Fractal Dynamics in Neural Network Boundaries

Omkar Vengurlekar
MS RAS AI
Arizona State University
ovengurl@asu.edu
1225369067

Shantanu Patne
MS EEE(AME)
Arizona State University
sapatne@asu.edu
1229496959

Aniruddha Sawant
MS CEN
Arizona State University
asawan15@asu.edu
1232066552

Abstract

The trainability of neural networks, or the ease with which they can learn from data, is a critical area of study. Recent research suggests that the boundaries delineating trainable from non-trainable configurations in parameter space are not merely irregular, but fractal. This study delves into the implications of such a fractal boundary, highlighting the intricate balance between neural network architecture, initial conditions, and training success. Through a series of experiments, this report validates the fractal nature of trainability boundaries and discusses its implications for understanding the dynamics and stability of neural network training. The findings offer significant insights into the optimization of neural network architectures and training methods, potentially leading to more robust and efficient learning algorithms.

1 Introduction

In the rapidly evolving field of artificial intelligence, neural networks stand out as a cornerstone technology, driving advancements in everything from image recognition to natural language processing. The effectiveness of neural networks hinges significantly on their ability to learn from and adapt to a wide array of data inputs. This learning process, known as training, is influenced by a multitude of factors including network architecture, optimization algorithms, and the nature of the training data itself. However, a lesser-known but potentially critical aspect of neural network training involves understanding the boundary conditions under which these systems transition from trainable to non-trainable states.

Recent research[1, 2] suggests that these boundary conditions may not be straightforward or smooth but are instead fractal. A fractal, broadly defined, is a complex structure that shows self-similarity across different scales and can include mathematical sets that exhibit a fine structure, often surprisingly detailed, at arbitrarily small scales[3, 4]. The notion that the boundary of neural network trainability might also be fractal introduces fascinating questions about the fundamental nature of learning systems and their sensitivities to initial conditions and parameter settings.

To investigate these fractal boundaries, our study conducts experiments using neural networks with varying architectures—specifically altering their width and depth. Additionally, the experiments explore the impact of different activation functions, namely the hyperbolic tangent (tanh) and identity functions, on the trainability of these networks. These variations are critical as they help illuminate how different structural configurations and nonlinear transformations influence the location and nature of the trainability boundaries.

This report aims to explore the implications of fractal boundaries in neural network trainability. It begins with an overview of how neural networks are trained, setting the stage for a deeper investigation into the nature of these training boundaries. The discussion then shifts to empirical evidence supporting the fractal nature of trainability boundaries and concludes with insights into the potential impacts of these findings on future neural network research and applications.

2 Fundamentals of Neural Network Training

Understanding the fundamentals of the complex neural network training process is essential for exploring the intricate nature of trainability boundaries in neural networks. This section covers the basic architecture of neural networks, the training process including forward propagation and back-propagation, and the role of loss functions and optimizers.

2.1 Neural Network Architecture

A neural network typically consists of layers of interconnected nodes or neurons, which are organized into three main types of layers: input, hidden, and output. The input layer receives the data, the hidden layers process the data through a series of transformations, and the output layer produces the final prediction or classification.

- **Input Layer:** This layer receives the raw input data and passes it to the first hidden layer.
- **Hidden Layers:** These layers apply weights to the inputs and pass them through activation functions, which introduce non-linearity into the system that allows the network to learn complex patterns.
- **Output Layer:** Depending on the task, this layer can produce a vector of values for regression, a probability distribution for classification, or other types of outputs.

2.2 Forward Propagation

In forward propagation, input data is passed through the network from the input layer to the output layer. At each layer, the input is multiplied by the layer's weights, and a bias is added. The result is then passed through an activation function, which is crucial for the network's ability to model non-linear phenomena. Common activation functions include ReLU, sigmoid, tanh, and identity.

