Neural Networks Loss Landscape Convergence in Hessian Low-Dimensional Space

Tem Nikitin
Moscow Institute of Physics and Technology
Moscow, Russia
nikitin.artem.a@phystech.su

Nikita Kiselev
Moscow Institute of Physics and Technology
Moscow, Russia
kiselev.ns@phystech.su

Andrey Grabovoy
Moscow Institute of Physics and Technology
Moscow, Russia
grabovoy.av@phystech.su

Vladislav Meshkov
Moscow Institute of Physics and Technology
Moscow, Russia
meshkov.ns@phystech.su

Abstract

Understanding how a neural network's loss landscape changes as we add more training data is important for efficient training. Although larger datasets are known to reshape this high-dimensional surface, the point when extra data stop making a big difference is still unclear.

In this paper, we study this issue and show that the loss landscape near a local minimum stabilizes once the dataset exceeds a certain size. To analyze this, we project the full parameter space onto a smaller subspace formed by the Hessian's top eigenvectors. This low-dimensional view highlights how the loss surface changes in its most important directions. We then apply Monte Carlo sampling within this subspace to estimate these changes more precisely.

We test our approach on standard image-classification tasks and find that our low-dimensional analysis pinpoints when the landscape stops evolving. These findings clarify how dataset size affects optimization and offer practical guidance for balancing training cost with performance gains.

Keywords: Neural networks, Loss landscape, Low-dimensional subspace, Hessian eigenvectors, Monte Carlo estimation, Dataset size threshold.

1 Introduction

Neural networks are now essential in many areas — image classification, language modeling, recommender systems, and more. As models and datasets grow, we often see better accuracy and new breakthroughs. But bigger networks and data also demand more computation time and resources, which can become prohibitive. While prior work has compared various optimization methods under fixed data and model sizes [1], the question of when adding more training samples stops yielding significant gains remains largely unanswered.

In this paper, we investigate exactly that: how the loss landscape changes as we increase the dataset size, and at what point further data have little effect. Knowing this "minimum viable dataset size" — a threshold beyond which new samples bring negligible improvement — can save both training time and data-collection effort. We also relate this threshold to generalization behavior, along the lines of [2].

Preprint. Under review.

To study this, we use the Hessian of the loss function as a proxy for local curvature. Computing the full Hessian is expensive, so we project the parameter space onto a low-dimensional subspace formed by its top eigenvectors. Our main contributions are:

1. Constructing a Hessian-based projection that captures the principal curvature directions.
2. Applying Monte Carlo method to identify when the loss landscape stabilizes as dataset size grows.
3. Visualizing the projected loss surface around local minima to illustrate these changes.
4. Deriving theoretical criteria — using results from The Matrix Cookbook [3] — to predict when additional samples have minimal impact.

This approach is novel in linking low-rank Hessian estimation to a concrete dataset threshold, enabling more cost-effective data collection and a clearer understanding of how data scale interacts with loss geometry. We validate our method on MNIST [4] and Fashion-MNIST [5], demonstrating how to balance computational cost against accuracy gains.

Organization. The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 introduces notation and preliminaries. Section 4 derives theoretical bounds on Hessian spectra and the loss-difference norm. Section 5 presents our empirical studies. We discuss the implications in Section 6 and conclude in Section 7. Additional experiments and proofs are provided in Appendix A.

2 Related Work

Hessian-based Analysis. The loss landscape of neural networks has been extensively studied using Hessian-based techniques, which illuminate convergence behavior and curvature properties [6]. Empirical work shows that near local minima the Hessian often has low effective rank, indicating a small "active subspace" spanned by a few large eigenvalues [7]. Ghorbani et al. [8] examined how these large eigenvalues emerge during training and shape optimization dynamics. Meshkov et al. [9] extended this analysis to fully connected and convolutional architectures, though broader validation across architectures remains an open challenge.

Loss Landscape Geometry and Dataset Size. Another research direction explores how dataset size influences model performance and landscape geometry. Networks trained on larger datasets tend to find flatter, wider minima that generalize better [2]. Yet, acquiring and processing massive data is costly, motivating studies of optimal data–model trade-offs [10]. Li et al. [11] developed visualization tools that reveal how dataset size, initialization, and architecture affect the topology of loss surfaces.

Algorithmic Stability and Landscape Sensitivity. Algorithmic stability, which measures how small changes in the training set affect the learned model, provides theoretical bounds on generalization error [12]. Elisseeff et al. [13] extended stability analysis to randomized algorithms such as bagging and ensemble methods, deriving non-asymptotic generalization guarantees. In our context, stability theory underpins the intuition that, beyond a certain dataset size, further samples cause negligible changes to the loss curvature. Our Monte Carlo–based Hessian projection method directly leverages this perspective, using top eigenvectors to quantify and visualize when the loss landscape stabilizes as new data are added.

3 Preliminaries

3.1 General notation

We consider a S-class classification problem. Let $\mathbf{x} \in \mathbf{X}$ be an input and $y \in \mathcal{Y} = \{1, \ldots, S\}$ its label. A neural network with parameters $\mathbf{w} \in \mathbf{\Omega} \subset \mathbb{R}^M$ defines a mapping $f_{\mathbf{w}} : \mathbf{X} \to \mathbb{R}^S$. Given a dataset

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$$

of N i.i.d. samples, and a per-sample loss $\ell\big(f_{\mathbf{w}}(\mathbf{x}_i), y_i\big)$ (e.g. cross–entropy), we define

$$\ell_i(\mathbf{w}) = \ell\big(f_{\mathbf{w}}(\mathbf{x}_i), y_i\big)\,.$$

Definition 1. *The empirical loss on the first k samples is*

$$\mathcal{L}_k(\mathbf{w}) = \frac{1}{k}\sum_{i=1}^{k}\ell_i(\mathbf{w}),$$

so that the full-sample loss is $\mathcal{L}(\mathbf{w}) = \mathcal{L}_N(\mathbf{w})$.

A simple telescoping gives the difference

$$\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w}) = \frac{\ell_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})}{k+1}\,.$$

Definition 2. *The Hessian of $\mathcal{L}_k$ at $\mathbf{w}$ is*

$$\mathbf{H}_k(\mathbf{w}) = \nabla^2_{\mathbf{w}}\mathcal{L}_k(\mathbf{w}) = \frac{1}{k}\sum_{i=1}^{k}\nabla^2_{\mathbf{w}}\ell_i(\mathbf{w}),$$

and, similarly, the full-sample Hessian is $\mathbf{H}(\mathbf{w}) = \mathbf{H}_N(\mathbf{w}) = \nabla^2_{\mathbf{w}}\mathcal{L}_N(\mathbf{w})$.

