# Neural Networks Loss Landscape Convergence in Different Low-Dimensional Spaces

**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 the loss landscape of neural networks evolves as the training set size increases is crucial for optimizing performance. It is well known that larger datasets can alter the shape of this high-dimensional landscape. But the exact point at which additional data no longer brings substantial changes remains underexplored.

In this paper, we examine neural network models and show that their loss landscapes begin to stabilize once the training set grows beyond a certain threshold, revealing a connection between dataset size and the geometry of the loss surface. To understand this phenomenon, we propose a method that projects the full parameter space onto a low-dimensional subspace derived from top eigenvectors of the Hessian. That provides a more interpretable view of how the loss surface in the vicinity of local minima behaves as more data become available. We use different sampling strategies, applying Monte-Carlo estimation to capture the structure of this reduced loss landscape more precisely.

We validate our insights through experiments on image classification tasks, demonstrating that this low-dimensional analysis can reveal when the landscape effectively settles, and thus helps determine a minimum viable dataset size. Our findings shed light on the relationship between dataset scale and optimization geometry, and suggest practical strategies for balancing computational costs with the benefits of additional training data.

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

## 1 Introduction

Neural networks have become irreplaceable in various aspects of life and have a lot of applications now (e.g. image classification, language models, recommender systems, etc.). However, as datasets and models grow larger and larger, we get higher accuracy and breakthrough results. However some problems connected with computation resources have appeared. Sometimes models have so many parameters so we need time we cannot provide for their training. Some attempts to explore and compare optimization methods have already been made [1]. The core problem of all these solutions hides in their locality, as the size of the neural network and training dataset remains unchanged.

Preprint. Under review.

In this paper we explore when adding more data no longer brings significant improvements in network performance and to explain obtained results in terms of the curvature and shape of the neural network's loss landscape.

Our research on the loss landscape of neural networks. Specifically, how it changes while adding new samples to the training dataset. There remains a problem: processing of large datasets requires a lot of time. We need to determine when increasing dataset size stops reshaping this loss landscape for general neural networks. But the question of a minimum viable dataset size — a threshold beyond which further data bring insignificant changes — remains underexplored. Moreover, connecting such a threshold to generalization capabilities [2] will reduce computation requirements as we show that there is an upper bound of "active" dataset samples.

We suggest using the Hessian of the loss function to calculate its close approximation. However, the computation of all Hessian requires a bunch of time, so we consider a projection of neural network parameters space to a certain subspace. We place the following tasks to the front:

1. Constructing a Hessian-based projection of the loss landscape which finds main curvature directions (top eigenvectors) in parameter space.
2. Using different sampling methods, identifying at which dataset size the shape of the loss function stabilizes.
3. We apply visualization methods to characterize the loss landscape next to the local minima.
4. Providing applicable theoretical criteria to determine when adding more samples has minimal effect. We refer to The Matrix Cookbook [3] as a hint for matrix operations.

The proposed solution is novel as it links Hessian estimation to the idea of a minimal dataset threshold. This enables advantages such as more cost-effective dataset collection and a clearer understanding of how dataset size interacts with the geometry of the loss function. And applying recent methods for low-rank Hessian estimation can speed up and solve the problem of high computation cost of top eigenvalues for very large networks.

We validate our analysis results on well-known image classification tasks like MNIST [4] and Fashion-MNIST [5]. By connecting practical results to theoretical bounds, we offer a method for identifying a computation-efficiency method that balances performance with accuracy gains.

**Contributions.** Our contributions can be summarized as follows:

- We present a Hessian-based approach that uses a projection into a low-dimensional subspace of top eigenvectors to find the critical sufficient dataset size.
- We demonstrate the validity of our theoretical framework through empirical studies on MNIST and Fashion-MNIST, incrementally increasing dataset size until additional data makes an impact to the loss function curvature.
- We highlight the implications of our findings for practical data collection strategies, showing how the detection of a sufficient dataset size can reduce computation time.

**Outline.** The rest of the paper is organized as follows. Section 2 divides existing results into some topics, highlighting their key contributions and findings. Section 3 considers general notation and some preliminary calculations. In Section 4, we provide theoretical bounds on the hessian and losses difference norms. Empirical study of the obtained results is given in Section 5. We summarize and present the results in Sections 6 and 7. Additional experiments and proofs of theorems are included in the Appendix A.

## 2 Related Work

**Hessian-based analysis.** The landscape of loss functions has been explored from various perspectives in the literature. Methods connected with Hessian are central to understanding the convergence process and loss function landscape [6]. Recent works observed that the Hessian of some models tends to have a low effective rank, with only a small quantity of eigenvalue differences from zero. Empirical explorations from [7] show that the Hessian rank is often very low in the vicinity of local minima, highlighting a significantly smaller "active subspace". Prior work investigating how large eigenvalues emerge during training [8].

Some results have already been reached for fully connected and convolutional architectures [9], but they need generalization.

**Loss Landscape Geometry.** Other recent researches aimed at how dataset size affects both performance and the geometry of loss function landscapes. In the assumption of enough training samples, neural networks often reach smoother and flatter minima with better generalization [2]. However, collecting large datasets is expensive, as well as the computation requirements for them, raising questions about the most effective ratio between dataset and model sizes [10]. As understanding the landscape of loss functions is key to the solution, prior work has looked at visualizing loss surfaces [11].