2.3 Back-propagation and Optimization

Back-propagation is the key algorithm at the heart of training a neural network. It involves adjusting the weights of the network to minimize the error between the predicted output and the actual target values. This process is performed using the following steps:

- **Error Calculation:** The difference between the actual output and the predicted output is calculated using a loss function. Common loss functions include mean squared error for regression tasks and cross-entropy loss for classification tasks. A typical representation of Mean Squared Error is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

  where,
  - n is the number of data points.
  - y_i represents the actual values.
  - $\hat{y}_i$ represents the predicted values.

- **Gradient Calculation:** The gradients of the loss function concerning each weight in the network are calculated using the chain rule of calculus. This process is efficiently performed using automatic differentiation tools in modern machine-learning libraries.

$$\frac{\partial}{\partial w_{jk}} MSE = \frac{\partial}{\partial w_{jk}} \left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \right) = \frac{2}{n}\sum_{i=1}^{n} -(y_i - \hat{y}_i) \cdot \frac{\partial \hat{y}_i}{\partial w_{jk}} \tag{2}$$

  where,
  - n is the number of data points.
  - y_i is the actual value for the i-th data point.
  - $\hat{y}_i$ is the predicted value for the i-th data point, which is a function of the weights w.
  - $\frac{\partial}{\partial w}$ is the partial derivative of the loss function with respect to a weight.

- **Weight Update:** The weights are then updated using the calculated gradients. This is typically done using optimization algorithms such as stochastic gradient descent (SGD) or its variants like Adam and RMSprop. The SGD optimization algorithm is a dynamic system and often displays a chaotic nature, meaning that varying the initial values and step size can result in divergent behaviour[5, 6, 7]. These optimizers adjust the weights to minimize the loss, with the learning rate as a key parameter controlling the size of the weight updates. A typical update step is given by:

$$w_{new} = w_{old} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w} \tag{3}$$

  where,
  - w_{new} represents the new value of the weight after the update.
  - w_{old} is the old value of the weight before the update.
  - η (eta) is the learning rate, a hyperparameter that determines the step size during the update.
  - $\frac{\partial \mathcal{L}}{\partial w}$ is the partial derivative of the loss function $\mathcal{L}$ with respect to the weight w.

2.4 Training Epochs and Batches

Training usually occurs over multiple iterations, known as epochs, during which the entire dataset is passed through the network multiple times. To improve efficiency and convergence, the data is often divided into smaller subsets called batches. Training in batches helps in stabilizing the update steps and often leads to faster convergence.

3 Understanding Fractals

Fractals are mathematical sets that are intricate and infinitely complex, exhibiting structure at every scale[3]. They are characterized by self-similarity, meaning that each small portion of the fractal can be viewed as a reduced-scale replica of the whole. This unique property of fractals enables them to model natural phenomena that are not easily described by traditional Euclidean geometry.

In the context of neural network trainability, the concept of fractals becomes significant when examining the parameter space - the multi-dimensional space formed by the parameters (weights, biases, and learning rates) of a neural network. The landscape of this space is determined by the performance of the network, typically measured by a loss function. It is in this high-dimensional space that the boundary between the regions where the network can effectively learn (trainable regions) and where it cannot (non-trainable regions) begins to show fractal-like properties[8, 9].

The fractal boundary suggests that the transition between learnable and non-learnable parameters is not abrupt but rather unfolds in a complex, layered fashion, with pockets of trainability nested within non-trainable regions and vice versa. This can have deeper implications for understanding how neural networks learn and how to navigate the parameter space efficiently during the training process[1].

In this study, although difficult to visualize, we see the boundary of convergence repeat itself in fascinating structures as we zoom into the original boundary. It is evident that the boundary is not a well-defined line, but instead is a dynamic system that exhibits fractal and chaotic nature. (See Appendix **??**)

