How Nonlinear Dynamics Shape Neural Scaling Laws and the Hyperparameter Landscape

Anonymous authors
Paper under double-blind review

Abstract

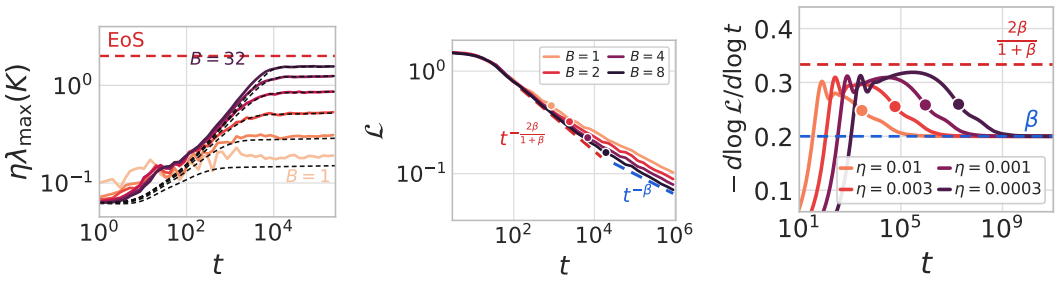
How does evolving curvature constrain the scaling of neural network training? Solvable models of neural scaling laws are mostly linear, with a fixed curvature, so they cannot capture how the growing curvature of real networks interacts with minibatch noise or the hyperparameter phenomena that depend on it. We study a minimal nonlinear extension, a two-layer linear network trained with SGD on a power-law task, and derive why and how fast its curvature grows, as feature learning ties the scale of the kernel to the norm of the function learned so far. The growing kernel amplifies minibatch noise until the noise level self-stabilizes at a fraction of the stochastic edge of stability set by the task, at any learning rate and even without finite-step-size effects. This feedback sets a speed limit on training, so the accelerated exponent that prior infinite-batch analyses found for feature learning is only a transient, after which the loss respects an information-theoretic bound that infinite-batch theory violates and the compute-optimal exponent returns to its lazy value. The same feedback explains why the loss is insensitive to the learning rate over a range that widens during training, why the effective learning rate follows the linear scaling rule up to a critical batch size without tuning, and why annealing yields large gains that tolerate long decays, effects that in linear models hinge on the learning rate chosen by the practitioner. SGD experiments across tasks, model sizes, batch sizes and learning rates confirm our quantitative predictions, which connect scaling laws, curvature dynamics and the hyperparameter landscape within a single tractable model.

1 Introduction

Neural scaling laws, the power-law improvement of the loss with model size, data and compute (Kaplan et al., 2020; Hoffmann et al., 2022), have become a central tool for planning the training of large models, and a growing line of theoretical work explains them with solvable models. In these models, a linear predictor trained on a task with a power-law spectrum already reproduces power-law loss curves, model-size bottlenecks and compute-optimal trade-offs (Maloney et al., 2022; Bordelon & Pehlevan, 2022; Bordelon et al., 2024; Paquette et al., 2024; Lin et al., 2024), and Bordelon et al. (2025) recently showed that adding feature learning accelerates the scaling exponent of hard tasks, whose targets are poorly aligned with the initial kernel. These analyses, however, describe training at infinite batch size or at a noise level fixed by the learning rate and batch size, and therefore miss any feedback between the evolving kernel and the noise of SGD.

Real networks display several phenomena that a linear model cannot capture, starting with their curvature, which rises during training in a process known as progressive sharpening until it saturates at an edge of stability (EoS) set by the learning rate (Cohen et al., 2021), and which at finite batch size is limited instead by minibatch noise (Jastrzebski et al., 2020; Agarwala & Pennington, 2024; Andreyev & Beneventano, 2024). Their loss improves with the batch size only up to a critical batch size (McCandlish et al., 2018; Shallue et al., 2019; Zhang et al., 2025), can be insensitive to the learning rate over a broad range that ends abruptly (Wortsman et al., 2024), and drops sharply when the learning rate is annealed (Hägele et al., 2024; Qiu et al., 2025). In a linear model the curvature is fixed, so none of these phenomena can arise from an interaction between curvature and noise. Our goal is to understand how this interaction constrains the scaling of training and shapes the hyperparameter landscape, and for this we need a model that sharpens like a real network yet remains solvable.

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.



(a) Self-stabilization (b) A speed limit (c) A transient

Figure 1: **Sharpening and self-stabilization set a speed limit on the training of a minimal model of neural scaling laws.** (a) During SGD, feature learning sharpens the kernel until minibatch noise self-stabilizes it, at a sharpness $\eta\lambda_{\max}(K)$ that follows our theory (dashed) at every batch size B (from 1, light, to 32, dark), growing linearly with B below the critical batch size and approaching the edge of stability above it. (b) The loss falls at the accelerated rate of infinite-batch theory (red) and then at the speed limit $t^{-\beta}$ (blue), crossing over at a predicted time (dots) that grows with B (from 1, light, to 8, dark). (c) The local exponent predicted by the closure stays at the accelerated value $2\beta/(1+\beta)$ over a range of times that widens as η decreases and returns to β after $t_\times$ (circles) ($B = 1$, $\gamma = 3$). All panels use $\alpha = 2$ and $\beta = 0.2$.

To this end, we study a minimal nonlinear extension of these models, a two-layer linear network (Saxe et al., 2014) trained by SGD on a power-law task, and show that it captures these phenomena while remaining analytically tractable (Figure 1). We summarize our main findings as follows:

1. **Sharpening is predictable and self-stabilizes (Section 4).** To predict how the curvature evolves, we show that feature learning ties the scale of the kernel to the norm of the function learned so far, $k(k-1) = \gamma\|w\|^2$, which for hard tasks grows without bound, and that the growing kernel amplifies minibatch noise until the noise kernel norm self-stabilizes at $1 - 2^{-\beta}$, a fraction of the stochastic edge of stability set by the source exponent β of the task and independent of the learning rate.
2. **Self-stabilization sets a speed limit on scaling laws (Section 5).** Asking what self-stabilization implies for the loss, we find that it caps the progress per sample, so the accelerated exponent of infinite-batch theory holds only until a predictable crossover time, after which the loss falls as $D^{-\beta}$ in the number of samples D, which respects an information-theoretic bound and returns the compute-optimal exponent to its lazy value.
3. **Nonlinear dynamics shape the hyperparameter landscape (Section 6).** To explain how real networks respond to their hyperparameters, we show that because the kernel sets the noise level itself, a broad range of learning rates performs nearly equally well late in training, the effective learning rate follows the linear scaling rule by itself up to a critical batch size, and annealing lowers the loss by up to 2^β while tolerating long decays, whereas lazy training prefers a sharp optimum and a short decay.

2 Background

Solvable Models of Scaling Laws. Solvable models of scaling laws study regression on features with a power-law spectrum and derive the loss as a function of training time or data and model size (Maloney et al., 2022; Bordelon & Pehlevan, 2022; Bordelon et al., 2024; Paquette et al., 2024; Lin et al., 2024). Bordelon et al. (2025) add a trainable feature map and show with dynamical mean-field theory that feature learning improves the time exponent of hard tasks from β to $2\beta/(1+\beta)$ at infinite batch size, arguing that minibatch noise adds only a subleading correction, and we show that this correction becomes leading once the kernel grows.

Sharpening and the Edge of Stability. Full-batch gradient descent typically increases the sharpness until it reaches $2/\eta$ and then hovers there (Cohen et al., 2021), which Damian et al. (2023) explain through a self-stabilization in which the cubic term of the loss lowers the sharpness whenever it exceeds $2/\eta$. SGD with a larger learning rate or a smaller batch size reaches regions of lower curvature (Jastrzebski et al., 2020), and Agarwala & Pennington (2024) show that its stability is

controlled by the noise kernel norm and that training settles just below the stochastic edge of stability where this norm equals one, with Andreyev & Beneventano (2024) reporting a related threshold for the minibatch sharpness of real networks. These works take progressive sharpening as an empirical input or analyze it only near initialization, and minimal models have explained its mechanism in second-order regression (Agarwala et al., 2023) and in minimal linear networks (Kalra et al., 2025; Yoo et al., 2025), which also serve to study large learning rates (Lewkowycz et al., 2020), but its consequences for scaling laws remain open. Our self-stabilization is driven by minibatch noise at small learning rates and holds the kernel strictly below the stochastic edge of stability, unlike the step-size-driven mechanism of Damian et al. (2023).

Batch Size and Learning Rate Schedules. The loss at a fixed number of samples is nearly independent of the batch size up to a critical batch size, beyond which larger batches waste samples (McCandlish et al., 2018; Shallue et al., 2019; Zhang et al., 2025), and below it the learning rate is commonly scaled linearly with the batch size (Goyal et al., 2017; Smith et al., 2018), trends that McCandlish et al. (2018) relate to the noise in the gradient and that the noisy quadratic model reproduces in a linear setting (Zhang et al., 2019). Annealing the learning rate at the end of training produces a rapid drop in the loss, which warmup-stable-decay schedules exploit (Hägele et al., 2024) and which has been modeled with bounds from convex optimization (Schaipp et al., 2025) and with a model of SGD noise (Qiu et al., 2025).

