A DEFENSE OF THE QUADRATIC MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Due to the complexity of neural network loss landscapes, optimization theory is forced to rely on idealized models, and there is generally a tradeoff between how theoretically tractable the model is, and how accurately it describes the true optimization dynamics. In this work, we stress test the simplest possible model of optimization – the quadratic model – and show that it can be surprisingly predictive in an LLM setting with 150M parameters and 3B training tokens. Specifically, we show that Taylor expanding the model and the loss function at intermediate checkpoints through training can accurately predict the optimization dynamics over windows that can last up to 10% of training. Having established this agreement, we then turn to analyzing the structure of these local quadratic optimization problems through two lenses: the Hessian spectrum and local stability. Using Lanczos quadrature with extremely deep probes, we are able to estimate the Hessian spectrum deep into the tail, and we find a surprising amount of structure in both the eigenvalues and eigenvectors, which depends on the batch size, preconditioner, and training time. We also empirically test local linear stability at intermediate checkpoints and compare it to theoretical predictions to demonstrate that optimization in LLMs typically occurs at a stochastic edge of stability, whose nature is also determined by batch size. Our results indicate the quadratic model may be a theoretically tractable proxy for pretraining optimization dynamics.

1 INTRODUCTION

Large Language Model (LLM) optimization has quickly advanced through a feedback loop between fast experimentation and theoretical intuition. For pretraining, many design choices such as learning rate scheduling, batch-size scaling, and optimizer are, at least informally, motivated by traditional optimization theory (Polyak, 1964; Zhang et al., 2024b; Polyak & Juditsky, 1992; Meterez et al., 2025; Kidambi et al., 2018). However, it is unclear which of these models are actually descriptive of the dynamics of large-scale pretraining.

Arguably the simplest setting in which optimization theory can give sharp problem-specific predictions is the quadratic model. The advantage of this setting is that the training dynamics can be understood mode-by-mode across the Hessian spectrum, which enables instance-dependent analyses that are tight for a given Hessian spectrum. This makes it a useful sandbox for studying stability, learning rate scheduling, batch size scaling, acceleration, and preconditioning, and developing principled algorithms that transfer to neural network optimization (Meterez et al., 2025; 2026; Zou et al., 2021; Wu et al., 2018; Kidambi et al., 2018; Jain et al., 2018; Morwani et al., 2026; Zhang et al., 2024b; 2019; Paquette & Paquette, 2021; Zhang et al., 2024a; Wu et al., 2025; Ferbach et al., 2026; Zhang et al., 2026b; Varre et al., 2021; Varre & Flammarion, 2022; Bordelon & Pehlevan, 2021).

These works largely use the quadratic model as a useful theoretically tractable sandbox for testing algorithms and ideas, and take a leap of faith when applying them to neural networks. In this work, we aim to tighten this connection and stress test the quadratic model by asking:

To what extent does the quadratic model actually capture the training dynamics of LLMs?

In Section 2, we test this by training a 150M transformer on 3B tokens from FineWeb. We Taylor expand the model and the loss around intermediate checkpoints and compare the resulting optimization trajectories to the original. Our results indicate that these approximations can hold for up to 10% of

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.

training, and the quality of the approximation is significantly better near the end of training than at the start.

In Section 3, we estimate the spectra of the Hessian and the Gauss-Newton matrix along the training trajectory, both with and without the Adam preconditioner. We find that the spectrum often splits into a head, whose size is determined by the vocab size and is dominated by the unembedding layer, and a power-law tail that is *universal*: identical across batch sizes and unaffected by the Adam preconditioner. Based on these curves, we estimate the source and capacity exponents and find a capacity exponent of $\alpha \approx 1$ in the resolvable tail, and a source exponent of $\beta < 1$, which contradicts the summability assumption common in literature.

Finally, in Section 4, we analyze training stability by estimating how close each saved checkpoint is to the *edge of stability* (Défossez & Bach, 2015; Jain et al., 2018; Ma et al., 2018; Wu et al., 2018; Ma & Ying, 2021; Wu et al., 2022; Velikanov et al., 2022; Mulayoff & Michaeli, 2024; Cohen et al., 2021; Jastrzębski et al., 2018; Jastrzebski et al., 2020; Cohen et al., 2022; Lewkowycz et al., 2020; Ahn et al., 2022; Arora et al., 2022; Damian et al., 2022; Cohen et al., 2024). We observe that at small to medium batch sizes, training occurs in a noise dominated stochastic edge of stability regime, while at large batch sizes, training operates at the deterministic edge of stability as described in Cohen et al. (2021).

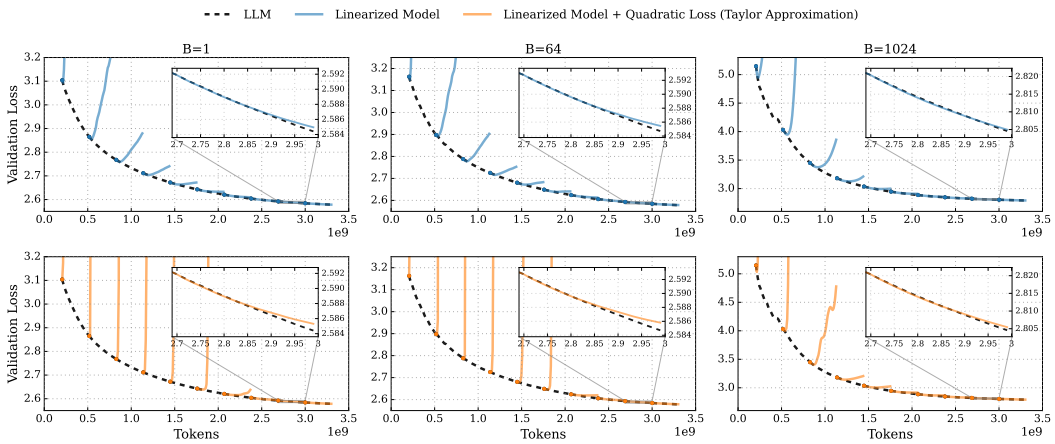

Figure 1: Validation loss curves for 150M models (black) trained with cosine for $1\times$ Chinchilla tokens, across batch sizes, at optimal learning rates. (Top) We show agreement between the LLM loss curve when compared to training the linearized model around checkpoints recorded every 10% of training (blue). (Bottom) We further quadratic Taylor expand the loss and train the quadratic model (orange). The losses are still in agreement for late training times, but diverge at early times, as opposed to the case where we only linearize the model. Note that the orange and blue curves are trained on the expansions, but we evaluate them on the true model loss. Table 1 (Appendix E) reports how long the approximation holds from each checkpoint. We provide further details in Section 2, and we also show a similar plot for a constant learning rate with EMA in Figure 11 (Appendix C).

We discuss additional related work on linearized dynamics, Hessian spectra, and optimization stability in Appendix A.

2 Quadratics are Predictive for Pretraining

Taylor's theorem is a central tool for studying smooth functions, as it allows complicated nonconvex objectives to be approximated locally by polynomials, which are often far easier to analyze. Studying arbitrary nonconvex objectives directly is often too broad to yield sharp predictions for real world training: worst-case analyses have to account for pathological functions that may be irrelevant to the training runs observed in practice. Taylor's theorem provides a natural way to make the problem local. Around a point on the training trajectory, a complicated objective can be approximated by a quadratic model whose behavior is governed by the local gradient and curvature. This turns the

question from one about arbitrary nonconvex optimization into a concrete empirical test: do local quadratic expansions during pretraining accurately predict the complete training dynamics?

2.1 LOCAL MODELS

Throughout the manuscript, we consider three related losses, suppressing the dependence on the minibatch for notational simplicity. Let ℓ be cross entropy loss, θ be the parameters, and f_θ be the model on a single sample so the loss is $\mathcal{L}(\theta) = \ell(f_\theta)$. Our first simplified loss, `prox`, linearizes f_θ around a reference point θ_{ref} but maintains the cross-entropy objective:

$$\mathcal{L}_{\text{prox}}(\theta) := \ell(f_\theta^{\text{lin}}) \quad \text{where} \quad f_\theta^{\text{lin}} = f_{\theta_{\text{ref}}} + \nabla_\theta f_{\theta_{\text{ref}}}(\theta - \theta_{\text{ref}}).$$

This simplification, also known as the NTK approximation (Jacot et al., 2018; Chizat et al., 2019) or the GN-proximal loss (Burke, 1985; Drusvyatskiy, 2017; Abreu et al., 2025), reduces the non-convex dynamics to solving a convex multi-class logistic regression problem where the samples are given by the network Jacobians at θ_{ref}.

We can further simplify by Taylor expanding the proximal loss to second order to get `quad`:

$$\mathcal{L}_{\text{quad}}(\theta) := \mathcal{L}(\theta_{\text{ref}}) + \nabla\mathcal{L}(\theta_{\text{ref}})(\theta - \theta_{\text{ref}}) + \tfrac{1}{2}(\theta - \theta_{\text{ref}})^T G(\theta - \theta_{\text{ref}})$$

where $G = \nabla^2\mathcal{L}_{\text{prox}}(\theta_{\text{ref}})$ is the Gauss-Newton matrix at θ_{ref}. Note that $\mathcal{L}_{\text{quad}}(\cdot)$ is not the usual second order Taylor expansion, as we use the Gauss-Newton matrix $G = \nabla^2\mathcal{L}_{\text{prox}}$, rather than the Hessian $H = \nabla^2\mathcal{L}$. This corresponds to keeping the positive semidefinite (PSD) term in the common Hessian decomposition:

$$H(\theta) \quad = \quad G(\theta) \quad + \quad \cancel{\nabla_f \ell(f_\theta)^T \nabla_\theta^2 f_\theta}.$$

Keeping only the Gauss–Newton term ensures that the quadratic objective is convex: its curvature matrix is positive semidefinite, so the local proxy cannot become unbounded below along negative-curvature directions. Thus, our quadratic model should be viewed as the second-order approximation induced by the linearized network, rather than the full Taylor expansion of the original model. We compare the Hessian and Gauss–Newton spectra in Section 3.2, and leave a more detailed study of the dropped term to future work. Experimental details are provided in Appendix E.

2.2 PRETRAINING AS A CHAIN OF LOCAL QUADRATIC PROBLEMS

For each pretraining run, we save checkpoints θ_k at every 10% of the total token budget. Starting from each θ_k, we continue training for the next 10% of the original budget, but replace the loss with its local expansion around $\theta_{\text{ref}} = \theta_k$: the optimizer is run on $\mathcal{L}_{\text{prox}}$ for the `prox` experiments and on $\mathcal{L}_{\text{quad}}$ for the `quad` experiments. To isolate the effect of the loss approximation, everything else is kept identical to the original run: Adam is initialized from the optimizer state at checkpoint k, we follow the original learning rate schedule, and we maintain the same 4% parameter EMA used for validation. After this continuation, we evaluate the resulting EMA parameters on the *true* nonlinear model and report the full validation loss $\mathcal{L}(\cdot)$.

We show our results in Figure 1 for cosine decay, and in Figure 11 (Appendix C) for a constant learning rate with EMA. While early time linearizations quickly diverge from the loss, middle to late time expansions match the ground truth for roughly 5–10% of the total training budget. In general, `prox` matches better than `quad` at early times as well, across all three batch sizes, and `quad` only begins to match after 60% of training. We leave a further study of why this is the case to future work.

3 THE HESSIAN OF LLMS

In this section, we will shed light on the structure of the population Hessian, which is arguably the most important quantity for quadratic optimization. Because Adam is a preconditioned method, we distinguish between the *raw* Hessian H and the *preconditioned* Hessian $H_P := P^{-1}H$ where $P := \text{diag}[\sqrt{\nu} + \epsilon]$ is the Adam preconditioner.[1] Note that the preconditioned Hessian has real

[1]We also bake in the CompleteP (Dey et al., 2026) multipliers into both the raw Hessian and preconditioned Hessian so that the contribution of different layers to the Hessian spectrum remains stable across scales.

