UNDERSTANDING LEARNING-RATE SCALING WITH BATCH SIZE IN LORA

Anonymous authors
Paper under double-blind review

ABSTRACT

LoRA and full fine-tuning (FullFT) exhibit different learning-rate scaling behaviors as batch size increases, with LoRA requiring substantially larger learning rates at large batch sizes. The origin of this difference is not well understood and is often attributed to the low-rank nature of LoRA. We show that the gap reflects a learning-rate scaling mismatch rather than a capacity deficit. Starting from the coupled updates of the low-rank factors, we derive a state-dependent linear operator that characterizes the induced weight-space dynamics and the resulting effect on local curvature. This analysis reveals that LoRA changes the relative retention of gradient signal and stochastic noise, leading to a larger effective gradient-noise scale and consequently a different batch-size dependence of the optimal learning rate. Building on this mechanism, we derive a closed-form learning-rate scaling law that captures the dependence on batch size, rank, and model width. Fitted on the measured optima, the law accurately reproduces the observed scaling and, with no refitting, transfers to unseen models of different widths and architectures. A two-point calibration scheme derived from the law reduces learning-rate search cost by over $30\times$.

1 INTRODUCTION

LoRA Hu et al. (2022) has become the standard approach to parameter-efficient fine-tuning because it achieves performance close to full fine-tuning (FullFT) while updating only a small fraction of model parameters. As models and datasets continue to scale, however, an increasingly important difference emerges: LoRA and FullFT no longer behave similarly under large-batch training. Although FullFT benefits from well-established learning-rate scaling rules, directly transferring these rules to LoRA consistently leads to degraded optimization and a persistent performance gap Schulman et al. (2025). Why LoRA requires a fundamentally different learning-rate scaling behavior remains poorly understood.

A natural explanation attributes this discrepancy to the low-rank parameterization itself. Since LoRA optimizes only a restricted subspace, it is tempting to view the large-batch gap as an inevitable consequence of limited capacity. This interpretation has motivated a series of methods that redesign LoRA's parameterization or optimizer to compensate for the presumed structural deficiency Wang et al. (2025); Yen et al. (2025). Yet these methods do not fully close the gap. More importantly, the underlying premise is difficult to reconcile with empirical evidence: if insufficient rank were the fundamental cause, the gap should gradually disappear as the rank increases. Instead, it remains remarkably stable across ranks Schulman et al. (2025), suggesting that the dominant factor lies elsewhere.

In this work, we revisit the problem from the perspective of learning dynamics rather than parameterization. Our starting observation is that LoRA and FullFT obey different optimal learning-rate scaling laws. As the batch size grows, LoRA requires substantially larger learning rates than FullFT, and applying FullFT's scaling rule systematically underestimates LoRA's optimal learning rate. Once the scaling law is measured and adapted accordingly, the large-batch gap largely disappears. This observation shifts the central question from how to modify LoRA's parameterization to *why* LoRA follows a different learning-rate scaling law in the first place.

We answer this question by deriving the operator that governs LoRA's weight-space dynamics. Starting from the coupled updates of the low-rank factors, we show that LoRA induces a state-dependent

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.

linear operator that acts as a subspace preconditioner on the Hessian. Spectral analysis reveals that this operator asymmetrically filters gradient signal, which is concentrated in sharp curvature directions, far more than stochastic noise, which is diffusely distributed, inflating the gradient noise scale in proportion to the ratio of model width to adapter rank. This mechanism yields a closed-form scaling law that captures the dependence on batch size, rank, and model width, which we develop in Section 3 and validate in Section 4.

Beyond explaining the phenomenon, the scaling law enables a practical calibration scheme: fitting the law at two batch sizes suffices to predict the optimal learning rate across all configurations, reducing search cost by over $30\times$.

Contributions. (1) The large-batch gap between LoRA and FullFT is a learning-rate scaling mismatch, not a capacity deficit. (2) We derive the Sylvester operator governing LoRA's weight-space dynamics and show, via spectral analysis, that it filters gradient signal far more than noise, inflating the noise scale in proportion to the width-to-rank ratio d/r and yielding a closed-form scaling law with three interpretable parameters. (3) We establish the generality of the law: with no refitting, it predicts the optimal learning rates of unseen models of different widths and architectures, with the exponent $s \approx 0.5$ stable across the fitted models. (4) A two-point calibration scheme predicts optimal learning rates from about 12 runs, cutting search cost by over $30\times$.

2 Related Work

Learning-rate scaling. Large-batch optimization has been studied through gradient noise scale and learning-rate scaling laws Goyal et al. (2017); McCandlish et al. (2018). SDE analyses reveal that Adam's noise structure yields a square-root rather than linear small-batch scaling Malladi et al. (2022); Compagnoni et al. (2025), while Li et al. (2024) show non-monotonic optimal learning rates in batch size. Whether these principles extend to parameter-efficient methods such as LoRA remains unexplored.

Learning rate and variants in LoRA. Several studies address how LoRA's learning rate interacts with its structure: LoRA+ Hayou et al. (2024) assigns different rates to factor matrices, rsLoRA Kalajdzievski (2023) normalizes by $\sqrt{r}$, Chen et al. (2026) derive rank–width scaling regimes via maximal-update parametrization, Zhang et al. (2026) analyze the scaling factor α through a signal–drift framework, and Lee et al. (2026) show that proper LR tuning eliminates performance differences among LoRA variants. Other variants, including DoRA Liu et al. (2024), LoRA-Pro Wang et al. (2025), LoRA-RITE Yen et al. (2025), and LoRAM Zhang et al. (2025), improve performance through architectural or algorithmic changes. Our work differs by showing that LoRA's low-rank parameterization itself induces a state-dependent preconditioning effect on the Hessian, and by deriving a scaling law for the optimal learning rate as a function of batch size, rank, and model width.

3 A Theory of Learning-Rate Scaling for LoRA

Our central hypothesis is that low-rank parameterization does not merely restrict the search space but actively induces a distinct optimization geometry. Figure 1 summarizes the resulting framework.

3.1 LoRA's Effective Hessian

The key difficulty in analyzing LoRA is that it optimizes the low-rank factors A and B rather than the weight matrix W directly, so its dynamics cannot be compared with FullFT in parameter space. To establish a common geometric view, we derive the equivalent first-order update in weight space.

Let $\mathcal{L}(W)$ denote the training loss and $G = \nabla_W \mathcal{L}$ the weight-space gradient. LoRA parameterizes the update as $\Delta W = \alpha BA$ for a weight matrix $W \in \mathbb{R}^{m\times n}$, where α is the LoRA scaling factor, r is the rank, $A \in \mathbb{R}^{r\times n}$, and $B \in \mathbb{R}^{m\times r}$. The gradients of the two factors are $\nabla_B \mathcal{L} = \alpha G A^\top$ and $\nabla_A \mathcal{L} = \alpha B^\top G$. Let η denote the learning rate. A single SGD step updates the factors as $\Delta B = -\eta\alpha G A^\top$ and $\Delta A = -\eta\alpha B^\top G$. To recover the induced weight-space update $\Delta W = W_{t+1} - W_t$,

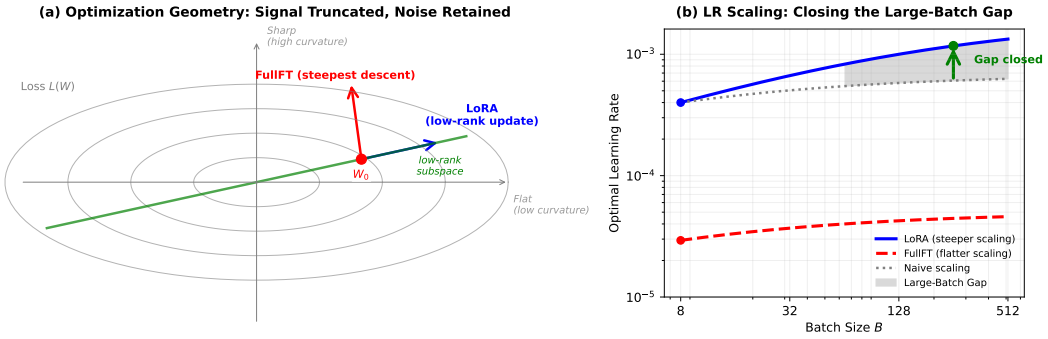


Figure 1: (a) LoRA's parameterization induces a state-dependent operator that restricts updates to a low-rank subspace, suppressing alignment with high-curvature Hessian directions and altering the optimal learning-rate scaling. (b) Applying FullFT's LR scaling law to LoRA yields a "naive" scaling that undershoots at large batch sizes, creating the Large-Batch Gap (gray region). Using the correct scaling law for LoRA closes the gap.

we expand $W_{t+1} = W_t + \alpha(B + \Delta B)(A + \Delta A)$ and keep first-order terms:

$$\Delta W \approx \alpha(\Delta B \cdot A + B \cdot \Delta A), \tag{1}$$

where the second-order term $\Delta B \cdot \Delta A = O(\eta^2)$ is omitted. Substituting the factor updates yields:

$$\Delta W = -\eta\alpha^2 \left(BB^\top G + GA^\top A\right). \tag{2}$$

Definition 1 (LoRA-induced Sylvester Operator)**.** *We define the LoRA-induced Sylvester operator* $\mathcal{P}_t : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ *as:*

$$\mathcal{P}_t[G] = BB^\top G + GA^\top A \tag{3}$$

so that $\Delta W = -\eta\alpha^2\mathcal{P}_t[G]$*, where* t *indexes the training step and the factors* $A(t), B(t)$ *evolve during training.*

We analyze the local landscape around $W(t)$ under two standard approximations: (i) *local quadraticity*, where the loss is approximated by its second-order Taylor expansion, $\mathcal{L}(W + \delta) \approx \mathcal{L}(W) + \langle G, \delta\rangle + \frac{1}{2}\langle\delta, H[\delta]\rangle$, with $G = \nabla_W\mathcal{L}$ and H the full-weight Hessian operator; (ii) *single-step factor stationarity*, where within one update step, $A(t)$ and $B(t)$ are treated as locally stationary when evaluating $\mathcal{P}_t$.

