Scaling Collapse Reveals Universal Dynamics in Compute-Optimally Trained Neural Networks

Shikai Qiu [1 †] **Lechao Xiao** [2] **Andrew Gordon Wilson** [1] **Jeffrey Pennington** [2] **Atish Agarwala** [2]

Abstract

Understanding neural network training dynamics at scale is an important open problem. Although realistic model architectures, optimizers, and data interact in complex ways that make predictive theory challenging, we show that compute-optimally trained models exhibit remarkably precise collective regularities. Specifically, loss curves from models of varying sizes collapse onto a single universal curve when training compute and loss are normalized to unity at the end of training. With learning rate decay, discrepancies between normalized curves fall below the noise floor of individual models' loss curves across random seeds, yielding an exceptionally tight collapse we term "*supercollapse*." We observe supercollapse across learning rate schedules, datasets, and architectures, including transformers trained on next-token prediction. This collapse breaks down when hyperparameters are scaled suboptimally, providing a practical indicator of proper scaling. We explain these phenomena by connecting collapse to the power-law structure in typical neural scaling laws, and analyzing a simple but effective model of SGD noise dynamics that accurately captures how learning rate schedules deform loss curves away from power laws while preserving universality, and why learning rate decay suppresses variance to enable supercollapse.

1. Introduction

As machine-learning systems grow in parameters, data, and compute, accurate predictive models of their training dynamics become increasingly valuable—both for interpreting costly experiments and for designing robust, efficient training pipelines (Wortsman et al., 2023; Achiam et al., 2023; Xiao, 2024). Yet the sheer complexity of modern architectures, optimizers, and datasets often renders exact, first-principles analyses intractable.

Nevertheless, recent work shows that key aspects of training become predictable when we shift focus from individual models to their collective scaling behaviour. Examples include power-law relations among optimal loss, model size, dataset size, and compute budget (Kaplan et al., 2020; Sharma & Kaplan, 2022; Hoffmann et al., 2022), as well as hyperparameter transfer from small to large models (Yang et al., 2021; Bordelon et al., 2023; Everett et al., 2024; Bordelon et al., 2024c).

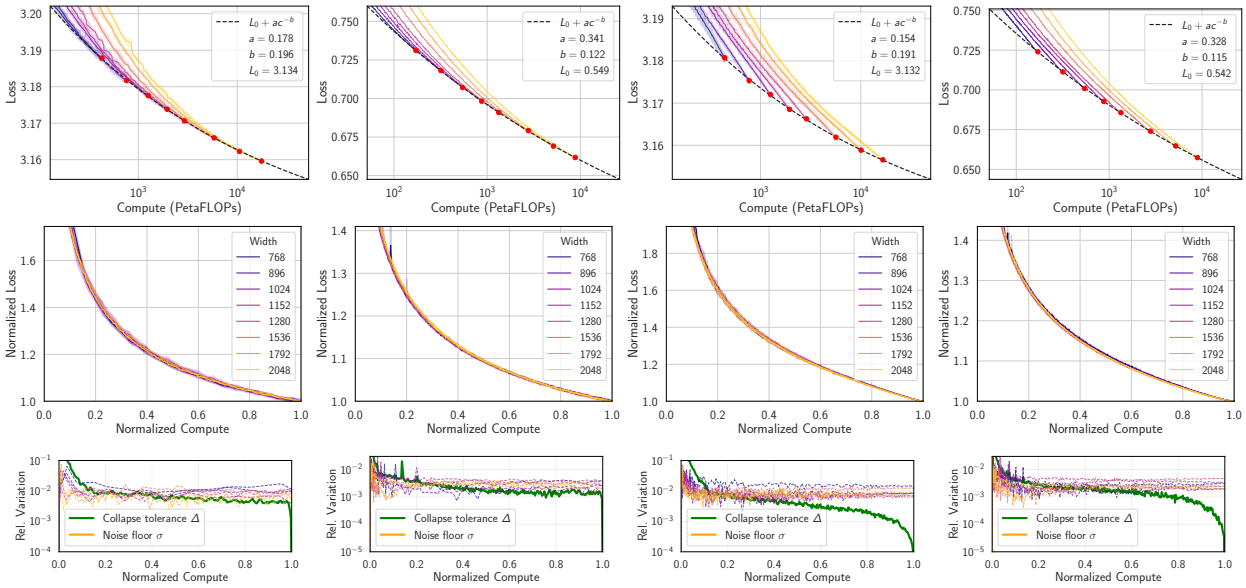
In this work, we push this line of inquiry further, showing that the loss curve throughout training — not just the final loss — follows remarkably precise scaling. Across architectures, datasets, and learning rate schedules, we show that compute-optimal models of different sizes have loss curves that collapse onto a single universal curve after a simple normalization. Learning rate decay amplifies this effect dramatically, producing what we call *supercollapse*: collapse so tight that cross-scale differences fall below the noise floor of individual models' loss due to random seeds. Beyond its theoretical interest, supercollapse provides a practical scaling diagnostic, as we find that deviations from collapse can signal misconfigured scaling choices, from improper learning rate scaling to suboptimal training horizons.

Our theoretical analysis reveals the key mechanisms behind this precise collapse. We first prove that for loss curves following typical neural scaling laws (sums of power laws), collapse occurs precisely when models are trained for constant multiples of their compute-optimal horizons. We then analyze a simple theoretical model of the SGD noise dynamics that predicts how learning rate schedules transform these curves, which explains two key observations: why normalized curves retain universal form despite losing their power-law structure, and how learning rate decay suppresses variance to produce supercollapse.

We make our code available here.

[†]Work done partly during an internship at Google DeepMind [1]New York University [2]Google DeepMind. Correspondence to: Shikai Qiu <sq2129@nyu.edu>, Atish Agarwala <thetish@google.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.



(a) CIFAR-5M (b) Chess (c) CIFAR-5M (LR decay) (d) Chess (LR decay)

Figure 1: **Scaling collapse of compute-optimal loss curves for transformers on next-token prediction tasks. (Top)** Compute-optimal loss curves across different model sizes with fitted scaling laws for each task. **(Middle)** Normalized reducible loss curves, where both final compute and reducible loss are normalized to unity. These curves collapse onto a single universal curve, independent of model size. **(Bottom)** Quantification of collapse quality. We measure (i) the noise floor σ: relative variation in reducible loss across random seeds for individual models, and (ii) the collapse tolerance Δ: relative variation in normalized reducible loss across all models and seeds. **(a-b)** Without learning rate decay, normalized curves collapse with tolerance comparable to the noise floor. **(c,d)** With linear learning rate decay, we observe "*supercollapse*", where the collapse tolerance is significantly smaller than the noise floor for a substantial fraction of training. In the top and middle rows, solid lines represent the estimated mean with 90% confidence interval shown in shaded regions.

2. Empirical Observations

In this section, we demonstrate the central empirical finding of our work: loss curves from compute-optimally trained models collapse to a single universal curve under appropriate affine rescaling, often with surprising precision. We verify this phenomenon in transformer and MLP architectures across multiple datasets with various learning rate schedules. We then quantify the quality of this collapse and identify the key conditions required to achieve it.

2.1. Experiment Setup

To study compute-optimal training dynamics, we need an experimentally tractable *scaling ladder* — a procedure for training a sequence of models with increasing compute, with hyperparameters that promote compute-optimal performance. We consider both Transformers on next-pixel prediction tasks and MLPs on a synthetic regression task. We scale the model size by increasing the width (embedding dimension) and keeping depth fixed unless stated otherwise. While our experiments are limited in scale, small-scale proxies have been shown to capture training behaviors that generalize to larger systems (Wortsman et al., 2023). We provide complete data and architecture specifications and training protocols in Appendix A.

Transformers Next-Token Prediction. We use decoder-only Transformers with embedding dimensions ranging from 768 to 2048, keeping the number of Transformer blocks fixed at 3. This results in models with parameters ranging from 12M to 79M, approximately log-uniformly spaced. We train on next-token prediction tasks using two datasets: 1) CIFAR-5M (Nakkiran et al., 2020), a dataset of 6M generated CIFAR-like images, and 2) Lichess, a collection of chess games recorded in algebraic chess notation. All models use μP parameterization (Yang & Hu, 2021; Yang et al., 2021) for initialization and learning rates, and are trained with Adam (Diederik P. Kingma, 2015). We include a depth-scaling experiment in Appendix B, finding similar results.

MLPs on Power-Law Fourier Features. To investigate other architectures and training objectives, we train MLP models with varying widths from 512 to 4096 on a synthetic regression task. The target function has a power-law Fourier spectrum, designed to elicit the power-law scaling laws observed in natural data. We count each example as 1 token.

2.2. Notation and Scaling Law Estimation

Let $L(t, p, \omega)$ be the loss after t tokens (proportional to steps) on a model with p parameters using random seed ω. Denote $\bar{L}(t, p) = \mathbb{E}_\omega[L(t, p, \omega)]$ the seed-averaged expected curves. We estimate $\mathbb{E}_\omega[\cdot]$ using 5 seeds throughout this paper. We always report the test loss on heldout data.

We estimate the optimal number of tokens $t^\star(p)$ for a p-parameter model as $t^\star(p) = (p/p_0)^\gamma$, where γ is the data exponent, by extracting the Pareto frontier of expected loss vs. compute under a constant learning rate schedule, following a procedure similar to Approach 1 in Hoffmann et al. (2022). We estimate compute as $c = 6tp$ FLOPs (Kaplan et al., 2020). We reuse the same $t^\star(p)$ as the training horizon for other learning rate schedules. For each task and schedule, we fit the resulting compute-optimal scaling law using the form $L_0 + ac^{-b}$ (Figure 1, top row), for constants $L_0, a, b \geq 0$. Following (Sharma & Kaplan, 2022; Hoffmann et al., 2022), we refer to L_0 as the *estimated irreducible loss*. Using the best-fit L_0, we define the reducible loss curve $\mathcal{L}(t, p, \omega) = L(t, p, \omega) - L_0$ and its expected value $\bar{\mathcal{L}}(t, p) = \mathbb{E}_\omega[\mathcal{L}(t, p, \omega)]$. We detail the procedure for fitting the compute-optimal training horizon in Appendix C.

