Convergence of the loss function surface in transformer neural network architectures

Egor Petrov
MIPT
Moscow, Russia
petrov.egor.d@phystech.edu

Nikita Kiselev
MIPT
Moscow, Russia
kiselev.ns@phystech.edu

Vladislav Meshkov
MIPT
Moscow, Russia
email@phystech.edu

Andrey Grabovoy
MIPT
Moscow, Russia
email@phystech.edu

Abstract

Training a neural network involves searching for the minimum point of the loss function, which defines the surface in the space of model parameters. The properties of this surface are determined by the chosen architecture, the loss function, and the training data. Existing studies show that as the number of objects in the sample increases, the surface of the loss function ceases to change significantly. The paper obtains an estimate for the convergence of the surface of the loss function for the transformer architecture of a neural network with attention layers, as well as conducts computational experiments that confirm the obtained theoretical results. In this paper, we propose a theoretical estimate for the minimum sample size required to train a model with any predetermined acceptable error, providing experiments that prove the theoretical boundaries.

Keywords: Neural networks, Transformer, Loss landscape, Hessian, Dataset size threshold.

1 Introduction

Neural networks are becoming more and more popular these days as a way to solve many practical problems. In particular, the Transformer [1] architecture has revolutionized the world of text processing, as well as many other popular tasks, such as image processing with Visual Transformer [2]. Current results show that as the training sample size increases, there is a trend for the quality of the [3] model to increase, but there are many areas, such as, for example, processing specific medical data, where the datasets are significantly limited [4] and obtaining new sample objects is costly. Moreover, many models using neural networks and transformers require large computational power and resources.

In our work, we propose an analysis of the model and loss function, as applied to the Transformer architecture, on the minimum sample size to achieve a predetermined quality set in advance. Based on the approach proposed in the paper [5], we provide a theoretical analysis of the Transformer architecture by considering the differences between the loss function when adding the next sample item to the dataset.

We obtain theoretical estimates for the convergence of this difference in a transformer neural network as the sample size approaches infinity. These results are derived through the analysis of the Hessian spectrum. These estimates allow us to determine the dependence of this difference on the structure of the neural network. We empirically verify these theoretical results by examining the behavior of

Preprint. Under review.

the loss surface on various datasets. The obtained plots substantiate the validity of the theoretical calculations.

Contributions. Our contributions can be summarized as follows:

- We apply a Hessian-based approach to find the critical sufficient dataset size for transformer architectures.
- We demonstrate the validity of our theoretical results through empirical studies on the task of image classification using ViT's
- 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. In Section 2, we review related work, categorizing existing research into key topics and highlighting their main contributions. Section 3 introduces the notation and presents preliminary calculations essential for our analysis. In Section 4, we derive theoretical bounds for the norm of the Hessian matrix and the norm of the difference between loss functions. Section 5 provides an empirical study validating these theoretical results. Sections 6 and 7 discuss and summarize our findings, offering insights and conclusions. Additional experiments and proofs of theorems are included in Appendix A.

2 Related Work

Geometry of Neural Network Loss Landscapes

The geometry of neural network loss landscapes, often analyzed via the Hessian matrix, is a well-studied area. [6] shows that in multi-label classification, the landscape exhibits exactly K directions of high curvature, where K is the number of classes. [7] use random matrix theory to explore loss surface dynamics and optimization. [8] models linear mode connectivity, demonstrating minima connectivity, while [9] explains double descent in finite-width networks. [10] notes landscape flattening under large learning rates. Studies like [8, 11, 12] further explore minima connectivity and landscape structure. However, none of these address how landscape geometry stabilizes with increasing dataset size, a gap our work targets.

Hessian-Based Analysis and Generalization

The Hessian matrix is key to understanding convergence, optimization, and generalization. [5] analyzes fully connected networks, showing how the Hessian spectrum reveals convergence smoothness, while [13] extends this to convolutional networks, noting the Hessian's low effective rank. Yet, these works overlook the impact of sample size on the loss landscape, especially for transformers, an area our study addresses.

Loss Landscapes in Transformers

Transformers, introduced by [1], are central to modern deep learning. [14] provides a theoretical Hessian analysis of transformers, paralleling methods for fully connected networks. [15] studies sample complexity in vision transformers, akin to our sample size focus, while [16] analyzes generalization and dynamics. [17] offers a geometric view of transformer landscapes but ignores minimal sample size. These studies advance transformer landscape understanding but do not explore convergence with increasing dataset size, a gap our work fills.

Dataset Size and Loss Landscape Convergence

Dataset size's impact on loss landscapes, particularly for transformers, is underexplored. [18] questions optimal dataset-model size balances, noting computational costs, and [19] links sufficient samples to flatter minima. However, the concept of a minimum viable dataset size—where additional data cause negligible landscape changes—lacks theoretical grounding. [20] hints at identifying such thresholds via visualization but offers no framework. Our work extends the Hessian-based analyses of [5, 13] to transformers, leveraging [14] to derive convergence bounds as a function of sample size, addressing this critical gap.

