Local Properties of Neural Networks Through the Lens of Layer-wise Hessians

Maxim Bolshim, ITMO University
maxim.bolshim@yandex.ru

Alexander Kugaevskikh, ITMO University
a-kugaevskikh@yandex.ru

May 29, 2025

Abstract

This paper introduces a novel methodology for analyzing neural networks through the lens of layer-wise Hessian matrices. We formalize the concept of the local Hessian for individual functional blocks (layers) of a neural network and demonstrate its utility for characterizing the geometry of the parameter space. The spectral properties of local Hessians are shown to provide quantitative insights into phenomena such as overfitting, underparameterization, and the expressivity of neural architectures. We conduct a comprehensive empirical study across 111 experiments on 37 datasets, revealing consistent patterns in the evolution and structure of local Hessians during training. These findings establish a foundation for principled diagnostics and informed design of neural network architectures based on their local geometric properties.

1 Introduction

Deep neural networks have demonstrated outstanding results in many fields, including computer vision, natural language processing, and other machine learning tasks [5, 6]. However, despite their practical success, the question of why certain architectures outperform others and how to formally and systematically improve neural network design remains open. The empirical approach, based on trial and error, is becoming increasingly costly as model sizes and data volumes grow.

A number of researches have demonstrated that analyzing the curvature of the loss landscape via Hessians and related spectral tools can shed light on the training dynamics and generalization ability of neural networks [7, 2, 3, 4]. In this work, we put forward the thesis that local properties of the parameter space of a neural network can provide early insights into the internal characteristics of the model, even at the initial stages of training [1]. Specifically, we propose to utilize local Hessian matrices—matrices of second derivatives of the objective function with respect to the parameters of individual layers—to analyze the geometry of the parameter space.

The concept of the local Hessian enables a formal and quantitative characterization of the geometric properties of the parameter space in the vicinity of an optimization point. In particular, we demonstrate that the spectrum of the local Hessian—such as the distribution and structure of its eigenvalues [8]—is closely related to the functional properties of the corresponding layers of a neural network.

The main contributions of this work are as follows:

- The introduction of a mathematically rigorous definition of the local Hessian for functional blocks of a neural network;
- A detailed analysis of the spectral properties of local Hessians during neural network training;
- An investigation of the geometric interpretation of the neural network parameter space through the lens of local Hessians.

The results obtained not only deepen our theoretical understanding of deep neural networks, but also open new perspectives for studying their internal dynamics.

2 Mathematical Framework of Neural Networks

2.1 Definition and Structure of a Neural Network

Definition 1. *A neural network $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^m$ is a parameterized function with parameters $\theta \in \mathbb{R}^P$, mapping input data $x \in \mathbb{R}^d$ to an output space via a sequence of functional transformations. We denote by $\mathcal{F}(x;\theta)$ the result of applying the network to input x with given parameters θ.*

Definition 2. *A functional block (layer) C_i of a neural network $\mathcal{F}$ is defined as a pair of modules (P_i, A_i), where:*

- *$P_i : \mathbb{R}^{d_i} \times \mathbb{R}^{p_i} \to \mathbb{R}^{q_i}$ is a parameterized transformation with parameters $\theta_i \in \mathbb{R}^{p_i}$.*
- *$A_i : \mathbb{R}^{q_i} \to \mathbb{R}^{q_i}$ is an activation function (potentially the identity).*

Definition 3. *A neural network $\mathcal{F}$ can be represented as a composition of n functional blocks:*

$$\mathcal{F}(x;\theta) = (C_n \circ C_{n-1} \circ \ldots \circ C_1)(x), \tag{1}$$

where $C_i(z) = A_i(P_i(z;\theta_i))$ for input z, and $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ is the complete set of network parameters.

This representation of a neural network enables the analysis of each functional block independently, which is essential for local analysis of network properties. Decomposing a complex model into simpler components is a key methodological approach that allows the application of spectral analysis tools to individual components.

2.2 Intermediate Representations and Activation Functions

Definition 4. *The intermediate representation z_i is defined as the input to block C_i:*

$$z_i = \begin{cases} x, & \text{if } i = 1 \\ (C_{i-1} \circ \ldots \circ C_1)(x), & \text{if } i > 1 \end{cases} \tag{2}$$

Accordingly, the output of block C_i is denoted as:

$$y_i = C_i(z_i) = A_i(P_i(z_i;\theta_i)) \tag{3}$$

Intermediate representations play a crucial role in neural network analysis, as they capture how the input signal is transformed at each processing stage. Of particular interest is the study of the geometry of these intermediate representations and their relationship to the parameters of the corresponding layers.

Definition 5. *For block C_i, we define the local scalar function $S_i : \mathbb{R}^{p_i} \to \mathbb{R}$ as:*

$$S_i(\theta_i) = \varphi(A_i(P_i(z_i; \theta_i))), \tag{4}$$

where $\varphi : \mathbb{R}^{q_i} \to \mathbb{R}$ is an aggregation function, typically $\varphi(y) = \sum_{j=1}^{q_i} y_j$.

The scalar function provides a means to assess the influence of the parameters of a specific layer on its output for a fixed input. This function is central for defining the local Hessian in the following section.

2.3 Typical Implementations in Neural Networks

In modern neural networks, the following component implementations are common:

- P_i — linear transformation $P_i(z_i; \theta_i) = W_i z_i + b_i$, where $\theta_i = \{W_i, b_i\}$
- A_i — nonlinear activation function, e.g., ReLU, Sigmoid, or Tanh
- $\varphi(y_i) = \sum_{j=1}^{q_i} y_{i,j}$ — summation of all components of the output vector

These definitions and notations will be used throughout the remainder of this work to ensure mathematical rigor and consistency.

3 Local Layer-wise Hessians in Neural Networks

3.1 Definition of the Local Hessian

Definition 6. *The local Hessian matrix $H_i \in \mathbb{R}^{p_i \times p_i}$ (hereafter, local Hessian, LH_i) for block C_i is defined as the matrix of second derivatives of the scalar function S_i with respect to the parameters θ_i:*

$$H_i = \nabla^2_{\theta_i} S_i(\theta_i) = \left[\frac{\partial^2 S_i(\theta_i)}{\partial \theta_{i,j} \partial \theta_{i,k}} \right]_{j,k=1}^{p_i} \tag{5}$$

Analysis of this matrix provides insights into:

- The degree of nonlinearity of the transformation performed by the layer
- Interdependencies among parameters and their influence on the layer output
- Geometric properties of the parameter space
- Sensitivity of the layer to small parameter perturbations

In differential geometry, the Hessian of a function at a point defines a quadratic form that approximates the curvature of the function's level surface. In the context of neural networks, LH_i characterizes the curvature of the functional response of a layer in its parameter space [9]. When analyzing the surface, the sign and distribution of the eigenvalues of LH_i are crucial, as they determine the local geometry.

