EMPIRICAL NTK TRACKS TASK COMPLEXITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Mathematical properties of the neural tangent kernel (NTK) have been related–both theoretically and empirically–to convergence of optimization algorithms and the ability of trained models to generalize. However, most existing theoretical results hold only in the infinite width limit and only for standard data distributions. In the present work, we suggest a practical approach to investigating the NTK for finite-width networks, by understanding the parameter space symmetries of the network in the presence of finite data sets. In particular, the NTK Gram matrix associated to any finite data set can naturally be regarded as an empirical version of the NTK. Moreover, its rank agrees with the *functional dimension* of the data set, the number of independent parameter perturbations affecting the model's outputs on the data set. In this work, we explore the evolution of the functional dimension of deep ReLU networks during training, focusing on the relationship to data set complexity, regularization, and training dynamics. Empirically, we find that functional dimension of deep ReLU networks: (1) tracks data set complexity, (2) increases during training until function stabilization, and (3) decreases with stronger weight decay, suggesting that gradient-based optimization algorithms are biased towards simpler functions for ReLU networks. Moreover, our experiments provide strong evidence that–contrary to conventional wisdom–the empirical NTK for deep finite-width ReLU networks is typically rank-deficient at initialization. We offer a potential theoretical explanation for this empirical phenomenon in terms of certain data-dependent hidden equivalences, emphasizing the connection between these equivalences and the geometry of the loss landscape. We also establish a theoretical upper bound on functional dimension in terms of the number of linear regions sampled by the data set.

1 INTRODUCTION

The neural tangent kernel (NTK) has emerged as a powerful tool for understanding the training dynamics and generalization properties of neural networks, especially in the infinite width limit Jacot et al. (2018); Lee et al. (2019). The spectrum of the NTK, in particular, has been shown to play a key role, and significant theoretical progress has been made in obtaining closed-form expressions for this spectrum Murray et al. (2023); Nguyen & Mondelli (2020); Nguyen et al. (2021). A full-rank NTK ensures a well-conditioned optimization problem, leading to efficient training and convergence Arora et al. (2019); Allen-Zhu et al. (2019). However, we still do not understand the effects of finite-width corrections, especially in the presence of nonstandard data distributions. Indeed, NTK theory has fallen short in predicting how real-world neural networks evolve when training on concrete data sets Geiger et al. (2019); Lee et al. (2020), and the Gram matrix of the NTK - referred to as the *empirical NTK* in the literature - frequently evolves significantly during training on real world data sets in a way that differs from the infinite-width predictions. Our goals here are:

(1) Track the evolution of the empirical NTK during training on synthetic data sets of increasing complexity (see Figures 1 and 3 below and Figures 11 in the Appendix);

(2) Relate this evolution to the task complexity (see Figure 2); and

(3) Relate the empirical NTK to a complementary theoretical framework involving *data-dependent parameter space symmetries* and their impact on the optimization dynamics of neural networks.

The starting point of our investigation are the following observations:

(1) The *rank* of the empirical NTK on a fixed batch of data agrees with the batch *functional dimension* (cf. Grigsby et al. (2022)), which can be viewed informally as the *effective local parametric dimension* on the batch;

(2) It has been observed empirically that for networks that are deeper than they are wide, the batch functional dimension is much lower than predicted by the existing theory of parameter space symmetries Grigsby et al. (2023).

Empirically we find:

- *For deep ReLU networks, the empirical NTK has low rank at initialization and increases during training until function stabilization.* The rank of the empirical NTK is precisely the batch functional dimension on the data set. In experiments (Figure 8), the batch functional dimension at initialization is consistently much smaller than the number of data points for heavily over-parameterized deep ReLU networks. During training, functional dimension on average increases with epoch until the function roughly "stabilizes." See Figure 8.
- *The rank of the empirical NTK tracks data set complexity during training.* Training with a small positive weight decay tends to select a function whose complexity, as measured by functional dimension, reflects the complexity of the data set. More complex data sets tend to result in trained functions that have higher functional dimension. See Figure 2.
- *Weight decay causes the rank of the empirical NTK to decrease after approximate function stabilization, and higher weight decay encourages lower rank.* If we continue training (using a small positive weight decay) past the point of approximate functional stabilization, functional dimension eventually decreases. (See Figures 7 and 2.) Weight decay has a damping impact on functional dimension. That is, higher (constant) weight decay leads to trained functions with lower functional dimension (see Figure 4).
- *With only modest weight decay, the functions learned by our heavily overparameterized function classes tend to* underfit *the training data.* See Figure 3.
- *Number of linear regions sampled is strongly correlated with the rank of the empirical NTK.* See Figure 10

