
# Sub-networks and Spectral Anisotropy in Deep Neural Networks

Master's thesis in Complex Adaptive Systems

HANWEN GE

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

# Sub-networks and Spectral Anisotropy in Deep Neural Networks

HANWEN GE


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Sub-networks and Spectral Anisotropy in Deep Neural Networks
HANWEN GE



Supervisor: Jan E. Gerken, Department of Mathematical Sciences
Examiner: Johan Jonasson, Department of Mathematical Sciences

Master's Thesis 2025
Department of Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Sub-networks and Spectral Anisotropy in Deep Neural Networks
HANWEN GE
Department of Mathematical Sciences
Chalmers University of Technology

# Abstract

Deep neural networks (DNNs) have achieved remarkable success across diverse domains, yet the fundamental reasons behind their efficacy and ability to generalize remain elusive. This thesis examines how over-parameterized DNNs learn and generalize by investigating two interconnected phenomena: the emergence of sparse, critical sub-networks (aligned with the Lottery Ticket Hypothesis) and the structural symmetry-breaking. Additionally, we explore the geometric structure of the parameter space, with a particular focus on the anisotropy of the Fisher Information Matrix (FIM) spectrum.

We demonstrate that different layers in a deep network exhibit varying degrees of symmetry breaking, which we link to the presence of sub-networks that encapsulate the model's core representational capacity. Using two distinct criteria—magnitude-based and change-based—we identify critical sub-networks and show that, despite the over-parameterization of DNNs, these sparse sub-networks play a central role in achieving high performance.

By analyzing the spectrum of the FIM, we reveal that DNNs evolve along a limited number of dominant eigendirections, spanning a subspace where training dynamics converge. This finding highlights an intrinsic anisotropy in the parameter manifold. Furthermore, we investigate how this anisotropy correlates with the emergence of sub-networks and the internal structure of the subspace.

Overall, this thesis provides a novel perspective on the roles of implicit regularization, loss landscape geometry, and sparse substructures in modern deep neural networks, offering insights into the geometric nature of DNNs.

Keywords: Deep Neural Networks, Information Geometry, Generalization, Spectral Analysis, Lottery Ticket Hypothesis.

# Acknowledgements

This work, emerging in its current form, is the tangible expression of a long and thoughtful journey through the realms of deep neural network theory—a journey spanning a year and a half of exploration, discovery, and quiet wonder.

I owe immeasurable gratitude to my supervisor, Jan E. Gerken, whose insight lit the path when I first envisioned a study centered on the evolution of parameter spaces. His gentle suggestion to embrace Fisher Information Geometry opened up entirely new vistas in understanding deep neural networks. Our countless meetings—filled with rigorous theoretical discussions and moments of scholarly reflection—not only nurtured my academic inquiry but also refined the very way I see and express my research. His generous support, be it in granting creative freedom or guiding me through intricate administrative details, has indelibly shaped this work.

I am equally indebted to my examiner, Johan Jonasson, whose brief but impactful early collaboration gave me the confidence to propose this master's thesis independently. His steadfast commitment and compassionate guidance provided not only academic wisdom but also the personal support that sustained me through every challenge.

My heartfelt thanks extend to Francesca Mignacco, my early external supervisor, whose numerous insightful pointers on deep learning theory broadened my understanding of implicit regularization, learning dynamics, training regimes, and the applications of dynamical systems. The literature she introduced and our spirited conversations—even when separated by thousands of miles from New York—infused my work with profound depth and clarity.

A significant chapter of my life unfolded at the Physics Department of the University of Gothenburg. I remain deeply grateful to my project advisor, Kristian Gustavsson, whose confidence in my ability to embark on an autonomous project spurred me to dare explore uncharted territories. I cherish the lively, varied discussions with my friend Ludvig Storm, whose wisdom and passion ignited countless moments of inspiration. I also wish to thank Linus Sundberg, Mathias Samuelsson, Jingran Qiu, Frida Brogren, Michael Quin, Ehsan Ghane, Petter Uvdal, Zhe Han, Enrique Rozas Garcia, Grigory Sarnitsky, Navid Mousavi, and many others—each conversation, academic debate, and friendly smile has woven itself into the fabric of my intellectual journey. My gratitude also goes to my Director, Mats Granath, for his unwavering support.

During the fleeting yet transformative days I spent in Neuchâtel, Switzerland, I encountered a form of scholarly inspiration unlike any other. Under the free-spirited guidance of Christos Dimitrakakis—and throughout the memorable walks in the mountains, shared lunches, and quiet moments with Andreas Athanasopoulos and Victor Villin—I absorbed not only the beauty of a vibrant summer but also a renewed academic outlook that continues to shape my path.

And then there are my many friends, now scattered across the globe, without whose light not only this work but my very life would surely be diminished. Estéban Antoine Nocet-Binois, I long for the profound, meandering stories we once wove together and the shared passion for academia that enriched our hearts. The countless days and nights spent with Napat Bhaholpolbhayuhasena in Denmark, Hungary, Serbia, China, and of course Sweden remain etched in my memory—each moment illuminated by his irrepressible optimism, unwavering enthusiasm, and steadfast persistence. I also treasure the warmth and vibrancy of my dear friends Siwakorn Sanchuensakul, Haik-David Avetian, and Marko Arnautovic, whose companionship banishes loneliness and fills every space with light. Rundong Zhou and I were blessed not only by the fine wine we savored and the sumptuous meals we enjoyed but also by the spark of insight ignited by his thoughtful views on physics. And to my dear friend Sameer Jathavedan, the shared praises and critiques, the joys intermingled with frustrations, and the absurdities blended with laughter have together revealed the true beauty of our world. Sharing an office and lively gossip with Edoardo Maria Manoni remains one of my fondest memories, as does the ever-bright optimism and attentive listening of Vasiliki Kostara. There are so many more hearts and stories to recount—from my loyal friend Sheik Meeran Rasheed Abdul Rahuman and my spirited defense opponent Cyrene Howland, to the two loving couples who enliven every Christmas and New Year dinner, Hon Lam Cheung with Xiaotian Zhang and Yuxin Fang with Noel Vincent. I also thank Nils Müller—a friend of remarkable integrity whom I met later but whose understanding of me remains profound—as well as every friend I have made over the past 26 years, and the countless strangers whose paths have crossed mine. To each of you, I offer my most heartfelt thanks.

Finally, my deepest gratitude is reserved for my family—my brother, sister-in-law, and most especially my parents—whose boundless love and the richness of my childhood continue to nurture my imagination and sustain my pursuit of knowledge.

The computation were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) and the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE), partially funded by the Swedish Research Council through grant agreement no. 2022-06725 and no. 2018-05973.

HANWEN GE, Gothenburg, April 2025

# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

| | |
|---|---|
| DNNs | Deep Neural Networks |
| MLPs | Multilayer Perceptrons |
| FIM | Fisher Information Matrix |
| NTK | Neural Tangent Kernel |
| SGD | Stochastic Gradient Descent |
| ReLU | Rectified Linear Unit |
| MSE | Mean Squared Error |
| LTH | Lottery Ticket Hypothesis |
| SDE | Stochastic Differential Equation |
| ODE | Ordinary Differential Equation |
| KL | Kullback-Leibler (Divergence) |
| erank | Effective Rank |

# Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

## Indices

| | |
|---|---|
| $i, j$ | Indices for neurons in a layer |
| $l$ | Index for layers in the network |
| $t$ | Index for iterations during training |
| $n$ | Index for data samples in a batch or dataset |

## Sets

| | |
|---|---|
| $\mathcal{L}$ | Set of layers in the network |
| $\mathcal{N}$ | Set of neurons in a layer |
| $\mathcal{S}$ | Set of data points in a mini-batch |
| $\mathcal{D}$ | Full dataset |

## Parameters

| | |
|---|---|
| $W^{(l)}$ | Weight matrix of layer $l$; dimensions are $M^{(l)} \times M^{(l-1)}$ (number of neurons in layer $l$ by number of neurons in layer $l-1$) |
| $b^{(l)}$ | Bias vector of layer $l$; dimensions are $M^{(l)} \times 1$ (one bias term per neuron in layer $l$) |
| $\phi$ | Activation function |
| $L(y, \hat{y})$ | Loss function |
| $\eta$ | Learning rate |
| $N$ | Total number of training samples |

| | |
|---|---|
| $M^{(l)}$ | Number of neurons in layer $l$ |
| $T$ | Total number of training iterations |

## Variables

| | |
|---|---|
| $x$ | Input feature vector |
| $y$ | Ground truth output |
| $\hat{y}$ | Predicted output of the network |
| $u_i^{(l)}$ | Pre-activation value of neuron $i$ in layer $l$ |
| $h_i^{(l)}$ | Post-activation value of neuron $i$ in layer $l$ |
| $g_i^{(l)}$ | Gradient of the loss function w.r.t. parameter $i$ in layer $l$ |
| $J$ | Jacobian matrix of the network output w.r.t. parameters |
| $\lambda$ | Eigenvalue of a matrix (e.g., FIM, NTK) |

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

*"More is different."* This phrase, the title of Nobel Physics Prize laureate Philip W. Anderson's seminal paper [4], encapsulates a foundational principle in complex systems theory. Complex systems [5] are characterized by the interactions of numerous components, yielding emergent, often nonlinear behaviors that cannot be inferred from studying individual elements in isolation. As a multidisciplinary field, complex systems theory encompasses diverse domains, including the human brain, biological organisms, infrastructure networks (such as power grids, transportation, and communication systems), complex software systems, ecosystems, the Earth's climate, and, potentially, the universe itself. Modern deep neural networks (DNNs) [35], initially inspired by the architecture of the human nervous system, embody this principle, exemplifying how the collective behavior of simple computational units can lead to remarkable emergent intelligence.

DNNs comprise millions, or even billions, of artificial neurons interconnected via weighted links. Each neuron executes simple nonlinear computations, yet the network's overall behavior emerges from intricate interactions across layers. The hierarchical structure of DNNs enables them to learn intricate representations not explicitly encoded in their training data, reflecting an emergent property akin to that observed in other complex systems. Through optimization processes such as backpropagation [6], DNNs dynamically adjust their parameters to minimize errors and adapt to the structure of input data, further reinforcing their complex nature.

Beyond the complexity of DNNs themselves, the natural data [12, 13] they process exhibits significant complexity. Natural data refers to information generated by natural processes in the physical world, including, but not limited to, images, speech, text, and biological structures. Over the past decade, DNNs have demonstrated groundbreaking performance in diverse fields, including image classification [14], video generation [15], large language models [16], and biological applications such as protein and drug design [17, 18]. Deep neural networks have become indispensable tools for uncovering patterns in complex natural data.

A fundamental illustration of data complexity arises from the sheer combinatorial explosion of possible configurations [20]. Consider a grayscale image of $n$ pixels: the total number of possible images is $2^n$, an exponentially large space. Assigning labels to each image (e.g., binary classification) further expands this space to $2^{2^n}$, an potential astronomical number. Even for a trivial 9-pixel black-and-white image ($n = 9$), the number of possible labeled datasets surpasses the estimated number

of atoms in the observable universe [19]. However, as noted by Lin et al. [92], the laws of physics are such that the data sets we care about for machine learning are drawn from an exponentially tiny fraction of all imaginable data sets. For instance, images are composed of objects, which in turn consist of smaller components organized in a structured manner. Humans naturally recognize patterns at multiple scales, a capability shaped by both experiential learning and evolutionary priors. This hierarchical perception enables humans to categorize images not by memorizing all possible permutations but by identifying meaningful features such as shapes, textures, and object relationships. In deep learning, the experiential learning and evolutionary priors manifest as feature learning (or representation learning) [100] and inductive bias [101]—mechanisms by which DNNs leverage hierarchical representations to capture essential structures in data.

Despite the advancements in theoretical works over the past decade, the underlying mechanism of deep neural networks (DNNs) remains elusive. These works, originating from diverse perspectives and theoretical backgrounds, aim to provide a comprehensive understanding of DNNs. For instance, Robert et al. [70] employed the renormalization group to establish an effective theory for DNNs. In a more complex systems perspective: Bahri et al. [73] extended statistical mechanics to elucidate the principles governing DNNs, incorporating Random Matrix theory (RMT) [71], Mean Field Theory [72], Phase Transitions and Chaos [21], Nonequilibrium Statistical Mechanics [28], Spin Glass Theory [29], and Free Probability Theory [30]. Storm et al. [86] identified structural patterns in DNNs that contribute to their robustness, in conjunction with Lagrangian coherent structures (LCSs) derived from dynamical systems. Beyond these illustrative cases, a multitude of empirical and theoretical investigations continue to probe the enigmatic nature of deep neural networks.

One of the central questions in deep learning is the generalization [7] of deep neural networks (DNNs). While we discussed the astronomical scale of natural data above, one might assume that DNNs with millions or billions of parameters are actually small. However, these two concepts are not directly related. What we emphasize when discussing natural datasets is that the inherent structure and features of natural data contribute to the performance of DNNs. Nevertheless, these models are often over-parameterized, meaning the number of parameters exceeds the sample size of the training data. In classical machine learning theory [94], over-parameterized models tend to overfit, memorizing training samples rather than learning generalizable patterns. This raises the fundamental question: *How do deep neural networks trained via gradient-based optimization achieve remarkable generalization to unseen data?* Resolving this paradox—where excessive over-parameterization paradoxically enhances generalization—remains a major open problem in deep learning theory [96, 7].

This thesis investigates structural symmetry breaking [1] as a central mechanism underlying generalization in deep learning. In parallel, it employs the Fisher Information Matrix (FIM) [104] as an analytical tool to explore the geometric structure of the loss landscape. While this approach is similar to the energy landscape analysis

of Bahri et al. [73], our work specifically emphasizes the interplay between sparsity and the geometric structure of the loss landscape in DNNs. Structural symmetry breaking refers to the phenomenon wherein different layers of a neural network exhibit distinct functional importance, with some layers being critical to performance while others remain redundant. A key experimental approach involves layer-wise re-initialization: resetting specific trained layers to their initial values and evaluating the resulting performance degradation. Layers whose re-initialization significantly impairs performance are deemed critical. We extend this method to the parameter level reveals that only a sparse subnetwork—rather than the entire dense network—is responsible for structural symmetry breaking. This finding aligns with the Lottery Ticket Hypothesis, which posits that within a randomly initialized dense network, sub-networks exist that, when trained in isolation, achieve comparable accuracy to the original model.

The Fisher Information Matrix (FIM) serves as the metric tensor that encapsulates the geometric structure of the parameter space. Through further analysis employing the FIM, it becomes evident that the effective dimensionality of DNN training is remarkably low, suggesting that learning predominantly transpires along a restricted number of parameter-space directions. This is evident in the anisotropic nature of the FIM spectrum. Intriguingly, we demonstrate that these anisotropic eigendirections correspond to the discovered sub-networks—when these sub-networks are removed, the anisotropic structure of the FIM disappears. This suggests a deep connection between spectral anisotropy in FIM and sub-network emergence. By establishing this link, this work lays the foundation for further exploration of the geometric structure of the parameter space in relation to deep neural network sparsity.

## Outline of the Thesis

In Chapter 2, we explore the foundations of DNNs, focusing on generalization. We discuss how classical measures often prove inadequate for modern over-parameterized networks, prompting new lines of theoretical exploration such as PAC-Bayes [40], the interplay between Stochastic Gradient Descent (SGD) algorithm and implicit regularization, and the role of the loss landscape geometry. We highlight that although gradient-based methods like SGD can favor flatter minima in many cases (the flatness bias hypothesis), the exact extent to which this phenomenon explains generalization is still a matter of debate [41, 42]. We also discussed the motivation behind exploring the geometry of the loss landscape and parameter space.

Chapter 3 narrows in on the **sub-networks** that exist within well-trained DNNs. We examine how iterative magnitude pruning (IMP) and an alternative parameter change-based criterion consistently identify smaller sets of critical parameters. We then illustrate how structural symmetry breaking arises: some parameters in certain layers show higher sensitivity to re-initialization, reflecting a non-uniform distribution of critical parameters across layers [1]. Our results suggest that these sub-

networks constitute a hidden low-complexity sub-network of the model, explaining why performance remains robust even when vast portions of parameters are pruned or re-initialized.

In chapter 4, it turns to a spectral analysis of deep neural networks from the standpoint of **information geometry**. We employ the Fisher Information Matrix (FIM) to analyze the unfolding of training trajectories along the dominant directions in its eigenspace. Crucially, we show that only a small portion of the parameter space—those directions associated with large eigenvalues—significantly drives the model's evolution during training [74]. We then connect the spectral anisotropy to the sub-networks identified in Chapter 3, revealing deeper geometric reasons for why these sparse substructures capture most of the model's capacity and shape its generalization behavior.

Finally, throughout the thesis, we focus on how symmetry breaking, sub-networks, and the FIM spectrum interact in practice. While the results do not single-handedly solve the mystery of generalization, they highlight a plausible story: Overparameterized deep neural networks (DNNs) are guided by the geometry of their parameter space toward sparse, critically important substructures, resulting in effective and robust generalization.

This thesis is inspired by the author's previous project Structural Symmetry Breaking: An Empirical Study on Re-initialization of Hidden Layers in Deep Neural Networks (Physics, project - FUF060), and this project is partially covered in this thesis.

# 2

# Deep Neural Networks

In this chapter, the foundational concepts of deep neural networks will be explored, including their architecture, initialization, optimization, and generalization. These elements collectively underpin the remarkable performance of modern deep learning systems.

We focus on a broad range of topics related to deep learning generalization, including implicit regularization induced by gradient-based optimization methods, the role of stochastic gradient descent algorithms in promoting generalization, and their limitations in explaining generalization in over-parameterized deep neural networks. Furthermore, we propose that integrating insights from the geometry of the loss landscape structure can offer new perspectives for advancing the understanding of generalization.

## 2.1 Architecture: Multilayer Perceptrons

Multilayer Perceptrons (MLPs) represent one of the earliest and most fundamental architectures in the evolution of deep learning [35]. This thesis primarily focuses on MLPs as a foundational model, serving as a basis for understanding the principles that drive many advanced variants of deep neural networks.

An MLP is a quintessential example of a feedforward neural network, composed of multiple layers of fully connected artificial neurons. Each neuron performs a linear transformation of its inputs, followed by the application of a nonlinear activation function. The input layer receives a vector $\mathbf{x} \in \mathbb{R}^{M_0}$ derived from a preprocessed dataset. Subsequently, each layer computes its outputs through a combination of linear and nonlinear operations, enabling MLPs to approximate highly complex mappings between input and output spaces, as illustrated in Figure 2.1.

The mathematical formulation of an MLP begins with the computation of the pre-activation for the $i$-th neuron in the $l$-th layer, defined as:

$$o_i^{(l)} = \sum_{j=1}^{M^{(l-1)}} w_{ij}^{(l)} h_j^{(l-1)} + b_i^{(l)}, \tag{2.1}$$

where $M^{(l-1)}$ represents the number of neurons in the $(l-1)$-th layer, $w_{ij}^{(l)}$ is the weight connecting the $j$-th neuron in the $(l-1)$-th layer to the $i$-th neuron in the

$l$-th layer, $h_j^{(l-1)}$ denotes the activation of the $j$-th neuron in the $(l-1)$-th layer, and $b_i^{(l)}$ is the bias associated with the $i$-th neuron in the $l$-th layer.

The output of each neuron, commonly referred to as its activation, is obtained by applying a nonlinear activation function $\Phi(\cdot)$ to the pre-activation value:

$$h_i^{(l)} = \Phi(o_i^{(l)}). \tag{2.2}$$

In the broader context of deep neural networks (DNNs), the objective is to learn a function $f_\theta : \mathbb{R}^{M_0} \to \mathbb{R}^{M_L}$, parameterized by the network's weights and biases, collectively denoted as the network parameters $\theta \in \mathbb{R}^P$, P is the number of parameters. The goal is to ensure that the network's output $f_\theta(\mathbf{x})$ closely approximates the target label $\mathbf{z}$ for a given input $\mathbf{x}$. The functional form of an MLP can be expressed as:

$$f_\theta(\mathbf{x}) = \Phi^{(L)}\left(W^{(L)}\Phi^{(L-1)}\left(W^{(L-1)}\cdots\Phi^{(1)}\left(W^{(1)}\mathbf{x}+B^{(1)}\right)+B^{(2)}\cdots\right)+B^{(L-1)}\right)+B^{(L)}, \tag{2.3}$$

where $\Phi^{(l)}$ represents the activations of the $l$-th layer, $W^{(l)}$ is the weight matrix, and $B^{(l)}$ is the bias vector of the $l$-th layer.

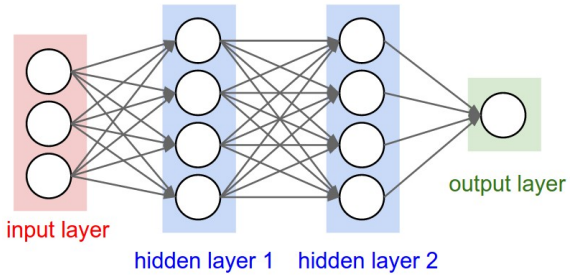



**Figure 2.1:** A fully connected feedforward neural network (MLP) with 2 hidden layers, each containing 4 neurons. Source: CS231n: Deep Learning for Computer Vision, Stanford University [31].

Two examples of activation functions are [75]:

- **Rectified Linear Unit (ReLU)**:

  $$\mathrm{ReLU}(x) = \max(0, x).$$

  ReLU introduces nonlinearity to the model while being computationally efficient. It helps mitigate the vanishing gradient problem, making it a popular choice for hidden layers[75].
- **Softmax Function**:

  $$\mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}.$$

  Typically used in the output layer for multi-class classification problems, the softmax function converts logits into probabilities.

Consequently, the representation produced by any given layer is recursively dependent on all preceding layers, creating intricate inter-dependencies that challenge the isolation and analysis of individual layers in theoretical studies [92].

The theoretical foundation for the expressiveness of MLPs is provided by the *Universal Approximation Theorem* (UAT) [64]. The UAT asserts that a feedforward neural network with at least one hidden layer, a linear output layer, and a suitable activation function can approximate any continuous function to arbitrary accuracy on a compact subset of $\mathbb{R}^n$. Moreover, the theorem states that this approximation is possible for sufficiently wide networks, yet it offers no explicit guidance on how wide they must be. Subsequent studies [60] demonstrated that this property is not confined to specific activation functions but is a fundamental feature of the multilayer feedforward architecture. While the UAT underscores the theoretical capabilities of these networks, it provides no direct insight into how well the model generalizes to unseen data or how to optimally configure network architectures [11]. These limitations motivate further exploration into generalization mechanisms and architectural design principles.

## 2.2 The Optimization of DNNs Under Supervised Learning

Supervised learning [35] is a paradigm where a model is trained to map input data to corresponding output labels. This approach forms the cornerstone of many machine learning applications, enabling models to learn features and representation from labeled datasets.

### 2.2.1 Initialization

The initial parameters in deep neural networks are sampled from a predefined distribution $\mathcal{P}_{\text{init}}$, in some cases, $\mathcal{P}_{\text{init}}$ is determined by structural characteristics such as the fan-in and fan-out of each layer, as outlined in [93].

Initialization schemes play a crucial role in ensuring stable signal propagation during both forward and backward passes in deep neural networks. One widely adopted scheme is LeCun initialization [99]. In this scheme, weights are drawn from a Gaussian distribution with a mean of zero and a standard deviation of $\sqrt{\frac{1}{n_{\text{in}}}}$, where $n_{\text{in}}$ denotes the number of input neurons to the layer. Alternatively, weights can be sampled from a uniform distribution within $[-\sqrt{\frac{1}{n_{\text{in}}}}, \sqrt{\frac{1}{n_{\text{in}}}}]$. This initialization ensures that the variance of activations and gradients remains stable across layers, facilitating efficient training and convergence.

He-Initialization [76], designed for DNNs employing ReLU activation functions. Building upon earlier methods like LeCun initialization, He-Initialization incorporates a gain factor to account for the increased variance introduced by non-linear activations. This approach aims to maintain consistent variance of input and output signals across layers, thereby promoting gradient stability during training [76]. He-Initialization is the default choice in popular deep learning frameworks such as TensorFlow [80] and PyTorch.

In our experiments, we primarily employ LeCun initialization. However, in Chapter 3, we utilize direct Gaussian initialization, as it provides a clearer view of the transformations in parameter distributions before and after training.

### 2.2.2 The Dynamics of Optimization

The training process involves minimizing a loss function which quantifies the discrepancy between the model's predictions and the ground truth labels. This minimization is achieved using optimization algorithms such as Gradient Descent (GD) or Stochastic Gradient Descent (SGD) [34], which iteratively update the model parameters to reduce the loss.