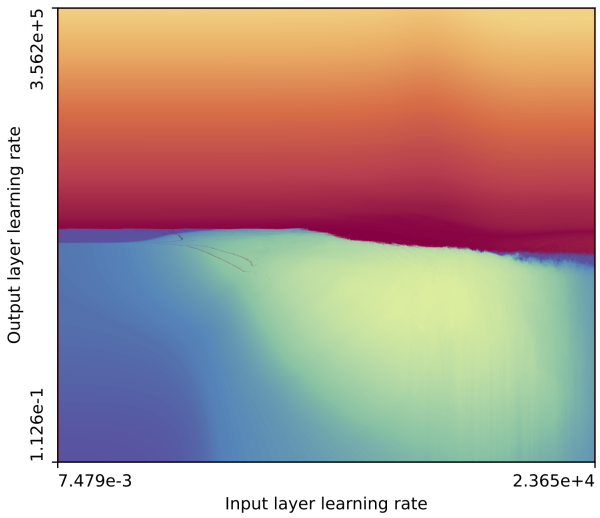

Figure 1: **Hyperparameter landscape:** This visualization illustrates the impact of learning rate hyperparameters on the success of neural network training. Each pixel represents a specific training session, defined by the learning rates of the input and output layers. Training outcomes that converged are depicted in shades of blue-green, whereas those that diverged appear in red-yellow, as designed by the author of [1]. The best-performing hyperparameters, shown in the lightest blue-green, are often found near the boundary with hyperparameters that lead to divergence, making this boundary area especially significant.

In summary, the understanding of fractals provides a fascinating lens through which we can view the fundamental aspects of neural network trainability. It underscores the non-linearity and complexity of the learning process, potentially guiding future research into more effective training algorithms and network designs.

4 Experiments

4.1 Impact of Activation Functions on Trainability

We conducted a foundational experiment to assess the impact of different activation functions on the trainability of a network with a fixed architecture. This section outlines the setup and findings of these preliminary trials.

4.1.1 Experimental Setup

We structured a neural network with intermediate complexity, similar to that in [1], setting the width at 8 units and depth at 2 layers, with the output dimension targeted at 1 unit. Randomized data drawn from the uniform distribution $\mathcal{U}(0, 1)$ was fed to the input layer to mitigate potential biases stemming from specific datasets, thereby enabling us to obtain a generalized understanding of the training boundary. The purpose of this setup was to create a consistent framework in which to test the influence of various activation functions.

Four widely-used activation functions were selected for this study:

- **Hyperbolic Tangent (tanh):** A function that scales the input data into the range between -1 and 1.
- **Sigmoid:** A function that compresses the input data into a range between 0 and 1, typically used for models where outputs are interpreted as probabilities.
- **Rectified Linear Unit (ReLU):** A linear function that outputs the input directly if it is positive, otherwise, it outputs zero.
- **Identity (linear):** A function that returns the input without any transformation.

Each of these activation functions has distinct characteristics that influence the learning dynamics and capacity of neural networks. Furthermore, we use the Lyapunov exponents (See Appendix **??**) to identify diverging and converging trajectories with perturbed weights.

4.1.2 Observations

The experiments were visualized through a series of heatmaps, each representing the training success rate across a spectrum of learning rates ranging from 10^{-3} to 10^{6} for the input and output layers. The following summarizes the key observations from each activation function's trial:

- **Tanh Activation Function:** The Tanh function presented a smoother transition from convergence to divergence in the hyperparameter landscape, denoting a certain level of robustness in training under a variety of learning rates.
- **Sigmoid Activation Function:** The sigmoid function revealed a more sensitive training boundary, where the fractal-like nature of the boundary appeared prominently but with narrower zones of successful training, indicating potential issues with vanishing gradients.
- **ReLU Activation Function:** ReLU activation function demonstrates a marked region of convergence (cool colors) abutting a stark boundary beyond which the learning rate combinations lead to divergence (warm colors). The gradient of change from convergence to divergence is abrupt, reflecting ReLU's linear nature for positive inputs, which can cause learning to either succeed dramatically or fail completely with little transition in between.
- **Identity Activation Function:** The linear activation function presents a more gradual transition between convergence and divergence across the hyperparameter space. This reflects the intrinsic nature of the Identity function, which applies no non-linearity to the input data. Its effect on the network's learning capacity is nuanced, often rendering the system more sensitive to the correct tuning of learning rates due to the absence of the mitigating effects of non-linear activation.