To capture the overall change in the landscape when adding one more sample, we introduce a Δ-function:

Definition 3. *Let $p(\mathbf{w})$ be a weighting (e.g. peaked near a local minimum). Define*

$$\Delta_k = \int \big(\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big)^2\, p(\mathbf{w})\, d\mathbf{w}.$$

We study the behavior of Δ_k as $k \to \infty$ and define a threshold:

Definition 4. *Fix a tolerance $\Delta > 0$. The* Δ-sufficient sample size *is*

$$K = \inf\{k \mid \forall\, k' \geq k :\ \Delta_{k'} < \Delta\}.$$

When $k \geq K$, adding further samples changes the loss landscape by less than Δ.

3.2 Assumptions

We make the following standard assumptions to simplify our analysis around a local minimizer $\mathbf{w}^*$.

Assumption 1. *The point $\mathbf{w}^*$ is a common local minimizer for both $\mathcal{L}_k$ and $\mathcal{L}_{k+1}$, i.e.*

$$\nabla\mathcal{L}_k(\mathbf{w}^*) = \nabla\mathcal{L}_{k+1}(\mathbf{w}^*) = 0.$$

Under this assumption, a second-order Taylor expansion of $\mathcal{L}_k$ about $\mathbf{w}^*$ gives

$$\mathcal{L}_k(\mathbf{w}) \approx \mathcal{L}_k(\mathbf{w}^*) + \tfrac{1}{2}\,(\mathbf{w} - \mathbf{w}^*)^{\mathsf{T}}\mathbf{H}_k(\mathbf{w}^*)\,(\mathbf{w} - \mathbf{w}^*).$$

Assumption 2. *At the minimizer $\mathbf{w}^*$, the loss values for consecutive sample sizes are the same:*

$$\mathcal{L}_{k+1}(\mathbf{w}^*) = \mathcal{L}_k(\mathbf{w}^*).$$

Assumption 3. *The parameters $\mathbf{w}$ are distributed according to a density $p(\mathbf{w})$ (e.g. a Gaussian prior centered at $\mathbf{w}^*$). Then*

$$\Delta_k = \int \big(\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big)^2\, p(\mathbf{w})\, d\mathbf{w} = \mathbb{E}\left(\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\right)^2 =$$

$$\mathbb{D}\big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big] + \Big(\mathbb{E}\big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big]\Big)^2.$$

4 Method: Projection onto Dominant Eigen-Directions and Loss Landscape Approximation

Working with the full Hessian $\mathbf{H}_k(\mathbf{w})$ of a neural network is computationally expensive due to the high dimensionality of the parameter space $\mathbf{\Omega}$. To reduce this cost, we project the parameters onto a lower-dimensional subspace spanned by the D dominant eigenvectors of the Hessian at a local minimum $\mathbf{w}^*$.

The justification for this projection is that, empirically, the Hessian has low effective rank near a minima [7]: only a few eigenvalues are significantly above zero, while the rest are negligible. Therefore, directions corresponding to small eigenvalues contribute little to loss variation. Retaining only the top D eigenvectors yields a compact yet informative representation of the most important curvature directions.

For illustration, Figure 1 plots the top few eigenvalues of $\mathbf{H}(\mathbf{w}^*)$ in descending order. We compute these dominant eigenvalues using the iterative power-method described at [14].

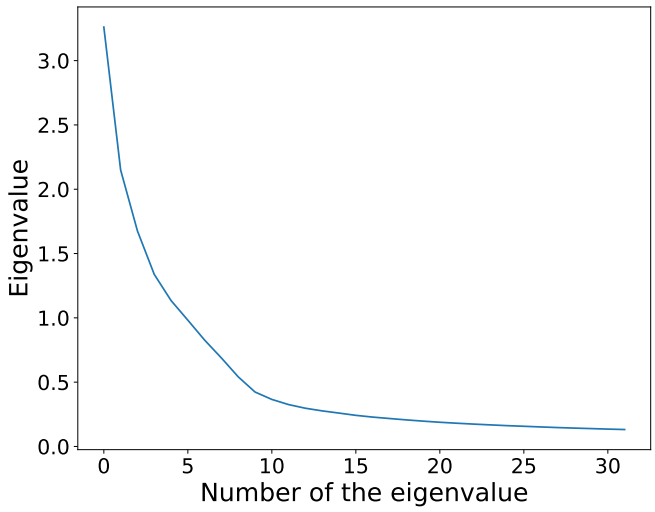

Figure 1: Top eigenvalues of $\mathbf{H}(\mathbf{w}^*)$ in descending order.

Concretely, let

$$\mathbf{P} = \big[\mathbf{e}_1, \ldots, \mathbf{e}_D\big]$$

be the matrix whose columns are the top D eigenvectors. We write any parameter vector as

$$\mathbf{w} = \mathbf{w}^* + \mathbf{P}\,\boldsymbol{\theta}, \quad \boldsymbol{\theta} \in \mathbb{R}^D.$$

Substituting into a second-order Taylor expansion of $\mathcal{L}_k$ around $\mathbf{w}^*$ gives

$$\mathcal{L}_k\big(\mathbf{w}^* + \mathbf{P}\,\boldsymbol{\theta}\big) \approx \mathcal{L}_k(\mathbf{w}^*) + \tfrac{1}{2}\,\boldsymbol{\theta}^{\mathsf{T}}\big(\mathbf{P}^{\mathsf{T}}\,\mathbf{H}_k(\mathbf{w}^*)\,\mathbf{P}\big)\,\boldsymbol{\theta} = \mathcal{L}_k(\mathbf{w}^*) + \tfrac{1}{2}\,\boldsymbol{\theta}^{\mathsf{T}}\,\mathbf{\Lambda}_k\,\boldsymbol{\theta},$$

where

$$\mathbf{\Lambda}_k = \mathbf{P}^{\mathsf{T}}\,\mathbf{H}_k(\mathbf{w}^*)\,\mathbf{P} = \operatorname{diag}\big(\lambda_k^{(1)}, \ldots, \lambda_k^{(D)}\big).$$

An identical expansion holds for $\mathcal{L}_{k+1}$, yielding $\mathbf{\Lambda}_{k+1}$ from the Hessian on $k+1$ samples.

Theorem 1 (Approximate Δ_k via Eigenvalues)**.** *Under the assumption that $\boldsymbol{\theta} \sim \mathcal{N}(0, \sigma^2\mathbf{I}_D)$, the change in loss can be estimated by*

$$\Delta_k = \mathbb{E}\big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big]^2 \approx \frac{\sigma^4}{4}\left(2\sum_{d=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})^2 + \Big(\sum_{d=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})\Big)^2\right).$$

Remark 1. *Assuming $\mathbb{E}(\boldsymbol{\theta}) = \mathbf{0}$ is natural when centering the sampling distribution at the minimum.*

This formula uses only the top D eigenvalues, so it is efficient to compute even for very large models. In practice, it provides an upper bound on the empirical Δ_k and thus a reliable criterion for determining when the dataset size is sufficient.

5 Experiments

To validate our theoretical estimates, we perform a set of experiments that track how the loss landscape changes as we grow the training set.

