COMPRESSIBILITY AND MULTIFRACTAL PROPERTIES OF SELF-COMPRESSING NEURAL NETWORKS

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Mathematics

with a Concentration in

Dynamical Systems

---

by

Keir Havel

Summer 2026

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Keir Havel:

Compressibility and Multifractal Properties of Self-Compressing Neural Networks

---

Uduak George, Chair
Department of Mathematics and Statistics

Ricardo Carretero
Department of Mathematics and Statistics

Bryan Donyanavard
Department of Computer Science

Approval Date

We sit together, the mountain and I, until only the mountain remains.

– Li Po

ABSTRACT OF THE THESIS

Compressibility and Multifractal Properties of Self-Compressing Neural Networks

by
Keir Havel
Master of Science in Applied Mathematics with a Concentration in Dynamical Systems
San Diego State University, 2026

Self-models are representations that a system forms of its own internal states. They are central to theories in cognitive science and are increasingly being explored in the context of artificial intelligence. However, the term has been used inconsistently and it remains unclear how self-modeling can be implemented in neural networks in a way that provides benefits or what effect it will have on neural network representations. Here we recast self-modeling in terms of compression. Our method relies on autoencoders, classic neural network architectures used for compression. We equip general neural networks with *internal autoencoders* that compress one of its own states, and the network is then encouraged to make that layer more like its compressed reconstruction, so that it becomes more compressible. We apply this method to neural networks trained for image classification and find that it leads to faster training and higher final accuracy, while simultaneously leading to more compressible representations relative to a null model. Furthermore, we develop analytic results for a dynamical system formulation of simplified neural network weight dynamics which suggest that this method works by increasing neural networks' ability to learn low-frequency features in the data, a phenomenon known as "spectral bias" in existing neural network literature. We investigated the geometric properties of our trained neural network relative to a null model, further providing evidence for spectral bias. We establish that fixed-mass methods for multifractal spectrum estimation perform competitively with box-counting on iterated function systems with known spectra and scale to higher dimensions than is feasible with box-counting. Lastly, we apply this multifractal analysis to the trained neural network representations, find evidence supporting multifractal structure, and results that suggest it is determined more by the data than the training method. Altogether, these results show that our compression method is a useful inductive bias and demonstrates a tractable model of how a neural network might represent and reshape its own internal processes.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

Activation The numerical output of a neuron or layer in a neural network, usually after applying a nonlinear function. In this thesis, activations are often treated as internal states of the network.

Autoencoder A neural network trained to reproduce an input after passing it through a compression bottleneck.

Auxiliary loss An additional task or objective for a neural network beyond its main task.

Chhabra–Jensen method A method for directly estimating the multifractal spectrum via a regression method, avoiding the sometimes unstable Legendre transform.

CIFAR-10 A standard image-classification dataset consisting of small color images from ten object classes, such as animals and vehicles.

Epoch The general term for a single pass through a training dataset during neural network training.

Fixed-mass approach The name for methods for estimating multifractal properties by using k-nearest neighbors.

Hidden state The state variables internal to a neural network.

Homunculus fallacy The error of explaining a system by positing another internal observer or interpreter, which itself requires a similar of explanation. In self-modeling, this is a concern if anything like a self-model is treated as an internal agent rather than as a functional part of the system.

Internal compression The method presented in this work, in which a neural network is encouraged to make its internal states closer to a compressed reconstruction of themselves.

Iterated function system (IFS) A collection of maps that are repeatedly applied to generate a fractal set or measure.

Latent representation The internal representation of something within a neural network, especially an autoencoder.

Log-rank test A statistical test from survival analysis used to compare times until an event occurs.

Minimum description length (MDL) A principle favoring models that describe data with a shorter total description. It is a formalization of Occam's razor.

MNIST A standard benchmark dataset of handwritten digit images, commonly used for testing image-classification methods.

Monofractal A fractal or measure whose local scaling behavior can be characterized by essentially one exponent. This contrasts with a multifractal, where local scaling varies across the domain.

Multifractal A measure or system whose scaling behavior varies from point to point, so that no single fractal dimension captures all of its structure.

Multifractal spectrum The function $f(\alpha)$ assigning to each local scaling exponent α the Hausdorff dimension of the set of points with that exponent.

Participation ratio An dimensionality measure computed from the eigenvalues of a covariance matrix. It is large when variance is spread across many directions and small otherwise.

Regularization A training technique that biases a model toward simpler or more stable solutions, often to improve performance outside of its training data. Our method, internal compression, is a form of regularization.

Self-model / self-modeling A representation that a system forms of states internal to it. This has been used in the past for representations of robot bodies and of decision variables in large language models.

Sparse autoencoder An autoencoder constrained so that only a small number of latent units are active at once. Sparse autoencoders are often used in interpretability work because their latent features may be easier to analyze.

Spectral bias An empirically observed and theoretically justified tendency of neural networks to large-scale or "low-frequency" features before learning fine-grained features.

Stop-gradient An operation that treats a quantity as constant during gradient computation in machine learning, meaning its gradient is considered to be 0.

Validation accuracy / validation loss Performance measured on data not used for training, which is called the validation data.

Weight initialization A term used for both the process of generating neural network parameters before training begins or a name for the result of that process, i.e. the initial parameters.

ACKNOWLEDGMENTS

I would like to thank Professor Uduak George for all the mentoring, encouragement in pursuing unconventional interests, and supporting me as I jumped around various thesis topics. To Professor George, Professor Carretero, and Professor Donyanavard, thank you for taking the time to review my thesis and serve on my thesis committee. I would also like to thank my family and my dog (of course!) for their support in my life.

CHAPTER 1
INTRODUCTION

Self-models have been invoked across cognitive science [1], robotics [2–4], and artificial intelligence [5, 6], being used to explain how systems are able to regulate and control their own behavior. The definition has varied and ranges from a system having a predictive model of its body [2], a model of itself as coherent source of behavior [5], or as modeling its own internal states [6]. These self-models have been suggested to subserve abilities like planning [4], avoiding damage [2, 3], and emulating social roles [5], making further research critical to both making neural networks more general and safer. While there has been extensive work on how body self-models are useful in robotics, their potential role in artificial intelligence outside of robotics has remained less clear.

Here, we propose a novel technique based on the idea of a system using a self-model to make itself more interpretable or compressible. A key part of building a model is discarding extraneous features from observations and retaining the core explanatory features that capture the essence of the thing being modeled, in short: compression. Here we explore the notion of a self-model via the idea of a system that forms *a compressed representation of* its internal states. While these representations could subserve numerous functions [7–9], we focus on systems that use compressed representations of their internal states solely to further simplify or compress the internal states themselves. As an illustrative example, consider the process of explanation itself: when explaining an idea or concept, we often simplify or throw away unnecessary technicalities. While this can lead to loss of important information, it simplifies the idea and can help further develop the idea for productive uses and reinterpretation [10].

Compression is a strong candidate for self-models because it involves discarding unnecessary features and retaining only those with largest amount of explanatory power. This connects to the minimum description length (MDL)

principle [11], an information-theoretic model capturing the idea of Occam's razor: a short or compressed model that explains observed data should be preferred to one with a long description. When this principle is applied internally, it means that a system is actively sculpting its internal states to be more compressible.

In examining self-models based on compressing internal states, we primarily follow and build upon the notion of self-modeling in neural networks of Premakumar et al. [6]. Their work itself built upon previous neural network models [12–14] of attention schema theory [15]. In their model, a neural network is trained to produce an output that faithfully represents one of its internal states. They showed that this task leads to shifts in the weight distribution of the neural network, toward less complex network structure. While they investigated how self-modeling can be used to *externally* represent the internal states of a neural network, we were interested in how self-models can be used internally.

There are other precedents in the literature based on the using a neural network to form compressed representation of the features of another neural network. The idea of "meta-representations" of neural networks has been implemented [16] using autoencoders to embed the *weights* of neural networks trained on different tasks to compare them. Self-reference in artificial neural networks has also been studied in the case of meta-learning [17–19]. Furthermore, a framework where some layers or modules in neural networks predict the states of other layers has connections to theories of predictive coding in the brain [20] and artificial neural networks [21]. Our method is distinct in that internal states are not predicted in a vacuum, instead the predictions themselves go on to actively shape the internal states to be more predictable.

The MDL principle has previously been investigated in the context of deep learning [22], including techniques to regularize neural networks to be more compressible [23, 24] based on controlling the precision of their weight matrices. Our method instead uses autoencoders, which are neural networks trained to compress and decompress inputs. Use of autoencoders to compress the states of a neural network has also been used in the field of AI interpretability, where sparse autoencoders have been highly successful [25] in allowing researchers to interpret

the internal states of large language models [26]. Autoencoder-like structure based on Oja's plasticity rule [27] applied to each layer in neural networks has been used to facilitate stable learning while avoiding the use of normalization methods or hand-engineered neural network initialization strategies [28]. These results suggest that there could be gains beyond compressibility with our method.

Our method consists of additional regularization objective where an autoencoder compresses the internal states of a neural network through a bottleneck and then the neural network is trained to be more like the reconstructed output of the autoencoder. We compare to an equivalent null model that also has an autoencoder building a compressed model of its internal states, but without the training to make its internal states more compressible with respect to that autoencoder. The core training task of the neural networks is image classification. We show that the internal compression objective improves training speed and final classification accuracy, while simultaneously leading to more compressible internal states. We show evidence that this is consistent with a spectral-bias toward using fewer degrees of freedom to encode more variance in the data.

Given tantalizing connections discovered between algorithmic compression and fractal geometry [29, 30], we further investigate how this method affects the multifractal spectra of the states of the neural networks. However, standard multifractal analysis often relies on the box-counting method, which scales poorly to high-dimensional spaces like those involved with neural networks. Given this, it is imperative to find methods of multifractal analysis that scale to spaces with large dimension. To this end, we apply fixed-mass methods [31], which we compare against other methods.

We briefly summarize our core goals and aims:

- Evaluate multifractal estimation methods that function well in high-dimensional spaces
- Compare the outlined internal compression method against a null neural network model
- Develop analytical results with simplified dynamical systems versions of the neural networks to characterize their fixed points and expected behavior

- Evaluate factors characterizing the geometry, including explained variance and fractal geometry, of the neural network representations

In Chapter 2, we review relevant background on fractal geometry, computational aspects of fractal geometry, neural networks, and cognitive science. We compare various computational methods for estimating fractal properties in Chapter 3, with the main goal of determining how to estimate the fractal properties of high dimension neural network states. In Chapter 4 we formally introduce our method and present our results. Then, in Chapter 5 we integrate our results and conclude.

CHAPTER 2
BACKGROUND AND PREVIOUS WORK

Here, we review the background necessary for understanding our methods and results. We begin by reviewing essentials of neural networks and some common architectures. Then, we briefly review some of the more uncommon statistical methods and metrics we use. Lastly, we review the core details of fractal geometry, including multifractal spectra and efficient computational estimation techniques for related quantities.

2.1 Neural Networks, Autoencoders, and Models in Cognitive Science

A typical neural network can be formulated as follows [32]: given a vector of inputs $x_0 \in \mathbb{R}^n$, a L-layer neural network transforms h_0 to an output $\vec{y} := \vec{h}_{L+1} \in \mathbb{R}^m$ as:

$$\vec{h}_{l+1} = f(W_l \vec{h}_l + \vec{b}_l), \tag{2.1}$$

where $W_l \in \mathbb{R}^{d_l \times d_{l+1}}$ is a weight matrix, $\vec{b}_l \in \mathbb{R}^{d_{l+1}}$ is additive bias term, f is a potentially nonlinear function that is applied to each element of a vector called the activation function, and $\{d_l\}$ are called the hidden dimensions of the neural network (with d_0, d_{L+1} equal to n, m). We will typically denote the full set of parameters (i.e. weights W_l and biases $\vec{b}_l$) determining a given neural network with Greek variables like θ, ϕ as this is standard convention. Furthermore, we will follow the convention of the neural network literature of referring to the states h_l as activation states or activations. Neural networks can be used for function approximation by defining the loss or objective function $\mathcal{L}(\{w_L\})$ and updating the weight matrices by gradient descent to minimize the loss function [32].

Classic tasks like classifying images are readily achievable with neural networks. Given a dataset $\mathcal{D}$ of N images and labels $\{\vec{x}, y\}_{i=1}^N$, where the labels correspond to numbers indicating what the image is of (e.g. 1 = "dog", 2 = "cat"),

classification can be formulated as updating the neural network parameters to maximize the chance of classifying an image correctly. Usually, this is done by minimizing the cross entropy loss:

$$\min_{\theta} L_{ce}(\vec{y}, \hat{y}) = \min_{\theta} \ -\log \frac{e^{\hat{y}_n}}{\sum_{i=1}^{m} e^{\hat{y}_i}} \mathbb{I}(y = n), \tag{2.2}$$

where $\vec{y}$ is the vector output of a neural network, $\mathbb{I}(y = n)$ is the indicator function that is 1 when $y = n$ and 0 otherwise, the ith element y_i corresponds to the log-probability of class i, the function on the RHS side is the cross-entropy between the neural network output and the true label y, and m is the number of classes in the data.