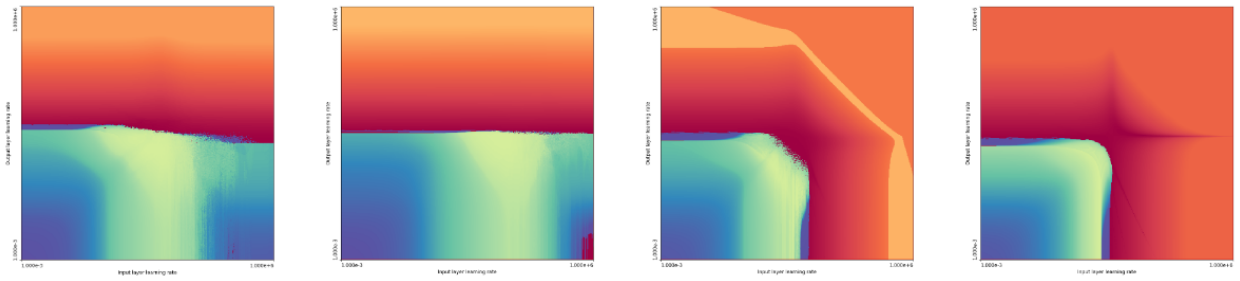

Figure 2: **Impact of Activation Functions on Learnability:** This sequence of visualizations delineates the influence of varying activation functions on the learnability of a neural network, under consistent architectural constraints and identical initial weight conditions. The depicted progression, from left to right, represents the outcomes utilizing Tanh, Sigmoid, ReLU, and Identity activation functions, respectively.

These preliminary findings provided initial insights into how the choice of activation function can significantly affect the regions of trainability in the hyperparameter landscape, thereby impacting the overall success of the network training process. The visual depictions shown in Fig. 2 from these experiments signify an intricate interplay between network architecture, activation functions, and learning rates, which is critical for advancing neural network design and optimization.

4.2 Fractal Boundaries in Neural Networks

In our quest to comprehend the complex nature of neural network trainability, we embarked on a series of experiments focusing on the fractal boundaries that demarcate the trainable and untrainable parameter spaces. The following describes our experimental setup and methodology:

4.2.1 Experimental Setup

The network's architecture featured two linear layers; the first layer projected the 2-dimensional input into a similarly sized 2-dimensional hidden space (2,2), and the subsequent second linear layer distilled this representation into a 1-dimensional output (2,1), defining a network depth of one.

4.2.2 Expanding Network Complexity

To dissect the influence of architectural complexity on the network's ability to be trained, we progressively expanded both the width and depth of the network. This expansion involved scaling the width of the hidden layers to 4, 8, 16, and 32 units, as well as escalating the network's depth with additional layers to reach depths of 1, 2, and 3, respectively.

In each architectural permutation, we conducted training across an expansive matrix of learning rate combinations. These combinations were carefully selected to assign specific learning rates for the input and output layers, thus creating a diverse landscape of learning rate pairs laid out across the plots—input layer rates along the x-axis and output layer rates along the y-axis.

4.2.3 Methodology and Observations

For our loss function, we employed the Mean Squared Error (MSE), renowned for its effectiveness in regression tasks and its propensity to provide clear gradients for back-propagation. The experiments were constrained to 100 steps to standardize the evaluation of trainability across the various configurations. We use Fractal Dimensions (See Appendix **??**) and Lyapunov Exponents to measure and characterize the nature of the trainable boundary.

Utilizing the JAX framework—a library known for its high-performance numerical computation and automatic differentiation capabilities—we orchestrated our experimental runs. This framework facilitated the efficient implementation of the training process, allowing us to iteratively adjust the weights based on the gradients calculated from the MSE loss function.