Datasets. We evaluate on MNIST [4] and Fashion-MNIST [5], each containing 60,000 training and 10,000 test grayscale images of size 28×28 pixels.

Model Architecture. All experiments use a simple multilayer perceptron (MLP) with ReLU activations. We keep the same architecture—number of hidden layers and hidden units—across all runs to ensure a fair comparison.

Experimental Procedure.

1. **Random-direction visualization 5.1:** As an initial sanity check, we project the parameters onto two random directions and plot the resulting 3D loss landscapes, their squared differences, and the corresponding Δ_k for various settings.
2. **Hessian-based analysis 5.2:** We repeat the above visualization using the top eigenvectors of the Hessian. This directly tests our hypothesis regarding the dominant role of the largest Hessian eigenvalues.
3. **Main experiment 5.3:** We compute both the theoretical and empirical values of Δ_k and compare them. This evaluates the accuracy of our approximation predicted by theory.

All code and detailed configuration files are available in our GitHub repository [1].

5.1 Preliminary Experiment: Projection onto Random Directions

5.1.1 Visualizing the Loss Landscape

To build intuition for how the loss surface changes with training-set size, we first project the model parameters onto two random directions. Specifically, we pick two random vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^P$ in parameter space and study the subspace they span.

We then evaluate the empirical loss

$$\mathcal{L}_k(\mathbf{w}) = \frac{1}{k} \sum_{i=1}^{k} \ell_i(\mathbf{w}),$$

where $\ell_i(\mathbf{w}) = \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$, over a grid in the (α, β) plane defined by

$$\mathbf{w}(\alpha, \beta) = \mathbf{w}^* + \alpha\, \mathbf{v}_1 + \beta\, \mathbf{v}_2, \quad (\alpha, \beta) \in [-1, 1]^2.$$

First, a uniform grid of (α, β) pairs on the domain $[-1, 1] \times [-1, 1]$ with fixed step size h is constructed. For each grid point (α, β), we set $\mathbf{w} = \mathbf{w}(\alpha, \beta)$, keeping the base weights $\mathbf{w}^*$ fixed, and compute the losses $\mathcal{L}_k(\mathbf{w})$ and $\mathcal{L}_{k+1}(\mathbf{w})$. This produces two scalar fields over the grid: the first, $\mathcal{L}_k(\alpha, \beta)$, illustrates how the mean loss varies across the parameter space, while the second,

$$\big(\mathcal{L}_{k+1}(\alpha, \beta) - \mathcal{L}_k(\alpha, \beta)\big)^2,$$

highlights the regions where adding a single sample induces the largest change. Finally, both fields are visualized as two adjacent 3D plots.

Loss Landscape for the Small Dataset In Figure 2, the loss surface is highly irregular, with sharp peaks and deep valleys. This irregularity indicates that with very few training samples, the model is extremely sensitive to small parameter perturbations. The squared difference surface also exhibits large spikes, confirming that adding a single data point can cause substantial local changes in the loss.

[1] GitHub

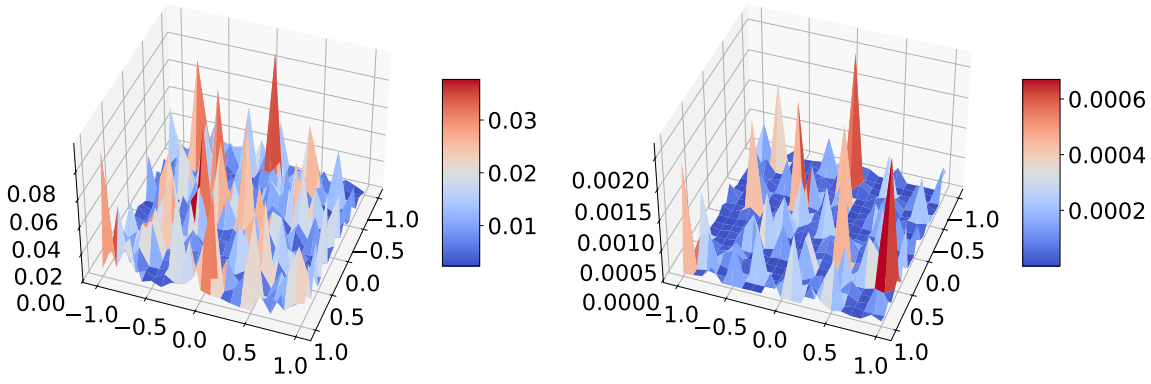

Figure 2: $\mathcal{L}_k$ (left) and $\big[\mathcal{L}_{k+1}(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta})\big]^2$ (right) for small k, projected onto the two random vectors.

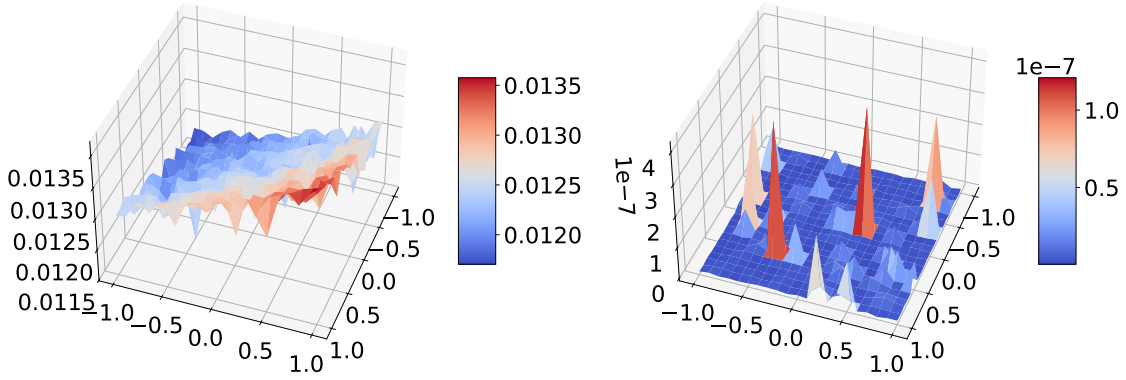

Figure 3: $\mathcal{L}_k$ (left) and $\big[\mathcal{L}_{k+1}(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta})\big]^2$ (right) for maximum k, projected onto the two random vectors.

Loss Landscape for the Full Dataset In contrast, Figure 3 shows a much smoother loss surface with gentle slopes. The overall loss values are lower — reflecting the averaging effect of more samples — and the squared difference surface is correspondingly much smaller. This smoothing demonstrates that as the dataset grows, the local optimum stabilizes, and additional samples produce only minor adjustments to the loss landscape. Hence, beyond a certain dataset size, further data have negligible impact on the shape of the loss surface.

5.1.2 Visualizing the Δ_k Function

To study how the loss landscape converges as the dataset grows, we estimate and plot the function

$$\Delta_k = \mathbb{E}\big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big]^2$$

via a Monte Carlo approximation using Gaussian perturbations in parameter space.

