SAPIENZA
UNIVERSITÀ DI ROMA

Beyond Gradient Descent for Deep Learning: Assessing the Volume Bias Hypothesis through Particle Swarm Optimization

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea in Applied Computer Science and Artificial Intelligence

Candidate

Matteo De Sanctis

ID number 1937858

Thesis Advisor

Prof. Iacopo Masi

Academic Year 2022/2023

Thesis defended on 14 December 2023
in front of a Board of Examiners composed by:

Prof. De Marsico (chairman)
Prof. Bottoni
Prof. Cinelli
Prof. Mancini
Prof. Masi
Prof. Gorla
Prof. Spognardi

Beyond Gradient Descent for Deep Learning: Assessing the Volume Bias Hypothesis through Particle Swarm Optimization
Bachelor's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: desanctis.1937858@studenti.uniroma.it

In deepest appreciation for the unwavering support from my family and the cherished moments shared with my friends.

Abstract

Artificial Neural Networks (ANNs) demonstrate remarkable expressive capabilities, effectively approximating diverse functions with an abundance of parameters and showcasing impressive ability to generalize to unseen data. Despite their pervasive success in practical applications, the underlying mechanisms facilitating this phenomenon remain elusive. The ability of ANNs to generalize, especially in scenarios of substantial overparameterization, where the number of parameters surpasses training examples, challenges conventional understanding.

Existing paradigms attribute small generalization errors to inherent properties of model families or the application of regularization techniques during training. However, these traditional approaches fail to explain why large neural networks generalize well in practice. It is believed that implicit bias (implicit regularization) of optimizers is a key factor for their generalization capability in the overparameterized regime, exhibiting a dependency, not strongly on the choice of optimizer, but rather on the geometry of the loss landscape as outlined by Chiang et al. [7].

Standard training procedures, driven by a simplicity bias, tend to discover simple models, exhibiting small differences between training and test performance. Chiang et al. propose that this bias may be linked to the substantial volume disparity between sets of generalizing and poorly generalizing minima. The "good" minima, found in expansive and flat basins reminiscent of wide-margin minimizers, display invariance to parameter perturbations.

This thesis introduces Particle Swarm Optimization (PSO) as a tool to compare the implicit regularization of Stochastic Gradient Descent (SGD) with that of gradient-free optimizers. Leveraging PSO's efficient and extensive exploration of parameter space, we validate the volume hypothesis. Through the application of Power Law, we estimate convergence rates and generalization accuracy concerning the number of particles used in the optimizer. This approach yields promising insights into the exploration and discovery of generalizing minima, contributing to a deeper understanding of the interplay between optimization algorithms and the generalization capabilities of neural networks.

Contents

Chapter 1

Introduction

Artificial Neural Networks have emerged as powerful tools with the capacity to tackle an extensive range of applications, showcasing their ability to successfully achieve tasks thought impossible before. Even surpassing what was previously considered unattainable.

Neural Networks have demonstrated their exceptional versatility, excelling not only in fundamental tasks such as classification and regression but also thriving in generative tasks, prediction, Natural Language Processing, image recognition, anomaly detection, Reinforcement Learning and a myriad of other domains, making them the cornerstone technology in the field of Artificial Intelligence and Machine Learning.

1.1 Expressiveness Versus Generalization

The efficacy of these models can be attributed, in part, to their expressiveness, which arises from their substantial parameter count. This extensive parameterization enables them to effectively learn and approximate virtually any function, provided they posses a sufficient number of parameters.

However, it is crucial to note that while Neural Networks have the potential to approximate any function, it does not guarantee that an optimizer will necessarily discover the best parameter set to minimize the loss function. Additionally, the quality of the dataset may not always be expressive enough to enable the network to learn the true underlying distribution.
Therefore it is not straightforward to assert that Neural Networks possess the ability to learn any desired function.

The true strength of Neural Networks lies in their ability to generalize effectively to unseen data, as long as this data originates from the same distribution.
This ability enables them to accurately perform their learned function on test data. It is important to note that the capacity of neural networks to generalize appears to conflict with their inherent expressiveness.

This is so as the inherent expressiveness of Neural Networks can lead to a phenomenon known as overfitting. Models can learn almost perfectly the training data,

while struggling to generalize to new, unseen data.

An over-fitted mathematical model contains more parameters than can be justified with data, in a mathematical sense, these surplus parameters are akin to an overly complex polynomial.

Consider for example a simple distribution with some injected noise, overfitting arises when the model inadvertently interprets some of the residual variation (noise) as underlying structural feature. Furthermore, if the number of parameter exceeds the amount of samples, the model can effectively memorize the training data in its entirety, failing severely when making predictions.

In practical applications, Neural Networks commonly possess a significantly larger number of learnable parameters compared to the training examples provided. In such overparameterized settings, one might expect overfitting to occur. It is surprisingly observed however, that gradient-based deep learning algorithms tend to prefer solutions that exhibit good generalization, despite the abundance of parameters and poorly generalizing minima [fig. 1.1].

Learning theory suggest that in order to mitigate overfitting, one should use a model that is "just expressive enough", not exceeding the necessary complexity. This principle finds its roots in Occam's Razor, a philosophical concept advocating a preference for simple explanations over complex ones.

The simplicity in neural networks can result from having a limited number of parameters, although modern deep learning often involves more parameters. Simplicity can also be achieved by introducing a regularization term during training. This term encourages networks to minimize a certain complexity measure, such as penalizing models with large Euclidean norm of the parameters. Despite this, in practice, neural network exhibit strong generalization even when trained without explicit regularization [55]. Hence the success of deep learning is not solely due to explicit regularization. Instead, it is believed that gradient-based algorithms induce an implicit bias or tendency (implicit regularization) favoring solutions that generalize well.

1.2 Learning through an Optimizer

A critical component in the process of training these intricate models is the optimizer. It is a fundamental algorithm that serves as the guiding force behind enhancing a Neural Network's predictive capability, fine-tuning its performance for a specific task. Essentially, it acts like an intelligent conductor, adjusting the myriad parameters within a Neural Network to align them to the desired outcome.
An optimizer is an algorithm used to minimize the loss function (maximization of an objective), which quantifies how well the model's parameters configuration performs in terms of predictive ability or other relevant criteria. Optimizers iteratively update the model parameters to converge toward optimal values that minimizes the loss.

The world of optimizers spans a wide spectrum of algorithms, ranging from zeroth-order to second-order and beyond, each with its unique approach to the optimization process.
These algorithms stretch from those that meticulously explore every nook and cranny

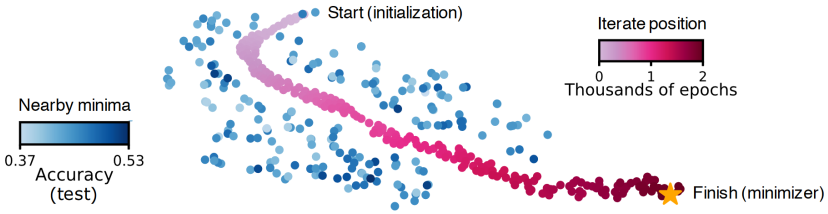


Figure 1.1. Dancing through a minefield of bad minima. Plot of the iterates of SGD after each tenth epoch (red dots). Plot of locations of nearby "bad" minima with poor generalization (blue dots). Visualization with t-SNE embedding. Image from Huang et al. 2020, [17]

of the mathematical terrain to those that navigate swiftly and efficiently toward promising solutions. The choice of optimizer for a given task can significantly impact the training process, convergence speed, stability and final model performance.

Optimizers are constrained to assess performance solely based on the training data's loss, which introduces the potential for discovering effective solutions specifically tailored to the training dataset while potentially yielding suboptimal results on unseen test data.
Despite the presence of such parameters configurations that may cause the model to poorly classify test data while achieving high accuracy on training, commonly used optimizers reliably avoid such "bad" minima of the loss function. Instead, they tend to converge to "good" minima that generalize effectively (fig. 1.1).
This inherent bias toward discovering good minima is an intriguing characteristic of some Neural Network optimizers. This property is often associated with implicit regularization of SGD, which is believed to be crucial for Neural Networks to generalize well, especially in the overparameterized regime.

1.3 Contributions

Throughout literature implicit bias has been attributed to gradient dynamics. This thesis contributes to the understanding of implicit bias (implicit regularization) by introducing Particle Swarm Optimization (PSO) as a tool for assessing the volume hypothesis. The exploration of the disparity in volume between sets of "good" and "bad" minima provides valuable insights into the implicit bias of gradient-based algorithms. This study follows the line of work advocating for implicit regularization of gradient dynamics being only secondary to the observed generalization performance of neural networks, focusing on the low-data regime. Gradient free optimizers showcase effective generalization comparable to SGD, results obtained might indicate that generalization can be explained by the geometry of the loss landscape.

Empirical evaluation of PSO on decision boundaries is employed for specific architectures and diverse datasets, including MNIST and CIFAR-10, enhancing our understanding of models performance in various scenarios. The comparison with SGD and Guess & Check optimizer, as well as the consideration of different architectures, contribute to a comprehensive analysis of implicit bias.

The application of Power Law to estimate convergence rates and generalization

of PSO adds a novel dimension to the study. The examination of improving generalization accuracy with an increase in particles provides valuable quantitative insights into the algorithm's behaviour.

The code implementation of PSO to allow GPU cuda-acceleration provides a valuable resource for future research.

This study acknowledges the role of swarm topology in the optimization process, this consideration contributes to refining our understanding of the factors influencing the behaviour of PSO. The thesis emphasizes the importance of comparing PSO with SGD, and discusses advantages and drawbacks of using PSO as an industrial optimizer, suggesting various techniques to employ PSO in realistic training scenarios. This practical consideration contributes to bridging the gap between theoretical insights and real-world applications.

Overall, these contributions collectively advance our knowledge of implicit bias, optimization algorithms, and the potential applications of PSO in the field of deep learning.

Chapter 2

Implicit Bias of Gradient-Based Algorithms:

As previously discussed, deep-learning algorithms exhibit remarkable performance, yet there is limited understanding of their effective generalization when training overparameterized models. The hypothesis is that the implicit bias within gradient-based deep learning models plays a crucial role in achieving good generalization. Recognition of implicit bias in classification and regression tasks has spurred substantial research into the dynamics of gradient-based algorithms, suggesting that investigating implicit bias may have implications extending beyond generalization.

Gaining theoretical insights is crucial for understanding the problem, advancing research comprehension, and analyzing empirical evidence, as emphasized in this study. This chapter is dedicated to conducting a brief examination of the theoretical foundations. The discussion will incorporate valuable insights from Vardi [46], adopting the analytical framework outlined in the survey, to introduce implicit bias and how it varies across different architectures.

Double-Descent Phenomenon

Belkin et al. [5], show how the implicit bias leads to the double-descent phenomenon. In a model, increasing parameters amount initially reduces training risk, enhancing model accuracy. Test risk decreases as well, reaching an optimal 'sweet spot' between underfitting and overfitting. However, surpassing a certain parameter threshold results in a U-shaped curve, where test risk rises (see also [34]).

Despite an initial rise in test risk, a second descent may occur, enabling *effective generalization with overparameterized neural networks*, a key feature in modern deep learning. This phenomenon is considered result of implicit bias in DL algorithms: in the context of overparameterized models, gradient methods converge to networks that generalize well by implicitly minimizing a certain measure of the network's complexity. However the dynamics of this process remain unclear, and characterizing this complexity presents a significant challenge.

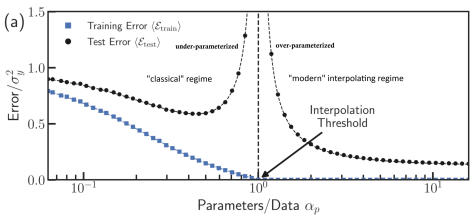


Figure 2.1. Double descent phenomenon. "Wikipedia"

2.1 Implicit Bias Characterization

Gradient based algorithms used to train NNs are gradient-descent (GD) and its variants, such as stochastic gradient descent (SGD). GD starts from a random initialization θ_0 of the network's parameters, updating in each iteration $t \geq 1$ the parameters $\theta_t = \theta_{t-1} - \eta \nabla \mathcal{L}(\theta_{t-1})$, where $\eta > 0$ is the step size and $\mathcal{L}$ is some *empirical loss* to minimize.

2.1.1 Gradient Flow

The analysis of implicit bias often employs the continuous version of gradient descent known as "gradient flow." The training of parameters θ is:

$$\frac{\delta \theta(t)}{\delta t} = -\nabla \mathcal{L}(\theta(t)) \tag{2.1}$$

for all time $t \geq 0$, having some random initialization $\theta(0)$.

For classification tasks, the focus is on *logistic loss function* (binary cross entropy): training on a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \{-1,\ 1\}$ some neural network $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}$ parameterized by θ, the empirical loss is defined as:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=n}^{n} \log(1 + e^{-y_i \cdot \mathcal{N}_\theta(\mathbf{x}_i)})$$

Gradient flow, akin to gradient descent with an infinitesimally small step size, is commonly used to demonstrate implicit bias due to its analytical convenience.

In regression tasks, gradient flow is applied to the *square-loss* function. Thus given a training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \mathbb{R}$ and a neural network $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}$:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (\mathcal{N}_\theta(\mathbf{x}_i) - y_i)^2$$

In an overparameterized setting, multiple choices of θ can yield $\mathcal{L}(\theta) = 0$, indicating many global minima. Implicit regularization function $\mathcal{R}(\theta)$ is introduced to guide gradient flow toward global minima, formalizing the consideration of implicit bias: $\theta^* \in \arg\min_\theta \mathcal{R}(\theta)\ s.t.\ \mathcal{L}(\theta^*) = 0$.

This framework provides a formal basis for understanding implicit bias in gradient methods.

2.1.2 Logistic and Linear Regression

Logistic Regression: Logistic Regression, corresponds to a neural network with a single neuron that does not have an activation function. The linear predictor is $x \mapsto \mathrm{w}^\top x$, where $\mathrm{w} \in \mathbb{R}^d$ are learned parameters. The assumption is that data is *linearly separable*: there exists some $\hat{w} \in \mathbb{R}^d$ such that $\forall i \in \{1, \ldots, n\} \Rightarrow y_i \cdot \hat{\mathbf{w}}^\top x_i > 0$. Note that $\mathcal{L}(\mathrm{w})$ is strictly positive for all w and it may tend to zero as $\|\mathrm{w}\|_2$ tends to infinity. However, there might be infinitely many directions $\hat{\mathrm{w}}$ that satisfy this condition.

Soudry et al. [43] showed that gradient flow converges in direction towards the ℓ_2 *maximum-margin predictor*: this predictor $\tilde{\mathrm{w}} = \lim_{t\to\infty} \frac{\mathrm{w}(t)}{\|\mathrm{w}(t)\|_2}$ points in the direction of the maximum-margin predictor defined as:

$$\arg\min_{\mathrm{w}\in\mathbb{R}^d} \|\mathrm{w}\|_2 \quad s.t. \quad \forall i \in \{1,\dots,n\} \quad y_i \cdot \mathrm{w}^\top \mathrm{x}_i \geq 1 \ .$$

This characterization of the implicit bias in logistic regression offers insight into generalization. In scenarios where $d > n$, infinitely numerous unit vectors (w) with $\|\mathrm{w}\|_2 = 1$ correctly classify the training data. This high amount of directions correctly labeling the training data *includes both directions that lead to effective generalization and those that do not.*
The result ensures that gradient flow converges to the maximum-margin direction, aligning with standard margin-based generalization bounds.