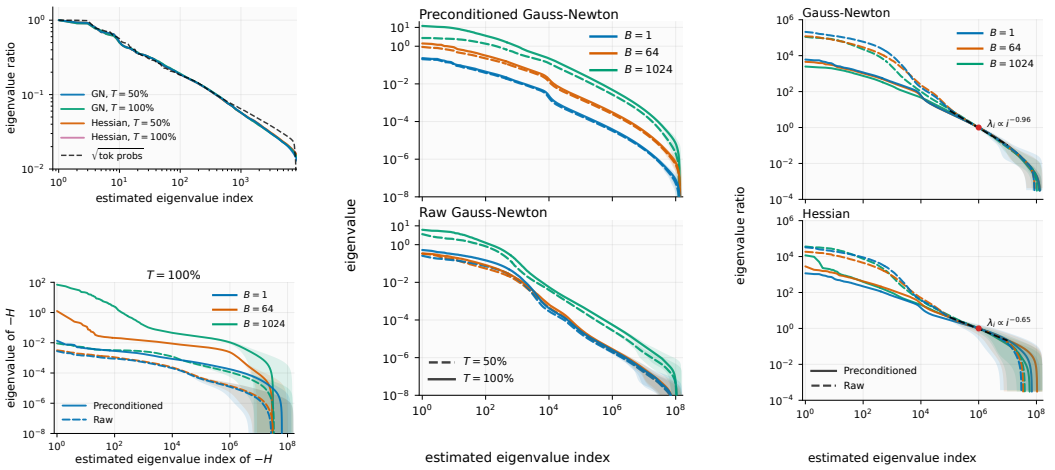

Figure 2: Analyses of the Hessian spectra for cosine decay. (Top left) Top eigenvalues at $B = 1$. (Bottom left) Negative Hessian eigenvalues. (Middle) Spectra at $T = 50\%, 100\%$. (Right) Final spectra aligned at $i = 10^6$. We provide a more detailed view of the spectrum in Figure 7 (Appendix B). See Figure 12 for the constant schedule.

eigenvalues as H_P is similar to $P^{-1/2}HP^{-1/2}$, which is symmetric. The raw Hessian governs the behavior of SGD, while the preconditioned Hessian governs the behavior of preconditioned methods like Adam (Cohen et al., 2022). We also distinguish between the Hessian of the linearized loss, the Gauss-Newton matrix G, and the Hessian of the original loss, the loss Hessian H. We will use G, H to refer to the raw Gauss-Newton and Hessian matrices, and use G_P, H_P to refer to the preconditioned versions.

3.1 Methodology

We estimate spectral decay, eigenvector distributions across parameter groups, and gradient–Hessian alignment using Lanczos quadrature (Golub & Meurant, 2009). For each checkpoint $B = 1, 64, 1024$ and $T = 10\%, 50\%, 100\%$, we probe G, H, G_P, H_P using one random start and one gradient start, at depth $m = 1200$, with matrix-vector products evaluated over 10M tokens and two-pass reorthogonalization in fp32. The shaded spectral bands are Gauss–Radau bounds accounting for finite Lanczos depth. Algorithms, implementation details, and accuracy ablations are provided in Appendix D.

3.2 The Hessian Spectrum

We summarize the spectra for the runs using cosine decay in Figure 2. We begin with two important points about plotting methodology and expected accuracy before interpreting the experimental results. Lanczos naturally estimates the map $i \to \lambda_i$. Specifically, the Lanczos recurrence guarantees that if $i^\star(\lambda)$ is the index of λ in the empirical spectral density $\mu(v_0)$ then $i^\star(\vartheta_j) \in [i_{j-1}, i_j]$ where $i_j := p\sum_{k=1}^{j} \omega_k$ are the partial sums of the Ritz weights. This guarantee can be extended to intermediate $\lambda \neq \vartheta_j$ using Gauss-Radau bounds (Algorithm 2). These guarantee that for any λ, $i^\star(\lambda) \in [i_-(\lambda), i_+(\lambda)]$. We can therefore create a uniform grid in log-space for λ, compute $i_\pm(\lambda)$, and shade the resulting band. We also plot a solid best fit curve for visualization purposes. A simple heuristic is that Lanczos can resolve the spectrum with small multiplicative error for eigenvalues $\lambda \gtrsim \lambda_1/m^2$. For $m = 1200$, this means we can resolve eigenvalues up to 1.5M times smaller than the top eigenvalue, which is consistent with the Gauss-Radau error bands in our experiments. We empirically found that the error bands for the Hessian are larger than for the Gauss Newton, which we attribute to the fact that Lanczos has to resolve both the positive and negative extremal eigenvalues before resolving the tail near $\lambda = 0$, which effectively halves its effective probe depth m.

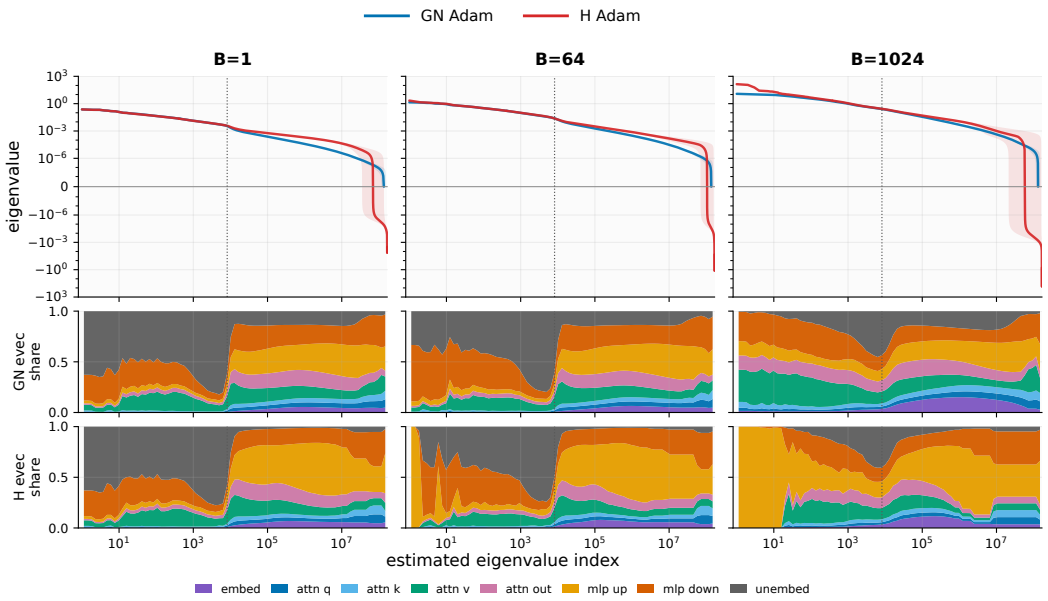

Figure 3: An estimate of the mass of each eigenvector in each parameter group using SLQ.

3.2.1 Experimental Results

Head-Tail Split: In most of the Hessian/Gauss Newton spectra, especially at smaller batch sizes, there is a sharp transition around $i = 8192$, which is our model's vocab size. In Figure 3, we show that this is not a coincidence as the top V eigenvectors are dominated by the $D \times V$ unembedding layer. We believe this may be related to neural collapse (Papyan, 2018; 2019; Papyan et al., 2020; Papyan, 2020), in which each token v has a unique unembedding vector $h_v \in S^{D-1}(\sqrt{D})$, and the Hessian of the last layer is well approximated by the rank V matrix: $\sum_{v \in [V]} p_v (h_v e_v^T)^{\otimes 2}$ where p_v are the marginal token probabilities. To test this, we plotted $\sqrt{p_v}$ against the top V eigenvalues of the preconditioned spectrum[2] and found a surprisingly good fit.

Spectrum Evolution: We observe that both the preconditioned and raw Hessian and Gauss Newton matrices do not change significantly between the 50% and 100% checkpoints. We fit the capacity exponent as well in Figure 2 (right) and see that in the tail we get roughly $\alpha \approx 0.96$ for Gauss-Newton and $\alpha \approx 0.65$ for Hessian. Note that as shown in Figure 2 (right), this exponent, when fitted on the tail, does not depend on batch size or whether we take into account the preconditioner. We observe that at small and medium batch sizes, the spectra of the G_P and H_P agree best in the head, while at large batches they agree best in the tails. Similarly, the spectra of the raw matrices G, H always agree in the head and agree best in the tail for $B = 1024$.

Tail Universality: Perhaps our most surprising finding is that regardless of the batch size or preconditioner, the tail of the spectrum appears to be ***universal***, meaning that the power law is identical and can only be shifted by an absolute constant. To visualize this, we aligned the spectra of both the raw and preconditioned Gauss Newton and Hessian matrices at $i = 10^6$. Especially for the Gauss Newton, the tail for $i \in [10^5, 10^7]$ appears to match for all batch sizes and regardless of whether the matrix is preconditioned, implying the Adam preconditioner does not affect the tail.

Negative Eigenvalues: Unsurprisingly, both the preconditioned Hessian and raw Hessian have negative eigenvalues, shown in Figure 2 (bottom left). Perhaps more surprising is that the negative eigenvalues of the preconditioned Hessian H_P are as large as the positive ones at $B = 64$ and significantly larger at $B = 1024$. In Figures 9 and 13 we can see that the corresponding eigenvectors are almost entirely localized to `mlp.up`, the MLP layer that reads from the residual stream. We hypothesize that these are related to dead neurons which actuate the rare token effect (see Appendix E).

[2]The square root is to account for Adam's preconditioner at small batch sizes

3.3 THE HESSIAN EIGENVECTORS

Because we maintain the basis V in Algorithm 1, we can also compute the Ritz vectors $u_1, \ldots, u_m$ associated to the Ritz values $\vartheta_1, \ldots, \vartheta_m$. This lets us estimate which parts of the network are contributing to which parts of the Hessian spectrum. Explicitly, given a unit vector v and a parameter block $\mathcal{I}$, we define the mass of v in $\mathcal{I}$ to be $\sum_{i \in \mathcal{I}} v_i^2$. Note that if we partition the parameters into disjoint subsets $\{I_1, \ldots, I_k\}$ then these masses must sum to 1. We plot the results in Figure 3 and highlight the following observations:

1) **The head is dominated by `unembed`:** In Section 3.2, we observed that for $B = 1, 64$, there was a clear split between the head and tail of the spectrum around $i = V$. In Figure 3, we can see that the corresponding eigenvectors live largely in the unembedding layer, which we connected to neural collapse (Papyan et al., 2020) in Section 3.2. This effect disappears for $B = 1024$. We attribute this to the fact that under neural collapse the Hessian of the unembedding layer is adapted to the standard coordinate basis, so full batch Adam is able to completely precondition it away (Cohen et al., 2024), while small batch Adam only applies a square root. After the head, the remaining eigenvectors appear to be spread evenly across the different types of layers, and the eigenvector mass is approximately proportional to the number of parameters in each layer.

2) **Large Hessian outliers are dominated by `mlp.up`:** We observed in Section 3.2 that at $B = 1024$, the Hessian has extreme outlier eigenvalues, both positive and negative. In Figure 3 (for a more complete plot across batch sizes and checkpoints, we refer the reader to Figure 9 for cosine and Figure 13 for constant) we can see that the corresponding eigenvectors live entirely in `mlp.up`. Our hypothesis is that nearly dead neurons in `mlp.up` cause the Adam second moment vector ν to collapse to 0, so that $\mathrm{diag}(\nu^{-1/2})H$ diverges in those directions. See Appendix E for additional discussion.

3.4 SOURCE CONDITIONS (HESSIAN-GRADIENT ALIGNMENT)

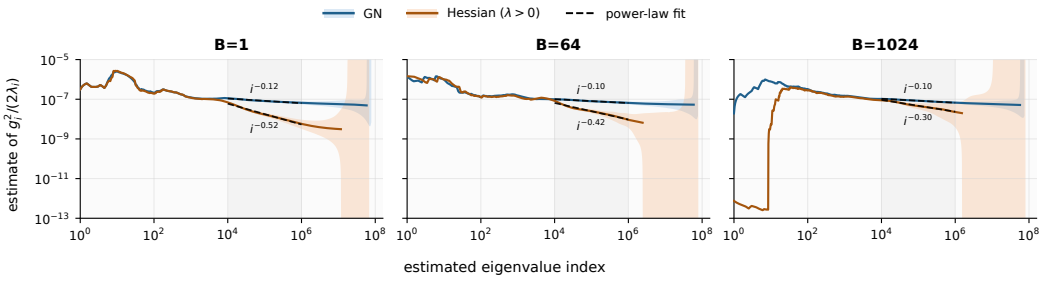

Figure 4: An estimate of the source condition $\langle u_i, \nabla L(\theta) \rangle^2 / \lambda_i$ using SLQ at the final checkpoint.

In addition to a random Lanczos probe, we also run Lanczos starting from the gradient, which captures the distribution of the gradient across the spectrum. In analyses of linear regression, this is often called the *source* condition and is parameterized by the exponent β:

$$\langle u_i, \nabla \mathcal{L}(\theta) \rangle^2 / \lambda_i \propto i^{-\beta} \qquad \lambda_i \propto i^{-\alpha} \tag{1}$$

where $\{(\lambda_i, u_i)\}$ are the eigenvalues and eigenvectors of the Hessian. This can also be interpreted as the amount of excess loss contained in the i-th eigenvector because for a quadratic objective:

$$\mathcal{L}(\theta) - \mathcal{L}^\star = \sum_{i=1}^{p} \frac{\langle u_i, \nabla \mathcal{L}(\theta) \rangle^2}{2\lambda_i}. \tag{2}$$

This easily generalizes to preconditioned methods. If we define the symmetrized preconditioned Hessian to be $P^{-1/2} H P^{-1/2}$ and replace the gradient with the semi-preconditioned gradient $g \leftarrow P^{-1/2} g$, then the same decomposition holds. In Figure 4, we estimate this source condition for both the preconditioned Gauss-Newton matrix and the preconditioned Hessian. We note that the map that Lanczos naturally estimates is the cumulative sum $i \to \sum_{j<i} (g \cdot u_j)^2$, so to estimate $i \to (g \cdot u_j)^2$,

we have to differentiate this map which is numerically unstable. We therefore first average the value over a multiplicative window of $[i/2, 2i]$ to smooth it before differentiating.