Starting from a fixed trained model to get $\mathbf{w}^*$, we draw T independent random direction vectors

$$\boldsymbol{\theta}^{(t)} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d), \quad t = 1, \ldots, T.$$

For each batch, we form a perturbed parameter set $\mathbf{w}^{(t)} = \mathbf{w}^* + \boldsymbol{\theta}^{(t)}$, and evaluate the losses $\mathcal{L}_k(\mathbf{w}^{(t)})$ for each $k = 1, \ldots, N$, after what compute the squared difference on given values:

$$d_t = \left[\mathcal{L}_{k+1}(\mathbf{w}^{(t)}) - \mathcal{L}_k(\mathbf{w}^{(t)})\right]^2.$$

We then approximate the average change $\Delta_k \approx \frac{1}{T}\sum_{t=1}^{T} d_t$, noting that increasing T reduces the variance of this Monte Carlo estimator. Finally, we plot Δ_k against k to observe its decay and include the rescaled quantity $\Delta_k \cdot k^2$ to highlight the expected $\mathcal{O}(1/k)$ convergence rate.

Figures 4 and 5 present the behavior of the function

$$\Delta_k = \mathbb{E}\left[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\right]^2$$

under different Monte Carlo sampling settings and projection subspace sizes. Each figure isolates one variable:

- Figure 4: varying the Gaussian variance σ.
- Figure 5: varying the subspace dimension D.

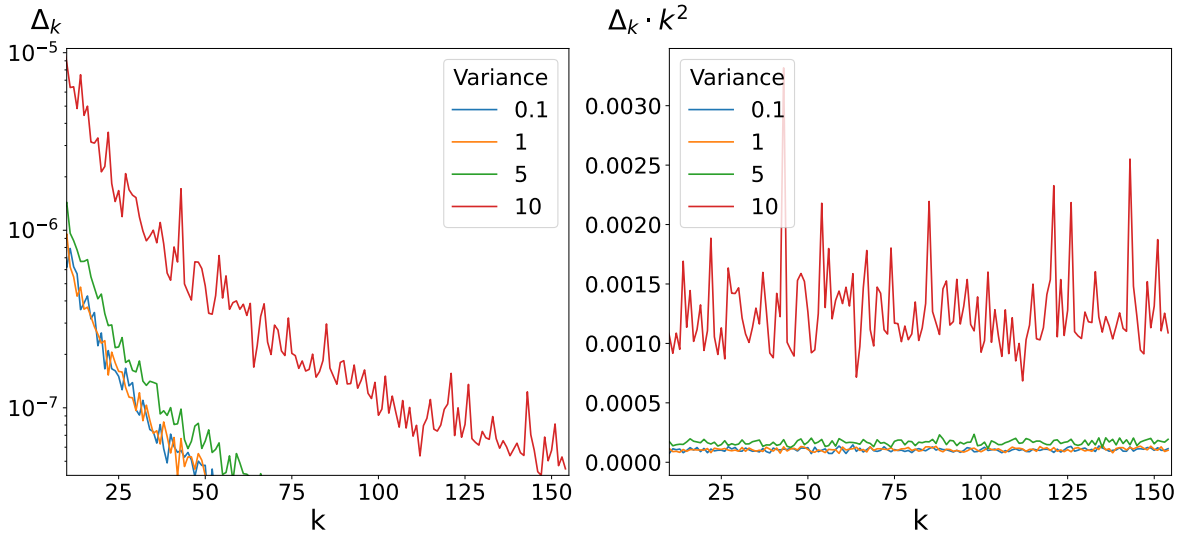

Figure 4: Δ_k (left) and $\Delta_k \cdot k^2$ (right), varying the Gaussian variance with fixed $d = 2$.

Comparing Different Variances σ Figure 4 shows that a high Gaussian variance σ produces a significant change in both the shape and magnitude of the Δ_k curves. This occurs because, as we move away from the local minimum, the value of the loss function changes. However, this change is modest: the insensitivity likely stems from using random projection directions, which do not align with the dominant curvature axes of the Hessian, so scaling those directions has little impact on the estimated loss differences. Moreover, variances not exceeding one yield very similar effects, so we choose $\sigma = 1$ for simplicity in our calculations.

Comparing Different Subspace Dimensions Figure 5 shows that, when using random directions, increasing the projection dimension D does not significantly alter the overall shape of Δ_k. This suggests that only a few directions dominate the variation — motivating our focus on Hessian-based projections in the main experiment.

In all cases, the right-hand plots (showing $\Delta_k \cdot k^2$) exhibit a roughly $\mathcal{O}(1/k^2)$ decay, confirming that the incremental effect of adding one more sample diminishes proportionally to $1/k$ as the dataset grows.

The preliminary experiments show that the loss landscape indeed smooths out and becomes more stable as the training set size N increases. However, because we used random projection directions, these results capture only a generic view of stability. This observation motivates our main experiment, in which we project onto the Hessian's top eigenvectors — i.e., the directions of greatest curvature — to obtain a more precise understanding of when and how the loss surface converges.

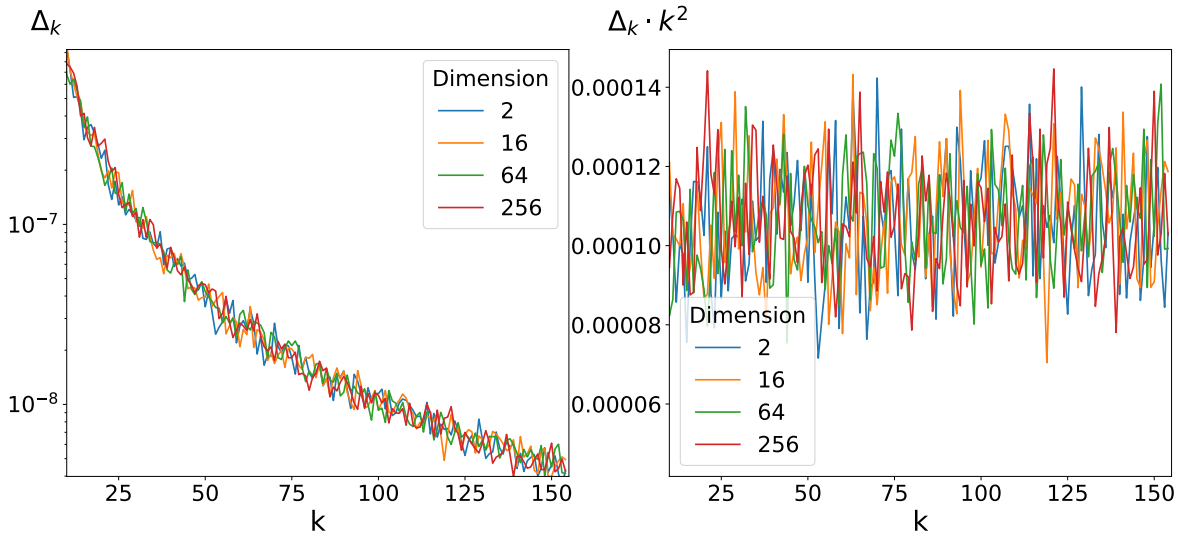

Figure 5: Δ_k (left) and $\Delta_k \cdot k^2$ (right), varying the subspace dimension with fixed $\sigma = 1$.