We suggest two theoretical mechanisms encouraging a rank-deficient NTK, and relate these mechanisms to the existence of hidden parameter space symmetries:

- *Hidden data-dependent parameter space symmetries encourage a rank-deficient empirical NTK.* Restricted activation patterns cause ReLU networks to behave like smaller networks with more parameter space symmetries. When hidden neurons in a network are either always-active or always-inactive on a batch of data, then the network behaves like a mixed linear-ReLU subnetwork, which enlarges the dimension of the space of symmetries to be *quadratic* rather than *linear* in the number of hidden neurons (Proposition A.10).
- *Fewer linear regions encourage a rank-deficient empirical NTK.* We prove a theoretical upper bound (Proposition C.3) on the batch functional dimension in terms of the number of linear regions sampled by the data set. In the case of architectures with input dimension 1, as in the case of our univariate experiments, the bound asserts that the functional dimension is bounded above by twice the number of linear regions sampled by the batch. Attaining this upper bound would require that the data set and parameter satisfy very specific constraints, so it is unlikely for a typical parameter to attain the bound. In our experiments the number of intervals sampled is typically greater than the batch functional dimension.

2 Related work

Neural Tangent Kernel (NTK) training and generalization at infinite and finite width: The NTK was first defined and studied in Jacot et al. (2018), where it was established that in the infinite width limit gradient flow is entirely determined by the NTK and can be described via kernel gradient flow. See also Lee et al. (2019).

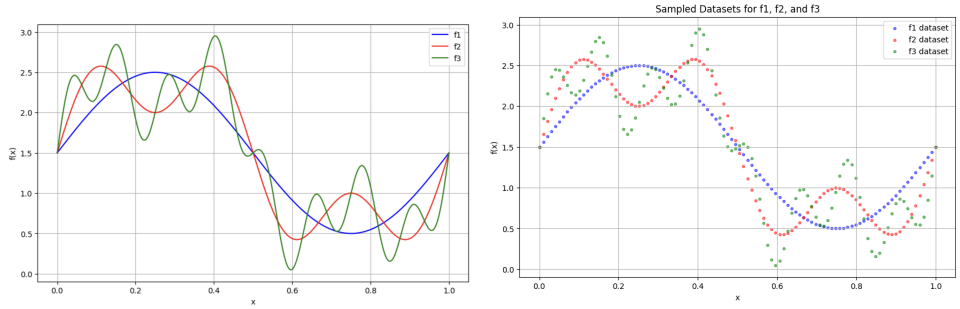

Figure 1: Graphs of three univariate functions of increasing complexity (left); uniformly sampled datasets for each function (right).

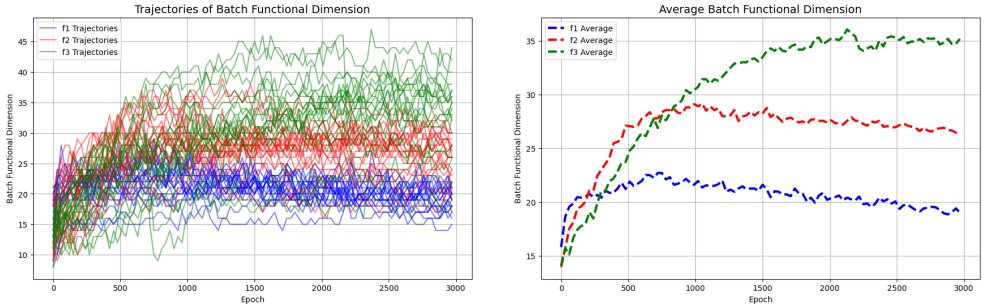

Figure 2: Functional dimension tracks task complexity. The rank of the empirical NTK after 3000 epochs of training correlates positively with the complexity of the function the model is learning.