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 similar to [] TODO ARTEM'S WORK.

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

Let

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

be a given dataset of size m consists of i.i.d. samples. Let $l(\cdot, \cdot)$ be a given twice differentiable 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:

$$l_i(\mathbf{w}) := l(f_{\mathbf{w}}(x_i), y_i).$$

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

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

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

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

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

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

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 $\mathcal{L}_k(\omega)$ at $\mathbf{w}^*$ we get:

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

3.3 Transformers and Self-Attention Layers

To apply the above concepts to transformer architectures, we define the operation of a single self-attention layer, following a generalized approach similar to [14], which builds on the foundational transformer framework of [1]. This definition will enable us to analyze the Hessian's structure and its implications for loss landscape convergence as the dataset size increases.

Consider a sequence of token embeddings $\mathbf{X} \in \mathbb{R}^{L \times d}$, where L is the sequence length and d is the embedding dimension. A self-attention layer maps $\mathbf{X}$ to an output sequence $\mathbf{F}(\mathbf{X}) \in \mathbb{R}^{L \times d}$ via the following operation:

$$\mathbf{F}(\mathbf{X}) = \mathbf{A}(\mathbf{X})\mathbf{X}\mathbf{W}_V,$$

where $\mathbf{W}_V \in \mathbb{R}^{d \times d}$ is the value weight matrix, and $\mathbf{A}(\mathbf{X}) \in \mathbb{R}^{L \times L}$ is the self-attention matrix, defined as:

$$\mathbf{A}(\mathbf{X}) = a(\mathbf{T}(\mathbf{X})),$$

Here, $\mathbf{T}(\mathbf{X}) \in \mathbb{R}^{L\times L}$ is the query-key similarity transformation, and $a : \mathbb{R}^{L\times L} \to \mathbb{R}^{L\times L}$ is an activation function. In the classical self-attention mechanism [1], the similarity transformation is given by:

$$\mathbf{T}(\mathbf{X}) = \frac{\mathbf{X}\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{X}^\top}{\sqrt{d_K}},$$

where $\mathbf{W}_Q\mathbf{W}_K \in \mathbb{R}^{d\times d_K}$ are the query and key weight matrices, respectively, d_K is the key dimension, and $a =$ softmax (applied row-wise). This formulation allows for flexibility in exploring variations of the attention mechanism, as discussed in [14].

The output $\mathbf{F}(\mathbf{X})$ is then fed into a loss function $l : \mathbb{R}^{L\times d} \times \mathbb{R}^{L\times d} \to \mathbb{R}$, which measures the discrepancy between the predicted sequence $\mathbf{F}(\mathbf{X})$ and the target sequence $\mathbf{Y}$. For simplicity, we assume a mean squared error loss, defined as:

$$l(\mathbf{F}(\mathbf{X}), \mathbf{Y}) = \frac{1}{Ld}\|\mathbf{F}(\mathbf{X}) - \mathbf{Y}\|_F^2,,$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. This loss corresponds to the per-sample loss $l_i(\mathbf{w})$ in our general notation, where ω includes the parameters $\{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\}$ of the self-attention layer.

3.4 Hessian Analysis for Self-Attention Layers

Our goal is to analyze the Hessian $\mathbf{H}_k(\mathbf{w})$ of the empirical loss $\mathcal{L}_k(\mathbf{w})$ with respect to the parameters $\mathbf{w} = \{\mathbf{W}_i\}$ of the self-attention layer, where $\mathbf{W}_i \in \mathbb{R}^{p_i\times q_i}$ (e.g., $i \in \{Q, K, V\}$ for the query, key, and value weights). Following [14], the Hessian can be decomposed into blocks $\frac{\partial^2(\mathcal{L}_k\circ\mathbf{F})}{\partial\mathbf{W}_i\partial\mathbf{W}_j}$, reflecting the interactions between different parameter matrices. This decomposition is crucial for understanding how the loss landscape evolves as the dataset size k increases.

To facilitate this analysis, we leverage the Gauss-Newton decomposition of the Hessian, a standard technique in neural network analysis [14]. For the composite function $\mathcal{L}_k \circ \mathbf{F}$, the Hessian block with respect to parameters $\mathbf{W}_i$ and $\mathbf{W}_j$ can be expressed as:

$$\frac{\partial^2(\mathcal{L}_k \circ \mathbf{F})}{\partial\mathbf{W}_i\partial\mathbf{W}_j} = \left(\frac{\partial\mathbf{F}}{\partial\mathbf{W}_i}\right)^\top \frac{\partial^2\mathcal{L}_k}{\partial\mathbf{F}^2}\left(\frac{\partial\mathbf{F}}{\partial\mathbf{W}_j}\right) + \left(\frac{\partial\mathcal{L}_k}{\partial\mathbf{F}} \otimes I_{p_iq_i}\right)\frac{\partial^2\mathbf{F}}{\partial\mathbf{W}_i\partial\mathbf{W}_j},$$

where the first term is the outer product Hessian (often related to the Gauss-Newton approximation) and the second term is the functional Hessian, capturing higher-order dependencies in the self-attention mechanism. This decomposition, detailed in [14], allows us to isolate the contributions of the self-attention matrix $\mathbf{A}(\mathbf{X})$ and the value transformation $\mathbf{X}\mathbf{W}_V$ to the curvature of the loss landscape.