5.2 Moving Forward: Projection onto Dominant Hessian Eigenvectors

Projection Directions. In the main experiment, we replace random directions with the D eigenvectors of the Hessian at the local minimizer $\mathbf{w}^*$ that correspond to its largest eigenvalues. Prior work [7] shows these top modes capture nearly all of the non-negligible curvature, and Figure 1 confirms a rapid eigenvalue decay. Fixing $D = 10$ balances computational cost and fidelity, and our tests on Fashion-MNIST demonstrate that this choice suffices even for complex architectures.

Objectives. By projecting onto these principal directions, we aim to:

- Obtain a clearer view of how the loss surface changes as k grows, since these directions drive the largest loss variations.
- Accurately identify the threshold k^* at which further data cause only negligible shifts in the local curvature.
- Empirically validate our theoretical Δ_k estimates by comparing them against Monte Carlo measurements in this eigenvector subspace.

5.2.1 Visualizing the Loss Landscape

Figures 6 and 7 display 3D plots of $\mathcal{L}_k(\boldsymbol{\theta})$ (left) and $\left[\mathcal{L}_{k+1}(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta})\right]^2$ (right), where $\boldsymbol{\theta}$ varies over the span of the top two Hessian eigenvectors. The procedure mirrors the random-direction case but uses these maximally informative axes.

Loss Landscape for the Small Dataset In Figure 6, the loss surface projected onto the top two Hessian eigenvectors is noticeably more structured and less erratic than in the random-direction case. This confirms that these eigenvectors capture the most influential directions of loss variation. The squared-difference plot also shows larger peaks, indicating that the model is highly sensitive to new data along these key axes when k is small.

Loss Landscape for the Full Dataset By contrast, Figure 7 shows a much smoother loss surface and a greatly reduced squared-difference. As the dataset grows, the loss changes less along the principal curvature directions, and the model's local behavior stabilizes. Notably, even at large k, the squared difference remains higher than in the random-direction experiment, providing a practical upper bound on landscape fluctuations. These findings reinforce that projecting onto the top eigenvectors effectively reveals the convergence behavior of the loss landscape and helps intuitively understand the existence of the point beyond which additional data have minimal impact.

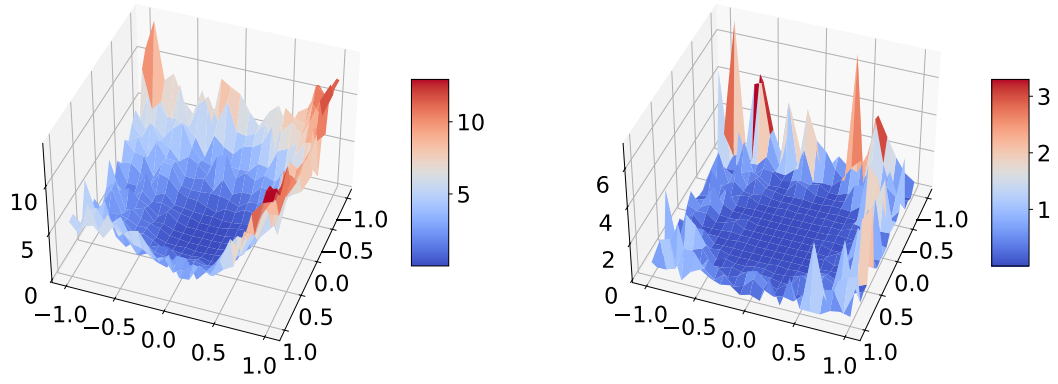

Figure 6: $\mathcal{L}_k$ (left) and $\left[\mathcal{L}_{k+1}(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta})\right]^2$ (right) for small k, projected onto the top two Hessian eigenvectors.

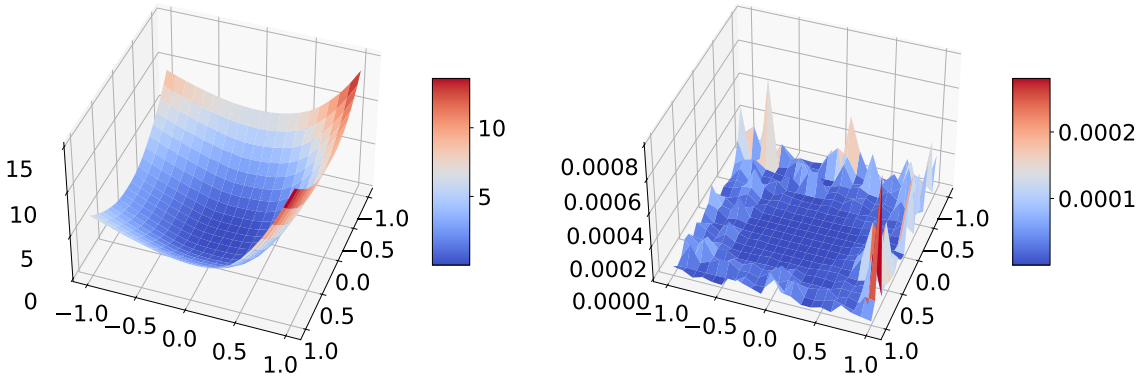

Figure 7: $\mathcal{L}_k$ (left) and $\left[\mathcal{L}_{k+1}(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta})\right]^2$ (right) for maximum k, projected onto the top two Hessian eigenvectors.

5.2.2 Visualizing the Δ_k Function

Figure 8 presents the Monte Carlo estimate of Δ_k in the subspace spanned by the top $d = 10$ Hessian eigenvectors. Compared to the random-direction experiment, Δ_k here has a higher value — reflecting the large curvature captured by these eigenvectors. The right-hand plot of $\Delta_k \cdot k^2$ confirms an approximate $\mathcal{O}(1/k^2)$ convergence rate. These observations show that projecting onto the leading eigen-directions not only highlights the greatest loss sensitivity but also yields a reliable upper bound on how quickly the landscape stabilizes as more data are added.

5.3 Main Experiment: Theoretical vs. Empirical Δ_k

In this experiment, we compare our Hessian-based theoretical estimate of Δ_k with the empirical Monte Carlo measurements.