The primary observation is that in the part of the tail we can resolve, the source condition for the Gauss Newton matrix is consistently around $\beta \in [0.1, 0.2]$ while the source condition for the Hessian is closer to $\beta \in [0.3, 0.6]$. This contradicts the common assumption $\beta > 1$ in the literature on power laws, which guarantees eq. (2) is summable as $p \to \infty$ and, in the standard bias-variance decomposition, gives rise to power laws in the bias which depend on the source and capacity exponents. Instead, when $\beta < 1$, as we measure here, the problem actually behaves like it is finite dimensional (Paquette et al., 2024, Figure 2).

4 Edge of Stability

Cohen et al. (2021) demonstrated that full-batch optimization typically occurs at the *edge of stability* (EOS), meaning that increasing the learning rate to $(1+\epsilon)\eta$ would cause the optimizer to diverge on the local quadratic Taylor expansion, while decreasing it to $(1-\epsilon)\eta$ would cause the optimizer to converge. For gradient descent, this is captured by condition $\eta \approx 2/\lambda_1(H)$. This was later generalized to adaptive optimizers, including Adam, in Cohen et al. (2022) where they showed that if you freeze the preconditioner, adaptive optimizers also typically operate at the edge of stability.

These empirical results primarily hold in the full-batch (or at least very large batch) setting. However, a similar phenomenon occurs at smaller batch sizes. In particular, there is a *stochastic edge of stability* regime in which decreasing your batch size can cause you to diverge on the local quadratic Taylor expansion. In this section, we empirically test whether optimization in LLMs occurs at either the deterministic or stochastic edge of stability, and we compare these results with the theoretical predictions made by linear stability analysis.

4.1 Methodology

To test whether optimization occurs at the edge of stability, we reuse a similar experimental setup as in Appendix E. We Taylor expand the model using `quad`, around the EMA iterate $\bar{\theta}$, so that the objective on each minibatch is quadratic. We then freeze Adam's second moment buffer ν to the EMA $\bar{\nu}$. We then run preconditioned SGD with this fixed preconditioner for 10% of the total token budget, or until the loss diverges. To test how close a checkpoint is to the edge of stability, we run this experiment over a 5×5 grid of learning rate and batch size multipliers: $\{1/4, 1/2, 1, 2, 4\}$.[3] We also repeat this experiment over 10 random seeds, where each random seed shuffles the sequences in the dataset. We say a run diverges if its loss hits 100, and we plot the fraction of runs which diverge as a function of the learning rate and batch size multipliers in Figure 5. While we arbitrarily cap the loss divergence thresholds at 100, Figure 5 is incredibly stable for any cutoff between 10 and 10^5. Note that freezing the preconditioner leads to extreme outliers which can cause divergence. For the remainder of this section, we filter them out as explained in Appendix E.

4.2 Theoretical Predictions for Stability on Quadratics

The dynamics of SGD on a quadratic are a linear dynamical system. In particular, if $H_{\mathcal{B}}$ denotes the Hessian on a minibatch $\mathcal{B}$ then we can rewrite the SGD update as:

$$\begin{aligned} \theta - \theta^\star &\leftarrow \theta - \theta^\star - \eta \nabla \mathcal{L}_{\mathcal{B}}(\theta) && \text{SGD update} \\ &= \theta - \theta^\star - \eta[\nabla \mathcal{L}_{\mathcal{B}}(\theta) - \nabla \mathcal{L}_{\mathcal{B}}(\theta^\star)] - \eta \nabla \mathcal{L}_{\mathcal{B}}(\theta^\star) && \text{add and subtract} \\ &= \underbrace{(I - \eta H_{\mathcal{B}})}_{\text{stochastic contraction}}(\theta - \theta^\star) - \underbrace{\eta \nabla \mathcal{L}_{\mathcal{B}}(\theta^\star)}_{\text{additive noise}}. && \text{gradients are affine} \end{aligned} \tag{3}$$

In other words, at every step we multiply the current displacement to $\theta^\star$ by a random matrix $I - \eta H_{\mathcal{B}}$ then add a random noise vector independent of θ. For linear dynamical systems of this form, there are two relevant notions of stability: mean square stability and Lyapunov stability.

Definition 1. *A linear dynamical system θ_t is **mean square** stable if $\lim_{t\to\infty} \mathbb{E}[\|\theta_t\|^2] < \infty$, and **Lyapunov** stable if $\lim_{t\to\infty} \mathbb{E} \log \|\theta_t\| < \infty$.*

[3] For batch size 1 we could not use a batch size multiplier less than 1 so we generated a 5×3 grid.

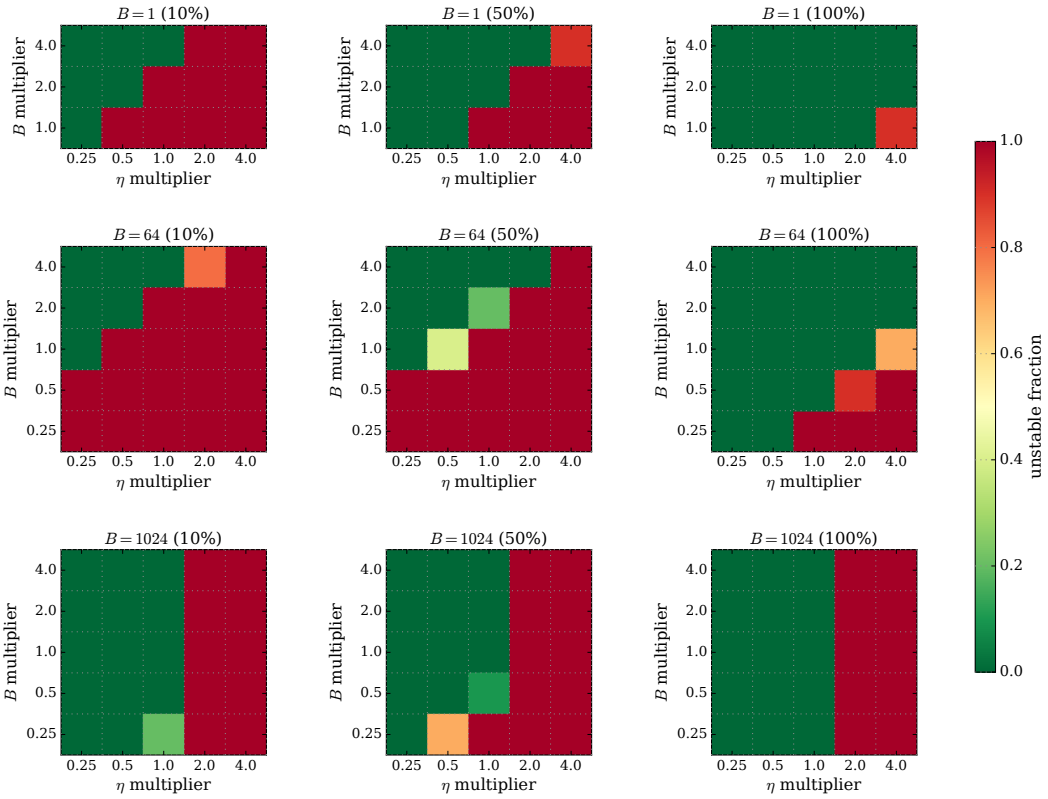

Figure 5: Heatmap showing instability for models trained with cosine decay as a function of (η, B) across batch size $B = 1$, 64 and 1024, at 10%, 50% and 100% of training. Cell color indicates how many seeds diverged out of a total of 10 seeds per run. We provide the analogous figure for constant learning rate in Figure 15 (Appendix C).

Note that by Jensen's inequality, mean square stability implies Lyapunov stability:

$$\mathbb{E}[\log \|\theta\|] = \tfrac{1}{2}\mathbb{E}[\log \|\theta\|^2] \leq \tfrac{1}{2} \log \mathbb{E}\|\theta\|^2.$$

For most processes, mean-square stability and Lyapunov stability nearly coincide. However, mean square stability is significantly more sensitive to heavy tails. As an illustrative example, let $\eta = B = 1$ and run SGD on the loss functions $\mathcal{L}_1 = \frac{1}{2}\theta^2$, $\mathcal{L}_2 = \frac{3}{2}\theta^2$. Then at every step you multiply θ by either $1 - \eta = 0$ or $1 - 3\eta = -2$, each with probability $1/2$. The marginal at time t is 0 with probability $1 - 2^{-t}$ and $(-2)^t$ with probability 2^{-t}. This process is mean square unstable because $\mathbb{E}[\theta^2] = 2^t \to \infty$, but it is Lyapunov stable, which reflects that $\theta_t \to 0$ almost surely. Mean-square stability captures wild fluctuations, while Lyapunov stability captures the "typical" behavior of the process. We will begin by briefly discussing mean-square stability before discussing Lyapunov stability, as it is more mathematically tractable and provides better intuition.

Mean Square Stability. There is a simple closed form for mean-square stability. If

$$\Sigma = \mathbb{E}[(\theta - \theta^\star)(\theta - \theta^\star)^\top]$$

then from eq. (3), Σ is updated by

$$\Sigma \leftarrow \mathbb{E}_{\mathcal{B}}[(I - \eta H_{\mathcal{B}})\Sigma(I - \eta H_{\mathcal{B}})] + \eta^2 C \quad \text{where} \quad C = \mathrm{Cov}_{\mathcal{B}}[\nabla \mathcal{L}_{\mathcal{B}}(\theta^\star)].$$

This is a deterministic linear system for Σ, so Σ will either converge or diverge depending on the eigenvalues of the linear operator $\Phi_{\eta,B}$:

$$\Phi_{\eta,B}[\Sigma] := \mathbb{E}_{\mathcal{B}}[(I - \eta H_{\mathcal{B}})\Sigma(I - \eta H_{\mathcal{B}})].$$

Stability of the Σ recurrence requires $\lambda_1(\Phi_{\eta,B}) \leq 1$. Because $\mathbb{E}\|\theta_t\|^2 = \mathrm{tr}\ \Sigma_t$, this is also the condition for mean square stability. Note that the learning rate η enters directly, and the batch size B

enters implicitly through the sampling of the minibatch $\mathcal{B}$. With some effort, it is possible to make these dependencies more explicit. For the remainder of this section, we will use $\tau := \eta/B$ to denote the "temperature" or learning rate to batch size ratio of the process.

Proposition 1 (Mulayoff & Michaeli, 2024, Theorem 5 & Proposition 6). *Let $\tau := \eta/B$ be the temperature and let $\mathcal{S}_{MSE} = \{(\eta, \tau) \mid \lambda_1(\Phi_{\eta,B}) < 1\}$ be the set of mean square stable hyperparameters for SGD. Then the stable set $\mathcal{S}_{MSE}$ satisfies the following properties:*

- *$\mathcal{S}_{MSE}$ is monotonic in η and τ, i.e. if $(\eta, \tau) \in \mathcal{S}_{MSE}$ then $(c_1\eta, c_2\tau) \in \mathcal{S}_{MSE}$ for any $c_1, c_2 \leq 1$.*
- *There is a maximum learning rate η_{crit} and maximum temperature τ_{crit}, i.e. $\mathcal{S}_{MSE} \subseteq [0, \eta_{crit}) \times [0, \tau_{crit})$. η_{crit} is given by $2/\lambda_1(H)$, as for GD, while τ_{crit} depends on the homogeneity of the minibatch Hessians.*

The effect of these observations is that in η, τ space, the set of hyperparameters for which SGD is mean-square stable is a rounded rectangle with edges $\eta < \eta_{\text{crit}}$ and $\tau < \tau_{\text{crit}}$. Transforming this into η, B space, this becomes a wedge shape. We provide an illustrative figure for these concepts in Figure 6 in Appendix B.

Lyapunov Stability. Lyapunov stability is significantly less structured than mean square stability. For example, Figure 6 shows a linear regression problem where (η, B) is Lyapunov unstable but $(\eta, 2B)$ is stable, so Lyapunov stability does not satisfy a monotonicity property. Furthermore, the set of Lyapunov stable hyperparameters is often not even connected. This distinction turns out to be relevant for LLMs. In our experiments, we observed settings where (η, B) is stable but $(\eta/2, B/2)$ is not stable, which violates monotonicity. This is inconsistent with mean-square stability and suggests that there are regimes during training in which SGD is Lyapunov stable but not mean square stable.