In the context of our study, this Hessian analysis is critical for understanding how $\mathbf{H}_k(\mathbf{w})$ changes as new samples are added to the dataset.

4 Method

In this section, we derive generalized Hessian expressions for the self-attention layer and extend them to a full transformer block, leveraging these to analyze the convergence of the loss function surface as the dataset size increases. Our approach builds on the theoretical framework of [14], adapting and generalizing their results to provide insights into the sufficient dataset size k^* for transformer architectures.

4.1 Hessian of the Self-Attention Layer

We begin by analyzing the Hessian of a single self-attention layer with parameters $\mathbf{w} = \{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\}$, where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d\times d_K}$ are the query and key weight matrices, and $\mathbf{W}_V \in \mathbb{R}^{d\times d}$ is the value weight matrix. The input is a sequence of token embeddings $\mathbf{X} \in \mathbb{R}^{L\times d}$, where L is the sequence length and d is the embedding dimension. The output of the self-attention layer is:

$$\mathbf{F}(\mathbf{X}) = \mathbf{A}(\mathbf{X})\mathbf{X}\mathbf{W}_V,$$

where $\mathbf{A}(\mathbf{X}) = \text{softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{X}^\top}{\sqrt{d_K}}\right)$, and d_K is the key dimension. The empirical loss is defined as:

$$\mathcal{L}_k(\mathbf{w}) = \frac{1}{k}\sum_{i=1}^{k} l(\mathbf{F}(\mathbf{X}_i), \mathbf{Y}_i),$$

where $l(\mathbf{F}(\mathbf{X}_i), \mathbf{Y}_i)$ is a general loss function, not specified here to maintain generality (unlike the mean squared error used in [14]).

The Hessian of $\mathcal{L}_k$ with respect to the parameters $\mathbf{w}$ is:

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

where $l_i(\mathbf{w}) = l(\mathbf{F}(\mathbf{X}_i), \mathbf{Y}_i)$. For parameters $\mathbf{W}_i, \mathbf{W}_j \in \{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\}$, the Hessian block is decomposed using the Gauss-Newton approximation:

$$\frac{\partial^2\mathcal{L}_k}{\partial\mathbf{W}_i\partial\mathbf{W}_j} = \mathbf{H}_o(\mathbf{W}_i, \mathbf{W}_j) + \mathbf{H}_f(\mathbf{W}_i, \mathbf{W}_j),$$

with $\mathbf{H}_o$ as the outer-product Hessian and $\mathbf{H}_f$ as the functional Hessian.

4.1.1 Generalized Outer-Product Hessian $\mathbf{H}_o$

The outer-product Hessian captures the second-order effects of the loss with respect to the output $\mathbf{F}$:

$$\mathbf{H}_o(\mathbf{W}_i, \mathbf{W}_j) = \left(\frac{\partial\mathbf{F}}{\partial\mathbf{W}_i}\right)^\top \frac{\partial^2\mathcal{L}_k}{\partial\mathbf{F}^2}\left(\frac{\partial\mathbf{F}}{\partial\mathbf{W}_j}\right),$$

where $\frac{\partial\mathbf{F}}{\partial\mathbf{W}_i}$ is the Jacobian of $\mathbf{F}$ with respect to $\mathbf{W}_i$, and $\frac{\partial^2\mathcal{L}_k}{\partial\mathbf{F}^2} \in \mathbb{R}^{L\times d\times L\times d}$ is the Hessian of the loss with respect to $\mathbf{F}$, averaged over k samples. We compute the Jacobians for all parameters:

- **For $\mathbf{W}_V$:**

$$\frac{\partial\mathbf{F}}{\partial\mathbf{W}_V} = \mathbf{A}(\mathbf{X})\mathbf{X} \otimes \mathbf{I}_d,$$

since $\mathbf{A}(\mathbf{X})$ is independent of $\mathbf{W}_V$, and the derivative is a tensor in $\mathbb{R}^{L\times d\times d\times d}$.