3 Problem Setup

Task. Following the kernel regression view of Bordelon et al. (2024; 2025), we describe each input by its eigenfeatures $x \in \mathbb{R}^V$, drawn from $\mathcal{N}(0, \Lambda)$ with $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \dots)$ and $\lambda_i = i^{-\alpha}$, and take $V \to \infty$. The target is linear in these features, $y(x) = w^{\star\top} x$, with coefficients chosen such that

$$\lambda_i (w_i^\star)^2 = i^{-(\alpha\beta+1)}, \qquad \text{so that} \qquad \sum_{i>j} \lambda_i (w_i^\star)^2 \sim j^{-\alpha\beta}, \tag{1}$$

where the capacity exponent $\alpha > 1$ controls how fast the spectrum decays and the source exponent β controls how difficult the target is to learn with the initial kernel. We focus on hard tasks with $\beta < 1$, whose targets have infinite norm in the reproducing kernel Hilbert space of the initial kernel, since this is the regime in which feature learning changes the scaling laws (Bordelon et al., 2025).

Model. We train a two-layer linear network on the top N eigenfeatures $x_{1:N}$,

$$f(x) = (Qu)^\top x_{1:N}, \qquad Q \in \mathbb{R}^{N\times N},\ u \in \mathbb{R}^N, \qquad Q(0) = I,\ u(0) = 0, \tag{2}$$

and refer to $w = Qu$ as the end-to-end weights. The population loss is

$$\mathcal{L} = \tfrac{1}{2}\,\mathbb{E}_x\big(f(x) - y(x)\big)^2 = \tfrac{1}{2}\sum_{i\le N} \lambda_i\,(w_i^\star - w_i)^2 + \mathcal{L}_\infty(N), \tag{3}$$

where $\mathcal{L}_\infty(N) = \frac{1}{2}\sum_{i>N} \lambda_i (w_i^\star)^2 \sim N^{-\alpha\beta}$ is the part of the target that the model cannot represent. This model is the variant of Bordelon et al. (2025) in which a trainable matrix acts on fixed features (their Appendix B), with one simplification: while their features are a random Gaussian projection of the eigenfeatures, which they analyze with dynamical mean-field theory, ours are the top N eigenfeatures themselves, so model size enters only by truncating the target through $\mathcal{L}_\infty(N)$ and the spectrum seen by the model is exactly a power law (Section A relates the two notations).

Training. We train online with minibatch SGD, drawing B fresh samples at each step and updating

$$u \leftarrow u - \eta\,\nabla_u \mathcal{L}_B, \qquad Q \leftarrow Q - \gamma\eta\,\nabla_Q \mathcal{L}_B, \tag{4}$$

where $\mathcal{L}_B$ is the minibatch loss and the feature learning parameter γ sets the relative learning rate of the first layer and hence the speed of feature evolution (Bordelon et al., 2025). At $\gamma = 0$ the features stay frozen and the model reduces to linear regression on the top N features, a simplified version of the random-feature setting of Bordelon et al. (2024); Paquette et al. (2024) with the random projection replaced by truncation, and we refer to $\gamma = 0$ as the lazy regime and to $\gamma > 0$ as the rich regime (Chizat et al., 2019; Woodworth et al., 2020).

Kernel Dynamics. The dynamics are simplest to describe in function space, where, writing $g_B = \nabla_w \mathcal{L}_B$ for the minibatch gradient with respect to w, the updates in Equation (4) give, to first order in η,

$$\begin{aligned} w &\leftarrow w - \eta M g_B, \\ M &\leftarrow M - \gamma\eta \left(g_B w^\top + w g_B^\top + 2\,(w^\top g_B)\, I \right), \end{aligned} \tag{5}$$

where $M = QQ^\top + \gamma \|u\|^2 I$ is the neural tangent kernel (Jacot et al., 2018) in w-space, initialized at $M(0) = I$. In the coordinates $\Delta_i = \sqrt{\lambda_i}\,(w_i^\star - w_i)$, the loss becomes $\mathcal{L} = \frac{1}{2}\|\Delta\|^2 + \mathcal{L}_\infty$ and the mean dynamics read $\Delta \leftarrow \Delta - \eta K \Delta$, where the kernel $K = \Lambda^{1/2} M \Lambda^{1/2}$ starts from the input covariance, $K(0) = \Lambda$. Feature learning is thus the evolution of K, and we refer to the largest eigenvalue of K as the sharpness.

Replacing Q by a scalar gate q, as in $f(x) = (qu)^\top x_{1:N}$, gives the kernel $M = q^2 I + \gamma u u^\top$, which obeys the same update Equation (5) to first order in η (Section A), so we run most experiments with this parameterization, which has $O(N)$ rather than $O(N^2)$ parameters.

Finite-Batch Fluctuations at Small Learning Rate. We study SGD at a small learning rate and a finite batch size, where the discreteness of the updates matters only through the deterministic edge of stability $\eta \lambda_{\max}(K) = 2$ (Cohen et al., 2021). Minibatch fluctuations are set instead by the noise kernel norm $\mathcal{K} \approx \eta\,\mathrm{Tr}(K)/(2B)$ of Agarwala & Pennington (2024), and the second moments of the residual grow without bound once $\mathcal{K}$ exceeds 1, a threshold they call the stochastic edge of stability. Since $\mathcal{K}$ grows with the kernel, treating the noise as a small correction fails once features are learned, however small the learning rate, so we keep the minibatch fluctuations in full, work in continuous time, and include the deterministic edge of stability, which matters at large batch sizes, as a constraint.

Goals. We aim to predict the loss $\mathcal{L}(t; N, B, \eta, \gamma)$ after t steps, and from it its exponent in time and the compute-optimal scaling law at a fixed compute $C = NBt$ (Section 5), and its dependence on the learning rate, batch size and schedule together with the optimal hyperparameters (Section 6).

Infinite-Batch Scaling Violates a Sample-Complexity Bound. The acceleration of hard tasks from $t^{-\beta}$ to $t^{-2\beta/(1+\beta)}$ found by Bordelon et al. (2025) at infinite batch size cannot hold at finite batch size in general. After t steps SGD has seen $D = Bt$ samples, and no algorithm can outperform the Bayes-optimal error for targets with the power-law structure of Equation (1), which is at least the target power outside the top D modes because D noiseless linear measurements determine at most D directions of the target (Section B):

$$\mathcal{L} \;\geq\; \tfrac{1}{2} \sum_{i>D} \lambda_i (w_i^\star)^2 \;\sim\; D^{-\alpha\beta}. \tag{6}$$

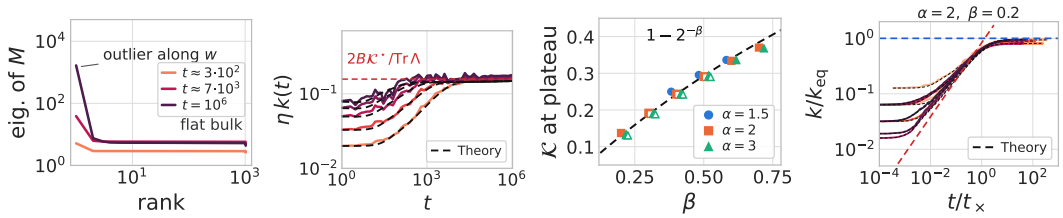
At $B = 1$ the infinite-batch rate $D^{-2\beta/(1+\beta)}$ therefore violates this bound whenever $\alpha(1+\beta) < 2$, which occurs among hard tasks with $\alpha < 2$. We will show that the resolution lies in the growth of $\mathcal{K}$: progressive sharpening amplifies minibatch noise until the kernel self-stabilizes below the stochastic edge of stability, and the resulting speed limit restores a rate consistent with Equation (6).

4 Predictable Sharpening and Self-Stabilization

Predicting the loss requires predicting how the kernel evolves, so in this section we reduce the kernel dynamics to a single scalar, explain why it grows, and derive the level and time at which minibatch noise stops its growth, taking $N \to \infty$ (Section D discusses finite model size) and B small enough that the deterministic edge of stability is never reached.

The Learned Kernel Is Isotropic up to a Rank-One Spike. Although the kernel is an $N \times N$ matrix, a single scalar suffices to track it, because its update Equation (5) consists of a rank-two part along w and g_B and an isotropic part proportional to $w^\top g_B$, and starting from $M(0) = I$ and $w(0) = 0$ it preserves the form

$$M = k\, I + \frac{\gamma}{k}\, w w^\top, \qquad k(k-1) = \gamma \|w\|^2, \tag{7}$$



(a) Spectrum of M (b) Invariance to η (c) Noise level (d) Collapse in $t/t_\times$

Figure 2: **Sharpening is predictable and self-stabilizes.** (a) In the matrix model, the eigenvalues of M form a flat bulk that rises during training and a single outlier along the learned function w, so the kernel is isotropic up to a rank-one spike ($B = 1$, $N = 1024$). (b) At $B = 1$, the effective learning rate ηk in SGD (solid) converges to $2B\mathcal{K}^\star/\operatorname{Tr}\Lambda$ (red) for every η from 0.02 (light) to 0.08 (dark), following the closure (dashed). (c) Across tasks with $\alpha \in \{1.5, 2, 3\}$ (markers) and β from 0.2 to 0.7, SGD self-stabilizes at the noise kernel norm $1 - 2^{-\beta}$ (dashed), for two learning rates at $B = 1$ (filled) and at $B = 2$ (open). (d) Rescaled by k_{eq} and $t_\times$ (Equation (12)), kernel trajectories of ten SGD runs with $\eta \in [0.003, 0.04]$, $\gamma \in [0.25, 3]$ and $B \in \{1, 2, 4\}$ (colored from early to late $t_\times$) collapse onto their closures (dashed), following the sharpening law (red) and then the plateau (blue). Unless stated otherwise, runs use $\alpha = 2$, $\beta = 0.2$, $\gamma = 0.75$ and $N = V = 2^{20}$.