3.2 Efficient Computation of LH_i

The analysis of neural networks using LH_i is a powerful tool; however, due to the quadratic scaling of the Hessian matrix size with respect to the number of parameters, its direct computation is often computationally infeasible for modern architectures. Consequently, various approximation methods for LH_i are widely used in practice [10, 11, 12, 13]. This section presents a methodology for working with LH_i that addresses these computational challenges.

For efficient computation of LH_i, we propose an algorithm based on the sequential calculation of the matrix rows.

Lemma 1. *The elements of the Hessian matrix H_i can be computed sequentially by rows:*

$$\begin{aligned} g_i = \nabla_{\theta_i} S_i(\theta_i) &= \left[\frac{\partial S_i}{\partial \theta_{i,j}}\right]_{j=1}^{p_i} \\ H_i[j,:] = \nabla_{\theta_i} g_{i,j} &= \nabla_{\theta_i}\left(\frac{\partial S_i}{\partial \theta_{i,j}}\right) \end{aligned} \tag{6}$$

Proof. By the definition of the Hessian matrix, its element $H_i[j,k]$ is given by:

$$H_i[j,k] = \frac{\partial^2 S_i(\theta_i)}{\partial \theta_{i,j} \partial \theta_{i,k}} \tag{7}$$

Denoting $g_{i,j} = \frac{\partial S_i}{\partial \theta_{i,j}}$, we have

$$H_i[j,k] = \frac{\partial g_{i,j}}{\partial \theta_{i,k}} \tag{8}$$

Thus, the j-th row of H_i is the gradient of the j-th component of the gradient of S_i. □

This approach significantly reduces computational costs when working with large models, as it does not require storing the entire Hessian matrix of size $p_i \times p_i$ in memory simultaneously.

The proposed method is particularly important for the analysis of modern deep neural networks containing millions of parameters, since the full LH_i matrix for such models would be prohibitively large. The local approach not only makes the computations practically feasible, but also enables focused analysis of individual network components, which is often more informative than global analysis.

3.3 Algorithm for Computing the Local Hessian

Algorithm 1 Computation of Local Hessians in a Neural Network

Require: Neural network $\mathcal{F}$, input $x \in \mathbb{R}^d$, aggregation function φ
Ensure: Set of local Hessian matrices $\{LH_1, LH_2, \ldots, LH_n\}$

Decompose $\mathcal{F}$ into functional blocks $\{C_1, C_2, \ldots, C_n\}$, where $C_i = (P_i, A_i)$
for $i = 1$ to n **do**
    Compute $z_i = (C_{i-1} \circ \ldots \circ C_1)(x)$ ▷ Input to block C_i
    Compute $y_i = A_i(P_i(z_i; \theta_i))$ ▷ Output of block C_i
    Compute $S_i = \varphi(y_i)$ ▷ Scalar function for the block
    Compute the gradient $g_i = \nabla_{\theta_i} S_i$
    Initialize $LH_i \in \mathbb{R}^{p_i \times p_i}$ as a zero matrix
    for $j = 1$ to p_i **do**
        if $g_{i,j}$ is not constant with respect to θ_i **then**
            Compute $LH_i[j,:] = \nabla_{\theta_i} g_{i,j}$
        else
            $LH_i[j,:] = \vec{0}$
        end if
    end for
end for
return $\{LH_1, LH_2, \ldots, LH_n\}$

The graphical representation of the algorithm is shown in Figure 1. This algorithm allows for the efficient computation of local Hessians for each layer of a neural network, focusing on the specific functional transformations performed by each layer. The sequential computation of the Hessian rows ensures that memory usage remains manageable, even for large models with millions of parameters.

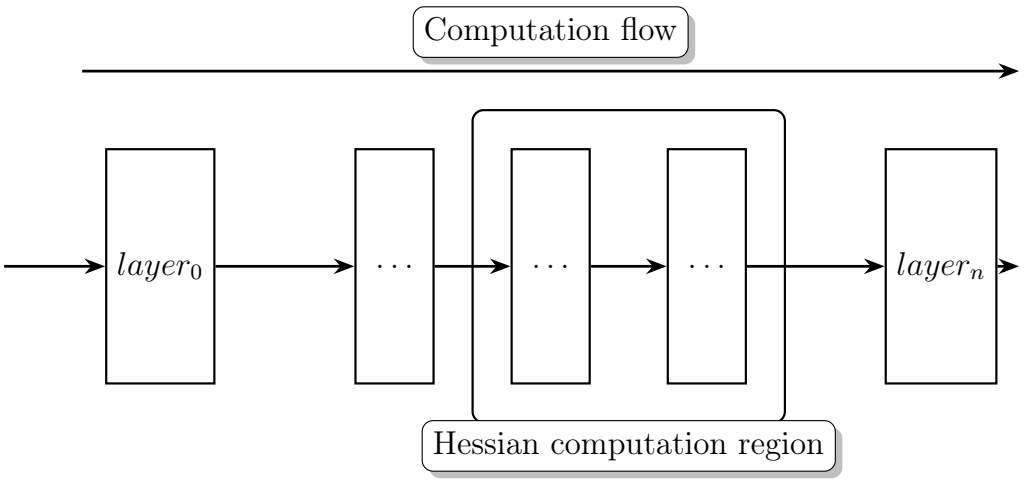

Figure 1: Visualization of local Hessian computation

3.4 Mathematical Implementation Details

3.4.1 Gradient Computation g_i

In the context of automatic differentiation, the gradient g_i is computed as:

$$g_i = \nabla_{\theta_i} S_i = \frac{\partial S_i}{\partial y_i} \cdot \frac{\partial y_i}{\partial P_i} \cdot \frac{\partial P_i}{\partial \theta_i} \tag{9}$$

where:

- $\frac{\partial S_i}{\partial y_i} = \nabla_{y_i} \varphi(y_i)$ is the gradient of the aggregation function,
- $\frac{\partial y_i}{\partial P_i} = \nabla_{P_i} A_i(P_i)$ is the Jacobian of the activation function,
- $\frac{\partial P_i}{\partial \theta_i}$ is the Jacobian of the parameterized transformation with respect to its parameters.

3.4.2 Computation of Hessian Matrix Rows

For each component j of the gradient g_i, its gradient with respect to θ_i is computed as:

$$H_i[j, :] = \nabla_{\theta_i} g_{i,j} = \nabla_{\theta_i} \left(\frac{\partial S_i}{\partial \theta_{i,j}} \right) \tag{10}$$

This requires repeated application of automatic differentiation to each component of the gradient.

3.5 Spectral Analysis of the Local Hessian

A detailed study of the spectrum of LH_i provides an important tool for understanding the geometry of the parameter space. The distribution and structure of the eigenvalues reflect key curvature properties of the function, which is especially relevant for analyzing the conditioning of the optimization problem [14, 15].