Lemma 1 (Properties of $\mathcal{P}_t$)**.** *The operator* $\mathcal{P}_t$ *is self-adjoint under the Frobenius inner product and positive semidefinite, admitting a unique PSD square root* $\mathcal{P}_t^{1/2}$*. Proof in Appendix A.1.*

By Lemma 1, $\mathcal{P}_t$ functions as a valid local preconditioner. Substituting the preconditioned update $\Delta W = -\eta\alpha^2\mathcal{P}_t[G]$ into the quadratic model directly yields the effective curvature governing LoRA's weight-space dynamics:

Theorem 1 (Effective Hessian under LoRA)**.** *Under local quadraticity and single-step factor stationarity, LoRA's instantaneous weight-space dynamics are equivalent to gradient descent governed by an effective Hessian:*

$$H_{\text{eff}}(t) = \alpha^2\mathcal{P}_t^{1/2}H\mathcal{P}_t^{1/2} \tag{4}$$

Proof in Appendix A.2.

3.2 Noise Scale and SGD Learning-Rate Scaling

With LoRA's effective geometry H_{eff} established, we now examine its impact on stochastic optimization dynamics. Under SGD, local curvature directly dictates the gradient noise scale B_{noise} McCandlish et al. (2018), defined as the ratio of noise variance to signal magnitude, which governs the optimal learning rate as a function of batch size B:

$$\eta^*(B) = \eta_{\max} \cdot \frac{B}{B + B_{\text{noise}}}. \tag{5}$$

This rational function, derived for SGD, yields linear scaling ($\eta^* \propto B$) when $B \ll B_{\text{noise}}$ and saturates to the maximum learning rate $\eta_{\max}$ when $B \gg B_{\text{noise}}$. We first develop the LoRA extension of this formula and then generalize it to AdamW in Section 3.3.

In FullFT, with Σ denoting the gradient noise covariance, the noise scale is $B_{\text{noise}}^{\text{FullFT}} = \text{tr}(\Sigma H)/(G^\top H G)$. Under LoRA, substituting the preconditioned gradient $\tilde{G} = \alpha \mathcal{P}_t^{1/2}[G]$, noise covariance $\tilde{\Sigma} = \alpha^2 \mathcal{P}_t^{1/2} \Sigma \mathcal{P}_t^{1/2}$, and effective Hessian H_{eff} yields:

$$B_{\text{noise}}^{\text{LoRA}} = \frac{\text{tr}(\tilde{\Sigma} H_{\text{eff}})}{\tilde{G}^\top H_{\text{eff}} \tilde{G}}. \tag{6}$$

To understand how LoRA transforms this noise scale relative to FullFT, we evaluate their ratio. Remarkably, the ratio decouples into Noise Retention over Signal Retention:

$$\frac{B_{\text{noise}}^{\text{LoRA}}}{B_{\text{noise}}^{\text{FullFT}}} = \frac{\overbrace{\text{tr}(\tilde{\Sigma} H_{\text{eff}})/\text{tr}(\Sigma H)}^{\text{Noise Retention}}}{\underbrace{\tilde{G}^\top H_{\text{eff}} \tilde{G} / G^\top H G}_{\text{Signal Retention}}}. \tag{7}$$

Because LoRA restricts updates to a low-rank subspace (LoRA's rank r is far smaller than the full weight dimension d), the effective Hessian H_{eff} acts as a subspace filter. How much signal and noise survive this filter depends on their distribution patterns. The following estimates are qualitative rather than proven; we confirm them experimentally in Section 4:

- **Signal Retention** ($\ll 1$): The true gradient G concentrates along a few sharp, high-curvature directions of H, which LoRA's rank-r subspace cannot fully align with, so H_{eff} truncates the gradient signal ($\tilde{G}^\top H_{\text{eff}} \tilde{G} \ll G^\top H G$).
- **Noise Retention** (≈ 1): Gradient noise Σ spreads diffusely across all dimensions and is not tied to specific curvature directions, so it survives the filter far better than the directional signal.

Because LoRA truncates the curvature-aligned signal far more aggressively than the diffuse noise, Eq. 7 guarantees $B_{\text{noise}}^{\text{LoRA}} > B_{\text{noise}}^{\text{FullFT}}$.

A larger noise scale B_{noise} means the optimal learning rate $\eta^*(B)$ in Eq. 5 stays in the linear scaling regime across much larger batch sizes. Consequently, LoRA requires a larger learning rate than FullFT at the same batch size; this is the core observation that motivates our study. Using FullFT's flatter scaling rule gives LoRA an undersized learning rate at large batch sizes, causing the performance drop.

As $r \to \infty$, $\mathcal{P}_t \to I$ and $H_{\text{eff}} \to H$, recovering FullFT geometry. This confirms that the large-batch gap is an optimization issue rather than a capacity limit: the fix is to adjust the learning rate schedule, not to modify LoRA's architecture.

3.3 From SGD to AdamW: Generalized Noise Scale

Equation 5 was derived for SGD McCandlish et al. (2018). In practice, however, AdamW is the standard optimizer for fine-tuning, whose adaptive normalization of the gradient by its running second moment fundamentally changes the noise structure. Recent stochastic differential equation (SDE) analyses Malladi et al. (2022); Compagnoni et al. (2025) show that the effective gradient noise under Adam scales as $\propto 1/\sqrt{B}$ rather than $1/B$ in the small-ϵ limit, where ϵ is Adam's numerical stability constant. This modifies the signal–noise balance and yields a square-root small-batch asymptote ($\eta^* \propto \sqrt{B}$) instead of SGD's linear one ($\eta^* \propto B$). In the opposite limit ($\epsilon \to \infty$), Adam reduces to SGD and the linear asymptote is recovered.

To capture both limits, we adopt a generalized noise-scale formula, i.e., the simplest monotonic form that recovers each limit at the corresponding endpoint:

$$\eta^*(B) = \eta_{\max} \cdot \frac{B^s}{B^s + B_{\text{noise}}^s}, \tag{8}$$

where the exponent $s \in [\frac{1}{2}, 1]$ is optimizer-dependent: $s = 1$ recovers SGD (Eq. 5), while $s = \frac{1}{2}$ corresponds to Adam in the $\epsilon \to 0$ (sign-gradient) limit. For practical $\epsilon > 0$, s interpolates between these extremes; its exact value depends on ϵ, the second-moment decay rate β_2, and the gradient magnitude distribution.

Crucially, the LoRA noise-scale theory developed in Section 3.2, including the spectral ratio (Eq. 7), the signal/noise retention decomposition, and the guarantee $B_{\text{noise}}^{\text{LoRA}} > B_{\text{noise}}^{\text{FullFT}}$, is independent of the optimizer. The exponent s is a property of the optimizer *alone*; LoRA and FullFT share the same s for a given optimizer. The LoRA–FullFT difference enters entirely through B_{noise}, which is determined by the optimization geometry ($H_{\text{eff}}, \tilde{\Sigma}, \tilde{G}$) and is unaffected by the choice of s. This separation is the key theoretical insight: the structural effect of LoRA on learning-rate scaling is captured by B_{noise}, not by s.

We adopt Eq. 8 with $s = \frac{1}{2}$ as a first-order approximation. In Section 3.4, we derive explicit rank- and width-dependent expressions for $\eta_{\max}$ and B_{noise}, and show that fitting Eq. 8 to data from six batch sizes yields $s \approx 0.52$, confirming the Adam limit and enabling joint estimation of all parameters.

3.4 Extending the Scaling Law with Width and Rank

The analysis in Sections 3.2–3.3 establishes that LoRA differs from FullFT through B_{noise} (a geometric property) and shares the same optimizer exponent s. However, B_{noise} and $\eta_{\max}$ in Eq. 8 remain unspecified functions of rank r and model width d. We now derive their explicit dependence, yielding a closed-form scaling law with only three free parameters.