## 3 Preliminaries

### 3.1 General notation

In this section, we introduce the general notation used in the rest of the paper and the basic assumptions.

We consider a $K$-label classification problem. So, let's consider $p(y|\mathbf{x})$ a conditional probability, which maps unobserved variable $\mathbf{x} \in \mathbf{X}$ to the corresponding output $y \in \mathbf{y}$. We consider $\mathbf{y}$ is a subspace (or same space) as $\mathbb{R}^K$. Let $f_{\mathbf{w}}(\cdot)$ be a neural network with a list of parameters $\mathbf{w}$. Let $\mathbf{\Omega}$ be a space of parameters ($\mathbf{w} \in \mathbf{\Omega}$).

Let

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \,|\, i = 1, \ldots, m\}$$

be a given dataset of size $m$ consists of i.i.d. samples. Let $\ell(\cdot, \cdot)$ be a given loss function (e.g. cross-entrophy) where first argument refers to neural network's result and the second argument refers to the true answer. To simplify, define:

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

**Definition 1.** *The empirical loss function for the first $k$ elements is:*

$$\mathcal{L}_k(\mathbf{w}) = \frac{1}{M} \sum_1^k \ell_i(\mathbf{w}), \;\; \mathcal{L}(\mathbf{w}) := \mathcal{L}_m(\mathbf{w}).$$

Thus, the difference between losses for neighbouring samplem sizes is:

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

**Definition 2.** *Let the Hessian of $\mathcal{L}_k(\mathbf{w})$ be:*

$$\mathbf{H}_k(\mathbf{w}) = \nabla^2_{\mathbf{w}} \mathcal{L}_k(\mathbf{w}) = \frac{1}{k} \sum_1^k \nabla^2_{\mathbf{w}} \ell_i(\mathbf{w}).$$

**Definition 3.** *To calculate the overall loss landscape changing, one has to integrate the absolute difference for the entire parameter space. We define **$\mathbf{\Delta}$-function** (delta-function) as:*

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

*where $p(\mathbf{w})$ describes the priority of the particular parameter points so we can make $p(\mathbf{w})$ have higher values next to the local minima.*

We firther investigate this difference and aimed at exploration of how adding a new object to the dataset changes the value. We interested in convergence of this value and properties of loss function when the training dataset size limits to $\infty$.

**Definition 4.** *Let $\Delta$ be a positive hyperparameter that indicates the stop-difference for $\Delta_k$. If*

$$k^* = \inf_k \{\forall m \geq k : \Delta_m < \Delta\}$$

*we can say that $k^*$ samples in the dataset are enough to describe the distribution of data from the general population. We call $k^*$ as **sufficient**.*

### 3.2 Assumptions

**Assumption 1.** *Let* $\mathbf{w}^*$ *be the local minimum of both* $\mathcal{L}_{k-1}(\mathbf{w})$ *and* $\mathcal{L}_k(\mathbf{w})$*. Thus,*

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

This assumption allows us to explore the behavior and the geometry of the loss function landscape at only one point.

Furthermore, using second-order Taylor's approximation for $\mathcal{L}_k(\mathbf{w})$ at $\mathbf{w}^*$ we get:

$$\mathcal{L}_k(\mathbf{w}) \approx \mathcal{L}_k(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^\intercal \mathbf{H}_k(\mathbf{w}^*)(\mathbf{w} - \mathbf{w}^*)$$

**Assumption 2.** *We can assume that* $\mathbf{w}^*$ *is a local minimizer, so that* $\mathcal{L}_{k+1}(\mathbf{w}^*) = \mathcal{L}_k(\mathbf{w}^*)$*.*

**Assumption 3.** *We can assume parameters* $\mathbf{w}$ *to be random, which will lead to quite natural condition:* $p(\mathbf{w})$ *can be even a prior distribution of* $\mathbf{w}$*, so:*

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

$$\mathbb{D}_{p(\mathbf{w})}\left(\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\right) + \left(\mathbb{E}_{p(\mathbf{w})}\left(\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\right)\right)^2$$

## 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 overcome this issue, we reduce the complexity by projecting the parameters onto a lower-dimensional subspace. This subspace is spanned by the $d$ dominant eigenvectors of the Hessian evaluated at the local minimum $\mathbf{w}^*$, i.e., $\mathbf{H}_k(\mathbf{w}^*)$.

The core idea behind the projection is that, in neural network models, the Hessian has a low effective rank according to [7]: only a few eigenvalues are significantly larger than zero, while the majority are negligibly small. As a result, the directions corresponding to these small eigenvalues contribute little to the variation of the loss function. By keeping only the eigen-directions associated with the largest eigenvalues, we obtain a compact but informative representation of the loss landscape that captures the most influential directions of curvature.

For further intuition, we present an experimental result on the plot 1 showing the dependence of the eigenvalues on their index (ordered in descending order). The dominant eigenvalues were computed iteratively using a method described in [12].