For each LH_i, one can compute:

$$LH_i = U_i \Lambda_i U_i^T = \sum_{j=1}^{p_i} \lambda_{i,j} u_{i,j} u_{i,j}^T \tag{11}$$

where $\lambda_{i,j}$ is the j-th eigenvalue and $u_{i,j}$ is the corresponding eigenvector.

Characteristic spectral indicators of the Hessian include:

- **Trace of the Hessian:** $\mathrm{tr}(LH_i) = \sum_{j=1}^{p_i} \lambda_{i,j}$ — the sum of the eigenvalues
- **Determinant of the Hessian:** $\det(LH_i) = \prod_{j=1}^{p_i} \lambda_{i,j}$ — the product of the eigenvalues

Of particular interest is the observation of the distribution of eigenvalues across network layers and their evolution during training, which enables tracking the changes in the geometry of the parameter space.

The distribution of eigenvalues of LH_i provides information about the geometry of the functional space of a layer. In particular, a concentration of eigenvalues near zero indicates the presence of manifolds of equal function values (plateaus), which complicates optimization by gradient-based methods.

- **Small networks:** If the network is too small (few layers d_l or narrow layers), the linear transformation $z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$ becomes poorly expressed, leading to a large mean $\mathbb{E}[|z^{(l)}|] \gg 1$ and shifting $z^{(l)}$ into the saturation regions of activation functions such as sigmoid or tanh, where $f'(z) \approx 0$. This exacerbates the vanishing gradient problem and significantly affects the structure of LH_i, concentrating its eigenvalues near zero.
- **Overly large networks:** An excessively large network can "memorize" noisy details of the data, resulting in an increase in the norm of the weights $||W^{(l)}|| \gg 1$ and shifting $z^{(l)}$ into the saturation region. Under such conditions, LH_i also acquires a specific structure with many very small eigenvalues, reflecting excessive freedom in the parameter space.
- **Inappropriate architecture or inputs:** The architecture may be unsuitable for the specific properties of the data (high nonlinearity, variability of distributions, multidimensional dependencies). Poorly normalized or noisy inputs further increase the spread of $z^{(l)}$, intensifying saturation. As a result, neurons "freeze" and cease to participate effectively in training.

4 Research Methodology

4.1 Experimental Design

To assess the spectral properties of local Hessians, a comprehensive analysis was conducted on neural networks of various architectures across a set of 37 datasets (22 for classification tasks and 15 for regression). For each dataset, the following parameters were varied:

- Number of layers and neurons in the networks
- Weight initialization methods
- Optimization algorithms (Adam, SGD, RMSProp)
- Activation functions (ReLU, Sigmoid, Tanh)

The total number of parameters in the studied models ranged from 13 to 9 million, and the number of layers varied from 1 to 5, enabling the analysis of networks with different levels of parameterization. For both tasks, the classical multilayer perceptron was used as the primary architecture. Cross-entropy loss was employed for classification tasks, and mean squared error was used for regression tasks. Hyperparameters for each dataset were selected empirically.

4.2 Experimental Data Collection

A specialized system was developed to monitor the evolution of internal characteristics of neural networks during training. The experiments followed the methodology outlined below:

1. For each dataset, three model variants were trained:
   - A model with a small number of parameters (type "no")
   - A model with a moderate number of parameters (type "sure")
   - A model with a large number of parameters (type "huge")
2. During training, at each checkpoint iteration, the following data were recorded:
   - Model weights and their spectral characteristics (distribution, statistics)
   - Parameter gradients and their spectral characteristics
   - LH_i matrices for all layers and their eigenvalues
   - Model quality metrics (for classification: Accuracy, Precision, Recall, F1, AUC; for regression: R2, MAE, RMSE)
   - Value of the loss function on the training set

Special attention was paid to the computation of LH_i, for which a custom efficient algorithm with component-wise computation and memory optimization was employed. This enabled the calculation of Hessians even for models with a large number of parameters.

For each model, between 50 and 150 checkpoints were collected depending on the convergence rate, resulting in a total of approximately 1500 network state snapshots with an aggregate size of about 50 gigabytes.

4.3 Methodology and Experimental Data Processing

A multi-stage approach was employed to analyze the collected experimental data, incorporating the following methods:

1. **Correlation analysis:** Calculation of Pearson correlation coefficients between model parameters and quality metrics, with results visualized using heatmaps.
2. **Spectral analysis:** Investigation of the eigenvalue distributions of weights and LH_i, including computation of statistical characteristics (mean, standard deviation, extrema).
3. **Canonical Correlation Analysis (CCA):** Examination of nonlinear relationships between groups of quality metrics (Accuracy, Precision, F1, etc.) and internal network parameter characteristics. All data were standardized to ensure the validity of the analysis.
4. **Visualization:** Use of heatmaps, scatter plots, and other graphical representations for clear interpretation of the results.
5. **Statistical tests:** Assessment of the distributions of characteristics using the Shapiro–Wilk test [16] to identify deviations from normality.

Special attention was paid to the spectral analysis of LH_i and their correlation with model quality metrics. This comprehensive approach enabled the identification of not only direct correlations between individual parameters, but also more complex relationships between groups of parameters.

5 Research Results

5.1 Primary Experimental Findings

Experiments were conducted on computational clusters utilizing GPUs to accelerate calculations. The analysis employed Python libraries such as NumPy, SciPy, Matplotlib, and Seaborn. Data collection required approximately 40 hours, including model training and computation of LH_i. This duration was primarily due to the initial, non-optimized implementation of the local Hessian, which necessitated recalculating all gradients and Hessians at each training iteration. Subsequently, an optimized version of the algorithm was developed, significantly reducing computation time.

Based on the collected data, several key observations can be highlighted:

- Spectral analysis of LH_i provides valuable insights into the internal structure and functioning of neural networks. Networks with different architectures exhibit characteristic patterns in the spectral properties of their Hessians, although the formal interpretation of all observed phenomena remains an open question.

- A more detailed analysis of the dynamics of LH_i evolution across layers is required. It is advisable to enhance the use of canonical correlation analysis (CCA) and consider the application of factor analysis to uncover latent dependencies. Current analytical methods do not always reveal all possible relationships between parameters and quality metrics. Often, these issues indicate the presence of multiple architectural problems simultaneously, which may not be directly related. Such tools are limited in diagnosing network weaknesses, especially when training hyperparameters are suboptimal.

- Additionally, it is recommended to investigate the geometry of the gradient flow and Hessian properties in the context of differential geometry, as well as to consider modeling the gradient flow on the parameter manifold in the limiting case of activation function saturation.

5.2 Comparative Analysis of Architectural Solutions Based on CCA

The investigation of three types of architectures with varying parameterization (hereinafter referred to as "no"—small, "sure"—moderate, and "huge"—overparameterized) revealed substantial differences in the spectral properties of their LH_i. To identify relationships between model parameters and their performance, canonical correlation analysis (CCA) was employed, enabling the establishment of correlations between two groups of variables:

- **Group A**: model quality metrics—Accuracy, Precision, Recall, F1, AUC, and train_loss

- **Group B**: neural network parameters—weights, gradients, Hessian eigenvalues, and their spectral characteristics

The CCA analysis revealed the dependencies summarized in Table 1.