#### Gradient Descent

The gradient descent update rule of a DNN[34] is:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t), \tag{2.4}$$

where $\eta$ is the learning rate, and $\nabla_\theta \mathcal{L}(\theta_t)$ is the gradient of the loss function with respect to the parameters $\theta_t$ at epoch $t$.

The loss function $\mathcal{L}(\theta_t)$ is defined as the average loss over a dataset $\boldsymbol{D}$, quantifying the difference between the true labels and the predicted outputs. Formally, it can be expressed as:

$$\mathcal{L}(\theta_t) = \frac{1}{|\boldsymbol{D}|} \sum_{\mathbf{x_n} \in \boldsymbol{D}} \ell(f_\theta(\mathbf{x_n}), \mathbf{z_n}), \tag{2.5}$$

where $\mathbf{z_n}$ is the true labels associated with input data, $|\boldsymbol{D}|$ is the size of the dataset $\boldsymbol{D}$, $\ell(f_\theta(\mathbf{x_n}), \mathbf{z_n}))$ is the instance-wise loss function.

To compute the gradient $\nabla_\theta \mathcal{L}(\theta)$ efficiently in deep neural networks, the *backpropagation algorithm* is employed [6]. Backpropagation leverages the chain rule of calculus to propagate gradients from the output layer back through the network, layer by layer, to compute the gradient with respect to all parameters. This algorithm is essential for updating the parameters $\theta$ during gradient descent.

For different tasks, the loss functions are defined as follows:

- In regression tasks, the Mean Squared Error (MSE) is commonly used:

$$\ell(\mathbf{z}, f_\theta(\mathbf{x})) = \frac{1}{2} \|\mathbf{z} - f_\theta(\mathbf{x})\|^2,$$

  where $\mathbf{z}$ is the true label and $\mathbf{x}$ represents a single input data point.
- In classification tasks, the Cross-Entropy loss is frequently applied:

$$\ell(\mathbf{z}, f_\theta(\mathbf{x})) = -\sum_c \mathbf{z}_c \log(f_\theta(\mathbf{x})_c),$$

where $c$ indexes the classes, $\mathbf{z}_c$ is the true label for class $c$ in one-hot encoding, and $f_\theta(\mathbf{x})_c$ is the predicted probability for class $c$.

To bridge the discrete-time gradient descent updates and the continuous-time framework, we introduce $\theta(t)$, the parameter vector as a function of continuous time $t$. By interpreting $\eta$ as the time step size $\Delta t$ in Equation 2.10, the discrete update can be rewritten in terms of finite differences:

$$\theta_{t+\Delta t} = \theta_t - \Delta t \nabla_\theta \mathcal{L}(\theta_t). \tag{2.6}$$

As $\Delta t \to 0$, the discrete-time updates transition into a continuous-time process, where the parameter vector evolves according to an ordinary differential equation (ODE). In this framework, we replace $\theta_t$ with $\theta(t)$, emphasizing its dependence on continuous time. The resulting ODE is:

$$\frac{d\theta(t)}{dt} = -\nabla_\theta \mathcal{L}(\theta(t)). \tag{2.7}$$

Here, $\frac{d\theta(t)}{dt}$ represents the instantaneous rate of change of the parameters, and $\mathcal{L}(\theta(t))$ is the loss function evaluated at the parameters $\theta(t)$. This formulation, known as gradient flow, describes the trajectory of the parameters in continuous time as they evolve to minimize the loss function under the assumption of infinitesimally small learning rates.

Based on Equation (2.7), we have:

$$\frac{d\mathcal{L}(\theta(t))}{dt} = \nabla_\theta \mathcal{L}(\theta(t)) \cdot \frac{d\theta(t)}{dt} = -\|\nabla_\theta \mathcal{L}(\theta(t))\|^2 \leq 0, \tag{2.8}$$

This result shows that the loss $\mathcal{L}(\theta_t)$ is non-increasing over time, as long as $\nabla_\theta \mathcal{L}(\theta_t) \neq 0$. This implies that gradient descent always moves in the direction of the steepest decrease in $\mathcal{L}(\theta_t)$. In this context, gradient descent can be viewed as an Eulerian numerical approximation to the continuous-time ODE in Equation (2.7).

This method is commonly referred to as *full batch gradient descent* [77], as it computes the gradient using all samples in the dataset. However, this approach has a significant drawback: real-world datasets often consist of tens of thousands or even millions of data points, making each iteration computationally expensive and, in many cases, impractical.

### Stochastic Gradient Descent

A common alternative to full-batch gradient descent is *stochastic gradient descent* (SGD) [77], which partitions the dataset $\boldsymbol{D}$ into equally sized-mini-batch, either with or without replacement (here we assume independent sampling for simplicity). In each iteration, a mini-batch $\boldsymbol{R} \subseteq \boldsymbol{D}$ of size $b$ is randomly selected, and the gradient is computed using only the data points in $\boldsymbol{R}$. Let the average loss over $\boldsymbol{R}$ be

$$\mathcal{L}_{\boldsymbol{R}}(\theta_t) = \frac{1}{b} \sum_{\mathbf{x}_n \in \boldsymbol{R}} \ell(f_\theta(\mathbf{x}_n), \mathbf{z}_n), \tag{2.9}$$

where $\ell$ is an instance-wise loss function, $f_\theta$ denotes the model, and $\mathbf{z}_n$ is the corresponding label. The standard update rule for gradient descent becomes

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_{\boldsymbol{R}}(\theta_t), \tag{2.10}$$

with $\eta$ as the learning rate. Note that while the full-batch loss $\mathcal{L}(\theta)$ is the true optimization objective, $\mathcal{L}_{\boldsymbol{R}}(\theta)$ is a noisy estimate due to the random sampling of $\boldsymbol{R}$. Hence, $\nabla_\theta \mathcal{L}_{\boldsymbol{R}}(\theta_t)$ is an unbiased but noisy approximation of $\nabla_\theta \mathcal{L}(\theta_t)$.

Define the gradient noise as

$$\xi_t := \nabla_\theta \mathcal{L}(\theta_t) - \nabla_\theta \mathcal{L}_{\boldsymbol{R}}(\theta_t). \tag{2.11}$$

Although in practice $\xi_t$ may be non-Gaussian and correlated, for qualitative analysis [32, 27] we assume it is a mean-zero Gaussian random variable whose covariance scales roughly as $1/b$. Now we can re-write the discrete update as

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) + \eta \xi_t, \tag{2.12}$$

and taking the limit of small $\eta$, we approximate the dynamics by the stochastic differential equation (SDE)

$$d\theta = -\eta \nabla_\theta \mathcal{L}(\theta)\, dt + \sqrt{\frac{\eta^2}{b}}\, \sigma(\theta)\, dW_t. \tag{2.13}$$

Here, $W_t$ denotes a standard Wiener process and $\sigma(\theta)$ characterizes the local noise intensity. For dimensional consistency, since $dW_t \sim \sqrt{dt}$, $\sigma(\theta)$ must have units such that the noise term has the same dimensions as $\theta$ (i.e., if $\theta$ has units $[X]$, then $\sigma(\theta)$ carries units of $[X]/\sqrt{\text{time}}$).

In a more general setting, the noise covariance may be anisotropic. In that case, one writes

$$d\theta = -\eta \nabla_\theta \mathcal{L}(\theta)\, dt + \eta \sqrt{\frac{\Sigma(\theta)}{b}}\, dW_t,$$

where $\Sigma(\theta)$ is the full (matrix-valued) covariance. For simplicity, the isotropic case corresponds to $\Sigma(\theta) = \sigma^2 I$.

The SDE (2.13) implies a Fokker–Planck equation governing the evolution of the probability density $P(\theta, t)$:

$$\frac{\partial P}{\partial t} = \nabla_\theta \cdot \Big[\eta \nabla_\theta \mathcal{L}(\theta)\, P(\theta, t)\Big] + \frac{\eta^2}{2b} \nabla_\theta \cdot \Big[\sigma^2(\theta) \nabla_\theta P(\theta, t)\Big]. \tag{2.14}$$

In the general anisotropic case, $\sigma^2(\theta)$ would be replaced by $\Sigma(\theta)$, with the isotropic assumption being a special case.

Assuming (i) that $\sigma^2(\theta)$ is constant (i.e., independent of $\theta$), and (ii) that the system is ergodic so that $P(\theta, t)$ converges to a stationary distribution $P(\theta)$, we impose detailed balance (i.e., zero net probability flux). This yields

$$\eta \nabla_\theta \mathcal{L}(\theta)\, P(\theta) + \frac{\eta^2\, \sigma^2}{2b} \nabla_\theta P(\theta) = 0. \tag{2.15}$$

Rewriting in terms of $\log P(\theta)$ and integrating gives

$$P(\theta) \propto \exp\Big(-\frac{2b}{\eta\,\sigma^2}\,\mathcal{L}(\theta)\Big). \tag{2.16}$$

Often, the factor of 2 and other constants are absorbed into an effective inverse temperature $\beta$, so that

$$P(\theta) \propto \exp\Big(-\beta\,\mathcal{L}(\theta)\Big), \quad \text{with} \quad \beta = \frac{2b}{\eta\,\sigma^2}.$$

If $\sigma^2(\theta)$ is not constant, the stationary distribution becomes more complicated and generally involves a $\theta$-dependent integral in the exponent.

The SDE analysis reveals that both the learning rate $\eta$ and the batch size $b$ modulate the noise level in SGD. For averaged gradients, the noise variance scales as $\eta^2/b$, so that increasing $b$ reduces the absolute variance of the gradient estimate. In contrast, if gradients were summed rather than averaged, the variance would scale as $\eta^2 b$. This interplay is critical in balancing exploration and exploitation [78, 83]:

- A larger $\eta$ amplifies the noise, promoting exploration of the parameter space and helping the optimizer escape shallow minima or saddle points.
- A smaller batch size $b$ increases noise, whereas a larger $b$ reduces it, leading to more stable but potentially less exploratory updates.

Although the above analysis suggests that increasing the batch size as training progresses could be beneficial, this approach drastically increases computational overhead. Consequently, this is limited to a range of batch sizes determined by the complexity of the task, model, and available computational resources, making it an uncommon approach. Modern practices often favor dynamic learning rate schedules over adjusting the batch size [47, 46, 48]. For example, cyclical learning rates [46] periodically increase $\eta$, effectively reheating the system to escape metastable states.

The stationary distribution $P(\theta) \propto \exp(-\beta\,\mathcal{L}(\theta))$ closely resembles the Bayesian posterior under a Gaussian prior when $\beta = 1$. This connection bridges SGD with variational inference and Bayesian deep learning [22, 23, 24].

This analysis is built on idealized assumptions—such as constant, isotropic Gaussian noise and ergodicity—that may not hold in practice for deep neural networks. In high-dimensional, non-convex landscapes typical of DNNs, noise is often heavy-tailed and parameter-dependent [27], and the system may reside in metastable states with non-negligible probability currents. Consequently, the predicted stationary distribution and related escape times are approximations that capture only the qualitative behavior of SGD, and additional empirical and theoretical investigations are needed to fully characterize its dynamics in realistic settings.

A practically used variant of SGD is SGD with momentum [98]. Momentum helps accelerate convergence by smoothing out gradient updates, much like a physical

object accumulating velocity through inertia. In this method, the velocity vector $\mathbf{v}$ accumulates an exponential moving average of past gradients, effectively dampening oscillations in directions where the loss landscape is steep while amplifying consistent descent directions. The update rules are given by:

$$\mathbf{v}_{t+1} = \mu \mathbf{v}_t - \eta \nabla_{\theta_t} \mathcal{L}(\theta_t), \tag{2.17}$$

$$\theta_{t+1} = \theta_t + \mathbf{v}_{t+1}. \tag{2.18}$$

where $\eta$ is the learning rate and $\mu$ is the momentum coefficient. This approach is related to the heavy-ball method [26] in classical optimization and serves to navigate the parameter space more effectively, especially in regions where the gradient direction remains consistent over iterations. Variants such as Nesterov's accelerated gradient [25] further refine this idea by incorporating a lookahead step, offering additional theoretical and empirical benefits.

## 2.3 Generalization

A generalized model effectively captures the underlying patterns in the data rather than merely memorizing the training set. In deep neural networks (DNNs), generalization is especially intriguing due to their highly over-parameterized architectures. Despite often having far more parameters than training samples, DNNs can still exhibit striking generalization performance. This unexpected phenomenon has challenged conventional machine learning theories [94] and spurred extensive research into the mechanisms driving modern deep learning generalization [7, 51, 52].

In machine learning, *generalization* refers to a model's ability to perform well on data that was not part of the training set. To evaluate a model's performance, the dataset is typically split into a training set and a test set. The training set is used to learn relevant patterns, while the test set serves as an unbiased measure of performance on previously unseen examples. One way to quantify generalization is through the *generalization gap*:

$$\text{Generalization Gap} = \mathcal{L}_{\text{train}}(\theta) - \mathcal{L}_{\text{test}}(\theta),$$

where $\mathcal{L}_{\text{train}}(\theta)$ and $\mathcal{L}_{\text{test}}(\theta)$ denote the training and testing losses, respectively. A small gap indicates that the model genuinely captures the underlying structure in the data rather than merely memorizing the training set.

Traditionally, machine learning theory explains generalization by examining the interplay of model complexity and sample size. Early results, such as those based on the Vapnik–Chervonenkis (VC) dimension [51], show that for a hypothesis class with finite VC dimension—an indicator of model complexity that quantifies the largest set of points a model can shatter—the true loss $\mathcal{L}(f)$ is guaranteed to remain close to the empirical loss $\mathcal{L}_{\text{train}}(f)$ provided that the number of training samples is sufficiently large. In other words, as the training set size grows relative to the VC dimension, the model's performance on unseen data converges to its performance

on the training set, thereby ensuring good generalization.

In this framework, *overfitting* arises when a model is excessively complex relative to the available data, leading to a small training error but a large test error. One way to understand this phenomenon is through the *bias-variance trade-off*: simpler models (high bias, low variance) tend to underfit the data, while more complex models (low bias, high variance) often overfit. Formally, for a model $f(\mathbf{x})$ approximating a target function $f^*(\mathbf{x})$ with additive noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$, the expected squared error at a data point $\mathbf{x}$ decomposes as:

$$\mathbb{E}[(f(\mathbf{x}) - z)^2] = \underbrace{(\mathbb{E}[f(\mathbf{x})] - f^*(\mathbf{x}))^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}\left[(f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})])^2\right]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Irreducible Noise}}, \quad (2.19)$$

where $z = f^*(\mathbf{x}) + \epsilon$. Specifically:

- **Bias**: The error due to the model's inability to capture the true function $f^*(\mathbf{x})$. High bias indicates underfitting.
- **Variance**: The error due to sensitivity to fluctuations in the training data. High variance indicates overfitting.
- **Irreducible Noise**: The inherent noise in the data that no model can remove.

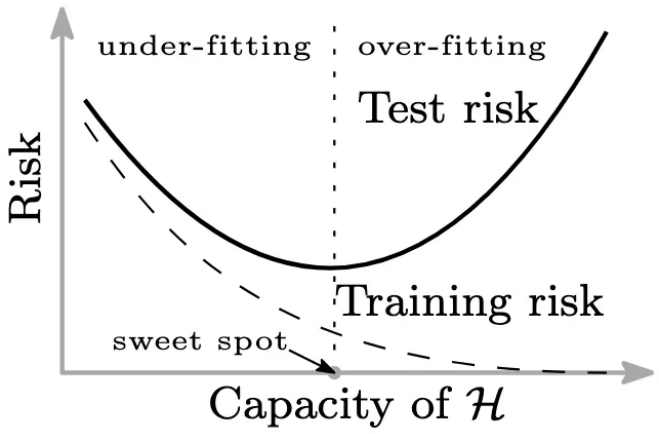



**Figure 2.2:** The bias-variance trade-off curve. In the context of this thesis, risk corresponds to the loss. Source: [94].

As illustrated in Fig. 2.2, increasing the capacity of the hypothesis class $\mathcal{H}$—that is, making the model more complex—can lead to overfitting, in which the training error approaches zero but the test error begins to rise. Conventional wisdom in machine learning advises controlling the capacity of $\mathcal{H}$ according to the bias-variance trade-off, striving to balance underfitting and overfitting. If $\mathcal{H}$ is too restricted, the model may underfit (yielding high empirical loss), whereas if $\mathcal{H}$ is too large, the empirical loss minimizer can latch onto spurious patterns, achieving low training loss but high

true loss on new data.

Regularization is introduced as a key strategy to mitigate overfitting by explicitly controlling model complexity. In practice, this is often achieved by adding a penalty term—commonly an $L^1$ or $L^2$ norm—to the training objective, biasing the learning process toward simpler models. This approach is grounded in the principle of structural loss minimization, where the goal is to minimize a combination of empirical loss and a complexity penalty, thereby balancing the trade-off between accurately fitting the training data and ensuring robust performance on unseen data. Consequently, classical machine learning theory underscores the importance of controlling model capacity via regularization to achieve strong generalization.

### 2.3.1 Generalization in Deep neural networks

Despite the statistical learning community's development of a relatively comprehensive framework for supervised learning [10], it has encountered difficulties in elucidating the extraordinary generalization capabilities of deep learning models.

This challenge was highlighted by Zhang et al. [50], who conducted extensive experiments to investigate the generalization properties of deep neural networks (DNNs). They trained multiple architectures on datasets like CIFAR-10 and ImageNet under several conditions: (1) with true labels, (2) with randomized labels, and (3) with shuffled pixel values. Remarkably, the over-parameterized DNNs performed well on regular datasets with true labels and are also capable of perfectly fitting data with random labels, achieving zero training error despite the lack of any correlation between the inputs and labels, with only a minor increase in training time for the random label scenario. The performance remained at its peak even without explicit regularization, demonstrating that the same over-parameterized DNN can memorize individual data points perfectly to reduce the loss to zero while still effectively learning underlying features and generalizing well to unseen data.

These findings challenge the conventional assumption held within the realm of traditional statistical learning theory, which suggests that over-parameterized models like DNNs should overfit the training data and generalize poorly without explicit regularization techniques. Zhang et al. concluded that while explicit regularization methods, such as weight decay or dropout, may improve generalization performance, they are neither necessary nor sufficient for controlling generalization error. Contrary to classical assumptions, DNNs were shown to generalize extremely well under the true label condition even in the absence of explicit regularization. Instead, their findings suggest that generalization arises from a combination of implicit regularization introduced by gradient-based optimization algorithms like stochastic gradient descent (SGD) and the inductive biases inherent in the architecture [9]. These insights highlight the need to rethink the theoretical frameworks for understanding generalization, as DNNs exhibit behaviors that deviate significantly from classical learning theory.

In recent years, the deep learning theory community has extensively studied the generalization properties of deep neural networks, seeking to reconcile their behavior with traditional statistical learning theories. A notable contribution is the work of Belkin et al. [94], who proposed the concept of the double-descent curve which extends the classical bias-variance trade-off. Traditionally, the bias-variance trade-off posits that minimizing total error requires achieving an optimal balance between bias and variance. However, deep learning models demonstrate a surprising phenomenon: generalization often improves as model complexity increases beyond the interpolation point. This behavior is captured by the double-descent curve, as illustrated in Figure 2.3 [94], offering a novel perspective on the relationship between model capacity and generalization.

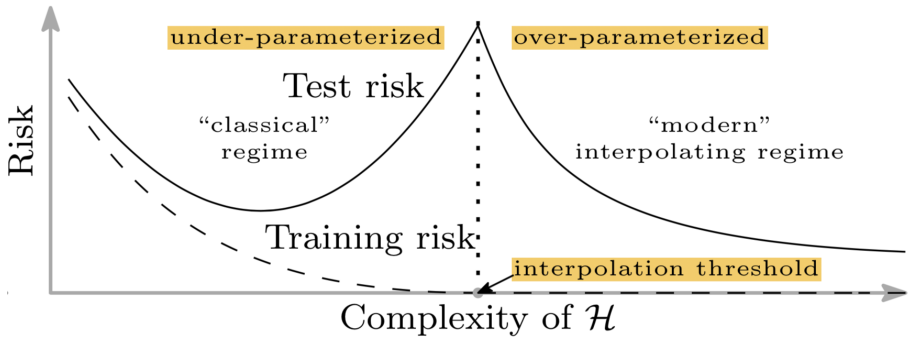



**Figure 2.3:** A double descent risk curve for deep neural networks. Source: [94].

The double-descent curve highlights two distinct phases of deep neural networks. The first phase pertains to under-parameterized models, whose complexity is lower than the number of data samples. This phase corresponds to the classical regime, well-described by traditional statistical learning theories. In contrast, as model complexity increases beyond the interpolation threshold, we observe the second phase—the modern interpolating regime—where the risk or loss undergoes a second descent. This unexpected behavior challenges conventional wisdom and underscores the need for a deeper understanding of generalization in over-parameterized models.

In order to explain why the second phases exist and to further understand the generalization principle of deep neural networks in the modern regime, studies have been conducted from three main perspectives [7, 51, 52]:

- **Complexity of Deep Neural Networks:** Despite the success of conventional statistical learning theory in establishing upper bounds on generalization error using measures such as VC-dimension, Rademacher complexity, and covering numbers, these bounds often become vacuous for deep neural networks due to their explicit dependence on model size. The colossal size of modern deep learning models renders these classical approaches insufficient for explaining generalization in practice [52]. Recently, efforts have been made

to refine these measures and extend their applicability to deep learning. For instance, PAC-Bayes theory [40] has emerged as a promising framework, providing tighter bounds by incorporating probabilistic reasoning about model weights and their distribution.

- **The role of stochastic gradient descent (SGD):** SGD and its variants are critical in determining the trajectory of parameters during training. By modeling SGD through stochastic differential equations (SDEs), researchers have gained insights into the optimization and generalization dynamics of DNNs [54, 53, 33, 9].
- **The geometry of the loss landscape and parameter space:** Deep neural networks involve highly non-convex, often non-smooth loss surfaces [2]. While this complexity complicates traditional optimization analyses, it can also create a rich structure of saddle points and local minima that influence both training trajectories and generalization outcomes. Understanding the geometry of these loss landscapes is key to explaining why over-parameterized, seemingly prone to overfitting, models can still generalize well [3, 8].

In practical research, the three ideas outlined here often intersect—a natural consequence of the inherent complexity of deep learning generalization. At its core, generalization in deep learning seeks to address a fundamental question:

*How do deep neural networks optimized using gradient-based methods generalize well to unseen datasets?*

This question inherently intertwines factors such as model complexity, optimization methods, architectures, and data structure, each contributing to the model's loss landscape directly or indirectly.

In the next section, we will briefly discuss one of the key considerations related to the second perspective, while the remainder of this thesis will primarily focus on the third perspective, which will be explored in detail in the next chapters.

### 2.3.2 The Role of SGD in Generalization

When delving deeper into stochastic gradient descent (SGD), its influence on the loss landscape becomes a focal point, particularly in the context of the widely discussed *sharp minima* or *flatness bias* hypothesis [54, 53, 33]. This hypothesis suggests that well-generalized deep neural networks trained via SGD tend to converge to flat minima, whereas poor generalization corresponds to sharp minima. In other words, according to this hypothesis, the Hessian $\nabla^2_\theta \mathcal{L}(\theta)$ of a well-generalized DNN should exhibit relatively small eigenvalues.

Note that, in theory, the Hessian of a non-convex loss can have both positive and negative eigenvalues. However, when we focus on local minima of DNNs, we typically look at whether there are large positive eigenvalues indicating high curvature (sharp directions). In practice, many analyses center on the largest (positive) eigenvalue

to gauge how sharp a minimum is, as large eigenvalues represent steep directions in the loss landscape. We give a brief description of the role of GD (SGD) here.

In addition to the discussion on the role of noise in stochastic gradient descent (SGD) algorithms in Section 2.2.2, gradient descent (GD) provides additional advantages in the context of flatness bias. A key starting point is the study by Barrett et al. [56], who describe the implicit regularization introduced by gradient descent as *Implicit Gradient Regularization*, proposing that GD inherently guides the optimizer toward flat minima.

Gradient descent can be viewed as a numerical integration method for the following ordinary differential equation (ODE):

$$\dot{\theta} = g(\theta) = -\nabla_\theta \mathcal{L}(\theta), \tag{2.20}$$

where the $g(\theta)$ is a shorthand of $\nabla_\theta \mathcal{L}(\theta)$.

Let us recall equation 2.6 which is the explicit Euler method, a first-order Runge–Kutta method [44]:

$$\theta_{t+\Delta t} = \theta_t + \Delta t\, g(\theta_t), \tag{2.21}$$

incurring a local truncation error of order $\mathcal{O}((\Delta t)^2)$ relative to the exact solution $\theta(t + \Delta t)$ of (2.20).