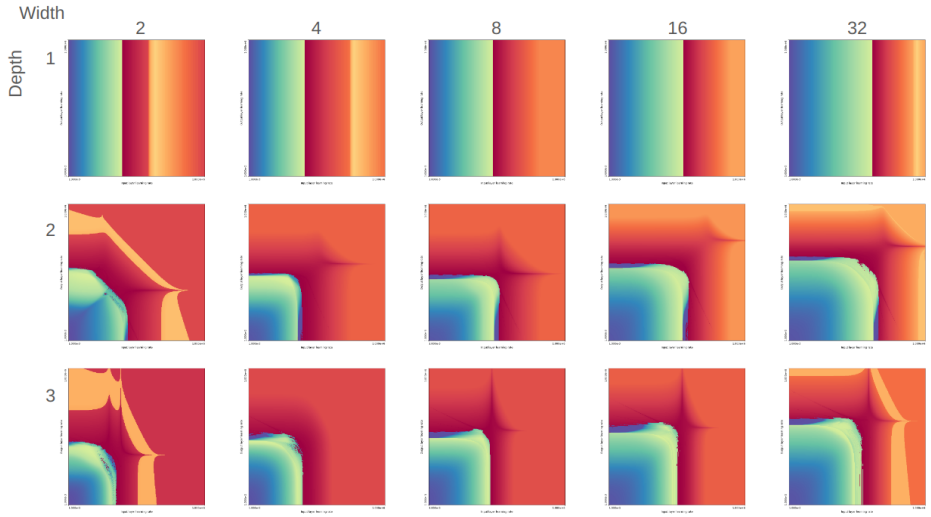

Figure 3: **Neural Network Trainability across Architectures:** This collection of heatmaps portrays the effects of varying network width and depth on trainability. Each plot corresponds to a distinct combination of width (horizontal axis) and depth (vertical axis), with color gradients indicating the transition from convergence (blue-green) to divergence (red-yellow) based on the learning rates of the input and output layers.

All code published on Github[1]

[1] https://github.com/OmkarV23/fractal_experiments

4.2.4 Visualizations and Interpretations

The outcomes of these experiments are illustrated in a series of color-coded matrices 3. Each pixel within these matrices embodies the result of a training run defined by the respective learning rate pair, with successful training depicted in hues of blue-green and unsuccessful training in tones of red-yellow. Specifically, for diverging training runs, the color intensity is proportional to $\sum_t \ell_t^{-1}$. For converging runs, color intensity is proportional to $\sum_t \ell_t(\cdot)$, where ℓ_t is the loss at training step t. The more intense the blue color, the longer the training run spent with higher loss[1]. The gradients of color transitions, demarcating the boundaries between convergence and divergence, offer a visual representation of the fractal-like patterns within the hyperparameter landscape.

From this visual data, several insights can be derived:

- **Influence of Width:** As the width of the network increases, we generally observe an expansion in the region of convergent learning rates. Wider networks have more capacity, which can potentially capture more complex patterns. However, they also pose a higher risk of overfitting and can be more sensitive to the choice of learning rate.
- **Influence of Depth:** Increasing the depth of the network, which corresponds to adding more layers, does not show a uniform effect on trainability across the varying learning rates. While deeper networks are capable of learning more abstract representations, they also complicate the training process, potentially leading to issues such as vanishing or exploding gradients, which can be observed in the varied convergence regions.
- **Complexity and Trainability:** The fractal-like patterns emerging in the heatmaps, particularly at higher depths and widths, hint at the intricate relationship between network architecture and trainability. The boundary between convergence and divergence becomes less distinct and more complex, suggesting a fractal dimension to the trainability landscape.
- **Robustness to Learning Rates:** The plots indicate that simpler networks (lower width and depth) have a more sharply defined boundary between convergent and divergent learning rates, while more complex networks exhibit a broader spectrum of successful training rates. This might imply that larger networks have a certain level of robustness to the choice of learning rate, possibly due to their greater representational capacity.