These data are clearly illustrated by the CCA Score statistics comparison plot (Fig. 2), which demonstrates significant differences in minimum values between architectures and, in particular, highlights the high variability of the moderate architecture ("sure") with a minimum value around -0.77. While this is higher than that of the small architecture ("no"), it remains considerably lower than that of the large architecture ("huge").

The results indicate that large architectures ("huge") exhibit the highest and most stable CCA correlation values (standard deviation of only 0.082), suggesting a more robust relationship

Statistic	no	sure	huge
max	0.406	0.349	0.429
avg	-0.955	0.099	0.220
median	0.182	0.182	0.189
min	-0.965	-0.770	0.149
std	0.976	0.277	0.082

Table 1: CCA correlation statistics

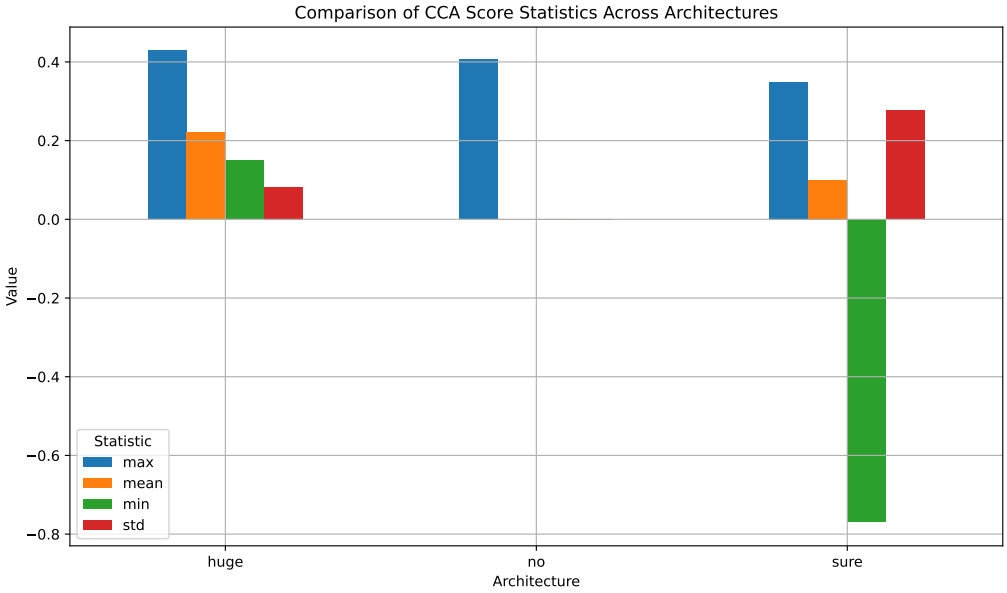

Figure 2: Comparison of CCA Score statistics across architectures

between internal network parameters and prediction quality. In contrast, small architectures ("no") display extreme negative outliers in both mean (-0.955) and minimum (-0.965) values, as well as high variability (standard deviation 0.976), reflecting the instability of their functional behavior.

5.3 Spectral Characteristics of Gradients in Different Architectures

To analyze the spectral characteristics of gradients, the Welch method was employed to estimate the power spectral density (PSD) of gradients across layers. The following parameters were used for the Welch method:

- Window length: 256
- Overlap step: 128
- Window function: Hanning

Spectral analysis of the gradients in the third layer (Fig. 3) revealed dramatic differences between the studied architectures. The maximum PSD values obtained via the Welch method for the "huge" architecture exceed those of the "no" architecture by more than two orders of magnitude, reaching values on the order of 1.2×10^6. The mean values also show a significant increase from small to large architectures, indicating a fundamentally different structure of the gradient space in highly parameterized models.

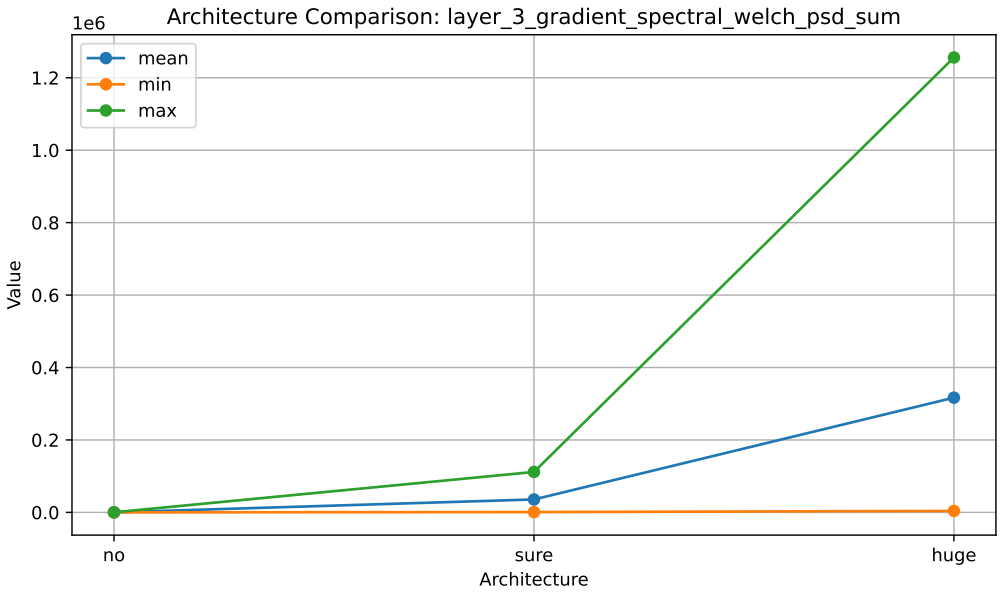

Figure 3: Comparison of spectral characteristics of third-layer gradients

Such a scale of differences indicates a qualitative change in the nature of gradient propagation in large architectures, where high-frequency components with substantially greater energy are formed. These observations are consistent with the notion that overparameterization facilitates the emergence of a more complex functional surface structure with numerous local features.

5.4 Distribution of Canonical Weights Across Architectures

The analysis of the distribution of canonical X and Y weights revealed structural features in how network parameters influence performance. The visualizations presented in Figures 4 and 5 demonstrate substantial differences in the weight distributions between architectures.