Another classical task in deep learning is compressing inputs, such as images. An autoencoder [32] is a classic type of neural network that can be split into an encoder neural network $f(\vec{x}, \theta)$ which produces a compressed representation $\vec{z}$ of an input x and a decoder neural network $g(z, \phi)$ which takes $\vec{z}$ and reproduces an approximation of $\hat{\vec{x}}$. In the construction, $\dim(\vec{z}) < \dim(\vec{x})$, so that the system compresses the input.

Neural networks have a number of properties that make them useful in machine learning. Core among these is *universal approximation* [33, 34]: neural networks of sufficient complexity can approximate any continuous function. A further property of particular interest to us is that of spectral bias [35], which refers to the fact that neural networks are biased to first learning low-frequency features of data. This aspect of neural networks often leads to greater resistance to noise in data and to better generalization outside of their training data.

Neural networks have been used as model systems in cognitive science. Attention schema theory [15] is a theory in cognitive science about how the brain uses models of its attention to control and apply it in guiding behavior. There have been several models of this theory implemented with neural networks [12–14], including the main work we follow [6]. This work is interested more generally in "self-modeling", where a neural network has some internal model of its own activity that is used for some task. In particular, they train neural networks to classify images, while simultaneously being able to "report" on its internal states.

Specifically, in addition to the output for classification $\hat{y}$, the neural network has an additional output $\hat{\vec{h}}_k$ which is an estimate of one of the neural network's internal state, $\vec{h}_k$ for some k. They optimize the parameters of the neural network to minimize the classification loss 2.2 and a new mean-square error of the estimate of its internal state:

$$\min_{\theta} L_s(\vec{h}_k, \hat{\vec{h}}_k) = \min_{\theta} \ ||\vec{h}_k - \hat{\vec{h}}_k||^2, \tag{2.3}$$

In this way, they train the neural network to report on its internal states.

2.2 Statistical Methods

The previous research we build on [6] found differences in the distribution of neural network weights when their "self-modeling" task was added. We opted to measure difference in the distribution of the weights using the Kolmogorov-Smirnov (KS) statistic between the sets of the weights of the null and internal compression model. Formally, the two-sample KS statistic is:

$$D_{n,m} = \sup_x |F_n^{(ic)}(x) - F_m^{(null)}(x)|, \tag{2.4}$$

where $F_n^{(null)}, F_m^{(ic)}$ are empirical cumulative distribution functions from samples of weights with sizes n, m for the null and internal compression methods, respectively. We track this statistic over training as a proxy for differences in weight distribution in the model over time and also report the nominal $\alpha = 0.05$ significance threshold for the pointwise KS statistic for the given samples, however, we emphasize that for whole trajectories there are issues with determining when differences "become significant" due to issues with multiple comparisons and autocorrelation between the KS statistics at nearby points in training. Nevertheless, the KS statistic itself provides a measure of the similarity of the distributions, which can be tracked during training.

We further used participation ratio [36] to measure differences in the structure of learning with internal compression. Given a set of data points

$\{\vec{x}_i\}, i \in 1, ..N$, we first compute the covariance matrix:

$$\Sigma = \frac{1}{N-1}\sum_{i=1}^{N}(\vec{x}_i - \vec{\bar{x}})(\vec{x}_i - \vec{\bar{x}})^T, \tag{2.5}$$

where $\vec{\bar{x}} = \frac{1}{N}\sum_{i=1}^{N}\vec{x}_i$ is the mean. If $\vec{x}_i \in \mathbb{R}^d$, then Σ is a $d \times d$ matrix. By construction, it is positive semi-definite and therefore has real, non-negative eigenvalues $\{\lambda_i\}, i \in 1, ..d$. The participation ratio is defined as:

$$\text{PR} = \frac{(\sum_i \lambda_i)^2}{\sum_i \lambda_i^2}, \tag{2.6}$$

Participation ratio captures the spread or dimensionality of the covariance matrix, it is maximized with value $\text{PR} = d$ when all eigenvalues have the same value λ and minimized with value 1 when all but one of the eigenvalues are 0 (note, for the covariance matrix $\lambda_i \geq 0$). The former case corresponds to all dimensions being equally necessary to explain variance in the data, the latter corresponds to a single axis explaining all variance in the data. This makes participation ratio a useful tool for evaluating the diversity of a set of data points.

Computing the eigenvalues of the covariance matrix further enabled us to calculate the percentage of variance explained by the model's learned representations, as is done in principle component analysis [37]. Specifically, we computed the cumulative proportion of the variance c_i in the data explained by the neural network representations as:

$$c_i = \frac{\sum_{j \leq i} \lambda_j}{\sum_i \lambda_i}, \tag{2.7}$$

where λ_i are the eigenvalues of the covariance matrix, in descending order.

2.3 Fractal Geometry

The word "fractal" was coined by Benoit Mandelbrot from the Latin word "fractus" for "broken" and was intended to demarcate mathematical objects beyond the smooth, regular ones encountered in elementary geometry [38]. While the term is often used imprecisely by both the public and mathematicians, core

defining characteristics of fractals are self-similarity and scaling characteristics (like dimension). In 1918, Felix Hausdorff provided some of the early formalization of fractal sets [39] as they occur in measure theory, which yielded the concept of fractal dimension:

Definition 2.1 (Hausdorff dimension)**.** *For a set $F \subset \mathbb{R}^n$, the s-dimensional Hausdorff measure is*

$$\mathcal{H}^s(F) = \lim_{\delta \to 0} \inf \left\{ \sum_i |U_i|^s : F \subset \bigcup_i U_i,\ |U_i| < \delta \right\} \tag{2.8}$$

The Hausdorff dimension is

$$\dim_H(F) = \inf\{s \geq 0 : \mathcal{H}^s(F) = 0\} = \sup\{s \geq 0 : \mathcal{H}^s(F) = \infty\} \tag{2.9}$$

Here $|U_i|$ is the diameter of the set U_i. Hausdorff dimension is often impractical, however, it is necessary for the definition of the multifractal spectrum (see Subsection 2.3.1).

Further development of the idea of fractal dimension occurred in 1959 with Kolmogorov and Tikhomirov's work on ε-entropy and ε-capacity [40], where they introduced a measure that would later be known as the box-counting or Minkowski–Bouligand dimension, defined as follows:

Definition 2.2. *Given a bounded set S in metric space (X, d), the box-counting dimensions is given by:*

$$dim_{box}(S) := \lim_{\epsilon \to 0} \frac{-\log N(\epsilon)}{\log(\epsilon)}, \tag{2.10}$$

where $N(\varepsilon)$ is the number of boxes of side length ε that cover S.

The limit in the previous definition may not exist, in which case we can say the set S has an upper and lower box counting dimension. Note that the previous definition implies power laws with a box counting dimension d:

$$-d = \frac{\log N(\epsilon)}{\log \varepsilon} \iff N(\varepsilon) \propto \epsilon^{-d}, \tag{2.11}$$

These scaling relations are critical to the practical computation of fractal dimensions, where the estimation can practically be converted into a regression problem.

Lastly, we define the generalized fractal dimensions [41] D_q:

Definition 2.3. *For a measure μ on $\mathbb{R}^n$, the generalized fractal dimension is defined by:*

$$D_q := \frac{1}{q-1} \lim_{\epsilon \to 0} \frac{-\log \int d\mu \ \mu(B_\epsilon(x))^{q-1}}{\log(\epsilon)}, \tag{2.12}$$

where $B_\varepsilon(x)$ is a ball of radius ε centered at x.

Of particular importance for us are D_0 (which is the box-counting dimension) and $D_1 = \lim_{q \to 1} D_q$ (also called the information dimension).

2.3.1 Multifractals

Fractal methods eventually became applied in the study of turbulence, where Kolmogorov showed [42] that turbulent systems exhibited power laws in how they dissipate energy. The word "multifractal" first appeared in [43, 44] and was used to describe the fact that the Navier-Stokes systems under study did not have scaling relations of the form of Equation 2.11 and instead the dimension d depended on points in the domain nontrivially. The idea of the multifractal spectrum was fully formalized in [45], which introduced the following definition:

Definition 2.4 (Hölder exponent / local singularity exponent)**.** *For a measure μ on $\mathbb{R}^n$, the Hölder exponent at a point x is*

$$\alpha(x) = \lim_{\varepsilon \to 0} \frac{\log \mu(B_\varepsilon(x))}{\log \varepsilon}, \tag{2.13}$$

where $B_\varepsilon(x)$ is a ball of radius ε centered at x. Equivalently, $\mu(B_\varepsilon(x)) \propto \varepsilon^{\alpha(x)}$ as $\varepsilon \to 0$.

The paper [45] also introduced the general method for estimating multifractal properties of dynamical systems and their attractors. This method depends on the following quantities:

Definition 2.5 (Partition function)**.** *Cover the support of measure μ with boxes $\{B_i\}$ of size ε. The partition function is*

$$Z(q, \varepsilon) = \sum_i \mu(B_i)^q, \tag{2.14}$$

For small ε, this scales as

$$Z(q, \varepsilon) \propto \varepsilon^{\tau(q)}, \tag{2.15}$$

where $\tau(q)$ is called ***mass exponent****.*

This has connection with thermodynamics [46] and the name "partition function" comes from that fact. A core relationship between the mass exponent and generalized dimension D_q [38] is

$$\tau(q) = (q-1)D_q, \tag{2.16}$$

which enables using the mass exponent to read off quantities like the box-counting dimension D_0.

Definition 2.6 (Multifractal spectrum)**.** *Let S_α denote the set of points where the Hölder exponent equals α:*

$$S_\alpha = \{x : \alpha(x) = \alpha\}, \tag{2.17}$$

The multifractal spectrum is

$$f(\alpha) = \dim_H(S_\alpha), \tag{2.18}$$

the Hausdorff dimension of the set S_α. This gives the fractal dimension of the subset of points with singularity strength α and a measure is said to be monofractal if $f(\alpha)$ is a delta function.

The same paper [45] showed that the mass exponent and multifractal spectrum are related by the Legendre transform:

$$f(\alpha) = \inf_q [q\alpha - \tau(q)] = q\alpha - \tau(q), \tag{2.19}$$

where the infimum is achieved at q satisfying $\alpha = \alpha(q) = \frac{d\tau}{dq}$. Therefore, in practice the mutlifractal spectrum was estimated by first estimating the mass exponent $\tau(q)$ using fixed values of q, then performing a Legendre transform to estimate α and $f(\alpha)$.

2.3.2 Chhabra-Jensen Method

The Legendre transform can often be sensitive to numerical issues due to the approximate estimation of the mass exponent $\tau(q)$. The Chhabra-Jensen Method [47] allows directly estimating α and $f(\alpha)$ without using $\tau(q)$. The method

involves normalizing the measure (for a given q):

$$\tilde{\mu}_q(B_i) = \frac{\mu(B_i)^q}{\sum_j \mu(B_j)^q}, \tag{2.20}$$

where B_i is the box or region given a partitioning method. Then computing:

$$\alpha(q) = \lim_{\varepsilon \to 0} \frac{\sum_i \tilde{\mu}_q(B_i) \log \mu(B_i)}{\log \varepsilon}, \tag{2.21}$$

$$f(\alpha(q)) = \lim_{\varepsilon \to 0} \frac{\sum_i \tilde{\mu}_q(B_i) \log \tilde{\mu}_q(B_i)}{\log \varepsilon}, \tag{2.22}$$

where ε is the box size (note the $\tilde{\mu}_q$ versus μ_q). All of these values are readily computable in practice by using box-counting and identifying $N(\varepsilon) \sim \mu(B_i)$. Then $\alpha, f(\alpha)$ can be estimated with a regression procedure from box-counts. This method avoids numerical differentiation of $\tau(q)$, yielding better estimates in practice.