for any sequence of minibatch gradients, as substituting Equation (7) into Equation (5) shows (Section C). Feature learning therefore rescales the initial kernel by a single factor k tied to the norm of the learned function, up to a rank-one spike along it. For the scalar gate, $k = q^2$ and the second relation is the conservation law $q^2 - \gamma\|u\|^2 = 1$, and the matrix model shows the same spike on top of a flat bulk (Figure 2a). The spike carries little of the loss decrease (Section A), so the sharpness is close to $k\lambda_1$. Averaging the isotropic part of Equation (5) over minibatches and accounting for the spike through Equation (7) gives (Section C)

$$\frac{dk}{dt} = \frac{2k}{2k-1}\,\gamma\eta\,(G - \operatorname{Tr}\Sigma), \qquad G = \bar{\Delta}^\top(\Delta(0) - \bar{\Delta}), \tag{8}$$

where $\bar{\Delta}$ and Σ are the mean and covariance of the residual over minibatch sampling. We refer to G as the drive, since it measures the overlap between the remaining residual and the function already learned and is positive, causing the kernel to grow. Minibatch noise enters with the opposite sign, as fluctuations of the residual reduce its average overlap with the learned function. The prefactor, which falls from 2 to 1 as k grows, accounts for the spike, which speeds up the learning of the function already learned and so reduces the drive.

A Two-Variable Closure. To close Equation (8), we make two further approximations. First, apart from the prefactor in Equation (8), the mean residual follows the lazy dynamics in the kernel clock $\theta = \int_0^t dt'\,\eta k(t')$, namely $\bar{\Delta}_i = \Delta_i(0)\,e^{-\lambda_i\theta}$, so the drive and the noiseless loss $\mathcal{L}_0 = \frac{1}{2}\|\bar{\Delta}\|^2$ depend on time only through θ and equal their lazy values at time θ. Second, since the covariance relaxes on the fast timescale of the top modes, it tracks its stationary value, which for Gaussian features depends on the kernel only through its trace (Section C), giving

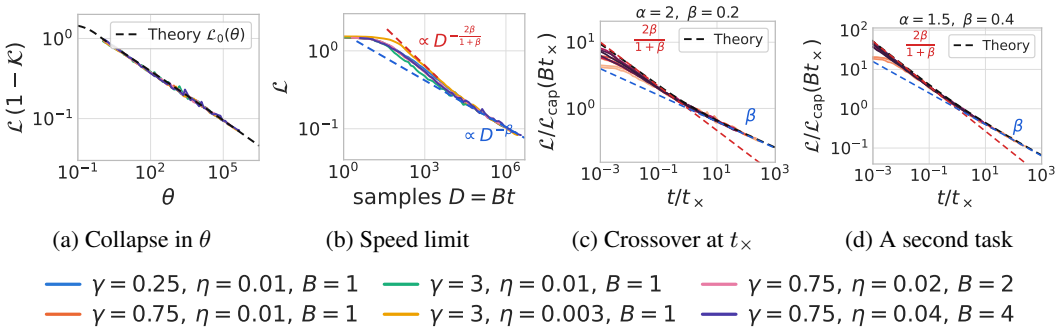
$$\mathcal{L} = \frac{L_0(\theta)}{1-\mathcal{K}}, \qquad \operatorname{Tr}\Sigma = \frac{2\mathcal{K}}{1-\mathcal{K}}\,\mathcal{L}_0(\theta), \qquad \mathcal{K} = \frac{\eta k \operatorname{Tr}\Lambda}{2B}. \tag{9}$$

Together with $d\theta/dt = \eta k$, Equations (8) and (9) form a closed system for the two scalars (θ, k). Minibatch noise inflates the loss by the factor $1/(1-\mathcal{K})$, which diverges at $\mathcal{K} = 1$, the stochastic edge of stability of Agarwala & Pennington (2024).

Feature Learning Causes Progressive Sharpening. The closure explains why the kernel sharpens, as on a power-law task the drive and the noiseless loss decay as $G(\theta) \approx c_G\,\theta^{-\beta}$ and $\mathcal{L}_0(\theta) \approx c_\mathcal{L}\,\theta^{-\beta}$, with $c_G = (1 - 2^{-\beta})\,\Gamma(\beta)/\alpha$ and $c_\mathcal{L} = 2^{-\beta}\,\Gamma(\beta)/(2\alpha)$, where Γ is the gamma function, so that while the noise is negligible, Equation (8) integrates to

$$k(k-1) \approx \frac{2\gamma c_G}{1-\beta}\,\theta^{1-\beta}, \tag{10}$$

which is Equation (7) with the norm of the learned function $\|w\|^2 = 2\int_0^\theta d\theta'\,G(\theta')$ of the lazy dynamics. Sharpening thus tracks the growth of the learned function, which gives a mechanism



(a) Collapse in θ (b) Speed limit (c) Crossover at $t_\times$ (d) A second task

$\gamma = 0.25,\ \eta = 0.01,\ B = 1$ $\gamma = 3,\ \eta = 0.01,\ B = 1$ $\gamma = 0.75,\ \eta = 0.02,\ B = 2$
$\gamma = 0.75,\ \eta = 0.01,\ B = 1$ $\gamma = 3,\ \eta = 0.003,\ B = 1$ $\gamma = 0.75,\ \eta = 0.04,\ B = 4$

Figure 3: **Feature learning is a change of clock with a speed limit, and the accelerated exponent is a transient.** (a) Against the kernel clock $\theta = \int dt\, \eta k$ computed from the measured k, and with the noise factor $1 - \mathcal{K}$ divided out, loss curves for six settings of (γ, η, B) (legend below) collapse onto the lazy noiseless loss $\mathcal{L}_0(\theta)$ (dashed). (b) Against the number of samples $D = Bt$, the same runs fall faster than $D^{-\beta}$ while the kernel sharpens (red) and then merge with the speed limit $\mathcal{L}_{\text{cap}}(D)$ (blue), which does not depend on η, γ or B. (c) With time rescaled by $t_\times$ and the loss by $\mathcal{L}_{\text{cap}}(Bt_\times)$, loss curves for the ten settings of Figure 2d collapse onto the closure (dashed), which bends from the accelerated slope $2\beta/(1+\beta)$ (red) to the self-stabilized slope β (blue) at $t \approx t_\times$. (d) The same collapse for $\alpha = 1.5$ and $\beta = 0.4$ (ten settings with $\eta \in [0.003, 0.02]$, $\gamma \in [0.25, 4]$, $B \in \{1, 2, 4\}$).

for progressive sharpening, typically taken as an empirical input in analyses of the edge of stability (Cohen et al., 2021; Damian et al., 2023), and since for hard tasks the target itself has infinite norm, $\sum_i w_i^{\star 2} = \infty$, the kernel grows without bound as $k \propto \sqrt{\gamma}\,\theta^{(1-\beta)/2}$, at a rate set by the source exponent. The key insight is that the drive $G = \sum_i \bar{\Delta}_i(\Delta_i(0) - \bar{\Delta}_i)$ multiplies, mode by mode, the part already learned and the part still to be learned, so it is largest for the modes currently being learned, and a larger kernel speeds up their learning. Progressive sharpening accelerates the clock to $\theta \propto (\eta\sqrt{\gamma}\,t)^{2/(1+\beta)}$ and recovers the infinite-batch result $\mathcal{L} \propto t^{-2\beta/(1+\beta)}$ of Bordelon et al. (2025).

Minibatch Noise Self-Stabilizes the Kernel. At finite batch size, sharpening cannot continue indefinitely, and to see where it stops, note that $\mathcal{K} \propto k$ makes the noise term in Equation (8) grow with the kernel until it cancels the drive, $G = \operatorname{Tr}\Sigma$, which by Equation (9) fixes

$$\mathcal{K}^\star = \frac{G}{2\mathcal{L}_0 + G} = 1 - 2^{-\beta}, \qquad k_{\text{eq}} = \frac{2B\,\mathcal{K}^\star}{\eta \operatorname{Tr}\Lambda}, \qquad \mathcal{L} = 2^\beta \mathcal{L}_0(\theta). \tag{11}$$

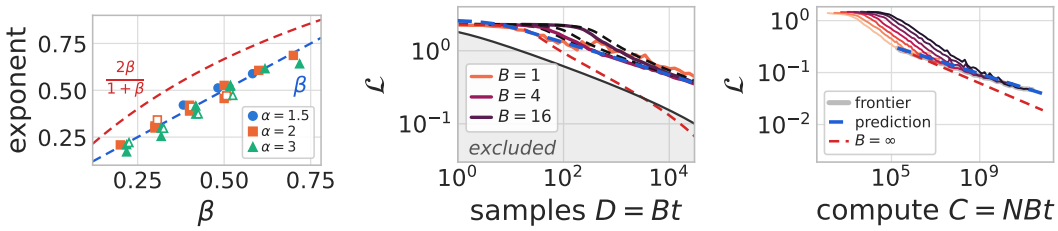
The fixed point is stable, since a kernel above k_{eq} experiences more noise than drive and shrinks back. Remarkably, the kernel settles at a fixed fraction $\mathcal{K}^\star$ of the stochastic edge of stability, set by the task through β, offering a mechanism for the self-stabilization of the noise kernel norm below 1 that Agarwala & Pennington (2024) observed in real networks and predicting that its level depends on the difficulty of the task. SGD confirms this prediction across tasks, learning rates and batch sizes once the model resolves the target (Figure 2c and Section D). Moreover, $k_{\text{eq}} \propto B/\eta$, so the sharpness falls as the learning rate grows or the batch size shrinks, as in real networks (Jastrzebski et al., 2020), and the effective learning rate ηk_{eq} is independent of η, as the kernel undoes any change in the learning rate (Figure 2b and Figure 8). Equating the noiseless growth Equation (10) with k_{eq} gives the time at which sharpening gives way to self-stabilization,

$$t_\times \approx B^{\frac{1+\beta}{1-\beta}}\,(\eta^2\gamma)^{-\frac{1}{1-\beta}}, \tag{12}$$

up to a task-dependent constant (Section C). Rescaled by $t_\times$ and k_{eq}, SGD runs spanning an order of magnitude in each of η, γ and B collapse onto a single curve on two tasks (Figure 2d and Figure 12a).