4.3 Experimental Results

Our experimental results demonstrate that the majority of training occurs within a factor of 2 of the edge of stability, and the shape of the plots is incredibly consistent with the theoretical predictions in Section 4.2. Figure 5 shows that for $B = 1$ and $B = 64$, the stability boundary follows the diagonal line, whereas at $B = 1024$ there is a flat cutoff, consistent with the illustration from Figure 6. Moreover, while the $(\eta, B) = (1, 1)$ cell is not always in the stability region, the $(0.5, 1)$ cell is always stable, indicating that every model is running at most a factor of 2 over the stability boundary, which is typical even for full-batch gradient descent (Cohen et al., 2021) and matches the conclusions of Cai et al. (2026). Our results also contextualize the finding of Cai et al. (2026) that the edge of stability disappears at small batch sizes. One confounder is that their probes run for a fixed number of steps (≤ 50), rather than a fixed number of tokens, so at small batch sizes the simulations process very few tokens and can fail to detect instabilities. Our results suggest that after normalizing by tokens, rather than steps, LLMs operate within a factor of 2 of the stochastic edge of stability at small batch sizes.

5 Conclusion and Limitations

Overall, our findings support a simple but powerful mental model: instead of asking for a single global theory of a highly nonconvex loss landscape, we can ask how the local quadratic model changes along the trajectory, how the optimizer interacts with its spectrum, and how we can leverage these insights to design more efficient training pipelines. In this sense, the quadratic model is not merely a toy problem, but an experimentally validated proxy for the optimization dynamics of pretraining, and a useful foundation for building theory that applies to real world models.

Limitations. Our experiments focus on a fixed model scale, architecture, and optimizer. While the agreement between the local and ground truth model is remarkable, it is an important future direction to explore how the model size and other optimizers influence the spectrum exponents and loss agreement. Moreover, early on in training, both approximations are much less accurate than at mid-to-late time, suggesting that higher-order terms in the loss play an important role in the early phase of pretraining. Finally, constant with EMA tracks the original very poorly (Figure 11). We speculate that this is related to the edge of stability behavior of the constant runs during training and we leave a further study to future work. Additional discussion is provided in Appendix F.

6 AI DISCLOSURE

In this work, we used generative AI tools for generating figures, proof checking the mathematical results, and proof reading the final manuscript. We have reviewed all AI-assisted work manually by deriving the mathematical formulations and the code, as well as text and plots. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Natalie Abreu, Nikhil Vyas, Sham Kakade, and Depen Morwani. The potential of second-order optimization for llms: A study with full gauss-newton. *arXiv preprint arXiv:2510.09378*, 2025.

Ryan P Adams, Jeffrey Pennington, Matthew J Johnson, Jamie Smith, Yaniv Ovadia, Brian Patton, and James Saunderson. Estimating the spectral density of large implicit matrices. *arXiv preprint arXiv:1802.03451*, 2018.

Kwangjun Ahn, Jingzhao Zhang, and Suvrit Sra. Understanding the unstable convergence of gradient descent. In *International conference on machine learning*, pp. 247–257. PMLR, 2022.

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

Arseniy Andreyev, Advikar Ananthkumar, Marc Walden, Tomaso Poggio, and Pierfrancesco Beneventano. Momentum further constrains sharpness at the edge of stochastic stability. *arXiv preprint arXiv:2604.14108*, 2026.

Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pp. 948–1024. PMLR, 2022.

Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. *arXiv preprint arXiv:2111.00034*, 2021.

Yu Bai and Jason D Lee. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. *arXiv preprint arXiv:1910.01619*, 2019.

Yu Bai, Ben Krause, Huan Wang, Caiming Xiong, and Richard Socher. Taylorized training: Towards better approximation of neural network training at finite width. *arXiv preprint arXiv:2002.04010*, 2020.

Blake Bordelon and Cengiz Pehlevan. Learning curves for sgd on structured features. *arXiv preprint arXiv:2106.02713*, 2021.

James V Burke. Descent methods for composite nondifferentiable optimization problems. *Mathematical Programming*, 33(3):260–279, 1985.

Yuhang Cai, Haofeng Huang, Haodong Wen, Deyi Liu, Yiyuan Ma, and Kaifeng Lyu. Does LLM pre-training typically occur at the edge of stability? In *Workshop on Scientific Methods for Understanding Deep Learning*, 2026. URL https://openreview.net/forum?id=QSb05IuPsy.

Avraam Chatzimichailidis, Janis Keuper, Franz-Josef Pfreundt, and Nicolas R Gauger. Gradvis: Visualization and second order analysis of optimization surfaces during the training of deep neural networks. In *2019 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, pp. 66–74. IEEE, 2019.

Tyler Chen, Thomas Trogdon, and Shashanka Ubaru. Analysis of stochastic lanczos quadrature for spectrum approximation. In *International Conference on Machine Learning*, pp. 1728–1739. PMLR, 2021.

Tyler Chen, Thomas Trogdon, and Shashanka Ubaru. Randomized matrix-free quadrature for spectrum and spectral sum approximation. *arXiv preprint arXiv:2204.01941*, pp. 55, 2022.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

Jeremy M Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E Dahl, et al. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

Jeremy M Cohen, Alex Damian, Ameet Talwalkar, J Zico Kolter, and Jason D Lee. Understanding optimization in deep learning with central flows. *arXiv preprint arXiv:2410.24206*, 2024.

Enea Monzio Compagnoni, Luca Biggio, Antonio Orvieto, Frank Norbert Proske, Hans Kersting, and Aurelien Lucchi. An sde for modeling sam: Theory and insights. In *International Conference on Machine Learning*, pp. 25209–25253. PMLR, 2023.

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. *arXiv preprint arXiv:2411.15958*, 2024.

Enea Monzio Compagnoni, Rustem Islamov, Frank Norbert Proske, Aurelien Lucchi, Antonio Orvieto, and Eduard Gorbunov. On the interaction of batch noise, adaptivity, and compression, under (l_0, l_1)-smoothness: An sde approach, 2026. URL https://arxiv.org/abs/2506.00181.

Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

Aaron Defazio. Schedulefree+: Scaling learning-rate-free & schedule-free learning to large language models. *arXiv preprint arXiv:2605.19095*, 2026.

Aaron Defazio, Ashok Cutkosky, Harsh Mehta, and Konstantin Mishchenko. Optimal linear decay learning rate schedules and further refinements. *arXiv preprint arXiv:2310.07831*, 2023.

Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *Advances in Neural Information Processing Systems*, 37: 9974–10007, 2024.

Alexandre Défossez and Francis Bach. Averaged least-mean-squares: Bias-variance trade-offs and optimal sampling distributions. In *Artificial Intelligence and Statistics*, pp. 205–213. PMLR, 2015.

Nolan Dey, Bin Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *Advances in Neural Information Processing Systems*, 38:137707–137739, 2026.

Kun Dong, David Eriksson, Hannes Nickisch, David Bindel, and Andrew G Wilson. Scalable log determinants for gaussian process kernel learning. *Advances in neural information processing systems*, 30, 2017.

Dmitriy Drusvyatskiy. The proximal point method revisited. *arXiv preprint arXiv:1712.06038*, 2017.

Damien Ferbach, Katie Everett, Gauthier Gidel, Elliot Paquette, and Courtney Paquette. Dimension-adapted momentum outscales sgd. *Advances in Neural Information Processing Systems*, 38: 112780–112977, 2026.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Gene H Golub and Gérard Meurant. *Matrices, moments and quadrature with applications*. Princeton University Press, 2009.

Diego Granziol. Hessformer: Hessians at foundation scale. *arXiv preprint arXiv:2505.11564*, 2025.

Diego Granziol and Khurshid Juarev. Hessian spectral analysis at foundation model scale. *arXiv preprint arXiv:2602.00816*, 2026.

Diego Granziol, Xingchen Wan, and Timur Garipov. Deep curvature suite. *arXiv preprint arXiv:1912.09656*, 2019.

Diego Granziol, Stefan Zohren, and Stephen Roberts. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *Journal of Machine Learning Research*, 23(173):1–65, 2022.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna B Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Prateek Jain, Sham M Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. *Journal of machine learning research*, 18(223):1–42, 2018.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*, 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. *arXiv preprint arXiv:2002.09572*, 2020.

Kaiqi Jiang, Jeremy Cohen, and Yuanzhi Li. Understanding the evolution of the neural tangent kernel at the edge of stability. *Advances in Neural Information Processing Systems*, 38:31996–32036, 2026.

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*, volume 2025, pp. 55966–56000, 2025.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=H1oyRlYgg.

Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, and Sham Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards*, 45(4): 255–282, 1950.

Clarissa Lauditi, Cengiz Pehlevan, and Blake Bordelon. Spectral dynamics in deep networks: Feature learning, outlier escape, and learning rate transfer. *arXiv preprint arXiv:2605.07870*, 2026.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

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.

Qianxiao Li, Cheng Tai, et al. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pp. 2101–2110. PMLR, 2017.

Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM review*, 58(1):34–65, 2016.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Philip M Long. Properties of the after kernel. *arXiv preprint arXiv:2105.10585*, 2021.

Chao Ma and Lexing Ying. On linear stability of sgd and input-smoothness of neural networks. *Advances in Neural Information Processing Systems*, 34:16805–16817, 2021.

Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. In *International Conference on Machine Learning*, pp. 3325–3334. PMLR, 2018.

Wesley J Maddox, Gregory Benton, and Andrew Gordon Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited. *arXiv preprint arXiv:2003.02139*, 2020.

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

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pp. 23610–23641. PMLR, 2023.

Alexandru Meterez, Depen Morwani, Jingfeng Wu, Costin-Andrei Oncescu, Cengiz Pehlevan, and Sham Kakade. Seesaw: Accelerating training by balancing learning rate and batch size scheduling. *arXiv preprint arXiv:2510.14717*, 2025.

Alexandru Meterez, Pranav Ajit Nair, Depen Morwani, Cengiz Pehlevan, and Sham Kakade. Anytime pretraining: Horizon-free learning-rate schedules with weight averaging. *arXiv preprint arXiv:2602.03702*, 2026.

Depen Morwani, Alexandru Meterez, Pranav Nair, and Sham Kakade. Compute efficiency and serial runtime tradeoffs for stochastic momentum methods. *arXiv preprint arXiv:2606.19179*, 2026.

Rotem Mulayoff and Tomer Michaeli. Exact mean square linear stability analysis for sgd. In *The Thirty Seventh Annual Conference on Learning Theory*, pp. 3915–3969. PMLR, 2024.

Lorenzo Noci, Alexandru Meterez, Thomas Hofmann, and Antonio Orvieto. Why do learning rates transfer? reconciling optimization and scaling limits for deep learning. *arXiv preprint arXiv:2402.17457*, 2024.

Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. What can linearized neural networks actually say about generalization? *Advances in Neural Information Processing Systems*, 34:8998–9010, 2021.

Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *arXiv preprint arXiv:1901.08244*, 2019.

Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020.

Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020.

Courtney Paquette and Elliot Paquette. Dynamics of stochastic momentum methods on large-scale, quadratic models. *Advances in Neural Information Processing Systems*, 34:9229–9240, 2021.

Elliot Paquette, Courtney Paquette, Lechao Xiao, and Jeffrey Pennington. 4+ 3 phases of compute-optimal neural scaling laws. *Advances in Neural Information Processing Systems*, 37:16459–16537, 2024.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Andres Potapczynski, Marc Finzi, Geoff Pleiss, and Andrew G Wilson. Cola: Exploiting compositional structure for automatic and efficient numerical linear algebra. *Advances in Neural Information Processing Systems*, 36:43894–43917, 2023.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

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. *arXiv preprint arXiv:2501.18965*, 2025.

Egor Shulgin, Dimitri von Rütte, Tianyue H Zhang, Niccolò Ajroldi, Bernhard Schölkopf, and Antonio Orvieto. Deriving hyperparameter scaling laws via modern optimization theory. *arXiv preprint arXiv:2603.15958*, 2026.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Aditya Varre and Nicolas Flammarion. Accelerated sgd for non-strongly-convex least squares. In *Conference on Learning Theory*, pp. 2062–2126. PMLR, 2022.

Aditya Vardhan Varre, Loucas Pillaud-Vivien, and Nicolas Flammarion. Last iterate convergence of sgd for least-squares in the interpolation regime. *Advances in Neural Information Processing Systems*, 34:21581–21591, 2021.

Maksim Velikanov, Denis Kuznedelev, and Dmitry Yarotsky. A view of mini-batch sgd via generating functions: conditions of convergence, phase transitions, benefit from negative momenta. *arXiv preprint arXiv:2206.11124*, 2022.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. Neurons in large language models: Dead, n-gram, positional. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1288–1301, 2024.

Nikhil Vyas, Yamini Bansal, and Preetum Nakkiran. Limitations of the ntk for understanding generalization in deep learning. *arXiv preprint arXiv:2206.10012*, 2022.

Jinbo Wang, Mingze Wang, Zhanpeng Zhou, Junchi Yan, Lei Wu, et al. The sharpness disparity principle in transformers for accelerating language model pre-training. *arXiv preprint arXiv:2502.19002*, 2025.