Linear Regression Liner Regression models a linear predictor similarly, where $\mathrm{w} \in \mathbb{R}^d$ are learned parameters. Studies by [55] and [12] show that gradient flow converges to the global minimum of $\mathcal{L}(\mathrm{w})$, with initialization at the closest ℓ_2 distance, so implicit regularization function $\mathcal{R}_{\mathrm{w}(0)}(\mathrm{w}) = \|\mathrm{w} - \mathrm{w}(0)\|_2$ emerges. Minimizing this ℓ_2 norm provides explanation for generalization using standard norm-based generalization bounds. These insights extend beyond gradient flow, encompassing methods like GD, SGD and their variants.

2.1.3 Implicit Bias in Different Models

Analysis of the implicit bias for different models has been done, especially for linear networks, where such topic has been extensively studied as a first step towards understanding implicit bias in deep nonlinear networks.

Linear Networks

Classification A linear *fully-connected* network $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}$ of depth k computes a function $\mathcal{N}(\mathrm{x}) = W_k \dots W_1 \mathrm{x}$, where $W_1, \dots, W_k$ represent weight matrices. The trainable parameters are a collection $\theta = [W_1, \dots, W_k]$ of the weights matrices. The study [18] suggests that if gradient flow converges towards zero loss ($\lim_{\alpha\to\infty} \mathcal{L}(\mathrm{w}) = 0$), the weights $W_1, \dots, W_k$ converge directionally toward the ℓ_2 maximum-margin predictor. This convergence aligns the gradient flow biases of deep linear networks and linear predictors, despite differing dynamics.

Additionally, [19] observes that in deep linear networks operating on linearly separable data, both gradient flow and GD reduce risk to zero. Normalized weight matrices asymptotically align with their rank-1 approximation, implying implicit regularization favoring rank minimization. The linear function induced by the network converges towards the maximum-margin solution, consistent with prior findings [43], framed under the assumptions of GD.

Regression Linear neural networks $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}$ operates with the identity activation function, computing a linear predictor $x \mapsto \mathrm{w}^\top x$, where w is obtained by the product of the weight matrices.The focus shifts from seeking an implicit regularization function $\mathcal{R}(\theta)$ to determining $\mathcal{R}(\mathrm{w})$.

Studies ([2], [51], [54]) provide precise formulations for $\mathcal{R}(\mathrm{w})$ in linear diagonal

and fully-connected networks. These formulations depend on the initialization scale (norm of w(0)) and the initialization "shape," reflecting the relative magnitudes of weights and layers.
Understanding implicit bias in these networks lays the groundwork for comprehending implicit bias in deep nonlinear networks.

Homogeneous Neural Networks

Classification In the realm of practical neural networks with nonlinear activations and predictors, margin maximization in nonlinear predictors is not straightforward. However, for specific neural networks, it has been established that gradient flow maximizes the margin in parameter space. Unlike linear networks, where margin maximization occurs in predictor space, homogeneous neural networks focus on margin maximization with respect to the network's parameters.

Consider a neural network, $\mathcal{N}_\theta$ is *homogeneous* if there exists $L > 0$ such that for every $\alpha > 0$ and θ, x, it holds that $\mathcal{N}_{\alpha\cdot\theta}(\mathrm{x}) = \alpha^L \mathcal{N}_\theta(\mathrm{x})$. In simpler terms, scaling the parameters by any factor $\alpha > 0$ scales the predictions by α^L. Homogeneity often implies symmetrical properties or certain invariances in neural networks, showcasing consistency under scaling.

Research by [18] and [28] reveals that if gradient flow on homogeneous networks reaches sufficiently small loss, the parameters converge directionally as $t \to \infty$. While margin maximization in predictor space elucidates generalization using well-known bounds, its counterpart in parameter space is less straightforward for nonlinear neural networks. Recent works ([36], [3], [11], [48], [8], [29], [41], [10], [40]) provide margin-based generalization bounds for neural networks, including nonlinear homogeneous networks. Combining these results with insights on implicit bias towards maximization in parameter space may contribute to understanding generalization in neural networks. However, the tightness of known margin-based bounds and their practical sample complexity remain open questions.

It is important to note that implicit bias in non-homogeneous neural networks, like ResNet, is not well understood.

Matrix Factorization

Regression Efforts towards comprehending implicit bias in neural networks were largely focused on matrix factorization, particularly matrix sensing problem.
This problem involves finding a matrix $W = W_2 W_1$ from a set of observations about an unknown matrix $W^* \in \mathbb{R}^{d\times d}$. The observations are of the form $y_i = \langle W^*, X_i\rangle$, where X_i are vectors and the inner product is defined as $\langle W, X\rangle = \mathrm{tr}(W^\top X)$. Gradient flow is employed on the objective $\mathcal{L}(W_1, W_2) = \frac{1}{n}\sum_{i=1}^{n}(\langle W_2 W_1, X_i\rangle - y_i)^2$ to find W_1 and W_2. The rank of W is not constrained by the parameterization, but implicit bias influences which compatible matrix gradient flow converges to.

The matrix factorization problem is analogous to training a two-layer linear network. Additionally, *deep matrix factorization* ($W = W_k W_{k-1} \dots W_1$) mirrors training a deep linear network. This problem serves as a natural test case for investigating implicit bias in neural networks and has been extensively studied ([13], [1], [26], [4], [27], [37]). Findings suggest that the observed implicit bias in matrix factorization (and tensor factorization [39], [38]) can be explained through rank minimization.

Both theoretical and empirical evidence indicate that gradient flow, with an infinitesimally small step size, aligns with simple heuristic rank-minimization algorithms.

While an explicit expression for a function R remains elusive in the context of matrix factorization, the implicit bias can be interpreted as a heuristic for rank minimization. However, the implications of these findings for more practical nonlinear neural networks are still uncertain.

Nonlinear Networks

Regression Considering nonlinear neural networks the situation is even less clear. For a single neuron network $\mathrm{x} \mapsto \sigma(\mathrm{w}^\top \mathrm{w})$ where $\sigma : \mathbb{R} \to \mathbb{R}$ is strictly monotonic, gradient flow converges to the closest global minimum in ℓ_2 distance, similarly to the case of linear regression, where $R_{\mathrm{w}(0)}(\mathrm{w}) = |\mathrm{w} - \mathrm{w}(0)\|_2$. However when ReLU is used, the implicit bias is not expressible by any non-trivial function of w. The approach of specifying the implicit bias of gradient flow via regularization function is not feasible in single-neuron ReLU settings, suggesting that this approach may not be useful for understanding implicit bias in the general case of ReLU networks [47]. On a positive note, for single-neuron ReLU networks, the implicit bias can be approximated, within a factor of 2, by the ℓ_2 norm [47].

The results on implicit bias in matrix factorization might suggest a certain tendency towards rank minimization in deep learning models employing the square loss. In [45], gradient flow does not exhibit a bias towards rank minimization in weight matrices within ReLU networks, at least in the context of two-layer networks and small datasets. Still, there is a certain tendency toward rank minimization in deep networks trained with square loss aided by weight decay or when implementing the logistic loss.

Overall the comprehension of implicit bias in nonlinear networks utilizing regression losses remains notably limited. In contrast to the relatively beneficial characterization of the implicit bias in nonlinear homogeneous models for classification tasks, the understanding of even the most elementary models in this context remains elusive.

2.1.4 Dynamic Balance in Gradient Methods

In Gradient Descent (GD) and Stochastic Gradient Descent (SGD) with a finite step size, the discrete and stochastic nature introduces additional biases not present in continuous gradient flow. Understanding these biases is essential for practical insights, with evidence suggesting that adjusting step size and batch size can impact generalization.

GD and SGD cannot stable converge to minima that are too sharp relative to the step size ([32],[33], [52]), when using appropriate step size, stable converge to certain sharp minima can be ruled out, favoring convergence to flat minima ([53], [22]). Note that a single function might be represented with many different networks, some may exhibit sharpness within parameter space while others manifest flatness [9]. Understanding the implications on the function space of the bias towards flat minima in parameter space remains a challenging question, and several works have investigated such implications ([35], [30], [32], [50]).

Further Analysis on Implicit Bias in Deep Learning

While the primary focus of studying implicit bias is understanding generalization in deep learning, its implications go beyond.

- Adversarial Examples: Gradient methods often lead to non-robust neural networks susceptible to adversarial examples, but the reasons for this vulnerability remain unclear [44]. The tendency of gradient methods to train non-robust networks is linked to the implicit bias [49].
- Hidden Representations and Privacy: Implicit bias may provide insights into the hidden representations learned by neural networks, impacting their vulnerability to privacy attacks. Studies show that training data can be reconstructed from neural networks, posing potential threats to privacy [14].

2.2 Good Minima Versus Bad Minima: Flatness and Sharpness

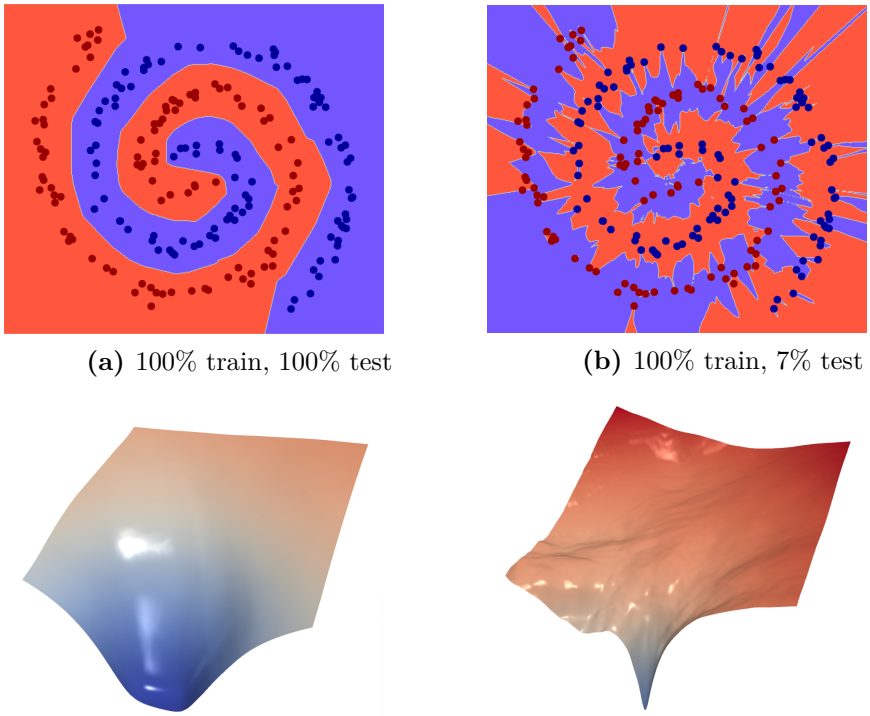
Minima represent points in the vast landscape of optimization where the neural network's loss function reaches its lowest value. Optimization minima in neural networks vary in quality, leading to the distinction between "good" and "bad" minima. The optimization landscape, representing the N-dimensional space of model parameters, resembles rugged terrain. "Good minima" are akin to flat valleys, facilitating effective generalization, while "bad minima" are treacherous valleys, trapping networks in configurations that overfit training data.

The distinction hinges on how model parameters affect classification performance. "Flat" minima, resilient to parameter perturbations, resemble wide-margin conditions in linear models, ensuring stable data classification. In contrast, "sharp" minima indicate close intersections with training data, risking misclassification upon perturbations. A visualization is available in Figure 2.2.

Neural networks substitute the conventional wide margin regularization with an implicit form of regulation that encourages the associated concept of "flatness." Links between flatness and generalization have been explored. Hochreiter and Schmidhuber [16] firstly proposed that flat minima tend to generalize well. Paving the way for research.

As previously discussed, in linear models, wide-margin priors select the linear classifier that maximizes the Euclidean distance from the decision boundary while maintaining accurate data classification. In this scenario, the minima indicate that the training data maintains a comfortable separation from the class boundaries. Consequently, minor shifts in these boundaries do not lead to changes in the classification of nearby data points. Conversely, sharp minima imply that the class boundaries closely intersect with the training data, placing nearby data points at risk of misclassification when subjected to perturbations.

Minima of this nature exist in distinct regions within the landscape, and this concept of spatial distribution plays a crucial role in investigating the phenomenon of implicit regularization. Research has shown that implicit regularization is prominently present in neural network optimizers, indicating that it primarily stems from the geometric properties of the loss function rather than being solely dependent on



(a) 100% train, 100% test **(b)** 100% train, 7% test

(c) Minimizer of network in (a) above **(d)** Minimizer of network in (b) above

Figure 2.2. Top: The decision boundaries of two networks with distinct parameter configurations are compared. Network (a) demonstrates effective generalization, attributed to its flatness and large volume, making it likely to be discovered by SGD. In contrast, network (b) generalizes poorly due to its sharpness and small volume, rendering this minimizer less probable. **Bottom:** A cross-section through the loss landscapes (depicting 2 dimensions out of many) around these minima highlights their sharpness or flatness.

"Huang et al. 2020, Understanding Generalization through Visualizations."

the choice of optimizer ([7], [17].)

It is important to handle sharpness metrics with care, as the straightforward definition of sharpness can be manipulated by adjusting the scale of network parameters.In instances where parameters are small, a slight perturbation might lead to significant performance degradation, unlike larger parameters, which experience minimal impact on performance. In alignment with the methodology presented in [7] and [17], this study employs the filter-normalization scheme introduced by Li et al. [25], wherein network filters are rescaled to have a unit norm.

2.3 Simplicity Bias

Many works have proposed Simplicity Bias (SB)—the tendency of standard training procedures to find simple models—to justify why neural networks generalize effectively. Shah et al. [42] carries out an empirical analysis observing that SB of SGD (and variants) can be extreme: the trained networks can solely rely on the simplest features and remain invariant to all predictive complex features. Such extreme tendency of relying on SB could explain the susceptibility to benign

distribution shifts, small adversarial perturbations and spurious correlations. They also observe that SB can hurt generalization, persisting even when the simplest features have less predictive power than the more complex features and that models lack reliable confidence estimates, having high confidence even if several complex features contradict the simple feature: if all features have full predictive power, NNs rely exclusively on the simplest feature, remaining invariant to all complex features. The problem of SB has not been overcome yet due to its insidious nature, and common approaches to improve generalization and robustness—ensembles and adversarial training—can fail in mitigating SB and its pitfalls.

As previously discussed, linear neural networks trained with SGD on linearly separable data converge to the maximum-margin linear classifier, hence explaining the superior generalization performance. However, maximum-margin classifiers are inherently robust to perturbations and this implication clashes with concrete evidence that neural networks are sensitive to distribution shifts and adversarial examples. Shah et al. [42] show how SGD implicit bias towards simplicity can result in small-margin and feature-impoverished classifiers instead of large-margin and feature-dense ones.

2.3.1 The Volume Hypothesis

Simplicity bias therefore affects significantly neural networks, degrading generalization, because standard training methods are prone to extreme SB. Chiang et al. [7] and Huang et al. [17] show empirical evidence that gradient-based implicit regularization of training dynamics is not required for generalization: gradient-free optimizers, that lack gradient dynamics, still perform well and comparable to SGD. Chiang et al. [7] suggests that the origin of simplicity bias is due to the volume hypothesis (high volume in parameter space), and that the choice of optimizing strategy is only secondary to the large disparity in volumes between non-robust and robust functions in the loss landscape. The strong performance of gradient-free optimizers (in particular G&C, used to directly estimate the volume of the two decision boundaries, measuring a large volume disparity), suggests that the disparate volume of good and bad hypothesis classes biases models toward obtaining simple decision boundaries (SB), being them strongly preferred compared to more complex ones.