- **For $\mathbf{W}_Q$:** Let $\mathbf{T}(\mathbf{X}) = \frac{\mathbf{X}\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{X}^\top}{\sqrt{d_K}}$, so $\mathbf{A}(\mathbf{X}) = \text{softmax}(\mathbf{T}(\mathbf{X}))$. Then:

$$\frac{\partial\mathbf{F}}{\partial\mathbf{W}_Q} = \frac{\partial\mathbf{A}}{\partial\mathbf{T}}\frac{\partial\mathbf{T}}{\partial\mathbf{W}_Q}\mathbf{X}\mathbf{W}_V.$$

The derivative $\frac{\partial\mathbf{T}}{\partial\mathbf{W}_Q} = \frac{1}{\sqrt{d_K}}\mathbf{X} \otimes (\mathbf{X}\mathbf{W}_K)$, and $\frac{\partial\mathbf{A}}{\partial\mathbf{T}}$ is the Jacobian of the softmax, a fourth-order tensor. Thus:

$$\frac{\partial\mathbf{F}}{\partial\mathbf{W}_Q} = \frac{1}{\sqrt{d_K}}\frac{\partial\mathbf{A}}{\partial\mathbf{T}}(\mathbf{X} \otimes \mathbf{X}\mathbf{W}_K)\mathbf{X}\mathbf{W}_V.$$

- **For $\mathbf{W}_K$:** Similarly:

$$\frac{\partial\mathbf{T}}{\partial\mathbf{W}_K} = \frac{1}{\sqrt{d_K}}(\mathbf{X}\mathbf{W}_Q) \otimes \mathbf{X},$$

so:

$$\frac{\partial\mathbf{F}}{\partial\mathbf{W}_K} = \frac{1}{\sqrt{d_K}}\frac{\partial\mathbf{A}}{\partial\mathbf{T}}(\mathbf{X}\mathbf{W}_Q \otimes \mathbf{X})\mathbf{X}\mathbf{W}_V.$$

For each pair $(\mathbf{W}_i, \mathbf{W}_j)$, $\mathbf{H}_o$ is computed by contracting the Jacobians with $\frac{\partial^2\mathcal{L}_k}{\partial\mathbf{F}^2}$. For example: - $\mathbf{H}_o(\mathbf{W}_Q, \mathbf{W}_K) = \frac{1}{d_K}\left((\mathbf{X} \otimes \mathbf{X}\mathbf{W}_K)^\top\frac{\partial\mathbf{A}^\top}{\partial\mathbf{T}}\mathbf{X}\mathbf{W}_V\right)^\top\frac{\partial^2\mathcal{L}_k}{\partial\mathbf{F}^2}\left(\frac{\partial\mathbf{A}}{\partial\mathbf{T}}(\mathbf{X}\mathbf{W}_Q \otimes \mathbf{X})\mathbf{X}\mathbf{W}_V\right)$.

The full set of expressions follows this pattern, remaining general without specifying $\frac{\partial^2\mathcal{L}_k}{\partial\mathbf{F}^2}$.

4.1.2 Generalized Functional Hessian $\mathbf{H}_f$

The functional Hessian accounts for the curvature of $\mathbf{F}$:

$$\mathbf{H}_f(\mathbf{W}_i, \mathbf{W}_j) = \left(\frac{\partial \mathcal{L}_k}{\partial \mathbf{F}} \otimes \mathbf{I}_{p_i q_i} \right) \frac{\partial^2 \mathbf{F}}{\partial \mathbf{W}_i \partial \mathbf{W}_j},$$

where $\frac{\partial \mathcal{L}_k}{\partial \mathbf{F}} \in \mathbb{R}^{L \times d}$ is the gradient of the loss with respect to $\mathbf{F}$, and $p_i q_i$ is the size of $\mathbf{W}_i$ (e.g., dd_K for $\mathbf{W}_Q$). The second derivative $\frac{\partial^2 \mathbf{F}}{\partial \mathbf{W}_i \partial \mathbf{W}_j}$ varies by pair:

- $\mathbf{H}_f(\mathbf{W}_V, \mathbf{W}_V)$**:** Since $\mathbf{F} = \mathbf{A}\mathbf{X}\mathbf{W}_V$ is linear in $\mathbf{W}_V$, $\frac{\partial^2 \mathbf{F}}{\partial \mathbf{W}_V \partial \mathbf{W}_V} = 0$, so $\mathbf{H}_f(\mathbf{W}_V, \mathbf{W}_V) = 0$. - $\mathbf{H}_f(\mathbf{W}_Q, \mathbf{W}_K)$**:**

$$\frac{\partial^2 \mathbf{F}}{\partial \mathbf{W}_Q \partial \mathbf{W}_K} = \frac{1}{d_K} \frac{\partial^2 \mathbf{A}}{\partial \mathbf{T}^2} \left((\mathbf{X} \otimes \mathbf{X}\mathbf{W}_K) \otimes (\mathbf{X}\mathbf{W}_Q \otimes \mathbf{X}) \right) \mathbf{X}\mathbf{W}_V + \frac{1}{\sqrt{d_K}} \frac{\partial \mathbf{A}}{\partial \mathbf{T}} (\mathbf{X} \otimes \mathbf{X}),$$

$$\mathbf{H}_f(\mathbf{W}_Q, \mathbf{W}_K) = \frac{1}{d_K} \left(\frac{\partial \mathcal{L}_k}{\partial \mathbf{F}} \otimes \mathbf{I}_{dd_K} \right) \left(\frac{\partial^2 \mathbf{A}}{\partial \mathbf{T}^2} \left((\mathbf{X} \otimes \mathbf{X}\mathbf{W}_K) \otimes (\mathbf{X}\mathbf{W}_Q \otimes \mathbf{X}) \right) \mathbf{X}\mathbf{W}_V + \sqrt{d_K} \frac{\partial \mathbf{A}}{\partial \mathbf{T}} (\mathbf{X} \otimes \mathbf{X}) \right).$$