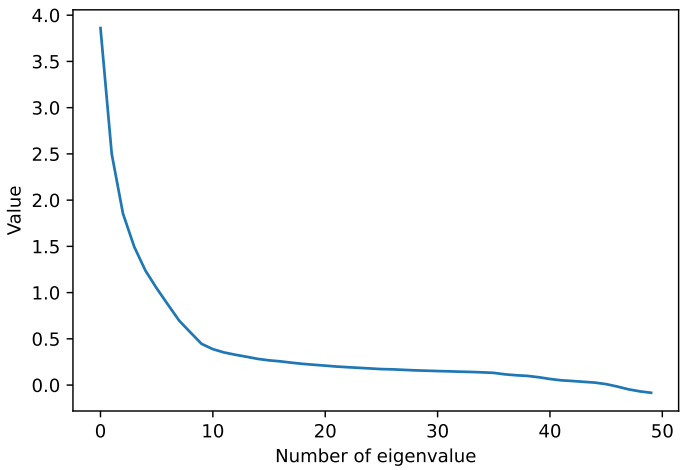


Figure 1: Dependence of eigenvalues on their index in descending order.

Concretely, let

$$\mathbf{V} = \{\mathbf{e}_1, \ldots, \mathbf{e}_d\}, \qquad \mathbf{P} = [\mathbf{e}_1, \ldots, \mathbf{e}_d],$$

be the corresponding eigenvectors (forming an eigenbasis) with the largest eigenvalues. Then, we represent the network parameters as

$$\mathbf{w} = \mathbf{w}^* + \mathbf{P}\,\boldsymbol{\theta},$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the coordinates in this reduced subspace. This representation allows us to focus on the most significant directions, leading to a more tractable approximation of the loss landscape.

Substituting $\mathbf{w} = \mathbf{w}^* + \mathbf{P}\,\boldsymbol{\theta}$ into the second-order Taylor approximation of the loss function, we obtain

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

Define

$$\boldsymbol{\Lambda}_k(\mathbf{w}) := \mathbf{P}^{\mathsf{T}}\,\mathbf{H}_k(\mathbf{w})\,\mathbf{P}, \quad \text{so that} \quad \boldsymbol{\Lambda}_k(\mathbf{w}^*) = \operatorname{diag}\big(\lambda_k^1, \ldots, \lambda_k^d\big),$$

where $\lambda_k^1, \ldots, \lambda_k^d$ are the top $d$ eigenvalues of $\mathbf{H}_k(\mathbf{w}^*)$. Thus, the loss is approximated as

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

Similarly, for the loss computed on $k+1$ samples, we have

$$\mathcal{L}_{k+1}\big(\mathbf{w}^* + \mathbf{P}\,\boldsymbol{\theta}\big) \approx \mathcal{L}_{k+1}\big(\mathbf{w}^*\big) + \frac{1}{2}\,\boldsymbol{\theta}^{\mathsf{T}}\,\boldsymbol{\Lambda}_{k+1}\boldsymbol{\theta}.$$

**Theorem 1.** *(Of Delta-function) Let $p(\boldsymbol{\theta})$ be a probability distribution over the reduced subspace (for instance, a Gaussian distribution), with mean $\mathbf{m} = \mathbf{0}$ and covariance $\boldsymbol{\Sigma} = \mathbb{D}_{p(\boldsymbol{\theta})}\big(\boldsymbol{\theta}\big)$. Then,*

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

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

The result shown on this theorem and applied to our $p(\boldsymbol{\theta})$ can estimate the $\Delta_k$ function so we can seay when the dataset size will be sufficient.

## 5 Experiments

To verify the theoretical estimates, in this section, we present experiments aimed at analyzing how the loss landscape evolves as the size of the training set increases.

**Data.** We use the MNIST [4] and Fashion-MNIST [5] dataset, which consists of $60{,}000$ training images and $10{,}000$ test images of handwritten digits (0–9). Each image is $28 \times 28$ pixels, grayscale.

**Model Architecture.** We use the following neural network model structures and fixations:

- A multilayer perceptron (MLP) with different number of hidden layers (ReLU activations).
- The choice of architecture remains consistent across all experiments.

The first two are preliminary experiments, conducted as illustrative examples on the MNIST dataset, while the others focusing on Hessian-based projections and show connections between eigenvalues and loss function convergence. That is our primary aim — to check our theoretical estimates in-deal. All experiments are avaliable at our github repository.

### 5.1 Visualizing the Loss Landscape in a Random Two-Dimensional Subspace

The idea of this initial experiment is to get an intuitive sense of how the loss function's landscape shifts with different training-set sizes. To do this, we pick two random directions $\mathbf{v}_1$ and $\mathbf{v}_2$ in the parameter space and project the network's parameters onto the subspace they span.

In order to visualize how the loss function behaves as the model parameters change along two chosen random directions, we compute loss values over a two-dimensional grid in the projected parameter space. Recall that, in our setup, the loss function is defined as

$$L_k(w) = \frac{1}{k} \sum_{i=1}^{k} \ell_i(w),$$

where $\ell_i(w)$ is the loss contribution of the $i$th sample.