As the Theorem 1 says, we approximate Δ_k by

$$\Delta_k \approx \frac{\sigma^4}{4}\left(2\sum_{d=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})^2 + \left(\sum_{i=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})\right)^2\right).$$

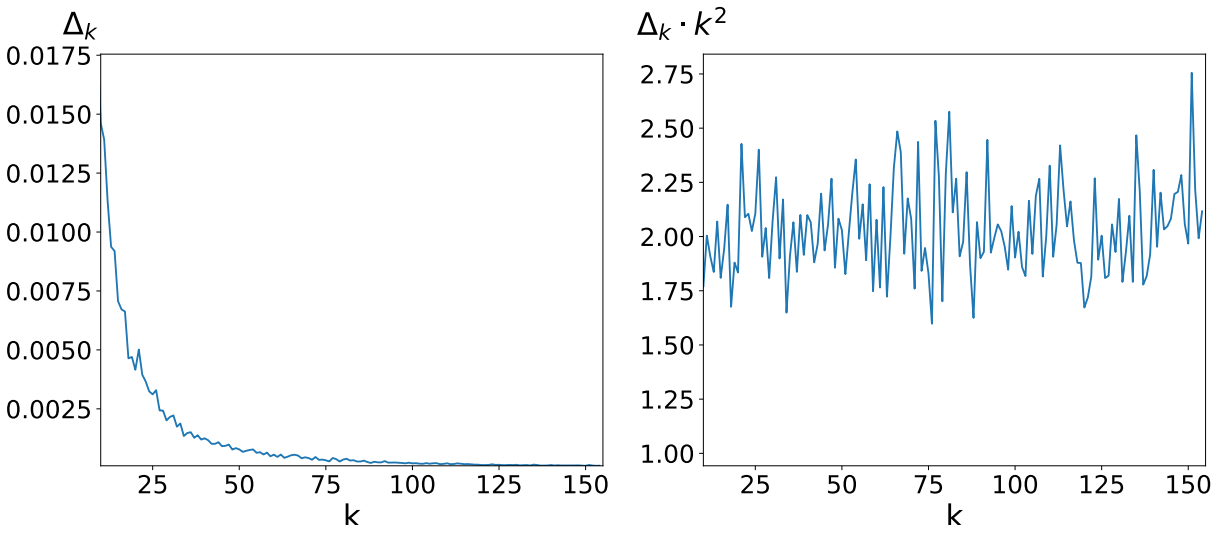

Figure 8: Δ_k (left) and $\Delta_k \cdot k^2$ (right)

where $\{\lambda_k^{(i)}\}_{i=1}^d$ and $\{\lambda_{k+1}^{(i)}\}_{i=1}^d$ are the top D Hessian eigenvalues at sample sizes k and $k+1$, and σ is the perturbation variance.

We compute both empirical and theoretical estimates of Δ_k. First, we train a model instance as described above. For each batch k, we apply the power-iteration method to the model to extract the top D Hessian eigenvalues $\{\lambda_k^{(d)}\}_{d=1}^D$. The theoretical estimate of Δ_k is then obtained by substituting these eigenvalues into our approximation. To reduce high-frequency fluctuations, we apply a five-point moving average to the sequence: $\frac{1}{5}\big(\Delta_{k-2} + \Delta_{k-1} + \Delta_k + \Delta_{k+1} + \Delta_{k+2}\big)$ for each k, where, at the boundaries, any index outside $[1, N]$ is replaced by the nearest valid endpoint (i.e. we replicate the border value), which is equivalent to convolution with constant-value padding.

In parallel, we compute empirical estimates of Δ_k via Monte Carlo sampling of Gaussian perturbations along the same top-D eigenvectors, following the procedure described above. Finally, we plot the theoretical and empirical Δ_k on the same axes, allowing us to assess how closely the eigenvalue-based bound tracks the observed convergence of the loss landscape.

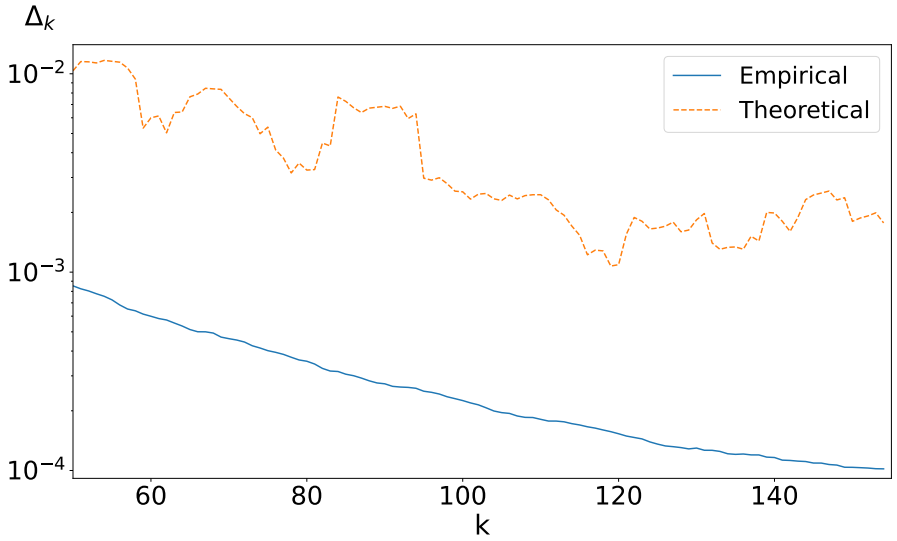

Figure 9: Empirical and Theoretical Δ_k

Figure 9 shows the theoretical estimate of Δ_k (dashed line) alongside the empirical Monte Carlo measurements (solid line). Across all values of k, the theory curve remains above the empirical one, demonstrating that our eigenvalue-based formula provides a valid upper bound for the true loss-landscape changes.

Now, to eliminate the explicit batch index k, we plot Δ_k directly against $\mathcal{L}_k$ and apply the same moving average for a smoother, more stable curve.

Figure 10 shows the relationship between the loss $\mathcal{L}_k(\mathbf{w}^*)$ and the corresponding change $\Delta_k(\mathbf{w}^*)$. In the top panel, each blue dot represents the empirical estimate of Δ_k versus $\mathcal{L}_k$, and the red line is the best-fit linear trend. The bottom panel shows the analogous theoretical estimates, again smoothed and fitted (green line). In both cases, there is a strong positive correlation: higher loss values correspond to larger observed and predicted changes. Although some scatter remains due to sampling noise, the overall linear relationship validates our hypothesis that the eigenvalue-based bound Δ captures the dominant curvature effects of the Hessian. Consequently, the Δ-sufficient condition can effectively limit changes in the loss function.