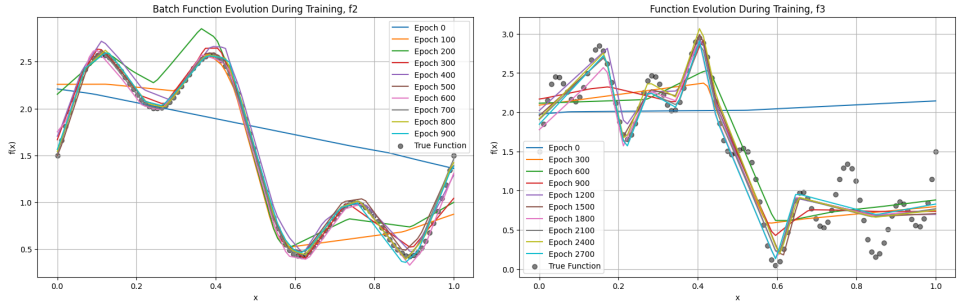

Figure 3: Function evolution for a single training run, for functions f_2 and f_3. Although the networks are overparameterized (100 data points and 1006 parameters), the learned function underfits. As indicated in the plot legends, we trained on f_3 for more total epochs than on f_2.

In Hanin & Nica (2020), the authors study the NTK at finite width and depth, arguing that sufficiently deep wide networks have interesting training behavior and can learn data-dependent features even in the so-called "lazy training" regime associated to very wide networks at fixed depth. In Huang & Yau (2020), the authors define and study the dynamics of gradient descent for finite-width networks under a so-called neural tangent hierarchy of differential equations, for which the NTK gives an approximation.

NTK eigenfunctions and spectrum analysis: In Xie et al. (2017), the authors study the dyanamics of the training loss for shallow ReLU neural networks, establishing a connection between the minimum singular value of the empirical NTK and the decay rates of both the training loss and the kernel spectrum associated to the arc-cosine kernel defined by Cho & Saul (2009). In Nguyen & Mondelli (2020) and Nguyen et al. (2021) the authors perform a spectrum analysis for deep ReLU networks with certain architecture restrictions, and in Murray et al. (2023), the authors derive a power series expansion for the NTK of arbitrarily deep feedforward networks in the infinite width limit that allows them to extract the eigenvalue spectrum.

Parameter space symmetries and optimization: Two largely-independent approaches to studying the relationships among parameter space symmetries, the geometry of the loss landscape, and the so-called *neuromanifold* (true function space after quotienting by symmetries) have emerged, as described in the recent survey papers Zhao et al. (2025), Marchetti et al. (2025), and the many references therein. The approach we take here is more closely aligned with the first survey article, although we are interested in connections to the second. We are not aware of any prior work explicitly discussing a relationship between the NTK spectrum and parameter space symmetries.

3 Background and notation

3.1 Fully-connected feedforward ReLU networks

We focus on fully connected neural networks with ReLU activation, denoting by $(n_0, \ldots, n_{d-1}|n_d)$[1] the architecture with input width n_0, hidden layer widths $n_1, \ldots, n_{d-1}$, and output width n_d.