Such observations support the idea that implicit bias is due from properties of the loss landscape geometry, rather than the optimizer, where flat basins occupy larger volumes, proposing the idea that *"loss landscape are all we need"*.

The Blessing of Dimensionality

Effective minima with strong generalization properties are situated in broad basins that occupy significant volumes in parameter space, contrasting sharply with minima in narrow basins. Consequently, during a random search algorithm, the likelihood of landing in the attraction basin of a good minimizer is higher than that of a bad one.

The discrepancy in volume between favorable and unfavorable minima is dramatically accentuated by the "curse", or rather, "*blessing of dimensionality*." The probability of encountering a region during a random search does not scale with its width, but rather its volume [17], stating that "high dimensionality crushes bad minima into dust." In the expansive, high-dimensional space where network parameters reside,

even slight differences in sharpness between minima result in exponentially large variations in the volume of their surrounding basins. The work by Huang et al. [17] shows how, as generalization accuracy decreases, the radii of the basins decrease as well, signifying a shift towards sharper minima.
Interestingly, given the neural nets' propensity to discover "flat" minima, the authors also pose a scenario where the notion of flatness leads to poor generalization.

Chapter 3

Particle Swarm Optimization

Particle Swarm Optimization (PSO) stands out as a derivative-free optimizer, steering clear of the inductive bias inherent in gradient methods. This quality renders it particularly beneficial for evaluating volume hypothesis and exploring the independence of implicit regularization from gradient-based algorithms as their main cause of origin, making it a primary source of insight.

3.1 Derivative-free & Zeroth-order optimization

Derivative-free optimization, is a discipline in mathematical optimization that does not use derivative information in the classical sense to find optimal solutions. Derivative-free optimization offers the advantage of preventing activation functions from causing the gradient to vanish during saturation. Moreover, it provides a pathway to investigate the learning process within a model.

As discussed previously, in the literature, a prevailing belief centers around the implicit bias of Stochastic Gradient Descent (SGD) in guiding network parameters toward favorable minima that exhibit strong generalization capabilities (implicit regularization): SGD is perceived to possess robust normalization capabilities, comparable to low-rank matrix multiplication.
Zeroth-order optimizers play a significant role in the assessment and comparison of various optimization techniques, including those relying on gradients, second-moment and higher-order information, and [7] and [17] show how derivative-free optimizers consistently reach performances comparable to SGD.

Zeroth-order subset of derivative-free

In optimization algorithms, the distinction between zeroth-order and derivative-free optimizers is important. Zeroth-order optimizers rely on function evaluations without gradient information, a subset of derivative-free techniques. In our research focused on generalization without the bias of Stochastic Gradient Descent (SGD), the difference between these optimizers is less impactful. We compare them with first-order methods like SGD, emphasizing their impact on generalization rather than intricacies of zeroth-order versus derivative-free techniques. Notably, derivative-free optimizers, while also avoiding gradients, might use different strategies, potentially involving higher-order information or strategies beyond direct function evaluations.

3.1.1 Derivative-free Optimization

The problem to be solved is to numerically optimize an objective function $f : A \to \mathbb{R}$ for some set A (usually $A \subset \mathbb{R}^n$). In essence, without loss of generality, the goal is to identify a point $x_0 \in A$ *s.t.* $f(x_0) \leq f(x)\ \forall\ x \in A$.

Derivative-free optimization relies on methods that tackle these challenges solely using function values of f without employing derivatives. While certain methods can be verified to uncover optimal solutions, some, like Particle Swarm Optimization, are considered more metaheuristic in nature as the problems they address are generally more intricate compared to convex optimization. In these scenarios assurances of achieving optimality are typically unattainable.

Derivative-free Optimization Algorithms

A variety of gradient-free algorithms have been developed in the literature and notable derivative-free optimization algorithms include Bayesian Optimization, Coordinate descent, Differential evolution and Evolution strategies, Genetic algorithms, Multilevel Coordinate Search, Nelder-Mead method, *Particle Swarm Optimization*, Pattern search, Random search, Simulated annealing, Stochastic optimization, Subgradient method, and others.

In the paper [7], zeroth-order optimizers such as Guess & Check, Pattern Search and Greedy Random Search undergo testing across different scales of MNIST & CIFAR-10 and on diverse architectures.
The aim is to investigate the universality of generalization concerning optimizers dynamics. These optimizers provide a valuable research tool by circumventing the use of optimizers with the same inductive bias as gradient methods, thus offering a distinct perspective for research purposes.

Guess & Check The Guess & Check algorithm optimizer functions by randomly producing parameter vectors. These vectors have entries sampled independently and uniformly from the range of $[-1, 1]$. If the model generated through this random sampling achieves 100% training accuracy and its training loss falls below a predetermined threshold, the model is retained, and the optimizer process concludes. However, if the generated vector fails to meet these criteria, it is discarded, and the process iterates by generating new vectors until the stipulated conditions are satisfied.

The theoretical significance of Guess & Check lies in its implicit bias, derived solely from the geometry of the loss landscape. Its success embodies the volume hypothesis. Using this optimizer, the probability of selecting a set of solutions aligns precisely with the volume of the set within the parameter space. If a model consistently exhibits strong generalization when trained with Guess & Check, it signifies that the set comprising "good" minima possesses a substantial volume among parameters with low loss.

Pattern Search Pattern Search operates by randomly selecting a parameter within the model and then taking a step of fixed size, randomly chosen to be positive or negative. If this step results in a reduction in loss, the parameter is accepted as the new starting point. However, if Pattern Search is unable to identify a step that decreases the loss after iterating through all the parameters, the step size is diminished by a constant factor. This process is repeated until a solution is

discovered that achieves 100% training accuracy. In their experiments, the starting value for the radius is 1, reduced by a factor of 2 when the algorithm fails to identify a descent direction.

Random Greedy Search The Random Greedy Search method involves introducing Gaussian noise to the initial parameter vector with standard deviation of σ. If the perturbed solution leads to an improvement in the training loss, then the noised solution is accepted as a new starting point. However, if no solution is found after a fixed number of steps, then the standard deviation σ is decreased by a chosen factor before the search process continues. This process is reiterated until a parameter is discovered that achieves 100% training accuracy. In the experiments of the paper [7] the procedure is started with $\sigma = 1$. In cases where no reduction in loss is observed after 30,000 random steps, σ is then decreased by a factor of 2.

3.2 Introduction to Particle Swarm Optimization

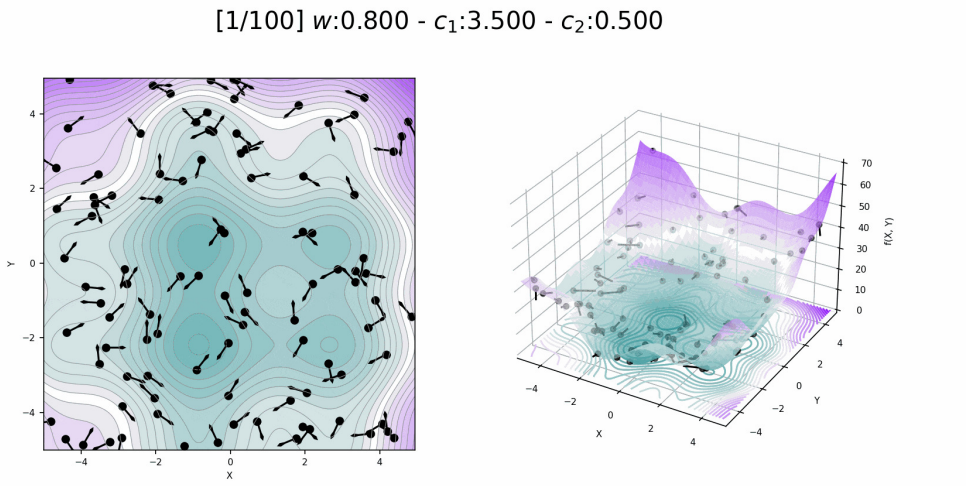


Figure 3.1. PSO Animation: Please use an appropriate PDF reader, such as Adobe, to view the GIF.

To explore the parameter space we therefore choose to shift from plain Guess and Check to a more substantial optimizer, called Particle Swarm Optimization (PSO). This optimizer is a method for the optimization of continuous nonlinear functions using particle swarm approach.

PSO is originally attributed to Kennedy, Eberhart and Shi [21], and was first intended for simulating social behaviour, as a stylized representation of the movement of organisms in a bird flock or fish school.

3.2.1 Metaheuristic Optimizer

PSO represents a computational technique employed for optimizing problems by iteratively enhancing a candidate solution's quality.
This process involves managing a population of candidate solutions, referred to as

particles, and manipulating their positions and velocities within the search-space using a mathematical formula over the particles' positions and velocities.

The movement of each particle is influenced by its knowledge of the best local position it has encountered so far. Simultaneously, particles are guided towards the best-known positions within the entire search-space, which are continually updated as other particles discover better solutions. This collective effort is aimed at driving the group of particles towards optimal solutions.

It's important to note that PSO is classified as a metaheuristic because it operates without making specific assumptions about the problem under consideration. This flexibility enables it to explore vast solution spaces effectively.
PSO can handle non-differentiable optimization problems, however, it's worth mentioning that while PSO is powerful, like other metaheuristics, it does not guarantee the discovery of an optimal solution.

PSO for High-Dimensional Search Spaces

PSO's strength lies in its ability to efficiently explore complex solution spaces, making it well-suited for optimization tasks where traditional mathematical approaches are less effective. It has been successfully applied to a wide range of problems, including function optimization, parameter tuning in machine learning algorithms, and the optimization of engineering and logistical systems.

Particle Swarm Optimization is hence well-suited for the optimization of problems characterized by high-dimensional search spaces: the collaborative effort among particles enables them to effectively explore and navigate complex search spaces with a substantial number of dimensions.

3.2.2 Particle Swarm Optimization Topologies

The swarm's topology determines which particles can exchange information with each other.
The standard version of the algorithm employs a global topology where all particles communicate with each other. This setup enables the entire swarm to share the best position determined by a single particle. However, this approach can potentially lead the swarm into local minima, limiting exploration.

To address this issue, various topologies have been developed to regulate the flow of information among particles. One example is the local topology, where particles only communicate with a subset of their peers. This subset can be defined geometrically, such as "the m nearest particles," or it can be based on social connections, irrespective of distance. In such cases, the PSO variant is described as 'local best,' in contrast to the 'global best' approach of basic PSO.

3.3 The Algorithm

Consider a function $\mathbb{R}^n \to \mathbb{R}$, it serves as a cost function that must be minimized. It takes a candidate solution in the form of a vector of real numbers and produces a real number, representing the objective function value of that candidate solution.
It's important to note that the gradient of this function, ∇f, is unknown.

The primary objective is to identify a solution, denoted as "a," for which $f(a) \leq f(b)$ holds for all "b" within the search-space. This condition signifies that "a" is the global minimum, as it outperforms all other solutions within the search-space with respect to the cost function f.

Consider a swarm denoted as "S," consisting of a total of particles. Each particle in this swarm possesses a position vector $\mathbf{x_i}$ in the search-space, where $\mathbf{x_i}$ is an element of $\mathbb{R}^n$, and a velocity vector $\mathbf{v_i}$ in the same space. Additionally, particle i maintains a record of its best-known position, denoted as $\mathbf{p_i}$, and the entire swarm shares a common best-known position represented by the vector $\mathbf{g}$.

A fundamental Particle Swarm Optimization (PSO) algorithm, with the aim of minimizing a given cost function, incorporates the following update procedure for particle velocities:

$$\mathbf{v}_{\mathbf{i,d}}^{t} \leftarrow w \cdot \mathbf{v}_{\mathbf{i,d}}^{t-1} + \phi_p \cdot r_p \cdot (\mathbf{p}_{\mathbf{i,d}}^{t-1} - \mathbf{x}_{\mathbf{i,d}}^{t-1}) + \phi_g \cdot r_g \cdot (\mathbf{g_d} - \mathbf{x}_{\mathbf{i,d}}^{t-1}) \tag{3.1}$$

where d is the dimension.
Following the velocity update, the subsequent step in the particle swarm optimization (PSO) algorithm involves updating the particle positions as follows:

$$\mathbf{x}_{\mathbf{i,d}}^{t} \leftarrow \mathbf{x}_{\mathbf{i,d}}^{t-1} + \mathbf{v}_{\mathbf{i,d}}^{t} \tag{3.2}$$

where x_i and v_i represent the particle positions and velocities.

Momentum can be introduced to have a more stable and smooth convergence, helping particles to keep moving in their current direction. Retaining previous velocities can help overcome local optima and explore the search space more effectively. Momentum vector is an element of $\mathbb{R}^n$ as well. Including momentum in PSO involves two steps, first updating momentum based on the previous velocities:

$$\boldsymbol{\mu}_{\mathbf{i,d}}^{t} \leftarrow \boldsymbol{\mu}_{\mathbf{i,d}}^{t-1} + \mathbf{v}_{\mathbf{i,d}}^{t-1}$$

where t is the time step or iteration of PSO. Second step is to update the velocities with the momentum, so, after (3.1), velocities are update as follows:

$$\mathbf{v}_{\mathbf{i,d}}^{t} \leftarrow \mathbf{v}_{\mathbf{i,d}}^{t} + \boldsymbol{\mu}_{\mathbf{i,d}}^{t}$$

after this, positions can be updated with (3.2).

These variables above are initialized with random vectors, each having dimensions of $N \times D$, where "N" stands for the number of particles and "D" signifies the number of parameters.
The initialization involves using uniformly distributed random vectors for both positions and velocities:

$$\mathbf{x_i} \sim \mathbb{U}(b_{lo}, b_{up})$$
$$\mathbf{v_i} \sim \mathbb{U}(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$$

The values b_{lo} and b_{up} define the lower and upper boundaries of the search-space, respectively. The parameter w corresponds to the inertia weight.
The parameters φ_p and φ_g are respectively the cognitive coefficient and social

coefficient, they are responsible for regulating the strategy employed by the algorithm, balancing exploration and exploitation. They dictate how much emphasis the particles place on their individual knowledge (cognitive) and the collective knowledge of the swarm (social) when determining their next moves. The values chosen for these coefficients significantly impact the algorithm's exploration of the search space and exploitation of promising regions.
The variables r_p, r_g are drawn from a uniform distribution over the interval $[0, 1]$:

$$r_p, r_g \sim \mathbb{U}(0, 1)$$

The particle's best known position and swarm's best position are updated in accordance to the positions reached during the iteration.

The termination criterion for the PSO algorithm can be based on either a specific number of iterations performed or upon reaching a solution where the adequate objective function value is found.

Furthermore, it's important to note that the parameters w, φ_p, and φ_g are chosen by the practitioner, and they play a pivotal role in shaping the behavior and effectiveness of the PSO method. The selection of these parameters influences how the PSO algorithm operates and its overall performance.