To systematically account for this discrepancy, backward error analysis [43] introduces a modified ODE:

$$\dot{\theta} = \tilde{g}(\theta) = g(\theta) + \Delta t\, g^{(1)}(\theta) + (\Delta t)^2 g^{(2)}(\theta) + \cdots, \tag{2.22}$$

whose solutions match the discrete updates $\theta_{t+\Delta t}$. In the context of gradient descent, we write:

$$\dot{\theta} = \tilde{g}(\theta) = g(\theta) + \Delta t\, g^{(1)}(\theta) + \mathcal{O}((\Delta t)^2). \tag{2.23}$$

Matching the discrete update Equation 2.21 to the Taylor expansion of $\theta(t + \Delta t)$ under $\dot{\theta} = \tilde{g}(\theta)$ reveals the first-order correction

$$g^{(1)}(\theta) = -\frac{1}{2}\, g'(\theta)\, g(\theta). \tag{2.24}$$

Since $g(\theta) = -\nabla_\theta \mathcal{L}(\theta)$, the correction simplifies to

$$g^{(1)}(\theta) = -\frac{1}{2}\, \nabla_\theta^2 \mathcal{L}\, \nabla_\theta \mathcal{L} = -\frac{1}{4}\, \nabla_\theta \|\nabla_\theta \mathcal{L}(\theta)\|^2. \tag{2.25}$$

Hence, the modified ODE becomes

$$\dot{\theta} = -\nabla_\theta \mathcal{L}(\theta) - \frac{\Delta t}{4}\, \nabla_\theta \|\nabla_\theta \mathcal{L}(\theta)\|^2 + \mathcal{O}((\Delta t)^2), \tag{2.26}$$

which can be viewed as the gradient flow of a modified loss function:

$$\tilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \frac{\Delta t}{4}\, \|\nabla_\theta \mathcal{L}(\theta)\|^2. \tag{2.27}$$

From this perspective, discretizing gradient flow inherently imposes a regularization effect by altering the gradient field with an additional term proportional to $\|\nabla_\theta \mathcal{L}(\theta)\|^2$. This term penalizes sharp minima (i.e., regions of high curvature), biasing optimization toward flatter minima with better generalization properties. To retain such benefits regardless of the step size $\Delta t$, one can explicitly add a gradient penalty term to the loss:

$$\mathcal{L}_{\text{new}}(\theta) = \mathcal{L}(\theta) + \frac{\gamma}{4} \|\nabla_\theta \mathcal{L}(\theta)\|^2, \tag{2.28}$$

This explicit penalty preserves the regularization effect even if $\Delta t$ is altered, thereby maintaining the bias toward flat minima.

While this implicit bias from SGD or GD has inspired the development of optimization algorithms [55], its universality remains an open question. Several studies have challenged the general applicability of this hypothesis [41, 42], emphasizing the need for further investigation into its validity and broader implications. In particular, Dinh et al. [41] argue that flatness alone cannot directly explain the generalization of deep neural networks. In some cases, we can find well-generalized models converge to sharp minima, while Ramasinghe et al. [45] demonstrated that models failing to generalize can also exhibit flat minima. These findings suggest that the mystery of generalization cannot be fully understood by focusing solely on the flatness bias of optimization methods.

Wu et al. [2] propose that rather than exclusively studying the implicit regularization induced by SGD, researchers should also consider the geometric structure of the loss landscape in deep neural networks. They argue that the loss landscape guides optimizers toward low-complexity solutions, ultimately facilitating generalization. Furthermore, Chiang et al. [3] argue that while optimization methods influence the speed and effectiveness of convergence, the generalization error—and even the structures of decision boundaries—are predominantly determined by the intrinsic properties of the loss landscape itself, rather than by the specific choice of gradient-based or non-gradient-based optimization methods. They demonstrated this conclusion across multiple datasets and deep neural network architectures.

These insights motivate further investigation into the interplay between generalization and the structural properties of the loss landscape and parameter space. In the next chapter, we explore two ubiquitous phenomena in the parameter space of deep neural networks and examine their interconnections. In Chapter 4, we delve deeper into these phenomena from the perspective of information geometry.

# 3

# The Sub-networks in Deep Neural Networks

Zhang et al. [50] have highlighted that the remarkable generalization power of over-parameterized deep neural networks is intimately tied to understanding their optimization dynamics and associated inductive biases. Subsequently, several studies [2, 3] have empirically and theoretically shown that the parameter space of deep neural networks can exhibit properties analogous to implicit regularization.

In this chapter, we focus on two phenomena that relate to such implicit regularization effect: structural symmetry breaking [1] and sub-networks of deep neural networks. Structural symmetry breaking highlights that different hidden layers in a multilayer perceptron may exhibit varying degrees of robustness when re-initialized. Only a limited subset of these layers genuinely encodes the data's structural information, suggesting an intrinsic layer-wise heterogeneity in how neural networks learn.

Sub-networks, on the other hand, often termed *winning tickets* in the lottery ticket hypothesis [69], are sparser architectures hidden within the original over-parameterized model. The hypothesis posits that among the multitude of randomly initialized sub-networks, a small subset—referred to as winning tickets—can independently train to match or surpass the performance of the full network when coupled with their original initialization. These winning tickets are typically identified through iterative magnitude pruning (IMP) after training, revealing that the crucial capacity of the model is concentrated in these sub-networks. Notably, the pruned sub-network demonstrates performance comparable to the original network, even without additional training [85].

In this work, we extend experiments on structural symmetry breaking to demonstrate that the phenomenon itself is also rooted in a sub-network akin to that identified by IMP. This provides fresh insight into how sparse sub-networks underlie the emerging layer-wise heterogeneity in over-parameterized deep neural networks. And in the next chapter, the nature of these sub-structures is sought qualitatively and experimentally through the lens of information geometry.

## 3.1 Structural Symmetry Breaking

Structural symmetry breaking refers to the phenomenon in which, despite the initial symmetry of the structure of the MLP's hidden layers, certain layers become disproportionately important for the network's performance. Neural networks are typically initialized layer by layer using the same initialization scheme, where each layer's parameters are drawn from an identical statistical distribution. In this context, symmetry signifies that the initialization of each hidden layer is statistically equivalent. Zhang et al. [1] summarized this phenomenon as a self-restriction mechanism in the number of critical layers, leading over-parameterized deep networks trained with stochastic gradient descent (SGD) to exhibit low complexity.

In this chapter, we build upon the work of Zhang et al. to further investigate the causes and implications of structural symmetry breaking. Our primary research questions are as follows:

- Can structural symmetry breaking be attributed solely to individual layers, or is there a more intricate structure within the parameter space?
- How does this "low complexity" manifest within the specific parameter space?

Additionally, we pose the following question:

- In what way do neural networks exhibit this sub-structure, where different layers take on distinct roles or generate unique sub-networks during training?

We focus on addressing the first two questions in this section, while the third question is explored in later chapter of the thesis.

Zhang et al. [1] introduce two operations to analyze structural symmetry breaking:

- ***Re-initialization***: After training, each layer is re-initialized individually by resetting its parameters to their initial values: $\theta_l^T \leftarrow \theta_l^0$, while leaving all other layers unchanged: $(\theta_1^T, \ldots, \theta_{l-1}^T, \theta_l^0, \theta_{l+1}^T, \ldots, \theta_L^T)$. Here, $\theta_l^T$ denotes the parameters of the $l$-th layer at the end of training epoch $T$, and $\theta_l^0$ represents the parameters at initialization. The performance of the modified network is then evaluated on a test set. The relationship between a layer and its performance impact following re-initialization is termed the *re-initialization robustness* of that layer.

- ***Re-randomization***: In this operation, the parameters of a single layer are replaced with new random values drawn from the same distribution $d$ used during initialization. Specifically, this is represented as $\theta_l^T \leftarrow \tilde{\theta}_l$, resulting in the modified network: $(\theta_1^T, \ldots, \theta_{l-1}^T, \tilde{\theta}_l, \theta_{l+1}^T, \ldots, \theta_L^T)$. Here, $\tilde{\theta}_l$ refers to the newly sampled parameters.

Crucially, no additional training or fine-tuning is performed after applying these two operations. If the network's performance experiences negligible degradation

following these operations, the layer is classified as *robust*. Conversely, if there is a significant drop in performance, the layer is deemed *critical*.

### 3.1.1 Critical and Robust Layers

This thesis adopts the same network architecture as Zhang et al. [1]. The model is a multilayer perceptron (MLP) comprising three fully connected layers, each with an output dimension of 256, followed by a Softmax layer [36] as the final classifier. The output dimension of the Softmax layer is 10, corresponding to the labels in the MNIST dataset [37]. MNIST is a handwritten digits dataset containing 70,000 images of numbers (0–9), requiring ten output neurons to represent the labels. Stochastic Gradient Descent (SGD) with a momentum [38] is used to optimize the multi-class cross-entropy loss [81].

The training process is conducted over 100 epochs with a stage-wise learning rate schedule [82], which reduces the learning rate by a factor of 0.2 at epochs 30, 60, and 90. A batch size of 128 is used during training.

The results of the re-randomization and re-initialization experiments on the trained model are presented in Figure 3.1 and Figure 3.2. The MLP achieves a high performance on the MNIST dataset, attaining an accuracy of 0.9880. However, due to the complex dependencies between the classification function and the parameters of each layer [1], re-randomization of any single layer completely disrupts the learned representations. As a result, the classification accuracy drops to the level of random guessing (10% accuracy for ten classes), as illustrated in Figure 3.1.

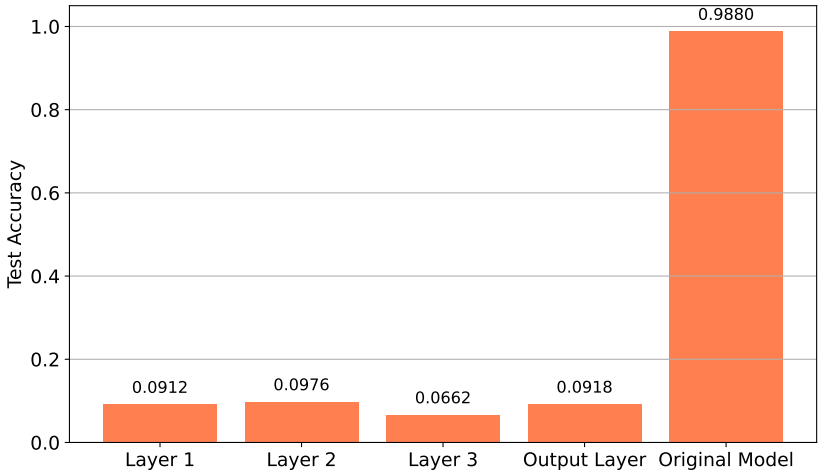



**Figure 3.1:** Layer-wise results of re-randomization showing significant performance drop, indicating layer dependencies.

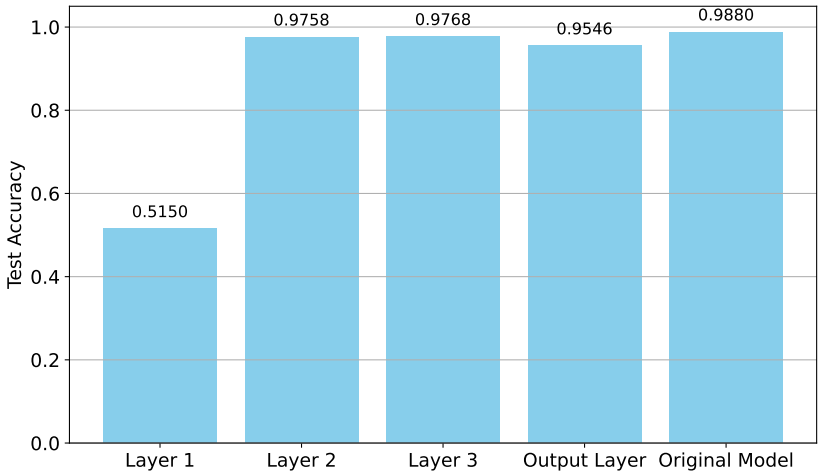



**Figure 3.2:** Layer-wise results of re-initialization showing the structural symmetry breaking, with first layer's criticality and other layers' robustness.

However, more intriguingly, as shown in Figure 3.2, re-initialization reveals that while the first layer is crucial for the network's performance, the remaining layers exhibit robustness to re-initialization. This stark contrast between the two types of layers is referred to as structural symmetry breaking by Zhang et al. [1]. Accordingly, they classify the first layer as *critical* and the others as *robust*.

Zhang et al. further argue that if the increase in gradients during backpropagation of SGD caused the bottom layers (those closest to the input layer) to update more aggressively than the top layers (those closest to the output layer), one would expect a smoother transition in the behavior of the first layer, rather than the observed drastic differences. This indicates that structural symmetry breaking cannot be fully explained by backpropagation alone but instead emerges from more subtle, underlying mechanisms. They hypothesize that structural symmetry breaking in over-parameterized deep networks trained with stochastic gradient descent reflects a specific form of implicit regularization [49], enabling the network to maintain low complexity by restricting the number of critical layers.

## 3.2 Beyond Layer-wise Re-initialization

We extend our analysis beyond the layer-wise level to examine parameter-specific re-initialization, focusing on the critical role played by certain parameters. Re-initializing these critical parameters results in a significant degradation of the model's performance. To further explore this phenomenon, we introduce parameter-wise re-initialization as a method for assessing the importance of individual parameters.

### 3.2.1 Parameter-wise Re-initialization

To empirically investigate how a network's parameters evolve during training, we propose a parameter-wise re-initialization method. This approach is based on analyzing the absolute differences between parameters before and after training. The method provides a more granular exploration of the parameter space in neural networks, enabling us to understand how specific parameter updates contribute structural symmetry breaking within each layer.

Algorithms 1 present the pseudocode for the re-initialization scheme. Specifically, Algorithm 1 details the process of re-initializing specific parameters within a layer of the network. The algorithm computes the differences between the final and initial parameter values, identifies parameters whose differences fall within a specified threshold range—from *min_threshold* to *max_threshold*—and resets these parameters to their initial values. The updated model is then returned for further evaluation. In the subsequent experiments, the *min_threshold* is set to zero.

**Algorithm 1** Parameter-Wise Re-initialization

**Input:** Model parameters $\theta$; initial parameters $\theta^0$; thresholds: *min_threshold*, *max_threshold*

**Output:** Updated parameters $\theta'$

$\theta' \leftarrow \theta$
**for** each $\theta_i \in \theta$ **do**
    $\Delta_i \leftarrow |\theta_i - \theta_i^0|$
    **if** *min_threshold* $\leq \Delta_i \leq$ *max_threshold* **then**
        $\theta'_i \leftarrow \theta_i^0$
    **end if**
**end for**
**return** $\theta'$

Next, we evaluate the model's performance across various threshold values. The procedure involves iterating over a range of *max_threshold* values, creating a new model instance for each threshold. In this experiment, the parameter-wise re-initialization process (Algorithm 1) is applied to reset parameters within the specified range for only one specific layer of the model, while the parameters of the remaining layers remain unchanged. The re-initialized model is then compiled and evaluated on the same test dataset.

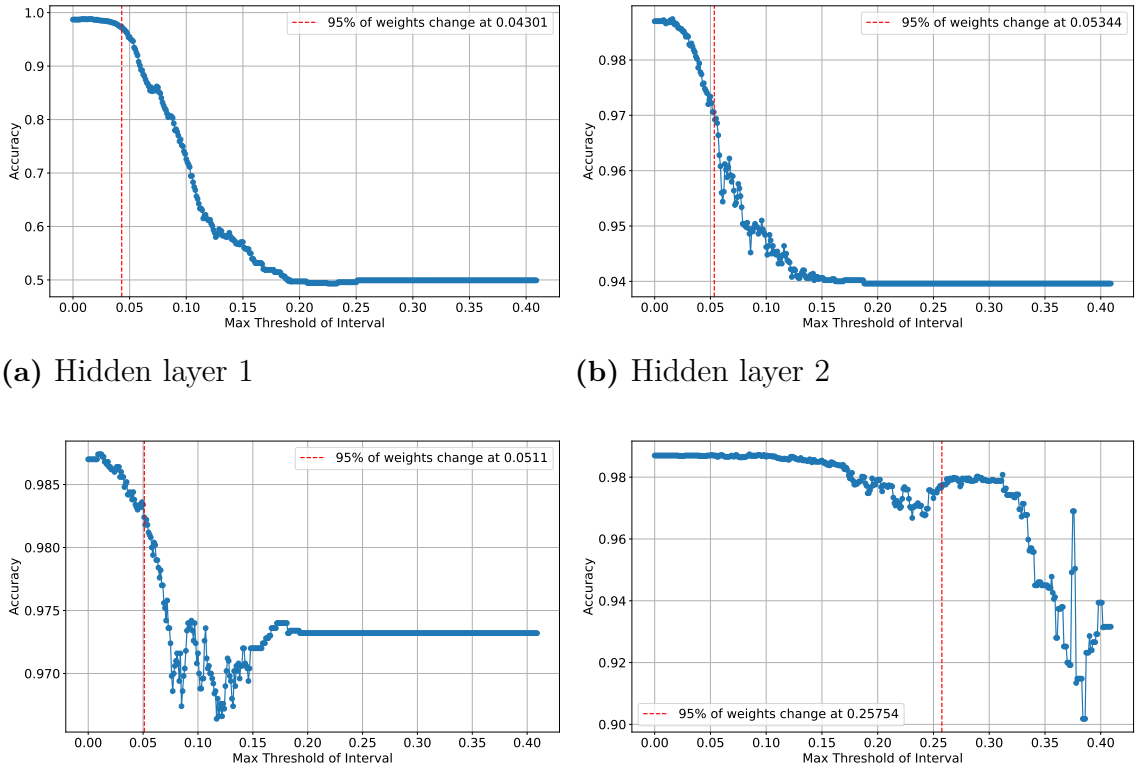
We reinitialize the parameters of each layer by gradually increasing the maximum threshold from 0 to 0.4 in increments of 0.005 at each time. The result of this re-initialization is shown in Figure 3.3. Additional details about the parameters distributions and the distribution of the absolute change of parameters before and after training are provided in Table A.1 and Figure A.2 in the Appendix.

In Figure 3.3, the decay in performance observed as the maximal threshold increases is expected. Re-initializing parameters that have undergone little or no change dur-

ing training does not significantly alter the model's weights, and thus, the performance remains largely unaffected. However, as more parameters with larger changes are included in the re-initialization process, the decay becomes evident.

The decay eventually reaches a level comparable to whole-layer re-initialization as the maximal threshold increases. This observation indicates that most parameters in the first layer are robust to re-initialization, with approximately 95% of parameters exhibiting changes of less than 0.05 (as shown in Figure 3.3). The significant decay observed when including parameters with larger changes suggests the presence of critical parameters—those with substantial updates during training that contribute disproportionately to the model's performance.

The decay caused by re-initialization in the last three layers is significantly smaller than in the first layer. This aligns with the findings of Zhang et al. [1], further highlighting the distinctive role of the first layer in driving structural symmetry breaking in the network.




**(a)** Hidden layer 1

**(b)** Hidden layer 2

**(c)** Hidden layer 3

**(d)** Output Layer

**Figure 3.3:** Parameter-wise re-initialization with varying thresholds per layer. The red dotted line indicates the threshold at which 95% of the parameters are re-initialized in each layer. Notably, parameter-wise re-initialization results are consistent with the layer-wise method. Subplot (a) exhibits a distinctly different range compared to the other three subplots, highlighting the unique role of the first layer.

When the decay begins (particularly evident in the first hidden layer), the majority

of parameters in each layer have already been re-initialized. This suggests that a small subset of parameters with significant changes plays a critical role in structural symmetry breaking, as revealed by parameter-wise re-initialization.

We then extend this parameter-wise re-initialization analysis to the entire model, as illustrated in Figure 3.4. The distribution of parameters across the whole model, based on the magnitude of absolute changes, is detailed in Table A.2. This broader perspective provides further insight into the parameter-specific dynamics influencing the network's structure and performance.

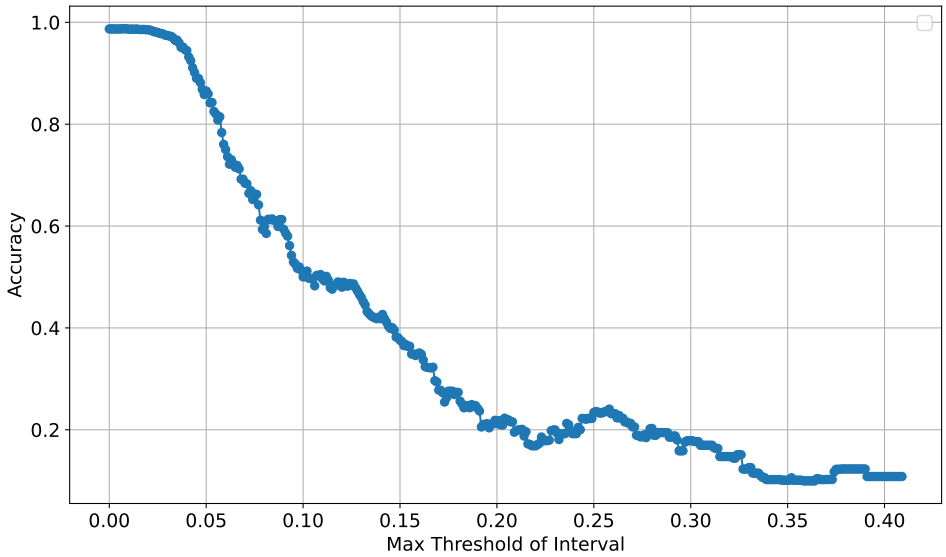



**Figure 3.4:** Parameter-wise re-initialization with different thresholds for the whole model

As shown in Figure 3.4, the same thresholds as in Figure 3.3 lead to significantly larger accuracy decay for critical parameters, with the eventual accuracy dropping below 0.2, equivalent to random guessing. Zhang et al. [1] hypothesized that critical layers, which house the primary learning capacity of the network, render other layers robust to re-initialization. However, even when the entire first layer is re-initialized, the lowest accuracy remains at 0.5150 (as shown in Figures 3.2 and A.1) or when parameter-wise re-initialization is applied per layer (as shown in Figure 3.3). This accuracy is still significantly better. This suggests that the remaining layers also learn meaningful features from the output of the first hidden layer.

In other words, for per layer re-initialization, whether layer-wise or parameter-wise, there are always some parameters that maintain the ability of classification in the layers which have not been re-initialized, preventing the model from disintegrating to random guessing. The parameters that caused only fairly small decays in the robust layer when the layers were re-initialized independently in Figure 3.3, once re-initialized together with the parameters that caused larger decays in the first layer, caused a larger decay than any of the individual layers.

Once we consider the entire parameter space and re-initialize parameters across different layers based on the magnitude of their absolute change before and after training, the accuracy drops significantly. This implies a crucial relationship between the parameters in different layers. Those potential parameters which are involved contain the capacity of learning, we call them critical parameters. Based on these observations, we postulate that these critical parameters, which represent only a small fraction of the total, form a sparse and compact sub-network within the MLP. This phenomenon can be connected with the Lottery Ticket Hypothesis described by Zhang et al. [69].

## 3.3 Lottery Ticket Hypothesis

The observation from Figure 3.4 reveals that the critical parameters of each layer are closely interconnected with those of other layers. This finding implies that the critical parameters collectively constitute a sub-network within a well-trained neural network. To further investigate this phenomenon, we employ parameter-wise re-initialization experiments, leveraging the Lottery Ticket Hypothesis (LTH) [69] to identify a comparable sub-network embedded within the trained deep neural network.

### 3.3.1 Background of the Lottery Ticket Hypothesis

Frankle et al. [69] proposed the Lottery Ticket Hypothesis (LTH), which suggests that within a large, well-trained neural network, there exists a much smaller and sparser sub-network (a winning ticket) that can be re-trained to achieve performance comparable to the original network after pruning and re-initialization. Crucially, a winning ticket consists not only of a sub-network but also of its specific initialization. If the sub-network is re-initialized with a different sample from the same distribution, it no longer demonstrates the high performance observed with the original initialization. This is the same scheme as re-initialization. This follows the same scheme as re-initialization.

Iterative Magnitude Pruning [91] is the process of identifying and removing less important parameters from the network, leading to the discovery of the winning ticket. The standard pruning process involves iteratively training the network, removing parameters with the smallest magnitudes as they are assumed to have minor contributions to learning, and re-training the pruned network. The goal is to retain the network's expressive power while significantly reducing its complexity.

While the term lottery might suggest that the sub-network is determined solely by initialization, this is not the case. The winning ticket or sub-network structure is identified only after training through pruning. It is a function of the training dynamics, which determine the importance of specific parameters across the network.