2.3.3 The Fixed Mass Approach

Fixed mass approaches [31, 48] to multifractal spectrum estimation use scaling properties of the number of nearest neighbors of a set of sample points. In this way, they avoid the requirement of grids, assumptions about the geometry of the fractal measure, and are straightforward to implement using standard functions from numerics libraries. We follow the method of [31], which derives a result that for a sample of n points from a measure:

$$\delta_x(k) \propto \left(\frac{k}{n}\right)^{1/\alpha(x)}, \tag{2.23}$$

where $\delta_x(k)$ is the distance from point x to its kth nearest neighbor and $\alpha(x)$ is the local Hölder exponent. Furthermore, they derive a result that, given the probability distribution $P(\delta, k)$ of kth neighbor distances, the multifractal spectrum is approximately given by:

$$f(\alpha) \propto \alpha \left(1 - \frac{\ln P(\delta, k)}{\ln(k/n)}\right) - 1, \tag{2.24}$$

While the relation only determines $f(\alpha)$ up to a scaling factor, it can be combined with the relation $f(\alpha) = \alpha$ at $\alpha = D_1$, the information dimension. The authors

show that another scaling relation:

$$\langle \delta_x(k) \rangle \propto \left(\frac{k}{n} \right)^{1/D_1}, \tag{2.25}$$

where $\langle \delta_x(k) \rangle$ is expected value of the nearest neighbor distance.

Practically speaking, $P(\delta, k)$ can be estimated using a histogram and $\langle \delta_x(k) \rangle$ can be evaluated using a sample mean. Overall, the practical estimation method starts with sampling points from the multifractal of interest and computing the kth-nearest neighbor distances for some value of k. First, this enables using equation 2.23 to regress estimates for α from k and $\delta_x(k)$. Then, a histogram of the kth-nearest neighbor distances $\hat{P}(\delta, k)$ is generated, which serves as a proxy for the true probability distribution $P(\delta, k)$. Then, equation 2.24 allows combining the estimate of α with the estimate of the distribution to generate an array with values proportional to $f(\alpha)$. Since D_1 can be estimated via a similar regression procedure, its value can then be used to set the intercept of the resulting curve, yielding $f(\alpha)$.

In addition to the practical benefits noted above, several additional properties make this method useful. First, like the Chhabra-Jensen method, its caclulation does not rely on computing a Legendre transform. Second, errors tend to *decrease* in higher dimensional space, due to how nearest neighbor relations scale with dimension. Third, by varying k, it is straightforward to verify that the method is converging given a fixed sample of n points.

2.4 Iterated Function Systems

Iterated function systems can be used to generate fractals with analytically known fractal properties. Formally, iterated function systems can be defined as [38]:

Definition 2.7 (Iterated function system)**.** *Let (X, d) be a complete metric space. A set of functions on X $\{f_i\}_{i=1}^m$ is an iterated function system if each $f_i : X \to X$ is a contraction map on X i.e.*

$$d(f_i(x_1), f_i(x_2)) \leq k d(x_1, x_2), \tag{2.26}$$

where $k \in (0, 1)$ is a constant.

A particularly simple IFS system with analytic fractal properties [49] has functions $f_i : \mathbb{R}^2 \to \mathbb{R}^2$ given by:

$$f_i(\vec{x}) = r_i O_i \vec{x} + \vec{t_i}, \tag{2.27}$$

where $r_i \in (0, 1)$ is a contraction ratio, O_i is an orthogonal/rotation matrix, and $\vec{t_i} \in \mathbb{R}^2$ is a translation vector. Supposing that each map f_i is applied with probability p_i, then the mass exponent $\tau(q)$ is given by the implicit equation [49]:

$$\sum p_i^q r_i^{-\tau(q)} = 1, \tag{2.28}$$

Practically, for given values of $r_i, O_i, \vec{t_i}, p_i$ the function $\tau(q)$ can be solved for using any nonlinear root finding method. We then use numerical differentiation and the Legendre transform to estimate the multifractal spectrum $f(\alpha)$. Further note that since $\tau(0) = -D_0$ of the set F, the box-counting dimension of an IFS is given by:

$$\sum r_i^{D_0} = 1, \tag{2.29}$$

An example of one of these IFS is depicted in Figure 2.1.

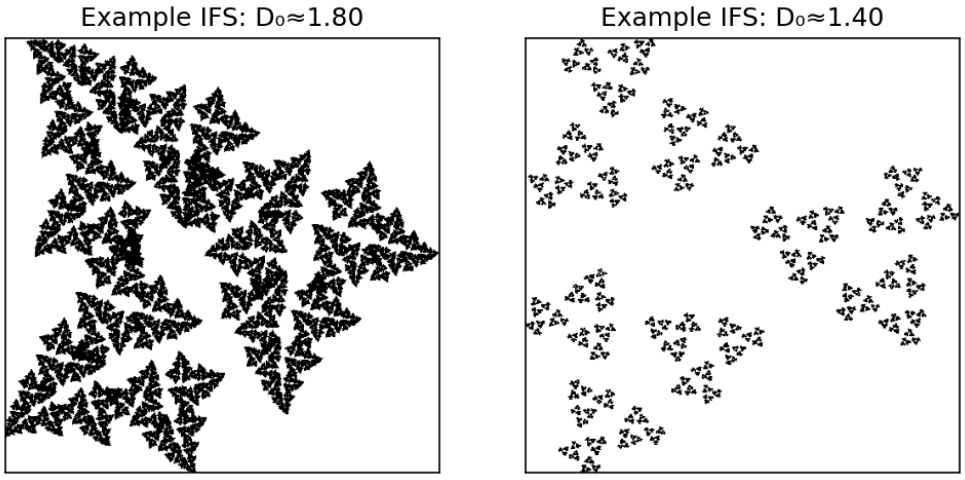

Figure 2.1. Two images generated by random iterated function systems of the form given by Equation 2.4. The approximate analytic monofractal dimension derived from Equation 2.28 is indicated above the image.

CHAPTER 3
COMPARISON OF METHODS FOR MULTIFRACTAL SPECTRUM ESTIMATION

The box counting method for estimation of the multifractal spectrum $f(\alpha)$ is standard and is made more numerically stable by the Chhabra-Jensen (CJ) method. Nevertheless, it scales poorly as dimension increases. With boxes of width ϵ, covering a single axis will generally require $N \propto O(\frac{1}{\epsilon})$ line segments. When $\epsilon \ll 1$, it follows that the number of boxes is large ($N \gg 1$). Covering all d axes will then require N^d boxes, which grows exponentially in d. In contrast, the fixed-mass method (hereafter referred to as BB method after the authors Badii and Broggi) only grows linearly in complexity as dimension increase: computing a pairwise distance requires $O(d)$ additions and multiplications.

Given that neural networks have high dimensional internal states, often with hundreds or thousands of hidden units which each add a dimension, we know from first principles that box counting will be infeasible in neural network state spaces. It is nevertheless useful to compare the CJ and BB method on problems with analytic results, in order to evaluate whether the BB method offers competitive performance relative to the more classic box-counting approach. To do this, we leverage the IFS method to randomly generate systems with varying but analytically known multifractal spectra.

We used a consistent procedure for generating random IFS systems. In general, we choose the number of maps f_i to be 3. We sample the rotation matrices O_i by uniformly sampling an angle from $[0, 2\pi]$ and similarly sample the translation vector $\vec{t_i}$ by fixing the length to 0.5 and sampling an angle uniformly. The contraction ratios r_i are sampled uniformly from $[0, 1]$ and the map probabilities p_i are sampled from a Dirchlet distribution.

Figure 3.1 shows qualitative examples of the true multifractal spectrum versus the estimate by the CJ and BB methods. Each panel is the multifractal spectrum for a randomly generated IFS system with roughly the given monofractal dimension (D_0). We generated systems with given D_0 by using Equation 2.29, which allowed us to solve for contraction ratios r_i that yielded a given D_0. Otherwise, parameters were generated randomly as described above. Both the CJ and BB methods use 10,000 sample points. While tentative, the qualitative comparison does suggest that the BB method is competitive with the CJ method for approximating the multifractal spectrum and that the BB spectrum has higher consistency.

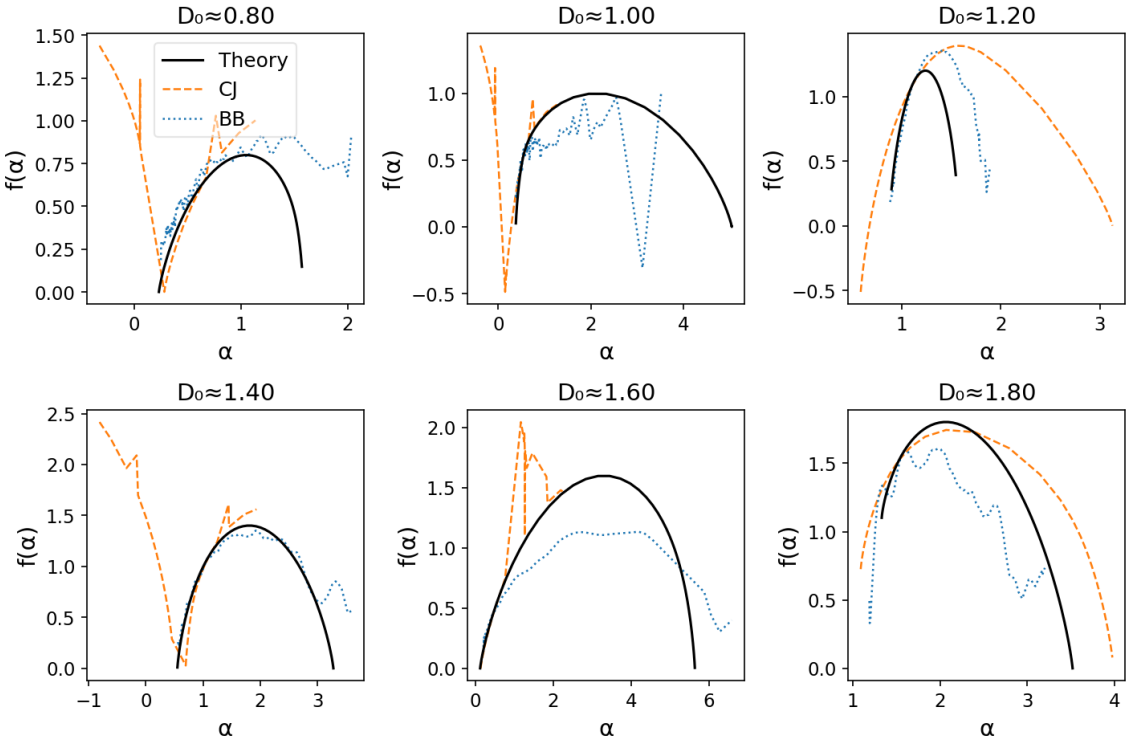

Figure 3.1. Several examples of the theoretical analytic multifractal spectra versus the Chhabra-Jensen (CJ) method and the fixed-mass method of Badii and Broggi (BB). The monofractal dimension is indicated at the top of each figure.

In order to evaluate the methods more quantitatively, we sampled 100 different IFS systems with varying dimensions and parameters, following the procedure described above. We generated varying numbers of sample points and evaluated the mean square error between the true analytical curve and the estimated curve. Since in general, the supports of the function may be different, we only evaluated the error on the shared support and chose to separately evaluate the fraction of the support shared between the methods and the ground truth. These results are depicted in Figure 3.2. Overall, both methods had similar MSE in reconstructing the analytical spectrum with modest variance across different systems. Error decreased with increasing sample size for both methods, as did estimation of the true support. The BB method estimated the support α more reliably, with nearly 100% agreement between the recovered support and the true support.

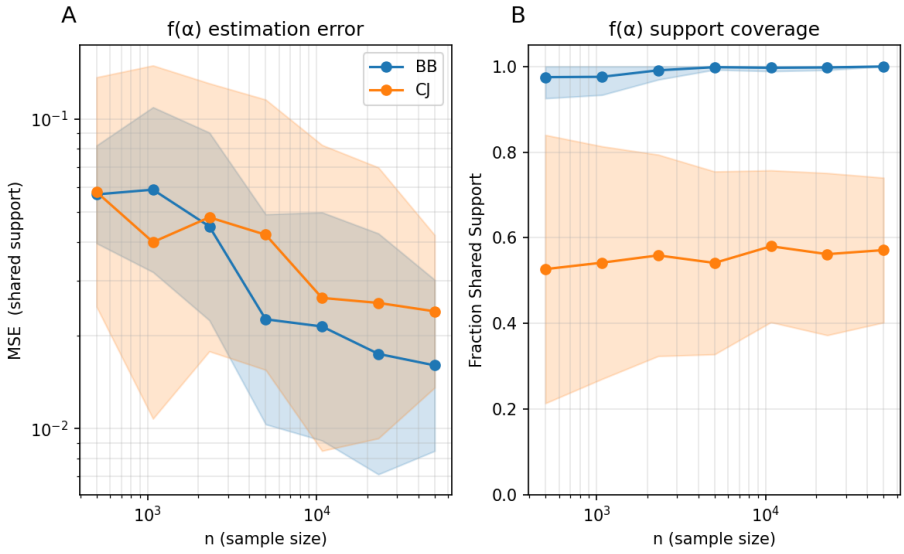

Figure 3.2. A) The mean square error (MSE) *on the support of the theoretical spectrum* as a function of number of points sampled for each method. B) The fraction of the shared support between the method's estimated support and the theoretical support as a function of number of sample points. The BB method had better estimation of the support (α). Shading indicates standard deviation across 10 seperate trials.