3.3.1 Parameter Selection

Among the hyperparameters in PSO algorithm, the most significant ones which heavily influence the behaviour of the swarm are:

- Inertia Weight (w): This hyperparameter determines the trade-off between exploration and exploitation. A higher value promotes exploration, while a lower value encourages exploitation.
- Cognitive Coefficient (φ_p): It controls the influence of a particle's best on its movement. Higher values make particles focus more on their own best-known positions.
- Social Coefficient (φ_g): This coefficient regulates the impact of the global best position on particle movement. Higher values lead to particles being more influenced by the collective best-known position of the swarm.
- Number of particles (S): The size of the swarm can significantly affect the algorithm's performance, as we will later see in this work. A larger swarm may enhance exploration but can be computationally intensive.
- Number of dimensions (D): The number of dimensions in the search space. It influences the complexity of the optimization problem. In the context of optimizing the parameters of an artificial neural network, the dimensionality corresponds to the total number of parameters in the neural network. Each parameter, such as weights and biases, represents a dimension in this space. As the complexity of the neural network increases by adding more layers or units, the number of parameters and, consequently, the dimensionality of the parameter space also grows. This relationship is vital to consider when applying optimization techniques like Particle Swarm Optimization (PSO) to

find optimal parameter values. The choice of hyperparameters, such as the number of particles in PSO or the search space bounds, should be aligned with the dimensionality of this parameter space for effective optimization.

- Search Space Bounds (b_{lo} and b_{up}): These define the lower and upper boundaries of the search space, effectively limiting where particles can explore.

The selection and tuning of these hyperparameters are crucial in achieving the desired performance and convergence behaviour of the PSO algorithm for a given optimization problem. The choice of values for these hyperparameters can significantly impact the algorithm's effectiveness.

In this study, the fine-tuning of these hyperparameters is postponed due to its computational intensity. The primary focus is to investigate the convergence behavior of PSO in the "average case" without hyperparameter optimization.
Additionally, the study intends to evaluate the performance of the optimized model on unseen data, with a particular emphasis on its ability to generalize. This emphasis is placed on the volume of favorable minima rather than relying solely on the inherent optimization ability of the optimizer.
This parameterization would then be compared against the average case to assess its ability to more effectively explore the search space and hopefully locate wide flat good-minima (high volume) rather than sharp bad-minima (low volume).

3.4 Code Implementation

Due to the heavy computational cost of attacking the experiments of [7], work has been done to speed up computation. PSO was implemented from scratch in order to enable tensor based GPU cuda-acceleration.

The algorithm slightly changes in the implementation as position and velocity updates are carried out simultaneously thanks to matrix operations, to make computations faster. Essentially the main components needed are matrix representation for particles and code implementation for the optimizer and the relative fitness function that evaluates the cost of each particle (parameters configuration).

```
class Particles_gpu:
    def __init__(self, model, num_particles, bounds, vel_range,
    momentum):

        self.positions = self.initialize_positions(...)
        self.velocities = self.initialize_velocities(...)
        if self.momentum:
            self.momentum = torch.zeros_like(sel.velocities)
        self.cognitive_best =  torch.zeros_like(self.positions)
        self.cognitive_best_cost = torch.full((num_particles,),
        np.inf)
        self.social_best = None
        self.social_best_cost = np.inf
        self.best_acc = 0

    def initialize_positions(self, model, num_particles, bounds):
        ''' Initialization of the particles positions starting from
            the model's flattened parameters. '''
            [...]

    def initialize_velocities(self, vel_range):
        ''' Initialization of particles velocities, same shape as
            their positions. '''
            [...]

    def update_velocities(self, inertia, c1, c2, momentum):
        ''' Function called by the optimizer to update the particles
            velocities. '''
        # Momentum term based on previous velocities
        if momentum:
           self.momentum = momentum * self.momentum + self.velocities
        # Velocities update with PSO rule
        self.velocities = inertia * self.velocities + c1 * rand1 *
            (self.cognitive_best - self.positions) + c2 * rand2 *
            (self.social_best - self.positions)
        # Add momentum
        if self.momentum: self.velocities += self.momentum

    def update_positions(self):
        ''' Function called by the optimizer to update the particles
            positions, clamping applied to not exceed bounds. '''
        self.positions = torch.clamp(self.positions+self.velocities,
            self.lower_bound, self.upper_bound)

    def update_cognitive_and_social_best(self, fitness_values):
        ''' Method used to update the social and cognitive best
            values and relative positions of the swarm'''
            [...]

    def set_parameters_from_flat(model, position):
        ''' Set the model's parameters using the flattened parameter
            tensor. Method needed to unflatten the vector lain in the
            parameter space '''
            [...]
```

Figure 3.2. Implementation of Particles class.

```
class GlobalParticleSwarmOptimizer_gpu:
    def __init__(self, model, num_particles, bounds, inertia,
    momentum, c1, c2, momentum):

        self.particles = Particles_gpu(...)
        self.inertia = inertia
        self.momentum = momentum
        self.c1 = c1
        self.c2 = c2
        self.num_particles = num_particles

    @staticmethod
    def fitness_function(model, positions, inputs, targets, val_data,
    val_labels):
        ''' Method to evaluate fitness of particles positions.'''
        losses = []; accs = []
        for position in positions:
            Particles_gpu.set_parameters_from_flat(model, position)
            output = model(inputs)
            loss, acc = calculate_loss_acc(output, labels)
            accs.append(acc)
            losses.append(loss)

        return accs, losses

    def optimize(self, model, inputs, labels, max_iters, fitness_f):
        '''Optimization method to find best position.'''
        values = fitness_f(...)
        self.particles.update_cognitive_and_social_best(values)
        # stopping criterion implementation
        [...]
        # Update velocities and positions
        self.particles.update_velocities(...)
        self.particles.update_positions()
    return best_cost, best_position # self.particles.
```

Figure 3.3. Implementation of Global Particle Swarm Optimizer.

Chapter 4

Empirical Insights into Model Generalization

To asses the volume hypothesis, the approach involves employing PSO, a derivative-free optimizer, on toy samples introduced in [7]. In this research, Particle Swarm Optimization is employed to leverage derivative-free dynamics and explore how generalization is constrained within the loss landscape of the hypothesis class. The following step is to extend this analysis to real-world datasets such as MNIST and CIFAR-10. Throughout this study, the results obtained will be juxtaposed with those presented in the referenced paper.

4.1 Overparameterization

As previously discussed, learning theory posits that an overparameterized model is susceptible to overfitting, yet double-descent phenomenon challenge this notion. Achieving meaningful generalization without implicit bias becomes nearly impossible in realistic use cases.
While the complexity of a hypothesis class escalates with the number of parameters, SGD often consistently discovers effective classifiers despite this heightened model complexity. Analysis of implicit regularization on gradient free optimizers is initially performed on selected toy classification problems such as the Wedge distribution.

4.1.1 Wedge Distribution

The Wedge Distribution "Λ" comprises two classes with a vertical margin between them.
In the context of overparameterized model, perfect fitting of the training data may occur, but this often results in poor classification of unseen data. In [7], this phenomenon is demonstrated through model poisoning, wherein the objective is to minimize loss on the training data while maximizing loss on the testing data, showing the capability of models to greatly overfit.

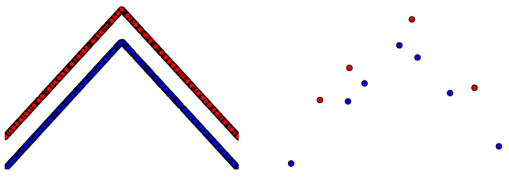
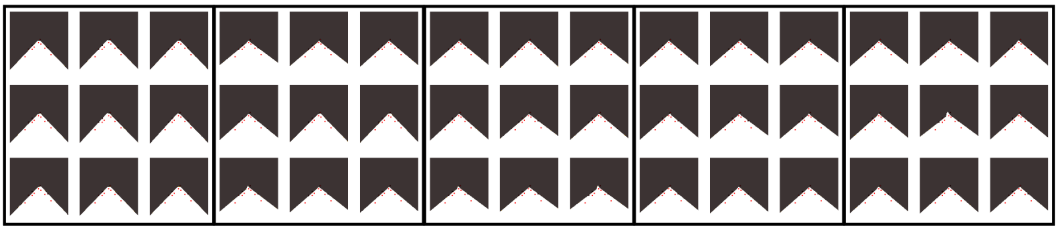
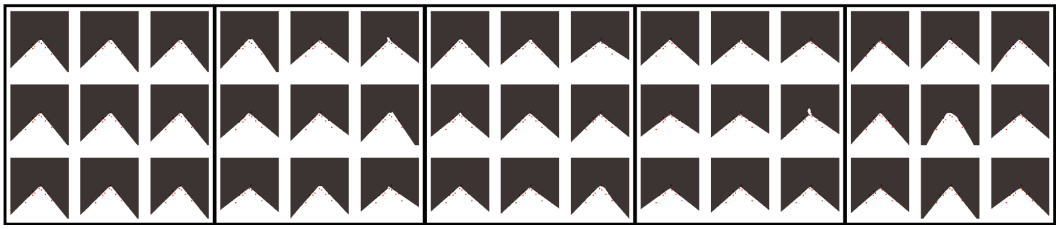


Figure 4.1. Wedge Distribution.
From "Chiang et al., 2023 [7]"

(a) Decision boundaries of poisoned models



(b) Decision boundaries of SGD trained models



(c) Decision boundaries of G&C trained models

Figure 4.2. Stable decision boundaries for both SGD and Guess & Check as increasing the number of parameters.
For each image, from left to right, decision boundaries are produced by 2, 4, 10, 15, 20 hidden units single hidden layer neural networks with the respective optimizer. For each size it is shown 9 random decision boundaries. Chiang et al., 2023, [7].

The analysis focuses on models achieving 100% accuracy. In line with learning theory, the poisoned models, representing a worst-case scenario for an optimizer, strike a delicate balance between fitting training examples and performing poorly on the poison objective when under-parameterized. With fewer parameters, the model, fitting the training data, tends to perform well on test data. However, increasing the number of parameters allows the model to overfit the data and exhibit poor generalization.

In their 2023 study, Chiang et al. [7] illustrate that even whit the availability of more intricate decision boundaries through an increase in the number of hidden units, such boundaries are never discovered when optimizing with SGD fig 4.2b.

PSO - Uniform Init. - 1 hidden layer

Within the model class, both regular and inconsistent decision boundaries coexist. Throughout literature, the implicit regularization responsible for this phenomenon has been attributed to gradient dynamics. Interestingly, similar levels of generalization are achieved with the Guess & Check

zeroth-order optimizer, which operates independently of gradient dynamics fig 4.2c.

PSO explores the parameter space much more efficiently than the Guess & Check algorithm, so applying PSO on this problem instance helps understanding how gradient free optimizers behave, free of SGD inductive bias. We can see that uniform initialization and xavier initialization for PSO produce slightly more inconsistent decision boundaries with respect to SGD and G&C. Note that the same architecture as [7] was employed (varying hidden units on one single hidden layer model, with bias only in the output layer).

PSO - Xavier init. - 1 hidden layer

It is important to highlight that PSO shares a similar initialization with G&C, uniformly initializing each particle. This mechanism is present in both optimizers. However, PSO distinguishes itself by utilizing social and cognitive informations to iteratively update the swarm based to the best positions found so far, employing a metaheuristic to explore the parameter space.

Contrary to expectations, the two images reveal that Particle Swarm Optimization exhibits more inconsistency compared to the Guess & Check algorithm, despite their shared initialization behaviour in the parameter space.
A potential explanation for this phenomenon is that PSO might be prone to selecting "bad minima" influenced by local minima during the training process. Unlike G&C, which randomly samples the parameter space and retains only 100% accuracy solutions, selecting minima proportionally to their volume, PSO, as it trains, converge toward local optima. In the vicinity of these local optima, there might be a higher volume of suboptimal sharp minima, leading to the discovery of worse generalizing solutions.

The global behavior of particle swarm could hence adversely impact exploration, resulting in "bad" minima being found caused by the influence of local optima. To mitigate these risks, alternative topologies may be considered to enhance exploration efficiency.

PSO-1024 samples-Uniform init.

PSO-1024 samples-Xavier init.

Figure 4.3. PSO Decision Boundaries: 2, 4, 10, 15, 20 hidden units, 3 hidden layers.

Given that the experiments operate in the low data regime, a potential strategy to better learn the underlying distribution is to augment the training data while maintaining an overparameterized setting. By employing a 1024-sample dataset and training on 70%, and expanding the models to include 3 hidden layers, the number of parameters effectively spans the range from 204 to 1 Million or more.
This configuration appears to yield decision boundaries that are somewhat more consistent, although they still fall short of the perfection achieved using SGD or G&C. Larger models towards the right side of the images having 10, 15 and 20 hidden units are the ones greatly overparameterized and such overparameterization shows slightly more inconsistent decision boundaries.

Figure 4.4. PSO Decision Boundaries for a model (dense layers) with 10 hidden units over 1 hidden layer, trained on 128 samples. Note that this setting is not overparameterized.

4.1.2 Different Architectures

It is worth to highlight that, during experimentation phase, the architecture of the models emerged as a key factor influencing the consistency of decision boundaries.

The models employed in the experiments, as detailed in [7], feature biases exclusively in the output layer, with no biases in the hidden layers. So input goes through weight (matrix) multiplication and gets applied the ReLU activation function at each layer. Only at the output layer this procedure differs by multiplying for the final weight matrix and adding the bias vector.

Dense Layers Significantly more inconsistent decision boundaries emerged when incorporating affine transformations at each layer [fig. 4.5]. Specifically, multiplying by the weight matrix, applying bias and employing the ReLU activation function at every layer.

This results underscore the dependency of implicit bias on the model's architecture. The complex interactions among features in the hidden layers significantly impact decision boundaries. The presence of multiple layers and non-linear activation functions leads to decision boundaries that exhibit high sensitivity to slight variations in the input space.

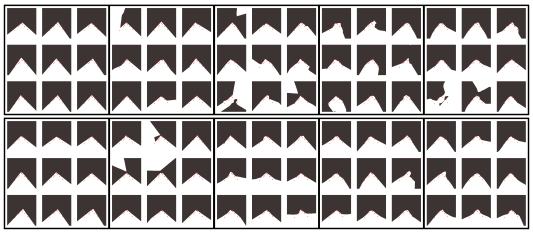


Figure 4.5. PSO Decision Boundaries. Dense Layers. 16 training samples

In contrast, the decision boundaries of the simple architecture are solely determined by the weights, and bias (at the output layer). With these configuration the input to the model undergoes scaling, shearing and rotation, with the translation necessary for an affine transformation occurring exclusively at the output layer. This simplicity contributes to more stable and consistent decision boundaries.

As evident, the simplicity in decision boundaries contributes to better generalization and robustness. This observation align with the idea of Occam's razor, advocating a preference for simpler models.

More results can be observed in [fig. 4.6], where models are trained on much more training samples, effectively going out from the low-data regime the previous experiments were focusing on. As it can be seen, models, that span from 2 hidden units to 50 hidden units, obtain dramatically more inconsistent decision boundaries when more parameterized. This results are consistent with learning theory.