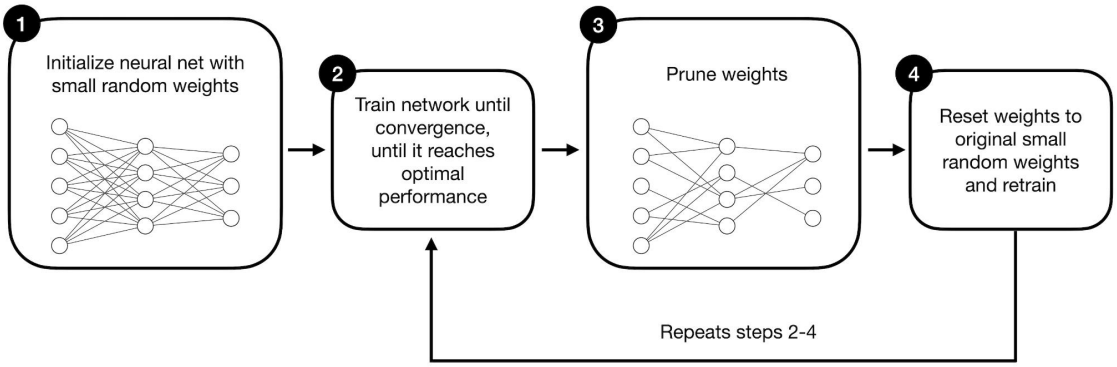



**Figure 3.5:** A flow chart illustrates the training procedure for the lottery ticket hypothesis, is adapted from Dr. Sebastian Raschka's blog [39].

Importantly, the Lottery Ticket Hypothesis is not constrained by layer size; the identified sub-network spans across layers, depending entirely on the absolute magnitude of the parameters. This highlights that sub-network is not tied to any specific layer or architecture but emerges as an optimal configuration discovered through the interplay of over-parameterization, training, and pruning.

### 3.3.2 Lottery Tickets Hypothesis for Parameter-wise Re-initialization

To investigate the sub-network identified through parameter-wise re-initialization of the entire model parameter space, we propose an alternative pruning method. Instead of selecting the winning ticket through iterative magnitude pruning and re-training the model after pruning, our approach combines parameter-wise re-initialization, as outlined in Algorithm 2, to reveal a similarly sparse sub-network. This method introduces a novel pruning criterion: rather than pruning based on the absolute magnitude of parameters in a post-trained neural network with good generalization, critical parameters are selected based on the extent of their deviation from their initialization.

This algorithm prunes the parameters of a given neural network model based on the absolute differences from their initial values, retaining only those parameters whose changes fall within a specified threshold. Notably, after pruning, we do not re-train the pruned model, aligning with the re-initialization experiment settings presented in this thesis. This approach differs from the Lottery Ticket Hypothesis (LTH), which aims to maintain the same performance through multiple iterations of pruning and re-training, with the goal of drastically reducing the network's size. Instead, our focus lies in uncovering an identical sub-network that exists within the trained model.

To investigate the sub-networks identified through pruning and re-initialization under two criteria—magnitude and the absolute value of changes between initialization

**Algorithm 2** Pruning via Parameter-Change

**Input:** Model parameters $\theta$, initial parameters $\theta^0$, pruning fraction $p$
**Output:** Pruned parameters $\theta'$

$\theta' \leftarrow \theta$
**for** each $\theta_i \in \theta$ **do**
    $\Delta_i \leftarrow |\theta_i - \theta_i^0|$
**end for**
**Determine threshold** $\tau$ **such that** fraction $p$ of all $\Delta_i \leq \tau$
**for** each $\theta_i \in \theta$ **do**
    **if** $\Delta_i \leq \tau$ **then**
        $\theta'_i \leftarrow 0$
    **end if**
**end for**
**return** $\theta'$

and post-training parameters—we conducted further experiments. Building on the assumptions of the Lottery Ticket Hypothesis (LTH) regarding magnitude-based parameter importance, we ranked parameter importance based on two criteria: the magnitude of the parameters themselves and the magnitude of their change during training. Here, a larger magnitude of change indicates greater importance for the model and a more significant contribution to its capacity. Parameters were then ranked from least to most important under both criteria. For each ranking, the same percentage of parameters was selected for pruning or re-initialization, expressed as a fraction of the total parameters. The resulting sub-networks, generated through these operations, were evaluated directly on the test set without re-training. The final results are presented in Figure 3.6.

We observe several notable phenomena in Figure 3.6. First, among the two re-initialization-based curves, the change-based approach demonstrates significant decay only after the vast majority of parameters have been re-initialized. This observation aligns with our earlier findings and reinforces the characteristic of trained neural networks where only a small subset of parameters across layers actively encode critical information about the data features. Secondly, random selections perform worse than a systematic, criteria-driven approaches. Moreover, when re-initialization is guided by well-defined criteria, it often preserves higher accuracy than pruning, because reverting parameters to their original initialization provides a stronger prior than removing them entirely.

In a more detailed comparison of re-initialization and pruning, re-initialization demonstrates superior accuracy under change-based criteria, while pruning maintains higher accuracy when adhering to magnitude-based criteria. The rationale behind pruning and re-initialization is that parameters with smaller magnitudes are set to zero during pruning, while those with smaller changes are resettled to their initial values during re-initialization. This is done to maintain the model's performance as much as possible after the operation. Consequently, magnitude-

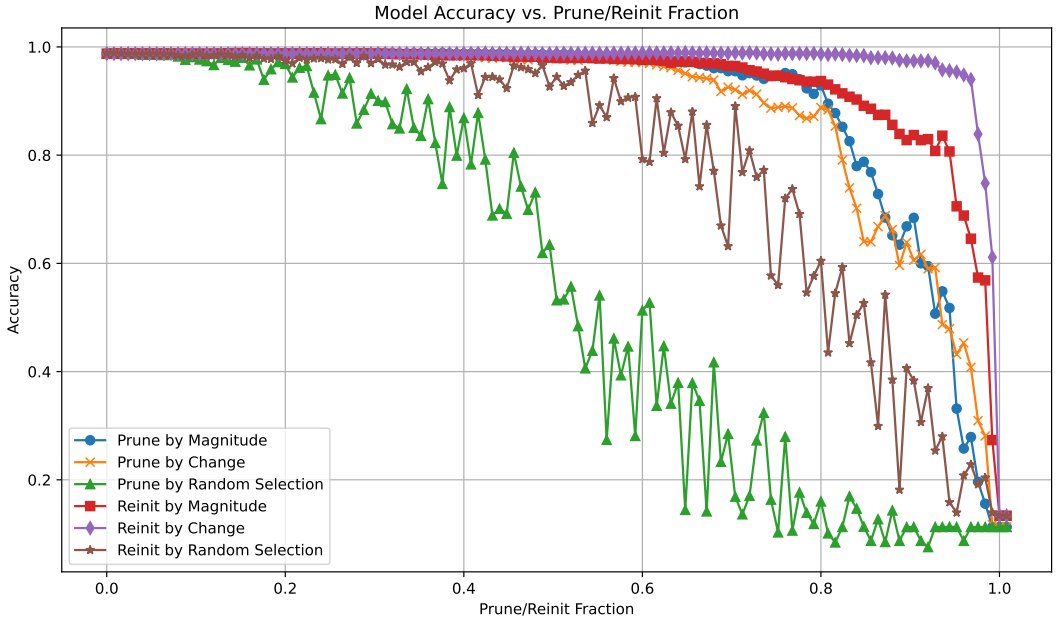



**Figure 3.6:** A plot of the comparison results of sub-networks based on pruning and re-initialization of parameter magnitude and absolute change on the test set which illustrates the impact of two parameter-wise methods—pruning and re-initialization—on the performance of neural networks. These methods are shown by two criteria: the absolute magnitude of the parameters and the absolute change in parameter values between their initialization and post-training states. These two methods can be seen as distinct approaches to perturb the neural network on a parameter-by-parameter basis.

based criteria and change-based criteria naturally possess advantages in these two approaches. Specifically, when setting parameters in a neural network to zero, the effective approach to minimize the damage on the model's capacity is to remove parameters that are already close to zero, which is the advantage of magnitude-based criteria over change-based criteria in pruning. The analogous advantage also applies to criteria based on change in re-initialization, as the small changed parameters are already closed to their initialization values.

The pruning-based experiments show that the results for the two criteria remain largely consistent. This outcome is not surprising, as pruning introduces a larger perturbation to the model compared to re-initialization. Specifically, pruning directly sets the selected parameters to zero, ensuring that the selected parameters no longer contribute to the model, rather than re-initializing them with initialization. When decay becomes apparent in both pruning models, the majority of the parameters have already been set to a value of 0. Additionally, due to our initialization strategy, where parameters are initially drawn from a Gaussian distribution (as shown in Figure A.2), parameters with larger absolute changes during training are more likely to correspond to those with larger absolute values in the trained model. This overlap in parameter selection between the two criteria and the nature

of pruning further contributes to the similarity of the pruned models during the pruning process.

## 3.4 The Sub-networks in Deep Neural Networks

It should be clarified that this work does not attempt to reject the importance of layers in deep neural networks (as pointed out by Zhang et al. [1]) but rather explores the more subtle structures on this basis, with the intention of finding low-complexity structures across layers, giving us a deeper and more concrete understanding of the training process of deep neural networks.

The primary focus here is not on comparing pruning methods but on gaining a deeper understanding of the sub-network structures inherent in trained neural networks. The choice of pruning method ultimately depends on the criterion used to assess the importance of neural network parameters. Both criteria, along with the perturbation methods—pruning and re-initialization—examined in this study, highlight, to varying extents, the existence of these sub-networks. These sub-networks are the central focus of this thesis and are further analyzed in the subsequent chapter.

It is worth mentioning that if pruning is done to reduce the size of the network, then one will select the network after 60% of it has been pruned for re-training (because at this point the network's performance is not significantly decayed, but most of the parameters have been removed), and then this pruning-re-training process is repeated, with the pruning fraction changing until the smallest network is found. Therefore, an interesting research direction is to comprehensively compare the performance of the final sub-network obtained by pruning according to the magnitude of change and the sub-network of LTH based on the benchmark of IMP [79].

One can calculate the overlap ratio of the two criteria for each fraction of pruned or re-initialized parameters. Let $N$ denote the total number of parameters in the model, and let $S_1$ and $S_2$ be two sets of indices corresponding to parameters selected by different criteria, with each selection comprising a fraction $p$ of the parameters (i.e., $|S_1| = |S_2| = pN$). We define the *overlap ratio* $R$ as

$$R = \frac{|S_1 \cap S_2|}{pN}. \tag{3.1}$$

This metric quantifies the proportion of parameters common to both selections relative to the number selected.

When parameters are chosen uniformly at random, the expected number of common parameters is $p^2N$, since each parameter has a probability $p$ of being selected in each set independently. Thus, the expected overlap ratio in the random case is given by

$$\frac{p^2N}{pN} = p.$$

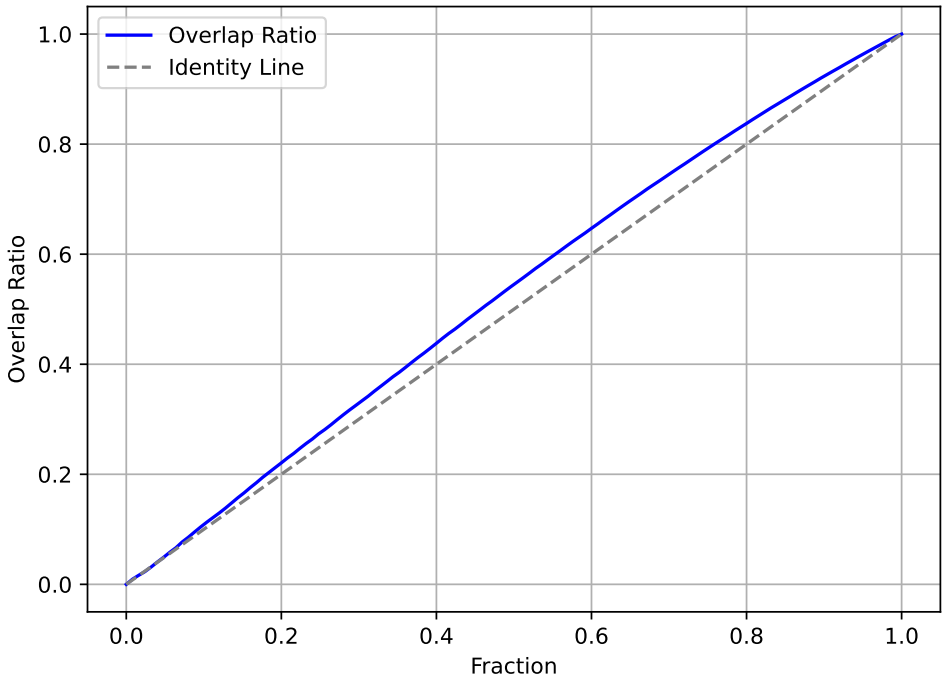



**Figure 3.7:** A plot illustrates the overlap ratio, aligning with the experimental settings from previous sections where a fraction of parameters is pruned or re-initialized.

In other words, if the selections are statistically independent, the overlap ratio is expected to equal $p$.

Figure 3.7 presents the results of our experiments. We compare two criteria: one based on the magnitude of the final weights and the other based on the absolute change from initialization. Our experimental results indicate that although the parameter selection outcomes using the magnitude-based criterion and the absolute change criterion outperform those of random selection, the two methods exhibit no statistical correlation when assessed via the overlap ratio. In the case of two independent random selections of a fraction $p$ of the parameters, the expected overlap ratio is exactly $p$. Our observation that the measured overlap ratio aligns with this expectation suggests that the two criteria are statistically independent in terms of the indices they select. Consequently, we can assert that the change-based criterion defines effective sub-networks in neural networks in a distinct manner.

Even the two independent selection methods yield uncorrelated sets of parameter indices, the resulting sub-networks can share a significant number of non-critical parameters. For example, as illustrated in Figure 3.6, when a large fraction of parameters is pruned (e.g., $p = 0.8$), the overlap ratio between the two criteria reaches 83.7%, while the model's accuracy remains around 90%. This observation indicates that the two criteria converge on a similar set of non-critical parameters. In other words, despite being statistically independent, the methods overlap considerably in pruning and re-initialization. This interplay between independent selection methods and parameter redundancy provides an intriguing direction for future research aimed

at further elucidating the importance of individual parameters in neural networks.

Based on our experiments, it becomes evident that the distribution of critical parameters forms a distinct critical sub-network. Furthermore, this sub-network, in conjunction with the varying roles of feature extraction across layers, contributes to the emergence of the observed structural symmetry-breaking phenomenon. In the case of this three-hidden-layer MLP, feature learning in the first layer—achieved with a relatively small number of critical parameters—emerges as a key factor in determining the network's overall performance. In contrast, subsequent layers perform layer-by-layer feature extraction, enhancing the network's learning capabilities. This process results in the formation of a critical, low-complexity sub-network that underpins the model's effectiveness.

The above answers the first two questions posed in the previous of this section: Firstly, structural symmetry breaking cannot be attributed solely to individual layers; rather, it is a property of the emergence of sparse sub-networks within the model. This is because re-initialization ultimately operates on the sub-networks identified according to the corresponding criterion. Furthermore, the low complexity described by Zhang et al. [1] corresponds to the sparse sub-network that encapsulates the capacity of the model.

The question of how stochastic gradient descent (SGD) shapes the parameter space of the network during training to form these sub-networks is challenging to address directly due to the complexity of deep neural network dynamics [92]. We propose considering the unique role of the parameter space itself before attributing a magic effect to SGD [2] in such cases. As noted by Wu et al.[2] and Ping-yeh et al.[3], the parameter space inherently influences generalization. To understand how this high-dimensional parameter space interacts with optimization dynamics and leverages the structure of natural datasets to construct a valid model containing a sparse and critical sub-network, we will examine the loss landscape from the perspective of information geometry in the next chapter.

# 4

# The Spectrum of Deep Neural Networks

In this chapter, we further explore the properties of sub-networks and elucidate their connection to the eigen-spectrum of the Fisher Information Matrix (FIM) [104] through a series of novel experiments. The FIM acts as a metric tensor on the statistical manifold induced by neural networks, thereby endowing the parameter space with a Riemannian structure—a structure we refer to as the 'neuromanifold' [106]—from an information geometry perspective.

Baratin et al. [74] demonstrated that deep neural networks evolve primarily along a limited number of eigendirections of the FIM, where a few large eigenvalues dominate the spectrum. This spectral analysis highlights the anisotropy of the FIM spectrum and its impact on parameter space evolution, while also allowing us to build on these findings to reveal how these anisotropic characteristics relate to the sub-networks identified earlier. We also examine the internal structure of the subspace spanned by the eigenvectors corresponding to the largest eigenvalues of the FIM spectrum, seeking spectral evidence for how models converge into sparse subnetworks—a topic that extends discussions from the previous chapter.

The architecture and dataset utilized in this chapter are consistent with those employed in the preceding chapters to ensure continuity.

## 4.1 Neuromanifold and Fisher Information Matrix

### 4.1.1 Statistical Manifold

We define the statistical manifolds from the perspective of *Information Geometry* as applied to neural networks [106]. Let $X \in \mathbb{R}^d$ be a random variable representing the network input, and let $Y = f_{\boldsymbol{\theta}}(X)$ denote the corresponding output, where $f_\theta$ represents the input-output mapping parameterized by neural network parameters $\theta$.

We introduce the following probability distributions:

- $p_X(\mathbf{x})$: the marginal distribution of inputs,

- $p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \theta)$: the conditional distribution of outputs given inputs,
- $p_{X,Y}(\mathbf{x}, \mathbf{y}; \theta)$: the joint input-output distribution,
- $p(\mathbf{x}, \mathbf{z})$: the training distribution, where $\mathbf{z}$ are the ground-truth labels associated with the inputs $\mathbf{x}$.

Using the conditional probability rule, the joint distribution of $(X, Y)$ can be expressed as:

$$p_{X,Y}(\mathbf{x}, \mathbf{y}; \theta) = p_X(\mathbf{x}) \cdot p_{Y|\mathbf{x}}(\mathbf{y} \mid \mathbf{x}; \theta). \tag{4.1}$$

Here, the marginal density $p_X(\mathbf{x})$ reflects the input distribution, while $p_{Y|X}(\mathbf{y} \mid \mathbf{x}; \theta)$ encapsulates the mapping from inputs to outputs under the parameterization $\theta$. Furthermore, while dropping indices for brevity, we ensure that all dependencies are clear from the context.

The family of density functions

$$\{\theta \mapsto p(\mathbf{x}, \mathbf{y}; \theta) \mid \theta \in \Theta\},$$

parametrized by $\theta$, forms a submanifold of the infinite-dimensional space of probability density functions. Here, $\Theta$ represents the parameter space. To ensure smoothness, we assume the following *regularity condition*: the partial derivatives of the probability density with respect to the parameters,

$$\frac{\partial}{\partial \theta_1} p(\mathbf{x}, \mathbf{y}; \theta), \ldots, \frac{\partial}{\partial \theta_N} p(\mathbf{x}, \mathbf{y}; \theta),$$

are linearly independent, where $\theta^T = (\theta_1, \ldots, \theta_N) \in \Theta$. This condition guarantees that the submanifold admits a tangent space at each point $p(\mathbf{x}, \mathbf{y}; \theta)$. This manifold, denoted as

$$\mathcal{S} = \{p(\mathbf{x}, \mathbf{y}; \theta) \mid \theta \in \Theta\},$$

is called a *statistical manifold* [106].

At initialization, in most practical settings, the training distribution $p(\mathbf{x}, \mathbf{z})$ is unlikely to lie exactly on the manifold $\mathcal{S}$. By adjusting the parameters $\theta$ in the neural network, the goal of the training is to minimize the proximity between the training data distribution $p(\mathbf{x}, \mathbf{z})$ and the corresponding statistical manifold $p(\mathbf{x}, \mathbf{y}; \theta)$. If the optimal parameter vector exists, it can be obtained by minimizing the *Kullback-Leibler (KL) divergence*:

$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}}(p(\mathbf{x}, \mathbf{z}) \| p(\mathbf{x}, \mathbf{y}; \theta)). \tag{4.2}$$

This optimization ensures that the model's predicted conditional distribution $p(\mathbf{y} \mid \mathbf{x}; \theta)$ aligns as closely as possible with the true data conditional distribution $p(\mathbf{z} \mid \mathbf{x})$. Consequently, the optimal parameters $\theta^*$ maximize the likelihood of the observed class labels under the model:

The Kullback-Leibler (KL) divergence between two probability distributions $p$ and $q$ over the same variable (e.g., $(\mathbf{x}, \mathbf{z})$) is defined as

$$D_{\mathrm{KL}}(p \,\|\, q) \;=\; \mathbb{E}_p\Big[\log\Big(\tfrac{p}{q}\Big)\Big] \;=\; \mathbb{E}_p\Big[\log p\Big] \;-\; \mathbb{E}_p\Big[\log q\Big]. \tag{4.3}$$

When $p$ represents the (unknown) true data distribution $p(\mathbf{x}, \mathbf{z})$ and $q$ is the model distribution $p(\mathbf{x}, \mathbf{y}; \theta)$, the KL divergence measures how far the model is from the true data distribution. Importantly, $D_{\mathrm{KL}}(p \,\|\, q) \geq 0$, with equality if and only if $p = q$ almost everywhere.

Minimizing the KL divergence in Equation (4.2) is equivalent to maximizing the expected log-likelihood of the data under the model. By expanding the KL divergence:

$$D_{\mathrm{KL}}\Big(p(\mathbf{x}, \mathbf{z}) \,\|\, p(\mathbf{x}, \mathbf{y}; \theta)\Big) \;=\; \underbrace{\mathbb{E}_{p(\mathbf{x},\mathbf{z})}\Big[\log p(\mathbf{x}, \mathbf{z})\Big]}_{\text{constant in } \theta} \;-\; \mathbb{E}_{p(\mathbf{x},\mathbf{z})}\Big[\log p(\mathbf{x}, \mathbf{y}; \theta)\Big].$$

Since the first term only depends on the true data distribution which makes it constant, minimizing $D_{\mathrm{KL}}$ is equivalent to maximizing the second term,

$$\mathbb{E}_{p(\mathbf{x},\mathbf{z})}\Big[\log p(\mathbf{x}, \mathbf{y}; \theta)\Big],$$

which is precisely the expected log-likelihood. In practice, we do not know $p(\mathbf{x}, \mathbf{z})$ exactly; instead, we use an empirical distribution derived from a training dataset $\{(\mathbf{x}_n, \mathbf{z}_n)\}_{n=1}^{N}$, making the optimization criterion

$$\frac{1}{N} \sum_{i=1}^{N} \log p\Big(\mathbf{x}_n,\, \mathbf{z}_n;\, \theta\Big),$$

which is the standard maximum log-likelihood objective [66] in statistics and machine learning.

The KL divergence is especially natural in this setting because minimizing $D_{\mathrm{KL}}$ with respect to $\theta$ corresponds directly to maximizing likelihood, giving rise to classical maximum likelihood estimation (MLE) [66]. More importantly for this thesis, the second-order Taylor expansion of $D_{\mathrm{KL}}$ around the optimum yields the FIM as the local metric, simplifying analyses of model sensitivity. This relationship will be shown in the next sub-section.

### 4.1.2 Fisher Information Matrix

Once the optimal parameters $\theta^*$ are determined by minimizing the KL divergence, we can analyze the local geometry of the statistical manifold $\mathcal{S}$. The KL divergence between the true data distribution $p(\mathbf{x}, \mathbf{z})$ and the model's predicted distribution $p(\mathbf{x}, \mathbf{y}; \theta)$ provides a natural distance measure on this manifold. In the framework of information geometry, it is a well-established result that $\mathcal{S}$ can be endowed with a *metric tensor*, turning it into a *Riemannian manifold*—sometimes called a *Neuromanifold* in neural network contexts [106]. This metric tensor arises naturally from

a perturbation perspective:

In the vicinity of $\theta^*$, the KL divergence can be approximated by a second-order Taylor expansion:

$$D_{\mathrm{KL}}\Big(p(\mathbf{x}, \mathbf{y}; \theta) \parallel p(\mathbf{x}, \mathbf{y}; \theta + \mathrm{d}\theta)\Big) \approx \frac{1}{2} \mathrm{d}\theta^\top \mathbf{F} \, \mathrm{d}\theta, \tag{4.4}$$

where the Fisher Information Matrix (FIM) $\mathbf{F}$ arises as the second derivative of the KL divergence with respect to the parameters:

$$\mathbf{F}(\theta) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p(\mathbf{x},\mathbf{y})}\Big[\nabla_\theta \log p(\mathbf{x}, \mathbf{y}; \theta) \, \nabla_\theta \log p(\mathbf{x}, \mathbf{y}; \theta)^\top\Big]. \tag{4.5}$$

This result establishes $\mathbf{F}$ as the Riemannian metric on the parameter space, capturing the curvature of the statistical manifold $\mathcal{S}$ in terms of parameter perturbations. By measuring how small changes in $\theta$ affect the model's predictive distribution, $\mathbf{F}$ provides a principled framework for analyzing the sensitivity properties of the model.