The procedure for computing the loss values and visualizing them in 3D is as follows:

1. **Fix a base model and directions.** Begin with a fixed trained on the same dataset neural network model. Also, obtain two parameter directions $\boldsymbol{\theta} = \{\mathbf{v}_1, \mathbf{v}_2\}$ that will be used to explore the parameter space. These directions are vectors in the model's parameter space.
2. **Grid Construction.** We generate a two-dimensional grid of coefficient pairs $(\alpha, \beta)$ using a fixed step size within the interval $[-1, 1]$. These coefficients will scale two pre-computed directions in the parameter space. The grid is formed by taking the Cartesian product.
3. **Loss Calculation on the Grid.** For each grid point $(\alpha, \beta)$, a new set of parameters is created by forming a linear combination of the two directions:

   $$\boldsymbol{\theta} = \alpha \cdot \mathbf{v}_1 + \beta \cdot \mathbf{v}_2.$$

   A temporary model is then initialized with these shifted parameters, and combined with the base model via straight sum:

   $$\mathbf{w} = \mathbf{w}^* + \boldsymbol{\theta}$$

   . The base model's parameters are kept fixed (close to $\mathbf{w}^*$) while applying these perturbations. Using a loss function evaluation, the losses are computed for each grid point.
4. **Computation of Aggregated Loss Measures.** The loss values for each grid point are aggregated in two ways:
   - The mean loss $\mathcal{L}_k$ computed over a specified number of samples is used to construct the loss landscape.
   - Similarly, the mean loss $\mathcal{L}_{k+1}$ for a larger sample size is computed, and the squared difference

     $$(\mathcal{L}_{\text{k+1}} - \mathcal{L}_{\text{k}})^2$$

     is evaluated at each grid point.
5. **3D Surface Visualization.** Two 3D surface plots are created and placed side by side:
   - *Left Plot:* Displays the loss landscape $\mathcal{L}_k(\boldsymbol{\theta})$, where the $x$ and $y$ axes correspond to the grid coordinates (the scaling coefficients along the two directions) and the $z$-axis shows the average loss value.
   - *Right Plot:* Shows the squared loss difference, $(\mathcal{L}_{k+1}(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta}))^2$. This plot emphasizes regions where an additional data sample induces substantial changes in the loss.

Figures 2 and 3 shows a set of four 3D plots that illustrate the loss landscape and the corresponding loss difference for various sample sizes. In these figures, the top two plots correspond to very small training sets, while the bottom two plots display the loss landscape and its squared difference for the maximum available dataset size.