5 Self-Stabilization Sets a Speed Limit on Training

We now ask what the evolution of the kernel implies for the loss, and in particular whether the accelerated exponent of infinite-batch theory survives at finite batch size, showing that pinning the kernel at $k_{\text{eq}} \propto B/\eta$ caps the rate at which the loss can fall.



(a) Late-time exponent (b) Bayes bound (c) Compute frontier

Figure 4: **Self-stabilization sets a speed limit on scaling laws.** (a) For the settings of Figure 2c ($B = 1$ filled, $B = 2$ open), the SGD loss exponent after self-stabilization follows β (blue), below the infinite-batch value $2\beta/(1+\beta)$ (red). (b) For $\alpha = 1.2$ and $\beta = 0.2$, where $\alpha(1+\beta) < 2$, the infinite-batch prediction (red) enters the region excluded by the Bayes-optimal bound Equation (6) (gray), while SGD at every batch size follows the closure (black) onto the speed limit (blue) and stays above the bound ($\eta = 0.01$, $\gamma = 64$, $N = V = 2^{20}$). (c) SGD loss curves against compute $C = NBt$ for model sizes N from 32 to 2048 (light to dark) at $B = 4$, whose lower envelope (gray) follows the predicted compute-optimal law $C^{-\alpha\beta/(1+\alpha)}$ (blue), shallower than the infinite-batch law (red), for $\alpha = 2$ and $\beta = 0.2$ with $V = 32N$ target modes.

Feature Learning Is a Change of Clock With a Speed Limit. To connect the kernel to the loss, note that by Equation (9) the loss at any time equals the lazy, noiseless loss evaluated at the clock $\theta(t)$, inflated by the noise factor, so that all dependence on γ, η and B enters through $\theta(t)$ and $\mathcal{K}(t)$. Plotting $\mathcal{L}\,(1-\mathcal{K})$ against θ therefore collapses loss curves across hyperparameters onto the single lazy curve $\mathcal{L}_0(\theta)$ (Figure 3a). In this picture, the benefit of feature learning is a faster clock, $d\theta/dt = \eta k$, and self-stabilization caps its rate at

$$\frac{d\theta}{dt} = \eta k_{\text{eq}} = \frac{2B\,\mathcal{K}^\star}{\operatorname{Tr}\Lambda}, \tag{13}$$

a fixed amount of progress per sample that is independent of η and γ. Measured in samples $D = Bt$, the self-stabilized loss is therefore the same function $\mathcal{L}_{\text{cap}}(D) = 2^\beta \mathcal{L}_0(2\mathcal{K}^\star D/\operatorname{Tr}\Lambda) \propto D^{-\beta}$ for every setting, and each run approaches this speed limit from above, falling faster than it while the kernel sharpens and merging with it once the kernel self-stabilizes (Figure 3b).

The Accelerated Exponent Is a Transient. How long the accelerated exponent lasts follows from combining the two regimes, since after an initial lazy phase and dropping constants the loss follows

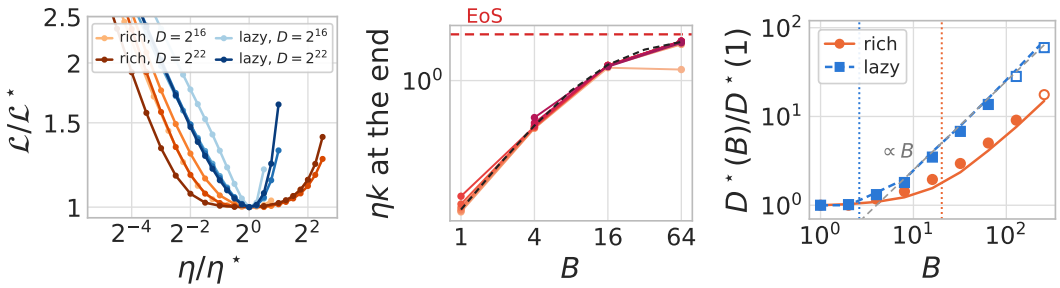
$$\mathcal{L}(t) \approx \max\Big\{ \underbrace{(\eta\sqrt{\gamma}\,t)^{-\frac{2\beta}{1+\beta}}}_{\text{sharpening}}, \underbrace{(Bt)^{-\beta}}_{\text{self-stabilized}} \Big\}, \tag{14}$$

where the two terms cross at $t_\times$ (Equation (12)), after which the clock runs at its capped rate and the loss falls with the lazy exponent β. SGD loss curves across η, γ and B collapse onto this form on two tasks (Figure 3c,d), with a late-time exponent β on every task we tested (Figure 4a), and a larger batch size keeps the loss on the accelerated rate for longer (Figure 1b). Since sharpening begins once the lazy clock ηt reaches $\gamma^{-1/(1-\beta)}$ and ends at $t_\times$, the accelerated phase spans a range of times that grows as $(B/\eta)^{(1+\beta)/(1-\beta)}$, as the closure at $N = \infty$ illustrates (Figure 1c).

Self-Stabilization Restores the Sample-Complexity Bound. We can now resolve the conflict with the sample-complexity bound raised in Section 3, since the self-stabilized rate $D^{-\beta}$ is consistent with the bound $D^{-\alpha\beta}$ of Equation (6) for every task because $\alpha > 1$, and at $\alpha = 1.2$ and $\beta = 0.2$ SGD at $B \in \{1, 4, 16\}$ follows the closure onto the speed limit and stays above the bound that the infinite-batch prediction violates (Figure 4b).

The Compute-Optimal Exponent Returns to Its Lazy Value. We finally ask whether the speed limit changes the compute-optimal scaling law. At finite model size, the loss saturates at the model bottleneck $\frac{1}{2}\sum_{i>N}\lambda_i(w_i^\star)^2 \propto N^{-\alpha\beta}$ (Bordelon et al., 2024), and since the speed limit makes the loss a function of the number of samples alone, a model of size N that has seen D samples reaches $\mathcal{L} \approx \mathcal{L}_{\text{cap}}(D) + c\,N^{-\alpha\beta}$ at compute $C = ND$ for $B < B_{\text{crit}}$. Minimizing over N at fixed C gives $N^\star \propto C^{1/(1+\alpha)}$ and

$$\mathcal{L}^\star(C) \propto C^{-\frac{\alpha\beta}{1+\alpha}}, \tag{15}$$



(a) Learning rate (b) Self-tuning (c) Critical batch size

Figure 5: **Nonlinear dynamics shape the hyperparameter landscape.** (a) At a constant learning rate, the loss after D samples against the learning rate, each relative to its optimum, flattens over a range that widens as D grows from 2^{16} to 2^{22} (light to dark) in the rich regime (orange), while the lazy optimum (blue) keeps its sharp shape ($B = 1$, $\alpha = 2$, $\beta = 0.5$). (b) In the rich regime, the effective learning rate ηk at the end of training grows with B along the same curve for every base learning rate (light to dark), following the closure (dashed) up to the edge of stability, whereas in the lazy regime it stays at η ($\alpha = 1.5$, $\beta = 0.2$, $D = 2^{22}$). (c) With annealing, the samples needed to reach a fixed loss (SGD: markers, closure: lines) stay flat up to a critical batch size (dotted) that is much larger in the rich regime ($\alpha = 1.5$, $\beta = 0.2$).

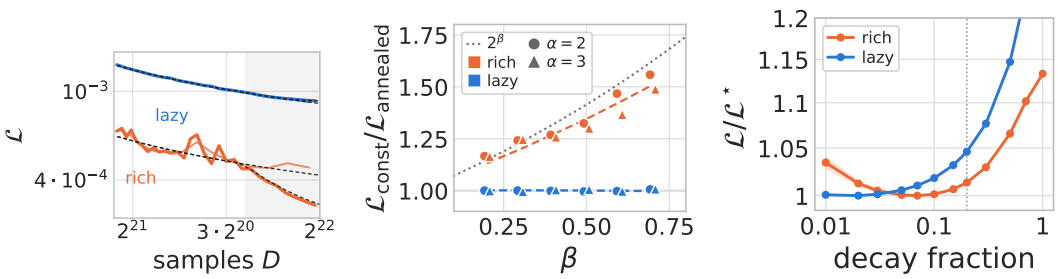
which is the compute-optimal law of lazy training, whereas the accelerated rate of infinite-batch theory would give the steeper $C^{-2\alpha\beta/(\alpha(1+\beta)+2)}$ (Bordelon et al., 2025). On two tasks, the lower envelope of SGD loss curves across model sizes follows Equation (15) and is clearly shallower than the infinite-batch prediction (Figure 4c and Figure 12c).