One might wonder whether different divergences can be associated with distinct geometric frameworks, which influence the corresponding optimization methods. Indeed, in Section 2.2.2, our discussion of gradient-based methods for DNNs implicitly adopts a Euclidean geometry. In this KL-based setting, performing standard gradient descent is equivalent to carrying out maximum likelihood estimation in a parameter space, assumed to have a standard Euclidean metric.

As noted above, minimizing the KL divergence amounts to maximizing the log-likelihood of the observed data under the model. Gradient descent (GD) on $\mathcal{L}(\theta) \approx -\frac{1}{N}\sum_{i=1}^{N} \log p\Big(\mathbf{x}_n, \, \mathbf{z}_n \, ; \theta\Big)$ is thus simply a numerical procedure for performing maximum likelihood estimation (MLE) in the parameter space.

Furthermore, one can transition to a *natural gradient descent* (NGD) framework by incorporating the Fisher Information Matrix (FIM) into the update rule, thereby adopting a Riemannian geometric viewpoint. In NGD, the gradient is preconditioned by the inverse of $\mathbf{F}(\theta)$, yielding [65]

$$\theta_{t+1} = \theta_t - \eta \mathbf{F}(\theta_t)^{-1} \nabla_\theta \mathcal{L}\Big(\theta_t\Big). \tag{4.6}$$

Intuitively, NGD accounts for the local curvature of the parameter manifold by rescaling gradient directions according to the model's sensitivity in each dimension. This often improves optimization stability and can lead to faster convergence in certain problems. However, computing and inverting $\mathbf{F}(\theta)$ is expensive, and becomes prohibitive for large-scale models.

Since exploring alternative optimization methods is beyond the scope of this thesis, we focus instead on how the FIM itself offers valuable insights into parameter evolution during training. We note the relevance of natural gradient methods in passing, referring interested readers to the corresponding literature [65, 67, 68] for more in-depth treatments.

### 4.1.3 Empirical Fisher Information Matrix

Although the FIM in (4.5) relies on an expectation over the true distribution $p(\mathbf{x}, \mathbf{y})$, the true distribution is rarely available in closed form. In practice, we define an *empirical* FIM by replacing the expectation $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p}$ with a sample average over a dataset $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$:

$$\widehat{\mathbf{F}}(\theta) \;=\; \frac{1}{N}\sum_{n=1}^{N}\Big[\nabla_\theta \log p(\mathbf{x}_n, \mathbf{y}_n; \theta)\Big]\Big[\nabla_\theta \log p(\mathbf{x}_n, \mathbf{y}_n; \theta)\Big]^\top. \tag{4.7}$$

As $N \to \infty$, $\widehat{\mathbf{F}}(\theta)$ converges to $\mathbf{F}(\theta)$ under standard regularity conditions.

Using Bayes' rule, we have

$$p(\mathbf{x}, \mathbf{y}; \theta) = p(\mathbf{y} \mid \mathbf{x}; \theta)\, p(\mathbf{x}),$$

and since $p(\mathbf{x})$ is independent of the model parameters $\theta$, it follows that

$$\nabla_\theta \log p(\mathbf{x}, \mathbf{y}; \theta) = \nabla_\theta \log p(\mathbf{y} \mid \mathbf{x}; \theta).$$

Therefore, Equation (4.7) can be equivalently written as

$$\widehat{\mathbf{F}}(\theta) \;=\; \frac{1}{N}\sum_{n=1}^{N}\Big[\nabla_\theta \log p(\mathbf{y}_n \mid \mathbf{x}_n; \theta)\Big]\Big[\nabla_\theta \log p(\mathbf{y}_n \mid \mathbf{x}_n; \theta)\Big]^\top. \tag{4.8}$$

A common example in machine learning assumes a Gaussian likelihood with unit variance:

$$p(\mathbf{y} \mid \mathbf{x}; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\!\Big(-\frac{1}{2\sigma^2}\|\mathbf{y} - f_\theta(\mathbf{x})\|^2\Big). \tag{4.9}$$

In practice, one might have $\sigma^2 \neq 1$; here we set $\sigma^2 = 1$ for simplicity. Consequently, the negative log-likelihood is proportional to the Mean Squared Error (MSE) loss:

$$-\log p(\mathbf{y} \mid \mathbf{x}; \theta) \propto \|\mathbf{y} - f_\theta(\mathbf{x})\|^2.$$

This formulation highlights that, from a probabilistic perspective, the loss function does more than merely measure the discrepancy between predictions and labels—it also encodes a *noise assumption*, that is, a prior belief about the error (or noise) distribution underlying the data. The model assumes that the observed outputs are generated as

$$y = f_\theta(x) + \epsilon,$$

with the error $\epsilon$ drawn from a specified distribution (e.g., a Gaussian distribution in the MSE case). This assumption allows us to derive the loss function as the negative log-likelihood of the noise model, thereby endowing the loss with the ability to capture the local curvature of the log-likelihood function—a curvature that is precisely characterized by FIM.

Substituting the Gaussian likelihood (with $\sigma^2 = 1$) into Equation (4.8) and integrating out $\mathbf{y}$ yields the metric tensor

$$\mathbf{F} := \sum_{k=1}^{C} \mathbb{E}_{\mathbf{x}\sim\rho}\Big[\nabla_\theta f_k(\mathbf{x})\, \nabla_\theta f_k(\mathbf{x})^\top\Big], \tag{4.10}$$

where $C$ is the number of classes (or outputs), $f_k$ denotes the $k$-th output of the neural network, and $\nabla_\theta f_k(\mathbf{x})$ is the Jacobian of $k$-th output with respect to all trainable parameters of the neural network.

When $N$ input samples $\mathbf{x}_n$ $(n = 1, \ldots, N)$ are available, the empirical FIM under MSE can be derived from the definition in Equation 4.7 as:

$$\mathbf{F} = \sum_{k=1}^{C} \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta f_k(\mathbf{x}_n) \nabla_\theta f_k(\mathbf{x}_n)^\top. \tag{4.11}$$

To streamline notation, we omit the hat $\widehat{\mathbf{F}}$, which typically denotes the empirical FIM, as all subsequent derivations are implicitly based on the empirical approximation. The equation 4.11 is equivalent to:

$$\mathbf{F} = \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta f_\theta(\mathbf{x}_n) \nabla_\theta f_\theta(\mathbf{x}_n)^\top. \tag{4.12}$$

where $\nabla_\theta f_\theta(\mathbf{x}_n)$ is the Jacobian matrix of the network's outputs with respect to all parameters, computed for a single input sample.

In the computation of empirical FIM, one can collect the per-sample Jacobians $\nabla_\theta f_\theta(\mathbf{x}_n)$ into a $P \times CN$ matrix:

$$\mathbf{J} = \begin{bmatrix} \nabla_\theta f_1(\mathbf{x}_1) & \nabla_\theta f_2(\mathbf{x}_1) & \ldots & \nabla_\theta f_c(\mathbf{x}_1) & \ldots & \nabla_\theta f_{c-1}(\mathbf{x}_N) & \nabla_\theta f_c(\mathbf{x}_N) \end{bmatrix},$$

where each column of $\mathbf{J}$ corresponds to $\nabla_\theta f_k(\mathbf{x}_n)$ $(k = 1, \ldots, C,\ n = 1, \ldots, N)$. Then the empirical FIM can be expressed as

$$\mathbf{F} = \frac{1}{N}\, \mathbf{J}\, \mathbf{J}^\top. \tag{4.13}$$

We consider this type of empirical metric tensor for arbitrary $N$, which may also be fixed at a constant value.

In parallel, we define

$$\mathbf{F}^* \;=\; \frac{1}{N} \mathbf{J}^\top \mathbf{J}. \tag{4.14}$$

Though $\mathbf{F}^*$ is not itself the FIM, it is closely related: by rearranging the Jacobian via matrix multiplication, one obtains a dual form. Applying the Singular Value Decomposition (SVD) of the Jacobian $\mathbf{J}$:

$$\mathbf{J} = U \Sigma V^T, \tag{4.15}$$

where $U$ and $V$ are orthogonal matrices and $\Sigma$ is a diagonal matrix of singular values, we compute:

$$\mathbf{F} = \frac{1}{N} U \Sigma \Sigma^T U^T, \quad \mathbf{F}^* = \frac{1}{N} V \Sigma^T \Sigma V^T.$$

The eigenvalues of $\mathbf{F}$ and $\mathbf{F}^*$ are determined by the same squared singular values of $\mathbf{J}$ divided by $N$, so $\mathbf{F}$ and $\mathbf{F}^*$ share the same non-zero eigenvalues, as shown in

[107].

It is noteworthy that, while the empirical Fisher Information Matrix (FIM) encapsulates local information about the neural network, its size—being a $P \times P$ matrix—renders direct computation infeasible for large-scale models. Nevertheless, as shown by Karakida et al. [102], $\mathbf{F}^*$, is the empirical Neural Tangent Kernel (NTK). The established duality between empirical FIM and empirical NTK offers an efficient alternative to compute the spectrum without incurring the computational cost of handling the full FIM. We will discuss the details of NTK in the next sub-section.

**Impact of Noise Assumptions on the FIM.** In the derivation above we assumed the standard case of independent Gaussian noise with unit variance. Under this assumption, the error $y_k - f_{k,\theta}(\mathbf{x})$ for each output is distributed as $\mathcal{N}(0,1)$, so that

$$\mathbb{E}_{\mathbf{y}|\mathbf{x}}\Big[(y_k - f_{k,\theta}(\mathbf{x}))(y_j - f_{j,\theta}(\mathbf{x}))\Big] = \delta_{kj},$$

where $\delta_{kj}$ is the Kronecker delta that selects only the diagonal terms. Hence, the FIM simplifies to Equation (4.10).

If, however, the noise is independent but with variance $\sigma^2 \neq 1$, then

$$\mathbb{E}_{\mathbf{y}|\mathbf{x}}\Big[(y_k - f_{k,\theta}(\mathbf{x}))(y_j - f_{j,\theta}(\mathbf{x}))\Big] = \sigma^2\,\delta_{kj},$$

and the empirical FIM becomes

$$\mathbf{F} = \sigma^2 \sum_{k=1}^{C} \mathbb{E}_{\mathbf{x}\sim\rho}\Big[\nabla_\theta f_{k,\theta}(\mathbf{x})\,\nabla_\theta f_{k,\theta}(\mathbf{x})^\top\Big].$$

On the other hand, if the noise terms are correlated—meaning that the errors follow a multivariate Gaussian with a full covariance matrix $\Sigma$—then

$$\mathbb{E}_{\mathbf{y}|\mathbf{x}}\Big[(y_k - f_{k,\theta}(\mathbf{x}))(y_j - f_{j,\theta}(\mathbf{x}))\Big] = \Sigma_{kj}.$$

In this case, the empirical FIM takes the more general form:

$$\mathbf{F} = \sum_{k,j=1}^{C} \mathbb{E}_{\mathbf{x}\sim\rho}\Big[\nabla_\theta f_{k,\theta}(\mathbf{x})\,\Sigma_{kj}\,\nabla_\theta f_{j,\theta}(\mathbf{x})^\top\Big],$$

which reflects the coupling between different output dimensions.

In many real-world scenarios, the noise is *heteroscedastic*, meaning its variance (or covariance) depends on the input $\mathbf{x}$. This is modeled by letting the covariance become a function $\Sigma(\theta, \mathbf{x})$. In this case, the empirical FIM generalizes to

$$\mathbf{F} = \mathbb{E}_{\mathbf{x}\sim\rho}\Big[\nabla_\theta f(\mathbf{x})^\top\,\Sigma(\theta,\mathbf{x})\,\nabla_\theta f(\mathbf{x})\Big],$$

which captures the input-dependent uncertainty directly in the geometry.

Even in numerous applications, the Gaussian assumption provides a convenient and effective approximation to the local geometry of the neuromanifold. However, it is important to note that alternative likelihoods can be adopted. For example, a uniform distribution might be assumed when noise is bounded, a Laplace distribution leads to an $L_1$ loss, and heavy-tailed distributions (e.g., Student's $t$) yield a loss with different sensitivities to outliers. In each case, the negative log-likelihood derived from the model prescribes a distinct curvature for the loss landscape. Consequently, the resulting empirical FIM will differ in its structure (for instance, by including additional weighting factors or non-quadratic behavior) to capture the inherent uncertainty of the model.

Thus, the empirical FIM is a general tool that encapsulates the local geometry of the neuromanifold for any smooth, differentiable likelihood function. The particular choice of likelihood (and its induced loss) determines how the model "sees" the data: it defines not only the performance measure but also the metric that guides the optimization dynamics. In this way, the selection of the loss function is fundamental to both model fitting and to shaping the effective geometry of the neuromanifold.

## 4.2 Neural Tangent Kernel

The Neural Tangent Kernel (NTK) [108] provides a key framework for analyzing the training dynamics of neural networks under gradient descent. It characterizes how updates to the parameters $\theta$ on one data sample influence predictions for other samples. The NTK formalism allows for the characterization of training trajectories and convergence behaviors in wide neural networks, providing critical insights into their generalization properties and scaling dynamics.

Now, we will recall the loss function and training dynamics from Section 2.2: The empirical loss function, defined in Equation 2.5, quantifies the discrepancy between predictions and true labels. The gradient of the loss with respect to the parameters $\theta$ can be expressed as:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{|\boldsymbol{D}|} \sum_{\mathbf{x}_n \in \boldsymbol{D}} \nabla_\theta f_\theta(\mathbf{x}_n) \nabla_f \ell(f_\theta(\mathbf{x}_n), \mathbf{z}_n),$$

where $\nabla_f \ell(f_\theta(\mathbf{x}_n), \mathbf{z}_n)$ denotes the gradient of the loss with respect to the model's output $f_\theta(\mathbf{x}_n)$, and $\nabla_\theta f_\theta(\mathbf{x}_n)$ is the Jacobian of the network outputs with respect to the parameters.

The parameters $\theta$ evolve under gradient descent. In the continuous-time limit, the evolution can be written as:

$$\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}(\theta).$$

Although this formulation assumes a continuous-time limit, it closely approximates the behavior of discrete gradient descent steps for small learning rates. Substituting this into the derivative of the network output $f_\theta(\mathbf{x})$, we obtain:

$$\frac{df_\theta(\mathbf{x})}{dt} = \nabla_\theta f_\theta(\mathbf{x})^T \frac{d\theta}{dt}.$$

By substitution, this becomes:

$$\frac{df_\theta(\mathbf{x})}{dt} = -\frac{1}{|\boldsymbol{D}|} \sum_{\mathbf{x}' \in \boldsymbol{D}} \nabla_\theta f_\theta(\mathbf{x})^\top \nabla_\theta f_\theta(\mathbf{x}_n) \nabla_f \ell(f_\theta(\mathbf{x}_n), \mathbf{z}_n). \tag{4.16}$$

Here the term $\nabla_\theta f_\theta(\mathbf{x})^\top \nabla_\theta f_\theta(\mathbf{x}_n)$ defines the Neural Tangent Kernel (NTK) [109]:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \nabla_\theta f_\theta(\mathbf{x})^\top \nabla_\theta f_\theta(\mathbf{x}'), \tag{4.17}$$

where $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ measures the similarity between inputs $\mathbf{x}$ and $\mathbf{x}'$ in the parameter space. The elements of the NTK at position $(m, n)$ are given by:

$$\mathbf{K}_{m,n}(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{P} \frac{\partial f_m(\mathbf{x})}{\partial \theta_p} \frac{\partial f_n(\mathbf{x}')}{\partial \theta_p}, \tag{4.18}$$

where $m, n$ mean the $m$-th and $n$-th output of the neural network.

The most significant conclusions about the NTK emerge when the width of the network approaches infinity, where the NTK demonstrates the following properties [108]:

- **Deterministic at Initialization:** The NTK depends only on the network architecture and is independent of the initialization values.
- **Constant During Training:** The NTK remains unchanged throughout training.

These properties allow the NTK to simplify the analysis of training dynamics, reducing the study of non-linear networks to linear methods. The network function can be approximated using a first-order Taylor expansion around its initialization:

$$f_\theta(\mathbf{x}) \approx f_{\theta_0}(\mathbf{x}) + \nabla_\theta f_{\theta_0}(\mathbf{x})^\top (\theta - \theta_0). \tag{4.19}$$

This linearization implies that the training dynamics in the NTK regime can be represented as:

$$\frac{df_\theta(\mathbf{x})}{dt} = -K(\mathbf{x}, \mathbf{x}')(f_\theta(\mathbf{x}') - \mathbf{y}), \tag{4.20}$$

where $K(\mathbf{x}, \mathbf{x}')$ is the NTK. This differential equation highlights how the NTK governs the evolution of the network's output towards the target $\mathbf{y}$.

Analogous to the empirical FIM, when a dataset of $N$ input samples $\{\mathbf{x}_n\}_{n=1}^N$ is available, we can define the empirical NTK by collecting the per-sample Jacobians

$\nabla_\theta f_\theta(\mathbf{x}_n)$ $(n = 1, \ldots, N)$ into the matrix $\mathbf{J}$, the *empirical* NTK matrix $\widehat{\mathbf{K}}$ is given by

$$\widehat{\mathbf{K}} = \frac{1}{N} \mathbf{J}\,\mathbf{J}^\top. \tag{4.21}$$

The empirical NTK is constructed by aggregating the Jacobians of the network outputs with respect to the parameters for $N$ data samples. This differs from the conventional NTK [62] as shown in 4.17, which measures pairwise similarities between input samples, as we focus on the self-similarity within single samples to align with the empirical FIM framework [103].

While the NTK framework is often presented in a differential-equation setting, in practice we use discrete gradient descent. It is crucial to acknowledge that the theoretical guarantees of the NTK are contingent upon employing a parameterization [57, 58] that is suitable for the infinite-width limit. In this limit, the continuous-time equations can offer valuable insights as the learning rate becomes sufficiently small, enabling parameter updates to be infinitesimal. In summary, this parameterization guarantees the existence of an infinite-width limit and ensures that the Neural Tangent Kernel (NTK) remains constant, as described above.

In practice, for sufficiently wide neural networks, the NTK becomes a more precise predictor of the training dynamics. Specifically, when the network is wide, small parameter updates have minimal effect on the feature map, causing the NTK to remain nearly constant throughout training [84]. Here, parameterization refers to how the network's parameters (weights and biases) are scaled and initialized so that signals and gradients flow in a controlled way. Parameterizations can include specific initialization schemes, output scaling factors, or other techniques (e.g., learning rate and normalization adaptations) that preserve signal propagation. Under certain schemes, the NTK remains relatively stable, but with other parameterizations [59], it can evolve significantly as parameters change, weakening or invalidating the linearized training dynamics analysis.

### 4.2.1 Analogy to Cauchy–Green Tensors in Continuum Mechanics

Having established the duality between the empirical Fisher Information Matrix (FIM) and the empirical Neural Tangent Kernel (NTK) in Section 4.1.3, we now draw a continuum-mechanics analogy to illustrate why two different matrix forms can describe equivalent geometric structures. This perspective parallels the relationship between the left and right Cauchy–Green deformation tensors [63], which represent the same physical deformation in different coordinate systems.

In continuum mechanics, a body moving in Euclidean space is described by a *deformation map*

$$\mathbf{X} \mapsto \chi(\mathbf{X}),$$

where the vector $\mathbf{X}$ denotes the position of a material point in the body at a reference time, and $\chi(\mathbf{X})$ is its position at the current time. From the *deformation gradient*

$\mathbf{E} = \nabla_{\mathbf{X}} \chi(\mathbf{X})$, one can define two fundamental tensor fields:

- **Right Cauchy–Green Tensor:** $\mathbf{C} = \mathbf{E}^\top \mathbf{E}$, commonly used in the *Lagrangian* (material) description [61].
- **Left Cauchy–Green Tensor:** $\mathbf{B} = \mathbf{E}\,\mathbf{E}^\top$, naturally suited to an *Eulerian* (spatial) perspective [61].

Although $\mathbf{C}$ and $\mathbf{B}$ appear as distinct matrices ($\mathbf{E}^\top\mathbf{E}$ vs. $\mathbf{E}\mathbf{E}^\top$), they describe the equivalent geometric content of the deformation. In particular, they share the same non-zero eigenvalues (the squares of the principal stretches), and their maximal eigenvectors determine the directions of maximal stretching, i.e. directions in which the material is deformed the most [86]. One can thus think of them as measuring curvature or strain in two different, but equivalently valid, coordinate frameworks.

**Parallel to FIM and NTK.** As described in Section 4.1.3, $\mathbf{F}$ and $\mathbf{K}$ also share the same non-zero eigenvalues, arising from interchanging the order of multiplication $\mathbf{J}\,\mathbf{J}^\top$ versus $\mathbf{J}^\top\,\mathbf{J}$. Here, we omit the hat $\widehat{\mathbf{K}}$, which denotes the empirical NTK. In close analogy to the Cauchy–Green tensors:

$$\underbrace{\mathbf{F} = \frac{1}{N}\mathbf{J}\,\mathbf{J}^\top}_{\text{Left metric tensor}} \quad \text{and} \quad \underbrace{\mathbf{K} = \frac{1}{N}\mathbf{J}^\top\mathbf{J}}_{\text{Right metric tensor}}$$

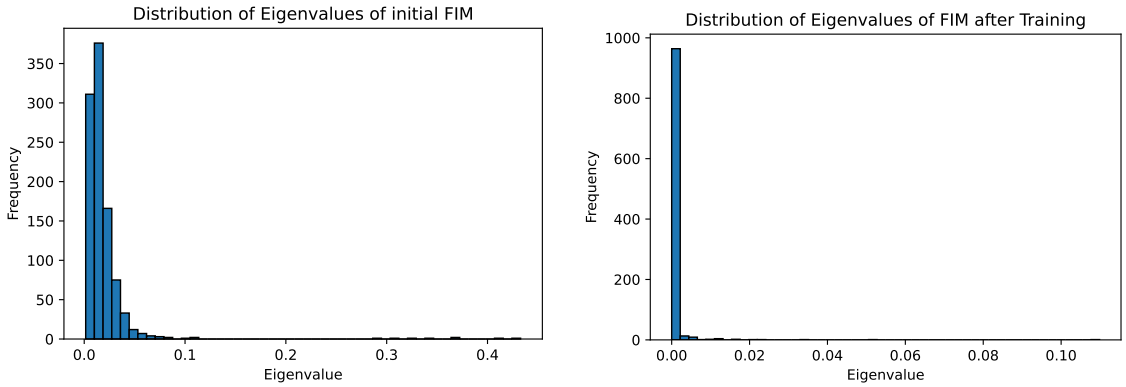
represent two ways of analyzing the same infinitesimal response of the model to parameter perturbations. Since $\mathbf{J}$ is typically not square, $\mathbf{F}$ and $\mathbf{K}$ differ in their dimensionalities. Nonetheless, when focusing on the diagonal of the NTK—that is, considering the case where the same input is evaluated—both matrices share the same non-zero eigenvalues. In the continuum-mechanics analogy, these two forms ($\mathbf{E}\mathbf{E}^\top$ vs. $\mathbf{E}^\top\mathbf{E}$) correspond to Eulerian vs. Lagrangian viewpoints, whereas in deep learning they correspond to:

- FIM($\mathbf{F}$): Focuses on how changes in $\theta$ (the parameter space) map directly to changes in the outputs $f_\theta(\mathbf{x})$.
- NTK ($\mathbf{K}$): Emphasizes pairwise interactions among different data samples in the output space via a kernel perspective, i.e. $\nabla_\theta f_\theta(\mathbf{x})^\top \nabla_\theta f_\theta(\mathbf{x})$.

Since both $\mathbf{F}$ and $\mathbf{K}$ share the same non-zero eigenvalues, they inherit the equivalent core geometric content regarding how a neural network "stretches" small parameter perturbations in terms of its output.

This analogy is particularly salient for understanding the *spectral anisotropy* of deep network training: the principal eigenvalues and eigenvectors of $\mathbf{F}$ (and equivalently $\mathbf{K}$) reveal the most influential directions along which the model evolves. In the next section, we show how this shared spectrum underlies the model's strongly anisotropic behavior.

## 4.3 Anisotropy in the Spectrum of the Fisher Information Matrix




**(a)** FIM Spectrum at Initialization **(b)** FIM Spectrum after Training

**Figure 4.1:** Distribution of the top 1000 non-zero eigenvalues of the FIM, illustrating its highly anisotropic spectrum both at initialization and after training.