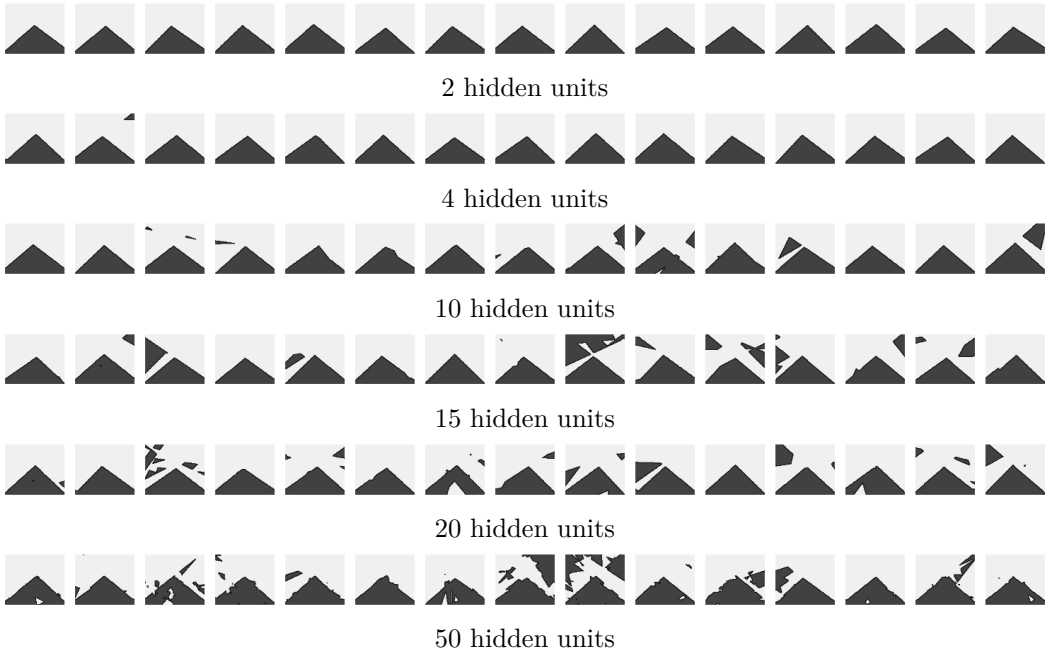


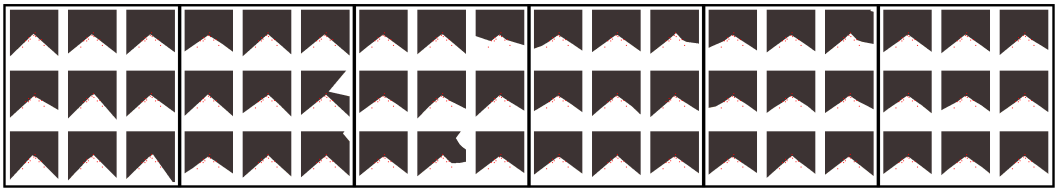
Figure 4.6. PSO Decision Boundaries for model with one single hidden dense layer. Training samples are 256 and the setting is no long overparameterized. Much more irregular decision boundaries arise when training larger models, consistently with learning theory.

Nonetheless having biases in every layer notably degrades the generalization power of these architectures with respect the previous one. This could be due to the fact that the specific architecture used in [7] is fortuitously well performing on the wedge distribution; the ReLU function may play a big role as well.

Lowering the training samples amount, as in [fig. 4.4], where a 10-hidden units model was trained on halved training samples, produces similar results.

How do SGD behave on these architectures? Running again the experiments tested in [7] shows how training with SGD leads to consistent decision boundaries, both in the low-data regime with 16 samples and out of it with 256 data samples. Images are not presented here as they are pretty similar to the results of the paper. It is interesting to study how SGD behaves on dense layers though, comparing it to the results obtained in [fig. 4.5] and in [fig. 4.6]. Decision boundaries of SGD trained models having dense layers, on 16 and 256 samples are illustrated in [fig 4.7], showing the tendency of SGD to achieve consistently regular decision boundaries.

Does this mean that implicit regularization is a feature of gradient-based optimizers? Well, even though PSO achieves irregular decision boundaries, consistently with learning theory, it is needed to study the culprit of such behaviour. SGD implicit regularization properties are known, and PSO may be showing suboptimal behaviour for a variety of reasons, from absence of hyperparameters tuning, to swarm topology that may play an important role.



(a) 16 samples

(b) 256 samples

Figure 4.7. SGD decision boundaries for dense layers models. From left to right, each box represents decision boundaries of 9 models, respectively having 2, 4, 10, 15, 20, 50 hidden units

To evaluate the impact of architecture on the volume disparity between "good" and "bad" minima, G&C was applied to the models with dense layers. However, it was observed that G&C failed to find solutions within a reasonable timeframe. This outcome highlights a notable bias towards solutions within the architectures without bias in hidden layers. The implication is that a more constrained hypothesis class, characterized by architectures without hidden layer bias, demonstrates superior performance in the context of this specific problem.

Bias-only Layers Since much more irregular decision boundaries have been found when introducing biases in hidden layer, experiments were carried out to verify whether, on such small toy-problem, bias-only models were enough to capture the underlying distribution.
Bias-only models operate within a highly restricted hypothesis class, primarily capable of translation. So the input data can only be translated, non-linearity is introduced by ReLU activation function.
Results obtained did not provide any interesting insight worth mentioning and therefore are not presented into this study.

4.2 Experiments on real-world datasets

In transitioning to real-world datasets and exploring the dependency of implicit bias on the geometry of the loss landscape instead of the optimizer gradient dynamics, this section employs Particle Swarm Optimizer on a LeNet model applied to the MNIST [24] and CIFAR-10 [23] datasets. The objective is to compare the results with those obtained in [7], focusing on a 2-class problem with fewer than 32 total training samples. Performance evaluations are conducted across various train loss levels after normalizing the model's weights—scaling the weights to a standard range (normalized by dividing by the ℓ_2 norm). This is important as lower loss levels correspond to better generalization and weights normalization aims to make the optimization landscape more favorable.

Brief description of LeNet models The LeNet model consists in a convolutional neural network (CNN), where at each convolutional layer a Conv2d is applied. The architecture used in [7] and in these experiments consist in 2 convolutional layers, with kernel size equal to 5, intuitively their role is to extract features, and 3 convolutional layers with kernel size equal to 1, needed to classify, which take the role of a fully connected layer.
The two convolutional layers after applying convolution, use ReLU activation function and max-pooling.
After the convolutional layers, the output undergoes flattening and is subsequently processed through three convolutional layers with 1×1 kernels. These layers are strategically designed to emulate the functionality of fully connected layers, despite differing in their underlying mechanisms. Notably, these convolutional layers share identical parameters across all channels, preserving crucial spatial invariance essential for image data. This design choice not only enhances computational efficiency but also accelerates overall processing speed.

Group Convolution In their work, Chiang et al. [7] leverage group convolution [6] as a clever technique to embed an ensemble of models within a single LeNet architecture.
Prior to inputting the data into the network, it is duplicated along the 'model count' dimension, enabling the simultaneous processing of multiple instances.
The group convolution is a CNN technique where input channels are divided into groups, and each group undergoes independent convolution with a specific subset of filters. The outputs of these groups are then concatenated to generate the final layer output.
For a layer with 'C' input channels, 'F' filters, and 'G' groups, each group consists of 'C/G' input channels and 'F/G' filters. This subset of input channels is convolved independently with its set of filters. Notably, filters within a group exclusively access the subset of input channels assigned to that group.
By aligning the number of groups with the required number of models in the ensemble, each group operates independently. The repeated input allows for every group to behave as an independent model processing a distinct subset of input channels, and producing a unique set of output channels.
This innovative approach enables each model to learn diverse features from the input data, effectively establishing an ensemble of models within a single architecture.

The findings from [7] reveal that the Guess & Check optimizer, under equivalent loss levels and sample sizes, performs comparably with SGD, especially at lower loss levels. Notably, in the case of CIFAR-10, Guess & Check even surpasses SGD solutions significantly, presenting an intriguing outcome as the models are capable of complete overfit the train dataset while totally misclassifying the validation data.

To underscore the substantial role of the architecture in implicit regularization, the paper also trains a linear model on MNIST with a significantly smaller hypothesis class. The results show that the linear model underperforms the G&C solution by more than 10% in many cases (despite the convexity of the linear problem, variations occur due to the stochastic nature of SGD).

In the low-data regime, emphasizing the effects of overparameterization, where a small number of samples samples are trained on a LeNet model with 11074 parameters, the model continues to generalize effectively compared to SGD. This suggests that a large volume of the good solution set is adequate to bias the optimizer toward

Sample Count	Arch	Optimizer	Best Test Acc	Train Loss (0.3, 0.35)	(0.35, 0.4)	(0.4, 0.45)	(0.45, 0.5)	(0.5, 0.55)	(0.55, 0.6)	(0.6, 0.65)
32	*LeNet*	*GlobalPSO*	91.89%±0.35%	91.89%±0.35%	91.16%±0.33%	89.76%±0.41%	89.55%±0.38%	87.68%±0.40%	85.75%±0.44%	82.48%±0.49%
	LeNet	*G&C*	93.02%±0.27%	93.02%±0.27%	92.39%±0.29%	90.59%±0.34%	89.18%±0.38%	87.22%±0.43%	86.23%±0.44%	83.15%±0.51%
	LeNet	*SGD*	**94.04%±0.25%**	–	–	94.04%±0.25%	93.49%±0.28%	92.54%±0.28%	91.63%±0.33%	88.60%±0.35%
16	*LeNet*	*GlobalPSO*	89.03%±0.43%	89.03%±0.43%	87.55%±0.48%	86.40%±0.52%	85.59%±0.51%	82.73%±0.57%	80.25%±0.55%	76.75%±0.65%
	LeNet	*G&C*	89.21%±0.47%	89.21%±0.47%	87.01%±0.50%	85.18%±0.56%	84.69%±0.54%	81.91%±0.62%	78.61%±0.65%	75.37%±0.63%
	LeNet	*SGD*	**91.24%±0.40%**	91.24%±0.40%	90.87%±0.41%	90.84%±0.38%	88.77%±0.48%	87.93%±0.48%	86.98%±0.47%	83.90%±0.49%
8	*LeNet*	*GlobalPSO*	**86.91%±0.57%**	85.25%±0.46%	86.91%±0.57%	84.71%±0.55%	76.98%±0.68%	67.45%±0.81%	–	–
	LeNet	*GlobalMomentumPSO*	85.86%±0.62%	85.86%±0.62%	83.89%±0.67%	81.70%±0.65%	76.96%±0.74%	75.08%±0.71%	72.13%±0.76%	68.42%±0.80%
	LeNet	*G&C*	83.05%±0.67%	83.05%±0.67%	80.72%±0.75%	78.23%±0.81%	78.05%±0.72%	76.40%±0.79%	70.76%±0.74%	67.48%±0.78%
	LeNet	*SGD*	84.82%±0.63%	83.63%±0.63%	84.82%±0.63%	82.62%±0.74%	81.85%±0.72%	79.70%±0.70%	79.74%±0.63%	76.51%±0.71%
4	*LeNet*	*GlobalPSO*	78.12%±0.81%	78.12%±0.81%	74.37%±0.88%	71.69%±0.84%	70.32%±0.83%	68.16%±0.85%	65.45%±0.81%	60.46%±0.79%
	LeNet	*GlobalMomentumPSO*	**78.28%±0.74%**	78.28%±0.74%	75.49%±0.87%	71.95%±0.89%	68.79%±0.97%	68.31%±0.84%	64.16%±0.84%	60.72%±0.78%
	LeNet	*G&C*	76.28%±0.90%	76.28%±0.90%	73.93%±0.92%	72.63%±0.86%	70.89%±0.90%	68.27%±0.83%	65.63%±0.92%	62.38%±0.91%
	LeNet	*SGD*	77.35%±0.81%	77.35%±0.81%	75.01%±0.85%	75.61%±0.83%	73.95%±0.85%	73.28%±0.85%	69.15%±0.84%	67.65%±0.84%
2	*LeNet*	*GlobalPSO*	66.09%±0.90%	66.09%±0.90%	64.88%±0.95%	64.01%±0.83%	62.19%±0.76%	60.59%±0.91%	57.86%±0.76%	55.21%±0.82%
	LeNet	*GlobalMomentumPSO*	68.61%±0.99%	68.61%±0.99%	65.29%±0.82%	62.61%±0.82%	61.86%±0.83%	59.74%±0.87%	58.20%±0.87%	55.59%±0.96%
	LeNet	*G&C*	66.89%±1.04%	66.89%±1.04%	65.87%±1.05%	64.03%±0.92%	62.81%±0.90%	61.02%±0.84%	59.90%±0.91%	56.82%±0.95%
	LeNet	*SGD*	**69.67%±0.98%**	69.67%±0.98%	67.11%±0.93%	64.94%±0.95%	63.42%±0.87%	64.38%±0.88%	63.82%±0.89%	62.33%±0.87%

Table 4.1. On the two class MNIST problem, PSO performs comparably to G&C and SGD across different train loss level and number of samples. Data outside of PSO taken from Chiang et al. 2023 [7], showing that despite large number of parameters, G&C solutions are implicitly regularized, the same goes for PSO behaviour. The paper show also a linear model to compare the degree of generalization of G&C not shown here. Estimated standard errors of the averages are computed over 175 random data splits and training seeds. Dashed lines are non converging results. Accuracy on the whole distribution.

generalization (table 4.1, table 4.2).

Particle Swarm Optimization demonstrates comparable performance to SGD, particularly at low loss levels, and in some instances, it outperforms SGD solutions. The efficient exploration of parameter space by PSO makes it a strong ally for validating the results in [7].
However, at certain loss levels, the behavior of PSO is less favorable than the two other optimizers, indicating that its global topology may play a crucial role in model training. Local minima could influence the entire swarm to converge to some suboptimal areas, where minima are sharper. Note that PSO shares initialization with G&C, the dynamics of the particles then build on top of it , so common sense would suggest at least similar behaviour to G&C, since differing in behaviour only by topology dynamics. Hence diminished generalization encountered could mean suboptimal behavior due to the PSO topology.

While generalization performance is comparable between SGD, G&C and PSO, the former slightly outperforms the others, hinting at additional bias that SGD may consider. Nevertheless, the hypothesis being evaluated suggests that *optimizer-specific bias might not necessary to explain generalization and may note be the primary cause of generalization behaviour.* The core of generalization in the examples can be attributed to the the geometry of the loss landscape.

It is important to remark the fact that to enable fair comparisons between PSO, G&C and SGD optimized models, the performance across different train loss levels is compared after the model's weights have been normalized.

Note that standard error is $SE = \frac{s}{\sqrt{n}}$, and it is a measure of the variability or dispersion of sample means in a sampling distribution. It quantifies how much the samples mean is expected to vary from the true population mean. Essentially, it represents the average amount by which sample means deviate from the population mean, and a small SE indicates a more precise estimate of the population mean.