6 How Nonlinear Dynamics Shape the Hyperparameter Landscape

Practitioners tune the learning rate, batch size and schedule of real networks guided by three empirical regularities: the loss can be insensitive to the learning rate over a broad range (Wortsman et al., 2024), the learning rate can be scaled linearly with the batch size up to a critical batch size (Goyal et al., 2017; McCandlish et al., 2018), and annealing lowers the loss substantially (Hägele et al., 2024). Linear models reproduce these trends only through a noise kernel norm $\mathcal{K} = \eta \operatorname{Tr}\Lambda/(2B)$ that the practitioner tunes by hand. In the rich regime the kernel sets $\mathcal{K}$ itself, and in this section we ask whether this single feedback explains each of them.

The Learning Rate Basin Widens as Training Proceeds. To understand why real networks tolerate a broad range of learning rates, note that in both regimes the loss after D samples depends on the hyperparameters through $\mathcal{K}$ alone, $\mathcal{L} = \mathcal{L}_0(2\mathcal{K}D/\operatorname{Tr}\Lambda)/(1-\mathcal{K})$, since $\mathcal{K}$ sets both the progress per sample, $d\theta/dD = 2\mathcal{K}/\operatorname{Tr}\Lambda$, and the noise inflation. In the lazy regime $\mathcal{K} = \eta \operatorname{Tr}\Lambda/(2B)$, and minimizing over η gives a sharp optimum at $\mathcal{K}_{\text{lin}} = \beta/(1+\beta)$ whose position does not change with the horizon. In the rich regime $\mathcal{K}$ self-stabilizes at $\mathcal{K}^\star$ for any η once $t > t_\times$, so the loss is nearly independent of η above the learning rate at which $t_\times = D/B$, which by Equation (12) is $\eta_{\min} \propto B\, D^{-(1-\beta)/2}\gamma^{-1/2}$, and the basin below the divergence threshold widens as training proceeds. Late in training a broad range of learning rates therefore performs nearly equally well, as SGD confirms (Figure 5a and Figure 10).

Self-Stabilization Tunes the Learning Rate to the Batch Size. To understand how the learning rate should scale with the batch size and where this scaling stops, note that the progress per sample is capped by the stochastic edge $\mathcal{K} < 1$ at small B and by the deterministic edge $\eta k\lambda_1 < 2$ at large B. In the lazy regime at a constant learning rate, the tuned noise level $\mathcal{K}_{\text{lin}} = \beta/(1+\beta)$ gives $\eta^\star = 2B\mathcal{K}_{\text{lin}}/\operatorname{Tr}\Lambda$ up to $B = \operatorname{Tr}\Lambda/(\mathcal{K}_{\text{lin}}\lambda_1)$, while with annealing the best lazy learning rate sits at the stability edge and the critical batch size drops to $\operatorname{Tr}\Lambda/\lambda_1$, of order one for power-law spectra. In the rich regime, the kernel sets $\mathcal{K} = \mathcal{K}^\star$ for any η, with or without annealing, so the effective learning rate $\eta k_{\text{eq}} = 2B\mathcal{K}^\star/\operatorname{Tr}\Lambda$ follows the linear scaling rule (Goyal et al., 2017; Smith et al.,



(a) Loss curves (b) Annealing gain (c) Decay length

Figure 6: **Learning rate annealing removes the noise that self-stabilization amplifies.** (a) At the same learning rate, annealing over the last 20% (shaded) lowers the rich loss well below its constant-rate value (thin) and barely changes the lazy loss, following the closure (dashed) ($B = 2$, $\eta = 0.06$). (b) The resulting gain grows with β close to 2^β (dotted) in the rich regime and stays near 1 in the lazy regime, for SGD at $\alpha = 2$ (circles) and 3 (triangles) and the closure (dashed). (c) Final loss after $D = 2^{24}$ samples, relative to its minimum, against the fraction of training over which η decays linearly to zero ($B = 8$, $\eta = 0.18$), with $\alpha = 2$ and $\beta = 0.5$ in (a) and (c).

2018) by itself and reaches the edge of stability only at

$$B_{\text{crit}} \approx \frac{\operatorname{Tr} \Lambda}{\mathcal{K}^\star \lambda_1}, \tag{16}$$

which exceeds both lazy values because $\mathcal{K}^\star < \mathcal{K}_{\text{lin}} < 1$. SGD confirms both the self-tuned effective learning rate (Figure 1a and Figure 5b) and the larger critical batch size (Figure 5c).

Learning Rate Annealing Removes the Amplified Noise. To understand why annealing helps real networks so much and how long the decay should last, recall that after self-stabilization minibatch noise accounts for the fraction $\mathcal{K}^\star = 1 - 2^{-\beta}$ of the loss at any learning rate. Since $\mathcal{K} \propto \eta k$ falls as soon as the learning rate decays while k changes only slowly, annealing removes this noise and lowers the loss by up to 2^β at any learning rate in the rich regime, which gives a mechanism for the large drops observed when real networks are annealed (Hägele et al., 2024), whereas in the lazy regime it helps only at large η/B. At the same learning rate, a linear decay to zero over the last 20% of training lowers the rich loss by a factor that grows with β as the closure predicts, and barely changes the lazy loss (Figure 6a,b). Rich training also tolerates the decays over the last 10 to 20% of training that work well in practice (Hägele et al., 2024), since the kernel grows again as the learning rate decays and preserves part of the progress per step, whereas lazy training prefers a much shorter decay (Figure 6c).

7 Discussion

We set out to understand how evolving curvature constrains the scaling of training, a question that linear models of scaling laws cannot address. In a two-layer linear network, feature learning sharpens the kernel at a rate set by the task, and minibatch noise self-stabilizes this sharpening at a level set by the task alone, a feedback that caps the progress per sample, turns the accelerated exponent of infinite-batch theory into a transient and returns the compute-optimal exponent to its lazy value. The same feedback gives a mechanism for the robustness of real networks to the learning rate, the self-tuned scaling of the learning rate with the batch size and the gains from annealing. Natural extensions of this analysis include SGD with momentum, adaptive optimizers and networks with more than two layers, each of which changes how the kernel evolves and how minibatch noise enters its dynamics.

Reproducibility Statement

All experiments simulate SGD on the synthetic task of Section 3, which is fully specified by $(\alpha, \beta, N, V, \gamma, \eta, B, t)$ and a random seed. Section F lists the settings behind every figure and describes how the closure is solved numerically, and the code for the simulations, the closure and all figures is included in the supplementary material.

REFERENCES

Atish Agarwala and Jeffrey Pennington. High dimensional analysis reveals conservative sharpening and a stochastic edge of stability. *arXiv preprint arXiv:2404.19261*, 2024.

Atish Agarwala, Fabian Pedregosa, and Jeffrey Pennington. Second-order regression models exhibit progressive sharpening to the edge of stability. In *International Conference on Machine Learning*, 2023.

Arseniy Andreyev and Pierfrancesco Beneventano. Edge of stochastic stability: Revisiting the edge of stability for SGD. *arXiv preprint arXiv:2412.20553*, 2024.

Blake Bordelon and Cengiz Pehlevan. Learning curves for SGD on structured features. In *International Conference on Learning Representations*, 2022.

Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. In *International Conference on Machine Learning*, 2024.

Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. How feature learning can improve neural scaling laws. In *International Conference on Learning Representations*, 2025.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, 2019.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *International Conference on Learning Representations*, 2023.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

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 *Advances in Neural Information Processing Systems*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020.

Dayal Singh Kalra, Tianyu He, and Maissam Barkeshli. Universal sharpness dynamics in neural network training: Fixed point analysis, edge of stability, and route to chaos. In *International Conference on Learning Representations*, 2025.

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

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: The catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Licong Lin, Jingfeng Wu, Sham M. Kakade, Peter L. Bartlett, and Jason D. Lee. Scaling laws in linear regression: Compute, parameters, and data. In *Advances in Neural Information Processing Systems*, 2024.

Alexander Maloney, Daniel A. Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Elliot Paquette, Courtney Paquette, Lechao Xiao, and Jeffrey Pennington. 4+3 phases of compute-optimal neural scaling laws. In *Advances in Neural Information Processing Systems*, 2024.

Shikai Qiu, Lechao Xiao, Andrew Gordon Wilson, Jeffrey Pennington, and Atish Agarwala. Scaling collapse reveals universal dynamics in compute-optimally trained neural networks. In *International Conference on Machine Learning*, 2025.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

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

Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112), 2019.

Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, 2020.

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale transformer training instabilities. In *International Conference on Learning Representations*, 2024.

Geonhui Yoo, Minhak Song, and Chulhee Yun. Understanding sharpness dynamics in NN training with a minimalist example: The effects of dataset difficulty, depth, stochasticity, and more. In *International Conference on Machine Learning*, 2025.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, Christopher J. Shallue, and Roger Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, 2019.