Formally, let $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ denote the function that applies the activation function $\text{ReLU}(x) := \max\{0, x\}$ component-wise. For an architecture $(n_0, \ldots, n_{d-1}|n_d)$, we denote the parameter space $\Omega := \mathbb{R}^D$ where a parameter $\theta := (W^1, b^1, \ldots, W^d, b^d) \in \Omega$ consists of the entries of weight matrices $W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and bias vectors $b^\ell \in \mathbb{R}^{n_\ell}$ for $\ell = 1, \ldots, d$. Accordingly, $D := \sum_{\ell=1}^{d} n_\ell(n_{\ell-1} + 1)$. From a parameter θ we define a neural network function:

$$F_\theta : \mathbb{R}^{n_0} \xrightarrow{F^1} \mathbb{R}^{n_1} \xrightarrow{F^2} \ldots \xrightarrow{F^d} \mathbb{R}^{n_d} , \tag{1}$$

with layer maps given by:

$$F^\ell(x) := \begin{cases} \sigma(W^\ell x + b^\ell) & \text{for } 1 \leq \ell < d \\ W^\ell x + b^\ell & \text{for } \ell = d. \end{cases} \tag{2}$$

For any $\theta \in \Omega$, F_θ is a *finite piecewise-linear* function – that is, a continuous function for which the domain may be decomposed as the union of finitely many closed, convex pieces, on each of which the function is affine-linear. One also naturally obtains from an input vector $x \in \mathbb{R}^{n_d}$ an *evaluation map* $E_x : \Omega \to \mathbb{R}^{n_0}$, where $E_x(\theta) := F_\theta(x)$.

To compactify notation, following Masden (2022) we let $F_{(\ell)} := F^\ell \circ \ldots \circ F^1$. We refer to the components of $F_{(\ell)}$ as the *neurons* in the ℓth layer. The *pre-activation* map $z_{\ell,i} : \mathbb{R}^{n_0} \to \mathbb{R}$ associated to the ith neuron in the ℓth layer is given by:

$$z_{\ell,i}(x) = \pi_i \left(W^\ell(F_{(\ell-1)}(x)) + b^\ell \right), \tag{3}$$

where $\pi_i : \mathbb{R}^{n_\ell} \to \mathbb{R}$ denotes the projection onto the ith component.

Given a point $x \in \mathbb{R}^{n_0}$ in the input space we can record its activation status with respect to all $N = \sum_{i=1}^{d-1} n_i$ hidden neurons by computing the N–tuple $s(x) = \{-1, 0, +1\}^N$ of pre-activation signs for neurons in the network. Explicitly, the component of $s(x)$ corresponding to the ith neuron in the ℓth layer is: $s_{\ell,i}(x) := \text{sgn}(z_{\ell,i}(x))$, where $\text{sgn}(z) = \begin{cases} \frac{z}{|z|} & \text{if } z \neq 0 \\ 0 & \text{if } z = 0. \end{cases}$

In the present work, we will also be interested in the activation pattern of each *neuron* in the network with respect to a finite data set $X = \{x_1, \ldots, x_m\}$. For neuron i in layer ℓ this is the m–tuple $(s_{\ell,i}(x_1), \ldots, s_{\ell,i}(x_m))$. If $s_{\ell,i}(x_j) = +1$ (resp., $= -1$) for all $x_j \in X$ we say that neuron i in layer ℓ is *always-active* (resp., *always-inactive*) on the data set X.

3.2 Spectrum of the NTK for ReLU networks

Recall that a *kernel* $k : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}_{\geq 0}$ is a symmetric, positive semi-definite similarity measure on the input space of a parameterized function class, most naturally obtained by pulling back an inner product from a *kernel feature map* $\Phi : \mathbb{R}^{n_0} \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$: $k_\Phi(x, y) := \langle \Phi(x), \Phi(y) \rangle$.

[1] We use nonstandard notation for the architecture to emphasize that the activation function for the final layer is the identity, Id.