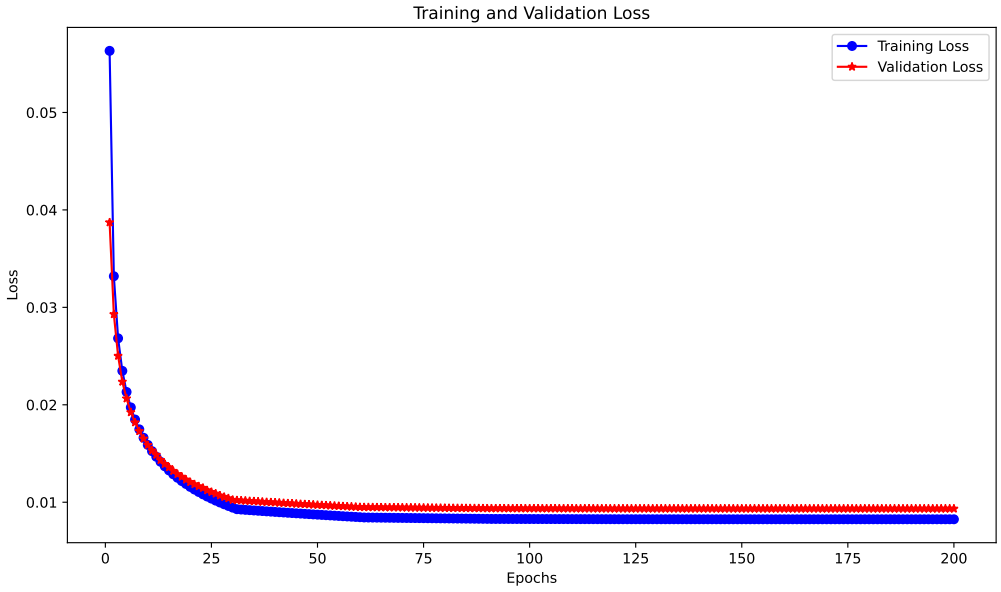



**Figure 4.2:** Training and Validation (Test) Loss Curves. The model achieves convergence after approximately 75 epochs.

The Fisher Information Matrix (FIM) is symmetric and positive semi-definite [104], ensuring that all its eigenvalues are non-negative. Empirically, the eigen-decomposition of the FIM in deep neural networks consistently reveals a *highly anisotropic* spectrum, where only a few eigenvalues are significantly larger than the rest. In our experiments, most eigenvalues remain clustered near zero after training, whereas a small subset separates clearly from the bulk. For a concrete example, Figure 4.1

shows top 1000 non-zero eigenvalues at initialization vs. post-training. We observe a pronounced spike in the largest eigenvalues, underscoring strong anisotropy.

Simultaneously, Figure 4.2 shows that the model converges (training and test losses flatten) after around 75 epochs. In the ensuing discussions and experiments, the FIM, denoted as $\mathbf{F}$, corresponds to Equation 4.13.

**Trace Ratio as a Measure of Anisotropy.** To quantify the extent of anisotropy, we define the *trace ratio*:

$$T_k = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{N} \lambda_j}, \tag{4.22}$$

where $\lambda_j$ denotes the $j$-th eigenvalue of the FIM (ordered from largest to smallest), and $k$ is the number of top eigenvalues considered. When $T_k$ approaches 1, it indicates that a small subset of eigenvalues dominates the spectrum. In our setup, $T_{100}$ increases from 0.360 at initialization to 0.999 after training , indicating a concentration in a few direction.

Notably, the trace ratio is invariant under uniform scaling of the eigenvalues. Suppose we uniformly rescale the network outputs by a factor $\alpha > 0$:

$$f_\theta(x) \mapsto \alpha\, f_\theta(x).$$

In an MSE-based likelihood, each gradient in (4.11) acquires a factor of $\alpha$, the Fisher Information Matrix (FIM) is multiplied by $\alpha^2$ factor which makes each eigenvalue scaled by $\alpha^2$. However, since $T_k$ depends on ratios of eigenvalues, this global $\alpha^2$ factor cancels out. Hence, we do not conflate genuine changes in *shape* of the spectrum with trivial global scaling.

### 4.3.1 The Dual Roles of FIM in DNNs

Regarding the significance of anisotropy in FIM in deep neural networks (DNNs), FIM can be understood from two perspectives. Firstly, FIM plays a role in the geometric structure of DNNs' evolution under certain optimization methods. Secondly, FIM's connection to the Hessian of the loss function at minima enables its leverage.

**FIM as a Local Metric.** One intuitive explanation for why so few eigenvalues dominate stems from the FIM's role as a local metric on the parameter manifold. The quadratic infinitesimal change in the network's output, measured via the Euclidean norm, can also be expressed in terms of the FIM:

$$\mathbb{E}\Big[\|f_{\theta+\mathrm{d}\theta}(\mathbf{x}) - f_\theta(\mathbf{x})\|^2\Big] \approx \mathrm{d}\theta^\top \mathbf{F}\, \mathrm{d}\theta. \tag{4.23}$$

where the expectation is taken over data points (or a local distribution), and $\mathrm{d}\theta$ is an infinitesimal change in the parameters. This equation is typically derived via a second-order (or probabilistic) expansion, so it holds *locally*. In differential-geometric terms, $\mathbf{F}$ acts like a Riemannian metric that measures how parameter

steps $\mathrm{d}\theta$ get *stretched* in output space. Directions corresponding to large eigenvalues of $\mathbf{F}$ imply that even small parameter updates can yield substantial changes in model output. Consequently, gradient-based optimization learns *mostly* along these large-eigenvalue directions, driving anisotropic training behavior.

**FIM as Hessian via the Gauss–Newton Matrix.** Beyond the geometric interpretation, an important theoretical link to Hessian-based analyses [105] arises under a mean-squared error (MSE) loss near well-converged solutions. The Hessian of the MSE loss function can be written as

$$\nabla^2 \mathcal{L}(\theta) = \underbrace{\frac{1}{N} \sum_{n=1}^{N} \nabla_\theta f_\theta(\mathbf{x}_n) \, \nabla_\theta f_\theta(\mathbf{x}_n)^\top}_{\mathbf{G}(\theta)\text{(Gauss–Newton Matrix)}} + \underbrace{\frac{1}{N} \sum_{n=1}^{N} r_n \, \nabla_\theta^2 f_\theta(\mathbf{x}_n)}_{\mathbf{R}(\theta)} .$$

where $r_n = f_\theta(\mathbf{x}_n) - \mathbf{z}_n$ is the residual at sample $\mathbf{x}_n$. If the residuals are small (i.e., the model nearly fits the data), then $\mathbf{R}(\theta)$ becomes negligible, making $\nabla^2 \mathcal{L}(\theta)$ well-approximated by the Gauss–Newton matrix $\mathbf{G}(\theta)$ [67]. Under these conditions, $\nabla^2 \mathcal{L}(\theta)$ in fact coincides with the empirical FIM, thus linking Hessian-based curvature analyses to the FIM's anisotropic structure. In other words, if the network is close to a good minimum (e.g., the minimun in Figure 4.2) under MSE loss functions, the FIM reveals the same eigenvectors and eigenvalues as Hessian.

When gradient descent (GD) or stochastic gradient descent (SGD) operates in the vicinity of such a solution, the Hessian—as a second-order approximation of the loss—governs how curvature (and thus gradient changes) distribute across parameter-space directions. Its eigenvalues quantify how rapidly the loss changes when moving along their corresponding eigenvectors. Large eigenvalues correspond to high curvature: even minor displacements along these directions induce substantial changes in the loss, producing stronger gradient signals. As a result, each update step heavily reinforces these directions, while directions associated with smaller eigenvalues (lower curvature) receive comparatively minor updates.

**Gradient Projection Visualization.** This anisotropy of FIM spectrum arises naturally from the local, second-order relationship between parameter changes and model outputs, and becomes particularly pronounced near minima of MSE-like objectives, where the Hessian and FIM largely coincide. Consequently, training unfolds primarily within a low-dimensional subspace determined by these top eigenvectors.

Here, we delve deeper into the convergence of training dynamics into the subspace through the alignment of the gradients of the parameters with the dominant eigenvectors of the FIM spectrum. This is achieved through a novel visualization technique.

To visualize how each dominant eigenvector $\mathbf{v}_{\theta_i}$ interacts with the model's parameters updates from the loss function on new data, we construct an *eigenfunction* $u_{\theta_i}$, which maps an input $\mathbf{x}$ to the normalized projection of its Jacobian onto $\mathbf{v}_{\theta_i}$. Specifically, let $f_k(\mathbf{x})$ be the network's $k$-th output (e.g., corresponding to the correct label

in a classification setting), and denote $\nabla_\theta f_k(\mathbf{x}) \in \mathbb{R}^P$ the gradient of that output with respect to all $P$ trainable parameters. Then, for the $i$-th dominant eigenvector $\mathbf{v}_{\theta_i}$ with eigenvalue $\lambda_{\theta_i}$, we define

$$u_{\theta_i}(\mathbf{x}) = \frac{1}{\sqrt{\lambda_{\theta_i}}} \Big\langle \mathbf{v}_{\theta_i},\, \nabla_\theta f_k(\mathbf{x}) \Big\rangle. \tag{4.24}$$

Here, $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathbb{R}^P$, $\{\lambda_{\theta_i}, \mathbf{v}_{\theta_i}\}_{i=1}^r$ are the top-$r$ eigenpairs of $\mathbf{F}$ (sorted by descending eigenvalues). Intuitively, $u_{\theta_i}(\mathbf{x})$ captures how strongly the gradient for input $\mathbf{x}$ aligns with the $i$-th principal direction of the empirical FIM. The normalization factor $\frac{1}{\sqrt{\lambda_{\theta_i}}}$ is derived in Appendix A.2 to ensure $u_{\theta_i}$ has unit norm in $L^2(\rho)$.

Since $\{\mathbf{v}_{\theta_i}\}$ represent the most influential directions for parameter updates, $u_{\theta_i}(\mathbf{x})$ can be seen as a spectral embedding of the gradient at $\mathbf{x}$. By evaluating $u_{\theta_i}(\mathbf{x})$ on the test set, one can identify dominant directions of variation. Because high absolute values of $u_{\theta_i}(\mathbf{x})$ indicate that small parameter changes along $\mathbf{v}_{\theta_i}$ significantly alter the model's prediction for $\mathbf{x}$. Hence, the construction in (4.24) provides a direct bridge between the eigenvectors in the subspace (as reflected in the FIM's anisotropic spectrum) and practical, input-dependent gradient.

Figure 4.3 shows the results of applying t-SNE dimensionality reduction [114] to the output of the last hidden layer, projecting it into a two-dimensional space. The visualization reveals ten distinct clusters, each corresponding to a different class label—a phenomenon known as *Neural Collapse* [113]. Since this gives us the **position** of each data point in the test set within the two-dimensional output space, we use these coordinates to visualize the projection between the eigenvectors and gradients.

We visualize this projection in the same neural collapse (output) space as shown in Fugre 4.3 across the entire test set in Figure 4.4. The visualization highlights the behavior of eigenvectors corresponding to different eigenvalues: the 1-st, 10-th, 100-th, and 500-th largest eigenvectors. Notably, the eigenvector associated with the largest eigenvalue exhibits the most significant response to changes in the data, suggesting its dominant role in capturing the gradient dynamics.

This observation aligns with the anisotropy of the FIM spectrum, where a small number of top eigenvectors dominate the training dynamics. These dominant eigenvectors likely encode critical information about the data structure and play a key role in determining the network's generalization capability. Conversely, the less significant eigenvectors show minimal response, consistent with their relatively lower contribution to the overall gradient evolution.

Empirical evidence further indicates that this subspace, determined by the top eigenvalues and eigenvectors, can remain stable throughout extended training [110]. Consequently, the anisotropic structure of both the Hessian and the FIM near an MSE optimum helps explain how high-dimensional neural networks effectively focus their

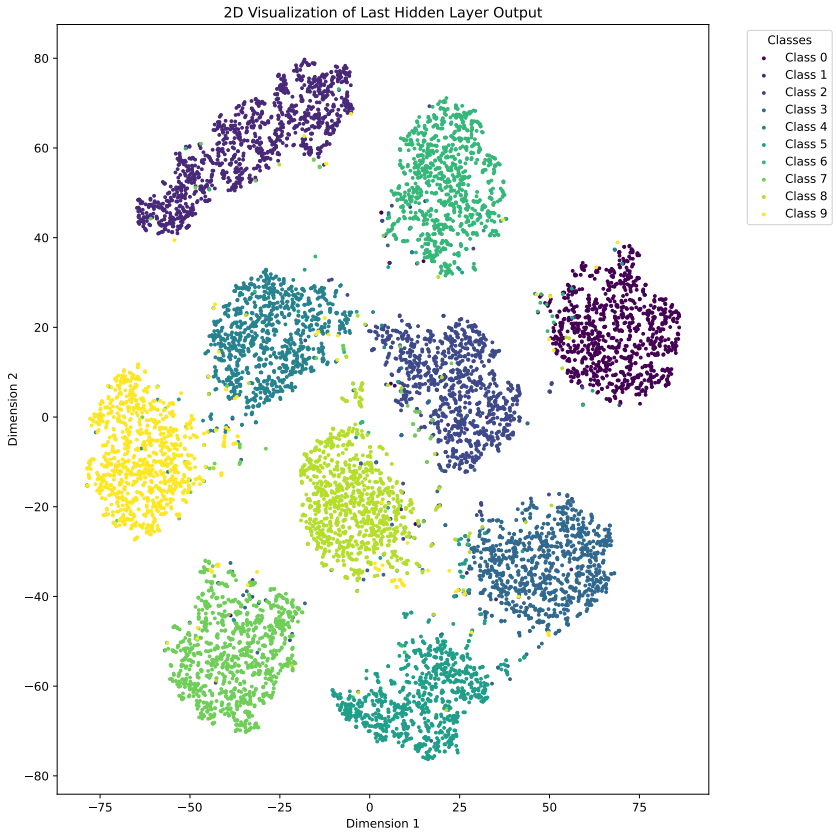



**Figure 4.3:** The Dimensional Reduction of the Last Hidden Layer Output into a Two-Dimensional Space.

capacity on a relatively small number of fundamental parameter directions.

This anisotropic picture raises several important questions, which we address in the following subsections:

1. Is the anisotropy ubiquitous across MLPs?
2. What are the implications of this anisotropy for generalization, particularly with respect to the emergence of sub-networks in MLPs, as observed in our previous findings?

### 4.3.2 Spectral Statistics of the Fisher Information Matrix in MLPs

The spectral statistics of the Fisher Information Matrix (FIM) offer valuable insights into the anisotropy of the FIM in deep neural networks (DNNs), particularly MLPs. These statistics play a crucial role in addressing the first question: the ubiquitousness of the anisotropy. Karakida et al. [102] introduced a mean-field approach to analyze the spectrum of the FIM, focusing on its mean, variance, and maximum eigenvalues. For sufficiently wide neural networks, where the width $M \gg 1$, the eigenvalue

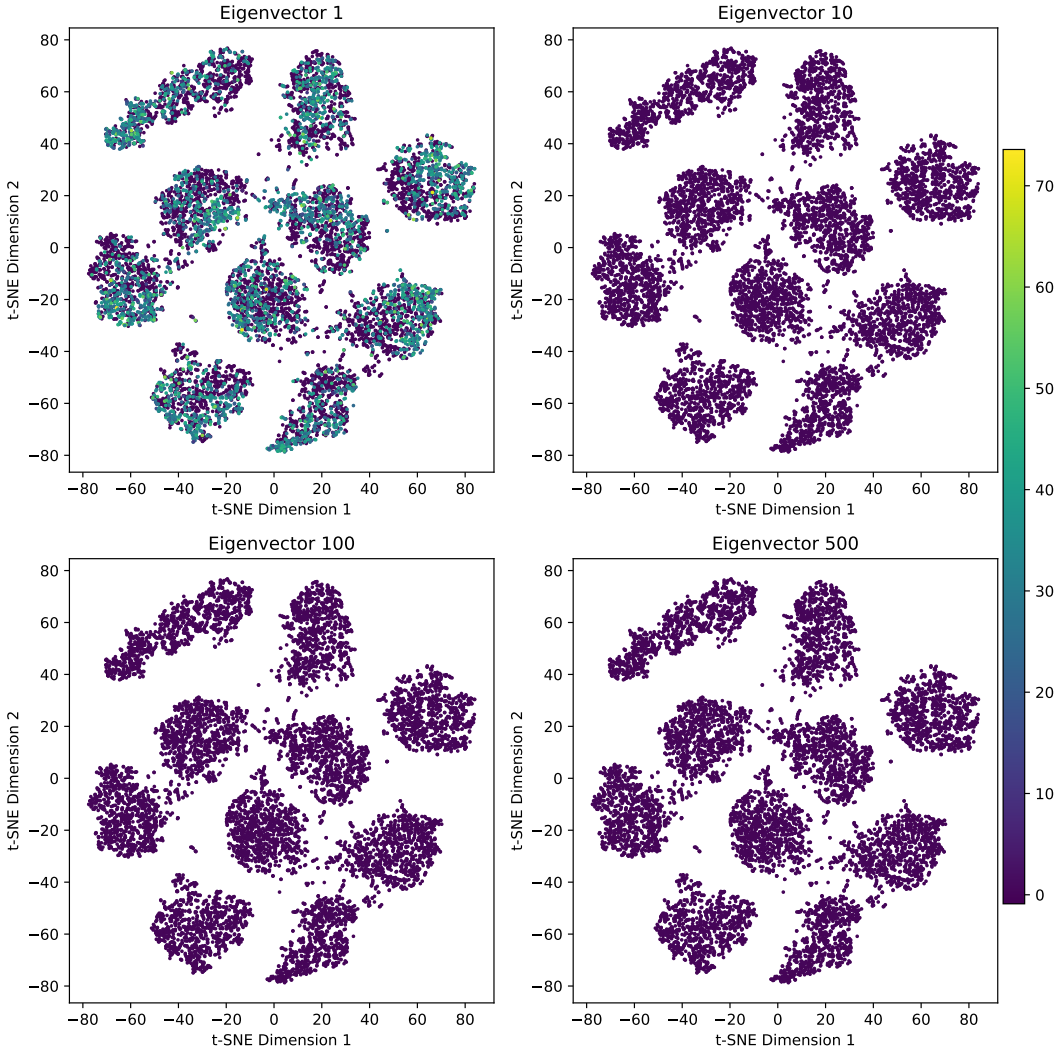



**Figure 4.4:** The Gradient Projection into the 1st, 10th, 100th, and 500th Largest Eigenvectors

statistics of the FIM can be asymptotically evaluated as:

$$\bar{\lambda} \sim \frac{\kappa_1 C}{M}, \quad s_\lambda \sim \alpha \left( \frac{N-1}{N} \kappa_2^2 + \frac{\kappa_1^2}{N} \right) C, \quad \lambda_{\max} \sim \alpha \left( \frac{N-1}{N} \kappa_2 + \frac{\kappa_1}{N} \right) M,$$

where $\alpha := \sum_{l=1}^{L-1} \alpha_l \alpha_{l-1}$. In their proof, they introduce $M_l = \alpha_l M$ as the width of each layer in the neural network, $\alpha_l$ is a constant. However, in my previous setting, this specific decomposition of $M_l$ is not necessary. $C$ is the number of classes in the datasets, $N$ is the number of training samples, $\kappa_1$ and $\kappa_2$ are positive constants derived from order parameters in the mean-field theory approach.

This analysis reveals a critical result: the shape of the eigenvalue spectrum becomes pathologically distorted as the network width increases.

Specifically:

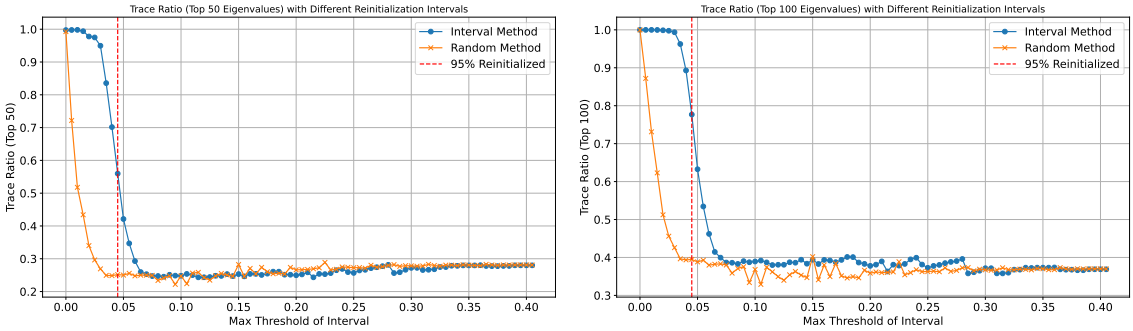
- The **mean eigenvalue**, $\bar{\lambda}$, decreases asymptotically as $\mathcal{O}(1/M)$.
- The **variance**, $s_\lambda$, remains at $\mathcal{O}(1)$.
- The **largest eigenvalue**, $\lambda_{\max}$, grows asymptotically as $\mathcal{O}(M)$.

These findings imply that, in sufficiently wide network with random initialization the majority of eigenvalues asymptotically approach zero as the network width grows. At the same time, a small number of eigenvalues are significantly larger, corresponding to $\mathcal{O}(M)$.

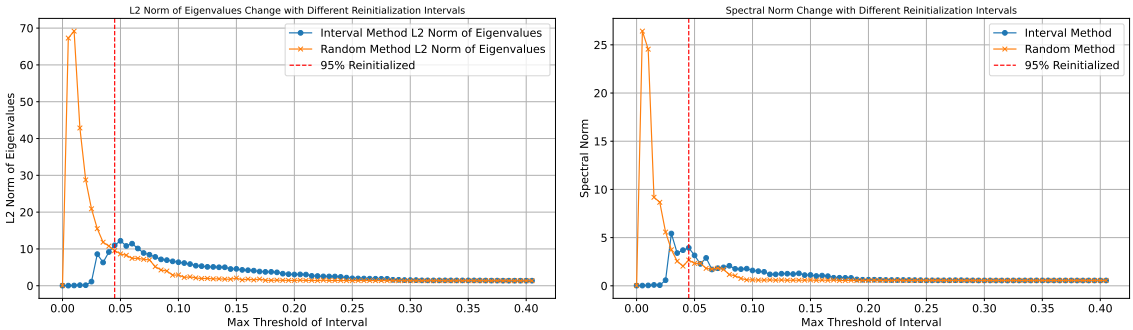
Given the statistical properties of the spectrum derived in this sub-section, this anisotropy is ubiquitous in multilayer pcerceptrons (MLPs), particularly in sufficiently wide architectures.

### 4.3.3 The Sub-network and Anisotropy

As demonstrated in the preceding chapter, the performance drop upon re-initialization provides a pruning-like criterion for identifying sub-networks in a well-trained DNN that capture essential aspects of the underlying data. A natural question, then, is whether this anisotropic related to the emergence of sub-networks. In the following, we investigate this empirically.




**(a)** Trace Ratio of Top 50 Eigenvalues **(b)** Trace Ratio of Top 100 Eigenvalues




**(c)** L2 Norm of Eigenvalues **(d)** Spectral Norm of Eigenvalues Change

**Figure 4.5:** Spectral Anisotropy shown by parameter-wise re-initialization.

In our experiments, we adopt a similar setting from Algorithm 1 but replace the

accuracy metric with the trace ratio. Using this approach, we analyze how re-initialization affects the anisotropy of the FIM spectrum. In other words, for each re-initialization of the same range of parameters as in the experiment of Figure 3.4 for the entire model, we display the trace ratio of the FIM corresponding to each re-initialization. The results are shown in Figure 4.5.

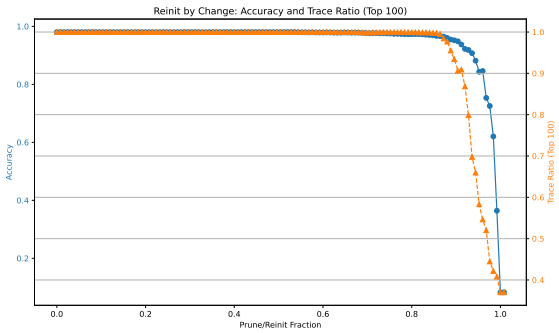
The two plots in the first row of Figure 4.5 depict the anisotropy of the FIM spectrum under re-initialization, focusing on $T_{50}$ and $T_{100}$, respectively. The results indicate that when relatively less critical parameters are reinitialized determined by the absolute change criterion the anisotropy remains largely unchanged, even after most of the parameters have been reset to their initial values. In contrast, random parameter selection significantly reduces the anisotropy from the start of training. This clearly show the direct connection between the critical parameters (the sub-network) and the anisotropy of the spectrum.