Sample Count	Arch	Optimizer	Best Test Acc	Train Loss (0.55, 0.57)	(0.57, 0.59)	(0.59, 0.61)	(0.61, 0.63)	(0.63, 0.65)	(0.65, 0.67)
24	*LeNet*	*G&C*	**66.59%±0.74%**	66.59%±0.74%	65.91%±0.80%	64.09%±0.96%	61.08%±0.89%	59.33%±0.88%	57.18%±0.89%
		SGD	63.16%±0.87%	63.16%±0.87%	62.02%±0.84%	60.74%±0.73%	58.21%±0.75%	57.62%±0.69%	56.24%±0.55%
16	*LeNet*	*GlobalPSO*	59.81%±0.92%	59.43%±0.54%	59.43%±0.55%	-	-	-	- -
		G&C	**61.10%±0.98%**	61.10%±0.98%	59.54%±0.98%	59.21%±0.90%	57.53%±0.86%	57.71%±0.81%	55.06%±0.70%
		SGD	58.98%±0.69%	58.58%±0.77%	58.98%±0.69%	57.86%±0.79%	57.11%±0.61%	56.77%±0.62%	53.90%±0.50%
8	*LeNet*	*GlobalMomentumPSO*	**58.04%±0.49%**	58.04%±0.49%	57.42%±0.47%	55.76%±0.47%	55.11%±0.45%	54.27%±0.50%	53.81%±0.38%
		G&C	57.17%±0.94%	54.39%±0.80%	53.99%±0.76%	57.17%±0.94%	54.61%±0.68%	52.66%±0.66%	52.82%±0.62%
		SGD	56.76%±0.71%	56.76%±0.71%	55.02%±0.62%	54.79%±0.72%	54.62%±0.68%	53.39%±0.66%	53.53%±0.55%
4	*LeNet*	*GlobalMomentumPSO*	54.20%±0.48%	54.20%±0.48%	53.53%±0.44%	53.16%±0.46%	52.92%±0.41%	51.96%±0.38%	51.71%±0.41%
		G&C	**55.51%±0.84%**	55.51%±0.84%	53.59%±0.96%	52.78%±0.82%	52.30%±0.67%	52.38%±0.63%	54.07%±0.72%
		SGD	53.75%±0.62%	53.49%±0.68%	52.14%±0.51%	53.75%±0.62%	51.53%±0.63%	52.18%±0.66%	50.44%±0.55%
2	*LeNet*	*GlobalMomentumPSO*	51.89%±0.41%	51.89%±0.41%	51.78%±0.43%	51.29%±0.33%	50.87%±0.35%	51.22%±0.38%	51.01%±0.39%
		G&C	**52.39%±0.67%**	51.66%±0.74%	52.39%±0.67%	52.00%±0.60%	51.37%±0.56%	50.01%±0.71%	50.66%±0.62%
		SGD	51.98%±0.59%	51.93%±0.66%	51.39%±0.47%	51.98%±0.59%	51.16%±0.48%	50.65%±0.45%	50.65%±0.45%

Table 4.2. Two class Cifar10. PSO compares comparably to SGD and G&C across different training losses and numbers of samples. This shows us that, despite the large number of parameters, gradient-free solutions are implicitly regularized. G&C consistently performs better than SGD in this low data regime, PSO sometimes even outperforms G&C. Standard errors are shown. Due to the heavy computational load, some results were not obtained (dashed lines). Accuracy on the whole distribution.

Despite the limited number of samples, this regime accentuates the impact of overparameterization. Remarkably, the model consistently generalizes well relative to SGD, underscoring that the substantial volume of the good solution set alone biases the optimizer towards favorable generalization.

Could gradient-free optimizers substitute SGD?

The paper extends its analysis to the 10-class CIFAR-10/MNIST problem, where G&C becomes impractical due to exponential time complexity. Pattern Search and Greedy Random Search are employed to assess generalization dependency on gradient-based optimizers. Surprisingly, these non-gradient-based optimizers yield comparable levels of generalization benefits to SGD, despite operating without any gradient information.

The average performance difference is a mere 0.9% across varying sample sizes, datasets, and optimizer combinations. In some instances Pattern Search outperform SGD. While a 0.9% difference may appear substantial in the pursuit of state-of-the-art accuracy, it falls within the expected margin for hyperparameter tuning. Experiments in [7] and the present study *did not fine-tune the zeroth-order optimizers*, yet they achieve a level of generalization comparable to SGD.

Comparing gradient-free optimizers with SGD is vital, and it's noteworthy to consider their performance in few-shot settings with industrial-scale models. [7] shows that gradient-free Pattern Search optimizer performs comparably to SGD in the 1-shot setting. Few-shot testing assesses the model's ability to generalize to unseen tasks with limited training examples.

Unlike the common belief that enlarging a model beyond a certain threshold leads to overfitting and reduced generalization, the observed phenomenon of increase in validation accuracy as the width of the model increases [7] suggests a hypothesis: widening a neural network may expand the volume of the good function class. If it becomes feasible to identify the specific function within this class that expands with more parameters, there's potential to achieve similar benefits with a more compact model. Such a model could capture favorable properties, enhancing efficiency and

effectiveness in deep learning without unnecessary parameter inflation.

It is crucial to ascertain whether a gradient-free optimizer, such as PSO, possesses the capability to navigate the parameter space with efficiency and comprehensiveness, enabling the discovery of "good" minima. Before contemplating the utilization of PSO as an industrial optimizer, an analysis of its behavior in real-world training scenarios is imperative. Additionally, investigating the role of validation in aiding the optimizer to identify generalizing solutions is essential. The upcoming section addresses these concerns, conducting experiments to validate the Volume Hypothesis and delving into more realistic training scenarios.

4.3 Assessing Volume Hypothesis

In order to validate the volume bias as a possible explanation for the simplicity bias and for the discovery of good minima, some experiments are conducted by studying of particles amount influences the convergence of the optimizer.

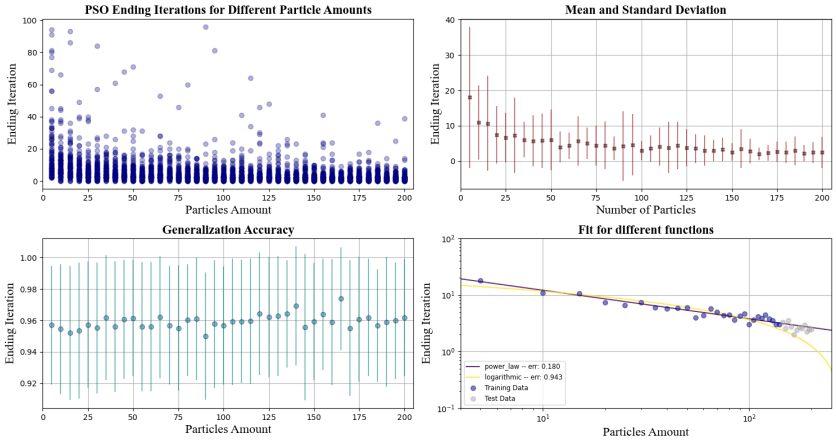


Figure 4.8. Convergence for different particles amounts, Wedge distribution on 16 samples. Model with single hidden layer having 2 hidden units. **Plot 1:** For each amount, 100 different initializations are plotted against the ending iteration. **Plot 2:** Means and standard deviations for each amount of particles (100 samples). **Plot 3:** Generalization over the entire underlying distribution. **Plot 4:** Fit of Power Law and Logarithmic function to predict future convergence rates (70-30 data split). Note that non-converging initialization are discarded after 200 optimization steps (rate of occurrence of non-converging initialization is very low).

In principle, an increase in the number of particles exploring the parameter space should facilitate the discovery of good minima, nevertheless, the optimizer must employ discerning metrics to differentiate between solutions that effectively generalize and those that do not. To asses such claims Power Law fitting is carried out, and compared to other function to assesses its power as a predictor, many functions were tested out such as Weibull, Rayleigh or exponential, in the end Power Law was the only consistent one, with small error and fitting correctly the data.

Power Law In statistics, the Power Law represents a functional relationship between two quantities. It states that a relative change in one quantity results in a relative change in the other quantity proportional to a power of the change,

regardless of the initial size of those quantities.
In this study, the adopted power law function is defined as:

$$f(x) = a \cdot x^b \tag{4.1}$$

where a is the scale parameter and b is the exponent.

One remarkable attribute of power laws is their scale invariance. For a relation $f(x) = ax^{-k}$, scaling the argument x by a constant factor c causes a proportionate scaling of the function:

$$f(cx) = a(cx)^{-k} = c^{-k} f(x) \propto f(x)$$

where $\propto$ denotes proportionality. Scaling by a constant c merely multiplies the original power-law relation by the constant c^{-k}. Consequently, all power laws with a specific scaling exponent are equivalent up to constant factors. This equivalence leads to a linear relationship when logarithms are taken of both $f(x)$ and x. The resulting straight line on the log–log plot is often referred to as the signature of a power law. In the context of real data, this straightness is a necessary, but not sufficient, condition for the data to follow a power-law relation.

If the set of good minima posses a larger volume than the bad minima set, we would expect that increasing the amount of particles exploring the parameter space can lead to improved generalization and faster convergence.
The results are illustrated in [fig 4.8] for the wedge distribution on a small model and in [fig 4.9] for the MNIST dataset, other experiments on Wedge are available at [fig 4.18]. Such results indicate that quantity of particles is a key factor for convergence. As the number of particles increases, the average required iterations for converge decrease logarithmically. Power Law is fitted to asses its predictive capability in modeling this relationship, suggesting a nonlinear relationship between particles and convergence. Notably, in this setting, generalization seems to be independent of particle count. It is important to note that the particles span a narrow range, and this setting may not sufficiently evaluate enhancements in discovering favorable minima, also, validation, a strong generalization estimator, is not considered here.

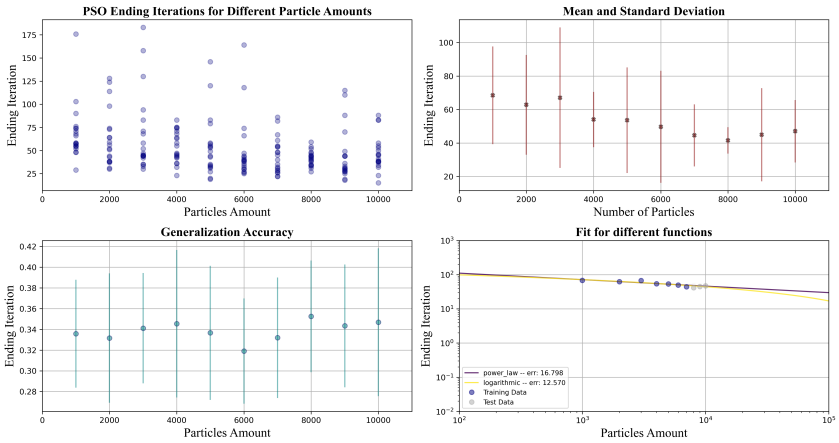


Figure 4.9. Power Law as a predictor of convergence for 10 class MNIST dataset, 16 samples. PSO with particles in range 1000-10000 training LeNet (width=0.5). Here generalization accuracy is quite stable.

Hence the apparent independence of generalization from particle count may be favored by the absence of a validation set guiding the training towards generalizing solution. In this setting the optimizer finds minima which generalization power is indistinguishable. The increase of particles shortens the discovery of minima, without validation estimation of the parameters generalizing ability, the rate of generalization is indeed the same, as particles explore parameter space with no information on favorable minima.

While the experiments on the Wedge distribution yield intriguing insights, their applicability is constrained by the small toy sample. Subsequent experiments will be conducted on larger and more complex datasets such as MNIST and CIFAR-10, allowing for a thorough examination of real-world data.
Experiments on these datasets are conducted on the whole 10 classes thanks to the efficiency of PSO, a much heavier problem, in terms of computation, for Guess & Check. Sample amount is 16.

4.3.1 Experimental Evaluation and Generalization Analysis

The preceding findings prompt consideration of the applicability of PSO in real-world scenarios for effectively identifying optimal minima. In practical training scenarios, the comparison of accuracy and loss on training data against validation data provides an estimate of the actual generalization power of the current set of parameters, and of the degree of overfit of the model.

To simulate such experiments and investigate the accuracy's dependence on generalization concerning the particle count, a parallel experiment was conducted. This involved using validation set to efficiently navigate toward favorable "good" minima, using it as an estimate.

The outcomes are illustrated in Figure 4.10. Consistent with earlier observations, the number of iterations required for convergence diminishes as the particle count

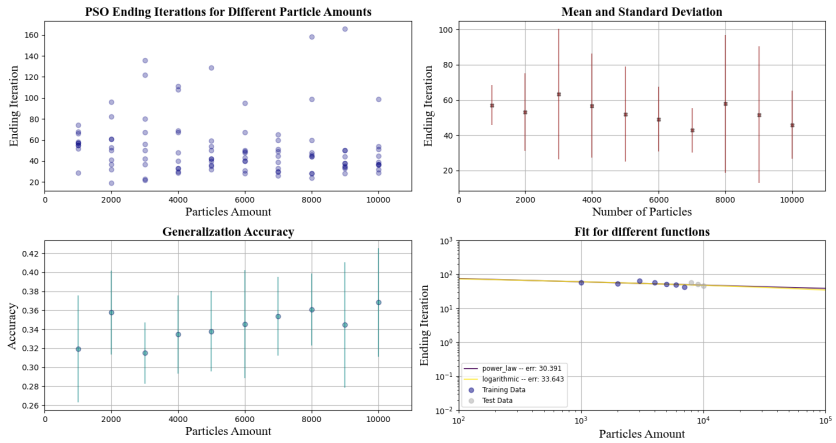


Figure 4.10. Power Law as a predictor for 10 class MNIST dataset, 16 samples. PSO with particles in range 1000-10000 training LeNet (width=0.5). **Plot 3:** Generalization Accuracy on whole MNIST dataset shown. Using validation the generalization possess an increasing trend.

increases. Notably, a discernible trend towards increased accuracy becomes apparent with a higher particle count. It is essential to emphasize that the evaluation of generalization accuracy employs metrics on the entire 10-class MNIST test dataset, while training performance is assessed based on 16 samples.

Accuracy on the validation set is depicted in [fig 4.12], revealing a consistent trend mirroring the observed generalization pattern in Figure 4.10, indicating that validation is a strong estimator of the generalization performance.

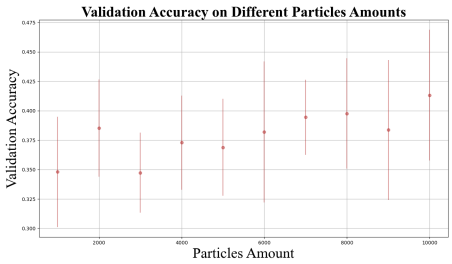


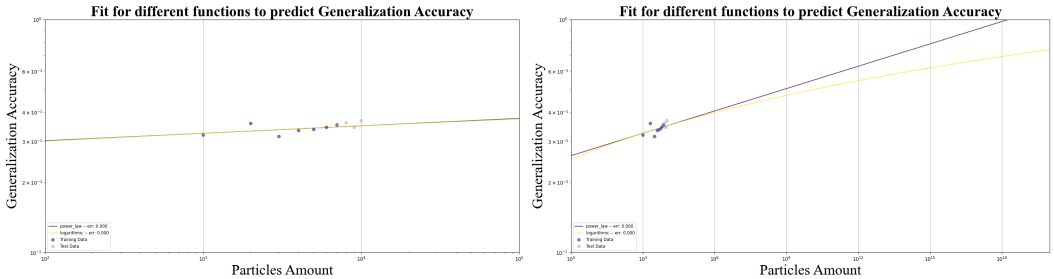
Figure 4.11. Validation accuracy trend as PSO increases its particles.

Notably, the generalization accuracy exhibits a discernible increasing tendency. It justifies thoughtful consideration to assess the feasibility of employing Power Law as an estimator for accuracy, shown in Figure 4.11.

The discernible correlation between the number of particles and generalization accuracy yields a promising outcome, affirming the validity of the volume hypothesis. Particle Swarm Optimization, leveraging the validation set as a benchmark for generalization, demonstrates its efficacy in steering parameter configurations toward favorable minima. This estimation serves as a potent signal for the optimizer, allowing it to discriminate between "bad" and "good" minima—a capability hindered in the absence of a validation set.