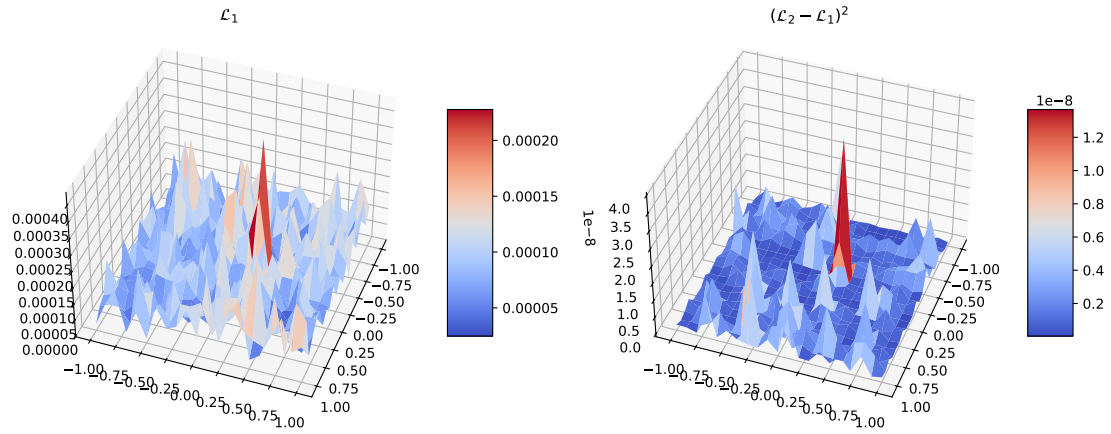


Figure 2: Loss function landscape for small $k$

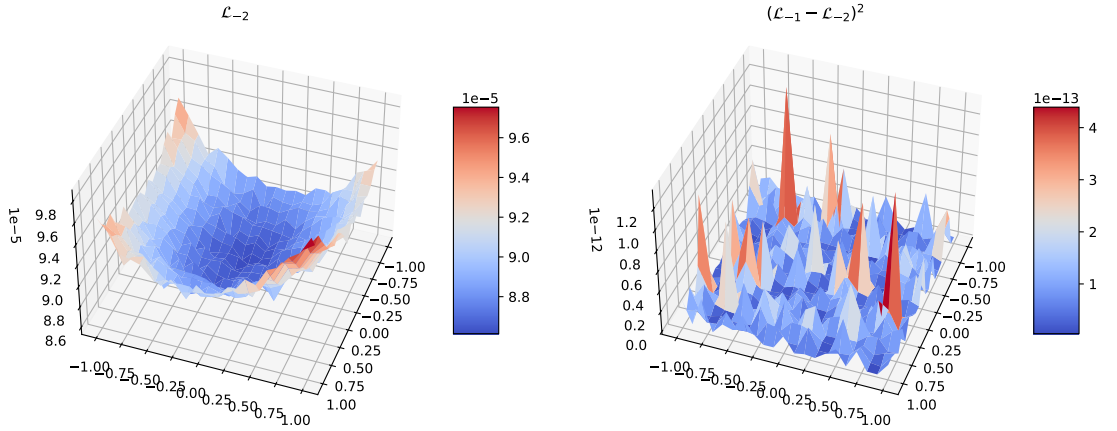


Figure 3: Loss function landscape for maximum $k$

**Loss Landscape for Small Sample Sizes** In the upper plots 2 ($k = 1$), the loss landscape exhibits sharp variations and high-contrast peaks. These characteristics indicate that when training with an extremely limited number of samples, the model is highly sensitive to small perturbations in the parameter space. In such cases, even a slight change in parameters leads to significant fluctuations in the loss function. This is further supported by the upper plot of the squared loss difference,

$$\left(\mathcal{L}_k(\boldsymbol{\theta}) - \mathcal{L}_k(\boldsymbol{\theta})\right)^2,$$

which shows large local differences that reflect high instability in the optimization landscape.

**Loss Landscape for Maximum Dataset Size** In contrast, the lower plots 3 correspond to training with the maximum available dataset size. Here, the loss surface appears much smoother and more regular. The sharp peaks have diminished considerably, and the surface displays a gradual slope. This smoothing of the loss landscape indicates that as more data become available, the model's local optimum stabilizes. Likewise, the squared loss difference in the lower plot is much smaller, implying that the addition of extra samples produces only minor adjustments to the loss function. Such convergence suggests that the network has captured the underlying data structure adequately, and that further increases in the training sample size yield diminishing returns in terms of reshaping the loss landscape.

### 5.2 Visualizing delta-function

To further analyze the convergence of the loss landscape, we also evaluate and visualize the $\Delta_k$ function. This function quantifies the change in the loss function when transitioning from a model trained on $k$ samples to one trained on $k + 1$ samples. Directly computing the expectation

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

over the entire distribution of parameter shifts is computationally infeasible. Therefore, we employ a Monte Carlo approximation, where we average the loss differences computed on a set of random Gaussian perturbations. This Monte Carlo approach is straightforward to implement and does not require explicit knowledge of the full distribution of parameter shifts.

The overall procedure is as follows:

1. **Fixing a model and sampling Directions.** Begin with a fixed trained on the same dataset neural network model. Generate $K$ independent random directions from a multivariate normal distribution
   $$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{I}),$$
   where $\mathbf{m} = \mathbf{0}$ and $\sigma$ as variance. These $K$ samples serve as the shifts in the reduced parameter space.
2. **Parameter Snapshot.**
   For each random direction, form a perturbed parameter vector around the base parameter $\mathbf{w}^*$:
   $$\mathbf{w}^{(t)} = \mathbf{w}^* + \boldsymbol{\theta}^{(t)}, \quad t = 1, \ldots, K.$$
   The base model's parameters are kept fixed (close to $\mathbf{w}^*$) while applying these perturbations.
3. **Loss Evaluation.**
   For each perturbed parameter $\mathbf{w}^{(t)}$, compute the loss values
   $$\mathcal{L}_{k+1}(\mathbf{w}^{(t)}) \quad \text{and} \quad \mathcal{L}_k(\mathbf{w}^{(t)}),$$
   and then calculate the squared loss difference
   $$\left(\mathcal{L}_{k+1}(\mathbf{w}^{(t)}) - \mathcal{L}_k(\mathbf{w}^{(t)})\right)^2.$$
4. **Averaging.**
   Average the squared loss differences over the $K$ samples to obtain the Monte Carlo estimate:
   $$\Delta_k \approx \frac{1}{K} \sum_{t=1}^{K} \left(\mathcal{L}_{k+1}(\mathbf{w}^{(t)}) - \mathcal{L}_k(\mathbf{w}^{(t)})\right)^2.$$
   Increasing $K$ results in a more reliable estimate of the expected loss difference.

5. **Visualization.**
The computed $\Delta_k$ values are then visualized as follows:

   - The left plot displays the evolution of $\Delta_k$ across different sample sizes. The $x$-axis represents the sample index $k$, while the $y$-axis shows the corresponding $\Delta_k$.
   - The right plot shows the product $\Delta_k \cdot k^2$ (or another suitable scaling), which highlights the convergence rate of the loss differences as the dataset size increases.

   These 2D plots are generated side-by-side to facilitate a comparative analysis of the convergence behavior.