In the case of the neural tangent kernel (NTK) associated to a parameter $\theta \in \Omega$ at initialization, $\mathcal{H}$ is the tangent space $T_\theta(\Omega) \cong \mathbb{R}^D$ at that parameter, equipped with its standard inner product, and the feature map $\Phi : \mathbb{R}^{n_0} \to \mathcal{H}$ is the assignment of the parameter gradient vector $\nabla E_z|_\theta$ of the evaluation map at each input vector $z \in \mathbb{R}^{n_0}$. Note that for ReLU network classes, this assignment is only well-defined away from a Lebesgue measure 0 set, cf. Grigsby & Lindsey (2022); Grigsby et al. (2022). Mercer's Theorem (cf. Schölkopf & Smola (2002)) associates to any kernel k on a compact set $\chi \subseteq \mathbb{R}^{n_0}$ a natural positive semi-definite integral operator T_k on $L_2(\chi)$, defined by $T_k f(\cdot) := \int_\chi k(\cdot, y) f(y) dy$, whose associated eigenfunctions can be viewed as a preferred orthonormal basis of $L_2(\chi)$ associated to the kernel. The relationship between the eigenbasis and spectrum of the NTK operator, optimization dynamics, and generalization has been widely studied, cf. Murray et al. (2023) and the references therein. It is frequently assumed that at initialization the empirical NTK will have full rank–i.e., will be equal to the number of data points in the overparameterized setting.

3.3 Batch functional dimension and the empirical NTK for ReLU networks

The (batch) functional dimension of a parameter $\theta \in \Omega$ on a finite data set $Z \subseteq \mathbb{R}^{n_0}$ was defined (away from a Lebesgue measure 0 set) for ReLU neural network classes in Grigsby et al. (2022), see also Stock (2023); Bona-Pellissier et al. (2022; 2024). It is the rank of the Jacobian matrix with respect to the parameters of the evaluation map on the batch Z: $\text{rk}(\mathbf{J}E_Z|_\theta)$. When the output dimension is 1, $Z = \{z_1, \dots, z_m\}$, and the parametric dimension is D, $\mathbf{J}E_Z$ is an $m \times D$ matrix whose rows are $\nabla E_{z_i}|_\theta$, the neural tangent kernel feature maps at the m points of Z.

In Section 6 of Grigsby et al. (2022) it is noted that the Gram matrix of the NTK at θ on a batch Z is $(\mathbf{J}E_Z)(\mathbf{J}E_Z)^T$. Moreover, it is a well-known linear algebra fact that for all matrices M over $\mathbb{R}$:

$$\text{rk}(M) = \text{rk}(MM^T),$$

so the rank of the Gram matrix, $(\mathbf{J}E_Z)(\mathbf{J}E_Z)^T$, of the NTK at θ on a batch Z is precisely the batch functional dimension of θ on the batch Z.

4 Parameter space symmetries, functional dimension, and the geometry of the loss landscape

Following Serra et al. (2020); Zhao et al.; Godfrey et al. (2022), we call two neural network functions[2] $F_i : \mathbb{R}^{n_0} \to \mathbb{R}^{n_d}$ for $i = 1, 2$ *equivalent* if $F_1(x) = F_2(x)$ for all $x \in \mathbb{R}^{n_0}$. We will also be interested in data-dependent equivalence for a proper subset $X \subsetneq \mathbb{R}^{n_0}$. In this case, we say F_i are equivalent for $i = 1, 2$ if $F_1(x) = F_2(x)$ for all $x \in X$. In the literature (cf. (Zhao et al.; Godfrey et al., 2022)), we typically see equivalences arising within a single architecture as a result of parameter space symmetries.

A loss function,

$$\mathcal{L} : \Omega \times (\mathbb{R}^{n_0} \times \mathbb{R}^{n_d}) \to \mathbb{R}, \quad \mathcal{L}(\theta, (x, y)) := \text{error}(F_\theta(x), y)$$

measures the error of a predicted output, $F_\theta(x)$, with respect to y, the given label on x, and the empirical loss function $\mathcal{L}_Z : \Omega \to \mathbb{R}$ associated to a finite training set Z is the average loss on the finite set, as a function of the parameter θ. The goal of a standard training algorithm is to find a *global minimizer* of $\mathcal{L}_Z$. Moreover, if θ_{min} is a global minimizer, then it is immediate that any $F_{\theta'}$ equivalent to $F_{\theta_{min}}$ on Z is also a global minimizer for any loss function.[3] The dimension of the space of global minimizers plays a crucial role in the dynamics of optimization algorithms at the end stage of training (Damian et al., 2021) and (per the preceding discussion) is bounded below by the dimension of the space of data-dependent function *equivalences* in the architecture.