Jingfeng Wu, Peter L Bartlett, Sham M Kakade, Jason D Lee, and Bin Yu. Risk comparisons in linear regression: Implicit regularization dominates explicit regularization. *arXiv preprint arXiv:2509.17251*, 2025.

Lei Wu, Chao Ma, et al. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31, 2018.

Lei Wu, Mingze Wang, and Weijie Su. The alignment property of sgd noise and how it helps select flat minima: A stability analysis. *Advances in Neural Information Processing Systems*, 35:4680–4693, 2022.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pp. 581–590. IEEE, 2020.

Chenxiang Zhang, Rustem Islamov, Enea Monzio Compagnoni, Jun Pang, Aurelien Lucchi, and Antonio Orvieto. Beyond a single explanation of the adam–sgd gap. *arXiv preprint arXiv:2606.14259*, 2026a.

Dechen Zhang, Xuan Tang, Yingyu Liang, and Difan Zou. Scaling laws for precision in high-dimensional linear regression. *arXiv preprint arXiv:2602.19241*, 2026b.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.

Haihan Zhang, Yuanshi Liu, Qianwen Chen, and Cong Fang. The optimality of (accelerated) sgd for high-dimensional quadratic optimization. *arXiv preprint arXiv:2409.09745*, 2024a.

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? *arXiv preprint arXiv:2410.21676*, 2024b.

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need adam: A hessian perspective. *Advances in neural information processing systems*, 37:131786–131823, 2024c.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik Durk Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *International Conference on Learning Representations*, volume 2025, pp. 28033–28063, 2025.

Difan Zou, Jingfeng Wu, Vladimir Braverman, Quanquan Gu, and Sham Kakade. Benign overfitting of constant-stepsize sgd for linear regression. In *Conference on Learning Theory*, pp. 4633–4635. PMLR, 2021.

A Additional Related Work

Linearized Dynamics. The Neural Tangent Kernel (NTK) (Jacot et al., 2018; Lee et al., 2019) approximates a network through its first-order Taylor expansion in parameter space, an approximation which is accurate whenever the parameter displacement is relatively small. This regime is also known as "lazy" training (Chizat et al., 2019). However, in practice, the parameters and the NTK evolve rapidly as features form (Fort et al., 2020; Atanasov et al., 2021; Long, 2021; Vyas et al., 2022; Ortiz-Jiménez et al., 2021), rendering the NTK dynamics inaccurate for practical learning rates and widths. Several other works have shown that using the "after kernel", meaning the linearized model at the end of training, is a more faithful approximation for interpretability and fine-tuning (Park et al., 2023; Long, 2021; Malladi et al., 2023). A complementary line of work studies higher-order Taylor expansions (Bai & Lee, 2019; Bai et al., 2020) as a better approximation for the true dynamics. Relative to this literature, we linearize locally throughout the trajectory rather than at initialization, and we measure the agreement window as a fraction of the total training budget.

Hessian Spectra. Neural network Hessian spectra have been previously studied in order to understand the optimization loss landscape. While for small models it is possible to compute Hessian eigenvalues exactly (Sagun et al., 2016; 2017), either by calculating the Hessian in closed form or through a relatively small number of power iterations, this quickly breaks down when parameter counts are in the millions. A more refined algorithm, which is also the basis of Section 3.2, is the Stochastic Lanczos Quadrature (SLQ) algorithm (Lanczos, 1950; Golub & Meurant, 2009). Since its introduction, numerous works have studied this algorithm, providing accuracy bounds on the spectral estimates given by the quadrature algorithm (Ubaru et al., 2017; Adams et al., 2018; Chen et al., 2021; 2022; Lin et al., 2016). Amongst the first applications of SLQ in Machine Learning was done by Dong et al. (2017), and later extended to neural network landscapes (Maddox et al., 2020; Wang et al., 2025). In vision models, several papers reported that Hessians of trained classification models tend to separate into a bulk and outlier eigenvalues, roughly equal to the number of classes (Papyan, 2018; 2019; 2020; Ghorbani et al., 2019), an effect related to "neural collapse" (Papyan et al., 2020). Several open-source toolkits implement SLQ for neural network spectra, including the Tensorflow/JAX accompanying Ghorbani et al. (2019), PyHessian (Yao et al., 2020), CoLA (Potapczynski et al., 2023), GradVis (Chatzimichailidis et al., 2019), and the Deep Curvature suite (Granziol et al., 2019). More recently, empirical spectral measurements have been used to derive optimization prescriptions in neural network training. Granziol et al. (2022) use SLQ to derive an on-the-fly hyperparameter learner from the Hessian spectra during training. In transformers, Zhang et al. (2024c) found that the Hessian spectrum is very heterogenous across parameter blocks, and further used these findings to design Adam-mini, a more memory efficient version of Adam (Zhang et al., 2025). To the best of our knowledge, the first application of SLQ at up to 10B parameter scale for LLMs are Granziol (2025); Granziol & Juarev (2026), running SLQ at a depth of around $10 - 20$ probes, which gives a poor resolution in the spectrum tails. Finally, there is a wide body of literature established on the connections between the neural tangent kernel (NTK) eigenvalues, and the Gauss-Newton (Noci et al., 2024; Lauditi et al., 2026; Jiang et al., 2026; Kalra et al., 2025).

Linear Stability. The dynamics of SGD on a quadratic are a stochastic linear dynamical system. In the least squares setting with batch size 1, Défossez & Bach (2015) showed that the expected loss diverges when the step size exceeds a critical threshold determined by the fourth moment of the data. Jain et al. (2018); Ma et al. (2018) later extended this to larger batch sizes and showed that the maximal stable learning rate grows linearly in the batch size up to a critical batch size. Motivated by the fact that at a minimizer, the loss landscape is locally convex, Wu et al. (2018); Ma & Ying (2021); Wu et al. (2022); Velikanov et al. (2022); Mulayoff & Michaeli (2024) applied this stability analysis to describe which minima can be attractors for SGD. Linear stability in neural networks has also been extensively studied in the full batch setting under the name *edge of stability*. Cohen et al. (2021) showed that the largest Hessian eigenvalue typically increases during GD training until it reaches the critical stability threshold $2/\eta$ after which it oscillates for the remainder of training around this value. This extends empirical observations made by Jastrzębski et al. (2018); Jastrzebski et al. (2020). Cohen et al. (2022) also studied the edge of stability phenomenon for adaptive optimizers like Adam and showed that the right notion of local stability is given by the eigenvalues of the preconditioned Hessian. Several mechanisms have been proposed for how training is sustained at the edge of stability including the catapult effect for large initial learning rates (Lewkowycz et al., 2020)

a self-stabilization effect resulting from the cubic term in the loss Taylor expansion (Damian et al., 2022), and progressive sharpening driven by normalization (Arora et al., 2022). More recently, Cohen et al. (2024) have proposed an ODE capturing the dynamics of GD and adaptive optimizers at the edge of stability which empirically captures the true training dynamics. In the case of noise, a separate line of work has analyzed the dynamics of SGD and momentum at the edge of stability (Andreyev & Beneventano, 2024; Andreyev et al., 2026).

Most closely related to our stability experiments in Section 4, concurrent work by Cai et al. (2026) studies the edge of stability phenomenon in LLMs at scales up to 1.7B parameters. They probe the edge of stability by perturbing the learning rate at a fixed batch size, and observe that large batch training occurs at the edge of stability while training well below the critical batch size does not. However, because their probes only perturb the learning rate, they cannot distinguish instability caused by discretization from instability caused by stochasticity. Jointly perturbing (η, B) allows us to separate the two, and we find that at small batch sizes training occurs at a stochastic rather than deterministic edge of stability. Our setting also differs in that we use the linearized model and the corresponding Gauss-Newton matrices as our convex quadratic proxy, while Cai et al. (2026) instead add a proximal L2 penalty to the local problem which shifts the negative eigenvalues up to zero.

Prescriptive Optimization Theory. Lastly, we acknowledge two complementary lines of work striving to connect optimization theory with training practice, providing theoretically grounded prescriptions for hyperparameter selection in pretraining. The first shows that convergence bounds from convex optimization theory can be used to derive practical prescriptions for hyperparameter scaling in LLMs (Schaipp et al., 2025; Hägele et al., 2024; Shulgin et al., 2026; Defazio et al., 2023; Defazio, 2026; Defazio et al., 2024). The second models the discrete optimizer dynamics through stochastic differential equations, yielding scaling rules for hyperparameters across batch sizes and principled comparisons between adaptive methods (Li et al., 2017; Malladi et al., 2022; Compagnoni et al., 2024; 2026; 2023; Zhang et al., 2026a).

B ADDITIONAL PLOTS

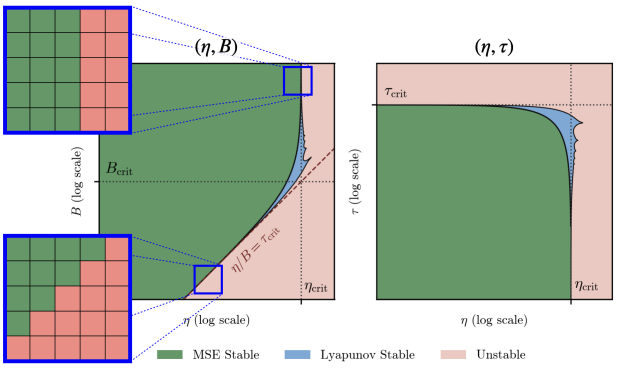

Figure 6: A toy simulation of the linear stability on a 1D regression problem where $H_i \overset{i.i.d.}{\sim} \text{Poisson}(B_{\text{crit}})$. Note the MSE stable region is monotonic in η, B, τ while the Lyapunov stable region is not. Our EOS experiments determine where on this cartoon training occurs by perturbing η, B.

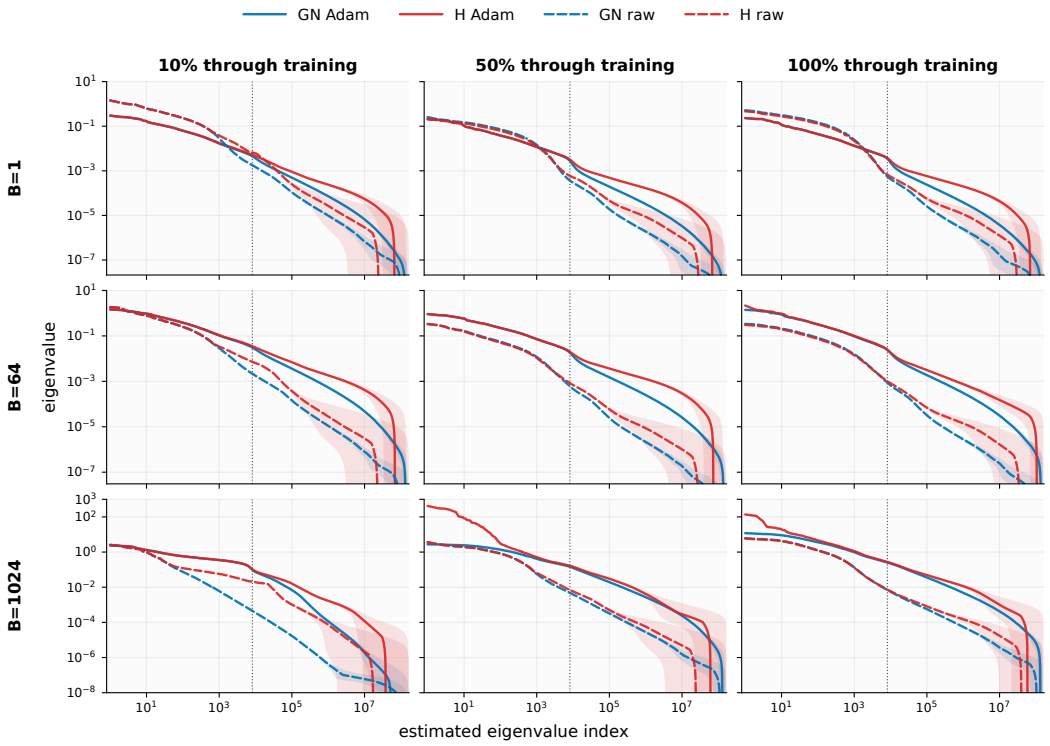

Figure 7: We estimate the spectrum of the Gauss Newton matrix, the Hessian, and their preconditioned versions at different checkpoints. The solid lines denote the best estimate for the spectrum, and the shaded bands represent error bands from finite depth Lanczos probes ($m = 1200$).