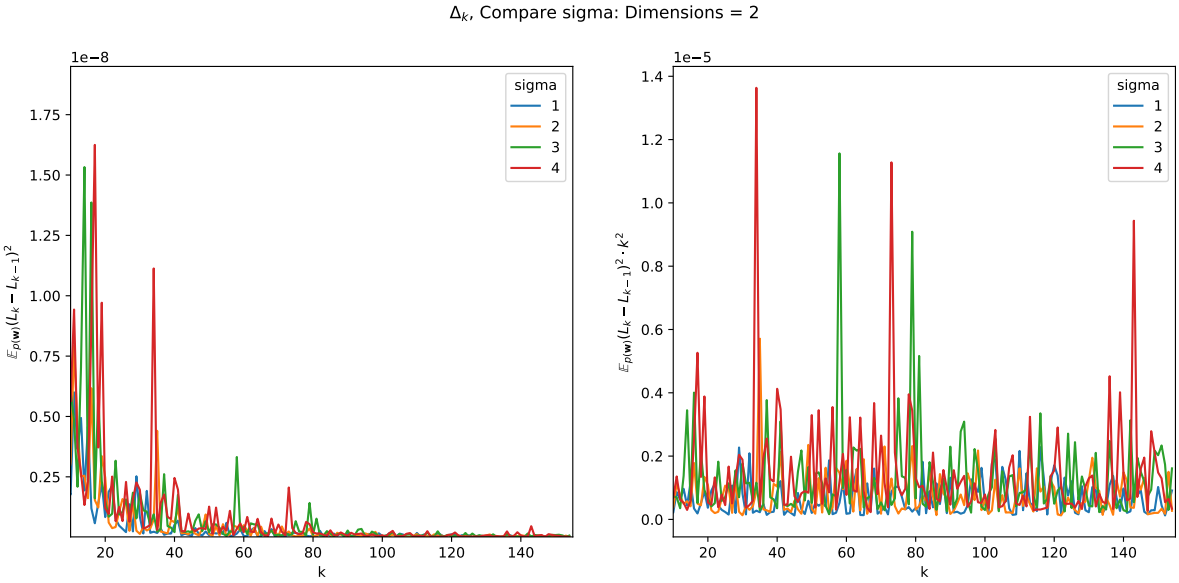


Figure 4: Delta function landscape for different variation

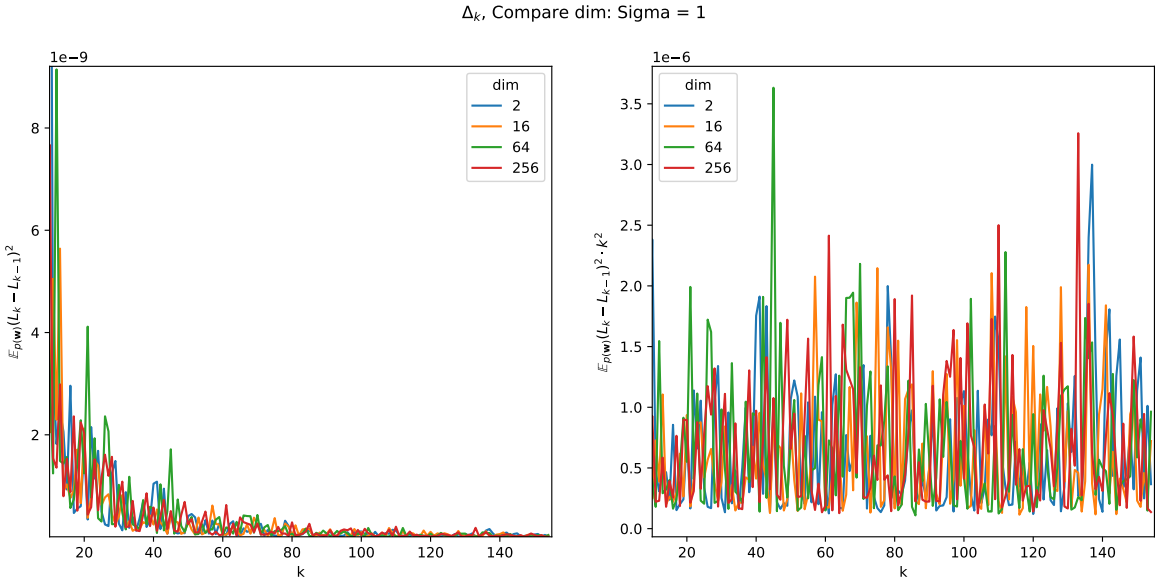


Figure 5: Delta function landscape for different number of dimensions

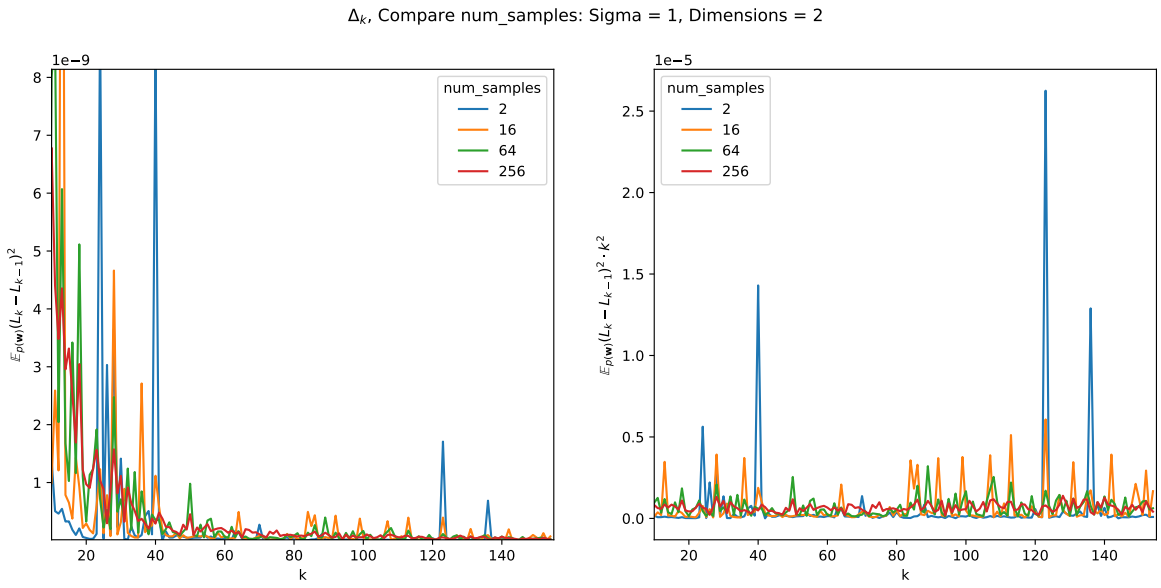


Figure 6: Delta function landscape for different number of samples ($K$)

Figures 4, 5, and 6 present the behavior of the function

$$\Delta_k \;=\; \mathbb{E}_{p(\mathbf{w})}\big[L_{k+1}(\mathbf{w}) - L_k(\mathbf{w})\big]^2$$

under varying parameters of the Monte Carlo sampling and the subspace dimension used for projection. Each figure focuses on a specific comparison: 4 — different values of the Gaussian variance $\sigma$, 4 – different sizes of the projected subspace (i.e., different dimensions), and 4 different numbers of random samples $K$ in the Monte Carlo algorithm.

**Comparing Different Variances $\sigma$** Figure 4 shows how the $\Delta_k$ curves evolve when using various values of $\sigma$ for the random directions. A larger $\sigma$ increases the magnitude and frequency of spikes in the curves. Intuitively, when the Gaussian sampling is more spread out (i.e., we sample directions farther from the base parameter $\mathbf{w}^*$), the resulting perturbations can induce more significant changes in the loss. Thus, $\Delta_k$ can become highly volatile. By contrast, a smaller $\sigma$ keeps the sampled points closer to the minimum, making the changes in loss more localized and stable. In practice, choosing too large a $\sigma$ can obscure the structure around $\mathbf{w}^*$ with excessive noise, while a smaller $\sigma$ focuses on the region near the local minimum, highlighting more relevant variations in the loss landscape.

**Comparing Different Subspace Dimensions** Figure 5 compares the $\Delta_k$ function for different projection dimensions. As the dimension of the subspace grows, the curves tend to exhibit fewer extreme spikes. This suggests that by considering more directions in the parameter space, the Monte Carlo estimation captures a broader view of potential perturbations. Consequently, individual outliers affect the overall measure less dramatically, leading to more stable curves. But a plenty of directions have negligible eigenvalues and thus minimal impact on the loss, so this does not necessarily mean we should always use a high-dimensional subspace. The idea of focusing on the dominant directions (with the largest eigenvalues) helps strike a balance between computational feasibility and capturing the most significant variations in the loss function.

**Comparing Different Numbers of Random Samples $K$** Figure 6 illustrates the effect of varying $K$, the number of random Gaussian samples used for Monte Carlo estimation. With a larger $K$, the curves become less noisy and more stable. This is because the average

$$\frac{1}{K}\sum_{t=1}^{K}\Big(\mathcal{L}_{k+1}(\mathbf{w}^{(t)}) - \mathcal{L}_k(\mathbf{w}^{(t)})\Big)^2$$