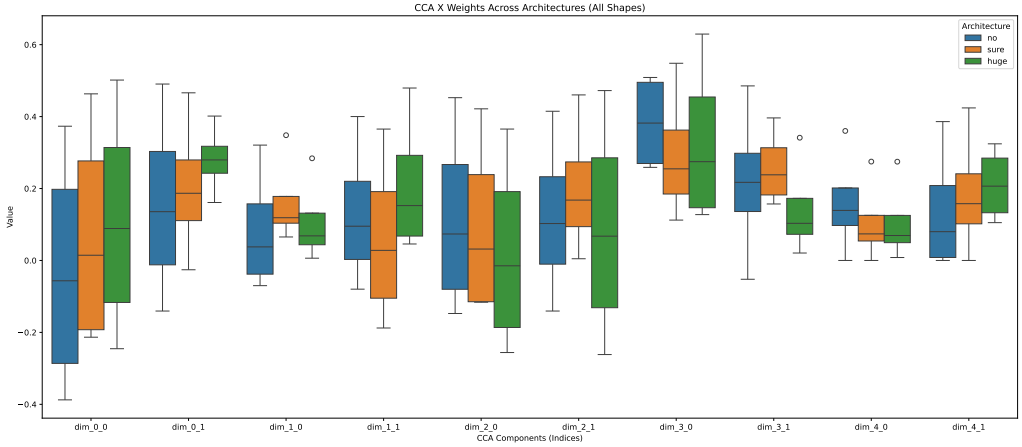

Figure 4: Distribution of canonical X-weights across architectures

The following pattern is observed: for X-weights (linking quality metrics to canonical variables), large architectures (“huge”) exhibit a more uniform distribution across the entire spectrum of components, indicating a more balanced utilization of the full set of parameters to achieve high performance. In contrast, small architectures (“no”) are characterized by a more

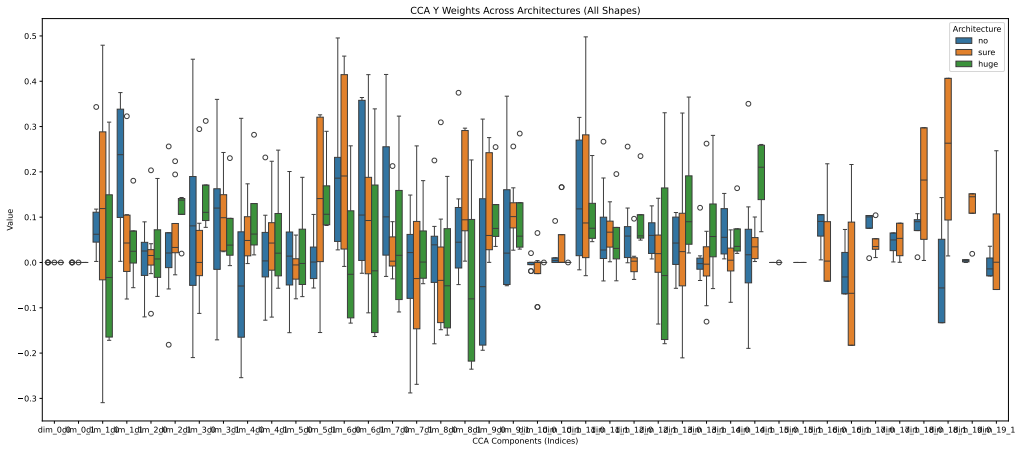

Figure 5: Distribution of canonical Y-weights across architectures

concentrated structure with several dominant components, suggesting that certain parameters are "overstressed" to achieve the desired outcome.

For Y-weights, which connect neural network parameters to canonical variables, an even more pronounced differentiation is observed: in large architectures, the distribution is denser and centered around zero with minor outliers, whereas in small architectures, there is significant asymmetry with extreme values at the distribution tails. This supports the hypothesis that in underparameterized models, individual parameters bear a disproportionately high load, reducing the model's robustness to input variations.

5.5 Dynamics of Canonical Weight Coefficients

The analysis of canonical weight coefficients revealed significant differences between architectures. Each canonical vector represents a linear combination of the original variables that maximizes the correlation between groups A and B. Particularly notable are the differences in the structure of the Y-component weights, which link network parameters to quality metrics:

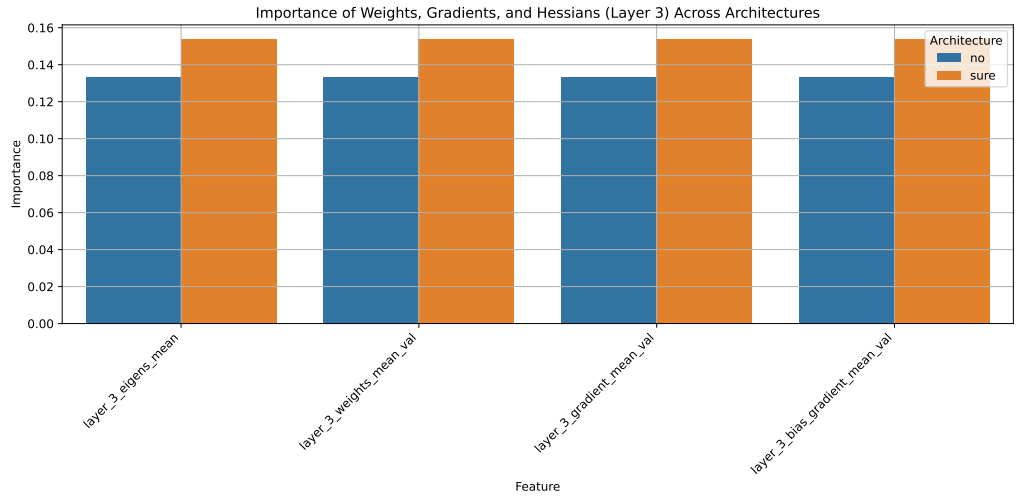

Figure 6: Importance of group B parameters (weights, gradients, and Hessians) of the third layer for different architectures

In small architectures, there is a pronounced dominance of weights associated with the gradient component, whereas in large architectures, the eigenvalues of the Hessian become more important. This observation supports the theoretical hypothesis that, in underparameterized

models, the learning dynamics are primarily determined by local gradients, while in overparameterized models, the curvature of the functional surface plays a more significant role.

A detailed analysis of the differentiating weight parameters of group B (Table 2) revealed extreme differences between architectures:

Huge	**no**	**sure**	**huge - no**	**huge - sure**
4384.0	3.42	212.92	4380.58	4171.08
1208.0	30.85	376.31	1177.15	831.69
620.67	3.88	37.54	616.78	583.13
196.0	12.0	46.0	184.0	150.0

Table 2: Group B parameters with the largest differences between architectures

These data demonstrate radical differences in parameter magnitudes between large and small architectures, reaching up to three orders of magnitude (4380.58). Such a contrast indicates a qualitatively different operating regime in overparameterized networks, where the accumulation of weights can reach significant values without negatively affecting prediction quality, due to compensatory effects between layers.