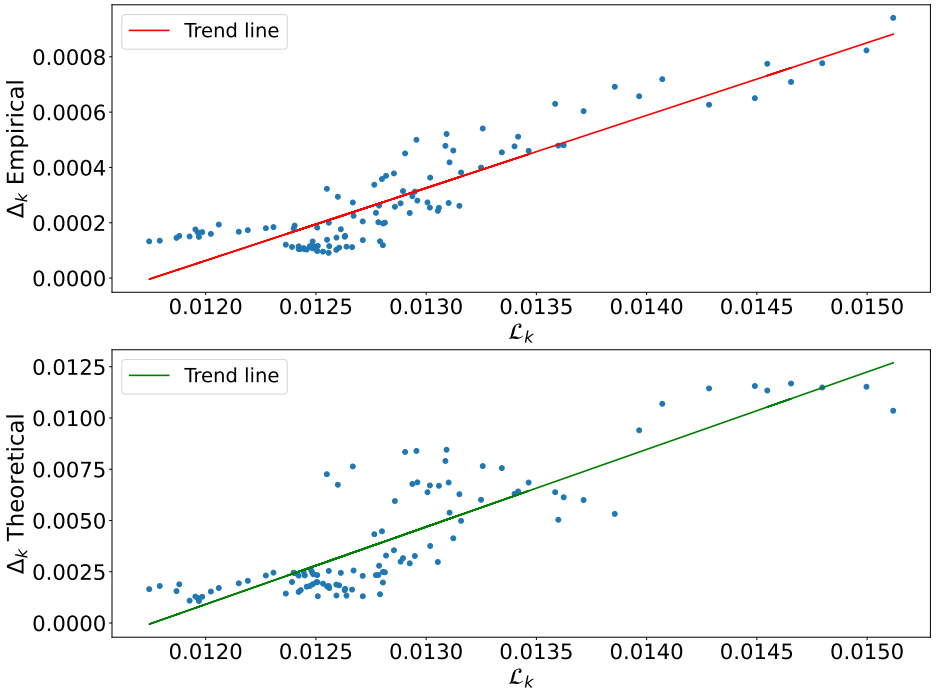

Figure 10: Empirical and Theoretical Δ_k

We are now ready to propose an algorithm to determine a Δ-sufficient dataset size. Both theoretically and empirically, once Δ_k falls below a prescribed tolerance Δ, the loss $\mathcal{L}_k$ changes negligibly. Thus we say that for a tolerance $\Delta > 0$. The Δ-*sufficient sample size* is

$$K = \inf\{k \mid \forall\, k' \geq k :\ \Delta_{k'} < \Delta\}.$$

When $k \geq K$, adding further samples changes the loss landscape by less than Δ. Smaller values of Δ correspond to higher accuracy. In our MNIST experiments, we found that $\Delta = 0.025$ yields $k = 100$ and $\mathcal{L}_k \approx 0.013$.

Table 1 summarizes the key results of our Δ-sufficient criterion applied to MNIST and Fashion-MNIST. For each dataset and model architecture, we report the tolerance Δ, the number of batches k required to reach Δ-sufficiency, the resulting loss $\mathcal{L}_k$, and the total runtime.

Table 1: Results of Δ-sufficient experiments on MNIST and Fashion-MNIST

Dataset	Model	Δ	k	$\mathcal{L}_k$	Time (s)
MNIST	Single-layer MLP	0.025	100	0.013	2000
	Multi-layer MLP	0.025	40	0.010	3500
	Convolutional CNN	0.025	60	0.024	1800
Fashion-MNIST	Single-layer MLP	0.030	120	0.020	2100
	Multi-layer MLP	0.030	90	0.017	4400
	Convolutional CNN	0.030	70	0.015	2400

Since our theoretical estimate Δ_k consistently bounds the empirical Δ_k and is much cheaper to compute, we employ the following algorithm for stopping criterion:

Algorithm 1 Determine Δ-sufficient dataset size

Require: target tolerance $\Delta > 0$, batch size B

$\mathcal{D} \leftarrow \emptyset,\ k \leftarrow 0$

repeat

  $k \leftarrow k + 1$

  Sample a new batch of B examples and append to $\mathcal{D}$

  Train or update the model on $\mathcal{D}$

  Compute the top-D Hessian eigenvalues and form Δ_k estimation via the eigenvalue-based approximation:

$$\Delta_k \approx \frac{\sigma^4}{4}\left(2\sum_{d=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})^2 + \Big(\sum_{i=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})\Big)^2\right).$$

until $\Delta_k > \Delta$

return $|\mathcal{D}|$

Moreover, in addition to determining the dataset size K, the procedure also yields a fully trained model as a by-product of the iterative updates.

This procedure incrementally grows the dataset until the eigenvalue-based estimate Δ_k falls below the prescribed tolerance Δ, at which point further additions have a negligible effect on the loss. The dominant cost per iteration is the extraction of the top-D Hessian eigenvalues via power iteration, which requires $\mathcal{O}(I\,D\,C_{\mathrm{Hv}})$ work where I is the number of power-iteration steps and C_{Hv} is the cost of a single Hessian–vector product (comparable to one gradient evaluation). Since $D \ll M$ and I is typically small (e.g. $I \approx 20$), the extra overhead remains linear in the number of mini-batches and is modest compared to full second-order methods. Thus, our algorithm offers an efficient, theoretically grounded approach to determining a Δ-sufficient dataset size with minimal computational burden.

6 Discussion

The gap between the theoretical and empirical curves arises because the estimate uses only the top D eigenvalues. Therefore, focusing on the leading modes yields a reliable and efficient criterion for determining when adding more data has negligible effect on the local loss geometry.

Our Hessian-projection framework provides a clear picture of how a neural network's loss landscape evolves with training–set size. By restricting attention to the D principal curvature directions (the top Hessian eigenvectors), we obtain far more structured 3D loss surfaces and a sharply decaying landscape-difference measure

$$\Delta_k = \mathbb{E}\big[\mathcal{L}_{k+1}(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta})\big]^2,$$

which empirically follows an approximate $\mathcal{O}(1/k^2)$ trend. Moreover, the Hessian-based theoretical bound consistently upper-bounds the Monte Carlo estimates (see Figure 9). This confirms that only a small number of eigen-modes drive the bulk of loss-landscape change.

A practical outcome is a stopping criterion and algorithm 1 for data collection (see Figure 10): once Δ_k falls below a fixed tolerance Δ, further additions have a negligible effect on the loss. This lets practitioners avoid unnecessary data collection and computation without sacrificing model stability.

Limitations and Future Work Our study used $D = 10$ principal directions and relatively small MLPs on MNIST-like tasks. Deeper or convolutional architectures, alternative loss functions, or non–i.i.d. data may shift the spectrum decay and require reconsidering D. Extending this Hessian-based convergence criterion to large models or streaming data remains an exciting direction.

7 Conclusion

We have introduced and validated a Hessian-projection approach to identify when a neural network's loss landscape "settles" as training data grows. By projecting onto the top eigenvectors of the Hessian, we both visualize and quantify the landscape's sensitivity via the Δ_k function, and derive a tight eigenvalue-difference bound that serves as a reliable upper bound on loss-landscape changes. Our results demonstrate a clear "sufficient sample size" threshold: beyond this, further data become insignificant in local curvature. This criterion provides a concrete guide for efficient dataset sizing and early stopping in model training.