2.3. Scaling Collapse of Compute-Optimal Loss Curves

The loss curves for different model sizes cover varying ranges of compute and loss values, but appear to follow a consistent shape, which motivates us to affinely rescale them to the *normalized loss curve* ℓ given by

$$\ell(x, p, \omega) = \frac{L(xt^\star(p), p, \omega) - \hat{L}}{L(t^\star(p), p, \omega) - \hat{L}}, \quad x \in [0, 1], \quad (1)$$

for some offset $\hat{L} > 0$. We refer to x as the normalized compute, as it measures the compute spent for training each model, normalized by its total compute budget. Note that the transformation depends on the random seed ω via the loss at

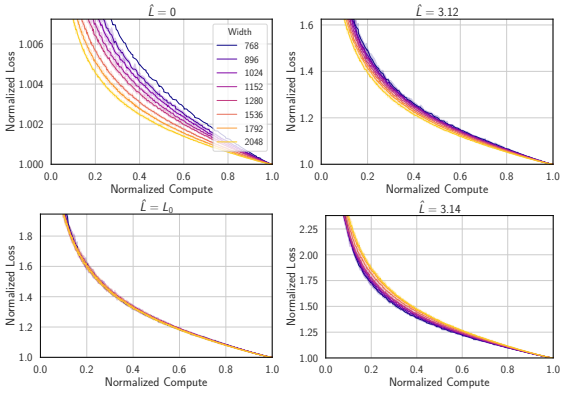


Figure 2: **Subtracting irreducible loss leads to the best collapse.** Setting $\hat{L}$ to values far from L_0 breaks the collapse on CIFAR-5M.

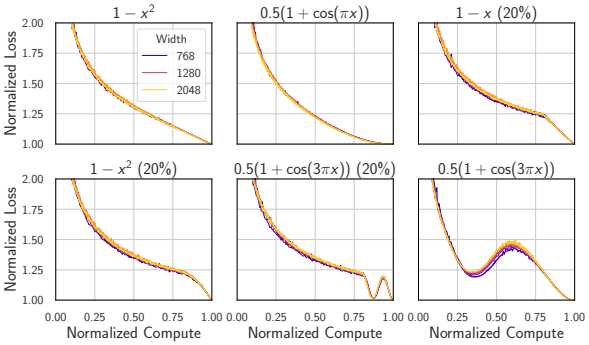


Figure 3: **Supercollapse occurs across a variety of learning rate schedules.** Each model is trained for the amount of compute that is estimated to be optimal under a constant learning rate. The universal curve is schedule-dependent, and collapse is best near the endpoint and when the learning rate is decreasing.

compute optimality for the particular curve being rescaled. We set $\hat{L} = L_0$ to subtract the estimated irreducible loss that bottlenecks the performance of large models, leading to $\ell(x, p, \omega) = \frac{\mathcal{L}(xt^\star(p), p, \omega)}{\mathcal{L}(t^\star(p), p, \omega)}$.

Remarkably, we observe that the family of normalized loss curves is nearly identical across p, revealing equal rates of relative progress (Figure 1, middle row). We say these curves *collapse*, as the phenomenon resembles the ubiquitous *scaling collapse* found in statistical physics, where observables from systems of different sizes collapse onto a single curve after appropriate rescaling, e.g. the rescaled magnetization-vs.-temperature curves of Ising lattices of different sizes collapse near criticality (Binder, 1981). The choice $\hat{L} = L_0$ is in fact necessary for the collapse (Figure 2).

2.4. Quantifying the Collapse Quality

We quantify the quality of collapse using the *collapse tolerance* Δ, defined as:

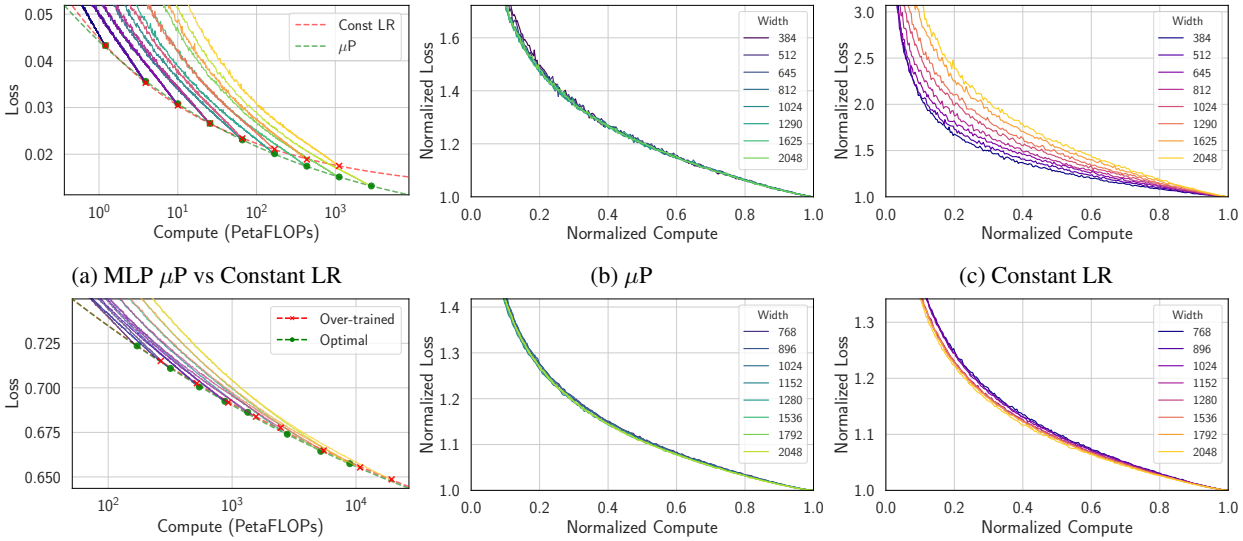
$$\Delta(x) = \frac{\mathbb{V}_{p,\omega}[\ell(x, p, \omega)]^{1/2}}{\mathbb{E}_{p,\omega}[\ell(x, p, \omega)]}, \quad (2)$$

where $\mathbb{E}_{p,\omega}$ and $\mathbb{V}_{p,\omega}$ denote the expectation and variance over the random seed and the empirical distribution of model size p in the scaling ladder (approximately log-uniformly distributed). The collapse tolerance represents the typical deviation between any two normalized loss curves for different p. For perspective, we can compare $\Delta(x)$ to the relative noise floor $\sigma(x, p)$

$$\sigma(x, p) = \frac{\mathbb{V}_\omega[\mathcal{L}(xt^\star(p), p, \omega)]^{1/2}}{\mathbb{E}_\omega[\mathcal{L}(xt^\star(p), p, \omega)]}, \quad (3)$$

which measures the relative fluctuation in the loss curve due to the random seed for a fixed model size p.

By the definition of ℓ, the collapse tolerance is always 0 at



(a) MLP μP vs Constant LR (b) μP (c) Constant LR

(d) Transformer Optimal vs Over-trained (e) Optimal (f) Over-trained

Figure 4: **Suboptimally scaling key hyperparameters breaks the collapse.** Even changes that only lead to minor worsening in the scaling law for small models can manifest as significant disruption to the collapse. **(Top)** Replacing μP with a constant learning rate cross models for MLPs. **(Bottom)** Increasing the data exponent γ from estimated compute-optimal value 1.02 to 1.2. for Transformer trained on chess. We perform a separate power-law fit to determine the value L_0 for each scaling ladder.

$x = 1$. For a constant learning rate, $\Delta(x)$ (solid green line) quickly rises to a level comparable to $\sigma(x, p)$ (dashed lines) (Figure 1(a-b), bottom row), and remains at that level for most x. This shows that variations in the normalized curves arise primarily from seed-to-seed fluctuations rather than model-to-model differences, quantitatively demonstrating that the observed collapse is non-trivial.

2.5. Supercollapse: Consistency Below the Noise Floor

Remarkably, for decaying learning rate schedules, we find that the collapse tolerance is less than the noise floor for a significant fraction of training; that is, $\Delta(x) < \sigma(x, p)$ for $x > 1 - \delta$ for some moderate δ as large as 0.5 (Figure 1 (b-d), bottom row). We refer to this stronger form of collapse as *supercollapse*. Supercollapse appears for any learning rate schedule which decays to a small value at the end of training (Figure 3).

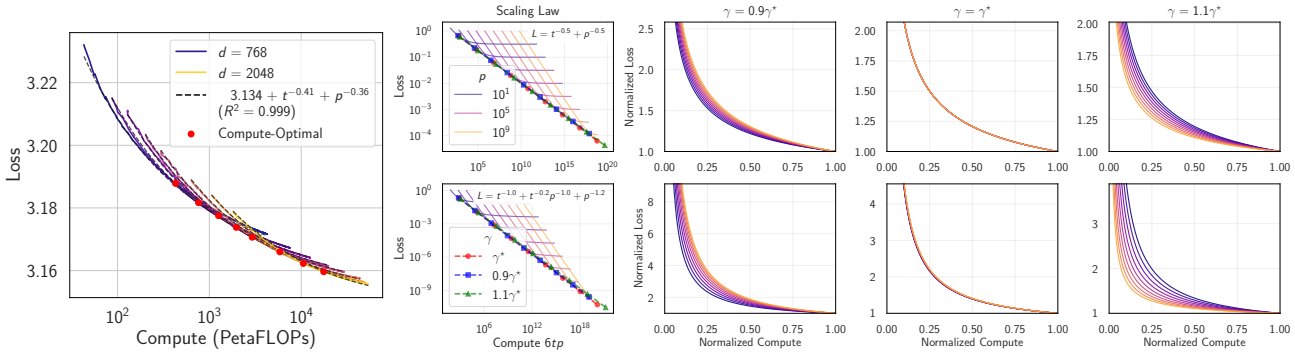
When supercollapse occurs, self-normalized reducible loss curves from different models collapse better than our ability to predict any individual model's loss. The key to supercollapse is the fact that the normalization uses the final loss of the specific loss curve, which leads to variance reduction by exploiting correlations at different times along a single optimization trajectory. We explain this phenomenon in detail in Section 3.3.

2.6. Suboptimal Scaling Breaks Supercollapse

Supercollapse provides a practical method for comparing inherently noisy training loss curves across model scales with precision that exceeds naive noise floor estimates, without the need for expensive multi-seed experiments typically required to obtain equally clean signals. This comparison can provide valuable diagnostic information about scaling where the ability to distinguish small signal from noise is often crucial (Xiao, 2024), which we now demonstrate.

Model Parameterization. Carefully parameterizing the model, i.e., scaling the initialization, learning rate, and possibly other hyperparameters as model size increases, is crucial for achieving stable and efficient training at scale (Yang et al., 2021; Bordelon et al., 2023; 2024c; Everett et al., 2024). When models are trained in the wrong parameterization, we expect the loss curves not to collapse due to a lack of consistent training dynamics across scales. Using the MLP setup, we show that replacing μP with a constant learning rate across widths breaks the collapse (Figure 4, top row). Remarkably, the normalized loss curves exposes inconsistent dynamics even in small models where the final losses are virtually identical between constant and μP scaling—demonstrating that collapse is a more sensitive probe of scaling behavior than final performance alone.