In Grigsby et al. (2022; 2023); Zhao et al.; Grigsby & Lindsey (2024) it is noted that in favorable situations the local dimension of *parameter space symmetries* is complementary to functional dimension. Explicitly (but informally), for a parameter θ: if D is the total parametric dimension, d_θ is the

[2] We do *not* assume that the functions are associated to networks of the same architecture, but they necessarily have the same input and output dimensions.

[3] Indeed, this is true for any t–level set $\mathcal{L}_Z^{-1}(t) := \{\theta \in \Omega \mid \mathcal{L}_Z(\theta) = t\}$ for $t \in \mathbb{R}$.

functional dimension at θ and s_θ is the dimension of the space of parameter space symmetries near θ, then $D = d_\theta + s_\theta$. The picture to have in mind is that parameter space is decomposed locally into directions that can change the function (hence contribute to functional dimension) and directions that preserve the function (hence contribute to the local space of symmetries).

In Section 4.1, we recall one rich source of parameter space symmetries, coming from the action of the *hidden symmetry group* on parameter space defined in Zhao et al.. In Section 4.2 we describe a mechanism, first described in Grigsby et al. (2023), whereby many architectures can develop a larger set of data-dependent parameter space symmetries than expected, leading to higher-dimensional-than-expected level sets in the loss landscape.

4.1 Lie Group Actions, Orbits, and Hidden Equivalences

Following Zhao et al., we note that the parameter space of any feedforward network architecture $(n_0, \dots, n_{d-1}|n_d)$ (for any choice of activation function) admits a natural action of the *hidden symmetry group*, $G_{hid} := GL_{n_1} \times \dots \times GL_{n_{d-1}}$, of the architecture. Here, GL_n denotes the general linear group of invertible $n \times n$ matrices over $\mathbb{R}$, so any $g = (g_1, \dots, g_{d-1}) \in G_{hid}$ gives rise to a map $g \cdot - : \Omega \to \Omega$ defined as follows. If $\theta = (W^1, b^1, \dots, W^d, b^d) \in \Omega$, then:

$$g \cdot (W^\ell, b^\ell) := (g_\ell W^\ell g_{\ell-1}^{-1}, g_\ell b^\ell), \tag{4}$$

where we set $g_0 := \mathrm{Id}_{n_0}, g_d := \mathrm{Id}_{n_d}$, the identity matrices of the appropriate dimensions.

G_{hid} is an example of a *Lie group*, and the assignment of a smooth map on a vector space associated to every element of a Lie group is an example of a *Lie group action*. One can and should understand this particular action as induced by a much more natural action of the Lie group G_{hid} on the product of the hidden layers, $\mathbb{R}^{n_1} \times \dots \times \mathbb{R}^{n_{d-1}}$, by layer-wise conjugation (change-of-basis).

In particular, if H is a subgroup of G_{hid} whose action commutes with the component-wise application of the activation function, then we can decompose parameter space into *orbits* under the action of H, and each *H–orbit*, $H\theta := \{h \cdot \theta \mid h \in H\}$, will consist of equivalent parameters; see Zhao et al. and Section A in the Appendix for more details. Moreover, the classical orbit-stabilizer theorem for Lie group actions (Corollary A.6, cf. Lee (2013)) then tells us that $H\theta$ is diffeomorphic to the quotient space, $H/\mathrm{Stab}_H(\theta)$, where

$$\mathrm{Stab}_H(\theta) := \{h \in H \mid h \cdot \theta = \theta\}$$

is the *stabilizer* of θ. In particular, the dimension of the orbit $H\theta$ is the dimension of the Lie group H minus the dimension of the stabilizer of any parameter θ in the orbit:

$$\dim(H\theta) = \dim(H) - \dim(\mathrm{Stab}_H(\theta)). \tag{5}$$

The picture and framework that emerges, described beautifully in **?** (see also for an algebro-geometric view on this framework), is that parameter space decomposes into orbits under the action of a high-dimensional data-dependent symmetry group. Understanding the details of this parameter symmetry group action - in particular, how the orbits interact in parameter space - will illuminate the optimization dynamics of neural network models.