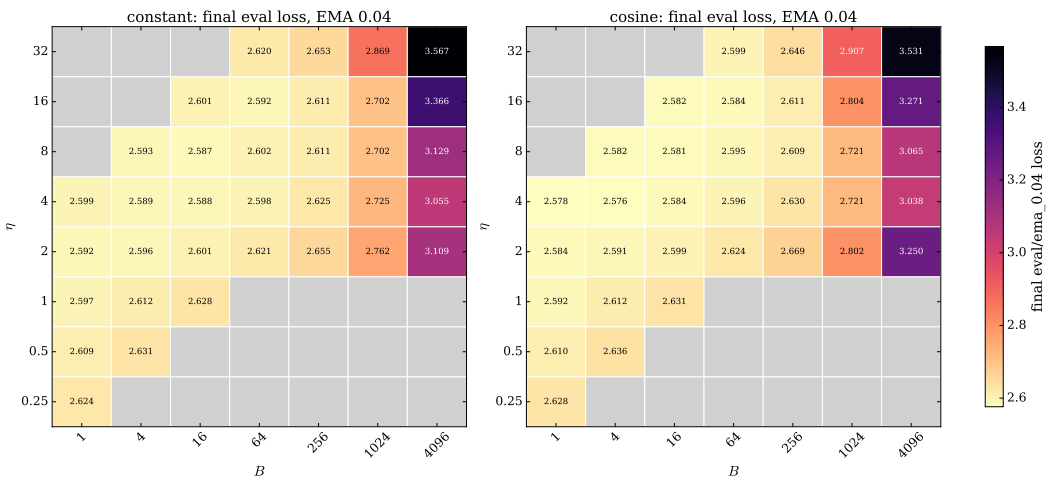

Figure 8: (η, B) pretraining sweep for the constant with EMA and cosine trained models.

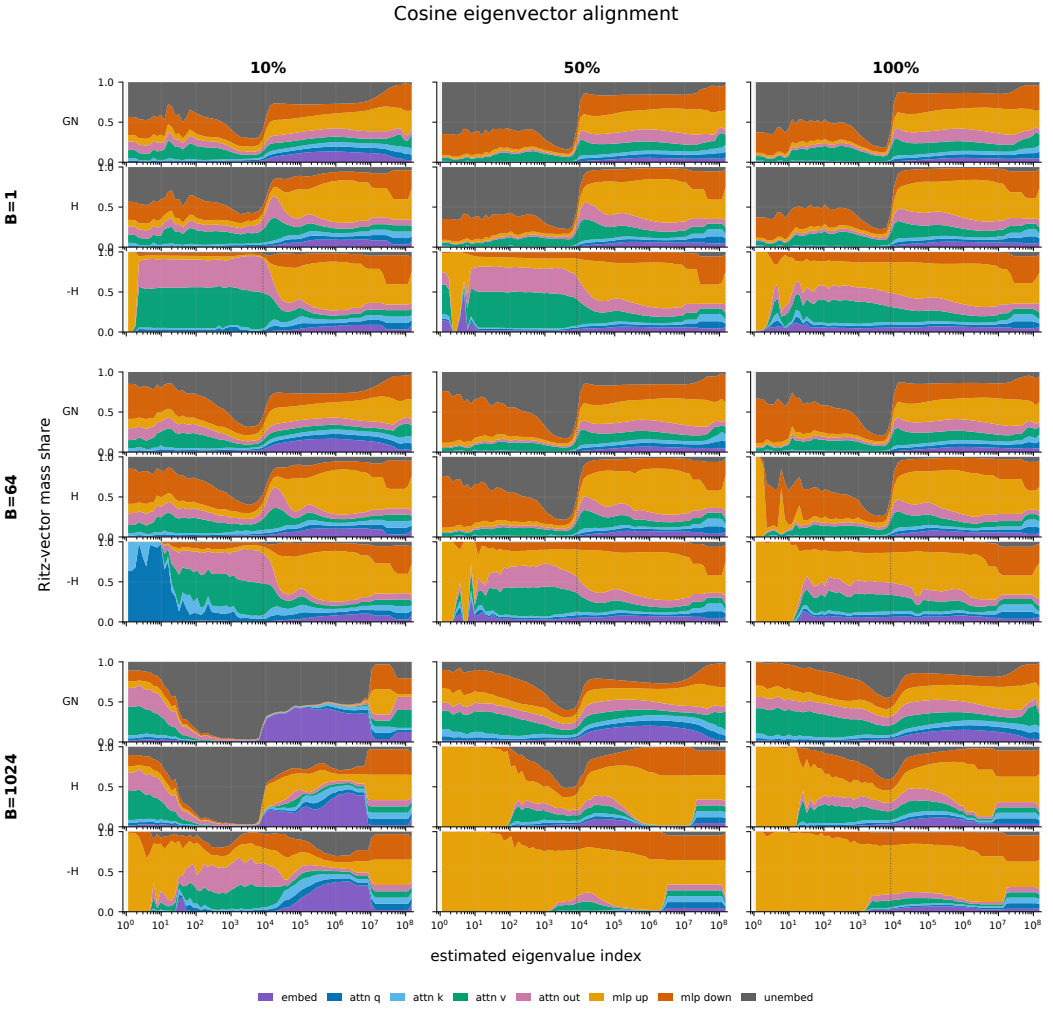

Figure 9: A full sweep of the eigenvector distribution at the end of training for cosine decay.

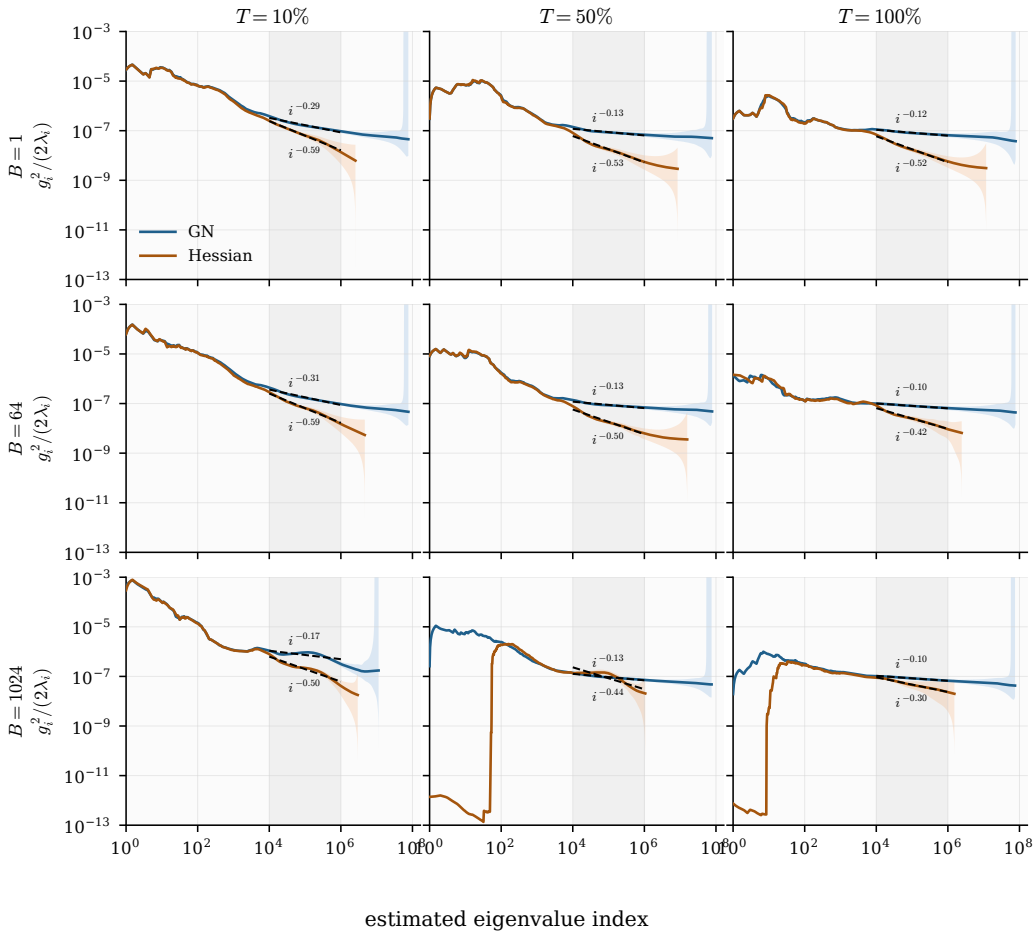

Figure 10: An extended version of Figure 4 with additional checkpoints. This is for the set of runs using cosine decay after warmup. See Figure 14 for the constant schedule runs.

C CONSTANT LEARNING RATE WITH EMA

All of the plots in the main paper were run with 10% warmup followed by cosine decay to 10% of the maximal learning rate. We also repeated the experiments with a constant learning rate schedule, which remains fixed after warmup.

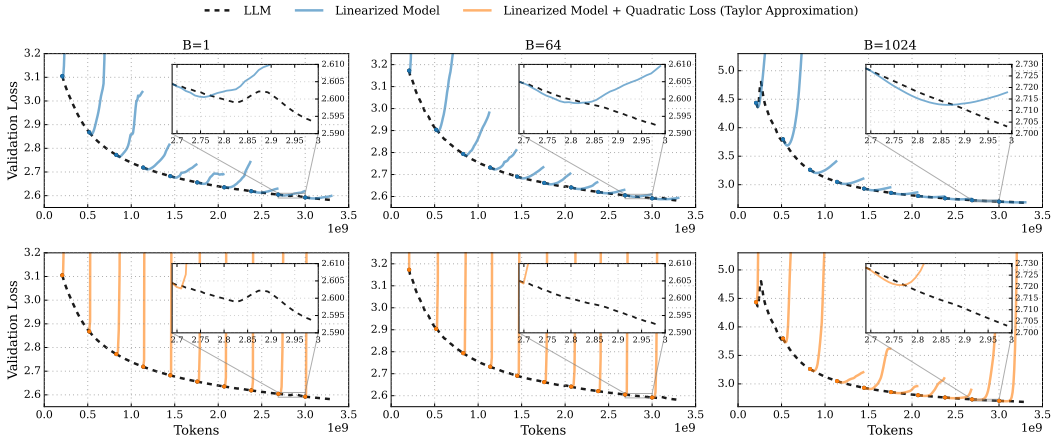

Figure 11: Validation loss curves for 150M models (black) trained with constant learning with EMA for $1\times$ Chinchilla tokens, across batch sizes, at optimal learning rates, under the same setup as Figure 1. The quadratic expansions are much more unstable, and we leave a study of this to future work. We provide further details in Section 2.

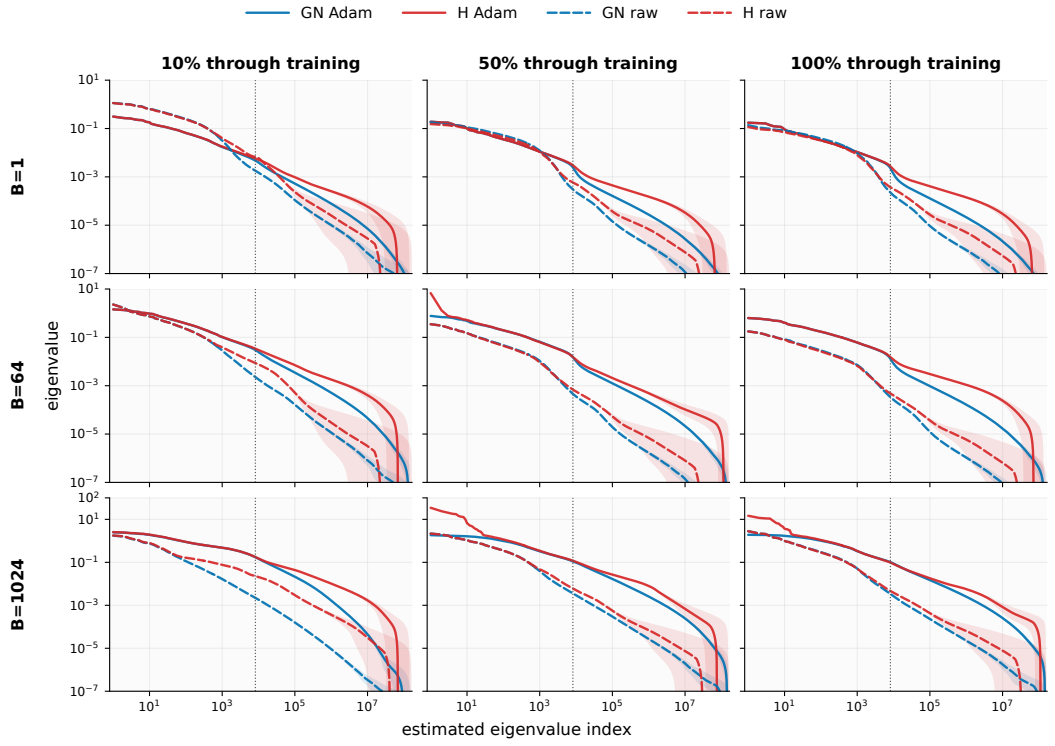

Figure 12: The spectra using a constant learning rate schedule after warmup, rather than cosine decay. This is otherwise equivalent to Figure 2.