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

A Matrix and Scalar Parameterizations

For the matrix model $w = Qu$, the gradients are $\nabla_u \mathcal{L} = Q^\top g$ and $\nabla_Q \mathcal{L} = g\, u^\top$ with $g = \nabla_w \mathcal{L}$. Under gradient flow, $\dot{u} = -\eta Q^\top g$ and $\dot{Q} = -\gamma\eta\, g u^\top$, so

$$\dot{w} = \dot{Q}u + Q\dot{u} = -\eta\big(QQ^\top + \gamma\|u\|^2 I\big)g = -\eta M g, \tag{17}$$

and

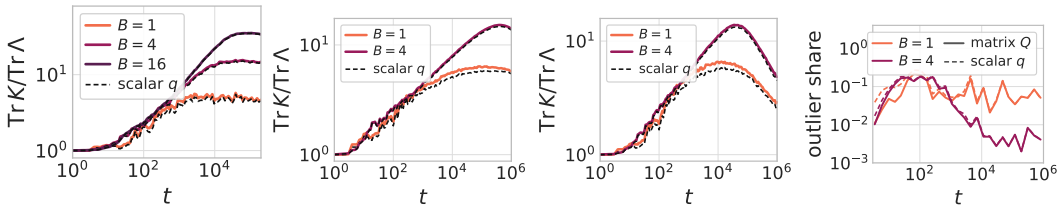
$$\dot{M} = \dot{Q}Q^\top + Q\dot{Q}^\top + \gamma\tfrac{d}{dt}\|u\|^2 I = -\gamma\eta\big(gw^\top + wg^\top + 2(w^\top g)I\big). \tag{18}$$

For the scalar model $w = qu$, $\dot{u} = -\eta q\, g$ and $\dot{q} = -\gamma\eta\, u^\top g$, so $\dot{w} = -\eta(q^2 I + \gamma u u^\top)g = -\eta M g$ with $M = q^2 I + \gamma u u^\top$, and

$$\dot{M} = 2q\dot{q}\, I + \gamma(\dot{u}u^\top + u\dot{u}^\top) = -\gamma\eta\big(gw^\top + wg^\top + 2(w^\top g)I\big), \tag{19}$$

identical to Equation (18). The matrix model is the model of Bordelon et al. (2025) (their Appendix B) with the random projection of the features replaced by truncation, and in their notation our u, $Q^\top$ and x correspond to w, B and ψ_∞, up to their output prefactor $1/N$, while their loss omits our factor $\frac{1}{2}$. With $M(0) = I$ in both cases, the two models follow the same trajectory $(w(t), M(t))$ under gradient flow. Under SGD, where g is replaced by the minibatch gradient g_B, the updates of w and M remain the same to first order in η, and the models differ only through second-order products of the parameter updates within a single step.

Both models carry the same rank-one outlier along w (Figure 2a), which Equation (7) fixes at $M = kI + (\gamma/k)ww^\top$ under gradient flow. Its direct effect on the loss is measured by its share of the mean loss-decrease rate $g^\top M g$, namely $\delta\,(v^\top g)^2/g^\top M g$, where δ is the outlier's excess over the bulk, v its eigenvector and g the full-batch gradient. This share peaks near 20% during early sharpening and falls to a few percent once the kernel has grown, identically in the two models (Figure 7d), which justifies neglecting its effect on the residual beyond the prefactor of Equation (8), through which it slows the growth of the kernel.



(a) $\alpha = 2, \beta = 0.2$ (b) $\alpha = 2, \beta = 0.5$ (c) $\alpha = 1.5, \beta = 0.4$ (d) Weight of the outlier

Figure 7: The matrix and scalar parameterizations, trained on identical minibatches with $N = 1024$. (a–c) Their kernel traces agree closely across batch sizes and for three tasks, to within about 2% at late times in (a) and within about 10% at $B = 1$ in (b,c), at $\eta = 0.033$ and $\gamma = 0.75$ in (a) and at $\gamma = 1$ with $\eta = 0.05$ and 0.03 in (b) and (c). (d) The share of the mean loss-decrease rate carried by the outlier of M is the same in both models, peaks near 20% during early sharpening and falls to a few percent afterwards ($\alpha = 2$, $\beta = 0.2$).

B Bayes-Optimal Error from D Samples

Write $a_i = \sqrt{\lambda_i}\, w_i^\star$ and draw the target from a Gaussian prior with independent components of variance $s_i = \lambda_i (w_i^\star)^2$, sorted in decreasing order, so that the loss is $\frac{1}{2}\|a - \hat{a}\|^2$ and each sample provides the noiseless measurement $y = z^\top a$ with $z \sim \mathcal{N}(0, I)$. Given D measurements $y = Za$ with $Z \in \mathbb{R}^{D\times V}$, the posterior mean $\hat{a}$ attains the Bayes-optimal error

$$\mathbb{E}\|a - \hat{a}\|^2 = \operatorname{Tr} S - \operatorname{Tr} P, \qquad P = SZ^\top (ZSZ^\top)^{-1} ZS, \tag{20}$$

where $S = \operatorname{diag}(s)$. The matrix P is positive semidefinite, has rank at most D, and satisfies $P \preceq S$ since $S - P$ is the posterior covariance, so that $\operatorname{Tr} P \le \sum_{i\le D} \lambda_i(P) \le \sum_{i\le D} s_i$ and therefore

$$\tfrac{1}{2}\,\mathbb{E}\|a - \hat{a}\|^2 \;\ge\; \tfrac{1}{2} \sum_{i>D} s_i \;\sim\; D^{-\alpha\beta}. \tag{21}$$

Any learning algorithm that uses D samples is thus at least this far from the target on average over the prior, and label noise, such as the unlearnable tail at finite N, can only increase the error.

C Derivation of the Two-Variable Closure

With $M = kI$, a minibatch step moves the residual by $\Delta \leftarrow \Delta + \eta k\, \Lambda^{1/2} g_B$, whose mean is $-\eta k \Lambda \Delta$, and for Gaussian features the per-sample noise in $\Lambda^{1/2} g_B$ has covariance $2\mathcal{L}\,\Lambda^2 + \Lambda\Delta\Delta^\top\Lambda$. Dropping the rank-one term and working in continuous time, the covariance of mode i obeys

$$\frac{d\Sigma_i}{dt} = -2\eta k \lambda_i\, \Sigma_i + \frac{\eta^2 k^2 \lambda_i^2}{B}\, 2\mathcal{L}. \tag{22}$$

Since the top modes relax within a time $1/(\eta k \lambda_1)$, much faster than k and $\mathcal{L}$ change, the covariance tracks its stationary value $\Sigma_i = \eta k \lambda_i \mathcal{L}/B$, so that $\operatorname{Tr}\Sigma = (\eta k \operatorname{Tr}\Lambda/B)\,\mathcal{L} = 2\mathcal{K}\mathcal{L}$, and combining

this with $\mathcal{L} = \mathcal{L}_0 + \frac{1}{2}\operatorname{Tr}\Sigma$ gives $\mathcal{L} = \mathcal{L}_0/(1-\mathcal{K})$ and $\operatorname{Tr}\Sigma = 2\mathcal{K}\mathcal{L}_0/(1-\mathcal{K})$, which is Equation (9). The mean residual obeys $d\bar{\Delta}/dt = -\eta k \Lambda \bar{\Delta}$, which becomes the lazy flow in the clock θ. For the kernel, the coefficient of I in Equation (5) is $-2\gamma\eta\, w^\top g_B$, and using $\Lambda^{1/2} w = \Delta(0) - \Delta$ and $\Lambda^{1/2} g = -\Lambda\Delta$ together with the decomposition $\Delta = \bar{\Delta} + \xi$, where $\mathbb{E}\,\xi\xi^\top = \Sigma$, the average of $-w^\top g$ equals $\bar{\Delta}^\top(\Delta(0) - \bar{\Delta}) - \operatorname{Tr}\Sigma$ to leading order.

To see that Equation (5) preserves Equation (7), we write it in continuous time, $dw/dt = -\eta M g$ and $dM/dt = -\gamma\eta\,(g w^\top + w g^\top + 2(w^\top g) I)$, and substitute $M = kI + (\gamma/k) w w^\top$ with $dk/dt = -2\gamma\eta\, w^\top g$. Since $dw/dt = -\eta\,(kg + (\gamma/k)(w^\top g)\, w)$, the derivative of $(\gamma/k) w w^\top$ equals $-\gamma\eta\,(g w^\top + w g^\top)$ plus the rank-one term $-2\gamma^2\eta\,(w^\top g)\, w w^\top/k^2$, which cancels against the contribution $2\gamma^2\eta\,(w^\top g)\, w w^\top/k^2$ from dk/dt, so the rank-two part of Equation (5) is reproduced for any g. Along this solution $d\|w\|^2/dt = -2\eta\, w^\top M g = -2\eta\,(k + \gamma\|w\|^2/k)\, w^\top g$, so that $\Phi = k(k-1) - \gamma\|w\|^2$ obeys $d\Phi/dt = (\Phi/k)\, dk/dt$ and remains zero from $\Phi(0) = 0$, which is the second relation in Equation (7).

The spike adds the term $-\eta(\gamma/k)(w^\top g)\, w$ to the update of w, which the lazy dynamics of the mean residual omit. Instead of tracking it, we require the learned function to follow the lazy dynamics in the clock θ, along which $d\|w\|^2/d\theta = 2(G - \operatorname{Tr}\Sigma)$. Along the true dynamics, the relation above gives $d\|w\|^2/d\theta = 2\,\frac{2k-1}{k}\,\mathbb{E}[-w^\top g]$, since $\gamma\|w\|^2/k = k - 1$, and equating the two replaces $\mathbb{E}[-w^\top g]$ by $\frac{k}{2k-1}(G - \operatorname{Tr}\Sigma)$, which yields Equation (8). For the scalar gate, the first relation in Equation (7) holds by construction, and SGD follows the second, $k(k-1) = \gamma\|w\|^2$, throughout the growth phase and departs from it only after self-stabilization, where minibatch noise at second order in η feeds the spike.