Definition 4.1. *We define the* hidden equivalence group*, $H_{eq} \subseteq G_{hid}$ for a fixed architecture and batch $X \subseteq \mathbb{R}^{n_0}$ of input data to be the largest subgroup of the hidden symmetry group that commutes with the component-wise application of the activation function for all $x \in X$.*

It is immediate that all functions in any H_{eq}-orbit of any parameter $\theta \in \Omega$ are equivalent on X. In particular, the dimension of H_{eq} will give us a lower bound on the dimension of the level sets of the empirical loss function (as long as the stabilizer is trivial on parameters in the orbit). It follows that if the hidden equivalence group is larger than expected, then the the critical locus will also be larger than expected (in the trivial stabilizer case):

Lemma 4.2. *Let $\mathcal{C} \subseteq \Omega$ be the critical locus for any empirical loss function $\mathcal{L}_X : \Omega \to \mathbb{R}$. If there exists $\theta \in \mathcal{C}$ with trivial stabilizer (for which $Stab_{H_{eq}}(\theta) = \{Id\}$), then $dim(\mathcal{C}) \geq dim(H_{eq})$.*

Remark 4.3. The *expected* dimension of the hidden equivalence group for a ReLU network of architecture $(n_0, n_1, \dots, n_{d-1}|n_d)$ is $N = \sum_{\ell=1}^{d-1} n_\ell$, the number of hidden neurons. Indeed, letting $PD_+(n)$ denote the group of $n \times n$ matrices representable as a product PD, where P is an $n \times n$ permutation matrix and D is an $n \times n$ diagonal matrix with strictly positive entries on the diagonal,

it is well-known that the subgroup $PD_+(n_1) \times \ldots \times PD_+(n_{d-1})$ of the hidden symmetry group commutes with the component-wise application of ReLU (Zhao et al.; Godfrey et al., 2022), resulting in the familiar *positive scaling invariance* and *permutation invariance* for hidden neurons in ReLU networks (Rolnick & Kording, 2020; Bui Thi Mai & Lampert, 2020; Grigsby et al., 2023).

Lemma 4.2, which follows from the orbit-stabilizer theorem and the results in Appendix A, tells us that if the hidden equivalence group *on a batch of data* X includes a subgroup of dimension larger than N, then level sets of the loss will be larger than the expected dimension N, decreasing the functional dimension on the batch. In the following section, we describe one way this can occur.

4.2 Restricted activation patterns induce hidden equivalences

If there are neurons for F_θ that are either always-active or always-inactive on a data set $X \subseteq \mathbb{R}^{n_0}$, then ReLU acts locally as the Identity function at always-active neurons and effectively ignores always-inactive neurons. This observation motivates the following, cf. Serra et al. (2020):

Definition 4.4. *Let $\sigma_{[1:k]} : \mathbb{R}^n \to \mathbb{R}^n$ denote the function that applies the activation function ReLU(x) (resp., $Id(x)$) to the first k components (resp., to the last $n-k$ components). A* mixed ReLU-linear *neural network of architecture $(n_0, (n_1^R, n_1^L), \ldots, (n_{d-1}^R, n_{d-1}^L) | n_d)$ is a neural network of architecture $(n_0, n_1^R + n_1^I, \ldots, n_{d-1}^R + n_1^L \mid n_d)$ whose ℓth layer map is $F^\ell(x) := \sigma_{[1:n_\ell^R]}(W^\ell x + b^\ell)$.*

The following results, whose formal statements and proofs appear in Appendix 4.1, together tell us that the dimension of the hidden equivalence group is *quadratic*, rather than *linear* in the number of neurons with fixed activation status with respect to the data set X.