is taken over a greater number of directions, thereby reducing variance in the estimate.

In addition, the right-side plots in these figures show the scaled quantity $\Delta_k \cdot k^2$, revealing an approximate $1/k$ decay for the unscaled $\Delta_k$. This suggests that the incremental change in the loss function diminishes on the order of $1/k$ as $k$ increases, confirming that once a sufficiently large dataset is reached, additional samples have only a minor impact on the local loss landscape.

**Conclusion from Preliminary Experiments.** These initial experiments demonstrate that the loss landscape becomes progressively more stable with increasing $N$. However, the directions used here are either random. This motivates a more rigorous approach in the *main experiment* below, where we focus specifically on the most critical directions of curvature — those corresponding to the top eigenvalues of the Hessian.

### 5.3 Main Experiment: Projection onto Dominant Hessian Eigenvectors

From prior work [7], we know that only a few Hessian eigenvectors (those associated with the largest eigenvalues) tend to capture the most significant curvature directions. By projecting onto these principal directions, we can more precisely measure how the landscape the loss surface changes as $k$ increases.

#### Procedure

1. **Hessian approximation.** For each trained model, approximate the Hessian $\mathbf{H}$ or its dominant eigenvalues/eigenvectors. Techniques ???
2. **Eigen-decomposition.** Identify the top $d$ eigenvectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$ with the largest eigenvalues.
3. **Calculation and analization.** For each model, calculate a $\Delta$ fuction via the same way as in the second experiment (using Monte-Carlo) and visualize via plots.

#### Goal and Expected Outcome

- By focusing on the directions of largest curvature, we expect a clearer measure of how the landscape of the loss surface changes with additional data.
- This approach should yield a more accurate estimate of the point at which the landscape effectively stabilizes, thus helping to pinpoint a minimum viable dataset size.
- By applying experiments we check our calculations above on the reality.