For power-law tasks, the sums take the form $\sum_i \Delta_i(0)^2 e^{-c\lambda_i\theta} \approx \frac{\Gamma(\beta)}{\alpha}(c\theta)^{-\beta}$, which gives $G = \frac{\Gamma(\beta)}{\alpha}\big(\theta^{-\beta} - (2\theta)^{-\beta}\big)$ and $2\mathcal{L}_0 = \frac{\Gamma(\beta)}{\alpha}(2\theta)^{-\beta}$, so that $G/(2\mathcal{L}_0) = 2^\beta - 1$ and $\mathcal{K}^\star = 1 - 2^{-\beta}$. The crossover time used in the figures is where the two terms of Equation (14) meet when their constants are kept, $t_\times = \big[(B\mathcal{K}^\star/\operatorname{Tr}\Lambda)\,(\frac{1+\beta}{2}\eta A)^{-2/(1+\beta)}\big]^{(1+\beta)/(1-\beta)}$ with $A^2 = 2\gamma c_G/(1-\beta)$, which scales as Equation (12). The theory curves in Figures 2 to 5 solve the same closure with the full noise kernel norm $\mathcal{K} = \frac{1}{B}\sum_i \eta k\lambda_i/(2 - (1 + 1/B)\eta k\lambda_i)$ for Gaussian features, which reduces to the trace form when $\eta k\lambda_1 \ll 1$.

D Finite Model Size

The most direct effect of model size is the model bottleneck: when the target extends beyond the top N eigenfeatures ($V > N$), the model cannot represent the part $\mathcal{L}_\infty(N) = \frac{1}{2}\sum_{i>N}\lambda_i (w_i^\star)^2 \propto N^{-\alpha\beta}$, which reaches it as label noise of variance $2\mathcal{L}_\infty(N)$ and sets the floor of the compute-optimal trade-off in Section 5. This label noise also adds to the minibatch covariance, and we account for its effect on the loss only through the additive form $\mathcal{L} \approx \mathcal{L}_{\mathrm{cap}}(D) + c\, N^{-\alpha\beta}$, which the compute-optimal envelopes of Figure 4c and Figure 12c follow.

Model size also corrects the self-stabilized noise level, even when the target lies within the span of the model ($N = V$), as in Sections 4 and 6. The value $\mathcal{K}^\star = 1 - 2^{-\beta}$ follows from the power laws $G \approx c_G\,\theta^{-\beta}$ and $\mathcal{L}_0 \approx c_{\mathcal{L}}\,\theta^{-\beta}$, which assume that the modes still to be learned extend indefinitely beyond the learning frontier $\lambda_i\theta \approx 1$. At finite N the modes beyond N are missing, and their share of the loss at clock θ is of order $(\theta/N^\alpha)^\beta$, which decays only slowly with N when $\alpha\beta$ is small. Since $\mathcal{L}_0$ then falls faster than its power law while the drive is less affected, the ratio $G/(2\mathcal{L}_0)$ rises above $2^\beta - 1$, and the kernel self-stabilizes at the finite-N level

$$\mathcal{K}_N^\star = \frac{G_N}{2\mathcal{L}_{0,N} + G_N}, \tag{23}$$

where G_N and $\mathcal{L}_{0,N}$ are the drive and the noiseless loss with the sums truncated at N, which exceeds $1 - 2^{-\beta}$ and drifts slowly upward as the clock advances. The closure solved at the same N captures this drift, and the noise kernel norm measured in SGD follows it also in the settings where it departs visibly from $1 - 2^{-\beta}$ (Figure 9c). For Figure 2c we therefore select the settings before training from the closure at $N = 2^{20}$, keeping those in which it stays within 3% of $1 - 2^{-\beta}$ over the averaging window, which removes the settings with small $\alpha\beta$, and we also remove the settings at large β and

small η whose plateau is reached too late for our longest runs. Since the correction scales as $(\theta/N^{\alpha})^{\beta}$, the smaller $N = 2^{16}$ suffices for the landscapes at $\alpha\beta = 1$ in Section 6.

E Additional Experiments

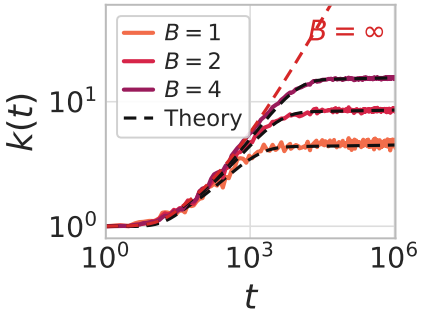
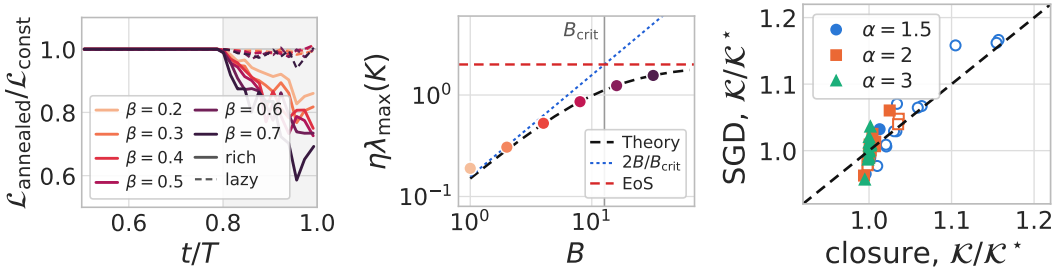
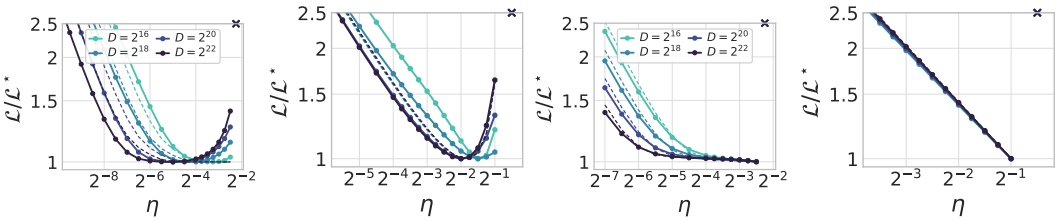


Figure 8: The kernel scale k measured in SGD (solid) follows the closure (dashed) at every batch size, first growing along the noiseless infinite-batch prediction (red) and then saturating at a plateau $k_{\text{eq}} \propto B$. Here $\alpha = 2$, $\beta = 0.2$, $\gamma = 0.75$, $\eta = 0.033$ and $N = V = 2^{20}$.



(a) Annealing trajectories (b) Sharpness against B (c) Finite-N corrections to $\mathcal{K}^\star$

Figure 9: (a) Ratio between the annealed and constant-learning-rate loss during the last half of training, for the settings of Figure 6b, in the rich (solid) and lazy (dashed) regimes. (b) Self-stabilized sharpness $\eta\lambda_{\max}(K)$ of SGD against B (circles, $\eta = 0.0625$, $\beta = 0.2$, $N = V = 2^{18}$), which follows the closure (dashed) at every batch size, rising as $2B/B_{\text{crit}}$ (blue) well below B_{crit} and approaching the edge of stability (red) above it. (c) Measured noise kernel norm at self-stabilization against the closure solved at the same $N = 2^{20}$, including the settings excluded from Figure 2c because of finite-N corrections (open markers).



(a) Rich, constant η (b) Lazy, constant η (c) Rich, annealed (d) Lazy, annealed

Figure 10: The learning rate landscape of Figure 5a at every horizon, relative to its minimum over η, for D from 2^{16} to 2^{22} (light to dark), at a constant learning rate (a, b) and with a linear decay over the last 20% (c, d), with the closure (dashed) and crosses marking divergence ($B = 1$, $\alpha = 2$, $\beta = 0.5$, $N = V = 2^{16}$). The lower edge of the rich basin moves to smaller η as training proceeds, while the lazy optimum stays in place.

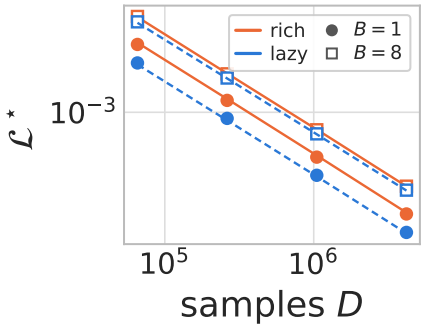
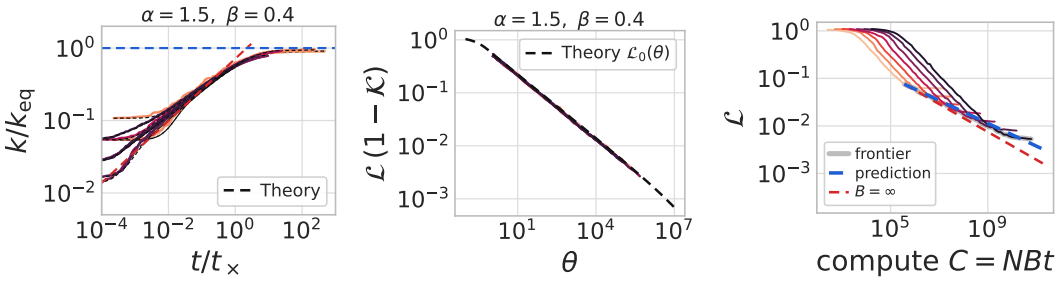


Figure 11: With annealing, lazy training at its best learning rate reaches a lower loss than rich training at small batch sizes. The best annealed loss over η after D samples (markers: SGD, lines: closure) at $B = 1$ (filled) is lower in the lazy regime (blue) than in the rich regime (orange) at every horizon, because the best lazy learning rate sits at the stability edge, where the progress per sample approaches its maximum $2/\operatorname{Tr}\Lambda$, while self-stabilization pins the rich model at $\mathcal{K}^\star < 1$. At $B = 8$ (open), close to B_{crit}, the two coincide, as Section 6 predicts ($\alpha = 2$, $\beta = 0.5$, $N = V = 2^{16}$).