These results suggest that not only is the BB method more suited to higher-dimensional spaces, it is competitive and even has some advantages to the CJ method in relatively low-dimensional spaces. Therefore, the BB method is the natural choice for estimation of the multifractal spectra of neural networks.

CHAPTER 4
INTERNAL COMPRESSION OF NEURAL NETWORKS

In this chapter we introduce the formal details of proposed method for neural networks that learn to make their internal states more compressible. We further derive results from a dynamical model based on a barebones formulation of the method. We present our computational results and show that the method leads to improvements in training speed, final performance, and compressibility. Lastly, we investigate the multifractal spectra associated with our neural network models.

4.1 Formulation

As mentioned in the introduction (Chapter 1), our method is inspired by previous work [6], though follows a different realization of "self-modeling". Instead of adding an auxiliary output to the neural network that predicts an internal state, we train a neural network to predict one of its internal state and adapt itself using that prediction as a target, as depicted in Figure 4.1. This makes the internal state more like the neural network's prediction of it, which is necessarily compressed. Formally, let a neural network be defined by a sequence of layers and hidden states $\vec{h}_l$ as:

$$\vec{h}_l = f(\vec{h}_{l-1}; \theta_l), \tag{4.1}$$

where $\vec{h}_l \in \mathbb{R}^{m_l}$ is a hidden state for a layer with m_l neurons, θ_l are weights/parameters, and application of f can include nonlinear functions. We denote the full set of parameters of the neural networks as θ. The neural network has a core task (here, classification) given by a loss function $\mathcal{L}_{task}$, which is used to update the weights via gradient descent. We introduce the internal compression objective by using a separate neural network to estimate a given internal state:

$$\hat{\vec{h}}_k = f(\vec{h}_j; \phi), \tag{4.2}$$

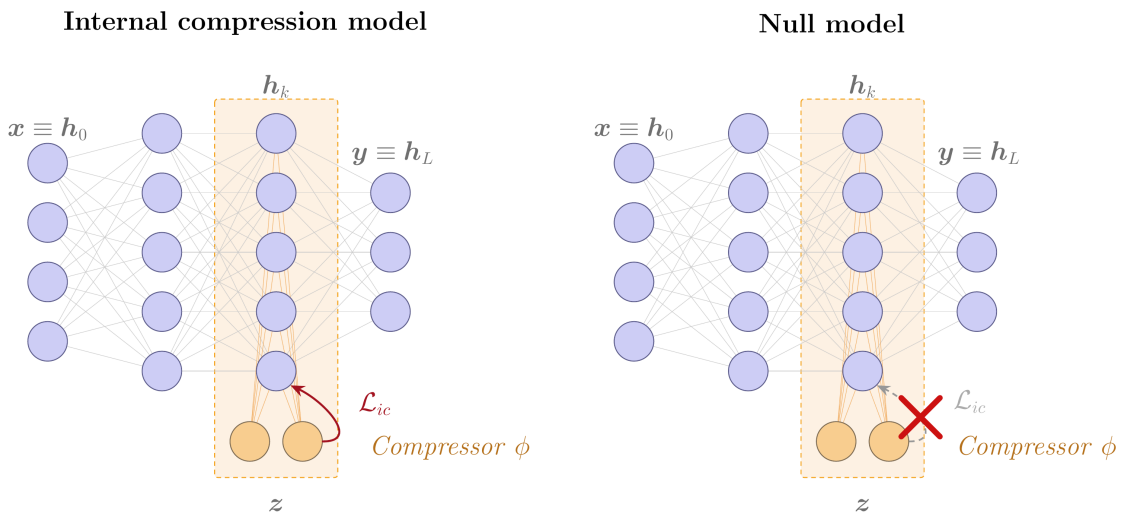

Figure 4.1. Our general method. A layer from a neural network (purple circles and edges), typically the penultimate one, is compressed with an autoencoder into a latent representation z (orange circles). In the internal compression case, the original network is trained to bring its state $\vec{h}_k$ closer to the decoded approximation $\hat{\vec{h}}_k$ via a loss term $\mathcal{L}_{ic} = ||\text{stopgrad}(\hat{\vec{h}}_k) - \vec{h}_k||^2$ (red arrow), making $\vec{h}_k$ more compressible. In the null case, the state $\vec{h}_k$ is still compressed, but there is no influence from the compressed state on the original neural network.

where ϕ are parameters of the internal compressor and k, j (where $k \leq j$) are indices of the layers used to predict one another. When $k = j$ we further impose an autoencoder or bottleneck condition where the prediction of $\hat{\vec{h}}_k$ must go through a bottleneck $\vec{z}$ with $\dim(\vec{z}) < \dim(\vec{h}_k)$; this ensures that internal model is compressing the latent states. We use the sum-of-squared-errors (SSE) loss to train the compressor:

$$\min_{\phi} \mathcal{L}_{\phi} = \min_{\phi} ||\hat{\vec{h}}_k - \vec{h}_k||^2, \tag{4.3}$$

Since the previous objective only affects the compressor network, we add an additional SSE loss term to the main network to encourage the network states to be more compressible:

$$\min_{\theta} \mathcal{L}_{ic} = \min_{\theta} ||\text{stopgrad}(\hat{\vec{h}}_k) - \vec{h}_k||^2, \tag{4.4}$$

where stopgrad indicates that we treat the argument as a constant, as is common in self-supervised learning where this is essential to prevent representation collapse [50, 51], convergence toward trivial solutions. Intuitively, this loss function encourages the states $\vec{h}_k$ to be more like the compressed version of itself, i.e. more compressible. We test both multilayer perceptrons and convolutional neural networks, using the GELU activation function [52]. For classification tasks, we use the cross-entropy loss function.

As our null case, we train a model which has the compression neural network, but drops the loss term $\mathcal{L}_{ic}$ giving a feedback effect. This leads to a case where the network being compressed is not influenced by the compression task, while still allowing us to track how "compressible" the network's activity is. This null model allows precisely disentangling the effects of initialization and parameter count from the core feature we want to investigate: how using feedback from an internal compressor can support making the network's states more compressible.

4.2 Linear Analysis

In order to better understand the effects of internal compression on steady state weights of the neural network. The following derivation largely follows [53], with the addition of the terms describing internal compression via an autoencoder. This additional term primarily affects the dynamics of the input weights of the neural network and not the output weights, which are only affected indirectly.

In the linear network case, where we have a set of P examples from a dataset $\mathcal{D}$ $\{\vec{x_i}, \vec{y_i}\} \in \mathcal{D}, i = 1, ..., P$ and consider a neural network given by:

$$\hat{y} = W_2 W_1 \vec{x} \equiv W_2 \vec{h} \tag{4.5}$$

$$\hat{h} = W_4 W_3 \vec{h} \equiv W_4 \vec{z} \tag{4.6}$$

where h is the hidden state and z is a compressed representation of. The training objectives can be formulated as minimizing the SSE terms:

$$SSE(W_1, W_2) = \frac{1}{2}(||\vec{y_i} - \hat{\vec{y}_i}||^2 + ||\vec{h_i} - \hat{\vec{h}_i}||^2) \tag{4.7}$$

$$SSE(W_3, W_4) = \frac{1}{2}||\vec{h_i} - \hat{\vec{h}_i}||^2 \tag{4.8}$$

Note that the relevant gradients are given by (note the hats on some terms):

$$\frac{\partial}{\partial W_1} SSE(W_1, W_2) = W_2^T(\vec{y}_i - \hat{\vec{y}}_i)\vec{x}_i^T + (\vec{h}_i - \hat{\vec{h}}_i)\vec{x}_i^T \tag{4.9}$$

$$\frac{\partial}{\partial W_2} SSE(W_1, W_2) = (\vec{y}_i - \hat{\vec{y}}_i)\vec{h}_i^T \tag{4.10}$$

$$\frac{\partial}{\partial W_3} SSE(W_3, W_4) = W_4^T(\vec{h}_i - \hat{\vec{h}}_i)\vec{h}_i^T \tag{4.11}$$

$$\frac{\partial}{\partial W_4} SSE(W_3, W_4) = (\vec{h}_i - \hat{\vec{h}}_i)\vec{z}_i^T \tag{4.12}$$

Let the step size λ be small so that weight changes can be treated as negligible between inputs and only significant over training epochs (i.e. the whole dataset of P examples). Then we have the following weight change for W_1:

$$\begin{aligned}
\Delta W_1(t) &\approx \lambda \sum_{i=1}^{P} W_2^T(\vec{y}_i - \hat{\vec{y}}_i)\vec{x}_i^T + (\vec{h}_i - \hat{\vec{h}}_i)\vec{x}_i^T = \\
&= \lambda \sum_{i=1}^{P} (W_2^T(\vec{y}_i - \hat{\vec{y}}_i) + \vec{h}_i - \hat{\vec{h}}_i)\vec{x}_i^T = \\
&= \lambda \sum_{i=1}^{P} (W_2^T(\vec{y}_i - W_2W_1\vec{x}_i) + W_1x - W_4W_3W_1\vec{x}_i)\vec{x}_i^T = \\
&= \lambda P(W_2^T\mathbb{E}(yx^T) - W_2^TW_2W_1\mathbb{E}(xx^T) + W_1\mathbb{E}(xx^T) - W_4W_3W_1\mathbb{E}(xx^T)) = \\
&= \lambda P(W_2^T(\mathbb{E}(yx^T) - W_2W_1\mathbb{E}(xx^T)) + (I - W_4W_3)W_1\mathbb{E}(xx^T))
\end{aligned}$$

where we have used the definition of the expectation over all pairs $(\vec{x}, \vec{y})$ from the dataset, $\mathbb{E}(g(x_i) = \frac{1}{P}\sum_{i=1}^{P} g(x_i)$. The first term is identical to the case with no internal autoencoder [53], while the second captures the effect of the internal autoencoder on the weight dynamics. Notice that in the case the autoencoder has converged ($W_4W_3 \approx I$), the second term goes to 0 and the dynamics are identical to the case with no internal autoencoder.

Importantly, the equations suggest the existence of a fixed point when $\mathbb{E}(yx^T) = W_2W_1\mathbb{E}(xx^T)$ and $I = W_4W_3$ hold. The former term is exactly the same as the fixed point when there is no internal compression and shows that the weights are driven to capture covariance between the inputs and outputs from the dataset.

However, the latter term cannot hold in general, since $\dim(z) < \dim(h)$, so typically it will converge to an operator that projects onto the top-$k = \dim(z)$ principal subspace of h [54]. Then denoting $W_4 W_3 = P^{(k)}$, note that $I - P^{(k)} = P_\perp^{(k)}$ i.e. the linear operator that projects onto the subspace orthogonal to the top-k principal components Then any fixed point must satisfy:

$$(W_2^T(\mathbb{E}(yx^T) - W_2 W_1 \mathbb{E}(xx^T)) + P_\perp^{(k)} W_1 \mathbb{E}(xx^T)) = 0 \tag{4.13}$$

Applying $P^{(k)}$ to both sides of the above equation and noting that $P^{(k)} P_\perp^{(k)} = 0$, we have:

$$P^{(k)}(W_2^T(\mathbb{E}(yx^T) - W_2 W_1 \mathbb{E}(xx^T)) = 0 \tag{4.14}$$

$$\implies P^{(k)} = 0 \text{ or } W_2^T(\mathbb{E}(yx^T) - W_2 W_1 \mathbb{E}(xx^T) = 0 \tag{4.15}$$

$$\implies \mathbb{E}(yx^T) = W_2 W_1 \mathbb{E}(xx^T) \tag{4.16}$$

Then the fixed point is the same as the case when there is no internal autoencoder *within the principle subspace*. Further, applying the projection $P_\perp^{(k)}$ to both sides and using $(P_\perp^{(k)})^2 = P_\perp^{(k)}$:

$$P_\perp^{(k)}(W_2^T(\mathbb{E}(yx^T) - W_2 W_1 \mathbb{E}(xx^T)) + (P_\perp^{(k)})^2 W_1 \mathbb{E}(xx^T)) = 0 \tag{4.17}$$

$$\implies P_\perp^{(k)}(W_2^T(\mathbb{E}(yx^T) - W_2 W_1 \mathbb{E}(xx^T)) + W_1 \mathbb{E}(xx^T)) = 0 \tag{4.18}$$

$$\implies P_\perp^{(k)} = 0 \text{ or } W_2^T(\mathbb{E}(yx^T) - W_2 W_1 \mathbb{E}(xx^T) = 0 \tag{4.19}$$

$$\implies \mathbb{E}(yx^T) = W_2 W_1 \mathbb{E}(xx^T) \tag{4.20}$$

The same relation as before, except outside of the principle subspace. Therefore, the method overall preserves the same fixed points of the method.