Compute-Optimal Data Exponent. For language models, Kaplan et al. (2020) showed that compute-optimal training corresponds to training each model to a fixed multiple of its converged loss. If this principle generalizes to our setting, the data exponent γ should match the compute-optimal value for collapse to occur. Indeed, when γ exceeds the optimal value, larger models will make more rapid relative initial progress but decelerate later as a function of nor-



(a) CIFAR-5M Fit

(b) Exact Power Laws

Figure 5: **Scaling collapse from sum of power-law curves.** (**a**) CIFAR-5M loss curves without learning rate decay agree well with the sum-of-power-laws fit $L(t,p) = L_0 + t^{-\mu} + p^{-\nu}$, a form commonly observed in natural data. We fit steps between $0.1\times$ and $10\times$ the compute-optimal training horizon. (**b**) Simulated exact sum-of-power-laws loss curves show scaling collapse precisely when the data exponent γ is the theoretical compute-optimal value $\gamma^\star$. Small variations of γ around $\gamma^\star$ lead to nearly negligible worsening in the resulting scaling law but dramatically disrupt the collapse.

malized compute, causing their normalized curves to shift downward. We indeed find this shift in Figure 4 (bottom row). This sensitivity suggests a novel application: rather than fitting power laws to sparse points on the Pareto frontier, one could tune γ to maximize collapse quality, leveraging the full statistical power of entire loss curves.

3. Explaining Loss Curve Scaling Collapse

In this section, we investigate theoretical explanations for the scaling collapse of compute-optimal loss curves and supercollapse. Our analysis starts with a simple observation: the numerator of the collapse tolerance $\Delta(x)$ can be decomposed as:

$$\mathbb{V}_{p,\omega}[\ell(x,p,\omega)] = \mathbb{V}_p\mathbb{E}_\omega[\ell(x,p,\omega)] + \mathbb{E}_p\mathbb{V}_\omega[\ell(x,p,\omega)]. \tag{4}$$

The first term corresponds to the variation between different scales p after averaging over all sources of randomness. We will first show how this term can be small:

- In Section 3.1, we prove that for a family of power-law neural scaling laws, compute-optimal loss curves indeed collapse after normalization. We show loss curves in our experiments fall into this family when using a constant learning rate schedule.
- In Section 3.2, we develop a simple theoretical model that successfully predicts the empirical loss curves under various learning rate schedules and explains why they collapse despite deviating from power laws. Given its effectiveness, we believe this model has value for understanding learning rate schedules more broadly.

We then analyze the second term, which captures the loss variance due to random seeds, averaged across model sizes:

- In Section 3.3, we show the same noise model enables us to reason about the noise in the loss curves, and quantitatively predict the variance reduction effect in supercollapse.

Together these findings provide an initial theoretical explanation for supercollapse, and uncover promising directions for future theoretical work.

3.1. Scaling Collapse from Power-Law Scaling

In this section, we consider deterministic models of the loss curves and assume all randomness has been averaged out.

Power-Law Pareto Frontier is Necessary. Given a family of loss curves $L(t,p)$, if $L(t,p)$ is differentiable in both t and p, the compute-optimal loss frontier after subtraction of $\hat{L}$ must be a power law in order for our affine transformation to induce scaling collapse (proof in Appendix D). This motivates our choice $\hat{L} = L_0$ since, by definition, L_0 is the offset that best induces a power law Pareto frontier. However, this is not sufficient to explain scaling collapse, which requires a more explicit form of $L(t,p)$.

Neural Scaling Laws. Motivated by empirical neural scaling laws in natural data[1] (Hestness et al., 2017; Kaplan et al., 2020; Hoffmann et al., 2022), we consider expected loss curves following a sum-of-power-laws scaling of the form

$$L(t,p) = L_0 + t^{-\mu} + p^{-\nu} \tag{5}$$

for constants $L_0 \geq 0, \mu, \nu > 0$, with potential constant multipliers absorbed via an appropriate choice of units. In Figure 5a, we show the CIFAR-5M loss curves are well-fit by Equation (5) if trained under a constant learning rate

[1]Note in Hoffmann et al. (2022), this form is only shown to hold for the loss at end of training.

schedule (averaged across 5 seeds). We also find decent fits in other datasets in Figure 11.

Equivalence by Balance of Power Laws. As before, let $t^\star(p)$ denote the training horizon. We will examine conditions under which $t^\star(p)$ (a) is compute-optimal, and (b) results in scaling collapse. We assume deterministic loss curves for now and omit the argument ω. To find compute-optimal $t^\star(p)$, we fix c so that $t(p) = c/(6p)$ and minimize the loss $\mathcal{L}(t(p), p) = t(p)^{-\mu} + p^{-\nu}$ with respect to p by setting $d\mathcal{L}/dp = 0$:

$$\frac{\partial \mathcal{L}}{\partial t}\frac{dt}{dp} + \frac{\partial \mathcal{L}}{\partial p} = -\mu t^{-\mu-1}(-t/p) - \nu p^{-\nu-1} = 0 \quad (6)$$

$$\iff \mu t^{-\mu} = \nu p^{-\nu} \quad (7)$$

which yields the relation $t^\star(p) = r^{-1/\mu} p^{\nu/\mu}$, with $r = \nu/\mu$. Under this scaling, the normalized loss curves are:

$$\ell(x,p) = \frac{(xt^\star)^{-\mu} + p^{-\nu}}{(t^\star)^{-\mu} + p^{-\nu}} = \frac{rx^{-\mu}\cancel{p^{-\nu}} + \cancel{p^{-\nu}}}{r\cancel{p^{-\nu}} + \cancel{p^{-\nu}}} = \frac{rx^{-\mu}+1}{r+1}. \quad (8)$$

All p dependences cancel, leaving the final expression independent of p and giving us an exact collapse. Moreover, it is clear that this is the unique choice for $t^\star(p)$ up to a constant multiplier that leads to such cancellation. This agreement is not an accident: compute-optimal scaling requires balancing the derivatives of two power laws, while collapse requires balancing the power laws themselves. The properties of power laws under differentiation make these two conditions coincide, up to a multiplicative constant.

In Figure 5b, we numerically verify the agreement between collapse and compute-optimal scaling. When the data exponent γ deviates from the optimal value ν/μ, we observe a suboptimal scaling law and no collapse. Note that the absence of an irreducible term in ℓ is also necessary. Had we set $\hat{L} = L_0 + E$ for some $E \neq 0$ in Equation (1), we would instead have $\ell(x,p) = \frac{(xt^\star)^{-\mu}+p^{-\nu}+E}{(t^\star)^{-\mu}+p^{-\nu}+E}$, where no $t^\star(p)$ can leave the numerator and denominator homogeneous in p.

In Appendix E, we study the more general form

$$L(t,p) = L_0 + \sum_{i=1}^{m} a_i t^{-\mu_i} p^{-\nu_i}, \quad (9)$$

which naturally arises in theoretical models of neural scaling laws (Paquette et al., 2024b; Bordelon et al., 2024a;b), and prove that compute-optimality implies scaling collapse by balancing the two dominant terms, though with $m > 2$ the collapse is only exact asymptotically due to finite-size effects.

Together with the close empirical fit in Figure 5a, this analysis provides a good explanation for scaling collapse in the constant learning rate setting; however Equation (5) fails to fit the empirical loss curves with most learning rate schedules, as varying the learning rate breaks the power law form of the individual loss curves, clearly shown in Figure 3, though it does retain the power law Pareto frontier (Figure 1 (b-d)). In the next section we develop a model of the noise dynamics in gradient descent that explains how collapses under one schedule can transfer to other schedules.

3.2. Universality of Learning Rate Schedules

To understand why scaling collapse is robust across learning rate schedules, we develop a quantitative model for how learning rate schedules affect the loss curves. While an exact theoretical model seems out of reach for the realistic training setup, we show that a simple model based on quadratic loss analysis proves surprisingly effective. Under this model, we demonstrate that although learning rate schedules deform the loss curves in a schedule-dependent way, the deformation is approximately independent of p. We consider stochastic effects that depend on the random seed ω, but omit ω as an explicit argument for brevity.

3.2.1. A Simple Model for LR Schedules

Let $w(t)$ and $L(w(t))$ denote the parameters and loss at step t, we can model the dynamics of full-batch gradient descent under a small learning rate $\eta(t)$ with a gradient flow $\frac{dw}{dt} = -\eta(t)\nabla L(w(t))$. To model stochastic effects, a noise term is added to the gradient, leading to the SDE (Malladi et al., 2022)

$$\frac{dw}{dt} = -\eta(t)\Big(\nabla L(w) + \Sigma^{1/2}(w)\xi(t)\Big), \quad (10)$$

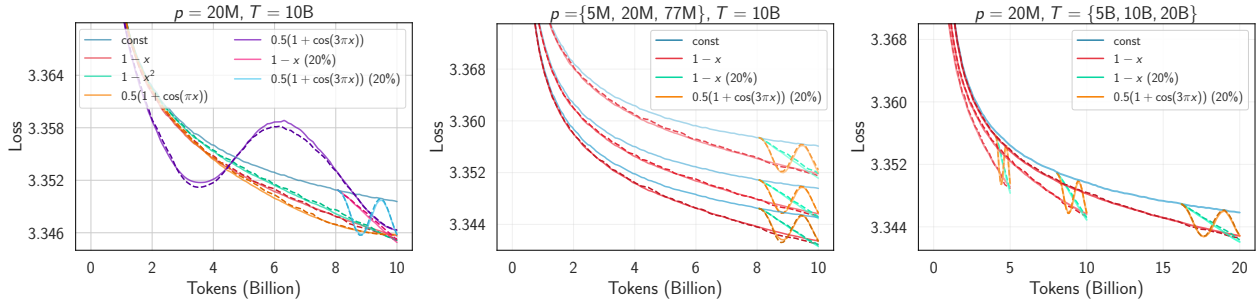
where the *mini-batch* gradient noise $\Sigma^{1/2}(w)\xi(t)$ is independent across t, satisfying $\mathbb{E}[\xi(t)\xi(t')] = \delta(t-t')I$, and we allow its covariance (which depends on batch size) $\Sigma(w)$ to be a function of the parameters. In *gradient flow time* $\tau(t) = \int_0^t \eta(s)ds$, we have

$$\frac{dw}{d\tau} = -\Big(\nabla L(w) + \Sigma^{1/2}(w)\xi(\tau)\Big), \quad (11)$$

and $\mathbb{E}[\xi(\tau)\xi(\tau')] = \delta(t-t')I = \eta(\tau)\delta(\tau-\tau')I$. For convenience, we overload the notation and use $\eta(\tau)$, $w(\tau)$, and $L(\tau)$ to denote the evolution of these quantities in gradient flow time.

Quadratic Loss. For the moment, let us suppose the loss function is quadratic $L(w) = L_0 + \frac{1}{2}w^\top H w$, where we assume the minimum is at the origin without loss of generality. Then $\nabla L(w) = Hw$ and the dynamics becomes a linear system driven by noise $\Sigma^{1/2}(w)\xi(\tau)$, with the solution

$$w(\tau) = e^{-H\tau}w(0) + \int_0^\tau ds\, e^{-H(\tau-s)}\Sigma^{1/2}(w(s))\xi(s). \quad (12)$$



(a) Vary Schedules (b) Scale Model Size (c) Scale Training Horizon

Figure 6: **A simple model predicts Transformer loss curves trained across learning rate schedules, model sizes p, and training horizons T on CIFAR-5M.** Dashed curves show the predicted loss as $\bar{L}'(\tau) = \bar{L}(\tau) + \alpha\,\delta\eta(\tau)\,\mathrm{Tr}(\Sigma'(\tau))$ (Equation (20)). α is the only free parameter and is set to 0.42. Each curve is smoothed with an exponential moving average with half life equal to 1% of total steps.