It's crucial to approach such predictions with caution, given the limited dataset of only 7 fitted points and 3 tested points. Although convergence behavior remains relatively regular with a small number of particles, as evidenced in Figure 4.18, showcasing convergence rates and power law estimates for both low and high particle counts, understanding *generalization* trends in extreme cases, particularly as accuracy approaches 100%, presents a more intricate challenge. This intricacy merits future investigation.

These findings provide a comprehensive evaluation of the volume hypothesis, instilling optimism about the potential application of PSO as an optimizer in real-world



(a) Power Law fit to predict generalization accuracy on the interval $[10^3, 10^5]$

(b) Power Law fit to predict generalization on the larger interval $[1, 10^{10}]$

Figure 4.12. Use of Power Law and Logarithmic function as a predictor of generalization.

training scenarios. However, it's essential to acknowledge the inherent limitations of this optimizer. As depicted in the plots, the number of particles required for robust generalization may become impractical as the algorithm scales.
It is important to note that sample amount plays an important role in convergence, quality of minima, and generalization. These experiments were carried out only on 16 samples, consequently a huge number of particles is predicted to achieve high generalization. When increasing the training samples, fewer particles will be needed, at the expense of more data to process to obtain similar results. In order to increase overall performance it is needed to augment in tandem data, computing power and model size. Such observation particularly relates to the neural scaling law phenomenon, where model performance increase as the dataset size, model size and amount of computation scale up, studied in [20] and [15].

The intrinsic gradient-free nature of PSO proves advantageous in this context, allowing for straightforward parallelization across multiple processing units with minimal communication requirements. Further insights into this aspect are detailed in Section 4.5.

4.4 Other Toy Samples Datasets

Following the Wedge distribution, training continued on some other datasets, like the Swiss-roll and Concentric-circles found in [17].

Concentric circles distribution can be seen in fig 4.13, experiments are carried out on 50 samples datasets with a 70-30 data split respectively for training and testing.

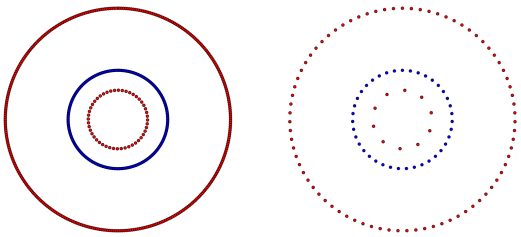


Figure 4.13. Concentric Circles.
Distribution from "Huang et al., 2020 [17]"

The concentric circles distribution is specially crafted to experiment what neural networks cannot solve. Essentially since neural networks solve complex classification problems by finding "flat" minima with class boundaries that are stable to parameter perturbations, when distance between the inner rings is large, a neural network consistently finds a well-behaved circular boundary as in [fig 4.14a].

The wide margin of this classifier makes the minimizer "flat," and the resulting high volume makes it likely to be found by SGD. However by pinching the margin between the two inner rings, the well-behaved minima can be removed, in this case networks' decision boundaries are shown in [fig 4.14b] and [fig 4.14c].

SGD finds networks that cherry-pick red points, and arc away from the more numerous blue points to maintain a large margin. In contrast, a simple circular decision boundary as in Figure 4.14a would pass extremely close to all points on the inner rings, making such a small margin solution less stable under perturbations and unlikely to be found by SGD.

Experiments in this section begin with wide margin between the two inner rings. The models used to attack these datasets are fully connected, hence having biases in their hidden layers too, differently to the architectures employed in [7] and in the

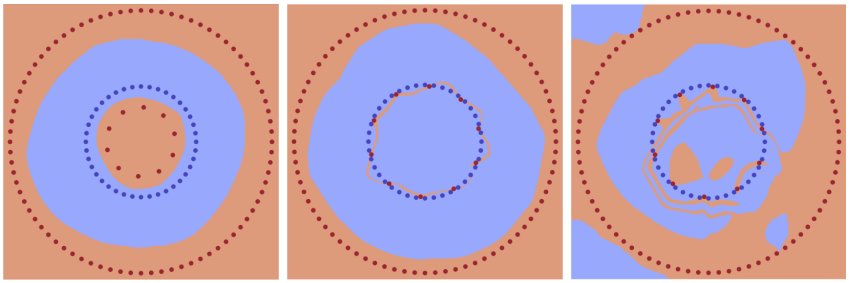


Figure 4.14. A neural network fails to solve a classification problem when the ideal solution is "sharp." Huang et al. 2020 [17].

first experiments attacking decision boundaries.

Models consist of architectures with 6 hidden layers of varying size, from 15 hidden units to 50 per layer.

Figure 4.16 illustrates decision boundaries found with PSO. The wide margin setting enables neural networks to correctly find minimizers, however they do not possess a so well-behaved minima, decision boundaries are rather complex and effectively generalizing solutions are coupled with more inconsistent solutions.
Such results suggest that not so generalizing solutions are easier to be found with PSO, indicating that the volume of such minima is larger than the volume of the set of good minima.
It is interesting to observe that often the more complex decision boundaries outside the class circles are coupled with irregular decision boundaries for the classification of inner rings.

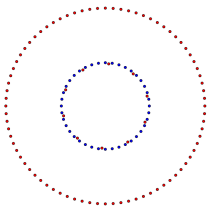


Figure 4.15. Concentric Circles with narrow margin (gap=0.01).

The reduction in margin gap significantly diminishes the volume of effective solutions, as demonstrated in [17]. This observation is further supported by the challenges faced by PSO in finding solutions under these conditions. Being a meta-heuristic algorithm, PSO does not guarantee convergence, and in this study, with up to 40,000 particles, it struggles to locate the relatively scarce good minima within the experiment settings. This problem is much harder to tackle, and the limited computing resources used in this study are not enough to find and compare the quality of different minima. While the inability to discover these solutions might be perceived negatively, it accurately mirrors the significant volume disparity of solutions inherent in the dataset. However, little can

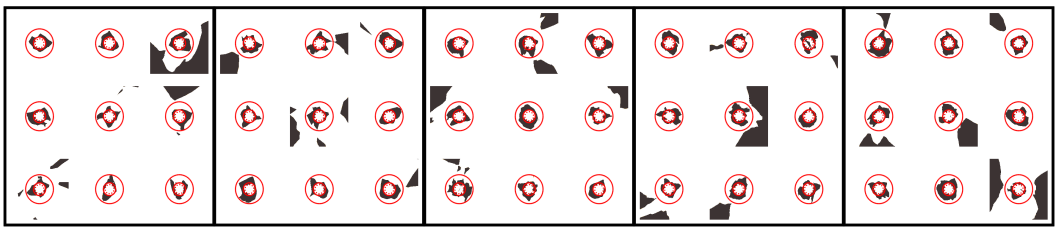


Figure 4.16. Quite irregular PSO Decision boundaries on wide margin concentric circles. Models with 3 Hidden layers, respectively 15, 20, 30, 40, 50 hidden units per layer. 50 samples dataset.

be said on the volume of good and bad minima as none has been found using PSO.

Spirals dataset The non-convergence of the algorithm takes on a similar significance in the context of the spirals dataset, as outlined in [17]. In this scenario, PSO encounters challenges in locating solutions as well. The paper suggests delving into the heightened sharpness of minima as they relate to a decrease in generalization accuracy. However, the settings and experiments in this study were unable to identify such minima for the spirals dataset.

The idea behind this is illustrated in [fig 2.2], where a "flat" solution and a "sharp" solution are compared. In the paper there are shown different decision boundaries for different level of generalization, as illustrated in figure [fig 4.17]

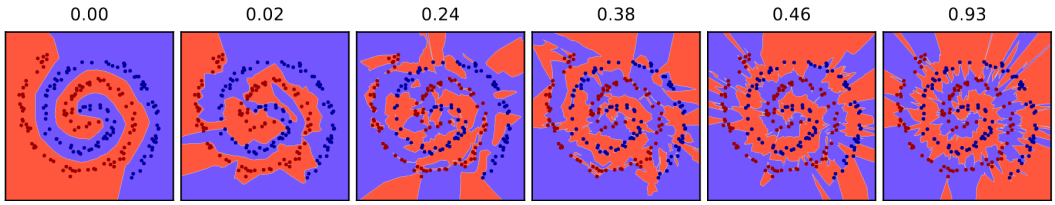


Figure 4.17. Swissroll decision foundaries for various levels of generalization gap (difference in train generalization and test generalization, indicated above plots). Huang et al. 2020 [17].

4.5 PSO Comparison with SGD, Parallel Distributed Optimization

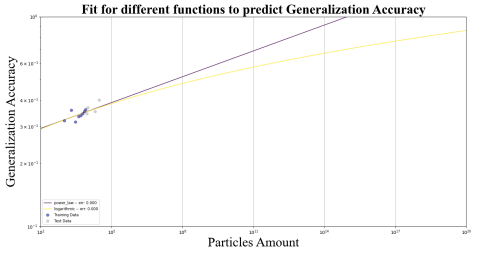
Leveraging Hardware Parallelism In contrast to optimization algorithms that heavily rely on gradient information, Particle Swarm Optimization (PSO) presents a unique opportunity for parallelization across multiple CPUs and GPUs.
This inherent parallelizability allows for efficient exploration of the parameter space with the potential to exploit the computational power of modern hardware and the high availability of processing power.
In a distributed PSO setup, each processing unit can operate independently, communicating only the best cost found thus far at each iteration, extremely reducing the load of communication, or, depending on the chosen topology, sharing the final position and cost. For instance each individual GPU could optimize the problem considering its particles as the set of closest neighbors. This communication strategy minimizes the need for extensive information exchange, making PSO particularly suited for parallel execution.

Implicit Regularization and Exploration of Good Minima Implicit regularization, often associated with algorithms that explore the parameter space extensively, becomes a compelling aspect in the context of PSO. The volume hypothesis posits that a larger exploration of the parameter space may lead to the discovery of good minima. In the case of PSO, the abundance of particles traversing the parameter space enhances the likelihood of encountering regions associated with lower costs that effectively generalize.

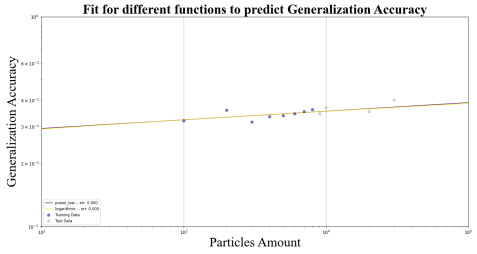
Exploit Validation Set for Minima Discovery The involvement of a validation set further contributes to the effectiveness of PSO. By leveraging the validation set, the algorithm can discern not only the global minima but also identify regions of the parameter space that generalize well. This approach aligns with the implicit regularization effect, as the diverse exploration facilitated by numerous particles aids in the discovery of solutions that exhibit superior generalization performance.

The parallel distributed nature of PSO, coupled with its implicit regularization characteristics and utilization of validation sets, positions it as a powerful optimization technique. The ability to explore the parameter space with a multitude of particles in parallel opens avenues for the discovery of high-quality minima, demonstrating the potential of distributed PSO in addressing complex optimization challenges coupled with parallel powerful computing resources.

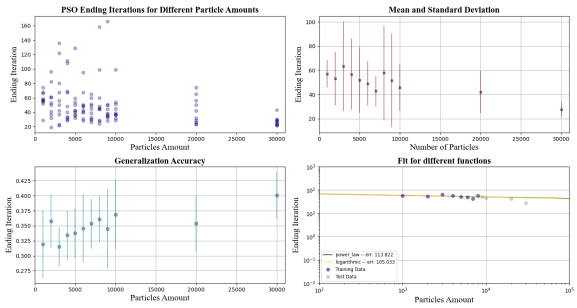
As an area for potential future research, Particle Swarm Optimization (PSO) can be synergistically integrated with Stochastic Gradient Descent (SGD). This dual-phase optimizer first leverages PSO for an extensive exploration of the parameter space. Subsequently, it refines the parameters utilizing gradient information through SGD. This process may entail applying SGD to the best-performing set of parameter configurations, ensuring a blend of efficient parameter space exploration and high-quality optimization results.



Power Law plot on “long” range.

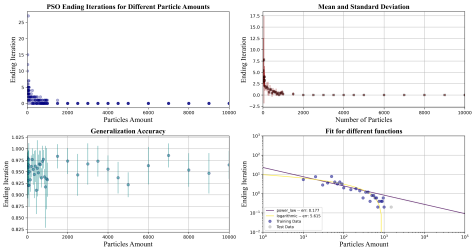


Power Law plot on “short” range.

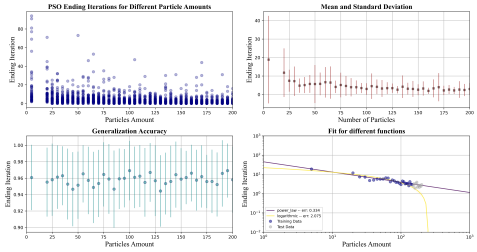


Plots to show convergence, generalization accuracy and Power Law fit.

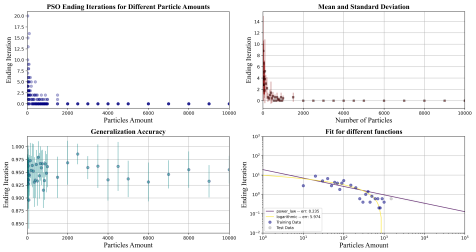
Figure 4.18. Augmenting the number of PSO particles to 20,000 and 30,000. 10 different random initializations per particle amount, 16 samples on 10 class MNIST. The objective is to assesses the coherence of generalization and convergence trends with the Power Law predictions outlined in section 4.3.1. Due to computational constraints, the particle count is capped at 30,000 for this investigation. Remarkably, the observed generalization accuracy trend aligns with the Power Law prediction, and convergence speed seems to decrease even faster than Power Law prediction. This intriguing observation prompts further analysis of these metrics at higher particle and sample counts in future research.