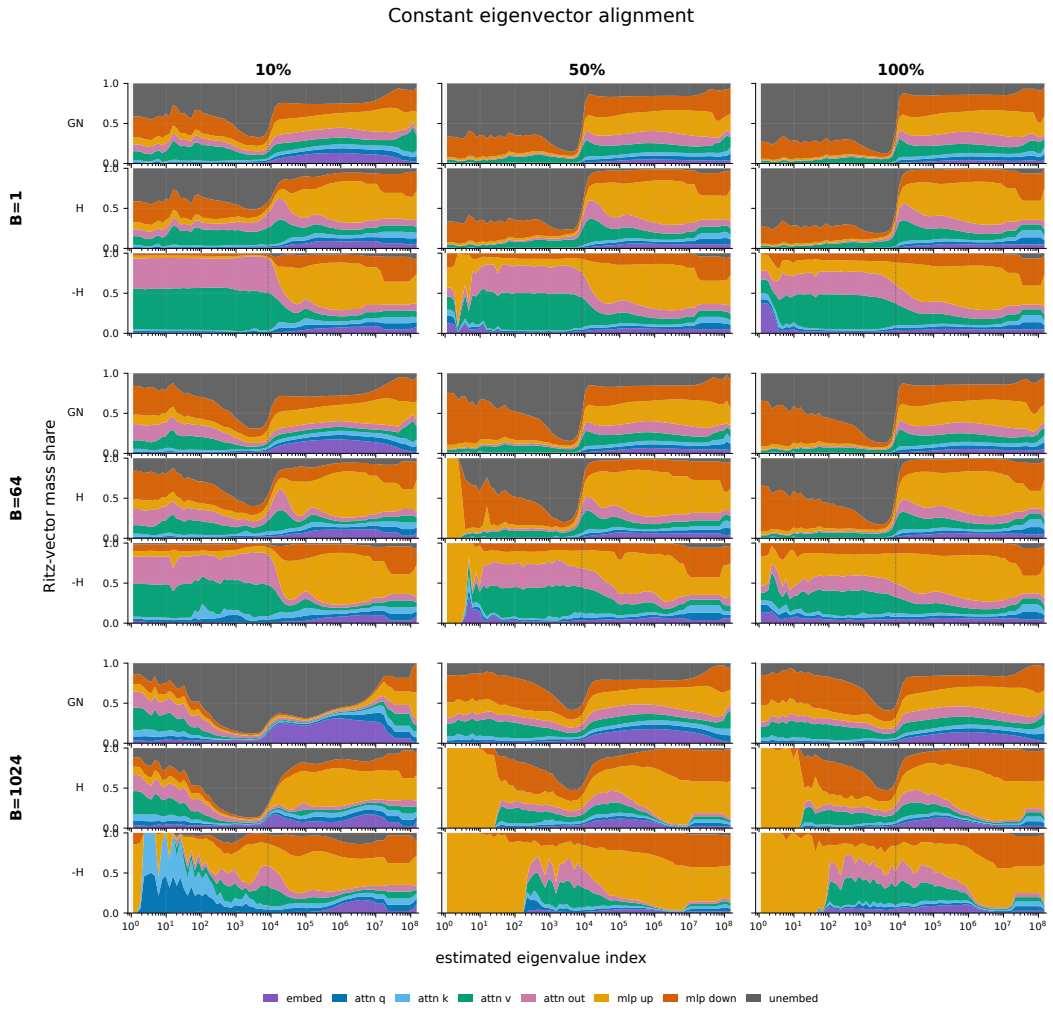

Figure 13: A full sweep of the eigenvector distribution at the end of training for the constant learning rate schedule. This is otherwise equivalent to Figures 3 and 9.

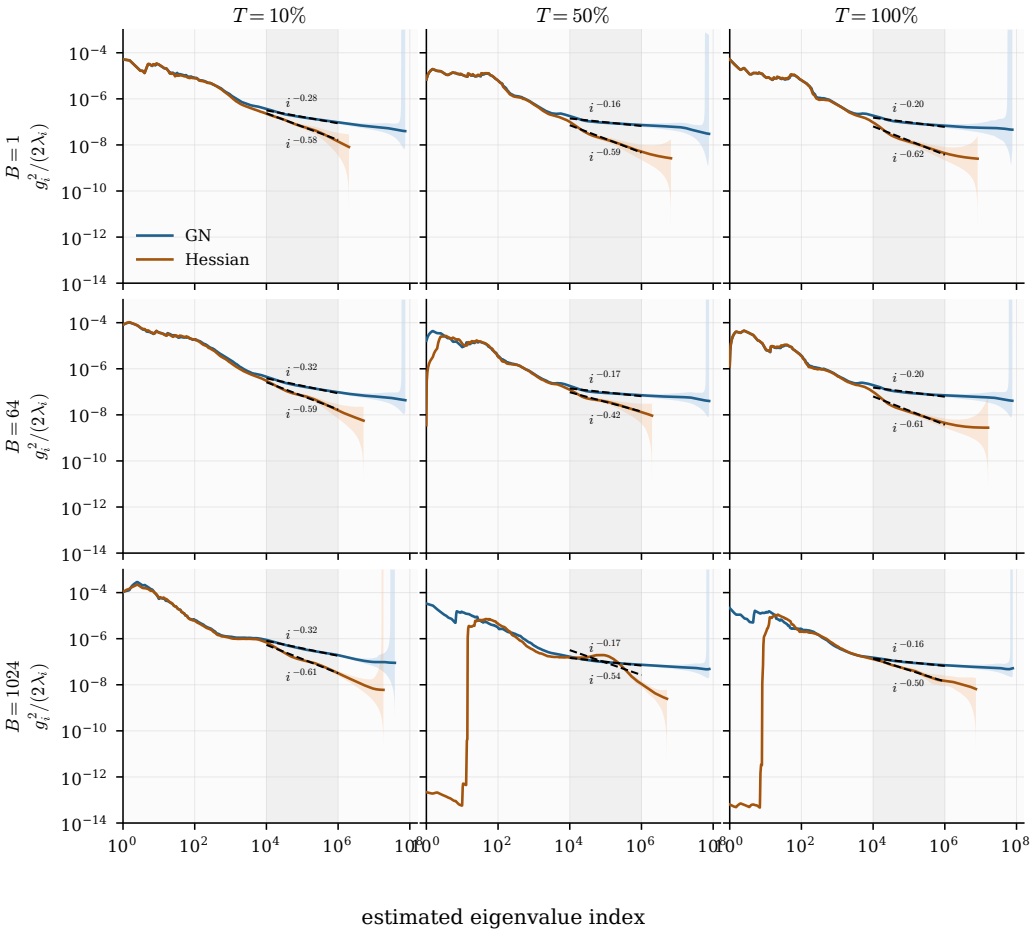

Figure 14: The source condition using a constant learning rate schedule after warmup, rather than cosine decay. This is otherwise equivalent to Figures 4 and 10.

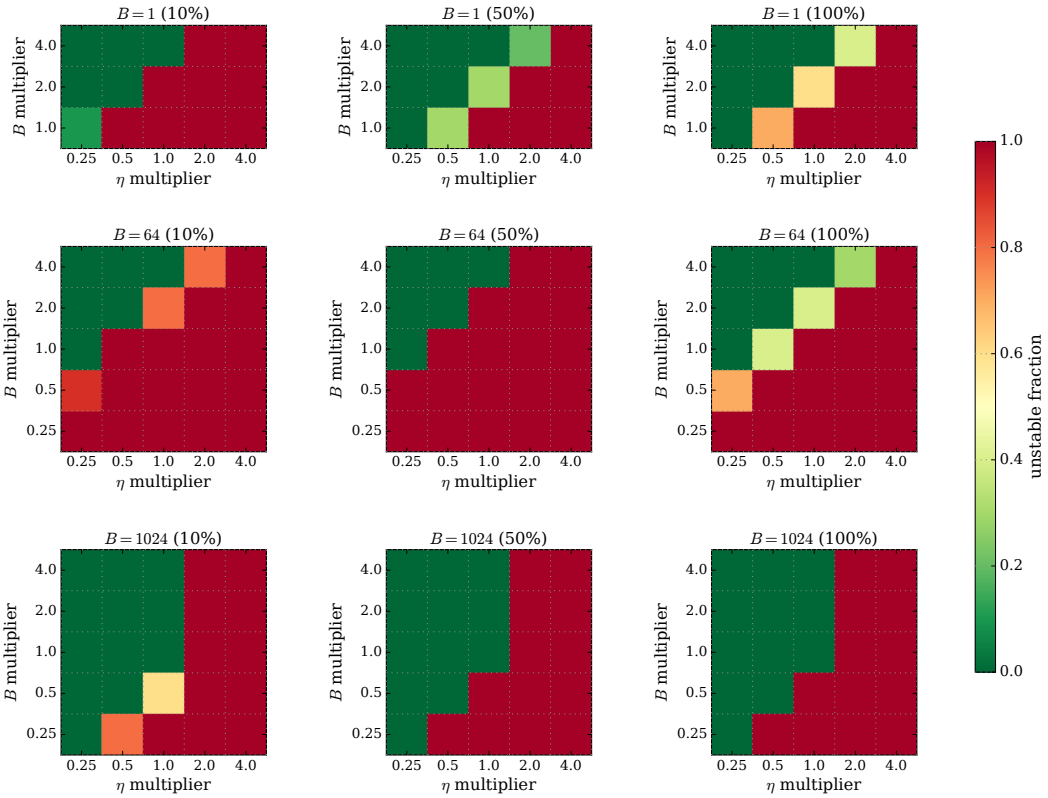

Figure 15: Heatmap showing instability for models trained with constant learning rate with EMA under the same setup as Figure 1. As in Figure 1, the smaller batch sizes operate near the stochastic edge of stability. However, because the stable region in the $B = 1024$ plots is diagonal then vertical, it operates at both the deterministic and stochastic edge of stability. This is the case whenever (η, B) is stable but both $(2\eta, 2B)$ and $(\eta, B/2)$ are unstable. This is distinct from our cosine schedule experiments for which $B = 1024$ operated entirely at the deterministic edge of stability.

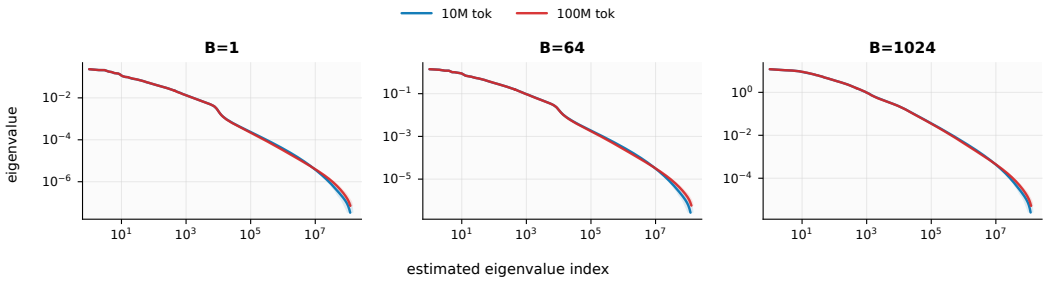

Figure 16: Comparison of the preconditioned Gauss Newton spectrum at $T = 100\%$ computed over 10M and 100M tokens, both at sequence length 1024. Each plot shows a different pretraining batch size, which is distinct from the number of samples used to estimate the Hessian. The spectra largely agree, even deep in the tail, so our remaining experiments estimate the Hessian over 10M tokens.

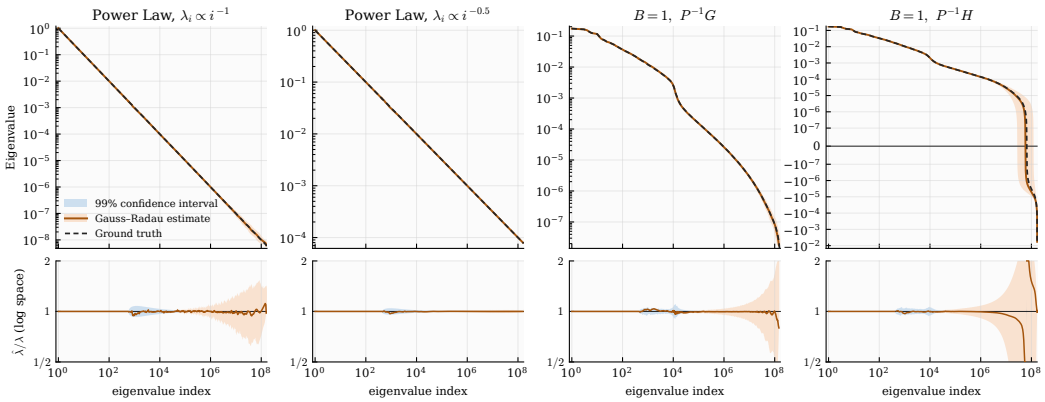

Figure 17: Comparing the ground truth (dashed black) with the SLQ estimate (orange) for various spectra. For the ground truth, we tested two power law spectra $\lambda_i \propto i^{-\alpha}$ along with the estimated spectra for $P^{-1}G$ and $P^{-1}H$ to check consistency with an additional SLQ probe. The light orange bands show the Gauss-Radau error bands from Algorithm 2 while the light blue bands show a 99% confidence interval for the error that comes from sampling a single random probe. The bottom row shows that the relative error $\log\left(\hat{\lambda}_i/\lambda_i\right)$ remains small deep in the spectrum.