4.3 Analyzing Parameter Perturbations and Their Influence on the Fractal Nature of Trainability Boundaries

Neural network training landscapes are notoriously complex, with trainability boundaries that can exhibit fractal-like properties. To further understand these properties, we investigate the influence of stochastic weight perturbations on these boundaries. This section explores how different magnitudes of perturbation, controlled by a factor ε, affect the fractal nature of the trainability boundaries. Weight perturbations are introduced using the following function, where θ represents the network weights, *rng* is a random number generator state, *depth* is the number of layers in the network, and ε is the perturbation factor (10^{-5} in our case).

$$perturb\_weights(\theta, rng, depth, \epsilon) = \{\theta_i + \epsilon \cdot N(0, I) \mid \theta_i \in \theta, \epsilon \in R\} \tag{4}$$

Here, $\mathcal{N}(0, 1)$ denotes a sample from a normal distribution with mean 0 and the identity covariance matrix. The function modifies each weight by adding a normally distributed random value scaled by ε, resulting in a perturbed set of weights θ_i for the i-th layer.

4.4 Observations

The systematic perturbation of neural network weights across varying levels of ε (epsilon) as shown in Fig. 4 provides critical insights into the robustness and sensitivity of the training process under stochastic influences. This experimental approach, visualized through a series of detailed heatmaps, elucidates several key aspects of neural network behavior in response to parameter perturbations.

- **Sensitivity to Initial Conditions:** The experiments highlight a pronounced sensitivity to initial conditions within neural networks, evidenced by the shifts in the trainability landscape as ε increases. For ε = 0, where no perturbation is applied, the heatmap exhibits well-defined regions of convergence and divergence. As ε is incrementally increased, these boundaries become increasingly blurred and complex, suggesting a chaotic nature. This observation is also supported by an increasingly negative maximum Lyapunov Characteristic Exponent (mLCE).
- **Emergence of Fractal Boundaries:** With increasing values of ε, the boundaries between convergent and divergent regions exhibit more pronounced fractal characteristics. At higher perturbation levels (e.g., ε = 1e-2 and ε = 1e-1), the complexity of these boundaries increases significantly. The fractal dimension of these boundaries has also been quantified to provide a more rigorous characterization of this behavior.
- **Robustness of Network Trainability:** Networks that maintain broader areas of convergence despite higher ε values may be inherently more robust against the noise and instability associated with high-dimensional parameter spaces. Conversely, networks that show significant disruption in trainability with minor weight perturbations might require more careful tuning and regularization to achieve optimal performance.

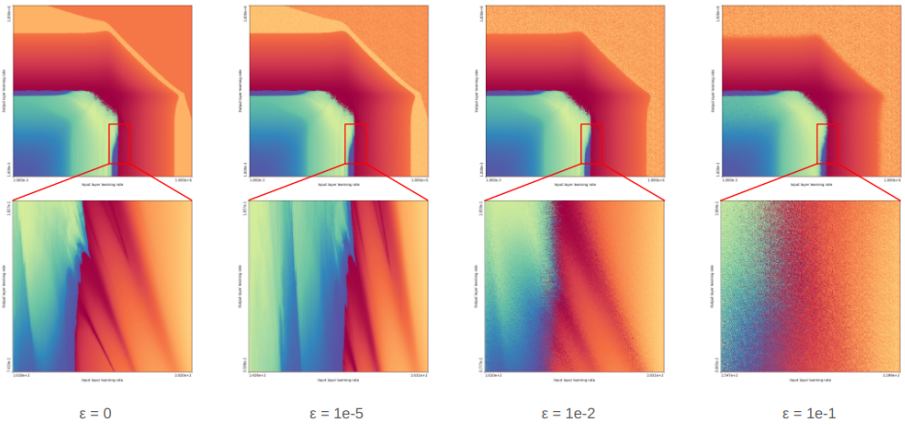

Figure 4: **Influence of Weight Perturbations on Neural Network Trainability:** The series of heatmaps demonstrates the impact of increasing weight perturbation levels, ε, on the fractal nature of trainability boundaries. From left to right, the images represent perturbation levels from $\varepsilon = 0$ (no perturbation) to $\varepsilon = 1e - 1$, illustrating the transition from clear to increasingly complex and fractal boundary regions between convergent and divergent training outcomes.