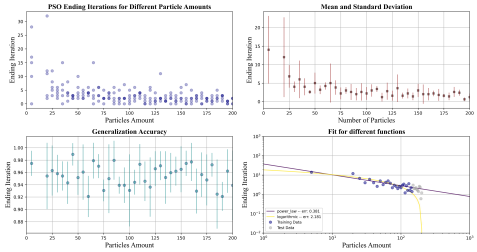
(a) 2 hidden units



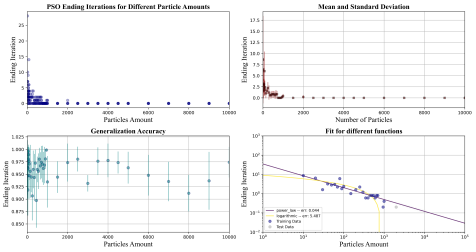
(b) 2 hidden units



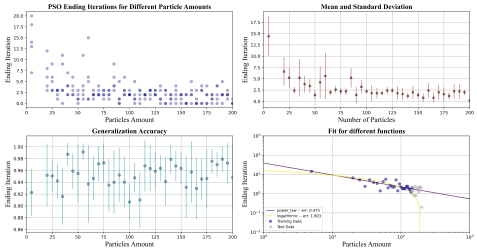
(c) 4 hidden units



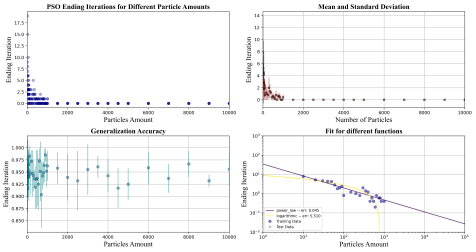
(d) 4 hidden units



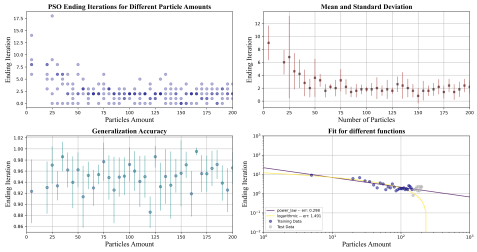
(e) 10 hidden units



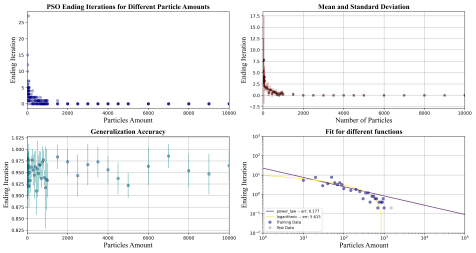
(f) 10 hidden units



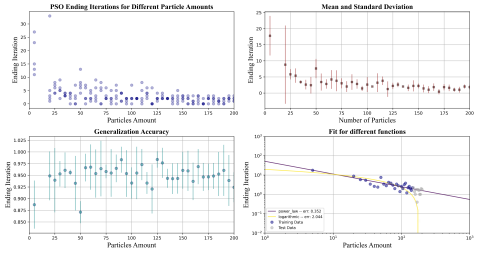
(g) 15 hidden units



(h) 15 hidden units



(i) 20 hidden units



(j) 20 hidden units

Figure 4.19. PSO Power Law estimates on Wedge distribution. **Left:** Particles amount up to 10000. Mean on 50 runs. **Right:** Particles up to 200. Mean over 5 runs.

Chapter 5

Conclusion

In conclusion, this thesis delved into the exploration of implicit bias in gradient-based algorithms, with a specific focus on the volume hypothesis, advocating for disparity in volume for the set of "good" 'minima and "bad" minima. The investigation employed Particle Swarm Optimization (PSO) as a derivative-free and zeroth-order optimization method to assess this disparity in volume.

The background on implicit bias extended to different models, including linear and nonlinear networks, homogeneous neural networks, matrix factorization, and dynamic balance in gradient methods. The investigation also touched upon the critical topic of simplicity bias and the blessing of dimensionality.

The thesis then introduced Particle Swarm Optimization, a metaheuristic optimizer suitable for high-dimensional search spaces, providing a detailed overview of the algorithm, parameter selection, and code implementation to allows GPU cuda-acceleration.

The experimental design involved testing PSO on decision boundaries of specific architectures, drawing comparisons with the results obtained from SGD and the Guess & Check optimizer as presented in Chiang et al. (2023) [7]. Subsequent experiments were conducted on well-known datasets such as MNIST and Cifar10, analyzing generalization accuracy. The use of Power Law aided in estimating convergence rates and understanding the generalization capabilities of PSO.

Empirical insights into model generalization were gained through experiments on overparameterization, wedge distribution, and different architectures, including LeNet, dense layers and bias-only layers.

Insights provided by the results obtained are of great interest. Even though PSO is a gradient-free optimizer as G&C, and explores the parameter space more effectively than G&C, its behaviour is sometimes worse. Topology of the swarm may play a crucial role in the optimization process.
Nevertheless, PSO is a promising algorithm for the exploration and discovery of favorable minima. The strategic utilization of validation set proves to be instrumental in providing a robust estimate of generalizing optima, thereby ensuring accurate generalization. The analysis based on Power Law reveals a positive correlation between an increase in the number of particles and enhanced generalization accuracy when using validation set as an estimator of generalization performance, further endorsing the potential of PSO in achieving effective optimization.

The efficacy of gradient-free optimizers beyond Stochastic Gradient Descent (SGD) in uncovering generalizing minima has been well-established. However, the journey towards achieving a consistently effective exploration and identification of such solutions is ongoing. This study critically examines the strengths and weaknesses of Particle Swarm Optimization (PSO) as an industrial optimizer. Notably, the analysis does not yet extend to computing time, a crucial aspect that warrants exploration. Before integrating PSO into practical training scenarios, a comprehensive comparison with SGD is imperative to assess its performance and ensure its suitability for realistic training conditions.

In conclusion, this thesis contributes valuable insights into the implicit bias of gradient-based algorithms, particularly in the context of simplicity bias and the volume hypothesis. Effective capacity of Neural Networks is large enough to make models memorize the training data. This phenomenon poses a conceptual challenge to statistical learning theory, as traditional measures of model complexity struggle to elucidate the generalization ability of expansive artificial neural networks. The empirical ease of optimization, even when resulting models fail to generalize effectively, as observed by Zhang et al. [55], suggests that the reasons for the ease of optimization may differ from the true causes of generalization.
The concept of simplicity bias posits that standard training procedures exhibit a propensity to discover simple models. Meanwhile, the volume hypothesis raises questions about whether such bias and resulting generalization are contingent on the abundance of generalizing minima. It is conceivable that models may not actively generalize but instead memorize training data, *adhering to the simplest features that facilitate accurate generalization.* This underscores the need for a deeper understanding of the mechanisms and causes of artificial learning, which remains an open and actively researched topic.

The insights presented in this thesis not only contribute to the current understanding of neural network behavior but also lay the groundwork for further exploration. The implications of these findings open new avenues for the application of derivative-free optimization algorithms in the realm of deep learning, providing promising directions for future research.

Reproducibility Statement

All experiments in this study were conducted using predetermined random seeds, ensuring complete reproducibility. To replicate the results, the scripts which are available at [31] can be executed.

Acknowledgments

I extend my sincere gratitude to Professor Iacopo Masi, whose unwavering support and guidance greatly enhanced this research. Working alongside him has been an invaluable learning experience, enjoying the exploration of the wonderful topics researched.
Heartfelt thanks are due to my family, whose constant support and attention have been a source of strength and encouragement. I appreciate their enduring emotional support and the assistance provided whenever needed.
I am also grateful to my friends, each of whom holds a special place in my heart. Their diverse perspectives and approaches to life have enriched my own journey, and I value the moments we've shared.
A debt of gratitude extends to all the educators who, from an early age, have contributed to shaping me into a better person. Special thanks go to the Scout community, which has illuminated my path with thoughtful guidance and provided continuous growth in both thought and spirit.
I also want to express my appreciation to my basketball teammates, whose camaraderie provided a welcome distraction through the joy of this wonderful sport.

Bibliography

[1] Sanjeev Arora et al. *Implicit Regularization in Deep Matrix Factorization*. 2019. arXiv: 1905.13655 [cs.LG].

[2] Shahar Azulay et al. *On the Implicit Bias of Initialization Shape: Beyond Infinitesimal Mirror Descent*. 2021. arXiv: 2102.09769 [cs.LG].

[3] Peter Bartlett, Dylan J. Foster, and Matus Telgarsky. *Spectrally-normalized margin bounds for neural networks*. 2017. arXiv: 1706.08498 [cs.LG].

[4] Mohamed Ali Belabbas. *On implicit regularization: Morse functions and applications to matrix factorization*. 2020. arXiv: 2001.04264 [cs.LG].

[5] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32 (July 2019), pp. 15849–15854. DOI: 10.1073/pnas.1903070116. URL: https://doi.org/10.1073%2Fpnas.1903070116.

[6] Hao Chen and Abhinav Shrivastava. *Group Ensemble: Learning an Ensemble of ConvNets in a single ConvNet*. 2020. arXiv: 2007.00649 [cs.CV].

[7] Ping-yeh Chiang et al. "Loss Landscapes are All You Need: Neural Network Generalization Can Be Explained Without the Implicit Bias of Gradient Descent". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=QC10RmRbZy9.

[8] Lenaic Chizat and Francis Bach. *Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss*. 2020. arXiv: 2002.04486 [math.OC].

[9] Laurent Dinh et al. *Sharp Minima Can Generalize For Deep Nets*. 2017. arXiv: 1703.04933 [cs.LG].

[10] Spencer Frei et al. *Implicit Bias in Leaky ReLU Networks Trained on High-Dimensional Data*. 2022. arXiv: 2210.07082 [cs.LG].

[11] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. *Size-Independent Sample Complexity of Neural Networks*. 2019. arXiv: 1712.06541 [cs.LG].

[12] Suriya Gunasekar et al. *Characterizing Implicit Bias in Terms of Optimization Geometry*. 2020. arXiv: 1802.08246 [stat.ML].

[13] Suriya Gunasekar et al. *Implicit Regularization in Matrix Factorization*. 2017. arXiv: 1705.09280 [stat.ML].

[14] Niv Haim et al. *Reconstructing Training Data from Trained Neural Networks*. 2022. arXiv: 2206.07758 [cs.LG].

[15] Tom Henighan et al. *Scaling Laws for Autoregressive Generative Modeling*. 2020. arXiv: 2010.14701 [cs.LG].

[16] Sepp Hochreiter and Jürgen Schmidhuber. "Flat Minima". In: *Neural Computation* 9.1 (Jan. 1997), pp. 1–42. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.1.1. eprint: https://direct.mit.edu/neco/article-pdf/9/1/1/813385/neco.1997.9.1.1.pdf. URL: https://doi.org/10.1162/neco.1997.9.1.1.

[17] W. Ronny Huang et al. *Understanding Generalization through Visualizations*. 2020. arXiv: 1906.03291 [cs.LG].

[18] Ziwei Ji and Matus Telgarsky. *Directional convergence and alignment in deep learning*. 2020. arXiv: 2006.06657 [cs.LG].

[19] Ziwei Ji and Matus Telgarsky. *Gradient descent aligns the layers of deep linear networks*. 2019. arXiv: 1810.02032 [cs.LG].

[20] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG].

[21] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[22] Nitish Shirish Keskar et al. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. 2017. arXiv: 1609.04836 [cs.LG].

[23] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *University of Toronto* (May 2012).

[24] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* (2010).

[25] Hao Li et al. *Visualizing the Loss Landscape of Neural Nets*. 2018. arXiv: 1712.09913 [cs.LG].

[26] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. *Algorithmic Regularization in Over-parameterized Matrix Sensing and Neural Networks with Quadratic Activations*. 2019. arXiv: 1712.09203 [cs.LG].

[27] Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. *Towards Resolving the Implicit Bias of Gradient Descent for Matrix Factorization: Greedy Low-Rank Learning*. 2021. arXiv: 2012.09839 [cs.LG].

[28] Kaifeng Lyu and Jian Li. *Gradient Descent Maximizes the Margin of Homogeneous Neural Networks*. 2020. arXiv: 1906.05890 [cs.LG].

[29] Kaifeng Lyu et al. *Gradient Descent on Two-layer Nets: Margin Maximization and Simplicity Bias*. 2021. arXiv: 2110.13905 [cs.LG].

[30] Chao Ma and Lexing Ying. *On Linear Stability of SGD and Input-Smoothness of Neural Networks*. 2021. arXiv: 2105.13462 [cs.LG].

[31] Iacopo Masi and Matteo De Sanctis. *PSO-NN*. https://github.com/iacopomasi/PSO-NN.git. 2023.

[32] Rotem Mulayoff, Tomer Michaeli, and Daniel Soudry. "The Implicit Bias of Minima Stability: A View from Function Space". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 17749–17761. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/944a5ae3483ed5c1e10bbccb7942a279-Paper.pdf.

[33] Mor Shpigel Nacson et al. "Implicit Bias of the Step Size in Linear Diagonal Neural Networks". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 16270–16295. URL: https://proceedings.mlr.press/v162/nacson22a.html.

[34] Preetum Nakkiran et al. *Deep Double Descent: Where Bigger Models and More Data Hurt*. 2019. arXiv: 1912.02292 [cs.LG].

[35] Kamil Nar and S. Shankar Sastry. *Step Size Matters in Deep Learning*. 2018. arXiv: 1805.08890 [cs.LG].

[36] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. *Norm-Based Capacity Control in Neural Networks*. 2015. arXiv: 1503.00036 [cs.LG].

[37] Noam Razin and Nadav Cohen. *Implicit Regularization in Deep Learning May Not Be Explainable by Norms*. 2020. arXiv: 2005.06398 [cs.LG].

[38] Noam Razin, Asaf Maman, and Nadav Cohen. *Implicit Regularization in Hierarchical Tensor Factorization and Deep Convolutional Neural Networks*. 2022. arXiv: 2201.11729 [cs.LG].

[39] Noam Razin, Asaf Maman, and Nadav Cohen. *Implicit Regularization in Tensor Factorization*. 2021. arXiv: 2102.09972 [cs.LG].

[40] Itay Safran, Gal Vardi, and Jason D. Lee. *On the Effective Number of Linear Regions in Shallow Univariate ReLU Networks: Convergence Guarantees and Implicit Bias*. 2023. arXiv: 2205.09072 [cs.LG].

[41] Roei Sarussi, Alon Brutzkus, and Amir Globerson. *Towards Understanding Learning in Neural Networks with Linear Teachers*. 2021. arXiv: 2101.02533 [cs.LG].

[42] Harshay Shah et al. *The Pitfalls of Simplicity Bias in Neural Networks*. 2020. arXiv: 2006.07710 [cs.LG].

[43] Daniel Soudry et al. *The Implicit Bias of Gradient Descent on Separable Data*. 2022. arXiv: 1710.10345 [stat.ML].

[44] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[45] Nadav Timor, Gal Vardi, and Ohad Shamir. *Implicit Regularization Towards Rank Minimization in ReLU Networks*. 2022. arXiv: 2201.12760 [cs.LG].

[46] Gal Vardi. *On the Implicit Bias in Deep-Learning Algorithms*. 2022. arXiv: 2208.12591 [cs.LG].

[47] Gal Vardi and Ohad Shamir. *Implicit Regularization in ReLU Networks with the Square Loss*. 2021. arXiv: 2012.05156 [cs.LG].

[48] Gal Vardi, Ohad Shamir, and Nathan Srebro. *The Sample Complexity of One-Hidden-Layer Neural Networks*. 2022. arXiv: 2202.06233 [cs.LG].

[49] Gal Vardi, Gilad Yehudai, and Ohad Shamir. *Gradient Methods Provably Converge to Non-Robust Networks*. 2022. arXiv: 2202.04347 [cs.LG].

[50] Yuqing Wang et al. *Large Learning Rate Tames Homogeneity: Convergence and Balancing Effect*. 2022. arXiv: 2110.03677 [cs.LG].

[51] Blake Woodworth et al. *Kernel and Rich Regimes in Overparametrized Models*. 2020. arXiv: 2002.09277 [cs.LG].

[52] Lei Wu, Chao Ma, and Weinan E. "How SGD Selects the Global Minima in Over-parameterized Learning: A Dynamical Stability Perspective". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/6651526b6fb8f29a00507de6a49ce30f-Paper.pdf.

[53] Lei Wu, Mingze Wang, and Weijie Su. *The alignment property of SGD noise and how it helps select flat minima: A stability analysis*. 2022. arXiv: 2207.02628 [stat.ML].

[54] Chulhee Yun, Shankar Krishnan, and Hossein Mobahi. *A Unifying View on Implicit Bias in Training Linear Neural Networks*. 2021. arXiv: 2010.02501 [cs.LG].

[55] Chiyuan Zhang et al. *Understanding deep learning requires rethinking generalization*. 2017. arXiv: 1611.03530 [cs.LG].