(a) Kernel collapse (b) Collapse in the kernel clock (c) Compute frontier

Figure 12: The second task of Figure 3d ($\alpha = 1.5$, $\beta = 0.4$). (a) Kernel trajectories of ten SGD runs with $\eta \in [0.003, 0.02]$, $\gamma \in [0.25, 4]$ and $B \in \{1, 2, 4\}$ collapse in $t/t_\times$, each on its closure (dashed). (b) The loss collapses onto $\mathcal{L}_0(\theta)$ in the kernel clock computed from the measured k, so the loss depends on the kernel only through the clock. (c) SGD loss curves against compute for model sizes N from 64 to 4096 with $V = 32N$, whose lower envelope (gray) follows the predicted compute-optimal law (blue), shallower than the infinite-batch law (red).

F Experimental Details

SGD Simulations. We simulate online SGD on the task of Section 3 with the scalar gate $f(x) = (qu)^\top x_{1:N}$, initialized at $q = 1$ and $u = 0$. At every step we draw B fresh inputs $x \sim \mathcal{N}(0, \Lambda)$ in $\mathbb{R}^V$ with $\lambda_i = i^{-\alpha}$, label them with $y = w^{\star\top} x$ where $\lambda_i (w_i^\star)^2 = i^{-(\alpha\beta+1)}$, and update u with learning rate η and q with learning rate $\gamma\eta$ as in Equation (4). Since the task is Gaussian and linear, we evaluate the population loss exactly as $\mathcal{L} = \frac{1}{2}\sum_{i \le V} \lambda_i (w_i^\star - w_i)^2$ with $w_i = q u_i$ for $i \le N$ and $w_i = 0$ otherwise, at 200 to 300 logarithmically spaced times, and we record the kernel scale $k = q^2$ and the sharpness $\eta\lambda_{\max}(K)$. Loss curves shown with a single line average two seeds unless stated otherwise, and a run is marked as diverged if its loss becomes non-finite. The simulations are written in JAX in single precision and run on GPUs and single TPU v6e chips.

Model and Target Sizes. In Sections 4 to 6 we take $N = V$, so that the target lies entirely within the span of the model and there is no irreducible loss, and we choose N large enough that the learning frontier stays far from N over the training horizon. We use $N = 2^{20}$ unless stated otherwise, and $N = 2^{16}$ for $\beta = 0.5$ in Figure 5, where finite-N corrections are negligible since $\alpha\beta = 1$. The comparison between parameterizations in Figure 2a and Figure 7 uses $N = 1024$, since the matrix model has N^2 parameters, with the part of the target beyond the top N modes entering the labels as Gaussian noise of the same variance. The compute-optimal experiments in Figure 4c and Figure 12c

vary N with $V = 32N$, so that the unlearnable part of the target is the same fraction of its $V = \infty$ value at every N.

Settings of the Remaining Experiments. The kernel and loss collapses (Figure 2d, Figure 3 and Figure 12) use runs of 2^{20} steps, or 10^6 steps for the first task, with two seeds per setting. The learning rate landscapes (Figure 1c, Figure 5a and Figure 10) sweep $\eta = 2^{j/2}$ for $j = -20, \dots, 2$, with quarter-octave steps near the divergence threshold, at $B \in \{1, 8\}$, $\alpha = 2$, $\beta = 0.5$, $\gamma \in \{0, 1\}$ and $N = V = 2^{16}$, with one constant-rate run to $D = 2^{22}$ samples, read at each horizon, and one run annealed over the last 20% of each horizon $D \in \{2^{16}, 2^{18}, 2^{20}, 2^{22}\}$, treating a run as diverged once its loss exceeds twice its initial value. The critical batch size (Figure 5c) uses runs annealed over the last 20% of $D \in \{2^{18}, 2^{20}, 2^{22}\}$ samples for $\eta = 2^{j/2}$ with $j = -12, \dots, 3$ and $B \in \{1, 2, \dots, 256\}$ at $\alpha = 1.5$, $\beta = 0.2$, $\gamma \in \{0, 1\}$ and $N = V = 2^{18}$. For each B and horizon we take the best loss over η, interpolate $\log \mathcal{L}$ linearly in $\log D$ between horizons, and read off the number of samples at which it reaches the best loss at $B = 1$ and $D = 2^{18}$. The effective learning rates in Figure 5b come from constant-rate runs with the same learning rates at $B \in \{1, 4, 16, 64\}$ and $D = 2^{22}$, taking the median of ηk over the last 20% of training and omitting diverged runs. The loss curves in Figure 1b use $\alpha = 2$, $\beta = 0.2$, $\gamma = 8$, $\eta = 0.01$, $N = V = 2^{20}$ and 2^{20} steps with four seeds per batch size, and the annealing curves in Figure 6a use $B = 2$, $\eta = 0.0625$, $\alpha = 2$, $\beta = 0.5$, $\gamma \in \{0, 1\}$ and $N = V = 2^{16}$ with eight seeds, showing the mean over seeds, which is what the closure predicts, since the constant-rate rich loss has a heavy upper tail. The decay lengths in Figure 6c use $B = 8$, $\eta = 0.18$, $\alpha = 2$, $\beta = 0.5$, $\gamma \in \{0, 1\}$, $N = V = 2^{16}$ and $D = 2^{24}$ samples with twelve seeds per decay fraction, and report the loss averaged over the last 1% of training. The sharpness in Figure 1a and Figure 9b comes from runs of 2^{18} steps at $\eta = 0.0625$, $\gamma = 0.75$ and $N = V = 2^{18}$ for each batch size.

Measuring the Self-Stabilized Noise Level. For each task and setting in Figure 2c we choose the learning rate so that the predicted plateau is $k_{\mathrm{eq}} \in \{6, 12\}$ with $\gamma = 1$, which keeps the fluctuations of k itself small, and we define t_p as the time at which the closure at $N = \infty$, solved without the prefactor $2k/(2k-1)$ in Equation (8), first comes within 3% of $\mathcal{K}^\star$. Since the prefactor delays the approach to the plateau, we run each setting to $40\,t_p$ and average the noise kernel norm $\mathcal{K}(\eta k)$ over $[10\,t_p, 40\,t_p]$, using the full expression for Gaussian features evaluated at the measured k. We show in Figure 2c only the settings for which the closure at $N = 2^{20}$ stays within 3% of $\mathcal{K}^\star$ over this window, and Figure 9c shows that the remaining settings, at small $\alpha\beta$ where the tail of the target converges slowly, agree with the closure solved at the same N. The late-time exponents in Figure 4a come from the same runs, as the least-squares slope of $\log \mathcal{L}$ against $\log t$ from $10\,t_p$ to the end of training, where t_p is the plateau time used above, averaged over seeds.

Measuring the Annealing Gain. The annealed runs decay the learning rate linearly to zero over the last 20% of training and are otherwise identical to the constant-rate runs of the noise-level experiment at $\alpha = 2$, $B = 1$ and $k_{\mathrm{eq}} = 6$, which have converged to $\mathcal{K}^\star$ long before the decay begins. Since the loss of a constant-rate run at $B = 1$ fluctuates from step to step, we evaluate it at the final time from a power-law fit over $[0.3T, 0.8T]$, while the annealed loss is smooth once the learning rate has decayed and we average it over the last 1% of training, with six seeds for the rich runs and two for the lazy runs.

Fitting the Compute-Optimal Frontier. We average the loss curves of two seeds for each model size, form their lower envelope as a function of $C = NBt$, and fit a power law to the envelope from the point at which the smallest model comes within 30% of its final loss, since at smaller compute the envelope traces the initial transient of the smallest model.

Solving the Closure. The theory curves solve Equations (8) and (9) for (θ, k) with an adaptive ODE solver. We tabulate $G(\theta)$ and $\mathcal{L}_0(\theta)$ on a logarithmic grid, summing the first 2^{16} modes exactly and the remaining modes up to N as an integral, which gives incomplete gamma functions, and we use the noise kernel norm for Gaussian features $\mathcal{K} = \frac{1}{B}\sum_i \eta k \lambda_i / (2 - (1 + 1/B)\eta k \lambda_i)$ in place of its trace approximation. Learning rate schedules enter through $\eta(t)$ in both $d\theta/dt = \eta k$ and $\mathcal{K}$.

AI USE STATEMENT

We used AI for coding assistance, including the implementation of experiments and making figures, and to assist with calculations and to polish writing.