Note the dynamics of the other weights are unchanged due to the stop gradient operation and match earlier work for the output weights W_2 [53] and the autoencoder [54].

4.3 Computational Results

4.3.1 Internal Compression Increases Training Speed and Performance on Classification Tasks

We trained multilayer perceptrons on the MNIST handwritten digit recognition task [55]. Across experiments, we initialized any neural network weights and hyperparameters to the same values for exact comparison of the effect the internal compression had on the training dynamics, yielding a clean comparison for each experiment. In total, we used 20 different weight initializations for comparing the models and computing statistics. The full details of the general architecture and training schedule can be found in Appendix A.

We first investigated whether internal compression facilitates or sabotages learning, since it is plausible that the compression objective could interfere with task learning, especially early in training when the autoencoder has low performance. Both models achieved high performance on the MNIST benchmark, with low training loss and high validation accuracy after five epochs (Figure 4.2, A, B). The internal compression model consistently achieved higher performance (validation accuracy: 97.03% $\pm$ 0.15 versus 96.46% $\pm$ 0.24, one-tailed Mann-Whitney U-test : p=8.25×10^{-8}) and achieved it more quickly (50% faster time to reach median accuracy, logrank test: p=1.91×10^{-11}). Despite the fact that the autoencoder was not necessarily performant at the beginning of training, there is no indication that the internal compression loss lead to any decline in early classification performance.

We further trained deep convolutional neural networks on the CIFAR10 classification task [56], which features small color images of ten classes of animals and vehicles. As before, neural network weights and all hyperparameters were initialized to the same values for comparing the effect that internal compression had on the training. The details of the architecture can be found in Appendix A. The same advantage of the internal compression method was present in this case, despite the classification task being more complicated and the network now having a convolutional component. In the case of CIFAR10 (Figure 4.2 C, D), the internal compression model achieved a top-1 validation accuracy of 66.34% $\pm$ 0.47 compared to 63.73% $\pm$ 0.74 for the null model. This difference was statistically

significant (one-tailed Mann-Whitney U-test : p=3.38×10^{-8}) and as in the MNIST case the training occurred more quickly in the internal compression model (22% faster time to reach median accuracy, logrank test: p=9.16×10^{-11}).

Therefore, the internal compression objective appeared to counterintuitively facilitate both learning speed and the final performance. This suggests that the objective could act as an inductive bias toward more parsimonious features that facilitate task performance.

4.3.2 Model Compressibility, Explained Variance, Participation Ratio, and Multifractal Spectra

If the internal compression model facilitates learning parsimonious features, this should be apparent in the activation statistics of the model. We find that, beyond having generally better final performance, the models with internal compression had more compressible activations relative to the null model (Figure 4.3). Thus, the method did function to increase the compressibility of the internal state with respect to the autoencoder attached to the classifier. While this may seem tautological, it is important to emphasize that the gradient computation for optimization of the classifier's weights was not directly influenced by the autoencoder due to the stopgradient operation; the objective $\mathcal{L}_{ic}$ did successfully work together with the autoencoder despite having no direct gradient flow between them.

While the co-trained autoencoder does capture some aspects of compressibility of the states, it could still be possible that the compressibility was not a general feature and depended on the autoencoder. If it is a more general feature, we would expect the internal states to be more linearly compressible. We further hypothesize that internal compression accelerates training by encouraging the model to first capture the largest eigenvectors of the data covariance matrix. This is consistent with the derivation in Section 4.2, which shows that in a linear version of the method, at equilibrium the internal autoencoder should project on to the top-k eigenvectors of the internal activation covariance matrix (where k is the

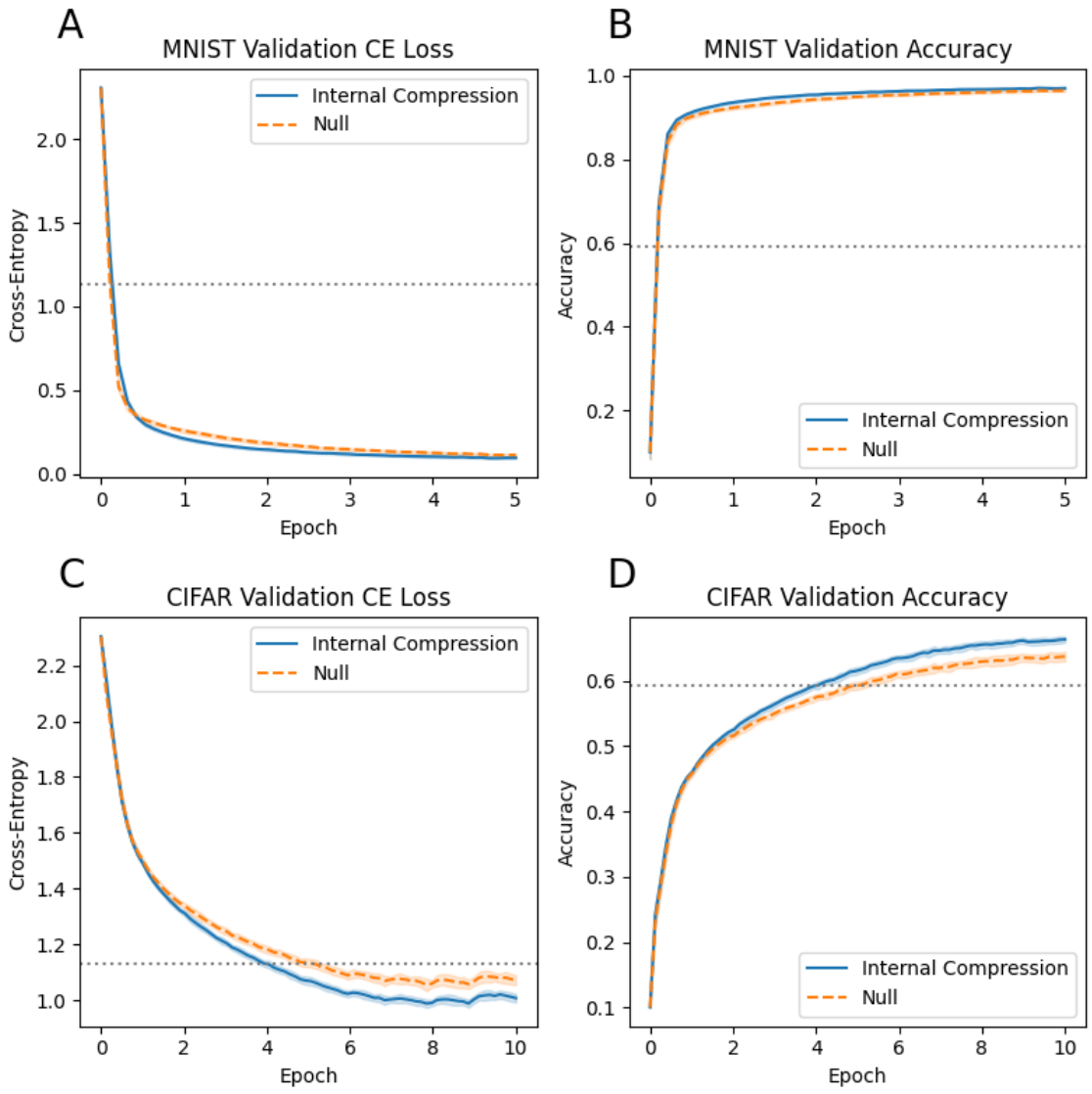

Figure 4.2. A) The validation set cross-entropy loss on the MNIST digit classification task over training. B) The validation accuracy for MNSIT on unseen test images. In both figures, the automatically determined threshold for the logrank test is depicted with a gray dashed line and solid lines indicate the mean training performance and transparent bounds indicate one standard deviation. C) The validation set cross-entropy loss on the CIFAR10 classification task over training time. D) The validation accuracy on unseen test images.

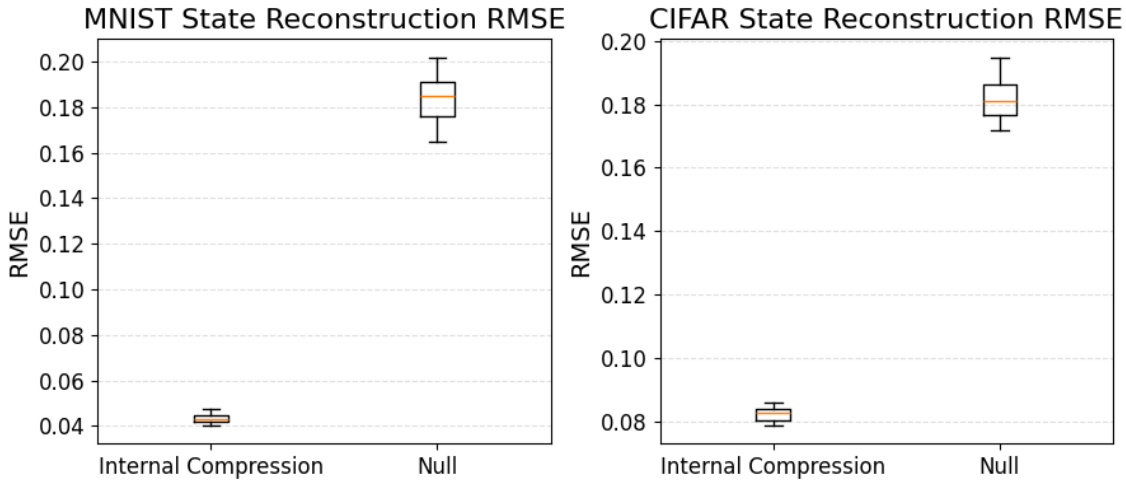

Figure 4.3. A boxplot of the final (RMSE) state reconstruction losses (eq. 4.1) for internal compression and null model for the two datasets. The differences are statistically significant (one-tailed Mann-Whitney U-test : p=3.40×10^{-8}).

dimension of the internal autoencoder bottleneck). In order to investigate this, we examined the participation ratio and cumulative explained variance over training.

In support of our hypothesis, we found that fewer dimensions in the internal compression model explained more variance in the data (Figure 4.4 Panels A,C) and had lower participation ratio (Figure 4.4 Panels B,D). This was especially pronounced in the CIFAR models. This was likely due to the low intrinsic dimension of MNIST [57], which means that both models need few dimensions to encode the features of the data. Note that at the beginning of training (dashed black line), all models covariance matrices had slowly decaying covariance spectrum, where many components were required to explain variance in the data. After 25 gradient steps (faintest lines in blue and orange), both models had much narrower spectra and a few dimensions could explain most of the variance in the data. This was especially rapid for the internal compression model, supporting the idea that internal compression encourages the models to first capture the largest eigenvectors of the data covariance matrix, suggesting that the method facilitates the known spectral bias [35] that is already known to be present in neural networks.