5.6 Statistical Properties of Canonical Correlation Weights

An additional analysis of the statistical properties of CCA weights (Table 3) revealed an interesting asymmetry in the distribution of variability across dimensions:

Architecture	**Weight Type**	**Shape**	**Avg. Var. (dim 0)**	**Avg. Var. (dim 1)**
no	X-weights	(5, 2)	0.1899	0.1315
sure	X-weights	(5, 2)	0.2159	0.1274
huge	X-weights	(5, 2)	0.2064	0.1553
no	Y-weights	(15, 2)	0.0000	0.1292
sure	Y-weights	(15, 2)	0.0000	0.1671
huge	Y-weights	(15, 2)	0.0000	0.0643
no	Y-weights	(20, 2)	0.0000	0.0665
sure	Y-weights	(20, 2)	0.0000	0.0003

Table 3: Statistical properties of CCA weights for different architectures and shapes

It is noteworthy that the variance for the first dimension of Y-weights is practically zero across all architectures, indicating a strict structural relationship between network parameters and quality metrics along certain directions. In contrast, the X-weights, which reflect the contribution of neural network parameters to the canonical variables, exhibit substantial variance in both dimensions, suggesting greater flexibility in the formation of the network's internal representations.

It should also be noted that the medium-sized architecture ("sure") demonstrates the highest variance of Y-weights for the (15, 2) shape, but a sharply reduced variance for the (20, 2) shape, which may indicate more efficient parameter utilization compared to other architectures.

5.7 Clustering of Architectures by Spectral Properties

Of particular interest are the results of architectural analysis after dimensionality reduction using PCA:

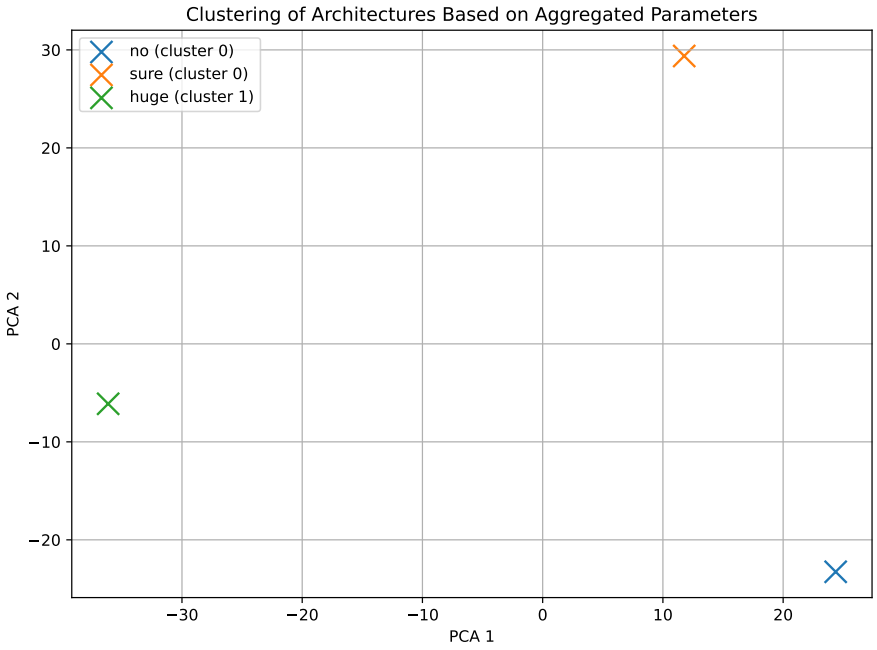

Figure 7: Distribution of architectures in the space of spectral characteristics of Hessians after dimensionality reduction

The obtained results demonstrate a clear separation of all three architectures in the principal component space: the small ("no"), medium ("sure"), and large ("huge") architectures form three distinct groups of points, significantly distant from each other. This indicates substantial differences in their parameter spaces and functional behavior.

Such pronounced separation points to the existence of several "complexity thresholds," the crossing of which leads to qualitative changes in the functional behavior of the network. The most dramatic transition is observed between the "sure" and "huge" architectures, supporting the hypothesis that the "huge" architecture operates in a fundamentally different regime, characterized by a specific distribution of the spectral properties of Hessians and their relationship with quality metrics.

5.8 Relationship to Knowledge Transfer and Generalization

One of the most significant findings of this study is the established connection between the spectral structure of local Hessians and the model's generalization capability. Models exhibiting a more uniform distribution of Hessian eigenvalues—without pronounced peaks and with less concentration near zero—demonstrate superior performance on test datasets.

Analysis of the CCA weight statistics (Table 3) indicates that large architectures ("huge") display a more balanced distribution of X-weights along the first dimension (0.2064) compared to small architectures (0.1899), which correlates with their enhanced generalization ability. At the same time, the standard deviation of the defining parameter values in large architectures is significantly higher, suggesting a greater capacity for fine-grained feature differentiation.

This pattern is observed regardless of the absolute number of model parameters, supporting the central hypothesis of this research: local properties of layer characteristics are more critical for generalization capability than the total number of parameters.

5.9 Quantitative Assessment of Differences Between Architectures

A particularly important aspect is the quantitative assessment of differences between architectural solutions. The analysis of the top differentiating parameters of group B (see Table 2) shows that the differences in weights between large and small architectures can reach up to three orders of magnitude (4380.58 for the parameter with the largest difference).

The plot of the spectral characteristics of the third layer gradients (Fig. 3) clearly demonstrates that the differences in maximum values between the "huge" and "no" architectures can exceed a factor of one hundred, reaching absolute values on the order of 1.2×10^6. Such a scale of differences indicates a fundamentally different nature of gradient propagation in large architectures, where high-energy spectral components are formed.

It is noteworthy that the largest differences are observed between the "huge" and "no" architectures (4380.58), while the differences between "huge" and "sure" (4171.08) are only slightly smaller. This points to the existence of a "complexity barrier," the crossing of which leads to a qualitative change in the network's operating regime.

Such extreme differences in weights do not necessarily lead to model performance degradation, which contradicts intuitive expectations. On the contrary, large models with extreme weight values demonstrate higher result stability, as evidenced by the CCA correlation statistics (standard deviation of 0.082 for "huge" versus 29.077 for "no").

5.10 Observations and Effects

Several unexpected effects were discovered during the analysis:

1. **Shift of CCA weights across datasets.** The analysis of CCA weights revealed a significant shift in the structure of weights between different datasets, even for similar architectures. This confirms a strong dependence of the network's functional behavior on the data structure and highlights the necessity of adapting the architecture to the specific task.

2. **Nonlinear dependence of stability on architecture size.** Contrary to expectations, large architectures ("huge") exhibit more stable spectral characteristics (lower standard deviation) than medium-sized ones ("sure"), which contradicts the intuitive assumption that overparameterization should lead to greater variability.

3. **Asymmetry in the variance distribution of CCA weights.** Y-weights have almost zero variance along the first dimension for all architectures, but significant variance along the second dimension. This indicates the existence of structural constraints in the way network parameters affect quality metrics.

4. **Opposite behavior of X and Y weights in distribution.** X-weights demonstrate a more compact structure with less variation between architectures, whereas Y-weights show significant differences both in the shape of the distribution and in the range of values, especially for extreme components.