Effect 1: Rank scaling ($\eta_{\max} \propto 1/r$). At initialization, the Kaiming-initialized factor A satisfies $\mathbb{E}[A^\top A] = (r/d)\, I_n$ (Marchenko–Pastur law). This attenuates the weight-space update by a factor of r/d, so matching FullFT's effective learning rate requires $\eta^*_{\text{LoRA}} \propto 1/r$ (with d and α absorbed into the constant η_0). See Appendix A.3 for the step-by-step derivation.

Effect 2: Noise inflation ($B_{\text{noise}}(r) = B_\infty \cdot d/r$). Under isotropic noise ($\Sigma \approx \sigma^2 I$), the noise scale depends on the ratio $\text{tr}(H)/(G^\top H G)$. Because neural-network Hessians are heavy-tailed, with a few eigenvalues dominating the trace, the gradient signal concentrates in these dominant directions. Projecting through LoRA's rank-r subspace captures a dominant direction with probability about r/d, attenuating the signal more than the isotropic noise and inflating the noise scale by a factor of d/r: $B_{\text{noise}}^{\text{LoRA}}/B_{\text{noise}}^{\text{FullFT}} = d/r$ (detailed derivation in Appendix A.3). We write:

$$B_{\text{noise}}(r) = B_\infty \cdot \frac{d}{r}, \tag{9}$$

where $B_\infty = B_{\text{noise}}^{\text{FullFT}}$ (recovered at $r = d$). Note that α cancels in the noise-scale ratio; B_{noise} is a geometric property independent of the scaling factor.

Complete scaling law. Substituting Effects 1 and 2 into Eq. 8:

$$\eta^*(B, r) = \frac{\eta_0}{r} \cdot \frac{B^s}{B^s + \left(B_\infty \cdot \frac{d}{r}\right)^s}, \tag{10}$$

with three free parameters (η_0, s, B_∞) and one known constant (d, the model's hidden size). FullFT corresponds to $r = d$, recovering $\eta^*(B) = (\eta_0/d) \cdot B^s/(B^s + B_\infty^s)$, i.e., Eq. 8 with $\eta_{\max} = \eta_0/d$.

The formula predicts two asymptotic regimes for the r-dependence of $\log_2 \eta^*$: a linear regime ($B \ll B_{\text{noise}}(r)$) with slope $s - 1 \approx -0.50$ and a saturation regime ($B \gg B_{\text{noise}}(r)$) with slope -1. As B increases, the transition point $B_{\text{noise}}(r) = B_\infty\, d/r$ shifts, causing the slope to steepen from about -0.50 toward -1. A pure power law $\eta^* \propto r^{-\beta}$ cannot produce B-dependent slopes; the multiplicative structure of Eq. 10 is required. Figure 4 visually confirms the predicted regime structure. (See Appendix A.3 for the asymptotic derivation.)

4 Experiments

We test the scaling law (Eq. 10) across multiple models and architectures. Section 4.1 details the experimental protocol. Sections 4.2–4.3 present three lines of evidence: (1) the three-parameter

model fits the data more accurately than power-law alternatives, (2) the exponent s is stable at ≈ 0.5 across all three models, and (3) a two-point calibration scheme enables practical LR prediction at $30\times$ reduced search cost.

4.1 Experimental Setup

Models and dataset. We use three models with two different hidden dimensions:

- **EXP-01**: Qwen2.5-0.5B ($d = 896$, 24 layers, 494M parameters) Qwen Team (2024)
- **EXP-02**: Llama-3.2-1B ($d = 2048$, 16 layers, 1.24B parameters) Meta AI (2024)
- **EXP-03**: Qwen2.5-3B ($d = 2048$, 36 layers, 3.09B parameters) Qwen Team (2024)

All use Alpaca-52K Taori et al. (2023) for supervised fine-tuning (sequence length 512, 8% validation split). The three models span two architectures (Qwen2.5 and Llama-3.2) and three scales (0.5B, 1B, 3B).

Training configuration. LoRA is applied to selected linear modules within each transformer block: MLP projections (gate_proj, up_proj, down_proj) in EXP-01 and EXP-02, expanded to additionally include attention projections (q_proj, k_proj, v_proj, o_proj) in EXP-03. Crucially, "FullFT" denotes full-rank fine-tuning of the *same* target modules: all parameters within those modules are unfrozen while the rest of the model (embeddings, layer norms, etc.) remains frozen, corresponding to the $r = d$ limit of LoRA. This ensures that LoRA and FullFT differ only in rank, not in which parameters are updated.

Ranks: $\{4, 16, 64, 256\}$, plus FullFT. The LoRA scaling factor is held constant across ranks. Optimizer: AdamW Loshchilov & Hutter (2019) ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay $= 0.01$, grad clip $= 1.0$). Schedule: linear warmup (5%), cosine decay to $0.1\times$ peak. Precision: bf16 with TF32. One epoch per run. Seed: 42.

Grid design. Batch sizes: $B \in \{8, 16, 32, 64, 128, 256\}$ (6 values, $2\times$ spacing). Learning rates: 13 values per configuration on a $\log_2$ grid with $2\times$ spacing. LoRA LR range: $\log_2 \eta \in [-12, 0]$ for EXP-01, $[-17, -5]$ for EXP-02, $[-18, -6]$ for EXP-03. FullFT LR range: $\log_2 \eta \in [-15, -3]$ for EXP-01, $[-19, -7]$ for EXP-02, $[-20, -8]$ for EXP-03. Total: 6 batch sizes $\times$ 5 ranks $\times$ 13 LRs $= 390$ runs per model, 1,170 runs total, 14 divergences (all in EXP-03).

Optimal LR identification. For each (batch size, rank) configuration, the U-curve of final exponential moving average (EMA) loss vs. $\log_2 \eta$ is scanned for its minimum $\hat{L}$ at $\hat{\eta}$. The optimal learning rate η^* is taken as $\hat{\eta}$, the grid point achieving the lowest EMA loss. All η^* values reported below are $\log_2 \eta^*$ on this integer grid.

4.2 Fitting the Scaling Law

4.2.1 Measured optima and fitting results

Figure 2 presents the measured $\log_2 \eta^*(B, r)$ for all three experiments. Clear patterns emerge: (1) η^* increases with B for all ranks, (2) η^* decreases with r at fixed B.

Figure 3 visualizes the full learning-rate U-curves for EXP-02 (representative; the other experiments show similar patterns). Each (B, r) configuration exhibits a clear U-shape with a well-defined minimum, confirming that η^* is reliably identified from the grid. The rightward shift of LoRA optima with increasing B, visually confirms the batch-size dependence predicted by Eq. 10. At the optimal learning rate, LoRA achieves final loss comparable to FullFT across all batch sizes, supporting the claim that the large-batch gap is an optimization issue rather than a capacity limit.

We fit Eq. 10 to the 30 data points (6 batch sizes $\times$ 5 ranks, treating FullFT as $r = d$) per experiment by minimizing squared error in $\log_2 \eta^*$ via Nelder–Mead optimization. Table 1 reports the fitted parameters and goodness-of-fit, and Figure 4 visualizes the fit. The optimizer exponent s is 0.52, 0.53, and 0.52 for the three models, all very close to the Adam limit of $\frac{1}{2}$.

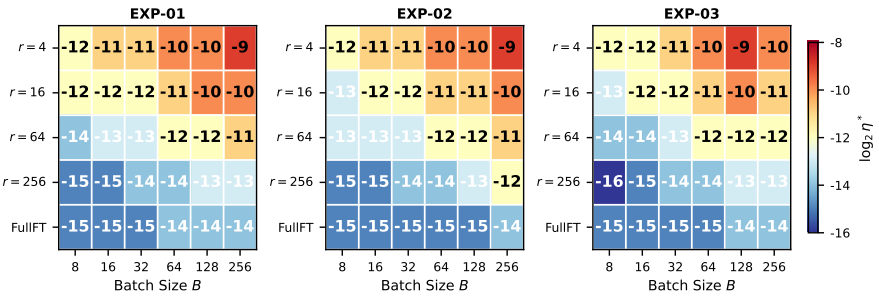


Figure 2: Optimal learning rates $\log_2 \eta^*(B, r)$ for all three experiments.

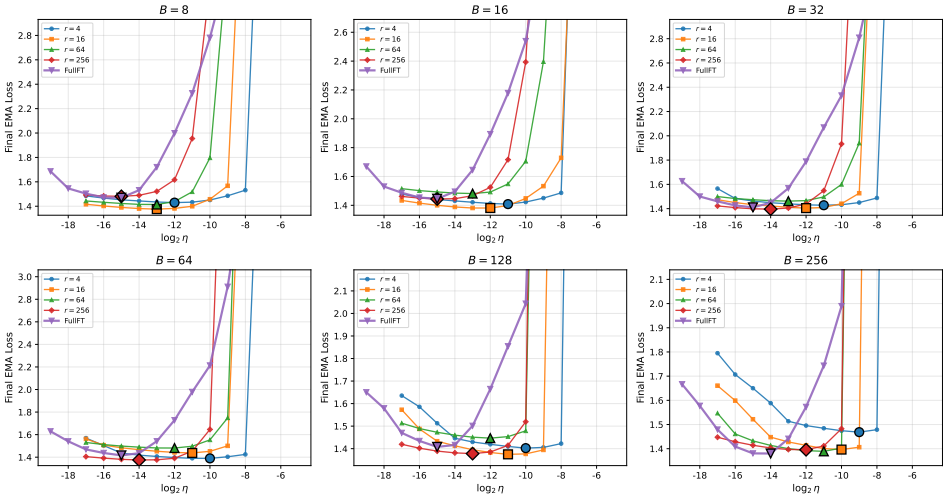


Figure 3: Learning-rate U-curves for Llama-3.2-1B. Each panel shows final EMA loss vs. $\log_2 \eta$ for one batch size, with lines for LoRA ranks ($r = 4, 16, 64, 256$) and FullFT (thick). Markers indicate optimal η^*.