- **Implications for Training and Architecture Design:** Understanding the sensitivity of networks to weight perturbations can guide the development of more robust training algorithms that are less susceptible to the initial weight settings and can handle noise better. Additionally, this understanding can assist in designing network architectures that are inherently more stable and capable of generalizing from training data to unseen data, a critical aspect of deploying neural networks in real-world applications.

5 Conclusion

In this report, we have embarked on an in-depth exploration of the fractal boundaries that demarcate the trainable regions within the parameter space of neural networks. Through a series of carefully designed experiments, we have investigated the effects of architectural variations and weight perturbations on the behavior of these trainability boundaries.

Our findings reveal that the transition between convergence and divergence in the training process is not a simple binary demarcation but a chaotic dynamic system characterized by complex, fractal-like patterns. These patterns become more pronounced as the network architecture grows in depth and width, suggesting that higher-dimensional parameter spaces exhibit a rich tapestry of trainability that is sensitive to both the initial conditions and intrinsic network properties.

The experiments with weight perturbations have further demonstrated the delicate nature of this dynamic system. Small perturbations can lead to significant changes in the trainability landscape, pointing to a high degree of sensitivity to initial weight configurations. This underscores the importance of precise weight initialization and careful hyperparameter tuning in the training of neural networks.

Moreover, the proximity of optimal performance regions to the boundaries of divergence illustrates the narrow margin within which neural networks operate most effectively. It brings to light the need for robust training methodologies that can navigate these intricate boundaries to harness the full potential of neural network learning capabilities.

As we conclude, it is evident that the dynamic nature of hyperparameter spaces and the fractal nature of trainability boundaries present both challenges and opportunities. While the complexity of these boundaries poses significant challenges in training stability and generalization, it also opens up new avenues for research in developing more sophisticated and resilient learning algorithms. Future work in this domain may focus on quantifying the fractal dimensions of these boundaries as a measure of model performance[10], developing adaptive learning rate strategies, and exploring the implications of these findings for other forms of neural network architectures.

This report contributes to a deeper understanding of the dynamics of neural network training and offers a foundation for future studies aimed at enhancing the reliability and efficiency of machine learning systems. As the field continues to advance, the insights gleaned from the fractal characteristics of trainability boundaries will undoubtedly play a crucial role in shaping the next generation of artificial intelligence applications.

References

[1] Jascha Narain Sohl-Dickstein. The boundary of neural network trainability is fractal. *ArXiv*, abs/2402.06184, 2024.

[2] Alexander Camuto, George Deligiannidis, Murat A Erdogdu, Mert Gurbuzbalaban, Umut Simsekli, and Lingjiong Zhu. Fractal structure and generalization properties of stochastic optimization algorithms. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18774–18788. Curran Associates, Inc., 2021.

[3] Benoit Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Co., 1989.

[4] Kenneth Falconer. *Fractal Geometry*. Wiley, 1990.

[5] Luis Herrmann, Maximilian Granz, and Tim Landgraf. Chaotic dynamics are intrinsic to neural network training with SGD. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[6] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof J. Geras. The break-even point on optimization trajectories of deep neural networks. *ArXiv*, abs/2002.09572, 2020.

[7] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Narain Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *ArXiv*, abs/2003.02218, 2020.

[8] Gonzalo Uribarri and Gabriel B. Mindlin. Dynamical time series embeddings in recurrent neural networks. *Chaos, Solitons Fractals*, 154:111612, 2022.

[9] Xuxing Chen, Krishnakumar Balasubramanian, Promit Ghosal, and Bhavya Agrawalla. From stability to chaos: Analyzing gradient descent dynamics in quadratic regression. *ArXiv*, abs/2310.01687, 2023.

[10] Valeri Alexiev. Fractal dimension generalization measure. *CoRR*, abs/2012.12384, 2020.