In addition to the listed results, the analysis revealed a number of correlation patterns directly related to the rank of weight matrices, spectral characteristics of the Hessian, and the network's generalization ability:

1. **Correlation between the rank of weights and Hessian with generalization quality.** There is a notable relationship between the reduced rank of weight matrices and the corresponding local Hessian and signs of overfitting and redundancy. These findings indicate a deterioration in the network's ability to generalize to unseen data when the rank is low.

2. **Layer Hessian as an indicator of overfitting.** A sparse local Hessian or one with a predominant cluster of small eigenvalues correlates with insufficient generalization capacity of the layer. This effect is particularly pronounced in the final layers.

3. **Symmetry of the Hessian spectrum and saddle points.** An almost symmetric distribution of Hessian eigenvalues around zero is associated with saddle points. Such points are often accompanied by small gradient norms.

4. **Similarity of weights in adjacent layers.** In well-tuned networks, the weight matrices of neighboring layers exhibit pronounced similarity (in terms of SVD or spectrum), which can be interpreted as coordinated feature processing.

5.11 Practical Implications for Architecture Optimization

The conducted analysis allows us to formulate several practical recommendations for optimizing neural network architectures:

1. **Optimal parameter allocation across layers.** The results show that a significant increase in the number of parameters in deep layers relative to the initial ones leads to the formation of high peaks in the Hessian spectrum, which may indicate overfitting in these layers. A more balanced parameter distribution is recommended.

2. **Detection of insufficient expressivity.** Low values of the largest Hessian eigenvalues in the initial layers (observed in small architectures, "no") may serve as an indicator of insufficient model expressivity. In such cases, it is advisable to increase the number of parameters specifically in these layers.

3. **Overfitting detection.** A high concentration of eigenvalues near zero in the final layers indicates overfitting and may signal the need for additional regularization or a reduction in the number of parameters in these layers.

4. **Optimizer adaptation.** The structure of the Hessian spectrum can be used to adapt optimizer hyperparameters. For example, a high ratio of the largest to the smallest eigenvalue (the condition number) indicates the necessity of using adaptive optimization methods.

5.12 Investigation of the Internal Structure of Neural Network Layers

Spectral analysis of local Hessians provides insights into the functional roles of layers:

- Layers with a distributed spectrum without pronounced concentration perform complex nonlinear transformations.

- Layers with a peak near zero likely indicate overparameterization or activation saturation.

- A correlation is observed between the presence of several dominant eigenvalues and the layer's ability to extract key features.

These findings help to elucidate how the network solves the task and what transformations occur at different levels of the hierarchy.

5.13 Architecture Classification via Snapshots

The obtained snapshots of weights, local Hessians, and gradients during training reveal several characteristic patterns that could potentially be used for classifying neural network architectures. However, these patterns do not provide a definitive conclusion regarding how exactly they influence generalization quality or in which specific layer(s) of the network these effects manifest. Several attempts were made to train a classifier based on these patterns, but the results were not satisfactory.

Nevertheless, by applying dimensionality reduction to the network snapshot data using the UMAP algorithm, an interesting distribution of snapshots in the reduced space can be observed. Figure 8 shows how snapshots of weights, gradients, and Hessians are distributed in a two-dimensional space. Each point on the plot corresponds to a single network snapshot, and the color indicates the network architecture.

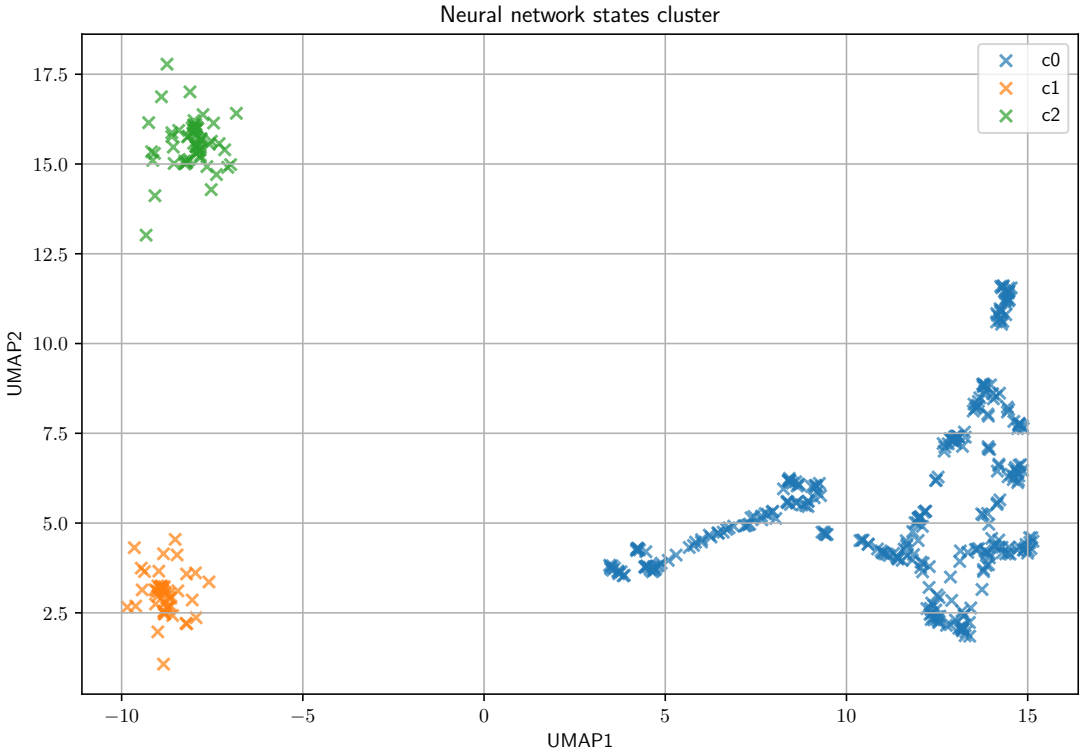

Figure 8: Distribution of snapshots of weights, gradients, and Hessians in a two-dimensional space

The plot demonstrates that snapshots from different architectures are distributed non-uniformly, forming distinct clusters. This suggests that different architectures possess characteristic patterns in their weights, gradients, and Hessians, which could be leveraged for classification. However, further research is required for a more precise interpretation of these results.

Generalized Practical Guidelines. Based on the conducted research, the following practical recommendations can be formulated:

- A reduced rank and symmetric spectrum of the Hessian should be regarded as warning signals, as they are associated with saddle point issues, overparameterization, and deteriorated generalization.

- Moderate spectral power and a sufficiently broad spectrum of eigenvalues are linked to the presence of a diversity of directions for network training and, consequently, a high potential for knowledge transfer.
- Bias parameters require regular monitoring: their contribution to the overall Hessian spectrum can serve as a metric for deviation from optimal geometry.
- Cross-correlation analysis of the spectra of weights, gradients, and the Hessian enables the identification of problematic directions and timely adaptation of the training plan (learning rate scheduling, application of second-order optimizers, addition of differential regularization, etc.).