Table 1: Fitted parameters for Eq. 10. RMSE is in $\log_2 \eta^*$ units.

Experiment	η_0	s	B_∞	RMSE	R^2
EXP-01 (Qwen2.5-0.5B)	0.111	0.52	207	0.371	0.956
EXP-02 (Llama-3.2-1B)	0.266	0.53	461	0.430	0.942
EXP-03 (Qwen2.5-3B)	1.051	0.52	10345	0.563	0.900

4.2.2 Comparison with a power law

As a robustness check, we compare Eq. 10 against a pure power law $\eta^* = C \cdot B^a / r^b$, which has the same number of free parameters. As shown in Table 2, the rational form achieves lower root mean squared error (RMSE) and Akaike Information Criterion (AIC) for EXP-01 and EXP-02; for EXP-03 the two are essentially tied (differing by < 0.01 in RMSE), because the large fitted B_∞ pushes the saturation regime beyond the measured range, reducing the rational form to the power law. This confirms that the structural form $B_\infty d/r$ is not redundant: it identifies when the power law suffices and when it does not.

4.2.3 Cross-model parameter stability

The theory predicts that s remains constant (Section 3.4), while η_0 and B_∞ are free parameters. Examining the fitted parameters in Table 1, we find that the exponent s is remarkably stable across

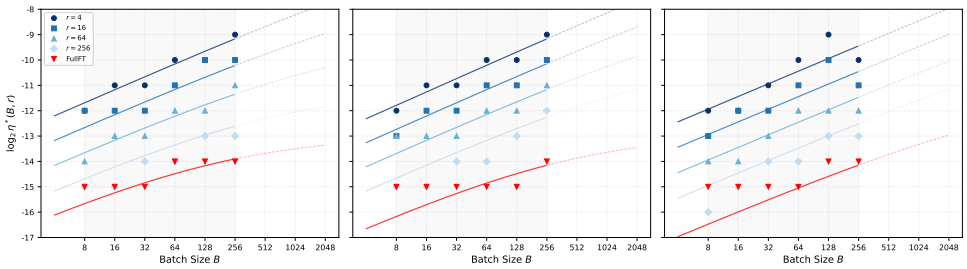


Figure 4: Measured $\log_2 \eta^*$ (markers) and fitted scaling law (lines) for all three experiments. Solid curves cover the data range ($B \in [8, 256]$); dashed curves extrapolate beyond data to show predicted saturation. LoRA ranks with small r remain in the linear regime throughout, while FullFT is already saturated.

Table 2: Comparison with power law across all three experiments. Both functional forms have three free parameters; lower RMSE and AIC are better.

	EXP-01			EXP-02			EXP-03		
Model	RMSE	R^2	AIC	RMSE	R^2	AIC	RMSE	R^2	AIC
Proposed	0.371	0.956	-53.5	0.430	0.942	-44.6	0.563	0.900	-28.5
Power law	0.389	0.951	-50.7	0.433	0.941	-44.3	0.562	0.901	-28.6

all three experiments (0.52, 0.53, 0.52), consistently near the Adam sign-gradient limit of $\frac{1}{2}$. In contrast, η_0 and B_∞ vary substantially across models, indicating that these parameters depend on model-specific factors. The stability of s is practically significant: it reduces the scaling law to a two-parameter family, enabling the calibration scheme of Section 4.3.

4.3 Two-Point Learning-Rate Prediction

4.3.1 The calibration scheme

The learning rate is the hyperparameter that most affects whether fine-tuning succeeds, yet its optimum is not a single number: it shifts with batch size and, for LoRA, with rank through the ratio r/d, so every change of configuration forces the practitioner to tune it again. The full sweep needed to locate each optimum is expensive, and for LoRA it is the dominant cost of an otherwise cheap adaptation. The scaling law of Eq. 10 offers a way out. Because its exponent s is stable across models and can be frozen at $\frac{1}{2}$ (Section 3.4), the law collapses to a two-parameter family that two measurements determine; once fixed, it predicts the optimal rate for *all* batch sizes and ranks at once. This section turns that observation into a concrete tuning scheme.

The scheme has three steps. (1) *Calibrate*: run two U-curves and solve Eq. 10 for the pair. The first is a FullFT curve at $B = 32$, which constrains B_∞; the second is a LoRA curve at $r = 64$, $B = 128$, which constrains η_0. (2) *Predict*: evaluate $\eta^*(B, r)$ from the fitted law with $s = \frac{1}{2}$ at every (batch size, rank) of interest; for $r/d \gtrsim 0.1$ the prediction is directly usable. (3) *Correct*: at $r \ll d$ the prediction over-shoots; when a run diverges, its loss flattens at the random baseline instead of falling, which means the rate is too high, so one searches *downward* by one to two grid steps. The two curves cost about 12 runs, against 390 for an exhaustive grid, more than a $30\times$ reduction, and the correction step reaches the $r = 16$ optimum in three to four runs rather than the eight of a full U-curve. Table 3 reports the accuracy of the prediction step on the models used for fitting.

4.3.2 Validation on unseen models

With no refitting, we apply the scheme to three models never used for fitting: GPT-2-large ($d = 1280$, decoder, Alpaca SFT), RoBERTa-large ($d = 1024$, encoder with a randomly initialized three-way head, MNLI), and Gemma-2-2B ($d = 2304$, decoder, Alpaca SFT); Table 4 summarizes the outcome. The calibration anchors pin both parameters whenever the fit lands at an interior B_∞. This

Table 3: Two-point calibration vs. full grid search. "Calibration" uses 12 runs (2 configurations $\times$ 6 LRs); "Full fit" uses 390 runs (30 configurations $\times$ 13 LRs). All errors in $\log_2$ units. The last column reports the percentage of predictions with error $\leq 1.0 \log_2$ (one grid step, i.e., within $2\times$ of the true optimum). EXP-03 full-data fit RMSE is 0.563 (Table 1).

Experiment	Method	Runs	RMSE	Max err	% err ≤ 1
EXP-01	2-point calibration	12	0.460	0.98	100%
EXP-01	Full-data fit	390	0.371	0.78	100%
EXP-02	2-point calibration	12	0.442	1.25	97%
EXP-02	Full-data fit	390	0.430	1.19	97%
EXP-03	2-point calibration	12	0.607	1.32	93%
EXP-03	Full-data fit	390	0.563	1.37	93%

Table 4: Two-point calibration transfer on three unseen models. Anchors are the measured optima ($\log_2 \eta^*$) of a FullFT curve at $B = 32$ and a LoRA $r = 64$ curve at $B = 128$; all other configurations are predicted. For $r/d \gtrsim 0.1$ the LoRA prediction is directly usable; at $r \ll d$ it over-shoots, which a downward search corrects.

Model	Arch. / Task	d	Anchors $\log_2 \eta^*$	Unseen result
GPT-2-large	dec. / Alpaca	1280	$-11.0, -8.0$	RMSE 0.33, max err 0.58; all within one step
RoBERTa-large	enc. / MNLI	1024	$-14.0, -10.5$	FullFT accurate; LoRA accurate for $r/d \gtrsim 0.1$; over-shoots at $r \ll d$ (search 1–2 steps down)
Gemma-2-2B	dec. / Alpaca	2304	$-14.5, -9.5$	FullFT accurate; LoRA accurate for $r/d \gtrsim 0.1$; over-shoots at $r \ll d$ (search 1–2 steps down)

holds for RoBERTa-large and Gemma-2-2B, but fails for GPT-2-large, whose solved B_∞ saturates the upper bound imposed during fitting and reduces the law to a power law in B; there the same two anchors reproduce the $r = 64$ batch sweep to within one grid step (RMSE 0.33 $\log_2$, max error 0.58), matching the in-sample full-data fit (RMSE 0.32). On all three models the predicted *FullFT* rate converges (EMA 0.34–0.40 on RoBERTa-large, 0.98–1.15 on Gemma-2-2B, 1.34–1.39 on GPT-2-large). On the two non-degenerate models the predicted *LoRA* rate is directly usable down to $r/d \approx 0.1$, where the loss at the predicted rate matches FullFT; only at smaller rank does the prediction over-shoot, and there the downward correction of Step (3) recovers the FullFT level. Table 8 scores every configuration of all three models, with the per-configuration record and divergence threshold in Appendix B.4.