Letting $\bar{\Sigma}(s) = \mathbb{E}[\Sigma(w(s))]$, the expected loss is then given by

$$\bar{L}(\tau) = L_0 + \underbrace{\frac{1}{2}\mathbb{E}\big[w(0)^\top H e^{-2H\tau} w(0)\big]}_{\mathscr{F}(\tau)} + \underbrace{\frac{1}{2}\int_0^\tau ds\,\eta(s)\,\mathrm{Tr}\Big(\bar{\Sigma}(s) H e^{-2H(\tau-s)}\Big)}_{\mathcal{E}(\tau)}. \tag{13}$$

The first term $\mathscr{F}(\tau)$ is the forcing function, equal to the reducible loss curve in the deterministic limit $\eta\Sigma \to 0$ and is independent of the learning rate schedule. The second term $\mathcal{E}(\tau)$ is the excess loss due to SGD noise, which is a sum of exponential moving averages (up to normalization) of the gradient variance scaled by the learning rate over each eigenmode. Note that substituting in the specific Σ recovers the convolutional Volterra equation for high-dimensional linear regression analyzed in Paquette et al. (2021; 2024a).

If $\eta\bar{\Sigma}$ varies slowly compared to the timescale of the exponential moving average, we can make the approximation $\eta(s)\bar{\Sigma}(s) \approx \eta(\tau)\bar{\Sigma}(\tau)$ inside the integrand, giving us:

$$\mathcal{E}(\tau) \approx \frac{1}{2}\mathrm{Tr}\left(\eta(\tau)\bar{\Sigma}(\tau) H \int_0^\tau ds\, e^{-2H(\tau-s)}\right) \tag{14}$$

$$= \frac{1}{4}\mathrm{Tr}\big(\eta(\tau)\bar{\Sigma}(\tau)\big(1 - e^{-2H\tau}\big)\big). \tag{15}$$

For large τ the expected loss is then approximately

$$\bar{L}(\tau) \approx L_0 + \mathscr{F}(\tau) + \frac{1}{4}\eta(\tau)\,\mathrm{Tr}\big(\bar{\Sigma}(\tau)\big). \tag{16}$$

Given access to $\mathrm{Tr}\big(\bar{\Sigma}(\tau)\big)$, we can derive a prediction for how the loss changes as we change the learning rate schedule without knowing $\mathscr{F}$.

General Case. In Appendix F, we discuss how well these results generalize to more realistic setups. For non-quadratic losses, we show via perturbation theory that, to first order in $\eta\bar{\Sigma}$, one can make similar approximations to derive Equation (16) given an additional assumption that the Hessian is slowly varying, and with the forcing function $\mathscr{F}(\tau)$ no longer admitting a quadratic form. We also argue in Appendix F that Σ should be the *preconditioned* gradient covariance when using adaptive optimizers.

3.2.2. Predicting Loss Curves Across Schedules

We apply this simple model to predict empirical loss curves in the CIFAR-5M experiments. We use a slightly different experimental setup with a reduced context length of 128 to reduce the experiment cost (see Appendix A for details). We measure the trace of the preconditioned gradient covariance on a fixed set of 2M tokens.

Let $\bar{L}, \eta, \bar{\Sigma}$ be a given reference trajectory and $\bar{L}' = \bar{L} + \delta\bar{L}, \eta' = \eta + \delta\eta, \bar{\Sigma}' = \bar{\Sigma} + \delta\bar{\Sigma}$ be the target trajectory, Equation (16) allows us to predict the target loss via

$$\delta\bar{L}(\tau) \approx \frac{1}{4}\mathrm{Tr}\big[\delta\big(\eta(\tau)\bar{\Sigma}(\tau)\big)\big], \tag{17}$$

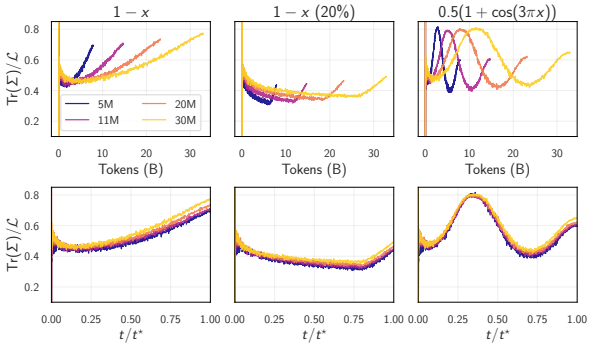


Figure 7: **Universality of gradient noise on CIFAR-5M.** Fixing a learning rate schedule, the ratio $\mathrm{Tr}\big(\bar{\Sigma}\big)/\bar{\mathcal{L}}$ is approximately a function of normalized compute alone, independent of model size. We show similar results with MLPs in Figure 14.

where $\delta\big(\eta(\tau)\bar{\Sigma}(\tau)\big) \equiv \eta'(\tau)\bar{\Sigma}'(\tau) - \eta'(\tau)\bar{\Sigma}(\tau)$. We use a constant learning rate for the reference trajectories and various schedules sharing the same peak learning rate for the target. Decomposing $\delta(\eta\Sigma) = \delta\eta\Sigma' + \eta\delta\Sigma$, we find the first term is typically 3 to 10 times larger than the second as the learning rate decays, which can be attributed to how learning rate interacts with curvature Figure 13a. In Figure 6, we only keep the first term, and predict the target loss as

$$\bar{L}'(\tau) \approx \bar{L}(\tau) + \alpha\,\delta\eta(\tau)\,\mathrm{Tr}(\Sigma'(\tau)), \tag{18}$$

where α is a shared hyperparameter. We find a single $\alpha = 0.42$ fits the target loss curves surprisingly well across schedules, model sizes, and training horizons. In Appendix G, we show even better fits for MLPs in Figure 12, and find that including the second term sometimes produces slightly worse fits.

3.2.3. Universal Scaling of Noise

For typical loss functions like mean-squared-error and cross-entropy, we expect some relation between the scale of the gradient covariance and the loss. For example, in noiseless high-dimensional linear regression with Gaussian features drawn from $\mathcal{N}(0, \hat{\Theta})$, we have $\mathrm{Tr}\big(\bar{\Sigma}\big) \approx 2\bar{\mathcal{L}}\,\mathrm{Tr}\Big(\hat{\Theta}\Big)$ (Paquette et al., 2021). For non-linear regression, $\hat{\Theta}$ should be taken to be the time-varying empirical neural tangent kernel for a first approximation. In this case, $\mathrm{Tr}\Big(\hat{\Theta}\Big)$ is known to depend strongly with the learning rate (Agarwala & Pennington, 2024), but we should expect weak dependence on model size given our models are trained with μP (see Noci et al. (2024) for evidence that curvature statistics depend weakly on model size in μP). Since in our experiments the schedule is a function of the normalized compute $x = t/t^\star$ alone, we hypothesize the ratio between $\mathrm{Tr}\big(\bar{\Sigma}\big)$ and the reducible loss $\bar{\mathcal{L}}$ to be largely a function of x and independent of p. That is, there exists a schedule-dependent function $h(x)$ such that

$$\mathrm{Tr}\big(\bar{\Sigma}(xt^\star(p))\big)/\bar{\mathcal{L}}(xt^\star(p)) \approx h(x). \tag{19}$$

Figure 7 confirms this hypothesis. As a result, combining Equation (18) and Equation (19) and making p-dependence explicit, we have

$$\bar{\mathcal{L}}'(\tau, p) \approx \bar{\mathcal{L}}(\tau, p)(1 - \alpha h(x)\delta\eta(\tau, p))^{-1}, \tag{20}$$

where x is the normalized compute at gradient flow time τ. We leave to future work a rigorous explanation of why this relation holds so well for cross-entropy loss, particularly the fact that gradient noise scales with the reducible loss rather than the total loss.

Scaling Collapse Across Schedules. Combining our insights so far, we can now finally understand why scaling collapse happens across schedules. Let $\bar{\ell}(x, p)$ and $\bar{\ell}'(x, p)$ be the expected normalized loss curves under two schedules S and S'. Let $y(x)$ map the normalized compute under S' to the normalized compute under S at matching gradient flow time, where y is independent of p *if the schedules are defined in terms of the normalized compute*. Let $\delta\hat{\eta}(x) = \delta\eta(xt^\star(p), p)$ be the difference between the two schedules measured in normalized compute. We have[2]

$$\bar{\ell}'(x, p) = \frac{\bar{\mathcal{L}}'(xt^\star(p), p)}{\bar{\mathcal{L}}'(t^\star(p), p)} \tag{21}$$

$$= \frac{\bar{\mathcal{L}}(y(x)t^\star(p), p)(1 + \alpha\delta\hat{\eta}(y(x)))^{-1}}{\bar{\mathcal{L}}(y(1)t^\star(p), p)(1 + \alpha\delta\hat{\eta}(y(1)))^{-1}} \tag{22}$$

$$= \bar{\ell}(y(x), p)\,\underbrace{\frac{1 + \alpha\delta\hat{\eta}(y(1))}{1 + \alpha\delta\hat{\eta}(y(x)))}}_{\text{independent of } p}, \tag{23}$$

which shows that, in expectation, collapse under one schedule (e.g. constant) implies collapse under any other schedule, provided we take Equation (20) to be exact. Since collapse under a constant learning rate can be attributed to the sum-of-power-laws scaling law, this result helps explain why we also observe collapse in other schedules.

Recent works have proposed more complex and empirically-driven models for how learning rate schedules affect loss curves (Tissue et al., 2024; Luo et al., 2025). Yet the surprising accuracy of our simple model (Equation (20)) suggests it captures the essential dynamics—and crucially, its transparent form reveals why collapse persists across schedules, providing theoretical insight beyond empirical curve-fitting.

3.3. Supercollapse as Variance Reduction

Finally, we turn to understanding the "super" in supercollapse: why does decaying the learning rate make averaging over multiple seeds unnecessary for collapse, and why is the collapse tolerance $\Delta(x)$ significantly smaller than the relative noise floor $\{\sigma(x, p)\}_p$ for a substantial fraction of training under learning rate decay? We can understand aspects of the phenomenon again using gradient flow time and the quadratic loss noise model.