Other pairs follow similarly, with $\mathbf{H}_f$ non-zero when $\mathbf{W}_i$ and $\mathbf{W}_j$ both influence $\mathbf{A}(\mathbf{X})$.

4.2 Hessian of the Transformer Block

A transformer block extends the self-attention layer with a feed-forward network (FFN), residual connections, and layer normalization. The output is:

$$\mathbf{Y} = \text{LayerNorm}(\mathbf{X} + \text{SelfAttention}(\mathbf{X})),$$
$$\mathbf{Z} = \text{LayerNorm}(\mathbf{Y} + \text{FFN}(\mathbf{Y})),$$

where $\text{FFN}(\mathbf{Y}) = \sigma(\mathbf{Y}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$, with $\mathbf{W}_1 \in \mathbb{R}^{d \times d_{\text{ff}}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}} \times d}$, $b_1 \in \mathbb{R}^{d_{\text{ff}}}$, $b_2 \in \mathbb{R}^d$, and σ as the activation (e.g., ReLU). The $\text{LayerNorm}(\mathbf{X})$ operation is defined as follows. For an input matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ (where n is the batch size and m is the feature dimension), we compute:

1. Feature-wise mean and variance:

$$\mu_i = \frac{1}{m} \sum_{j=1}^{m} \mathbf{X}_{i,j}, \quad \sigma_i^2 = \frac{1}{m} \sum_{j=1}^{m} (\mathbf{X}_{i,j} - \mu_i)^2,$$

   where $\epsilon > 0$ ensures numerical stability.

2. Normalized output with learnable parameters $\gamma, \beta \in \mathbb{R}^m$:

$$\text{LayerNorm}(\mathbf{X})_{i,j} = \gamma_j \cdot \frac{\mathbf{X}_{i,j} - \mu_i}{\sqrt{\sigma_i^2}} + \beta_j.$$

Therefore the $\text{LayerNorm}(\mathbf{X})$ operation can be represented as follows:

$$\text{LayerNorm}(\mathbf{X}) = \gamma \odot \frac{\mathbf{X} - \mu \mathbf{1}^\top}{\sqrt{\sigma^2 \mathbf{1}^\top + \epsilon}} + \beta.$$

The parameters are $\mathbf{w} = \{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2, \gamma, \beta\}$, where γ and β are the scale and shift parameters of LayerNorm. For simplicity in Hessian analysis, one may assume γ and β are fixed (e.g., $\gamma = \mathbf{1}$, $\beta = \mathbf{0}$), though they are typically learnable.

Theorem 1. *NEED TO BE COMPLETED TODO*

For a transformer block with parameters $\mathbf{w}$*, the Hessian of the loss* $\mathcal{L}_k$ *with respect to parameters* $\mathbf{w}_i, \mathbf{w}_j \in \mathbf{w}$ *is:*

$$\mathbf{H}_k = \mathbf{H}_o + \mathbf{H}_f,$$

where:

$$\mathbf{H}_o(\mathbf{w}_i, \mathbf{w}_j) = \left(\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_i} \right)^\top \frac{\partial^2 \mathcal{L}_k}{\partial \mathbf{Z}^2} \left(\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_j} \right),$$

$$\mathbf{H}_f(\mathbf{w}_i, \mathbf{w}_j) = \left(\frac{\partial \mathcal{L}_k}{\partial \mathbf{Z}} \otimes \mathbf{I}_{p_i q_i} \right) \frac{\partial^2 \mathbf{Z}}{\partial \mathbf{w}_i \partial \mathbf{w}_j}.$$

The Jacobians $\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_i}$ are computed via the chain rule through the layers, incorporating derivatives from both the self-attention and FFN components, adjusted for layer normalization. The proof is provided in Appendix B.1.

