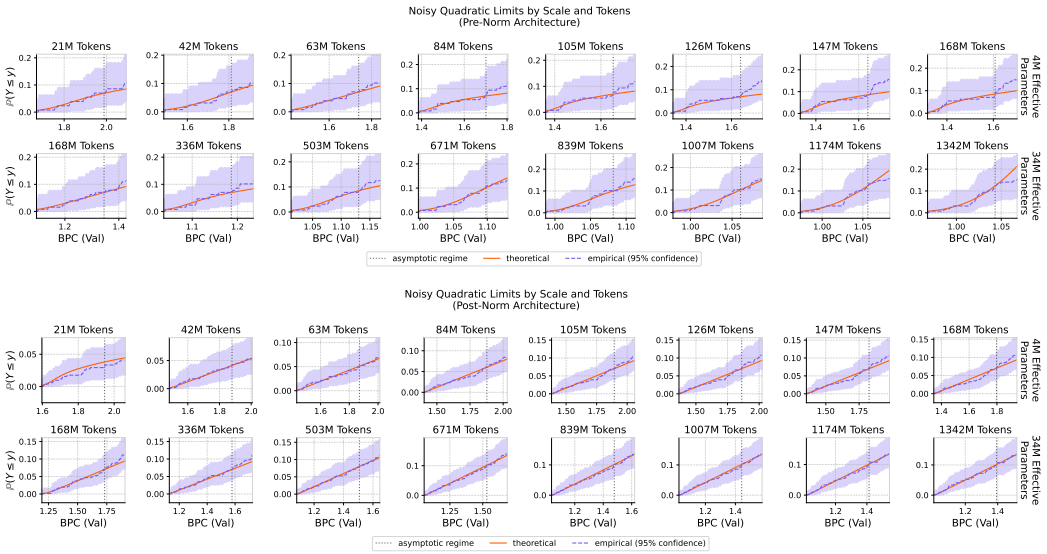


Figure 17: *The noisy quadratic limit emerges for both architectures, confirming the hyperparameters are well-tuned.* We fit the noisy quadratic to the distribution of validation losses from random search at 4M and 34M parameters for the pre-norm (top) and post-norm (bottom) architectures. The asymptotic regime is smaller for post-norm than pre-norm, an early sign of its hyperparameter sensitivity.

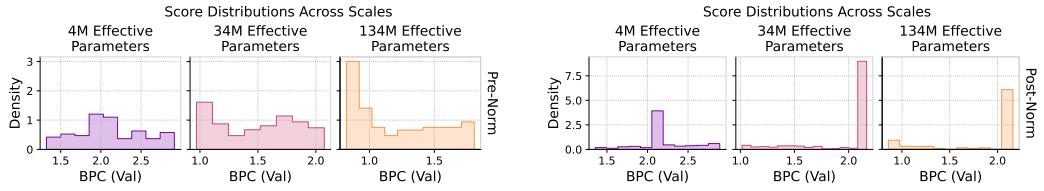


Figure 18: *Hyperparameter sensitivity decreases with scale for both the pre-norm and post-norm architectures.* Across scales, we compare the score distributions (best 85%, to exclude divergent runs) for the pre-norm (left) and post-norm (right) architectures. Both place more probability mass near the optimum as scale increases; post-norm, however, maintains a large peak at a suboptimal value, showing persistent tuning difficulty.

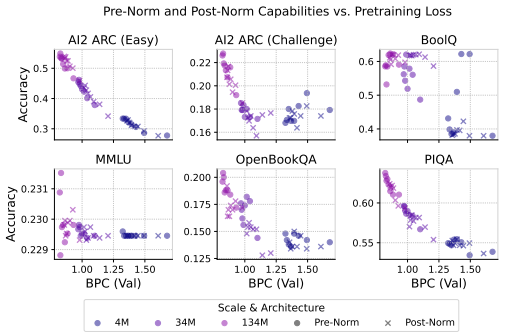


Figure 19: *Pretraining loss corresponds to capabilities both within and across pre-norm and post-norm architectures.* Each panel plots pretraining loss against accuracy for a different task. Across scales and architectures, all points fall on a common trend: equal loss implies equal capability.

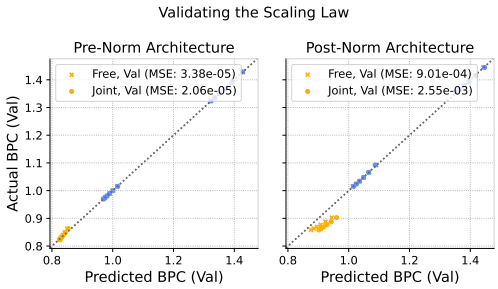


Figure 20: *The scaling laws extrapolate comparably well with free vs. joint irreducible error terms.* We compare sharing the irreducible error term across the architectures (`joint`) vs. allowing them to differ (`free`). The pre-norm scaling law extrapolates very well in both cases. The post-norm law extrapolates somewhat, but could likely benefit from even further hyperparameter tuning.

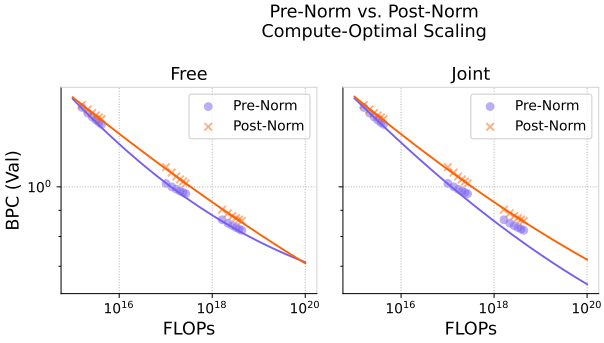


Figure 21: *Pre-norm scales better than post-norm near the data; far from it, extrapolation reflects the irreducible error.* We fit the scaling laws to all available data and compare them under two conditions: separate (left) vs. shared (right) irreducible errors. Across the range we explore, pre-norm scales better. Outside that range, pre-norm continues to scale better if you believe that both architectures saturate at the same final loss. Under this assumption, the gap grows with compute and we recover the large-scale advantage of pre-norm from our small-scale experiments.