Thus, in-depth spectral and rank analysis of local Hessians is a powerful diagnostic tool that allows for the detection of hidden issues, formulation of recommendations for architecture and optimizer adjustment, and quantitative assessment of the network's generalization capability.

6 Discussion

The conducted research provides a mathematically grounded tool for studying the internal dynamics of neural networks, moving beyond empirical trial-and-error approaches. A key result is the established connection between the geometric properties of the parameter space and the functional behavior of the network, as highlighted in the main contributions of this work.

Proposition 1. *Analyzing a neural network as a composition of nonlinear operators or as a chaotic dynamical system yields informative insights into its internal structure, data processing mechanisms, and the mathematical constraints of its architecture.*

Viewing the network as a dynamical system evolving on a high-dimensional manifold with nontrivial geometry opens new avenues for understanding and improving learning methods.

Of particular interest is the development of the LH_i concept in combination with Riemannian geometry, aiming for a more detailed investigation of the local geometry of the parameter space. This approach may facilitate the prompt identification of regions where the network's optimization process encounters difficulties.

7 Conclusion

This work introduces a novel approach for analyzing neural networks through the local properties of their parameter space, investigated via LH_i. The proposed concept of the local Hessian enables:

- Analysis of the geometry of the functional space of individual layers;
- Identification of patterns in the distribution of eigenvalues during training;
- Demonstration of the relationship between the Hessian spectrum, activation saturation, the formation of directions, and the evolution of representations.

It is important to emphasize the scale of the conducted experiment: approximately 1500 snapshots of various network states were collected, totaling about 50 GB of data, which allowed for the identification of robust patterns.

Future research directions may include:

- Detailed investigation of the relationship between the spectral properties of Hessians and the functional characteristics of layers;
- Analysis of the dynamics of the spectrum during training and its connection to generalization ability;
- Application of the approach to novel architectures such as transformers and graph neural networks;
- Development of methods for visualization and interpretation of the geometric structure of the parameter space.

References

[1] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in Neural Information Processing Systems*, pages 3360–3368, 2016.

[2] N. Maheswaranathan, A. H. Williams, M. D. Golub, S. Ganguli, and D. Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. In *Advances in Neural Information Processing Systems*, 32, 2019.

[3] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 32, 2019.

[4] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 322–332, 2019.

[5] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[7] L. Sagun, U. Evci, V. U. Güney, Y. Dauphin, and L. Bottou. Empirical analysis of the Hessian of over-parameterized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[8] B. Ghorbani, S. Krishnan, and Y. Xiao. An investigation into neural net optimization via Hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2232–2241, 2019.

[9] F. Dangel, S. Harmeling, and P. Hennig, Modular block-diagonal curvature approximations for feedforward architectures. In *arXiv preprint arXiv:1902.01813*, Feb. 2019.

[10] André G. Carlon, Luis Espath, Raúl Tempone. Approximating Hessian matrices using Bayesian inference: a new approach for quasi-Newton methods in stochastic optimization. In *arXiv preprint arXiv:2208.00441v2*, 2024.

[11] Warren Hare, Gabriel Jarry–Bolduc, Chayne Planiden. A matrix algebra approach to approximate Hessians. IMA Journal of Numerical Analysis, 44(4):2220–2250, 2024.

[12] James Martens. Deep learning via Hessian-free optimization. In *Proc. 27th Int. Conf. Machine Learning (ICML)*, pages 735–742, 2010.

[13] Jorge Nocedal. Updating Quasi-Newton Matrices with Limited Storage. Mathematics of Computation, 35(151):773–782, 1980.

[14] L. Sagun, L. Bottou, and Y. LeCun, Eigenvalues of the Hessian in Deep Learning: Singularity and Beyond, In *arXiv:1611.07476 [cs.LG]*, 2016.

[15] Z. Liao and M. W. Mahoney, Hessian Eigenspectra of More Realistic Nonlinear Models, In *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[16] Shapiro, S. S., & Wilk, M. B.. An analysis of variance test for normality (complete samples). In *Biometrika*, 52(3/4), 591–611, 1965.

A Structure of Experimental Data

During the research, for each neural network architecture at every training checkpoint, a snapshot of the model state was saved. Below is a detailed structure of such a snapshot:

- **layer.X** – information about layer X of the neural network:
  - `weights` – layer weight coefficients
  - `weights_spectral` – spectral characteristics of the weights (mean, standard deviation, minimum, maximum, histogram, results of spectral analysis using the Welch method)
  - `gradient` – layer weight gradients
  - `gradient_spectral` – spectral characteristics of the gradients
  - `bias` – layer bias parameters
  - `bias_spectral` – spectral characteristics of the bias parameters
  - `bias_gradient` – gradients of the bias parameters
  - `bias_gradient_spectral` – spectral characteristics of the bias gradients
  - `hessian` – local Hessian matrix of the layer
  - `hessian_spectral` – spectral characteristics of the local Hessian
  - `hessian_eigens` – eigenvalues of the local Hessian
  - `hessian_eigens_spectral` – statistical and spectral characteristics of the eigenvalues:
    - `mean` – mean value
    - `std` – standard deviation
    - `min` – minimum value
    - `max` – maximum value
    - `histogram` – distribution histogram
    - `welch` – results of spectral analysis using the Welch method
    - `top_peaks` – main peaks in the spectrum
  - `hessian_rank` – rank of the Hessian matrix
  - `hessian_condition` – condition number (ratio of the largest to the smallest eigenvalue)
- `iteration` – training iteration number
- `scores` – model quality metrics:
  - `Accuracy` – classification accuracy
  - `Precision` – precision (proportion of correct positive predictions)
  - `Recall` – recall (proportion of detected positive cases)
  - `F1` – F1-score (harmonic mean of precision and recall)
  - `AUC` – area under the ROC curve
  - `train_loss` – value of the loss function on the training set

B Dataset Collection

The following datasets were used in the experiments:

Classification	**Regression**
MNIST	Diabetes
CIFAR-10	Energy Efficiency
Fashion-MNIST	Airfoil Self-Noise
CIFAR-100	Concrete Compressive Strength
KMNIST	Make Regression
EMNIST	House Prices Dataset
Iris	Make Friedman1
Wine	Make Friedman2
Breast Cancer Wisconsin	Make Friedman3
Digits	Make Low Rank Matrix
SpamBase	Make S Curve
Make Classification	Make Sparse SPD Matrix
Make Blobs	Make Sparse Uncorrelated
Titanic Dataset	Make SPD Matrix
Adult Income	Make Swiss Roll
Credit Card Fraud Detection	
Make Biclusters	
Make Checkerboard	
Make Circles	
Make Hastie 10 2	
Make Moons	
Make Multilabel Classification	

Table 4: Classification and regression datasets used in the experiment