This over-shoot is systematic and one-sided: the residual always lies *below* the prediction, by 0.95 and 2.33 steps at $r = 16$ on RoBERTa-large ($B = 64$ and 128), 1.62 and 1.78 on Gemma-2-2B, and 1.00 on GPT-2-large at both batch sizes (Table 8). Since a single step separates convergence from the divergence cliff, Step (3) corrects only downward, on the safe side of the optimum, where one to two steps recover it in three to four runs; the anomaly is thus priced into the scheme rather than left as an open defect.

5 Conclusion

We show that the large-batch gap between LoRA and FullFT is primarily a learning-rate scaling mismatch, not a structural defect. The underlying reason is that LoRA's Sylvester operator $\mathcal{P}_t$ induces a different optimization geometry, reshaping the effective Hessian and changing the optimal scaling law. We derive a closed-form scaling law $\eta^*(B, r) = \frac{\eta_0}{r} \cdot \frac{B^s}{B^s + (B_\infty d/r)^s}$ with three free parameters, validate it on three models across two architectures with $R^2 \geq 0.90$, and show that the structural form, with $s \approx 0.5$, transfers across model scales. A two-point calibration scheme reduces LR tuning cost by $30\times$.

AI USE STATEMENT

In this work, we used generative AI tools for writing code. We have not used generative AI tools for processing experimental data, and no other task with required disclosure is applicable to this work. Additionally, we used generative AI tools for language polishing and code debugging; all of these uses were through the code assistant inside the Trae IDE (v3.3). We have reviewed all AI-assisted work, and AI-assisted code was inspected line by line by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the material needed to reproduce all results. The assumptions and complete proofs behind the theoretical claims of Section 3 are given in Appendices A.1, A.2, and A.3, the last of which also derives the width–rank scaling law and the three regimes of Section 3.4. All models and datasets used are public, and the model configurations, optimizers, and rank and batch-size grids are described in Section 4.1. The fitted parameters of the law are reported in Table 1, and the per-configuration optima underlying every figure and table of the main text are listed in Appendix B (Tables 5–7). The two-point calibration scheme, including the divergence threshold that the downward correction relies on, is specified in Section 4.3, with the per-configuration record of the transfer experiment on unseen models given in Appendix B.4 (Table 8). The code used for training and for fitting the law will be released as anonymous supplementary material.

REFERENCES

Nan Chen, Soledad Villar, and Soufiane Hayou. Learning rate scaling across LoRA ranks and transfer to full finetuning. *arXiv preprint arXiv:2602.06204*, 2026.

Enea Monzio Compagnoni, Tianlin Liu, Rustem Islamov, Frank Norbert Proske, Antonio Orvieto, and Aurelien Lucchi. Adaptive methods through the lens of SDEs: Theoretical insights on the role of noise. In *ICLR*, 2025.

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.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *ICML*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023.

Yu-Ang Lee, Ching-Yun Ko, Pin-Yu Chen, and Mi-Yen Yeh. Learning rate matters: Vanilla LoRA may suffice for LLM fine-tuning. In *ICML 2026 Workshop on CoLoRAI*, 2026.

Shuaipeng Li, Penghao Zhao, Hailin Zhang, Xingwu Sun, Hao Wu, Dian Jiao, Weiyan Wang, Chengjun Liu, Zheng Fang, Jinbao Xue, Yangyu Tao, Bin Cui, and Di Wang. Surge phenomenon in optimal learning rate and batch size scaling. In *NeurIPS*, 2024.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the SDEs and scaling rules for adaptive gradient algorithms. In *NeurIPS*, 2022.

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

Meta AI. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

John Schulman et al. Lora without regret. *Thinking Machines Blog*, 2025. URL https://thinkingmachines.ai/blog/lora/.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Yifan Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following LLaMA model, 2023. URL https://github.com/tatsu-lab/stanford_alpaca.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters properly optimized? In *ICLR*, 2025.

Jui-Nan Yen, Si Si, Zhao Meng, Felix Yu, Sai Surya Duvvuri, Inderjit S. Dhillon, Cho-Jui Hsieh, and Sanjiv Kumar. Lora done rite: Robust invariant transformation equilibration for lora optimization. In *ICLR*, 2025.

Zicheng Zhang, Haoran Li, Yifeng Zhang, Guoqiang Gong, Jiaxing Wang, Junxing Hu, Pengzhang Liu, and Qixia Jiang. The primacy of magnitude in low-rank adaptation. In *NeurIPS*, 2025.

Zicheng Zhang, Haoran Li, Jiaxing Wang, Guoqiang Gong, Anqi Li, Yudong Hu, Ting Xiong, Yurong Gao, Junxing Hu, Zhida Jiang, Yifeng Zhang, Pengzhang Liu, and Qixia Jiang. The hidden power of scaling factor in lora optimization. *arXiv preprint arXiv:2606.12883*, 2026.

APPENDIX

A THEORETICAL ANALYSIS

A.1 PROOF OF LEMMA 1

$\mathcal{P}_t$ is self-adjoint under the Frobenius inner product $\langle X, Y\rangle_F = \mathrm{tr}(X^\top Y)$ and positive semidefinite. For any $X \in \mathbb{R}^{m\times n}$:

$$\langle X, \mathcal{P}_t[X]\rangle_F = \mathrm{tr}(X^\top BB^\top X) + \mathrm{tr}(X^\top XA^\top A) = \|B^\top X\|_F^2 + \|XA^\top\|_F^2 \geq 0.$$

Self-adjointness: $\langle X, \mathcal{P}_t[Y]\rangle_F = \mathrm{tr}(X^\top BB^\top Y) + \mathrm{tr}(X^\top YA^\top A) = \langle \mathcal{P}_t[X], Y\rangle_F$. □

Since $\mathcal{P}_t$ is self-adjoint and PSD, it admits a unique PSD square root $\mathcal{P}_t^{1/2}$.

For the noise-scale derivation, we additionally require the gradient covariance $\Sigma = \mathbb{E}[(g-G)(g-G)^\top]$ to be locally stationary, i.e., constant within a neighborhood of the current parameters. This is needed only for the noise-scale result, not for the operator analysis itself.

A.2 PROOF OF THEOREM 1

Treating $\mathcal{P}_t$ as constant within a single step, write the LoRA update as $\Delta W = -\eta\,\mathcal{C}[G]$ where $\mathcal{C} = \alpha^2\mathcal{P}_t$ is the effective preconditioner. Since $\mathcal{P}_t$ is self-adjoint and PSD (Lemma 1) and $\alpha > 0$, $\mathcal{C}$ is also self-adjoint and PSD, admitting a unique PSD square root $\mathcal{C}^{1/2} = \alpha\mathcal{P}_t^{1/2}$.

Substituting into the local quadratic model:

$$\Delta L = \langle G, \Delta W\rangle_F + \frac{1}{2}\langle \Delta W, H[\Delta W]\rangle_F = -\eta\,\langle G, \mathcal{C}[G]\rangle_F + \frac{\eta^2}{2}\langle \mathcal{C}[G], H[\mathcal{C}[G]]\rangle_F.$$

Define the effective gradient $\tilde{G} = \mathcal{C}^{1/2}[G]$. By self-adjointness of $\mathcal{C}^{1/2}$:

$$\langle G, \mathcal{C}[G]\rangle_F = \langle \mathcal{C}^{1/2}[G], \mathcal{C}^{1/2}[G]\rangle_F = \|\tilde{G}\|_F^2,$$

$$\langle \mathcal{C}[G], H[\mathcal{C}[G]]\rangle_F = \langle \mathcal{C}^{1/2}[G],\, \mathcal{C}^{1/2}H\mathcal{C}^{1/2}[\mathcal{C}^{1/2}[G]]\rangle_F = \langle \tilde{G}, H_{\text{eff}}[\tilde{G}]\rangle_F,$$

where $H_{\text{eff}} = \mathcal{C}^{1/2} H \mathcal{C}^{1/2} = \alpha^2 \mathcal{P}_t^{1/2} H \mathcal{P}_t^{1/2}$. Substituting back:

$$\Delta L = -\eta \, \|\tilde{G}\|_F^2 + \frac{\eta^2}{2} \langle \tilde{G}, H_{\text{eff}}[\tilde{G}] \rangle_F .$$

This is the standard gradient-descent loss decrement with step size η, effective gradient $\tilde{G}$, and effective Hessian H_{eff}. Under the stationary gradient condition, the transformed noise covariance is $\tilde{\Sigma} = \mathcal{C}^{1/2} \Sigma \mathcal{C}^{1/2}$, and the generalized noise scale follows from the standard derivation (McCandlish et al., 2018) applied to $(\tilde{G}, H_{\text{eff}}, \tilde{\Sigma})$. □

A.3 Detailed Derivations of Width–Rank Scaling

This appendix provides the step-by-step derivations for the two effects summarized in Section 3.4.