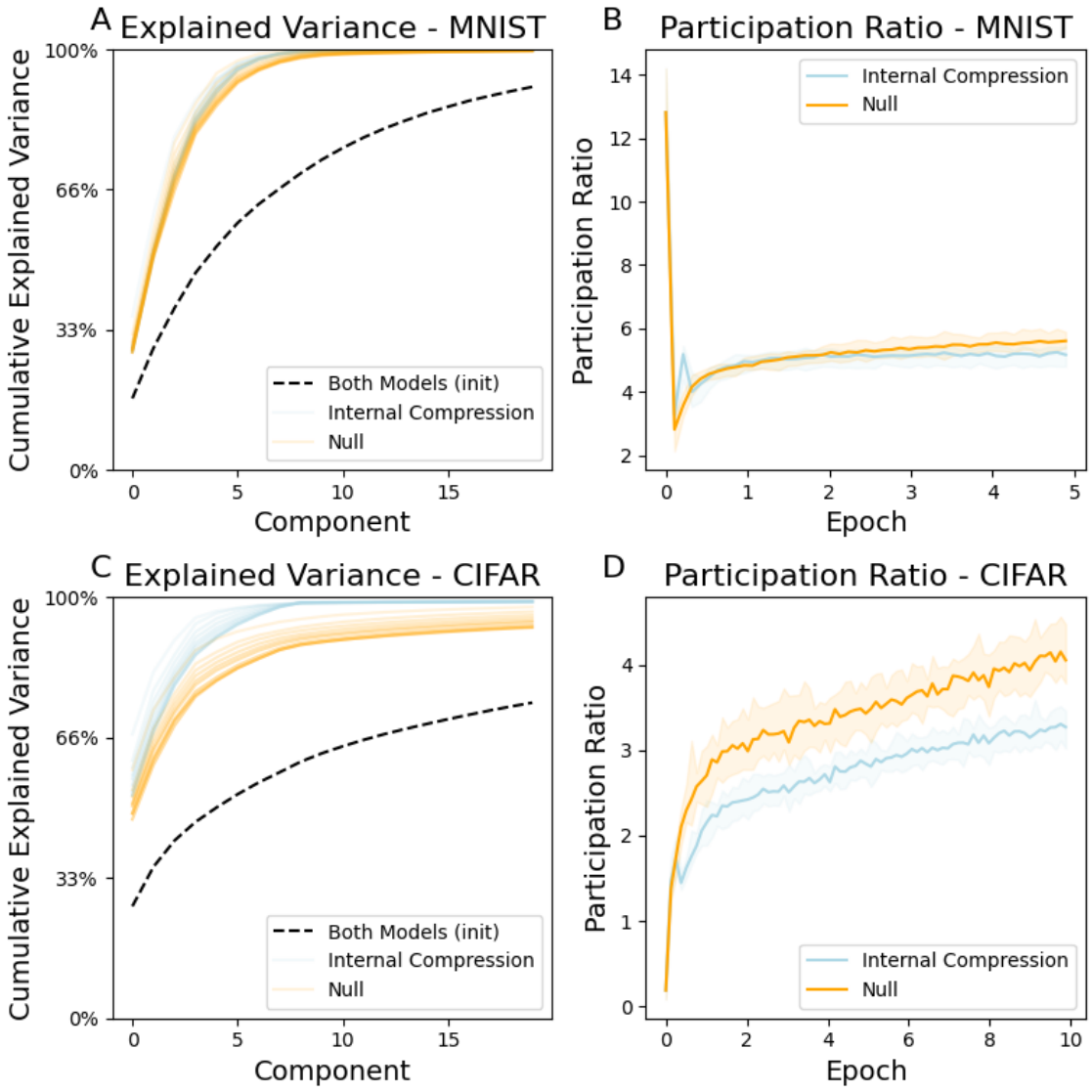

Figure 4.4. Explained variance and participation ratio, by dataset. A, C: Explained variance for MNIST and CIFAR, respectively. Lighter shades indicate earlier in training, darker shades indicate later in training. The dashed black line indicates the starting value, shared by both models. In both cases, but especially in the case of CIFAR, the model with internal compression had greater percentage of variance explained by fewer dimensions. In general, as training progressed, data variance became more spread out across embedding dimensions. B,D: The participation ratio for MNIST and CIFAR, respectively. In the case of MNIST, the participation ratio quickly declined, then slowly rose over training. This is likely due to the low intrinsic dimensionality of the data. For CIFAR, the participation ratio grew with training and remained consistently lower for the internal compression model.

Finally, we evaluate the multifractal spectra between the two methods using the BB estimator. The results are depicted in Figure 4.5. We plot multifractal spectrum of the model prior to training, which is depicted with a black, dashed line. In both MNIST (Figure 4.5, A) and CIFAR (Figure 4.5, B) the model's monofractal dimension decreased with model training. This is expected in the context of the previous results of this section: the neural network's states occupied a lower dimensional manifold as training progressed.

Furthermore, there was somewhat surprisingly no significant difference between multifractal spectra for the null model and the model with internal compression. We hypothesize that this could be related to the Platonic representation hypothesis [58], which is based on observations that different types of neural networks tend to converge to having similar internal representations when trained on the same data. This would further suggest that the multifractal spectra observed correspond *more to the data* than the training method.

One concern with applying the BB method, or any multifractal spectrum estimation method, to neural networks is that it could mislead and produce spurious results when there is no true underlying multifractal structure. While we cannot rule out possible alternatives, we generate two counterfactual scenarios. In the first scenario, we ran a permutation test, where we permute the elements across the vector embeddings of the images. This generates a permuted dataset where any correlations between data dimensions i, j are broken. The result is shown with a dark grey, dashed line in Figure 4.6. As can be seen, this strongly distorted the estimated spectrum, confirming that the estimated multifractal spectrum was dependent on the particular structure induced by the neural network.

The second counterfactual test generated uniform vectors with a similar extent to the true embeddings, by sampling from the hypercube defined by the ranges of the true embeddings. The result of this is shown by the light gray, dotted line in Figure 4.6. Unsurprisingly, this led to a $f(\alpha)$ curve with a high monofractal dimension, since it generated a dense cube in the ambient dimension.

The previous two counterfactuals support at least monofractal behavior of the neural network embeddings, but they don't help distinguish monofractal

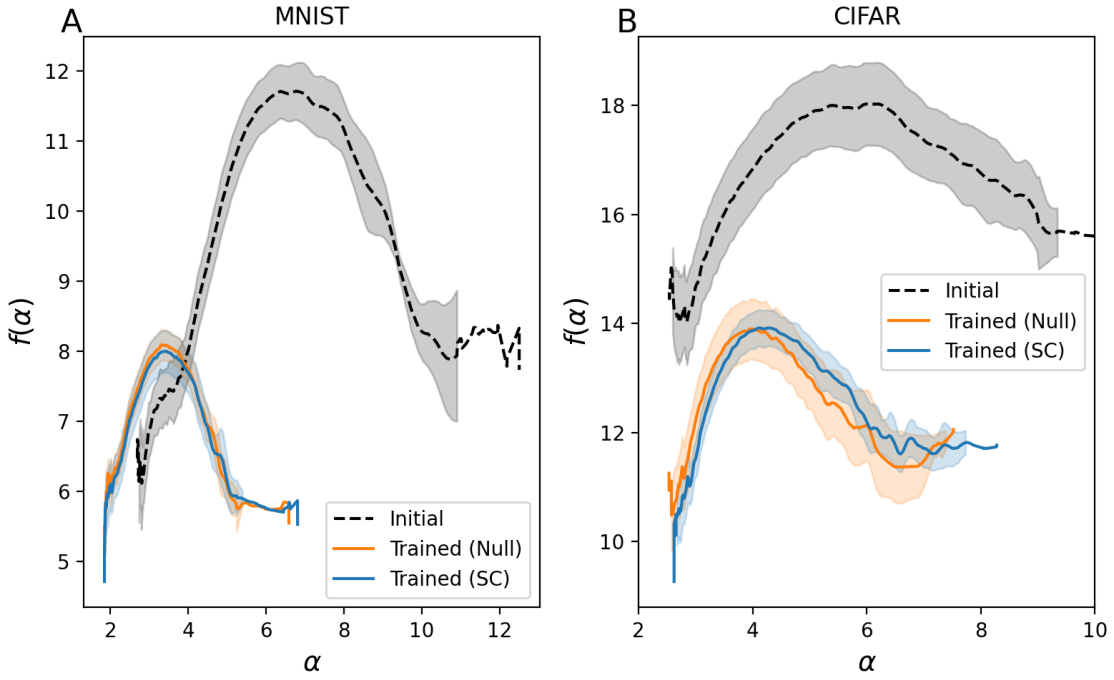

Figure 4.5. The estimated multifractal spectra for A) MNIST and B) CIFAR. The black curve indicates the spectrum for the initial state of the model. In both cases, training shifted the monofractal dimension to be lower.

behavior from multifractal behavior. Unfortunately, the BB method is not compatible with estimating the $\tau(q)$ and D_q functions necessary to distinguish multifractal from monofractal behavior. For this reason, we opted to measure the 90% interdecile range of the estimated α values as sample size grows. Theoretically, a monofractal like a Cantor set or a non-fractal like a uniform box should be expected to converge to a δ function which has $f(\alpha^*) = D_0$ for some value α^* and is undefined elsewhere. Therefore, as we vary the sample size, a true multifractal should be expected to remain converge to a non-zero value in the range of its α values, while a monofractal or non-fractal should go to 0.

Figure 4.7 shows the α interdecile range as we vary the number of samples. The width of the α range was consistently larger for the estimator based on the

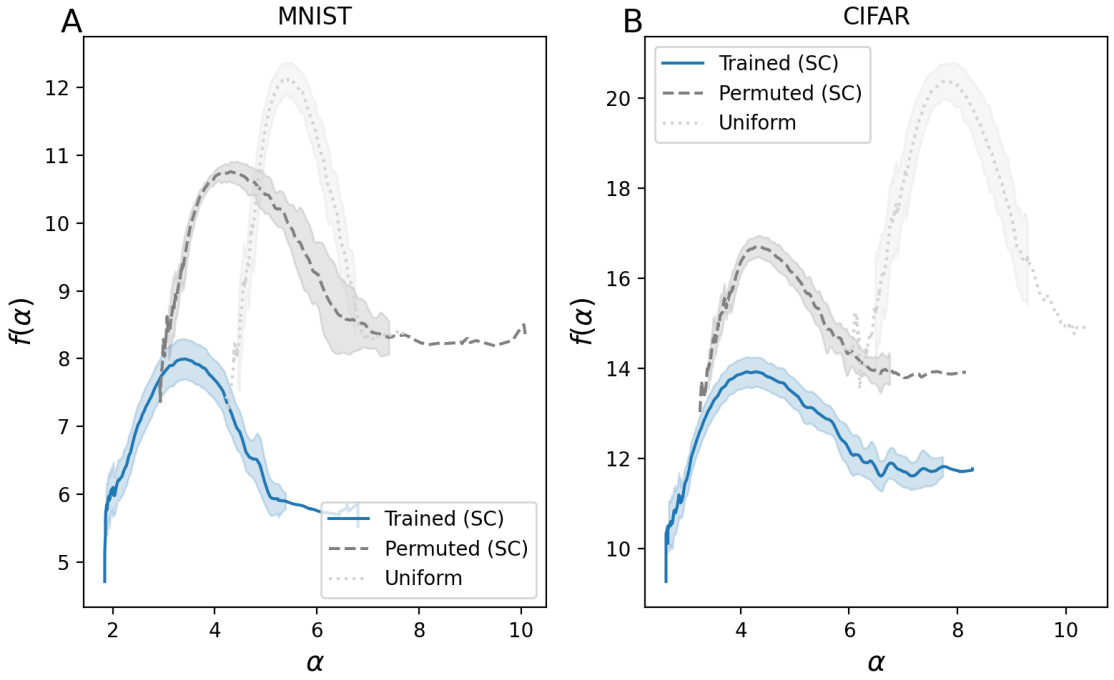

Figure 4.6. The counterfactual examples for CIFAR. The first counterfactual consists of permuting the elements between vectors, breaking any correlations that may be present (dashed grey curve). The second counterfactual involves uniformly sampling within the region defined by the data bounds.

neural network states than the uniform counterfactual. Furthermore, while the estimate based on the neural network states grew or remained relatively constant with more samples, the estimate from the uniform counterfactual decreased and remained constant. While we had insufficient data to examine $N > 10,000$, the plot nevertheless provides evidence in favor of the hypothesis that the neural network states exhibited multifractal behavior and not monofractal/non-fractal.