4.3 Convergence of the Loss Function Surface

Similarly to [5] let us use second-order Taylor approximation for the mentioned above loss functions at $\mathbf{w}^*$. We suppose that decomposition to the second order will be sufficient to study local behavior. The first-order term vanishes because the gradients $\nabla \mathcal{L}_k(\mathbf{w}^*)$ and $\nabla \mathcal{L}_{k+1}(\mathbf{w}^*)$ are zero:

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

where we denoted the Hessian of $\mathcal{L}_k(\mathbf{w})$ with respect to parameters $\mathbf{w}$ at $\mathbf{w}^*$ as $\mathbf{H}_k(\mathbf{w}^*) \in \mathbb{R}^{P \times P}$. Moreover, the total Hessian can be written as the average value of the Hessians of the individual terms of the empirical loss function:

$$\mathbf{H}_k(\mathbf{w}) = \nabla^2_{\mathbf{w}} \mathcal{L}_k(\mathbf{w}) = \frac{1}{k} \sum_{i=1}^{k} \nabla^2_{\mathbf{w}} l(\mathbf{f}_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i) = \frac{1}{k} \sum_{i=1}^{k} \mathbf{H}_i(\mathbf{w}).$$

Therefore, using the obtained second-order approximation (1), the formula for the difference of losses becomes:

$$\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w}) = \frac{1}{k+1} \left(l(\mathbf{f}_{\mathbf{w}^*}(\mathbf{x}_{k+1}), \mathbf{y}_{k+1}) - \frac{1}{k} \sum_{i=1}^{k} l(\mathbf{f}_{\mathbf{w}^*}(\mathbf{x}_i), \mathbf{y}_i) \right) +$$

$$+ \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^\top \left(\mathbf{H}_{k+1}(\mathbf{w}^*) - \frac{1}{k} \sum_{i=1}^{k} \mathbf{H}_i(\mathbf{w}^*) \right) (\mathbf{w} - \mathbf{w}^*).$$

After that, using the triangle inequality, we can derive the following:

$$|\mathcal{L}_{k+1}(\mathbf{w}) - \mathcal{L}_k(\mathbf{w})| \leqslant \frac{1}{k+1} \left| l(\mathbf{f}_{\mathbf{w}^*}(\mathbf{x}_{k+1}), \mathbf{y}_{k+1}) - \frac{1}{k} \sum_{i=1}^{k} l(\mathbf{f}_{\mathbf{w}^*}(\mathbf{x}_i), \mathbf{y}_i) \right| +$$

$$+ \frac{1}{2(k+1)} \|\mathbf{w} - \mathbf{w}^*\|_2^2 \left\| \mathbf{H}_{k+1}(\mathbf{w}^*) - \frac{1}{k} \sum_{i=1}^{k} \mathbf{H}_i(\mathbf{w}^*) \right\|_2.$$

So the problem of the boundedness and convergence of the losses difference is reduced to the analysis of the two terms:

- Difference of the **loss functions at optima** for new object and previous ones:

$$\left| l(\mathbf{f}_{\mathbf{w}^*}(\mathbf{x}_{k+1}), \mathbf{y}_{k+1}) - \frac{1}{k} \sum_{i=1}^{k} l(\mathbf{f}_{\mathbf{w}^*}(\mathbf{x}_i), \mathbf{y}_i) \right|,$$

- Difference of the **Hessians at optima** for new object and previous ones:

$$\left\| \mathbf{H}_{k+1}(\mathbf{w}^*) - \frac{1}{k} \sum_{i=1}^{k} \mathbf{H}_i(\mathbf{w}^*) \right\|_2.$$

It should be mentioned that the first term can be easily upper-bounded by a constant, since the loss function itself takes limited values. However, the expression with Hessians is not so easy to evaluate. The rest of the work is devoted to a thorough analysis of this difference. Thus, we analyze the local convergence of the landscape of the loss function using its Hessian. As a consequence of Theorem 1 regarding the decomposition of the transformer block, the Hessian $\mathbf{H}_k$ can be expressed as $\mathbf{H}_k = \mathbf{H}_o + \mathbf{H}_f$, where $\mathbf{H}_o$ and $\mathbf{H}_f$ are the outer-product and functional Hessians, respectively, as derived earlier.

5 Experiments

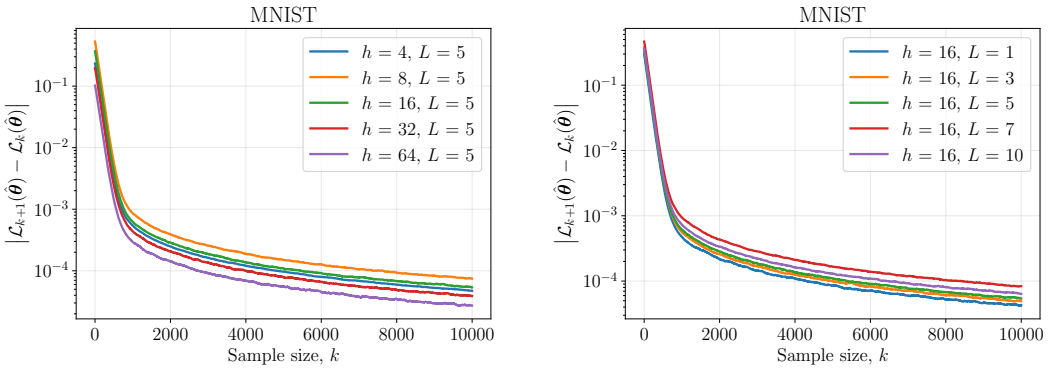
To verify the theoretical estimates obtained, we conducted a detailed empirical study. The experiments are divided into two main parts: the base experiment, which demonstrates the general behavior of the loss function convergence, and the transformer-specific experiment, which focuses on the Vision Transformer (ViT) architecture. Both experiments aim to validate the theoretical results and provide insights into the practical implications of our findings.