Effect 1: Rank scaling derivation. The Sylvester operator $\mathcal{P}_t$ (Definition 1) governs the effective update magnitude. At initialization ($B = 0$, A drawn from Kaiming initialization with $\text{Var}(A_{ij}) = 1/d$), the factor $A^\top A$ concentrates around its expectation:

$$\mathbb{E}[A^\top A] = \frac{r}{d} I_n, \tag{11}$$

a standard result for $r \times n$ matrices with i.i.d. $\mathcal{N}(0, 1/d)$ entries (Marchenko–Pastur law / Wishart mean). Substituting into the weight-space update (Eq. 3), and noting that $B = 0$ at initialization simplifies $\mathcal{P}_t[G] = GA^\top A \approx (r/d)\, G$:

$$\Delta W = -\eta \alpha^2 \mathcal{P}_t[G] \approx -\eta \alpha^2 \frac{r}{d} G. \tag{12}$$

The effective learning rate experienced by the weight matrix is $\eta_{\text{eff}} = \eta \, \alpha^2 r/d$. For the effective dynamics to match FullFT (where $\eta_{\text{eff}} = \eta$), the nominal learning rate must compensate for the r/d attenuation. Since d and α are fixed for a given model, this yields $\eta^*_{\text{LoRA}} \propto 1/r$. This is a purely geometric effect: $A^\top A \approx (r/d)\, I$ means the same nominal η produces updates that grow with r, requiring $\eta \propto 1/r$ to compensate. The constant η_0 in the final formula (Eq. 10) absorbs α^2 and model-dependent factors.

Effect 2: Noise inflation derivation. The noise scale $B_{\text{noise}} = \text{tr}(\Sigma H)/(G^\top H G)$ depends on how gradient noise Σ and curvature H interact. Under the approximation that mini-batch gradient noise is approximately isotropic ($\Sigma \approx \sigma^2 I$, justified by the central limit theorem applied to mini-batch sampling), the noise scale simplifies to:

$$B_{\text{noise}} \approx \frac{\sigma^2 \, \text{tr}(H)}{G^\top H \, G}. \tag{13}$$

For FullFT, H is the full d-dimensional Hessian. For LoRA rank-r, the effective quantities are $H_{\text{eff}} = \alpha^2 \mathcal{P}_t^{1/2} H \mathcal{P}_t^{1/2}$, $\tilde{\Sigma} = \alpha^2 \mathcal{P}_t^{1/2} \Sigma \mathcal{P}_t^{1/2}$, and $\tilde{G} = \alpha \mathcal{P}_t^{1/2} G$, giving $B_{\text{noise}}^{\text{LoRA}} = \text{tr}(\tilde{\Sigma} \, H_{\text{eff}})/(\tilde{G}^\top H_{\text{eff}} \, \tilde{G})$.

Under the Marchenko–Pastur approximation $A^\top A \approx (r/d)\, I$, the projection $\mathcal{P}_t$ acts as a uniform scaling by r/d. Both the noise numerator and signal denominator then scale by the same overall factor $\alpha^4 (r/d)^2$:

$$\text{tr}(\tilde{\Sigma} \, H_{\text{eff}}) \approx \alpha^4 (r/d)^2 \, \text{tr}(\Sigma \, H), \tag{14}$$

$$\tilde{G}^\top H_{\text{eff}} \, \tilde{G} \approx \alpha^4 (r/d)^2 \, G^\top H \, G, \tag{15}$$

so uniform scaling alone would leave the ratio unchanged. The d/r inflation arises from the heavy-tailed Hessian spectrum.

Neural-network Hessians are heavy-tailed: a few eigenvalues dominate $\text{tr}(H)$, while the bulk is small. The gradient signal concentrates in these dominant directions, so $G^\top H G$ is disproportionately determined by whether the projection captures a dominant eigenvector. For a random rank-r

subspace, this occurs with probability $\sim r/d$. When the dominant direction is missed (probability $1 - r/d$), the signal drops to the bulk scale, yielding an additional attenuation factor of r/d in expectation:

$$\tilde{G}^\top H_{\text{eff}}\, \tilde{G} \approx \alpha^4 (r/d)^3\, G^\top H\, G. \tag{16}$$

The noise numerator, being a trace (sum over all directions), averages uniformly and retains the $(r/d)^2$ scaling of Eq. 14. The ratio is therefore:

$$\frac{B_{\text{noise}}^{\text{LoRA}}}{B_{\text{noise}}^{\text{FullFT}}} = \frac{(r/d)^2}{(r/d)^3} = \frac{d}{r}. \tag{17}$$

This is consistent with the qualitative prediction of Section 3.2. Note that α cancels in the ratio because it appears symmetrically in $\tilde{\Sigma}$, H_{eff}, and $\tilde{G}$; B_{noise} is a geometric property independent of the scaling factor α.

Asymptotic regime analysis. The complete scaling law (Eq. 10) predicts two asymptotic regimes for the r-dependence of $\log_2 \eta^*$:

- **Linear regime** ($B \ll B_{\text{noise}}(r)$): the rational factor $\approx (B/B_{\text{noise}})^s$, so $\eta^* \approx \eta_0 B^s r^{s-1}/(B_\infty d)^s$, and the slope of $\log_2 \eta^*$ vs. $\log_2 r$ is $s - 1 \approx -0.50$ for $s \approx 0.5$.
- **Saturation regime** ($B \gg B_{\text{noise}}(r)$): the rational factor ≈ 1, so $\eta^* \approx \eta_0/r$, and the slope is -1.

The transition between these regimes occurs at $B = B_{\text{noise}}(r) = B_\infty\, d/r$, which decreases with r. For small r (large B_{noise}), all experimental batch sizes lie in the linear regime and the slope is $\approx s - 1$. For large r (small B_{noise}), some batch sizes enter the saturation regime and the slope steepens toward -1.

B Full Experimental Results

This appendix documents every individual run for full reproducibility. Tables 5, 6, and 7 give the full learning-rate U-curves for EXP-01–03 (final EMA loss at every tested rate for every (B, r) configuration, optimal η^* bolded), the data on which the law was fitted. Section B.4 validates the two-point scheme end-to-end on the unseen models of Section 4.3, scoring every configuration in Table 8.

B.1 EXP-01: Qwen2.5-0.5B, $d = 896$

Table 5: EXP-01 (Qwen2.5-0.5B, $d = 896$): Final EMA loss at each learning rate for all batch sizes. Bold marks the optimal η^* (lowest EMA loss).

	$B = 8$					$B = 16$				
$\log_2 \eta$	$r = 4$	$r = 16$	$r = 64$	$r = 256$	FT	$r = 4$	$r = 16$	$r = 64$	$r = 256$	FT
	$B = 8$					$B = 16$				
-7.0	4.4282	4.3376	3.9019	4.0536	—	4.5920	4.3111	4.2355	4.1800	—
-8.0	1.4371	2.7496	4.5176	3.9647	—	1.4183	4.6666	3.8645	4.0448	—
-9.0	1.4052	1.4907	2.8553	3.7666	—	1.3909	1.4289	2.0794	3.5777	—
-10.0	1.3812	1.4124	1.7043	2.5779	—	1.3735	1.3666	1.5197	2.0766	—
-11.0	1.3678	1.3741	1.4673	1.5992	1.9776	**1.3638**	1.3441	1.3980	1.4985	1.8568
-12.0	**1.3636**	1.3592	1.4065	1.3744	1.7139	1.3655	**1.3378**	1.3627	1.3638	1.6313
-13.0	1.3663	**1.3592**	1.3868	1.2978	1.5240	1.3714	1.3412	**1.3542**	1.3142	1.4636
-14.0	1.3737	1.3656	**1.3819**	1.2703	1.4111	1.3780	1.3481	1.3558	1.3029	1.3750
-15.0	1.3819	1.3756	1.3858	**1.2650**	**1.3811**	1.3881	1.3567	1.3612	**1.3027**	**1.3599**
-16.0	1.3932	1.3852	1.3943	1.2684	1.3950	1.4016	1.3659	1.3706	1.3090	1.3777
-17.0	1.4082	1.3970	1.4047	1.2772	1.4196	1.4196	1.3784	1.3810	1.3189	1.4036
-18.0	1.4233	1.4113	1.4168	1.2868	1.4515	1.4412	1.3927	1.3923	1.3300	1.4357
-19.0	—	—	—	—	1.5031	—	—	—	—	1.4794
-20.0	—	—	—	—	1.5940	—	—	—	—	1.5615