The second row shows the $L_2$ norm and spectral norm of the FIM. The spectral norm is defined as the square root of the maximum eigenvalue of the matrix. Both norms initially increase as the re-initialization interval grows, followed by a gradual decrease toward the initialization norms as most parameters are progressively re-initialized to their initial values. These findings suggest that the emergence of the sub-network plays a dual role: not only is it essentially related with the anisotropy of the FIM, but it also contributes significantly to robustness.

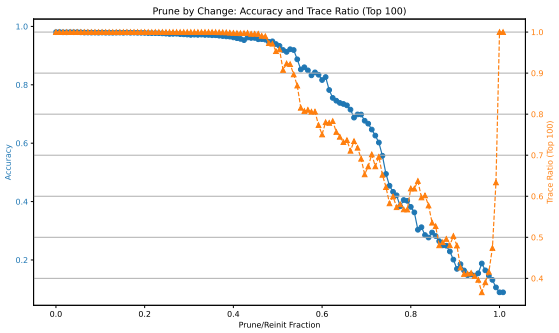
On the one hand, the plots of trace ratio demonstrate that the trace ratio decreases only when the parameters that form the sub-network are re-initialized. Only after the sub-network has been replaced does the anisotropy of the FIM decrease. Two facts regarding the evolution of parameters in DNNs are presented in the thesis: first, a sub-network with a significantly lower number of parameters compared to the original one emerges during training and concentrates the model's capacity; second, the evolution or update of a neural network occurs along the directions of the top eigenvectors corresponding to the largest eigenvalues. The aforementioned experiment connects these two concepts, suggesting that empirically, the spectral anisotropy of the FIM can also be attributed to the sub-network.

On the other hand, robustness is reflected in the reduction of the norms of spectrum during training. The smaller spectrum norms indicates that the model converges to a flatter and more stable minimum, consistent with the concept of flatness bias. Flat minima are less susceptible to parameter perturbations, making the model more resilient to noise and better equipped to maintain stable performance across varying inputs.

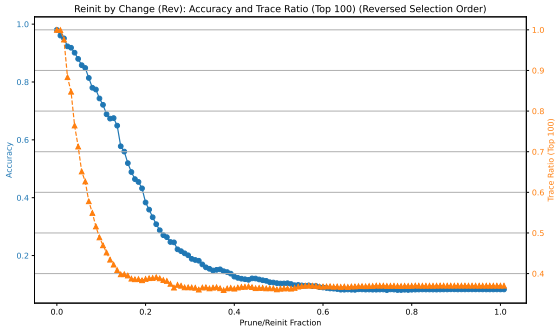
One may question whether the observed decay in anisotropy is an artifact of our experimental setup, given that we re-initialized most of the parameters prior to observing it. To address this, we conducted an additional experiment to isolate the effects of pruning and re-initialization with the same pipeline as the experiment in Figure 3.6. Instead, here we employed the one additional trace ratio curve.



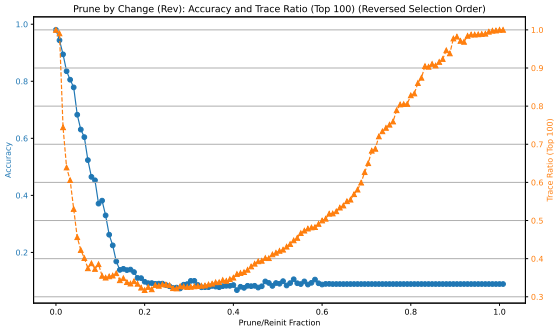

**(a)** Top 100 Eigenvalues by Re-initialization




**(b)** Top 100 Eigenvalues by Pruning




**(c)** Top 100 Eigenvalues by Re-initialization (decresing direction)




**(d)** Top 100 Eigenvalues by Pruning (decresing direction)

**Figure 4.6:** Spectral Anisotropy: trace ratio and accuracies shown by re-initialization and pruning.

Figure 4.6 presents the results of this experiment. In the first row, the plots illustrate pruning and re-initialization performed in increasing order of absolute parameter change, i.e., the parameters with smaller changes are pruned or reinitialized first. Conversely, the second row shows results where the operations are conducted in reverse order: the parameters with larger changes are targeted first. Sub-figures (b) and (d) illustrate an increase in the trace ratio at the end of pruning, attributed to the removal of all parameters in the model.

From these experiments, we observe a clear correlation between the sub-networks and anisotropy. Specifically, when critical parameters are disturbed—whether by pruning or re-initialization—the anisotropy of the FIM spectrum consistently declines, accompanied by a corresponding drop in model performance. These results suggest that the anisotropy is closely tied to the generalization capabilities of neural networks via the sub-network.

In this subsection, we demonstrate a strong correlation between the anisotropy of the FIM spectrum and the emergence of sub-networks. It is plausible that the

anisotropy is directly linked to the existence of sub-networks. In the next two subsections, we will further explore the structure of the subspace where the training dynamics converge. Understanding this subspace could provide deeper insights into the mechanisms underlying neural network generalization, particularly the correlation with the sub-network.

### 4.3.4 The Top Eigenvectors

Given the deep connection between sub-networks and the anisotropy of neural networks, it is worthwhile to explore the internal structure of the subspace which is spanned by the top eigenvectors which are corresponding to the largest eigenvalues in the FIM spectrum. To this end, we conducted a series of experiments to analyze the structure of the eigenvectors.

We are motivated by the results presented in Figures 4.5 and 4.6 to investigate whether the eigenvectors exhibit related low-dimensional characteristics with the sub-networks. More precisely, we measure the distributions of the components of specific eigenvectors after re-initialization. The relative magnitudes of the components in the eigenvectors serve as a measure of the significance of each parameter dimension in a specific eigen-direction. They can elucidate the contribution of each parameter dimension to the principal direction of curvature or sensitivity within the model. A large positive or negative value in a particular parameter location suggests that the eigen-direction is heavily influenced by that parameter. Given that sub-networks are composed of those critical parameters, we aim to establish a more direct connection here. We employed two metrics, kurtosis and Shannon entropy, for our experiments.

***Kurtosis*** [112] quantifies the tailedness of a distribution. For a vector, where a few elements are substantially larger than the remainder, the kurtosis value is elevated. The kurtosis of a vector $v = [v_1, v_2, \ldots, v_n]$ is given by:

$$K(v) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{v_i - \mu}{\sigma}\right)^4 - 3, \tag{4.25}$$

where:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} v_i$$

is the mean of the vector, and

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \mu)^2}.$$

is the standard deviation of the vector. The subtraction of $-3$ ensures that the kurtosis of a normal distribution is zero.

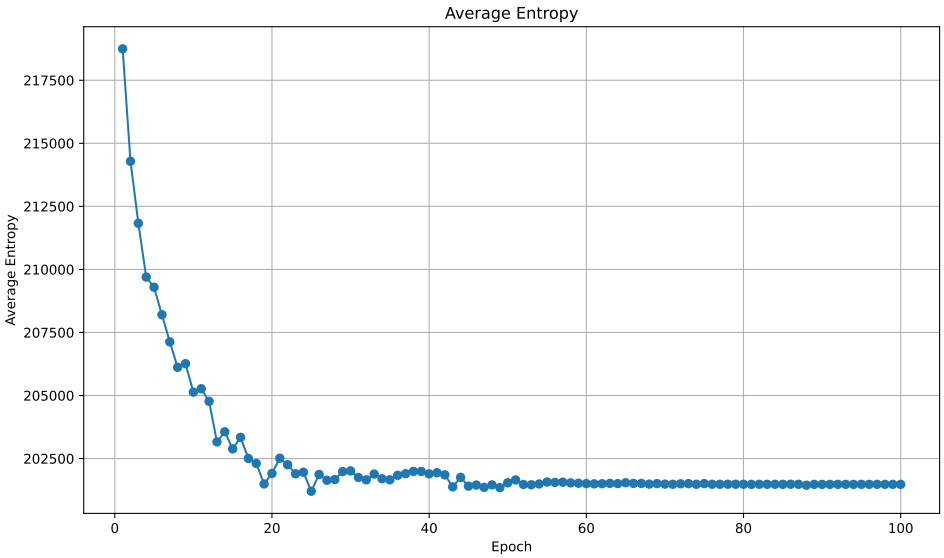



**Figure 4.7:** Average Shannon Entropy of Top 100 Eigenvectors Across 100 Training Epochs.

***Shannon Entropy*** measures the disorder or uncertainty in the magnitude distribution of the eigenvector elements. For a vector, where the magnitude distribution is concentrated on a few elements, the entropy value is low. The entropy of the normalized magnitudes $p_i$ of the eigenvector elements $v_i$ is defined as:

$$H(v) = -\sum_{i=1}^{n} p_i \log(p_i), \tag{4.26}$$

where:

$$p_i = \frac{|v_i|}{\sum_{j=1}^{n} |v_j|}$$

is the normalized magnitude of each element, treated as a probability distribution.To ensure numerical stability when computing $\log(p_i)$ for very small values of $p_i$, a small constant $\epsilon$ can be added:

$$H(v) = -\sum_{i=1}^{n} p_i \log(p_i + \epsilon). \tag{4.27}$$

In some presented results, the entropy is shown using the effective rank [111] for a better visualization, which is derived from the Shannon entropy. The effective rank provides an interpretable measure of dimensionality and is computed as:

$$\text{erank} = \exp(H(v)). \tag{4.28}$$

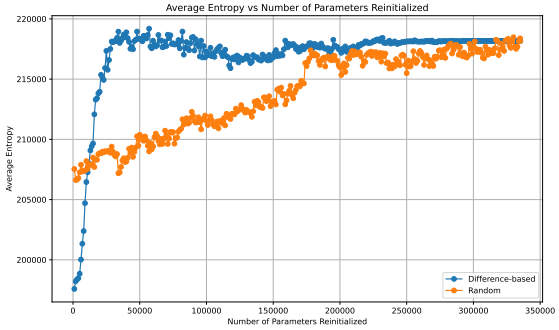
Figure 4.7 illustrates a clear trend in which the average effective ranks of the top eigenvectors, corresponding to the 100 largest eigenvalues, consistently decrease throughout training until convergence to a minimum. This trend aligns with the loss curves shown in Figure 4.2. As previously discussed, at the end of training, the dynamics of neural networks converge into a subspace spanned by the top eigenvectors.

This observation presented here suggest that, as training progresses, the neural network increasingly operates within a lower-dimensional subspace. This empirically indicates the emergence of a low-dimensional sub-network by the end of training.

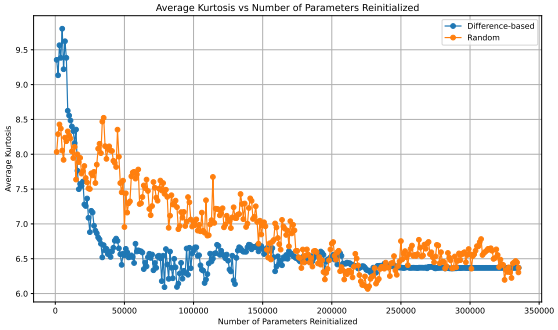
We employ a re-initialization procedure based on absolute change and random selection to analyze two metrics: effective rank and kurtosis. In this process, we repeatedly re-initialize the parameters and calculate the corresponding metrics for the eigenvectors of FIM until we have re-initialized the entire model. The results, presented in Figure 4.8, demonstrate that when the parameters with the largest changes are reinitialized based on the absolute change criterion, where we commence with the parameters that have the most significant alterations as the setting from the experiments in the second row of Figure 4.6, the entropy exhibits a substantial increase. Conversely, the kurtosis exhibits a clear decrease for both the top 50 and 100 eigenvectors. In contrast, for the same experiments pipeline based on random selection, there is a distinct difference: there are no rapid increases or decreases for both of the metrics.

These findings further support the notion that parameter evolution is concentrated in a limited number of parameters, not solely by examining the performance through pruning or re-initialization, but also from the distribution of top eigenvectors. In other words, the subspace where the training dynamics converge to may be rooted in the sub-networks within a well-trained neural network. Because both of the metrics demonstrate the declines of the concentration from a few components in the top eigenvectors, when we re-initialize the neural networks.

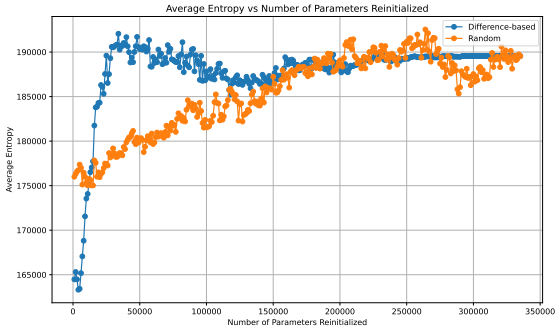
We can now address Question 2 posed at the beginning of this section: Our experimental findings strongly suggest that the anisotropy of the FIM spectrum is tightly linked to the formation of sub-networks in multilayer perceptrons. Moreover, if we examine the components of the top eigenvectors, we can find that the corresponding top eigenvectors also exhibit patterns of sparsity that align with these sub-networks. An open question is whether there exists an underlying mechanism within the neuromanifold that governs the network's evolution, not only along the eigenvectors associated with the largest eigenvalue, but also within a subspace spanned by these dominant eigenvectors that may exhibit a sparse structure. Finally the top eigenvectors have an extremely significant and direct effect on the gradient propagation during training, thereby exerting a significant influence on how the model learns and generalizes.



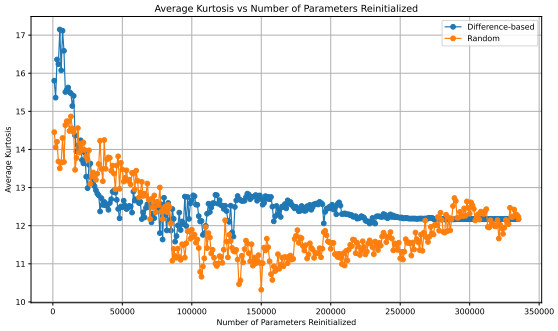

**(a)** Average Effective Rank of Top 100 eigenvectors




**(b)** Average Kurtosis of Top 100 eigenvectors




**(c)** Average Effective Rank of Top 50 eigenvectors




**(d)** Average Kurtosis of Top 50 eigenvectors

**Figure 4.8:** Average Effective Rank and Kurtosis of Top 100/50 eigenvectors by Parameter-wise Re-initialization

# 5

# Conclusion

The final aspect that may be of particular interest is the unique role of the critical layer. Several recent studies [88, 89] have shown that iterative magnitude pruning can uncover the structures of receptive fields (RFs) in the first layer of multilayer perceptrons, resembling those found in convolutional neural networks (CNNs) [97] and the visual cortex of animals [90]. The primary function of RFs is to extract specific features from the data. This observation suggests that receptive fields can naturally emerge in MLPs to identify and distinguish specific features in real-world datasets. Moreover, this finding highlights the critical role of the sparse sub-network structures we identified in the first layer, as these substructures serve a similar role to the convolutional layers in CNNs. Building on this, an intriguing research direction would be to investigate the similarities and differences between the sub-networks identified by various methods, with a particular focus on the correlation of their receptive fields.

This thesis explores the intricate relationships between the sub-networks and the anisotropy of the FIM spectrum in deep neural networks (DNNs), as well as investigate the structure of the subspace spanned by the eigenvectors corresponding to the largest eigenvalues, focusing on the fundamental architecture of multilayer perceptrons (MLPs). The findings presented in this work offer insights into the underlying mechanisms that enable over-parameterized DNNs to achieve remarkable performance.

We employed parameter-wise re-initialization and pruning experiments, which highlighted that the existence of critical sub-networks is the cause of the structural symmetry breaking. These sub-networks retain the primary learning capacity of the model. They align with the Lottery Ticket Hypothesis (LTH), emphasizing the importance of specific parameter configurations. The sub-networks exhibited sparse structures, suggesting that effective feature extraction and hierarchical learning occur within a constrained subspace of the parameter space.

By introducing the neuromanifold framework and analyzing the spectrum of Fisher Information Matrix (FIM), this work provided a geometric perspective on the sub-networks of DNNs. The highly anisotropic spectrum of the FIM revealed that the training dynamics of DNNs predominantly evolve along a limited number of eigen-directions.The experimental connections discovered between the spectrum of FIM and the sub-network of neural networks reveal that the sub-network exhibits a strong correlation with spectral anisotropy. Furthermore, the subspace spanned by the

top eigenvectors is directly associated with the low-dimensional nature of the subnetwork. Through a novel visualization technique, we directly demonstrate that the dominant influence of the top eigenvectors on the parameter evolution in the parameter space is evident.

These findings collectively advance our understanding of how DNNs achieve both efficiency and effectiveness in solving complex learning tasks. The interplay between low-dimensional subspaces, parameter criticality, and optimization dynamics suggests promising directions for future research, including:

- Developing improved pruning method based on re-initialization.
- Explore the relationship between the layer-wise FIM and the gradient projection, to see how hidden layers' output interplays with the subspace.
- Explore how sparsity evolves in conjunction with existing results on the evolution of anisotropy in the FIM spectrum with training.

In conclusion, this thesis investigates the hidden structures of deep neural networks, an important step towards developing more robust and generalizable models. By applying principles of information geometry and spectral analysis, we have gained insights into the intricate mechanisms that drive modern deep learning. These findings lay a foundation for future research to further investigate and refine our understanding of the complex dynamics underlying DNNs.

# Bibliography


[1] Zhang, C., Bengio, S., & Singer, Y. (2022). Are All Layers Created Equal? *Journal of Machine Learning Research*, 23(67), 1–28. Retrieved from `http://jmlr.org/papers/v23/20-069.html`.

[2] Wu, L., Zhu, Z., & E, W. (2017). Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes. *CoRR*, abs/1706.10239. Retrieved from `http://arxiv.org/abs/1706.10239`.

[3] Chiang, P., Ni, R., Miller, D. Y., Bansal, A., Geiping, J., Goldblum, M., & Goldstein, T. (2023). Loss Landscapes are All You Need: Neural Network Generalization Can Be Explained Without the Implicit Bias of Gradient Descent. *The Eleventh International Conference on Learning Representations*. Retrieved from `https://openreview.net/forum?id=QC10RmRbZy9`.

[4] Anderson, P. W. (1972). More is different. *Science*, *177*(4047), 393-396. Retrieved from `https://api.semanticscholar.org/CorpusID:34548824`.

[5] Wikipedia contributors. (n.d.). Complex system. *Wikipedia, The Free Encyclopedia*. Retrieved from `http://en.wikipedia.org/wiki/Complex_system`. (Accessed: March 3, 2025).

[6] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323, 533–536. Retrieved from `https://api.semanticscholar.org/CorpusID:205001834`.

[7] Chatterjee, S., & Zielinski, P. (2022). On the Generalization Mystery in Deep Learning. Retrieved from `https://arxiv.org/abs/2203.10036`.

[8] Neyshabur, B., Tomioka, R., Salakhutdinov, R., & Srebro, N. (2017). Geometry of Optimization and Implicit Regularization in Deep Learning. *CoRR*, abs/1705.03071. Retrieved from `http://arxiv.org/abs/1705.03071`.

[9] Vardi, G. (2022). On the Implicit Bias in Deep-Learning Algorithms. Retrieved from `https://arxiv.org/abs/2208.12591`.

[10] Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

[11] Weng, L. (2019). Are Deep Neural Networks Dramatically Overfitted? *lilianweng.github.io*. Retrieved from `https://lilianweng.github.io/posts/2019-03-14-overfit/`.

[12] Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. *Annual Review of Neuroscience*, *24*, 1193-1216. Retrieved from `https://api.semanticscholar.org/CorpusID:147618`.

[13] Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America*

*A, Optics and Image Science*, *4*(12), 2379-2394. Retrieved from https://api.semanticscholar.org/CorpusID:1600874.

[14] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[15] Liu, Y., Zhang, K., Li, Y., Yan, Z., Gao, C., Chen, R., Yuan, Z., Huang, Y., Sun, H., Gao, J., He, L., & Sun, L. (2024). Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*. Retrieved from https://arxiv.org/abs/2402.17177.

[16] Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). Large language models: A survey. *arXiv preprint arXiv:2402.06196*. Retrieved from https://arxiv.org/abs/2402.06196.

[17] Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., Bodenstein, S. W., Evans, D. A., Hung, C. C., O'Neill, M., Reiman, D., Tunyasuvunakool, K., Wu, Z., Žemgulytė, A., Arvaniti, E., Beattie, C., Bertolli, O., Bridgland, A., Cherepanov, A., Congreve, M., Cowen-Rivers, A. I., Cowie, A., Figurnov, M., Fuchs, F. B., Gladman, H., Jain, R., Khan, Y. A., Low, C. M. R., Perlin, K., Potapenko, A., Savy, P., Singh, S., Stecula, A., Thillaisundaram, A., Tong, C., Yakneen, S., Zhong, E. D., Zielinski, M., Žídek, A., Bapst, V., Kohli, P., Jaderberg, M., Hassabis, D., & Jumper, J. M. (2024). Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, *630*(8016), 493-500. https://doi.org/10.1038/s41586-024-07487-w.

[18] Askr, H., Elgeldawi, E., Aboul Ella, H., Elshaier, Y. A. M. M., Gomaa, M. M., & Hassanien, A. E. (2023). Deep learning in drug discovery: An integrative review and future challenges. *Artificial Intelligence Review*, *56*(7), 5975-6037. https://doi.org/10.1007/s10462-022-10306-1.

[19] Eddington, A. (1939). *The philosophy of physical science*. Tarner Lectures 1938. Cambridge University Press.

[20] Roberts, D. A. (2021). Why is AI hard and physics simple? *arXiv preprint arXiv:2104.00008*. Retrieved from https://arxiv.org/abs/2104.00008.

[21] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., & Sohl-Dickstein, J. (2016). On the Expressive Power of Deep Neural Networks. Retrieved from https://arxiv.org/abs/1606.05336.

[22] Chaudhari, P., & Soatto, S. (2018). Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. *arXiv preprint arXiv:1710.11029*. Retrieved from https://arxiv.org/abs/1710.11029.

[23] Mandt, S., Hoffman, M. D., & Blei, D. M. (2018). Stochastic gradient descent as approximate Bayesian inference. *arXiv preprint arXiv:1704.04289*. Retrieved from https://arxiv.org/abs/1704.04289.

[24] Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the International Conference on Ma-*

*chine Learning (ICML)*. Retrieved from https://api.semanticscholar.org/CorpusID:2178983.

[25] Su, W., Boyd, S., & Candès, E. J. (2015). A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. *arXiv preprint arXiv:1503.01243*. Retrieved from https://arxiv.org/abs/1503.01243.

[26] Danilova, M., & Malinovsky, G. (2021). Averaged heavy-ball method. *arXiv preprint arXiv:2111.05430*. Retrieved from https://arxiv.org/abs/2111.05430.

[27] Simsekli, U., Sagun, L., & Gurbuzbalaban, M. (2019). A tail-index analysis of stochastic gradient noise in deep neural networks. *arXiv preprint arXiv:1901.06053*. Retrieved from https://arxiv.org/abs/1901.06053.

[28] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*. Retrieved from http://arxiv.org/abs/1503.03585.

[29] Jung, G., Alkemade, R. M., Bapst, V., Coslovich, D., Filion, L., Landes, F. P., Liu, A. J., Pezzicoli, F. S., Shiba, H., Volpe, G., Zamponi, F., Berthier, L., & Biroli, G. (2025). Roadmap on machine learning glassy dynamics. *Nature Reviews Physics*, *7*(2), 91–104. https://doi.org/10.1038/s42254-024-00791-4.

[30] Pennington, J., Schoenholz, S. S., & Ganguli, S. (2018). The emergence of spectral universality in deep networks. *arXiv preprint arXiv:1802.09979*. Retrieved from https://arxiv.org/abs/1802.09979.

[31] Stanford University. (2024). Convolutional Neural Networks for Visual Recognition. Retrieved from https://cs231n.github.io/convolutional-networks/.

[32] Su, J. (2018, June). Optimization algorithms from a dynamics perspective (I): From SGD to momentum acceleration. Retrieved from https://www.spaces.ac.cn/archives/5655.

[33] Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring Generalization in Deep Learning. *CoRR*, abs/1706.08947. Retrieved from http://arxiv.org/abs/1706.08947.

[34] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533-536. Retrieved from https://api.semanticscholar.org/CorpusID:205001834.

[35] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, 521, 436-444. https://doi.org/10.1038/nature14539.

[36] Pearce, T., Brintrup, A., & Zhu, J. (2021). Understanding Softmax Confidence and Uncertainty. Retrieved from https://arxiv.org/abs/2106.04972.