Recall the variance in the collapsed curves can be decomposed as (omitting ω argument)

$$\mathbb{V}_{p,\omega}[\ell(x, p)] = \mathbb{E}_p\mathbb{V}_\omega[\ell(x, p)] + \mathbb{V}_p\mathbb{E}_\omega[\ell(x, p)]. \tag{24}$$

The second term is the variation we would get had we averaged over multiple seeds, while the first term is the extra expected variation due to using a single seed. Since, empirically, variations in the normalized curves primarily arise from seed-to-seed fluctuations rather than model-to-model differences (Section 2.4) under a constant schedule, and

[2] Assuming relative fluctuation in $\mathcal{L}$ is small so $\mathbb{E}[\mathcal{L}(x)/\mathcal{L}(y)] \approx \mathbb{E}[\mathcal{L}(x)]/\mathbb{E}[\mathcal{L}(y)]$.

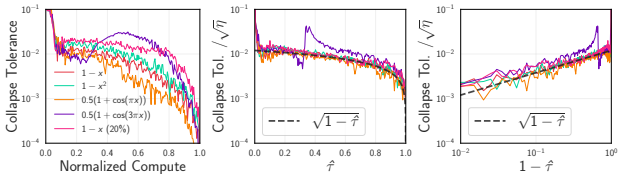


Figure 8: **Collapse tolerance follows the predicted $\sqrt{\eta(1-\hat{\tau})}$ scaling across schedules, explaining the variance reduction in supercollapse from learning rate decay.**

switching to other schedules does not significantly increase the model-to-model differences (Section 3.2), we will assume the first term of $\mathbb{V}_{p,\omega}[\ell(x,p)]$ dominates. We will analyze $\mathbb{V}_{\omega}[\ell(x,p)]$ for a fixed p. To simplify notation, we temporarily omit p-dependence and write ℓ in terms of t instead of x. Decomposing the reducible loss as its mean plus the relative fluctuation from the mean: $\mathcal{L}(t) = \bar{\mathcal{L}}(t)(1+\psi(t))$ and assuming $\psi \ll 1$, we have

$$\ell(t) = \frac{\bar{\mathcal{L}}(t)(1+\psi(t))}{\bar{\mathcal{L}}(t^\star)(1+\psi(t^\star))} \approx \bar{\ell}(t)(1+\psi(t)-\psi(t^\star)), \tag{25}$$

$$\mathbb{V}_{\omega}[\ell(t)/\bar{\ell}(t)] \approx \mathbb{E}[(\psi(t)-\psi(t^\star))^2] \tag{26}$$

We see it is the difference $\psi(t) - \psi(t^\star)$ rather than $\psi(t)$ itself that controls the relative variance in $\ell(t)$. This variance depends only on the amount of noise accumulated *between* time t and time $t^\star$. Since the optimization noise per step scales with the instantaneous learning rate, decaying the learning rate over time will precisely serve to decrease the variance in ℓ. By contrast, the noise floor $\sigma^2(x,p)$ normalizes by the expected final loss, and therefore scales as $\mathbb{E}[\psi^2(t)]$.

We can compute this variance under the quadratic model in Section 3.2. Let $\Delta w(\tau)$ and $\Delta\mathcal{L}(\tau)$ be the fluctuations of the parameters and reducible loss from their means in gradient flow time. To first order in $\Delta w(\tau)$, we have $\Delta\mathcal{L}(\tau) = g(\tau)^\top \Delta w(\tau)$ where $g(\tau)$ is the expected gradient, and

$$\Delta w(\tau) = \int_0^\tau ds\, e^{-H(\tau-s)}\Sigma^{1/2}(s)\xi(s). \tag{27}$$

Close to the end of training, for $\tau = \tau^\star - \delta\tau$ where $\tau^\star$ is the final gradient flow time and $\delta\tau > 0$ is small, let us approximate $\bar{\mathcal{L}}(\tau^\star) \approx \bar{\mathcal{L}}(\tau), g(\tau^\star) \approx g(\tau)$, then

$$\begin{aligned}
&\mathbb{E}[(\psi(\tau)-\psi(\tau^\star))^2] \\
&\approx \bar{\mathcal{L}}^{-2}(\tau) g(\tau)^\top \int_\tau^{\tau^\star} ds\, e^{-2H(\tau^\star-s)}\eta(s)\bar{\Sigma}(s)g(\tau) \quad (28)\\
&= \bar{\mathcal{L}}^{-2}(\tau) g(\tau)^\top \eta(\tau)\bar{\Sigma}(\tau)g(\tau)\delta\tau + O(\delta\tau^2), \quad (29)
\end{aligned}$$

where the last step follows from Taylor expanding the integrand around $s = \tau$. Dropping $O(\delta\tau^2)$ terms and noting the combination of $g^\top\bar{\Sigma}g \sim \mathcal{L}^2$, we have $\mathbb{E}[(\psi(\tau) - \psi(\tau^\star))^2] \sim \eta(\tau)\delta\tau$. Since this relation holds for each model size p, by taking expectation over p and recall the term $\mathbb{E}_p\mathbb{V}_\omega[\ell(x,p)]$ is assumed to be dominant in $\mathbb{V}_{p,\omega}[\ell(x,p)]$, we predict $\Delta^2(\hat{\tau}) \approx \mathbb{E}_p\mathbb{V}_\omega[\ell(\hat{\tau},p)/\bar{\ell}(\hat{\tau},p)] \propto \eta(\hat{\tau})(1-\hat{\tau})$, where $\hat{\tau} = \tau/\tau^\star$ denotes the normalized gradient flow time.

In Figure 8, we show this prediction indeed tracks our measurement for $\Delta(x)$, with $\Delta(\hat{\tau})/\sqrt{\eta(\hat{\tau})}$ approximately following the same $\sqrt{1-\hat{\tau}}$ scaling across many schedules, and explains why decaying the learning rate improves the collapse – by moving more slowly in gradient flow time and having a lower instantaneous learning rate that scales the noise.

Normalizing by the stochastic final loss is essential for supercollapse, where it acts as a control-variate (Glasserman, 2004): by leveraging the strong time-correlation of stochastic fluctuations along the loss trajectory, it cancels much of the shared noise and thereby sharply reduces the variance of the collapsed curve, beating the naive noise floor σ where normalization is done using the expected final loss.

4. Discussion

While deep learning theory has traditionally lagged behind empirical advances, recent work on neural network scaling limits has begun yielding practical insights, notably enabling hyperparameter transfer across model sizes. However, these frameworks typically consider limits where only the model size approaches infinity. To fully bridge the theory-practice gap, we must understand scaling limits where other quantities, particularly training duration, grow jointly—a significantly more challenging theoretical problem that better reflects how models are actually scaled in practice.

Our discovery that compute-optimal loss curves exhibit precise scaling collapse provides compelling empirical evidence that such a joint scaling limit exists under compute-optimal training and provides a practical procedure to measure it. Moreover, we found the quality of collapse can reveal information about whether hyperparameters are properly configured in a scaling ladder, suggesting a potentially valuable application.

Although we have identified the key ingredients underlying supercollapse—power-law scaling laws and learning rate-dependent noise dynamics—our analysis contains many approximations and is far from fully rigorous, suggesting deeper theoretical principles may be at work. In addition, both the power-law structure of loss curves and the scaling relationship between gradient noise and reducible loss are themselves observed regularities that demand explanations. Overall, we believe further investigating supercollapse holds exciting promise for deepening our scientific understanding of scaling neural networks.

Acknowledgements

We thank Courtney Paquette and Zixi Chen for helpful comments on an earlier version of this paper. SQ was supported by Google's TPU Research Cloud (TRC) program: https://sites.research.google/trc/.

Contribution Statement

SQ conducted the majority of experiments, led the theory development, and wrote the paper. LX initially observed supercollapse, contributed to theory, experimental design, and writing the paper. AGW advised SQ and edited the paper. JP contributed to theory, experimental design, and writing the paper. AA managed the research project, proved some theorems, guided the theory development, contributed to experimental design, and helped write the paper.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Agarwala, A. and Pennington, J. High dimensional analysis reveals conservative sharpening and a stochastic edge of stability, 2024. URL https://arxiv.org/abs/2404.19261.

Binder, K. Finite size scaling analysis of ising model block distribution functions. *Zeitschrift für Physik B Condensed Matter*, 43:119–140, 1981.

Bordelon, B., Noci, L., Li, M. B., Hanin, B., and Pehlevan, C. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. *arXiv preprint arXiv:2309.16620*, 2023.

Bordelon, B., Atanasov, A., and Pehlevan, C. A dynamical model of neural scaling laws. *arXiv preprint arXiv:2402.01092*, 2024a.

Bordelon, B., Atanasov, A., and Pehlevan, C. How feature learning can improve neural scaling laws. *arXiv preprint arXiv:2409.17858*, 2024b.

Bordelon, B., Chaudhry, H., and Pehlevan, C. Infinite limits of multi-head transformer dynamics. *Advances in Neural Information Processing Systems*, 37:35824–35878, 2024c.

Diederik P. Kingma, J. B. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Everett, K., Xiao, L., Wortsman, M., Alemi, A. A., Novak, R., Liu, P. J., Gur, I., Sohl-Dickstein, J., Kaelbling, L. P., Lee, J., et al. Scaling exponents across parameterizations and optimizers. *arXiv preprint arXiv:2407.05872*, 2024.

Glasserman, P. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

Hendrycks, D. and Gimpel, K. Gaussian Error Linear Units (GELUs). *Preprint arXiv 1606.08415*, 2016.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Luo, K., Wen, H., Hu, S., Sun, Z., Liu, Z., Sun, M., Lyu, K., and Chen, W. A multi-power law for loss curve prediction across learning rate schedules. *arXiv preprint arXiv:2503.12811*, 2025.

Malladi, S., Lyu, K., Panigrahi, A., and Arora, S. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35: 7697–7711, 2022.

Nakkiran, P., Neyshabur, B., and Sedghi, H. The deep bootstrap framework: Good online learners are good offline generalizers. *arXiv preprint arXiv:2010.08127*, 2020.

Noci, L., Meterez, A., Hofmann, T., and Orvieto, A. Super consistency of neural network landscapes and learning rate transfer. *Advances in Neural Information Processing Systems*, 37:102696–102743, 2024.

Paquette, C., Lee, K., Pedregosa, F., and Paquette, E. Sgd in the large: Average-case analysis, asymptotics, and stepsize criticality. In *Conference on Learning Theory*, pp. 3548–3626. PMLR, 2021.

Paquette, C., Paquette, E., Adlam, B., and Pennington, J. Homogenization of sgd in high-dimensions: Exact dynamics and generalization properties. *Mathematical Programming*, pp. 1–90, 2024a.

Paquette, E., Paquette, C., Xiao, L., and Pennington, J. 4+ 3 phases of compute-optimal neural scaling laws. *arXiv preprint arXiv:2405.15074*, 2024b.

Sharma, U. and Kaplan, J. Scaling laws from the data manifold dimension. *Journal of Machine Learning Research*, 23(9):1–34, 2022.

Tissue, H., Wang, V., and Wang, L. Scaling law with learning rate annealing. *arXiv preprint arXiv:2408.11029*, 2024.