continued from previous page

$\log_2 \eta$	$r=4$	$r=16$	$r=64$	$r=256$	FT	$r=4$	$r=16$	$r=64$	$r=256$	FT
	$B=32$					$B=64$				
-7.0	4.4566	4.4661	4.7726	5.3929	—	4.6714	4.9992	4.7377	5.2306	—
-8.0	1.4622	1.5152	4.4102	4.3078	—	1.4149	1.4600	4.4703	4.7415	—
-9.0	1.4401	1.4242	1.7284	3.9464	—	1.3858	1.3883	1.5634	4.2602	—
-10.0	1.4246	1.3790	1.4606	1.6816	—	**1.3789**	1.3556	1.3976	1.5342	—
-11.0	**1.4235**	1.3606	1.3823	1.3984	1.7712	1.3800	**1.3483**	1.3574	1.3739	1.7162
-12.0	1.4268	**1.3599**	1.3638	1.3199	1.5695	1.3868	1.3528	**1.3510**	1.3277	1.5330
-13.0	1.4328	1.3668	**1.3629**	1.2949	1.4214	1.3954	1.3619	1.3532	1.3112	1.4053
-14.0	1.4419	1.3759	1.3661	**1.2886**	1.3474	1.4083	1.3715	1.3597	**1.3074**	**1.3552**
-15.0	1.4543	1.3853	1.3745	1.2919	**1.3415**	1.4253	1.3827	1.3686	1.3131	1.3620
-16.0	1.4714	1.3970	1.3834	1.3011	1.3618	1.4448	1.3970	1.3787	1.3231	1.3835
-17.0	1.4920	1.4120	1.3942	1.3117	1.3881	1.4720	1.4144	1.3909	1.3339	1.4078
-18.0	1.5201	1.4294	1.4061	1.3229	1.4156	1.5205	1.4392	1.4051	1.3442	1.4351
-19.0	—	—	—	—	1.4548	—	—	—	—	1.4717
-20.0	—	—	—	—	1.5318	—	—	—	—	1.5635
	$B=128$					$B=256$				
-7.0	1.3830	6.6861	5.8862	5.4645	—	1.4522	7.1330	7.1993	6.1783	—
-8.0	1.3553	1.4128	4.9422	5.6600	—	1.3806	1.4351	6.2724	6.5962	—
-9.0	1.3468	1.3399	1.4587	4.8539	—	**1.3734**	1.3626	1.4315	6.9777	—
-10.0	**1.3459**	**1.3306**	1.3908	1.4848	—	1.3745	**1.3554**	1.3721	1.4558	—
-11.0	1.3527	1.3328	1.3751	1.3915	1.6115	1.3838	1.3593	**1.3649**	1.3578	1.5342
-12.0	1.3620	1.3398	**1.3740**	1.3596	1.4682	1.3965	1.3675	1.3687	1.3346	1.4073
-13.0	1.3733	1.3487	1.3787	**1.3493**	1.3700	1.4131	1.3789	1.3779	**1.3319**	1.3287
-14.0	1.3890	1.3603	1.3874	1.3496	**1.3396**	1.4352	1.3943	1.3873	1.3375	**1.3090**
-15.0	1.4102	1.3741	1.3973	1.3580	1.3487	1.4642	1.4124	1.3979	1.3460	1.3235
-16.0	1.4391	1.3905	1.4087	1.3670	1.3700	1.5335	1.4359	1.4100	1.3556	1.3484
-17.0	1.4972	1.4139	1.4213	1.3771	1.3943	1.6618	1.4727	1.4243	1.3672	1.3738
-18.0	1.6277	1.4494	1.4366	1.3879	1.4215	1.7797	1.5595	1.4450	1.3812	1.4088
-19.0	—	—	—	—	1.4712	—	—	—	—	1.5088
-20.0	—	—	—	—	1.6492	—	—	—	—	1.7580

B.2 EXP-02: LLAMA-3.2-1B, $d = 2048$

Table 6: EXP-02 (Llama-3.2-1B, $d = 2048$): Final EMA loss at each learning rate for all batch sizes. Bold marks the optimal η^* (lowest EMA loss).

$\log_2 \eta$	$B=8$ $r=4$	$r=16$	$r=64$	$r=256$	FT	$B=16$ $r=4$	$r=16$	$r=64$	$r=256$	FT
	$B=8$					$B=16$				
-5.0	5.0842	4.5405	4.3168	4.8238	—	5.2542	5.5869	5.1190	5.8454	—
-6.0	4.9785	4.4142	4.4663	5.3550	—	5.0039	5.1918	5.5515	5.4464	—
-7.0	4.9844	4.3709	4.2560	4.7389	3.6329	5.0058	4.6108	4.8353	4.6176	3.5745
-8.0	1.5316	4.7422	4.5431	4.4654	3.4440	1.4862	1.7293	4.3871	4.5913	3.5297
-9.0	1.4856	1.5675	3.4608	4.5251	3.3361	1.4501	1.5329	2.3961	4.4271	3.2695
-10.0	1.4509	1.4574	1.7965	3.2196	2.7810	1.4226	1.4477	1.7053	2.3933	2.5389
-11.0	1.4334	1.3997	1.5182	1.9549	2.3260	**1.4084**	1.3984	1.5483	1.7166	2.1787
-12.0	**1.4296**	1.3821	1.4380	1.6171	1.9988	1.4130	1.3813	1.4919	1.5247	1.8938
-13.0	1.4347	**1.3763**	**1.4141**	1.5215	1.7208	1.4212	**1.3813**	**1.4815**	1.4667	1.6453
-14.0	1.4433	1.3808	1.4153	1.4895	1.5316	1.4306	1.3886	1.4847	1.4460	1.4951
-15.0	1.4525	1.3903	1.4227	**1.4811**	**1.4697**	1.4408	1.4001	1.4928	**1.4438**	**1.4457**
-16.0	1.4668	1.4021	1.4323	1.4836	1.4719	1.4559	1.4153	1.5019	1.4501	1.4538
-17.0	1.4848	1.4162	1.4431	1.4926	1.5035	1.4778	1.4340	1.5154	1.4617	1.4871
-18.0	—	—	—	—	1.5460	—	—	—	—	1.5303
-19.0	—	—	—	—	1.6860	—	—	—	—	1.6695
	$B=32$					$B=64$				
-5.0	5.4435	6.1264	5.7708	5.4625	—	6.4401	6.4042	5.8142	5.5500	—

continued from previous page

$\log_2 \eta$	$r=4$	$r=16$	$r=64$	$r=256$	FT	$r=4$	$r=16$	$r=64$	$r=256$	FT
-6.0	5.5924	5.5438	5.7032	5.7385	—	5.8759	6.2146	6.0118	6.7645	—
-7.0	5.1959	5.6197	5.2491	4.8670	3.6202	5.2762	6.3218	6.2824	5.9242	3.7831
-8.0	1.4880	4.8201	5.0397	4.8970	3.4740	1.4247	5.2620	6.2912	5.9405	3.9284
-9.0	1.4502	1.5270	1.9395	4.8185	2.8086	1.4037	1.5005	1.7485	5.9112	2.9110
-10.0	1.4321	1.4406	1.5998	1.9329	2.3319	**1.3894**	1.4521	1.5533	1.6461	2.2131
-11.0	**1.4280**	1.4096	1.4965	1.5479	2.0687	1.3900	**1.4375**	1.4971	1.4534	1.9787
-12.0	1.4314	**1.4044**	1.4663	1.4383	1.7882	1.3968	1.4423	**1.4812**	1.3907	1.7282
-13.0	1.4396	1.4117	**1.4619**	1.4054	1.5680	1.4065	1.4526	1.4817	1.3768	1.5418
-14.0	1.4488	1.4217	1.4655	**1.3958**	1.4483	1.4182	1.4665	1.4890	**1.3753**	1.4352
-15.0	1.4629	1.4325	1.4735	1.3987	**1.4106**	1.4364	1.4835	1.4986	1.3811	**1.4159**
-16.0	1.4840	1.4499	1.4834	1.4086	1.4266	1.5032	1.5066	1.5119	1.3916	1.4374
-17.0	1.5652	1.4737	1.4993	1.4226	1.4599	1.5693	1.5622	1.5304	1.4050	1.4689
-18.0	—	—	—	—	1.4999	—	—	—	—	1.5408
-19.0	—	—	—	—	1.6269	—	—	—	—	1.6292
	$B=128$					$B=256$				
-5.0	6.5244	6.6941	6.8460	6.5136	—	7.2922	6.9354	7.2922	7.3884	—
-6.0	6.6646	6.4566	7.0544	6.2964	—	7.3148	6.9622	7.3771	6.9623	—
-7.0	6.6727	6.6812	6.1307	6.4849	3.9395	6.8166	7.3552	7.4347	7.1717	4.7068
-8.0	1.4225	5.7085	6.1074	6.6975	4.3708	1.4787	6.9646	5.9721	7.1496	5.0937
-9.0	1.4056	1.3945	5.4060	6.6891	4.1267	**1.4685**	1.4063	6.2159	6.7192	4.8800
-10.0	**1.4024**	1.3768	1.4790	1.5195	2.0435	1.4744	**1.3964**	1.4026	1.4826	1.9895
-11.0	1.4099	**1.3748**	1.4535	1.4146	1.8543	1.4845	1.4005	**1.3891**	1.4114	1.7434
-12.0	1.4193	1.3830	**1.4467**	1.3852	1.6647	1.4956	1.4139	1.3921	**1.3957**	1.5728
-13.0	1.4297	1.3953	1.4510	**1.3780**	1.5014	1.5135	1.4279	1.4003	1.3968	1.4412
-14.0	1.4463	1.4128	1.4600	1.3814	1.4165	1.5883	1.4478	1.4136	1.4031	**1.3801**
-15.0	1.5127	1.4331	1.4727	1.3893	**1.4079**	1.6501	1.5218	1.4324	1.4142	1.3807
-16.0	1.5857	1.4875	1.4893	1.4021	1.4342	1.7067	1.5988	1.4611	1.4288	1.4089
-17.0	1.6354	1.5727	1.5131	1.4194	1.4701	1.7949	1.6606	1.5468	1.4475	1.4792
-18.0	—	—	—	—	1.5792	—	—	—	—	1.5774
-19.0	—	—	—	—	1.6503	—	—	—	—	1.6665

B.3 EXP-03: QWEN2.5-3B, $d = 2048$

Table 7: EXP-03 (Qwen2.5-3B, $d = 2048$): Final EMA loss at each learning rate for all batch sizes. Bold marks the optimal η^* (lowest EMA loss). *div.* indicates training divergence; — indicates configurations not evaluated.