[37] LeCun, Y., & Cortes, C. (2005). The MNIST Database of Handwritten Digits. Retrieved from https://api.semanticscholar.org/CorpusID:60282629.

[38] Liu, Y., Gao, Y., & Yin, W. (2020). An Improved Analysis of Stochastic Gradient Descent with Momentum. *Advances in Neural Information Processing Systems*, 33, 18261–18271.

[39] Raschka, S. (2025). FAQ: Lottery Ticket Hypothesis. Retrieved from https://sebastianraschka.com/faq/docs/lottery-ticket.html (Accessed: 2025-01-17).

[40] Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks. *CoRR*, abs/1707.09564. Retrieved from `http://arxiv.org/abs/1707.09564`.

[41] Dinh, L., Pascanu, R., Bengio, S., & Bengio, Y. (2017). Sharp Minima Can Generalize For Deep Nets. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1019–1028). PMLR. Retrieved from `https://proceedings.mlr.press/v70/dinh17b.html`.

[42] Zhang, S., Reid, I., Valle Pérez, G., & Louis, A. A. (2021). Why Flatness Correlates With Generalization For Deep Neural Networks. *CoRR*, abs/2103.06219. Retrieved from `https://arxiv.org/abs/2103.06219`.

[43] Reich, S. (1999). Backward Error Analysis for Numerical Integrators. *SIAM Journal on Numerical Analysis*, 36, 1549–1570. Retrieved from `https://api.semanticscholar.org/CorpusID:15179418`.

[44] Ixaru, L. G., & Vanden Berghe, G. (2004). Runge-Kutta Solvers for Ordinary Differential Equations. Retrieved from `https://api.semanticscholar.org/CorpusID:116795115`.

[45] Ramasinghe, S., Macdonald, L. E., Farazi, M., Saratchandran, H., & Lucey, S. (2023). How Much Does Initialization Affect Generalization? In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th International Conference on Machine Learning* (Vol. 202, pp. 28637–28655). PMLR. Retrieved from `https://proceedings.mlr.press/v202/ramasinghe23a.html`.

[46] Smith, L. N. (2017). Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 464–472). https://doi.org/10.1109/WACV.2017.58

[47] Wu, Y., Liu, L., Bae, J., Chow, K.-H., Iyengar, A., Pu, C., Wei, W., Yu, L., & Zhang, Q. (2019). Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 1971–1980). https://doi.org/10.1109/BigData47090.2019.9006104

[48] Smith, L. N., & Topin, N. (2018). Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. Retrieved from `https://arxiv.org/abs/1708.07120`.

[49] Neyshabur, B. (2017). Implicit Regularization in Deep Learning. *CoRR*, abs/1709.01953. Retrieved from `https://arxiv.org/abs/1709.01953`.

[50] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding Deep Learning Requires Rethinking Generalization. Retrieved from `https://arxiv.org/abs/1611.03530`.

[51] Suh, N., & Cheng, G. (2024). A Survey on Statistical Theory of Deep Learning: Approximation, Training Dynamics, and Generative Models. Retrieved from `https://arxiv.org/abs/2401.07187`.

[52] He, F., & Tao, D. (2021). Recent Advances in Deep Learning Theory. Retrieved from `https://arxiv.org/abs/2012.10931`.

[53] Hochreiter, S., & Schmidhuber, J. (1997). Flat Minima. *Neural Computation*, 9, 1–42. Retrieved from https://api.semanticscholar.org/CorpusID:733161.

[54] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. Retrieved from https://arxiv.org/abs/1609.04836.

[55] Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2020). Sharpness-Aware Minimization for Efficiently Improving Generalization. *arXiv preprint*, arXiv:2010.01412. Retrieved from https://arxiv.org/abs/2010.01412.

[56] Barrett, D. G. T., & Dherin, B. (2020). Implicit Gradient Regularization. *CoRR*, abs/2009.11162. Retrieved from https://arxiv.org/abs/2009.11162.

[57] Chizat, L., Oyallon, E., & Bach, F. (2020). On Lazy Training in Differentiable Programming. Retrieved from https://arxiv.org/abs/1812.07956.

[58] Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., & Pennington, J. (2020). Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12). https://doi.org/10.1088/1742-5468/abc62b.

[59] Geiger, M., Petrini, L., & Wyart, M. (2021). Landscape and Training Regimes in Deep Learning. *Physics Reports*. Retrieved from https://api.semanticscholar.org/CorpusID:234848577.

[60] Hornik, K. (1991). Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4, 251–257. Retrieved from https://api.semanticscholar.org/CorpusID:7343126.

[61] Batchelor, G. K. (1968). An Introduction to Fluid Dynamics. Retrieved from https://api.semanticscholar.org/CorpusID:80869282.

[62] Novak, R., Sohl-Dickstein, J., & Schoenholz, S. S. (2022). Fast Finite Width Neural Tangent Kernel. Retrieved from https://arxiv.org/abs/2206.08720.

[63] Fülöp, T. (2022). Compatibility Condition for the Eulerian Left Cauchy-Green Deformation Tensor Field. Retrieved from https://arxiv.org/abs/2108.01567.

[64] Hornik, K., Stinchcombe, M. B., & White, H. L. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2, 359–366. Retrieved from https://api.semanticscholar.org/CorpusID:2757547.

[65] Amari, S. (1998). Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2), 251–276. https://doi.org/10.1162/089976698300017746

[66] Murphy, K. P. (2013). *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA.

[67] Martens, J. (2014). New Insights and Perspectives on the Natural Gradient Method. *Journal of Machine Learning Research*, 21, 146:1–146:76. Retrieved from https://api.semanticscholar.org/CorpusID:10284405.

[68] Santambrogio, F. (2016). Euclidean, Metric, and Wasserstein Gradient Flows: An Overview. Retrieved from https://arxiv.org/abs/1609.03890.

[69] Frankle, J., & Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. Retrieved from https://arxiv.org/abs/1803.03635.

[70] Roberts, D. A., Yaida, S., & Hanin, B. (2022). Frontmatter. In *The principles of deep learning theory: An effective theory approach to understanding neural networks* (pp. i–iv). Cambridge University Press.

[71] Baskerville, N. P. (2023). Random matrix theory and the loss surfaces of neural networks. *arXiv preprint arXiv:2306.02108*. Retrieved from https://arxiv.org/abs/2306.02108.

[72] Schoenholz, S. S., Gilmer, J., Ganguli, S., & Sohl-Dickstein, J. (2017). Deep information propagation. *arXiv preprint arXiv:1611.01232*. Retrieved from https://arxiv.org/abs/1611.01232.

[73] Bahri, Y., Kadmon, J., Pennington, J., Schoenholz, S., Sohl-Dickstein, J. N., & Ganguli, S. (2020). Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*. Retrieved from https://api.semanticscholar.org/CorpusID:273727294.

[74] Baratin, A., George, T., Laurent, C., Hjelm, R. D., Lajoie, G., Vincent, P., & Lacoste-Julien, S. (2021). Implicit Regularization via Neural Feature Alignment. Retrieved from https://arxiv.org/abs/2008.00938.

[75] Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. Retrieved from https://arxiv.org/abs/1811.03378.

[76] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. Retrieved from https://arxiv.org/abs/1502.01852.

[77] Ruder, S. (2016). An Overview of Gradient Descent Optimization Algorithms. *CoRR*, abs/1609.04747. Retrieved from http://arxiv.org/abs/1609.04747.

[78] Samuel L. Smith and Pieter-Jan Kindermans and Chris Ying and Quoc V. Le (2017). Don't Decay the Learning Rate, Increase the Batch Size. *CoRR*, abs/1711.00489. Retrieved from http://arxiv.org/abs/1711.00489.

[79] Li, H., Li, C., Xue, M., Fang, G., Zhou, S., Feng, Z., Wang, H., Wang, Y., Cheng, L., Song, M., & Song, J. (2024). PruningBench: A Comprehensive Benchmark of Structural Pruning. Retrieved from https://arxiv.org/abs/2406.12315.

[80] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. Retrieved from https://arxiv.org/abs/1605.08695.

[81] Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., & Chavez-Urbiola, E. A. (2023). Loss Functions and Metrics in Deep Learning. Retrieved from https://arxiv.org/abs/2307.02694.

[82] Leclerc, G., & Madry, A. (2020). The Two Regimes of Deep Network Training. Retrieved from https://arxiv.org/abs/2002.10376.

[83] Masters, D., & Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks. Retrieved from https://arxiv.org/abs/1804.07612.

[84] Seleznova, M., & Kutyniok, G. (2022). Analyzing Finite Neural Networks: Can We Trust Neural Tangent Kernel Theory? Retrieved from https://arxiv.org/abs/2012.04477.

[85] Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in Deep Learning: Pruning and Growth for Efficient Inference and Training

in Neural Networks. *Journal of Machine Learning Research*, 22(241), 1–124. Retrieved from http://jmlr.org/papers/v22/21-0366.html.

[86] Storm, L., Linander, H., Bec, J., Gustavsson, K., & Mehlig, B. (2023). Finite-time Lyapunov Exponents of Deep Neural Networks. *Physical Review Letters*, 132(5), 057301. Retrieved from https://api.semanticscholar.org/CorpusID:259224472.

[87] Bonnaire, T., Ghio, D., Krishnamurthy, K., Mignacco, F., Yamamura, A., & Biroli, G. (2024). High-dimensional non-convex landscapes and gradient descent dynamics. *Journal of Statistical Mechanics: Theory and Experiment*, *2024*(10), 104004. https://doi.org/10.1088/1742-5468/ad2929.

[88] Pellegrini, F., & Biroli, G. (2022). Neural Network Pruning Denoises the Features and Makes Local Connectivity Emerge in Visual Tasks. In *Proceedings of the International Conference on Machine Learning*. Retrieved from https://api.semanticscholar.org/CorpusID:250340597.

[89] Redman, W. T., Wang, Z., Ingrosso, A., & Goldt, S. (2024). On How Iterative Magnitude Pruning Discovers Local Receptive Fields in Fully Connected Neural Networks. Retrieved from https://arxiv.org/abs/2412.06545.

[90] Hubel, D. H., & Wiesel, T. N. (1962). Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex. *The Journal of Physiology*, 160. Retrieved from https://api.semanticscholar.org/CorpusID:17055992.

[91] Saleem, T. J., Ahuja, R., Prasad, S., & Lall, B. (2024). Insights into the Lottery Ticket Hypothesis and Iterative Magnitude Pruning. Retrieved from https://arxiv.org/abs/2403.15022.

[92] Lin, H. W., Tegmark, M., & Rolnick, D. (2017). Why Does Deep and Cheap Learning Work So Well? *Journal of Statistical Physics*, 168(6), 1223–1247. https://doi.org/10.1007/s10955-017-1836-5.

[93] Kumar, S. K. (2017). On Weight Initialization in Deep Neural Networks. Retrieved from https://arxiv.org/abs/1704.08863.

[94] Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling Modern Machine-Learning Practice and the Classical Bias–Variance Trade-Off. *Proceedings of the National Academy of Sciences*, 116(32), 15849–15854. https://doi.org/10.1073/pnas.1903070116

[95] Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., ... Zdeborová, L. (2019). Machine Learning and the Physical Sciences. *Reviews of Modern Physics*, 91(4). https://doi.org/10.1103/revmodphys.91.045002.

[96] He, F., & Tao, D. (2021). Recent Advances in Deep Learning Theory. Retrieved from https://arxiv.org/abs/2012.10931.

[97] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999–7019.

[98] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the 30th International Conference on Machine Learning*, 28(3), 1139–1147.

[99] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[100] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798-1828. https://doi.org/10.1109/TPAMI.2013.50.

[101] Neyshabur, B., Tomioka, R., & Srebro, N. (2015). In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*. Retrieved from `https://arxiv.org/abs/1412.6614`.

[102] Karakida, R., Akaho, S., & Amari, S. (2019). Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach. Retrieved from `https://arxiv.org/abs/1806.01316`.

[103] Karakida, R., Akaho, S., & Amari, S. (2020). Pathological Spectra of the Fisher Information Metric and Its Variants in Deep Neural Networks. Retrieved from `https://arxiv.org/abs/1910.05992`.

[104] Amari, S., & Nagaoka, H. (2000). Methods of Information Geometry. Retrieved from `https://api.semanticscholar.org/CorpusID:116976027`.

[105] Kunstner, F., Balles, L., & Hennig, P. (2020). Limitations of the Empirical Fisher Approximation for Natural Gradient Descent. Retrieved from `https://arxiv.org/abs/1905.12558`.

[106] Calin, O. (2020). Neuromanifolds. *Deep Learning Architectures*. Retrieved from `https://api.semanticscholar.org/CorpusID:240689529`.

[107] Kanatani, K. (2021). Linear Algebra for Pattern Processing: Projection, Singular Value Decomposition, and Pseudoinverse. In *Linear Algebra for Pattern Processing*. Retrieved from `https://api.semanticscholar.org/CorpusID:249416969`.

[108] Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural Tangent Kernel: Convergence and Generalization in Neural Networks. *CoRR*, abs/1806.07572. Retrieved from `http://arxiv.org/abs/1806.07572`.

[109] Weng, L. (2022, September). Some Math behind Neural Tangent Kernel. *Lil'Log*. Retrieved from `https://lilianweng.github.io/posts/2022-09-08-ntk/`.

[110] Gur-Ari, G., Roberts, D. A., & Dyer, E. (2018). Gradient Descent Happens in a Tiny Subspace. *CoRR*, abs/1812.04754. Retrieved from `http://arxiv.org/abs/1812.04754`.

[111] Roy, O., & Vetterli, M. (2007). The Effective Rank: A Measure of Effective Dimensionality. In *2007 15th European Signal Processing Conference* (pp. 606–610). Retrieved from `https://api.semanticscholar.org/CorpusID:12184201`.

[112] DeCarlo, L. T. (1997). On the Meaning and Use of Kurtosis. *Psychological Methods*, 2, 292–307. Retrieved from `https://api.semanticscholar.org/CorpusID:18829068`.

[113] Kothapalli, V. (2023). Neural Collapse: A Review on Modelling Principles and Generalization. Retrieved from `https://arxiv.org/abs/2206.04041`.

[114] Cai, T. T., & Ma, R. (2022). Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data. *arXiv preprint*, arXiv:2105.07536. Retrieved from `https://arxiv.org/abs/2105.07536`.

# A

# Appendix

## A.1 Supplemental Experimental Results

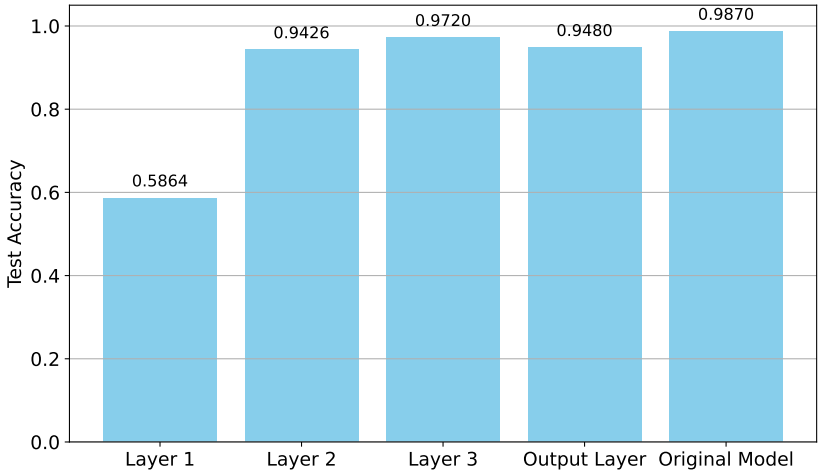



**Figure A.1:** Layer-wise results of re-initialization for FCN with Random Gaussian initialization.

| Layer | Initial weights (Mean, SD) | Final weights (Mean, SD) | Correlation Coef |
|---|---|---|---|
| Hidden 1 | 0.000, 0.050 | 0.000, 0.054 | 0.930 |
| Hidden 2 | 0.000, 0.050 | 0.003, 0.058 | 0.906 |
| Hidden 3 | 0.000, 0.050 | 0.004, 0.058 | 0.921 |
| Output | 0.002, 0.049 | 0.002, 0.158 | 0.753 |

**Table A.1:** Statistics of initial and final weights, and the correlation coefficient between them for each layer. Mean is mean value of the weights; SD is the standard deviation of the weights. Correlation Coefficient measures the linear correlation between the same layer's weights before and after training.

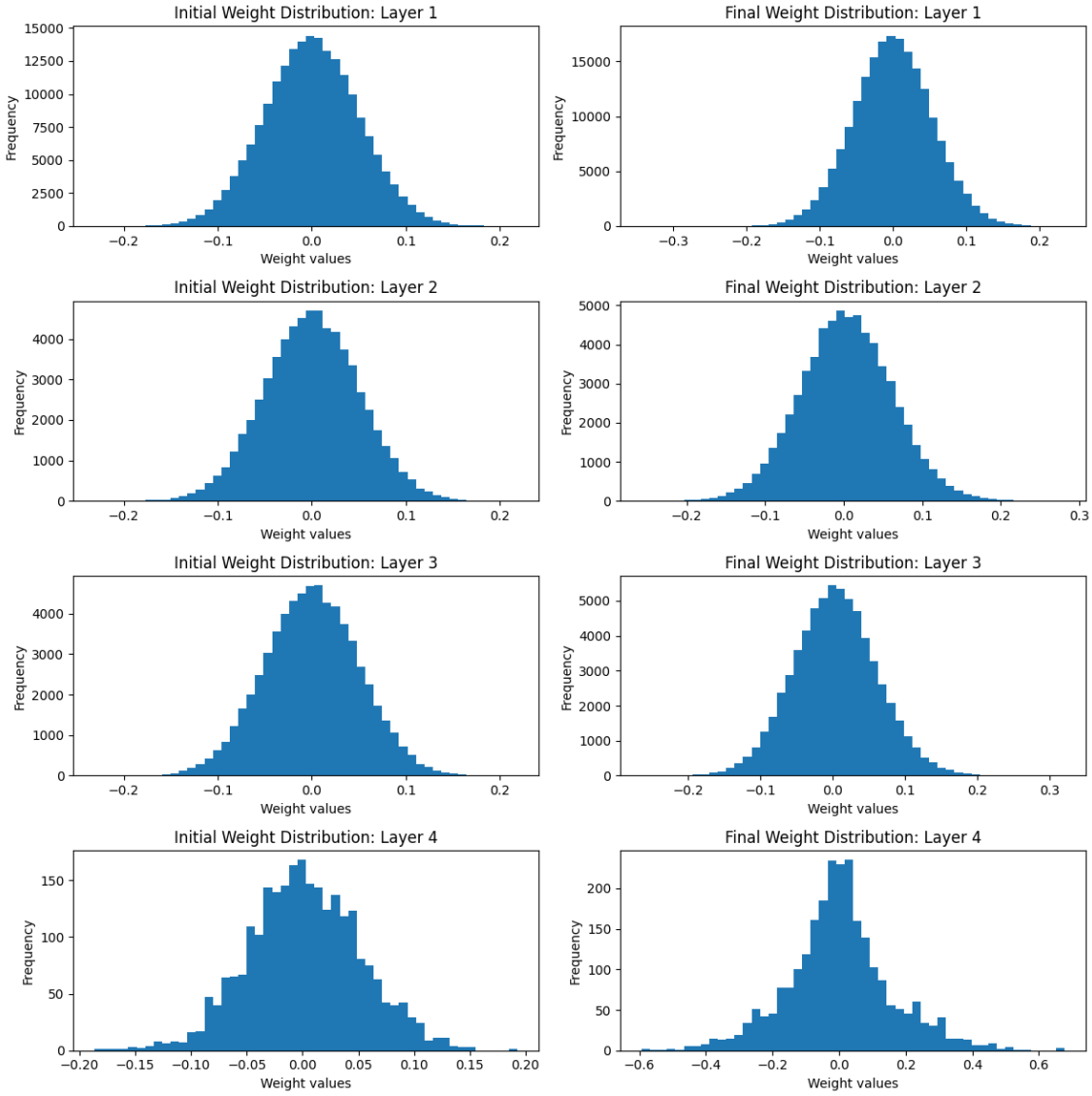



**Figure A.2:** Weights Distribution before and after training for every layer.

| Interval | Percentage |
|---|---|
| 0.00000 to 0.00800 | 53.97% |
| 0.00800 to 0.01600 | 16.48% |
| 0.01600 to 0.02400 | 10.45% |
| 0.02400 to 0.03200 | 6.79% |
| 0.03200 to 0.04000 | 4.39% |
| 0.04000 to 0.04800 | 2.80% |
| 0.04800 to 0.05600 | 1.77% |
| 0.05600 to 0.06400 | 1.12% |
| 0.06400 to 0.07200 | 0.73% |
| 0.07200 to 0.08000 | 0.44% |
| 0.08000 to 0.08800 | 0.30% |
| 0.08800 to 0.09600 | 0.19% |
| 0.09600 to 0.10400 | 0.12% |
| 0.10400 to 0.11200 | 0.08% |
| 0.11200 to 0.12000 | 0.06% |
| 0.12000 to 0.12800 | 0.05% |
| 0.12800 to 0.13600 | 0.03% |
| 0.13600 to 0.14400 | 0.03% |
| 0.14400 to 0.15200 | 0.03% |
| 0.15200 to 0.16000 | 0.02% |
| 0.16000 to 0.16800 | 0.01% |
| 0.16800 to 0.17600 | 0.01% |
| 0.17600 to 0.18400 | 0.01% |
| 0.18400 to 0.19200 | 0.01% |
| 0.19200 to 0.20000 | 0.01% |
| 0.20000 to 0.20800 | 0.01% |
| 0.20800 to 0.21600 | 0.01% |
| 0.21600 to 0.22400 | 0.01% |
| 0.22400 to 0.23200 | 0.01% |
| 0.23200 to 0.24000 | 0.01% |
| 0.24000 to 0.24800 | 0.01% |
| 0.24800 to 0.25600 | 0.01% |

**Table A.2:** Different absolute change interval and its corresponding parameters' percentage of the whole model.

## A.2 Normalization of the Eigenfunction

For completeness, we provide here the derivation showing why the function

$$u_{\theta_i}(x) \;=\; \frac{1}{\sqrt{\lambda_{\theta_i}}} \left\langle \mathbf{v}_{\theta_i}, \nabla_\theta f_k(x) \right\rangle$$

has unit norm in $L^2(\rho)$.

Recall that the empirical Fisher Information Matrix (FIM) is defined as

$$\mathbf{F} \;=\; \mathbb{E}_{x\sim\rho}\Big[\nabla_\theta f_k(x)\,\nabla_\theta f_k(x)^\top\Big].$$

Assume $\mathbf{v}_{\theta_i}$ is an eigenvector of $\mathbf{F}$ with eigenvalue $\lambda_{\theta_i}$, i.e.,

$$\mathbf{F}\,\mathbf{v}_{\theta_i} \;=\; \lambda_{\theta_i}\,\mathbf{v}_{\theta_i}.$$

By multiplying on the left by $\mathbf{v}_{\theta_i}^\top$ and using the fact that $\mathbf{v}_{\theta_i}$ is normalized, we get

$$\mathbf{v}_{\theta_i}^\top \mathbf{F}\,\mathbf{v}_{\theta_i} = \mathbb{E}_{x\sim\rho}\left[\mathbf{v}_{\theta_i}^\top \nabla_\theta f_k(x)\,\nabla_\theta f_k(x)^\top\,\mathbf{v}_{\theta_i}\right] = \mathbb{E}_{x\sim\rho}\left[\langle \mathbf{v}_{\theta_i}, \nabla_\theta f_k(x)\rangle^2\right].$$

Define the eigenfunction

$$u_{\theta_i}(x) \;=\; \frac{1}{\sqrt{\lambda_{\theta_i}}}\Big\langle \mathbf{v}_{\theta_i}, \nabla_\theta f_k(x)\Big\rangle.$$

Then its $L^2(\rho)$ norm is

$$\mathbb{E}_{x\sim\rho}\Big[u_{\theta_i}(x)^2\Big] \;=\; \mathbb{E}_{x\sim\rho}\Big[\frac{1}{\lambda_{\theta_i}}\,\langle \mathbf{v}_{\theta_i}, \nabla_\theta f_k(x)\rangle^2\Big] \;=\; \frac{1}{\lambda_{\theta_i}}\,\mathbb{E}_{x\sim\rho}\Big[\langle \mathbf{v}_{\theta_i}, \nabla_\theta f_k(x)\rangle^2\Big] \;=\; \frac{1}{\lambda_{\theta_i}}\,\lambda_{\theta_i} \;=\; 1.$$

Hence $u_{\theta_i}$ is normalized in $L^2(\rho)$, completing the proof.

**DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS
UNIVERSITY OF TECHNOLOGY