Wortsman, M., Liu, P. J., Xiao, L., Everett, K., Alemi, A., Adlam, B., Co-Reyes, J. D., Gur, I., Kumar, A., Novak, R., et al. Small-scale proxies for large-scale transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023.

Xiao, L. Rethinking conventional wisdom in machine learning: From generalization to scaling. *arXiv preprint arXiv:2409.15156*, 2024.

Yang, G. and Hu, E. J. Feature Learning in Infinite-Width Neural Networks. *International Conference on Machine Learning (ICML)*, 2021.

Yang, G., Hu, E. J., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., and Gao, J. Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Zhang, B. and Sennrich, R. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

A. Experiment Details

Transformer Architecture. We use GeLU activations (Hendrycks & Gimpel, 2016), RMSNorm (Zhang & Sennrich, 2019), and learned positional embeddings. We untie the embedding matrix from the output head and do not use bias anywhere. The readout layer is always zero-initialized. We denote the embedding dimension with D. We set the intermediate dimension in the feedforward layers to D instead of the usual $4D$, which enables us to explore larger widths more efficiently.

CIFAR-5M. We use the CIFAR-5M dataset (Nakkiran et al., 2020) of 6 million CIFAR-like images. We convert the $32 \times 32 \times 3$ images to greyscale and flatten them into sequences of length 1024. The model autoregressively predicts the pixel intensities in raster-scan order. The vocabulary is the set of pixel intensities $\{0, \ldots, 255\}$. Following μP we parameterize the learning rate for each weight matrix as $\eta = \eta_{\text{base}}/D$ where d is the model dimension, except for the embedding matrix which has $\eta = \eta_{\text{base}}$. We use a parameter multiplier a on the embedding matrix. We use $\eta_{\text{base}} = 4$ and $a = 0.1$ as they led to good performance in our early experiments. We initialize the embedding matrix as $W_{ij}^{\text{emb}} \sim \mathcal{N}(0, 1)$, the output head as $W^{\text{head}} = 0$, all other non-readout matrices W as $W_{ij} \sim \mathcal{N}(0, 1/D)$. These hyperparameters were determined with a small amount of tuning in early experiments. We use a batch size of 256 images. We use a linear warmup for 1000 steps.

For the experiments in Section 3.2.2, we use a context length of 128, which means each image is divided into 8 examples. We use μP where the base embedding dimension is 128 and base learning rate is 0.003. We initialize the embedding matrix with a standard deviation (std) of 0.1 and multiply its learning rate by 10 relative to the base learning rate. The output projection and the feedforward and attention layers are zero-initialized. All other non-readout matrices are initialized with std $1/\sqrt{D}$. We use a batch size of 65536 tokens. We use a linear warmup for 10M tokens.

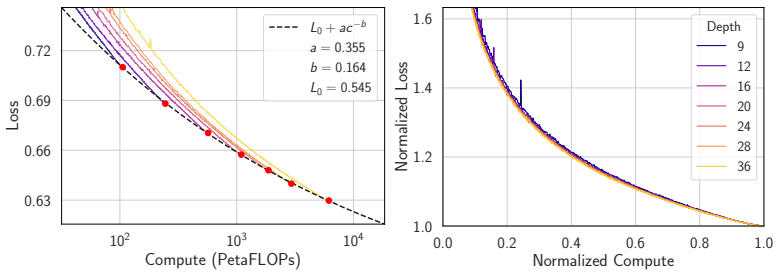
Chess. We run our experiments on the Lichess dataset available on Hugging Face at `https://huggingface.co/datasets/Lichess/standard-chess-games`. We used character-level tokenization and a context length of 128. We use μP where the base embedding dimension is 128 and base learning rate is 0.01. We initialize the embedding matrix with standard deviation (std) 0.1 and multiply its learning rate by 10 relative to the base learning rate. The output projection and the feedforward and attention layers are zero-initialized. All other matrices are initialized with std $1/\sqrt{D}$. We use a batch size of 65536 tokens. We use a linear warmup for 10M tokens.

MLP Experiments. Our MLP architecture is identical to the transformer with attention layers removed and the token and position embedding layers replaced by a linear layer. We use μP where the base embedding dimension is 128 and base learning rate is 0.001. The output projection and the feedforward and attention layers are zero-initialized. All other non-readout matrices are initialized with std $1/\sqrt{D}$. We use a batch size of 4096 examples. We do not use warmup.

The target function is defiend as $\phi(x) = \sum_{i=1}^{M} w_i \sqrt{2} \cos\left(2\pi k_i^\top x + b_i\right)$, with $x \in \mathbb{R}^8, w_i \sim \mathcal{N}(0, 1), b_i \sim \frac{\pi}{2}\text{Bernoulli}(0.5)$, $k_i = \text{round}(s_i v_i)$ where s_i is a scalar sampled from a power law with support $[1, \infty)$ and exponent -2, v_i is a random unit vector, and round rounds to the nearest point in $\mathbb{Z}^8$. During training, x is sampled uniformly from $[-0.5, 0.5]^8$, making the Fourier features orthonormal over the data distribution.

B. Scaling Collapse Across Transformer Depths

For scaling depth, we additionally apply a branch multiplier of $3/\text{depth}$ on the output of every feedforward and attention layer, as suggested by Bordelon et al. (2024c). We find a decent degree of collapse in Figure 9 when training on chess data.



(a) Scaling Law (b) Collapse

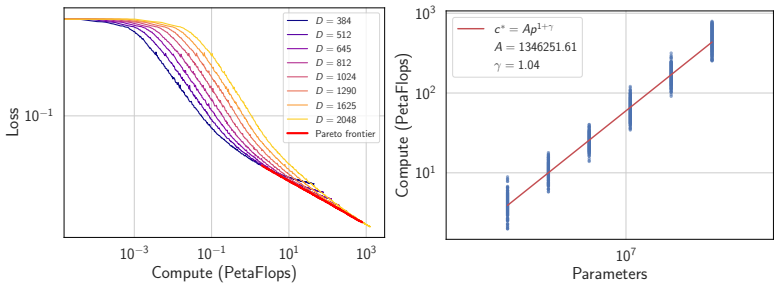
Figure 9: **Depthwise scaling collapse for transformers trained on chess.**

C. Estimating Compute-Optimal Training Horizon

To estimate the optimal compute for training each model, we perform the following steps in each experiment:

- We trained each model *without* learning rate decay but keeping the initial warmup. We chose a large enough number of steps so that the largest model could reach the compute-optimal loss frontier. We average the loss curves from 5 seeds.
- We numerically computed the compute-loss Pareto frontier to obtain an estimate of $c^\star(p)$ - the optimal compute for each model size p. We use logarithmically spaced points for c and find the p that achieves the best loss given c training FLOPs.
- We fit a power law $c^\star(p) = Ap^{1+\gamma}$ where A and γ are fit parameters. The optimal number of training tokens is then $t^\star(p) = c^\star(p)/(6p)$, which scales as p^γ. We remove outliers in this fit by dropping points from the smallest or largest model.

Figure 10 illustrates this procedure for the MLP experiment.



(a) Compute-Optimal Pareto Frontier (b) Fitting Data Exponent

Figure 10: **Estimating compute-optimal data exponent, illustrated with the MLP experiment.**

D. Power-Law Pareto Frontier is Necessary for Collapse

Recall $t^\star(p)$ is the optimal training horizon for model size p, i.e. $L(t^\star(p), p) = \min_{t',p':t'p'=t^\star(p)p} L(t', p')$. Let $c^\star(p) = 6t^\star(p)p$ be the optimal compute for p. In what follows, rather than writing $L(t, p)$, we will find it convenient to express the loss curves in terms of compute and model size. Letting $L(c, p)$ be the loss curves expressed this way, we have the following theorem:

Theorem D.1. *Let $\mathcal{L}(c, p)$ be C^1 in (c, p) and let $c^\star(p)$. Write $L(t, p) = \mathcal{L}(t, p) - \hat{L}$ for any offset $\hat{L}$ (e.g., $\hat{L} = L_0$). Define the normalized loss curve*

$$\ell(x, p) = \frac{L(xc^\star(p), p)}{L(c^\star(p), p)}, \quad x \in [0, 1]. \tag{30}$$

1. ***Necessity.*** *If ℓ is independent of p (collapse), then the Pareto frontier of $\{\mathcal{L}(c, p)\}_{c,p}$*

$$\mathcal{L}^\star(c) := \min_p \mathcal{L}(c, p) = \mathcal{L}(c^\star(p), p) \tag{31}$$

is a power law $\mathcal{L}^\star(c) = Ac^{-\delta}$ for some constants A, δ.

2. ***Sufficiency at first order.*** *Conversely, suppose $\mathcal{L}^\star(c) = Ac^{-\delta}$, then*

$$\frac{d}{dx}\ell(x, p)\Big|_{x=1} = -\delta, \tag{32}$$

independent of p; hence all curves share the same first-order behavior around $x = 1$, i.e., they collapse to first order around $x = 1$.

Proof of Necessity. Choose any p and write $c^\star = c^\star(p)$ for its compute–optimal budget. Because the collapsed curve $\ell(x, p)$ does not depend on p, its derivative at the endpoint $x = 1$ is a constant indepedent of p; set

$$\ell'(1) \;=\; -\delta \qquad (\delta > 0).$$

From the definition $\ell(x, p) = \frac{\mathcal{L}(xc^\star, p)}{\mathcal{L}(c^\star, p)}$ we obtain, by the chain rule,

$$\ell'(1) = \frac{c^\star}{\mathcal{L}(c^\star, p)}\, \partial_c \mathcal{L}(c, p)\big|_{c=c^\star} = \partial_{\log c} \log \mathcal{L}(c, p)\big|_{c=c^\star}.$$

Hence

$$\partial_{\log c} \log \mathcal{L}(c, p)\Big|_{c=c^\star(p)} = -\delta \quad \text{for } \textit{every } p. \tag{33}$$

By optimality, the point $(c^\star(p), p)$ lies on the compute–loss Pareto frontier $\{(c, \mathcal{L}^\star(c)\}_c$. Because $\mathcal{L}(c, p)$ is C^1 in c, p, the value and first derivative of the curve $\mathcal{L}(\cdot, p)$ matches those of the frontier at $c = c^\star(p)$:

$$\mathcal{L}^\star(c^\star(p)) = \mathcal{L}(c^\star(p), p), \quad \mathcal{L}'^\star(c^\star(p)) = \partial_c \mathcal{L}(c, p)\big|_{c=c^\star(p)}.$$

Combining with (33) yields

$$\frac{d}{d\log c} \log \mathcal{L}^\star(c) = \frac{c}{\mathcal{L}^\star(c)}\, \mathcal{L}^{\star\prime}(c) \xrightarrow{c=c^\star(p)} \frac{c^\star(p)}{\mathcal{L}(c^\star(p), p)}\, \partial_c \mathcal{L}(c, p)\Big|_{c=c^\star(p)} = -\delta,$$

i.e.,

$$\frac{d}{d\log c} \log \mathcal{L}^\star(c)\Big|_{c=c^\star(p)} = -\delta \qquad \text{for all } p.$$

Therefore the log–log slope of the frontier is the same $-\delta$ everywhere:

$$\frac{d}{d\log c} \log \mathcal{L}^\star(c) = -\delta \quad \forall\, c.$$

Integrating this leads to a power law $\mathcal{L}^\star(c) = A\, c^{-\delta}$.