$\log_2 \eta$	$r=4$	$r=16$	$r=64$	$r=256$	FT	$r=4$	$r=16$	$r=64$	$r=256$	FT
	$B=8$					$B=16$				
-6.0	*div.*	7.2293	*div.*	*div.*	—	6.4046	6.7787	*div.*	*div.*	—
-7.0	35.7999	*div.*	*div.*	*div.*	—	6.0077	6.3897	*div.*	*div.*	—
-8.0	5.4919	5.9056	6.1068	*div.*	3.6828	5.6360	6.1044	8.9633	6.7164	3.7945
-9.0	1.7133	5.6082	5.7923	7.7436	3.6558	1.6591	5.6420	7.2099	*div.*	3.6678
-10.0	1.6595	5.4065	5.3663	23.5442	3.7766	1.6215	1.6801	5.5246	5.7655	3.5261
-11.0	1.6294	1.6178	1.7119	5.4277	3.6995	1.6095	1.6221	1.6295	5.4865	2.1424
-12.0	**1.6257**	1.5918	1.5830	5.0408	1.9762	**1.6095**	**1.6130**	1.5800	5.0275	1.9083
-13.0	1.6286	**1.5887**	1.5627	1.6075	1.7612	1.6132	1.6154	1.5691	1.5843	1.7136
-14.0	1.6336	1.5932	**1.5593**	1.5842	1.6493	1.6199	1.6200	**1.5690**	1.5692	1.6242
-15.0	1.6407	1.5990	1.5623	1.5759	**1.6158**	1.6301	1.6274	1.5733	**1.5638**	**1.5978**
-16.0	1.6515	1.6049	1.5674	**1.5748**	1.6237	1.6448	1.6358	1.5781	1.5649	1.6100
-17.0	1.6669	1.6137	1.5732	1.5781	1.6446	1.6634	1.6481	1.5842	1.5691	1.6316
-18.0	1.6852	1.6257	1.5815	1.5838	1.6753	1.6838	1.6648	1.5944	1.5759	1.6627
-19.0	—	—	—	—	1.7220	—	—	—	—	1.7040
-20.0	—	—	—	—	1.9561	—	—	—	—	1.9579
	$B=32$					$B=64$				
-6.0	6.9136	7.0724	*div.*	7.1299	—	7.0735	6.6658	6.8185	7.0149	—

continued from previous page

$\log_2 \eta$	$r = 4$	$r = 16$	$r = 64$	$r = 256$	FT	$r = 4$	$r = 16$	$r = 64$	$r = 256$	FT
-7.0	6.1571	7.1430	6.5126	*div.*	—	6.0138	6.5491	7.1697	7.0521	—
-8.0	5.7867	5.8033	7.1684	6.7803	3.8304	5.3274	5.6277	7.1141	6.6572	4.2822
-9.0	1.6423	5.9429	9.9616	5.9244	3.6636	1.6070	1.6126	5.7673	6.2630	3.8013
-10.0	1.6145	1.5463	5.6934	5.9505	2.3722	**1.6000**	1.5532	5.9172	5.7569	2.2360
-11.0	**1.6133**	1.5254	1.5441	5.2293	2.0403	1.6049	**1.5434**	1.5574	5.3884	1.9819
-12.0	1.6190	**1.5217**	1.5179	1.6154	1.8044	1.6113	1.5452	**1.5448**	1.5576	1.7396
-13.0	1.6249	1.5258	**1.5134**	1.5899	1.6389	1.6196	1.5499	1.5454	1.5432	1.6043
-14.0	1.6333	1.5309	1.5157	**1.5834**	1.5742	1.6332	1.5571	1.5490	**1.5415**	1.5536
-15.0	1.6475	1.5387	1.5201	1.5837	**1.5601**	1.6506	1.5676	1.5545	1.5441	**1.5503**
-16.0	1.6653	1.5500	1.5259	1.5869	1.5746	1.6723	1.5821	1.5623	1.5496	1.5675
-17.0	1.6868	1.5654	1.5344	1.5934	1.5982	1.7044	1.6022	1.5738	1.5572	1.5900
-18.0	1.7166	1.5854	1.5466	1.6021	1.6272	1.7636	1.6276	1.5899	1.5673	1.6172
-19.0	—	—	—	—	1.6642	—	—	—	—	1.6624
-20.0	—	—	—	—	1.9641	—	—	—	—	2.0525
	$B = 128$					$B = 256$				
-6.0	7.0928	7.2112	6.9417	6.9752	—	7.2804	7.4309	7.6888	7.6631	—
-7.0	6.0359	6.6793	6.7426	6.8757	—	7.2299	7.3809	7.4588	7.3371	—
-8.0	6.8886	7.0884	7.2826	7.0557	4.2717	1.5825	5.7749	6.8448	7.0141	5.2516
-9.0	**1.5634**	1.6382	5.8979	6.4924	3.8163	1.5553	1.5804	7.8557	7.6704	4.1843
-10.0	1.5639	**1.5822**	1.5676	6.2858	2.0736	**1.5483**	1.5615	1.5916	7.4291	3.7230
-11.0	1.5695	1.5831	1.5311	1.5519	1.9103	1.5543	**1.5608**	1.5588	1.5293	1.7897
-12.0	1.5765	1.5891	**1.5281**	1.5159	1.7091	1.5636	1.5676	**1.5582**	1.4984	1.6381
-13.0	1.5876	1.5961	1.5302	**1.5086**	1.6007	1.5800	1.5772	1.5622	**1.4958**	1.5555
-14.0	1.6045	1.6064	1.5353	1.5105	**1.5557**	1.6040	1.5928	1.5688	1.4989	**1.5263**
-15.0	1.6284	1.6227	1.5434	1.5143	1.5617	1.6401	1.6169	1.5798	1.5032	1.5389
-16.0	1.6630	1.6453	1.5551	1.5210	1.5819	1.7186	1.6494	1.5973	1.5101	1.5611
-17.0	1.7300	1.6738	1.5715	1.5292	1.6070	2.0282	1.7032	1.6222	1.5215	1.5888
-18.0	2.0198	1.7197	1.5945	1.5427	1.6368	2.3791	1.9277	1.6549	1.5393	1.6387
-19.0	—	—	—	—	1.7894	—	—	—	—	2.0229
-20.0	—	—	—	—	2.3006	—	—	—	—	2.4858

B.4 TWO-POINT CALIBRATION TRANSFERS TO UNSEEN MODELS

Table 8 scores three models never used for fitting, end to end and refitting no constant; the protocol and the resulting fit are described in Section 4.3. The one quantity a reader needs in order to tell whether the predicted *LoRA* rate can be used directly or must be searched one to two grid steps *downward* is the divergence threshold of the model at hand: EMA ≥ 3 for the Alpaca SFT models, which have no random-loss baseline, and the $\ln 3$ random baseline for the three-way MNLI head.

Table 8: Calibration-transfer validation on the three unseen models. From the two anchors alone the law predicts the optimal rate $\log_2 \eta^*$ of every configuration; both the prediction and its FullFT companion are then trained. *EMA at pred.* and *EMA at opt.* are the losses at the predicted and at the measured-optimal rate, *below* the gap between the two in unit $\log_2$ steps (0: the prediction is kept), and *FullFT EMA* is the converged full fine-tuning loss at the same batch size. The prediction is directly usable at $r/d \gtrsim 0.1$ and over-shoots at $r \ll d$, where one step downward recovers the FullFT level on the two non-degenerate models; on GPT-2-large, whose anchors are degenerate, the same step leaves the $r = 16$ loss 0.21–0.30 above its FullFT companion.

Model	B	r	r/d	Pred. $\log_2 \eta^*$	EMA at pred.	below	EMA at opt.	FullFT EMA
RoBERTa-large	64	256	0.250	-12.23	0.3634	0.00	0.3634	0.3987
RoBERTa-large	64	16	0.016	-9.55	1.1015	0.95	0.3743	0.3987
RoBERTa-large	128	16	0.016	-9.17	1.1050	2.33	0.3705	0.3399
Gemma-2-2B	64	256	0.111	-11.43	1.0414	0.00	1.0414	0.9846
Gemma-2-2B	64	16	0.007	-7.88	10.6295	1.62	1.0275	0.9854
Gemma-2-2B	128	16	0.007	-7.72	5.4554	1.78	1.0464	1.0851
GPT-2-large	64	16	0.013	-7.50	17.0399	1.00	1.6354	1.3359
GPT-2-large	128	16	0.013	-7.00	35.4168	1.00	1.6032	1.3919