D LANCZOS METHODOLOGY

Algorithm 1 Lanczos quadrature with reorthogonalization

Require: Symmetric matrix-vector product $v \mapsto Hv$, probe v_0, depth m

$q_1 \leftarrow v_0/\|v_0\|$
for $j = 1, \dots, m$ **do**
  $z \leftarrow Hq_j$
  $\alpha_j \leftarrow q_j^\top z$
  for $i = 1, \dots, j$ **do**
    $z \leftarrow z - q_i(q_i^\top z)$
  end for
  $\beta_j \leftarrow \|z\|$
  $q_{j+1} \leftarrow z/\beta_j$
end for
$T_m \leftarrow \mathrm{tridiag}(\beta_1, \dots, \beta_{m-1}; \alpha_1, \dots, \alpha_m)$
Diagonalize $T_m = U\mathrm{diag}(\vartheta)U^\top$
$\omega \leftarrow U_{1,:}^{\odot 2}$, $V \leftarrow QU$ where $Q = [q_1, \dots, q_m]$
return Ritz values, weights and vectors $\{(\vartheta_i, \omega_i, V_{:,i})\}_{i=1}^m$

Algorithm 2 Gauss–Radau quadrature

Require: Tridiagonal Lanczos matrix T_m, final residual β_m, dimension n, target value $\lambda \notin \mathrm{spec}(T_m)$

$\gamma \leftarrow \lambda + \beta_m^2 e_m^\top (T_m - \lambda I)^{-1} e_m$
$T \leftarrow \begin{pmatrix} T_m & \beta_m e_m \\ \beta_m e_m^\top & \gamma \end{pmatrix}$
$T = U\,\mathrm{diag}(\vartheta)U^\top$
$w_k \leftarrow nU_{1k}^2$ for $k = 1, \dots, m+1$
$i_-(\lambda) \leftarrow \sum_{k:\vartheta_k > \lambda} w_k$
$i_+(\lambda) \leftarrow i_-(\lambda) + w_j$ where $\vartheta_j = \lambda$
return interval $[i_-(\lambda), i_+(\lambda)]$

Naively, the Hessian spectrum is impossible to compute because a model with p parameters has a Hessian $H \in \mathbb{R}^{p \times p}$, which is far too large to store or eigen-decompose. However, we can efficiently compute Hessian-vector products $v \to Hv$ which allows us to sketch the Hessian and estimate its spectrum using Lanczos quadrature. For details, we refer the reader to Golub & Meurant (2009), but we describe the basic idea here. Given a starting vector v_0 and access to matrix-vector products $v \to Hv$, Lanczos sketches the distribution $\mu(v_0)$ where $\mu(v)$ is defined by:

$$\mu(v) := \sum_{i=1}^{p} (v \cdot u_i)^2 \delta_{\lambda_i}$$

and $\{(\lambda_i, u_i)\}$ are the eigenvalues and eigenvectors of H. This can be interpreted as a reweighted empirical spectral distribution in which each eigenvalue λ_i is given probability $(v \cdot u_i)^2$.

Using m Hessian vector products, the Lanczos recurrence returns m Ritz values $\vartheta_1, \dots, \vartheta_m$ and m weights $\omega_1, \dots, \omega_m$ which sum to 1, and produces the estimate:

$$\mu(v_0) \approx \sum_{i=1}^{m} \omega_i \delta_{\vartheta_i}.$$

This approximation is guaranteed to match up to the first $2m - 1$ moments. An imprecise but useful interpretation is that each Ritz value ϑ_i is a representative for a group of eigenvalues $\{\lambda_j\}_{j \in \mathcal{I}}$, and the corresponding weight $\omega_i \approx \sum_{j \in \mathcal{I}} (v_0 \cdot u_i)^2$ is the total weight of this group of eigenvalues.

In this section we use the Lanczos recurrence to inspect three properties of the Hessian:

1. **Spectral Decay:** estimate the map $i \to \lambda_i$ to look for power laws in the spectrum
2. **Eigenvector Distribution:** estimate the map $i \to \|\Pi_\ell u_i\|^2$ where Π_ℓ denotes the orthogonal projection onto layers of type ℓ (`embed`, `unembed`, `attn.q/k/v`, `mlp.up/down`)
3. **Gradient-Hessian Alignment:** estimate the map $i \to (\nabla \mathcal{L}(\theta) \cdot u_i)^2$

Experimental Details For (1) and (2) we use Lanczos probes with random $v_0 \sim S^{d-1}$ so that $\mathbb{E}_{v_0}\mu(v_0)$ is the empirical spectral distribution of H. For (3) we run an additional gradient-start probe with $v_0 = \nabla \mathcal{L}(\theta)$. For each of the checkpoints $B = 1, 64, 1024$ and $T = 10\%, 50\%, 100\%$ we run these Lanczos probes on the preconditioned Gauss Newton G_P, the preconditioned Hessian H_P, the raw Gauss Newton G, and the raw Hessian H. All Lanczos probes were run with a probe depth of $m = 1200$ and stored the full Lanczos basis with two-pass modified Gram–Schmidt reorthogonalization in fp32 precision. The matrix-vector products were computed over 10M tokens.

Accuracy and Ablations These large scale Lanczos runs require a lot of memory and compute, so we ran ablations to test which parameters were most important:

- **Probe count:** We empirically found that a single random probe was sufficient to resolve the spectrum $i \to \lambda_i$ to high accuracy, and given a fixed matrix-vector product budget, the primary bottleneck is always the Lanczos depth m, so we ran one random probe and one gradient probe for each (checkpoint, matrix) combination. This is justified by the fact that deep in the tails, the weight of each Ritz value is standing in for the average weight of a large block of eigenvalues, which creates an averaging effect even with a single probe.
- **Precision:** We found that the Lanczos three term recurrence quickly lost accuracy around depth 100 regardless of precision, so we elected to store the full Lanczos basis and do multi-pass reorthogonalization for all runs. We found that storing the basis in fp16 led to a drop in accuracy, but there was no drop in accuracy between fp32 and fp64, so we ran in fp32. The basis for a single checkpoint was around 750GB in memory which we sharded across 16 H100 GPUs.
- **Sample Size:** We compared the spectrum at 10M tokens and 100M tokens and found no significant difference, even deep in the tails, so we elected to compute the matrix-vector products using 10M tokens for compute efficiency (Figure 16).

We also tested our methodology against synthetic spectra for which the ground truth is known. These comparisons can be found in Figure 17. We found extremely strong agreement between the SLQ estimates and the ground truth spectra, even deep into the tails.

E EXPERIMENTAL DETAILS

Table 1: Percentage of the full training token budget for which the linearized loss predicts the true loss decrease from the starting checkpoint to within 10%, i.e., $\left|(\hat{\mathcal{L}}_t - \mathcal{L}_t)/(\mathcal{L}_{\text{ref}} - \mathcal{L}_t)\right| \leq 0.1$. Each continuation spans 10% of training, so the maximum entry is 10.

		Starting checkpoint									
B	Approximation	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
1	PROX	0.0	0.0	0.3	0.6	1.0	1.6	4.5	9.0	10.0	10.0
	QUAD	0.0	0.0	0.0	0.3	0.0	0.3	1.0	5.5	9.0	10.0
64	PROX	0.0	0.0	0.0	0.3	1.0	1.9	3.5	7.1	10.0	9.4
	QUAD	0.0	0.3	0.0	0.0	0.0	0.3	1.9	5.2	8.4	9.4
1024	PROX	0.3	0.3	1.6	1.9	2.3	3.9	7.7	7.4	10.0	10.0
	QUAD	0.3	0.3	0.3	1.3	1.9	2.6	4.8	6.5	10.0	5.8

We train a 150M (non-embedding) parameter transformer — 168M with embedding and unembedding matrices — on 3B tokens from FineWeb at batch sizes $B = 1, 64, 1024$, sequence length 1024, at the optimally tuned learning rates for each batch size (see Figure 8 for the full sweep). We train our models using Adam with no weight decay, warmup of 10%, $\epsilon = 10^{-8}$, $\beta_1 = 0.9$ and we dynamically set β_2 at step t to $\max(0.95, 1 - \frac{1}{0.01t})$, so the second moment buffer ν averages over the last 1% of tokens. For evaluation, we also track exponential moving averages (EMA) of both the parameters and the second moment buffer with parameter $\gamma_t = \max(0, 1 - \frac{1}{0.04t})$ which averages over the last 4% of tokens. For cosine, we decay to 10% of the peak learning rate.

For the architecture, we use a standard OLMO-style transformer with embedding dimension 1024, depth 12, 16 heads. We also use the CompleteP (Dey et al., 2026) parameterization to set the per-coordinate learning rates for Adam. To ensure our experiments are not dominated by embedding layers, we set our vocab size to 8192 by training our own byte-pair encoding (BPE) tokenizers. This ensures that the embedding and unembedding layers take up 10% of the total parameters, which matches the ratio for many larger models (Liu et al., 2024).[4] We note that with CompleteP multipliers, the learning rates appear much larger than those standard in the literature. This is because the intermediate layers in the model have a multiplier of $1/D$ over the base learning rate, where $D = 1024$. Thus $\eta = 1$ with CompleteP approximately corresponds to $\eta = 0.001$ under standard parameterization.

Extreme Outliers An important detail, which results from freezing the preconditioner, is that the `quad` training is now susceptible to divergence due to extreme outliers in the dynamics, which we observed were connected to rare tokens or dead neurons (Voita et al., 2024). We believe an illustrative example is to look at the dynamics of the embedding matrix E. In every minibatch, if token i does not occur then its corresponding embedding E_i gets no gradient. Because the second moment buffer ν is an EMA of these zero gradients, if the token isn't seen for roughly $\frac{1}{1-\beta_2}$ steps, then those entries of ν become exponentially small. If this token later reoccurs, then Adam will divide its gradient by $\sqrt{\nu} + \epsilon \approx 0$, which can cause a spike. While this phenomenon is clearest for the embedding matrix, it can also occur throughout the model, e.g. from neurons which are rarely activated (Voita et al., 2024). To circumvent this issue, we estimate the Frobenius norm of the preconditioned Gauss-Newton matrix for each sequence $\|P^{-1}G_i\|_F^2$ and filter out the top 1% outliers. Without this correction, nearly all hyperparameters diverged after a single outsized spike.

[4] Using the standard GPT-2 vocab size of 50257 would cause the embedding and unembedding layers to take up 40% of the total parameters, which is extremely unrealistic at larger scale.

F Additional Discussion

Quadratics as a Pretraining Sandbox. While LLM loss landscapes are not *globally* quadratic, the optimizer appears to enter regions that are well approximated by a prox-linear or Gauss-Newton Taylor expansion for a nontrivial fraction of the training budget (up to 10%). This may partially explain why intuition and algorithms for the quadratic model seem to extend to LLM pretraining.

Structure of the Local Quadratic. When the local dynamics are captured by the `quad` Taylor expansion, we can understand these dynamics through the spectrum of the Hessian/Gauss-Newton matrix. We estimated these spectra using deep Lanczos quadrature probes and found that the local quadratic models have extremely structured spectra, with power-law like behavior in both the head and the tail. We also find that the tails are *universal* meaning that their shape does not vary with training time or batch size and is not affected by the Adam preconditioner. The SLQ probes also allow us to localize different parts of the spectrum to different layers in the network.

Local Stability. Motivated by the observation that full-batch training typically occurs at the edge of stability (Cohen et al., 2021; 2022) and by theoretical analyses of linear stability of SGD (Mulayoff & Michaeli, 2024; Défossez & Bach, 2015), we empirically test whether LLM pretraining operates near a (stochastic) stability boundary. We first study the set of locally stable hyperparameters (η, B), and show how the shape of its boundary determines whether the instability is caused by discretization or stochasticity. We then locate where training occurs within this set by perturbing (η, B) and checking for instability. At small batch sizes, training typically occurs at a stochastic edge of stability, while at large batch sizes it typically occurs at a deterministic edge of stability.

Progressive Sharpening and Curvature Regularization. It has been widely observed that the stochasticity in SGD/Adam drives optimization towards flatter regions of the loss landscape Keskar et al. (2017); Jastrzębski et al. (2018). This curvature regularization effect cannot be captured by the quadratic model since the curvature is by definition fixed. Therefore, the fact that the quadratic model does seem to capture the training dynamics near the end of the training suggests that this curvature regularization effect is relatively weak by the end of training. However, we do observe evidence of progressive sharpening for our runs with cosine decay. As the learning rate decays, the runs leave the edge of stability and the curvature once again begins to grow, which matches the observations in Cai et al. (2026).