Proof of sufficiency at first order. Assume the Pareto frontier is a power law

$$\mathcal{L}^\star(c) = A\, c^{-\delta}, \qquad A > 0,\ \delta > 0.$$

For every model size p the compute–optimal budget $c^\star = c^\star(p)$ satisfies $\mathcal{L}(c^\star, p) = \mathcal{L}^\star(c^\star)$. Because $\mathcal{L}$ is C^1 and $\mathcal{L}^\star$ is the Pareto frontier $\min_p \mathcal{L}(c, p)$, we have the tangency condition

$$\partial_c \mathcal{L}(c, p)\Big|_{c=c^\star} = \mathcal{L}^{\star\prime}(c^\star) = -\delta\, A\, (c^\star)^{-\delta-1}. \tag{S.1}$$

Differentiate the normalized loss $\ell(x, p) = \frac{\mathcal{L}(xc^\star, p)}{\mathcal{L}(c^\star, p)}$:

$$\frac{d}{dx}\ell(x, p)\Big|_{x=1} = \frac{c^\star}{\mathcal{L}(c^\star, p)}\, \partial_c \mathcal{L}(c, p)\Big|_{c=c^\star} = \frac{c^\star}{\mathcal{L}^\star(c^\star)}\, \mathcal{L}^{\star\prime}(c^\star).$$

Insert $\mathcal{L}^\star(c) = Ac^{-\delta}$ and (S.1):

$$\frac{d}{dx}\ell(x, p)\Big|_{x=1} = \frac{c^\star}{A(c^\star)^{-\delta}} \left[-\delta\, A\, (c^\star)^{-\delta-1}\right] = -\delta.$$

Therefore, the slope $\ell'(1, p) = -\delta$ is the same for *all* p; thus every normalized curve shares identical value $\ell(1, p) = 1$ and identical first derivative at $x = 1$. Consequently the family $\{\ell(x, p)\}$ collapses to first order around $x = 1$, completing the proof. □

Remark. This result shows that power-law compute-optimal frontier is necessary and itself already promotes a weaker form of the collapse. We note that the theorem doesn't directly apply to schedules which decay to 0 at the Pareto frontier as the loss curves have non-differentiable end points (only differentiable from the left) and are usually no longer tangent to the frontier.

E. Collapse for General Sum-of-Power-Laws Loss Curves

Theorem E.1 (Compute–optimal horizon implies asymptotic collapse)**.** *Let the expected loss be a* sum of power laws

$$L(t,p) \;=\; L_0 \;+\; \sum_{i=1}^{m} a_i\, t^{-\mu_i}\, p^{-\nu_i}, \qquad a_i > 0,\;\; \mu_i, \nu_i \geq 0,$$

where t is the number of optimization steps, p the model size, and L_0 an irreducible loss. Assume the training horizon scales as a power of the model size,

$$t^\star(p) \;=\; \kappa\, p^\gamma, \qquad \kappa > 0,\;\; \gamma > 0,$$

and define the combined exponents

$$\beta_i \;:=\; \mu_i \gamma + \nu_i, \qquad i = 1, \ldots, m.$$

Order them so that $\beta_1 \leq \beta_2 < \cdots \leq \beta_m$.

1. *If $t^\star(p)$ is* asymptotically compute-optimal*—that is, it minimizes L under the constraint $6t\,p = c$ (constant total compute) for large p—then the two smallest exponents coincide,*

$$\beta_1 = \beta_2, \qquad \Longrightarrow \qquad \gamma = \frac{\nu_1 - \nu_2}{\mu_2 - \mu_1}$$

   with $\mu_1 \neq \mu_2$. Exactly the terms $i = 1, 2$ dominate the loss as $p \to \infty$.

2. *With this choice of γ, the normalized loss curves*

$$\ell(x,p) := \frac{L\big(x\, t^\star(p),\, p\big) - L_0}{L\big(t^\star(p),\, p\big) - L_0}, \qquad 0 < x \leq 1,$$

   collapse asymptotically:

$$\ell(x,p) \;=\; u(x) \;+\; O\big(p^{-\varepsilon}\big), \qquad u(x) := \frac{b_1 x^{-\mu_1} + b_2 x^{-\mu_2}}{b_1 + b_2}, \quad \varepsilon := \beta_3 - \beta_1 > 0,$$

   where $b_i := a_i \kappa^{-\mu_i}$. All explicit p-dependence vanishes in the limit $p \to \infty$.

3. *The horizon required for collaspe is only determined up to an overall multiplicative constant: any $\lambda\, t^\star(p)$ with fixed $\lambda > 0$ works just as well.*

Proof. Fix the total compute $c := 6tp$ and regard p as the free variable, so that $t(p) = c/(6p)$ and $dt/dp = -t/p$. Differentiating $L\big(t(p), p\big)$ along this path gives

$$\frac{dL}{dp} \;=\; \sum_{i=1}^{m} \big(\partial_t L\big)\, \frac{dt}{dp} \;+\; \partial_p L \;=\; \frac{1}{p} \sum_{i=1}^{m} a_i (\mu_i - \nu_i)\, t^{-\mu_i}\, p^{-\nu_i}.$$

Compute-optimality demands $dL/dp = 0$, hence

$$\sum_{i=1}^{m} a_i (\mu_i - \nu_i)\, t^{-\mu_i}\, p^{-\nu_i} = 0$$

Insert the ansatz $t = \kappa p^\gamma$; then $t^{-\mu_i} p^{-\nu_i} = \kappa^{-\mu_i} p^{-\beta_i}$ and

$$\sum_{i=1}^{m} b_i (\mu_i - \nu_i)\, p^{-\beta_i} = 0, \qquad b_i := a_i \kappa^{-\mu_i}.$$

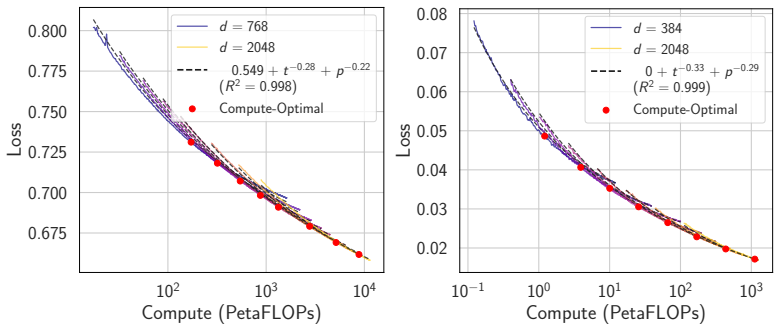
As $p \to \infty$ the terms with the smallest β_i dominate. For their weighted sum to vanish there must be at least two such terms, and they must share the same exponent, giving $\beta_1 = \beta_2$ and consequently $\gamma = (\nu_1 - \nu_2)/(\mu_2 - \mu_1)$.

With this γ,

$$L\big(t^\star(p), p\big) - L_0 = (b_1 + b_2)\, p^{-\beta_1}\big[1 + O\big(p^{-(\beta_3-\beta_1)}\big)\big].$$

Replacing $t^\star \mapsto x\, t^\star$ multiplies each b_i by $x^{-\mu_i}$; dividing by the expression above yields the claimed form for $\ell(x, p)$ and establishes collapse. Finally, multiplying $t^\star$ by any fixed $\lambda > 0$ just rescales every b_i and preserves the asymptotic collapse. □

Remarks. (i) For $m > 2$, the third-smallest exponent governs the leading finite-size error $O(p^{-\varepsilon})$ (ii) For $m > 2$, compute-optimal scaling implies asymptotic collapse, but the converse is not always true, since there can be multiple choices of γ that lead to balanced dominant power laws, which imply collapse, but only one of them can be compute-optimal. (iii) With $m = 2$ the remainder vanishes and we recover the perfect collapse of Section 3.1. (iii) Note we did not determine the compute-optimal γ, as the ordering of β_i depends on γ. Finding the compute-optimal γ can be done by finding all possible solutions for γ that balance the two dominant terms and taking the one that leads to the fastest asymptotic decrease in loss.



(a) Transformer on Chess (b) MLP

Figure 11: **Sum-of-power-laws fit on additional datasets.** Both tasks have loss curves that can be approximated by the sum of two power laws when using a constant learning rate schedule.

F. A Perturbative Model of Learning Rate Schedules

The dynamics of stochastic gradient descent in gradient flow time are given by

$$\frac{dw'}{d\tau} = -\Big(\nabla L(w') + \Sigma^{1/2}(w')\xi(\tau)\Big), \tag{34}$$

with noise correlation $\mathbb{E}[\xi(\tau)\xi(\tau)^\top] = \eta(\tau)\delta(\tau - \tau')$. For convenience, we rewrite this using $\xi(\tau) = \eta^{1/2}(\tau)\tilde{\xi}(\tau)$ so that

$$\frac{dw'}{d\tau} = -\Big(\nabla L(w') + \eta^{1/2}(\tau)\Sigma^{1/2}(w')\tilde{\xi}(\tau)\Big), \tag{35}$$

where now $\mathbb{E}[\tilde{\xi}(\tau)\tilde{\xi}(\tau')^\top] = \delta(\tau - \tau')$.

Our strategy is to solve $w'(\tau)$ as $w(\tau) + \delta w(\tau)$ where $w(\tau)$ is the deterministic trajectory satisfying $\frac{dw}{d\tau} = -\nabla L(w)$, up to first order in the gradient noise scale $\eta\Sigma$. Here $w(\tau)$ satisfies

Letting $\delta w \equiv w' - w$, $g \equiv \nabla L$ and taking the difference of the two differential equations:

$$\frac{d(\delta w)}{d\tau} = -\Big(g(w') - g(w) + \eta^{1/2}(\tau)\Sigma^{1/2}(w')\tilde{\xi}(\tau)\Big). \tag{36}$$

At first order,

$$g(w') \approx g(w) + H(w)\delta w \tag{37}$$

where $H(w) = \nabla^2 L(w)$ is the Hessian.

Our SDE for δw becomes:

$$\frac{d(\delta w)}{d\tau} = -H(w)\delta w - (\eta\Sigma)^{1/2}\tilde{\xi}(\tau) \tag{38}$$

We define the propagator $G(\tau, s)$ that satisfies:

$$\frac{dG(\tau, s)}{d\tau} = -H(w(\tau))G(\tau, s) \tag{39}$$

with $G(s, s) = I$.