4.3.3 Weight Distributions

Finally, we asked if this form of self-modeling lead to large differences in the model weight distributions. We emphasize that while the initial weights vaired across experiments, the models were initialized to have the same starting weights in

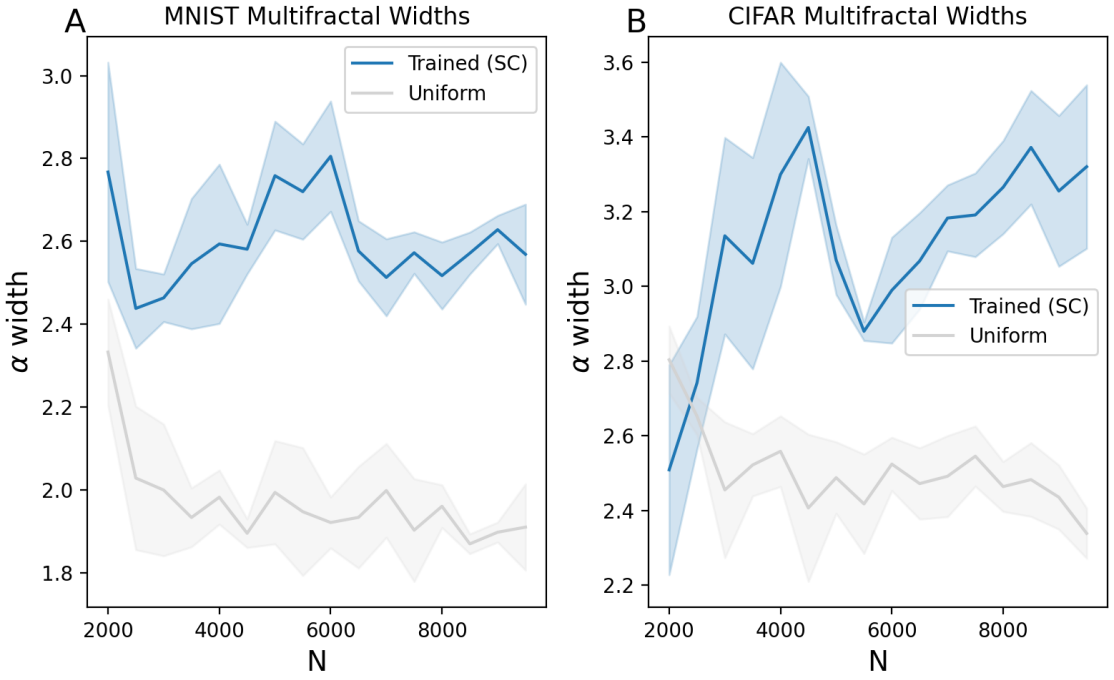

Figure 4.7. The 90% interdecile range for α for states of the trained model and the uniform counterfactual, as sample size (N) is varied, for A) MNIST and B) CIFAR datasets. Data limitations meant that only 10,000 samples could be used.

a given experiment. This ensures they were initially exactly identical. Figures 4.8 shows the evolution of the KS statistics across the MNIST and CIFAR10 datasets. Internal compression had the strongest effects on multilayer perceptron weights (here referred to as "linear" weights, since they correspond to dense matrices), whereas convolutional weights appeared to be unaffected. For the MNIST classification task, the weight distributions for linear weights rapidly diverged, while the divergence was slower and consistent for the CIFAR10 classification task. This shows that the regularization was in fact leading to large changes in the internal structure of the neural network.

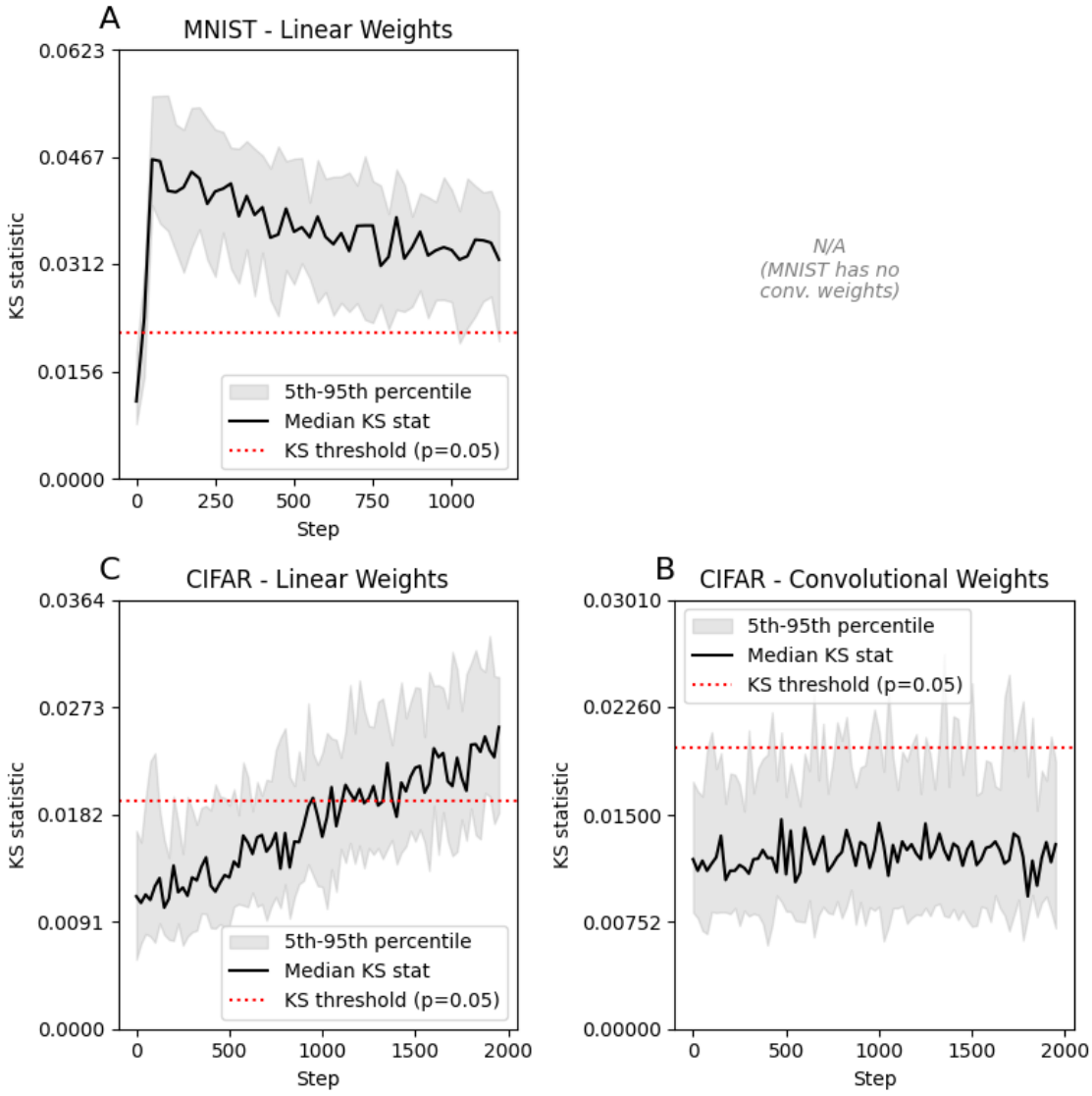

Figure 4.8. The Kolmogorov-Smirnov statistic between the null and internal compression models for various datasets and weight types, with shading indicating the 5th and 95th percentiles across runs. Red dashed lines indicates the significance threshold for a nominal pointwise KS test evaluated at a single point in time, see the methods sections for caveats about extending signifance to training trajectories. A) MNIST, linear weights: The statistic consistently increased at the beginning of training, then slowly decreased. B) For the linear weights in the CIFAR classification task, the null model and internal compression model grew increasingly dissimilar with time. C) In contrast, the convolutional weights in the CIFAR classification task remained relatively similar over time. Note there are no convolutional weights for the MNIST case.

CHAPTER 5
DISCUSSION AND CONCLUSION

We found that internal compression was a useful inductive bias, improving training speed and final accuracy. Our analysis suggests this is accompanied by increases in compressibility and stronger features that explain more variance in the data. Intriguingly, we found that the method had no discernable effect on the multifractal characteristics of the data, beyond those found in an ordinary classification model. These changes were accompanied by changes in the weight distribution of the trained networks, showing that the method changed the overall network organization.

We interpret the method as facilitating the existing spectral bias [35] of neural networks, which is supported by our linear network analysis (Section 4.2). Further evidence for this interpretation comes from participation ratio and explained variance being concentrated on fewer dimension in the model activations, which is most pronounced *early in training*, as would be expected if the method leads to a stronger spectral bias. Overall, this suggests that the method helps the network settle into a compact representational basis earlier when learning the task.

We also applied fixed-mass multifractal estimation techniques to the states of a neural network; this is the first time this has been done, to our knowledge. We found that fixed-mass methods work well in high dimensional neural network state spaces and provide consistent estimates across varying samples of the data, suggesting that they are indeed capturing some underlying multifractal characteristics in the data. We generated several counterfactual tests which support that this reflects true multifractal properties of the neural network representations and is not merely a result of noise in the data. Most intriguingly, we found that our regularization technique had no significant effect on the multifractal spectrum when compared to a similar model without the regularization technique. We view this as potential evidence in favor of the Platonic

representation hypothesis [58], which asserts that neural networks tend to learn similar representations regardless of architectural choice. We note that there is no paradox here with the observed difference in compressibility of the internal representations: while the representations were more compressible, the interrelationships among the representations across the data were similar.

We believe this also represents a positive step toward the study of self-modeling in neural networks where the neural network must use its "self-model" toward some end. This avoids some of the issues with the homunculus fallacy [59] that can accompany ideas of self-modeling, where a "self-model" needs some additional internal model to be interpreted. We avoid this by having the model and the "self-model" mutually influence each other. While simple artificial neural networks like this are a far-cry from any realistic cognitive system, simple models of self-modeling such as this one nevertheless help highlight potential uses actual cognitive systems may have for representing and modifying their own internal processes.

Potential future work includes investigating how these methods scale up to larger neural network models and to tasks other than classification. Furthermore, it remains to be seen how our method works with other common methods in machine learning like batch normalization or advanced optimizers. Further research we see as especially promising are: (1) Testing the method with transformer and attention-based [60] architectures, (2) Applying the method to reinforcement learning, (3) Determining if the method provides any benefit to interpretability, given that the framework equipped with a sparse autoencoder [25] could be interpreted as a network that makes itself more interpretable.

BIBLIOGRAPHY

[1] Thomas Metzinger. *Being No One: The Self-Model Theory of Subjectivity*. The MIT Press, 01 2003.

[2] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[3] Boyuan Chen, Robert Kwiatkowski, Carl Vondrick, and Hod Lipson. Full-body visual self-modeling of robot morphologies, 2021.

[4] Robert Kwiatkowski, Yuhang Hu, Boyuan Chen, and Hod Lipson. On the origins of self-modeling, 2022.

[5] Raymond Douglas, Jan Kulveit, Ondrej Havlicek, Theia Pearson-Vogel, Owen Cotton-Barratt, and David Duvenaud. The artificial self: Characterising the landscape of ai identity, 2026.

[6] Vickram Premakumar, Michael Vaiana, Florin Pop, Judd Rosenblatt, Diogo Schwerz de Lucena, Kirsten Ziman, and Michael S. A. Graziano. Unexpected benefits of self-modelling in neural systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 384(2320):20240531, 05 2026.

[7] Thomas O. Nelson. Metamemory: A theoretical framework and new findings. volume 26 of *Psychology of Learning and Motivation*, pages 125–173. Academic Press, 1990.

[8] Nicholas Shea, Annika Boldt, Dan Bang, Nick Yeung, Cecilia Heyes, and Chris D. Frith. Supra-personal cognitive control and metacognition. *Trends in Cognitive Sciences*, 18(4):186–193, 2014.

[9] Joseph E. LeDoux and Richard Brown. A higher-order theory of emotional consciousness. *Proceedings of the National Academy of Sciences*, 114(10):E2016–E2025, 2017.

[10] Michael Levin. Self-improvising memory: A perspective on memories as agential, dynamically reinterpreting cognitive glue. *Entropy*, 26(6), 2024.

[11] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[12] Erik Boogaard, Jan Treur, and Maxim Turpijn. A neurologically inspired network model for graziano's attention schema theory for consciousness. 06 2017.

[13] Andrew I. Wilterson and Michael S. A. Graziano. The attention schema theory in a neural network agent: Controlling visuospatial attention using a descriptive model of attention. *Proceedings of the National Academy of Sciences*, 118(33):e2102421118, 2021.

[14] Dianbo Liu, Samuele Bolotta, He Zhu, Yoshua Bengio, and Guillaume Dumas. Attention schema in neural agents, 2023.

[15] Michael S A Graziano and Taylor W Webb. The attention schema theory: a mechanistic account of subjective awareness. *Front Psychol*, 6:500, April 2015.

[16] Ryota Kanai, Ryota Takatsuki, and Ippei Fujisawa. Meta-representations as representations of processes. *Neuroscience of Consciousness*, 2025(1):niaf038, 02 2025.

[17] Jürgen Schmidhuber. Steps towards 'self-referential' neural learning: A thought experiment ; cu-cs-627-92. 1992.

[18] Louis Kirsch and Jürgen Schmidhuber. Eliminating meta optimization through self-referential meta learning, 2022.

[19] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. A modern self-referential weight matrix that learns to modify itself, 2022.

[20] Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, Jan 1999.

[21] James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, 29(5):1229–1262, 05 2017.

[22] Léonard Blier and Yann Ollivier. The description length of deep learning models, 2018.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9:1–42, 01 1997.

[24] Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural networks : the official journal of the International Neural Network Society*, 10 5:857–873, 1997.

[25] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.

[26] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[27] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[28] Navid Shervani-Tabar, Marzieh Alireza Mirhoseini, and Robert Rosenbaum. Oja's plasticity rule overcomes several challenges of training neural networks under biological constraints, 2025.