## 6 Discussion

TADAAA!

## 7 Conclusion

TADAAA!

## 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] 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.

# A Appendix

## A.1 Proof of the Theorem 1

*Proof.* The difference between the two losses is given by

$$\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w}) \approx \Big[\mathcal{L}_{k+1}(\mathbf{w}^*) - \mathcal{L}_k(\mathbf{w}^*)\Big] + \frac{1}{2}\,\boldsymbol{\theta}^\intercal \Big(\boldsymbol{\Lambda}_{k+1} - \boldsymbol{\Lambda}_k\Big)\boldsymbol{\theta}.$$

Now, we apply a second assumption regarding the $\Delta_k$ function in the context of the projection. Let $p(\boldsymbol{\theta})$ be a probability distribution over the reduced subspace (for instance, a Gaussian distribution), with mean

$$\mathbf{m} = \mathbb{E}_{p(\boldsymbol{\theta})}\big(\boldsymbol{\theta}\big)$$

and covariance

$$\boldsymbol{\Sigma} = \mathbb{D}_{p(\boldsymbol{\theta})}\big(\boldsymbol{\theta}\big).$$

Then, following [3] (see page 35), one can show that

$$\mathbb{E}_{p(\boldsymbol{\theta})}\Big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\Big] \approx \Big[\mathcal{L}_{k+1}(\mathbf{w}^*) - \mathcal{L}_k(\mathbf{w}^*)\Big] + \frac{1}{2}\Big(\mathrm{Tr}(\mathbf{A}\,\boldsymbol{\Sigma}) + \mathbf{m}^\intercal\,\mathbf{A}\,\mathbf{m}\Big),$$

where

$$\mathbf{A} := \boldsymbol{\Lambda}_{k+1}(\mathbf{w}^*) - \boldsymbol{\Lambda}_k(\mathbf{w}^*).$$

Note that $\mathrm{Tr}(\boldsymbol{\Lambda}_k) = \sum_{i=1}^{d} \lambda_k^i$ and, consequently,

$$\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{d} \lambda_{k+1}^i - \sum_{i=1}^{d} \lambda_k^i.$$

Assume further that the distribution in the reduced space is given by

$$\boldsymbol{\theta} \sim \mathcal{N}\big(\mathbf{m}, \sigma^2\mathbf{I}\big),$$

so that $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. Then, according to [3] (page 43), the variance of the loss difference is

$$\mathbb{D}_{p(\boldsymbol{\theta})}\Big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\Big] = \frac{1}{4}\left(2\sigma^4\,\mathrm{Tr}(\mathbf{A}^2) + 4\sigma^2\,\mathbf{m}^\intercal\,\mathbf{A}^2\,\mathbf{m}\right).$$

Finally, the overall measure $\Delta_k$ is defined as

$$\Delta_k \;=\; \mathbb{D}_{p(\boldsymbol{\theta})}\Big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\Big] + \Big(\mathbb{E}_{p(\boldsymbol{\theta})}\Big[\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})\Big]\Big)^2.$$

Substituting the above expressions, we have

$$\Delta_k \;\approx\; \frac{1}{4}\left(2\sigma^4\,\mathrm{Tr}(\mathbf{A}^2) + 4\sigma^2\,\mathbf{m}^\intercal\,\mathbf{A}^2\,\mathbf{m}\right) + \Big(\mathcal{L}_{k+1}(\mathbf{w}^*) - \mathcal{L}_k(\mathbf{w}^*)\Big)^2$$
$$+\; 2\Big(\mathcal{L}_{k+1}(\mathbf{w}^*) - \mathcal{L}_k(\mathbf{w}^*)\Big)\cdot\frac{1}{2}\Big(\mathrm{Tr}(\mathbf{A}\,\boldsymbol{\Sigma}) + \mathbf{m}^\intercal\,\mathbf{A}\,\mathbf{m}\Big) + \;\frac{1}{4}\Big(\mathrm{Tr}(\mathbf{A}\,\boldsymbol{\Sigma}) + \mathbf{m}^\intercal\,\mathbf{A}\,\mathbf{m}\Big)^2.$$

Assuming that $\mathbf{w}^*$ is a local minimizer and remembering that $\mathbf{m} = \mathbf{0}$, the above expression simplifies to

$$\Delta_k \;\approx\; \frac{1}{4}\left(2\sigma^4\,\mathrm{Tr}(\mathbf{A}^2)\right) + \frac{1}{4}\Big(\mathrm{Tr}(\mathbf{A}\,\boldsymbol{\Sigma})\Big)^2.$$

Since $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$, it follows that

$$\mathrm{Tr}(\mathbf{A}\,\boldsymbol{\Sigma}) = \sigma^2\,\mathrm{Tr}(\mathbf{A}),$$

and therefore,

$$\Delta_k \;\approx\; \frac{\sigma^4}{4}\left(2\,\mathrm{Tr}(\mathbf{A}^2) + \Big(\mathrm{Tr}(\mathbf{A})\Big)^2\right).$$

Recalling that

$$\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{d}\left(\lambda_{k+1}^i - \lambda_k^i\right)$$

and

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

we obtain

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

□