5.1 Base Experiment

The goal of the base experiment is to empirically observe the convergence of the loss function as the dataset size increases. Specifically, we aim to verify whether the difference between the loss functions for consecutive dataset sizes, $|\mathcal{L}_{k+1}(\theta) - \mathcal{L}_k(\theta)|$ decreases as k grows, as predicted by our theoretical analysis.

We consider a simple neural network architecture, such as a fully connected network, trained on a synthetic dataset. The dataset is generated from a known distribution to ensure control over the data generation process. The loss function used is the mean squared error (MSE), and the network is trained using stochastic gradient descent (SGD).

The results of the base experiment are presented in Figure 1. The plot shows the convergence of $|\mathcal{L}_{k+1}(\theta) - \mathcal{L}_k(\theta)|$ as a function of the dataset size k. As expected, the difference decreases monotonically, indicating that the loss function surface stabilizes as the dataset size increases. This behavior aligns with our theoretical predictions, confirming that the loss function converges as the dataset grows.



(a) (b)

Figure 1: Convergence of the loss function difference $|\mathcal{L}_{k+1}(\theta) - \mathcal{L}_k(\theta)|$ as a function of the dataset size k. The plots demonstrates the stabilization of the loss function surface as the dataset size increases.

5.2 Transformer-Specific Experiment

In this experiment, we focus on the Vision Transformer (ViT) architecture to validate our theoretical results in a more complex and practical setting. The goal is to observe the convergence of the loss function and the quality of classification as the dataset size increases, using a pre-trained ViT model fine-tuned on a small dataset.

We use a pre-trained Vision Transformer model from HuggingFace's transformers library. The model is fine-tuned on a small dataset of images, such as anime characters, cats, or dogs. The fine-tuning process is conducted in two ways:

1. **LoRA (Low-Rank Adaptation)**: We apply parameter-efficient fine-tuning using LoRA, which allows us to adapt the model with minimal changes to the original parameters.
2. **Unfreezing the Last Layers**: We unfreeze the last few layers of the model and attach a classification head, allowing the model to adapt to the specific classification task.

The loss function used is the cross-entropy loss, and the model is trained using the Adam optimizer. We monitor both the classification accuracy and the convergence of the loss function as the dataset size increases.

The results of the transformer-specific experiment will be presented in two parts:

1. **Classification Accuracy**: We will report the classification accuracy on a validation set as a function of the dataset size. This will demonstrate how the model's performance improves with more data.
2. **Loss Function Convergence**: We will plot the difference $|\mathcal{L}_{k+1}(\theta) - \mathcal{L}_k(\theta)|$ as a function of the dataset size k, similar to the base experiment. This will show how the loss function surface stabilizes as the dataset size increases.

6 Discussion

TODO

7 Conclusion

TODO

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. URL `https://arxiv.org/abs/1706.03762`.

[2] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Chen, Peizhao Zhang, Zhicheng Sun, Wei Yu, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2010.11929*, 2020. URL `https://arxiv.org/abs/2010.11929`.

[3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL `https://arxiv.org/abs/2001.08361`.

[4] Miguel Moreno, Isabelle Bichindaritz, Adam Wilcox, Eric Tchetgen Tchetgen, and Milos Hauskrecht. Few-shot learning with semi-supervised transformers for electronic health records. *arXiv preprint arXiv:2402.14474*, 2024. URL `https://arxiv.org/abs/2402.14474`.

[5] 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.

[6] Stanislav Fort and Stanislaw Jastrzebski. Emergent properties of the local geometry of neural loss landscapes. *Placeholder - replace with actual entry*, 2019. Please provide full citation details.

[7] Jeffrey Pennington et al. Placeholder for pennington et al. 2017. *Placeholder - replace with actual entry*, 2017. Please provide full citation details.

[8] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P. Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *arXiv preprint arXiv:1802.10026*, 2018. URL `https://arxiv.org/abs/1802.10026`.

[9] Sidak Pal Singh et al. Phenomenology of double descent in finite-width neural networks. *Placeholder - replace with actual entry*, 2022. Please provide full citation details.

[10] Lei Wang et al. Instabilities of large learning rates in neural network training. *Placeholder - replace with actual entry*, 2023. Please provide full citation details.

[11] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2019. URL https://arxiv.org/abs/1803.00885.

[12] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. The loss surface of deep and wide neural networks. *arXiv preprint arXiv:1704.08045*, 2017. URL https://arxiv.org/abs/1704.08045.