[29] Krishna B. Athreya, John M. Hitchcock, Jack H. Lutz, and Elvira Mayordomo. Effective strong dimension, algorithmic information, and computational complexity, 2003.

[30] Jack H. Lutz and Elvira Mayordomo. Algorithmic fractal dimensions in geometric measure theory, 2020.

[31] R. Badii and G. Broggi. Measurement of the dimension spectrum $f(\alpha)$: Fixed-mass approach. *Physics Letters A*, 131(6):339–343, 1988.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[33] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[34] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[35] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019.

[36] Stefano Recanatesi, Serena Bradde, Vijay Balasubramanian, Nicholas A Steinmetz, and Eric Shea-Brown. A scale-dependent measure of system

dimensionality. *bioRxiv*, 2020.

[37] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[38] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, Chichester, UK, 3rd edition, 2014.

[39] Felix Hausdorff. Dimension und äußeres Maß. *Mathematische Annalen*, 79(1–2):157–179, 1918.

[40] A. N. Kolmogorov and V. M. Tikhomirov. ε-entropy and ε-capacity of sets in functional spaces. *Uspekhi Matematicheskikh Nauk*, 14(2):3–86, 1959. English translation: American Mathematical Society Translations, Series 2, vol. 17, pp. 277–364, 1961.

[41] H.G.E. Hentschel and Itamar Procaccia. The infinite number of generalized dimensions of fractals and strange attractors. *Physica D: Nonlinear Phenomena*, 8(3):435–444, 1983.

[42] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Doklady Akademii Nauk SSSR*, 30:9–13, 1941. Reprinted in Proceedings of the Royal Society of London A, 434, 9–13 (1991).

[43] Giorgio Parisi and Uriel Frisch. On the singularity structure of fully developed turbulence. In M. Ghil, R. Benzi, and G. Parisi, editors, *Turbulence and Predictability in Geophysical Fluid Dynamics and Climate Dynamics*, Proceedings of the International School of Physics "Enrico Fermi", Course LXXXVIII, pages 84–87. North-Holland, Amsterdam, 1985.

[44] G Boffetta, A Mazzino, and A Vulpiani. Twenty-five years of multifractals in fully developed turbulence: a tribute to giovanni paladin. *Journal of Physics A: Mathematical and Theoretical*, 41(36):363001, August 2008.

[45] Thomas C. Halsey, Mogens H. Jensen, Leo P. Kadanoff, Itamar Procaccia, and Boris I. Shraiman. Fractal measures and their singularities: The characterization of strange sets. *Physical Review A*, 33(2):1141–1151, 1986.

[46] David Ruelle. *Thermodynamic formalism: the mathematical structures of classical equilibrium statistical mechanics*, volume 5 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley Pub. Co., Reading, Mass., 1978.

[47] Ashvin B. Chhabra and Roderick V. Jensen. Direct determination of the

$f(\alpha)$ singularity spectrum. *Physical Review Letters*, 62(12):1327–1330, 1989.

[48] Willem van de Water and Piet Schram. Generalized dimensions from near-neighbor information. *Phys. Rev. A*, 37:3118–3125, Apr 1988.

[49] Balázs Bárány and Manuj Verma. Hausdorff dimension of self-similar measures and sets with common fixed point structure, 2025.

[50] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020.

[51] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs, 2021.

[52] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[53] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, May 2019.

[54] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[55] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 1998.

[56] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[57] Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning, 2021.

[58] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis, 2024.

[59] Kimberly Baltzer-Jaray. *Homunculus*, chapter 31, pages 165–167. John Wiley & Sons, Ltd, 2018.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[61] Alfréd Rényi. On measures of entropy and information. In Jerzy Neyman,

editor, *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561, Berkeley, CA, 1961. University of California Press.

[62] A. N. Kolmogorov. A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high Reynolds number. *Journal of Fluid Mechanics*, 13(1):82–85, 1962.

[63] Benoit B. Mandelbrot. How long is the coast of Britain? Statistical self-similarity and fractional dimension. *Science*, 156(3775):636–638, 1967.

[64] Benoit B. Mandelbrot. Intermittent turbulence in self-similar cascades: divergence of high moments and dimension of the carrier. *Journal of Fluid Mechanics*, 62(2):331–358, 1974.

[65] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman, New York, 1982.

[66] H. G. E. Hentschel and Itamar Procaccia. The infinite number of generalized dimensions of fractals and strange attractors. *Physica D: Nonlinear Phenomena*, 8(3):435–444, 1983.

[67] Peter Grassberger. Generalized dimensions of strange attractors. *Physics Letters A*, 97(6):227–230, 1983.

[68] Peter Grassberger and Itamar Procaccia. Characterization of strange attractors. *Physical Review Letters*, 50(5):346–349, 1983.

[69] Peter Grassberger and Itamar Procaccia. Measuring the strangeness of strange attractors. *Physica D: Nonlinear Phenomena*, 9(1–2):189–208, 1983.

[70] Roberto Benzi, Giovanni Paladin, Giorgio Parisi, and Angelo Vulpiani. On the multifractal nature of fully developed turbulence and chaotic systems. *Journal of Physics A: Mathematical and General*, 17(18):3521–3531, 1984.

[71] Mogens H. Jensen, Leo P. Kadanoff, Albert Libchaber, Itamar Procaccia, and Joel Stavans. Global universality at the onset of chaos: Results of a forced Rayleigh-Bénard experiment. *Physical Review Letters*, 55(25):2798–2801, 1985.

[72] C. Meneveau and K. R. Sreenivasan. Simple multifractal cascade model for fully developed turbulence. *Physical Review Letters*, 59(13):1424–1427, 1987.

[73] Benoît B. Mandelbrot. Multifractal measures, especially for the geophysicist. *Pure and Applied Geophysics*, 131(1–2):5–42, 1989.

[74] Benoît B. Mandelbrot. Random multifractals: negative dimensions and the resulting limitations of the thermodynamic formalism. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 434(1890):79–88, 1991.

[75] C. Meneveau and K. R. Sreenivasan. The multifractal nature of turbulent energy dissipation. *Journal of Fluid Mechanics*, 224:429–484, 1991.

[76] J. F. Muzy, E. Bacry, and A. Arneodo. Wavelets and multifractal formalism for singular signals: Application to turbulence data. *Physical Review Letters*, 67(25):3515–3518, 1991.

[77] J. F. Muzy, E. Bacry, and A. Arneodo. Multifractal formalism for fractal signals: The structure-function approach versus the wavelet-transform modulus-maxima method. *Physical Review E*, 47(2):875–884, 1993.

[78] J. F. Muzy, E. Bacry, and A. Arneodo. The multifractal formalism revisited with wavelets. *International Journal of Bifurcation and Chaos*, 4(2):245–302, 1994.

[79] C.-K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger. Mosaic organization of DNA nucleotides. *Physical Review E*, 49(2):1685–1689, 1994.

[80] Benoît B. Mandelbrot, Adlai J. Fisher, and Laurent E. Calvet. A multifractal model of asset returns. Cowles Foundation Discussion Paper 1164, Cowles Foundation for Research in Economics, Yale University, 1997.

[81] Plamen Ch. Ivanov, Luís A. Nunes Amaral, Ary L. Goldberger, Shlomo Havlin, Michael G. Rosenblum, Zbigniew R. Struzik, and H. Eugene Stanley. Multifractality in human heartbeat dynamics. *Nature*, 399(6735):461–465, 1999.

[82] Jan W. Kantelhardt, Stephan A. Zschiegner, Eva Koscielny-Bunde, Shlomo Havlin, Armin Bunde, and H. Eugene Stanley. Multifractal detrended fluctuation analysis of nonstationary time series. *Physica A: Statistical Mechanics and its Applications*, 316(1–4):87–114, 2002.

[83] Luc Devroye, László Györfi, Gábor Lugosi, and Harro Walk. On the measure of voronoi cells, 2015.

[84] Damiano Lombardi and Sanjay Pant. Nonparametric k-nearest-neighbor entropy estimator. *Phys. Rev. E*, 93:013310, Jan 2016.

[85] Rien van de Weygaert, Bernard J.T. Jones, and Vincent J. Martínez. The

minimal spanning tree as an estimator for generalized dimensions. *Physics Letters A*, 169(3):145–150, 1992.

[86] Henry Adams, Manuchehr Aminian, Elin Farnell, Michael Kirby, Joshua Mirth, Rachel Neville, Chris Peterson, and Clayton Shonkwiler. *A Fractal Dimension for Measures via Persistent Homology*, page 1–31. Springer International Publishing, 2020.

[87] Jonathan Jaquette and Benjamin Schweinhart. Fractal dimension estimation with persistent homology: A comparative study. *Communications in Nonlinear Science and Numerical Simulation*, 84:105163, May 2020.

[88] Sifan Wang, Shyam Sankaran, Xiantao Fan, Panos Stinis, and Paris Perdikaris. Simulating three-dimensional turbulence with physics-informed neural networks, 2025.

[89] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020.

[90] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[92] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks, 2015.

[93] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[94] Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, January 2017.

[95] Omar Sallam and Mirjam Fürth. On the use of fourier features-physics informed neural networks (ff-pinn) for forward and inverse fluid mechanics problems. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 237(4):846–866, 2023.

[96] Jongwon Kim and Ramana M. Pidaparti. Computational analysis of lung and isolated airway bifurcations under mechanical ventilation and normal breathing. *Fluids*, 6(11), 2021.

[97] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[98] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[99] JongWon Kim, Rebecca L. Heise, Angela M. Reynolds, and Ramana M. Pidaparti. Aging effects on airflow dynamics and lung function in human bronchioles. *PLOS ONE*, 12(8):1–20, 08 2017.

[100] T. Kawabata and A. Dembo. The rate-distortion dimension of sets and measures. *IEEE Transactions on Information Theory*, 40(5):1564–1572, 1994.

[101] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

APPENDIX

Training and Architecture Details

Training and Architecture Details

We implemented all neural networks in PyTorch [97].

A.1 MNIST

We used the standard train and test split for MNIST from the PyTorch Vision library [101]. We trained for 5 epochs with a constant learning rate of 0.001 and a weight on internal compression loss of 2.25. We chose 5 epochs based on initial early stopping tests where this was found to be the number of epochs where both the null and internal compression model were not overfit, given the architecture and learning rate. The neural network's layers consisted of the following:

Layer Number	Layer Type	Layer Size
1	Linear	(784, 64)
2	Linear	(64, 64)
3	Linear	(64, 64)
4	Linear	(64, 10)

Between all layers was the GELU activation function [52].

The structure of internal autoencoder was:

Layer Number	Layer Type	Layer Size
1	Linear	(64, 19)
2	Linear	(19, 64)

Where the GELU activation was used between the layers. The layer size of 19 was chosen as the floor of 0.3×64, which was based on a grid search of ratios between autoencoder bottleneck dimensions and hidden dimension.

A.2 CIFAR10

Again, we used the standard train and test split for CIFAR10 from the PyTorch Vision library [101]. We trained for 20 epochs with a constant learning rate of 0.0002 and a weight on internal compression loss of 2.25. We chose 20 epochs based on the same initial early stopping tests where this was found to be the number of epochs where both the null and internal compression model were not overfit, given the architecture and learning rate. The neural network's layers consisted of the following:

Layer Number	Layer Type	Layer Size
1	Convolutional	(input channels = 3, output channels = 32, kernel size = 3, stride=1, padding=1)
2	Convolutional	(input channels = 32, output channels = 64, kernel size = 3, stride=2, padding=1)
3	Convolutional	(input channels = 64, output channels = 96, kernel size = 3, stride=2, padding=1)
4	Convolutional	(input channels = 96, output channels = 128, kernel size = 3, stride=2, padding=1)
5	Convolutional	(input channels = 128, output channels = 256, kernel size = 3, stride=2, padding=1)
6	Linear	(1024, 512)
7	Linear	(512, 10)

Between all layers was the GELU activation function [52].

The structure of internal autoencoder was:

Layer Number	Layer Type	Layer Size
1	Linear	(512, 153)
2	Linear	(153, 512)

Where the GELU activation was used between the layers. The layer size of 153 was chosen as the floor of 0.3×512, based on the same grid search as before.

Layer Number	Layer Type	Layer Size
1	Linear	(512, 256)
2	Reshape Upsample	(256)
3	Convolutional	(input channels = 64, output channels = 64, kernel size = 3, stride=1, padding=1)
4	Upsample	2
5	Convolutional	(input channels = 64, output channels = 64, kernel size = 3, stride=1, padding=1)
6	Upsample	2
7	Convolutional	(input channels = 64, output channels = 64, kernel size = 3, stride=1, padding=1)

where the Upsample indicates we use the default nearest neighbor interpolating function from PyTorch.