References

[1] Derya Soydaner. A comparison of optimization algorithms for deep learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 34(13):2052013, April 2020. ISSN 1793-6381. doi: 10.1142/s0218001420520138. URL `http://dx.doi.org/10.1142/S0218001420520138`.

[2] Lei Wu, Zhanxing Zhu, and Weinan E. Towards understanding generalization of deep learning: Perspective of loss landscapes. *preprint arXiv:1706.10239*, 2017. Characteristics of the loss landscape explain the good generalization capability.

[3] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook. `https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`, 2012. A collection of facts about matrices and matters relating to them.

[4] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6), 2012. URL `https://api.semanticscholar.org/CorpusID:5280072`. The database of handwritten digits.

[5] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *preprint arXiv:1708.07747*, 2017. A dataset of Zalando's article images.

[6] Nikita Kiselev and Andrey Grabovoy. Unraveling the hessian: A key to smooth convergence in loss function landscapes. *arXiv preprint arXiv:2409.11995*, 2024. Upper bounds via Hessian for fully connected neural networks.

[7] Levent Sagun, Utku Evci, Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *preprint arXiv:1706.04454*, 2018. In over-trained neural networks the Hessian rank is often low; the active subspace may be significantly smaller.

[8] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, 2019. URL `https://proceedings.mlr.press/v97/ghorbani19b/ghorbani19b.pdf`. Observed large isolated eigenvalues in the spectrum and a surprising concentration of the gradient in the corresponding eigenspaces.

[9] Vladislav Meshkov, Nikita Kiselev, and Andrey Grabovoy. Convnets landscape convergence: Hessian-based analysis of matricized networks, 2024. URL `https://ieeexplore.ieee.org/document/10899113`. Upper bounds via Hessian for convolutional neural networks.

[10] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *preprint arXiv:2203.15556*, 2022. Motivations behind overloaded models and the importance of optimal training size estimation.

[11] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *preprint arXiv:1712.09913*, 2018. Methods of visualizing loss function curvature and comparing different functions.

[12] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002. Submitted July 2001; Published March 2002.

[13] André Elisseeff, Theodoros Evgeniou, and Massimiliano Pontil. Stability of randomized learning algorithms. *Journal of Machine Learning Research*, 6:55–79, 2005. Submitted February 2004; Revised August 2004; Published January 2005.

[14] Noah Golmant, Zhewei Yao, Amir Gholami, Michael Mahoney, and Joseph Gonzalez. pytorch-hessian-eigenthings: efficient pytorch hessian eigendecomposition, October 2018. URL `https://github.com/noahgolmant/pytorch-hessian-eigenthings`.

[15] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, volume 32 of *NeurIPS 2018*, Montréal, Canada, 2018. URL `https://arxiv.org/abs/1802.08241`. arXiv:1802.08241v4 [cs.CV], 2 Dec 2018.

[16] Pegah Golestaneh, Mahsa Taheri, and Johannes Lederer. How many samples are needed to train a deep neural network? *arXiv preprint arXiv:2405.16696v1*, May 2024. URL `https://arxiv.org/abs/2405.16696`. [math.ST].

[17] Andrew Gordon Wilson. Deep learning is not so mysterious or different, 2025. URL `https://arxiv.org/abs/2503.02113`.

A Appendix

A.1 Proof of Theorem 1

Proof. Starting from the second-order Taylor expansion around the local minimizer $\mathbf{w}^*$, we have

$$\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w}) \approx \big[\mathcal{L}_{k+1}(\mathbf{w}^*) - \mathcal{L}_k(\mathbf{w}^*)\big] + \tfrac{1}{2}\,\boldsymbol{\theta}^{\mathsf{T}}\big(\boldsymbol{\Lambda}_{k+1} - \boldsymbol{\Lambda}_k\big)\,\boldsymbol{\theta},$$

where $\boldsymbol{\theta} = \mathbf{P}^{\mathsf{T}}(\mathbf{w} - \mathbf{w}^*)$ and $\boldsymbol{\Lambda}_j = \mathbf{P}^{\mathsf{T}}\mathbf{H}_j(\mathbf{w}^*)\mathbf{P}$.

Let $p(\boldsymbol{\theta})$ be a distribution with mean $\mathbf{m} = \mathbb{E}[\boldsymbol{\theta}] = 0$ and covariance $\boldsymbol{\Sigma} = \mathbb{D}[\boldsymbol{\theta}]$. Denote

$$\mathbf{A} = \boldsymbol{\Lambda}_{k+1} - \boldsymbol{\Lambda}_k.$$

Then, by standard matrix-expectation results (see [3, p. 35]),

$$\mathbb{E}\big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big] = \mathbb{E}\big[\tfrac{1}{2}\,\boldsymbol{\theta}^{\mathsf{T}}\mathbf{A}\,\boldsymbol{\theta}\big] = \tfrac{1}{2}\,\big(\mathrm{Tr}(\mathbf{A}\boldsymbol{\Sigma}) + \mathbf{m}^{\mathsf{T}}\mathbf{A}\mathbf{m}\big) = \tfrac{1}{2}\,\mathrm{Tr}(\mathbf{A}\boldsymbol{\Sigma}).$$

Similarly, using [3, p. 43],

$$\mathbb{D}\big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\big] = \tfrac{1}{4}\Big(2\,\mathrm{Tr}(\mathbf{A}^2\boldsymbol{\Sigma}^2) + 4\,\mathbf{m}^{\mathsf{T}}\mathbf{A}^2\mathbf{m}\,\sigma^2\Big) = \tfrac{1}{2}\,\sigma^4\,\mathrm{Tr}(\mathbf{A}^2),$$

where we have set $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$ and used $\mathbf{m} = 0$.

Since

$$\Delta_k = \mathbb{D}\big[\mathcal{L}_{k+1} - \mathcal{L}_k\big] + \big(\mathbb{E}[\mathcal{L}_{k+1} - \mathcal{L}_k]\big)^2,$$

we get

$$\Delta_k \approx \tfrac{1}{2}\,\sigma^4\,\mathrm{Tr}(\mathbf{A}^2) + \tfrac{1}{4}\,\big(\mathrm{Tr}(\mathbf{A}\boldsymbol{\Sigma})\big)^2 = \tfrac{\sigma^4}{4}\Big(2\,\mathrm{Tr}(\mathbf{A}^2) + \big(\mathrm{Tr}(\mathbf{A})\big)^2\Big).$$

Finally, since

$$\mathrm{Tr}(\mathbf{A}) = \sum_{d=1}^{D}(\lambda_{k+1}^d - \lambda_k^d), \quad \mathrm{Tr}(\mathbf{A}^2) = \sum_{d=1}^{D}(\lambda_{k+1}^d - \lambda_k^d)^2,$$

we arrive at the stated approximation

$$\Delta_k \approx \frac{\sigma^4}{4}\left(2\sum_{d=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})^2 + \Big(\sum_{d=1}^{D}(\lambda_{k+1}^{(d)} - \lambda_k^{(d)})\Big)^2\right).$$

□