[13] 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.

[14] Weronika Ormaniec, Felix Dangel, and Sidak Pal Singh. What does it mean to be a transformer? insights from a theoretical hessian analysis. *arXiv preprint arXiv:2410.10986*, 2024. Self-Attention Block decomposition.

[15] Hongkang Li, Meng Xu, Tianyang Wang, Shuai Yang, Feng Shen, Wei Xu, and Trevor Darrell. A theoretical understanding of shallow vision transformers: Learning, generalization, and sample complexity. *OpenReview*, 2023. URL https://openreview.net/forum?id=jClGv3Qjhb.

[16] Yingying Zhang, Meng Xu, Tianyang Wang, Shuai Yang, Feng Shen, Wei Xu, and Trevor Darrell. Understanding generalization in transformers: Error bounds and training dynamics under benign and harmful overfitting. *arXiv preprint arXiv:2502.12508*, 2025. URL https://arxiv.org/abs/2502.12508.

[17] Anonymous. Stagewise development in transformers and the geometry of the loss landscape. *OpenReview*, 2024. URL https://openreview.net/forum?id=xEZiEhjTeq.

[18] Jordan Hoffmann et al. Training compute-optimal large language models. *Placeholder - replace with actual entry*, 2022. Please provide full citation details.

[19] Yuhuai Wu et al. Towards understanding generalization of deep learning. *Placeholder - replace with actual entry*, 2017. Please provide full citation details.

[20] Tiankai Xie, Xiangyu Li, Yan Zhang, Yiming Wang, Hao Zhang, Mingyuan Liu, and Jie Zhang. Losslens: Diagnostics for machine learning through loss landscape visual analytics. *arXiv preprint arXiv:2412.13321*, 2024. URL https://arxiv.org/abs/2412.13321.

A Appendix / supplemental material

A.1 Additional experiments

TODO

B Appendix / Proofs of the Theorems

B.1 Proof of Theorem 1

NEED TO BE COMPLETED TODO

Proof. Consider the transformer block output $\mathbf{Z} = \text{LayerNorm}(\mathbf{Y} + \text{FFN}(\mathbf{Y}))$, where $\mathbf{Y} = \text{LayerNorm}(\mathbf{X} + \text{SelfAttention}(\mathbf{X}))$, and the empirical loss $\mathcal{L}_k(\mathbf{w}) = \frac{1}{k}\sum_{i=1}^{k} l(\mathbf{Z}_i, \mathbf{Y}_i)$. The Hessian is:

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

For parameters $\mathbf{w}_i, \mathbf{w}_j \in \mathbf{w}$, apply the chain rule to the composite function $\mathcal{L}_k \circ \mathbf{Z}$:

$$\frac{\partial^2 \mathcal{L}_k}{\partial \mathbf{w}_i \partial \mathbf{w}_j} = \frac{\partial}{\partial \mathbf{w}_i}\left(\left(\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_j}\right)^\top \frac{\partial \mathcal{L}_k}{\partial \mathbf{Z}}\right).$$

Using the product rule:

$$\frac{\partial^2 \mathcal{L}_k}{\partial \mathbf{w}_i \partial \mathbf{w}_j} = \left(\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_i}\right)^\top \frac{\partial^2 \mathcal{L}_k}{\partial \mathbf{Z}^2}\left(\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_j}\right) + \left(\frac{\partial \mathcal{L}_k}{\partial \mathbf{Z}} \otimes \mathbf{I}_{p_i q_i}\right)\frac{\partial^2 \mathbf{Z}}{\partial \mathbf{w}_i \partial \mathbf{w}_j}.$$

- **Outer-Product Term:** TODO The first term, $\mathbf{H}_o$, arises from the derivative of the loss with respect to $\mathbf{Z}$, requiring the Jacobians $\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_i}$, calculated as follows:

$$\frac{\partial \mathbf{Z}}{\partial \mathbf{w}_i} = \frac{\partial \text{LayerNorm}}{\partial(\mathbf{Y} + \text{FFN}(\mathbf{Y}))}\left(\frac{\partial \mathbf{Y}}{\partial \mathbf{w}_i} + \frac{\partial \text{FFN}}{\partial \mathbf{Y}}\frac{\partial \mathbf{Y}}{\partial \mathbf{w}_i}\right) =,$$

where $\frac{\partial \mathbf{Y}}{\partial \mathbf{w}_i}$ involves the self-attention derivatives adjusted for layer normalization.

- **Functional Term:** TODO The second term, $\mathbf{H}_f$, involves the second derivative of $\mathbf{Z}$, which is non-zero for pairs affecting intermediate layers (e.g., $\mathbf{W}_Q$ and $\mathbf{W}_K$) and zero when $\mathbf{Z}$ is linear in a parameter (e.g., $\mathbf{W}_2$).

This decomposition holds for all parameter pairs, completing the proof. □