For time-dependent $H(w(\tau))$, the propagator is:

$$G(\tau, s) = \mathcal{T} \exp\left(-\int_s^\tau dt\, H(w(t))\right) \tag{40}$$

where $\mathcal{T}$ denotes time-ordering.

Assuming the initial perturbation $\delta w(0) = 0$, the solution for δw is:

$$\delta w(\tau) = -\int_0^\tau ds\, G(\tau, s)(\eta\Sigma)^{1/2}(s)\tilde{\xi}(s). \tag{41}$$

Now expanding $L(w') = L(w + \delta w)$ to second order in δw gives

$$\begin{aligned}\delta L(\tau) &= L(w'(\tau)) - L(w(\tau)) \\ &\approx g(w(\tau))^\top \delta w(\tau) + \frac{1}{2}\,\delta w(\tau)^\top H(w(\tau))\,\delta w(\tau).\end{aligned} \tag{42}$$

Since $\mathbb{E}[\tilde{\xi}(s)] = 0$ and δw is linear in $\tilde{\xi}$, $\mathbb{E}[\delta w(\tau)] = 0$, so

$$\mathbb{E}\big[g(w(\tau))^\top \delta w(\tau)\big] = 0. \tag{43}$$

Thus the leading non–vanishing contribution to the *expected* loss shift comes from the quadratic term.

Using the solution for δw,

$$\delta w(\tau)\,\delta w(\tau)^\top = \int_0^\tau ds \int_0^\tau du\; G(\tau, s)(\eta\Sigma)^{1/2}(s)\tilde{\xi}(s)\tilde{\xi}(u)^\top(\eta\Sigma)^{1/2}(u)G(\tau, u)^\top. \tag{44}$$

Taking the expectation with $\mathbb{E}[\tilde{\xi}(s)\tilde{\xi}(u)^\top] = \delta(s - u)\, I$ gives

$$\mathbb{E}\big[\delta w(\tau)\,\delta w(\tau)^\top\big] = \int_0^\tau ds\; G(\tau, s)\,\eta(s)\Sigma(w'(s))\,G(\tau, s)^\top. \tag{45}$$

Substituting (45) into the quadratic term of (42),

$$\mathbb{E}[\delta L(\tau)] = \frac{1}{2}\int_0^\tau ds\;\; \mathrm{Tr}\left[H(w(\tau))\,G(\tau, s)\,\eta(s)\Sigma(w'(s))\,G(\tau, s)^\top\right]. \tag{46}$$

Using $\mathrm{Tr}[ABC] = \mathrm{Tr}[CAB]$,

$$\mathbb{E}[\delta L(\tau)] = \frac{1}{2}\int_0^\tau ds\;\; \mathrm{Tr}\left[G(\tau, s)^\top H(w(\tau))\,G(\tau, s)\eta(s)\Sigma(w'(s))\right]. \tag{47}$$

Equation (47) is the *exact* leading-order expression for the noise-induced change in expected loss. Conceptually, the derivation shows that—although the full dynamics are non-linear and the loss is not assumed quadratic—the *pertubation*

generated by small gradient noises behaves in a simple, linear–quadratic fashion: the weight perturbation δw is *linear* in the injected noise, and the resulting loss shift δL is *quadratic* in that perturbation.

Consequently the derivation and final formula completely mirrors the familiar quadratic-loss result, the only difference being that the constant Hessian H is now replaced by the time-dependent Hessian $H\big(w(\tau)\big)$ carried along the deterministic trajectory. In other words, small gradient noise "sees" the network through an instantaneous linearization, so all schedule effects enter through the propagator $G(\tau, s)$, the local Hessian, and the noise covariance, exactly as in the linear case.

Slow–variation and late–time limit. As in the quadratic loss case, we can simplify the result under an adiabatic approximation where the Hessian, schedule, and noise covariance changes slowly compared to the time-scale set by the instantaneous Hessian. Specifically, if $H(w(t)) \approx H(w(\tau)) \equiv H$ over the support of $G(\tau, s)^\top H(w(\tau))\, G(\tau, s)$, then $G(\tau, s) \approx e^{-H(\tau - s)}$ and $G(\tau, s)^\top H G(\tau, s) \approx H\, e^{-2H(\tau - s)}$. Assuming the exponential decay is fast compared to the variation of the noise scale $\eta\Sigma$, and taking $\tau \to \infty$, we have

$$\mathbb{E}[\delta L(\tau)] \approx \frac{1}{4} \operatorname{Tr}\left[\eta(\tau)\Sigma(w'(\tau))\right]. \tag{48}$$

Adaptive Optimizers and Preconditioned Gradient Covariance. When using adaptive optimizers with a preconditioner $P(t)$, the equation of motion becomes

$$\frac{dw}{dt} = -P^{-1}(t)\Big(\nabla L(w) + \Sigma^{1/2}(w)\xi(t)\Big), \tag{49}$$

If the preconditioner varies slowly, the dynamics can be treated as if there is no preconditioner, but in a transformed coordinate system:

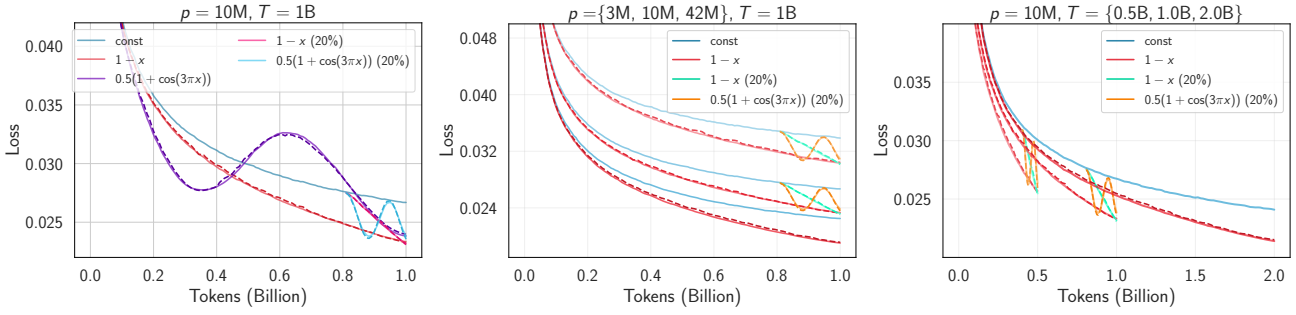
$$\tilde{w}(t) \;=\; P^{\frac{1}{2}}(t)\, w(t).$$

Differentiating and neglecting the $\mathcal{O}(\dot{P})$ term gives

$$\frac{d\tilde{w}}{dt} \;=\; P^{\frac{1}{2}}\frac{dw}{dt} \;=\; -P^{-\frac{1}{2}}\Big(\nabla_w L(w) + \Sigma^{\frac{1}{2}}(w)\,\xi(t)\Big) \;=\; -\nabla_{\tilde{w}} L(\tilde{w}) - \underbrace{P^{-\frac{1}{2}}\Sigma^{\frac{1}{2}}(w)\,\xi(t)}_{\text{noise}}.$$

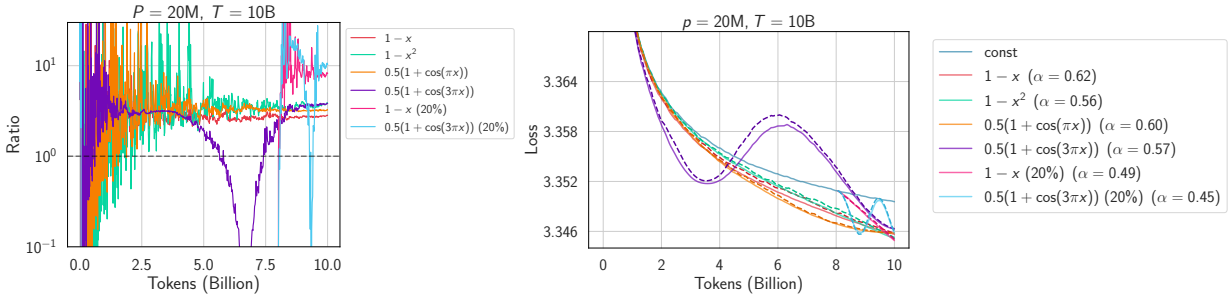
Therefore the gradient noise covariance in the new coordinates are $P^{-\frac{1}{2}}\,\Sigma(w)\,P^{-\frac{1}{2}}$, i.e. the preconditioned gradient covariance. In our experiments, we take P to be the instantaneous Adam preconditioner composed with a diagonal matrix encoding the per-layer learning rate introduced by μP, following what is done in Noci et al. (2024).

G. Additional Results on Learning Rate Schedules



(a) Vary Schedules (b) Scale Model Size (c) Scale Training Horizon

Figure 12: **A simple model predicts MLP loss curves trained across learning rate schedules, model sizes p, and training horizons T on the syntheticic regression task.** Dashed curves show the predicted loss as $\bar{L}'(\tau) = \bar{L}(\tau) + \alpha\,\delta\eta(\tau)\,\operatorname{Tr}(\Sigma'(\tau))$ (Equation (20)). α is the only free parameter and is set to 0.52. Each curve is smoothed with an exponential moving average with half life equal to 1% of total steps.



(a) $\frac{\delta\eta\Sigma'}{\eta\delta\Sigma}$ on CIFAR-5M

(b) Fit with both terms

Figure 13: Out of the two terms that make up $\delta(\eta\Sigma)$, the term $\eta\delta\Sigma$ is typically much smaller than $\delta\eta\Sigma'$ (a). Moreover, including it to sometimes produces a worse fit and make the optimal α vary more across schedules (b).

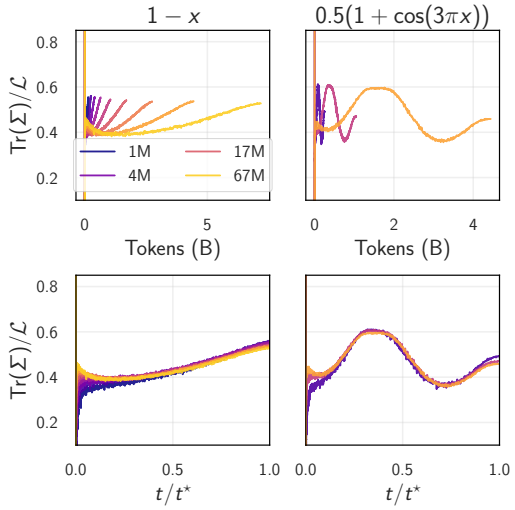


Figure 14: **Universality of gradient noise in MLPs.** Fixing a learning rate schedule, the ratio $\mathrm{Tr}(\bar{\Sigma})/\bar{\mathcal{L}}$ is approximately a function of normalized compute alone, independent of model size. On this regression task, the estimated irreducible loss is negligible (≈ 0.001) so $\mathcal{L} \approx L$.