Proposition 4.5. *The hidden equivalence group of a mixed ReLU-linear neural network of architecture $(n_0, n_1^R + n_1^I, \ldots, n_{d-1}^R + n_{d-1}^L \mid n_d)$ has dimension at least $d = \sum_{\ell=0}^{d-1} n_\ell^R + \sum_{\ell=1}^{d-1} (n_\ell^L)^2$.*

Theorem 1. *Let $\theta \in \Omega$ be almost any parameter[4] in a feedforward ReLU network architecture, $(n_0, \ldots, n_{d-1} | n_d)$, with parametric dimension D, and let $X \subseteq \mathbb{R}^{n_0}$ be a subset of the domain for which n_ℓ^{fixed} of the hidden neurons from layer ℓ are either always-active or always-inactive on all of X. Then the batch functional dimension of θ on X is at most $D - d$, where $d = \sum_{\ell=1}^{d-1} \left(n_\ell - n_\ell^{fixed}\right) + \sum_{\ell=1}^{d-1} \left(n_\ell^{fixed}\right)^2$.*

The idea of the proof of Theorem 1 is that near θ, the space of data-dependent parameter space symmetries on X looks locally like the hidden equivalence group of a mixed ReLU-linear network with $n_\ell^R = n - n_\ell^{fixed}$ and $n_\ell^L = n_\ell^{fixed}$.

5 Experimental setup

To empirically investigate how the rank of the empirical NTK evolves during training, we define functions of varying complexity, use those functions to construct toy datasets, and train feedforward ReLU neural networks on those datasets. This section describes the details of this setup.

Functions and data sets: For our toy datasets, we choose univariate (input dimension 1) and bivariate (input dimension 2) functions so we can easily plot their evolution during training.

Univariate: The univariate functions $f_1, f_2, f_3 : [0, 1] \to \mathbb{R}$ are defined as follows:

$$
\begin{aligned}
f_1(x) &= 1.5 + \sin(2\pi x) \\
f_2(x) &= 1.5 + \sin(2\pi x) + 0.5\sin(4\pi x) \\
f_3(x) &= 1.5 + \sin(2\pi x) + 0.5\sin(4\pi x) + 0.4\sin(16\pi x).
\end{aligned}
$$

For each function f_i, we create a data set $\mathcal{D}_i = \{x_j, y_j\}_{j=1}^{100} \subset \mathbb{R} \times \mathbb{R}$ by sampling the domain at 100 equally spaced points. Figure 1 shows the graphs and datasets for the functions f_i.

[4] θ is in the complement of a Lebesgue measure 0 set defined in the proof of Proposition A.8 in the Appendix.

Bivariate: The bivariate functions $g_1, g_2, g_3 : [0,1]^2 \to \mathbb{R}$, defined by

$$g_1(x,y) = 1.5 + \sin(2\pi x) + \sin(2\pi y)$$
$$g_2(x,y) = 1.5 + \sin(2\pi x) + \sin(2\pi y) + 0.5\sin(4\pi x)$$
$$g_3(x,y) = 1.5 + \sin(2\pi x) + \sin(2\pi y) + 0.5\sin(4\pi x) + 0.4\sin(4\pi y).$$

For each bivariate function g_i, we create a dataset $\mathcal{D}_{g_i} = \{x_j, y_j\}_{j=1}^{400} \subset \mathbb{R}^2 \times \mathbb{R}$ by sampling the domain at 400 equally spaced points (a 20×20 grid). Figure 11 in the Appendix shows the graphs and datasets for the bivariate functions g_i.

The purpose of the vertical shift 1.5 in the functions is to make all functions strictly positive. We do not add noise to our data sets.

Training setup: For all experiments, we used the fixed architecture $(\text{input dimension}, 15, 15, 15, 15, 1)$ of fully-connected, feedforward ReLU neural networks. For the univariate experiments, the associated number of trainable parameters is $D = 1006$, and for the bivariate experiments, the parametric dimension is $D = 1021$. We train the networks using the Adam optimizer with MSELoss, learning rate 0.01, weight decay 1e-4 (unless otherwise specified in text), and shuffled minibatches of batchsize 16. We initialize each training run by randomly selecting all weights and biases using the He initialization (also called Kaiming initialization), i.e. all weights and all biases for a given layer are selected randomly from a normal distribution with mean 0 and standard deviation $\sqrt{\frac{2}{\text{fan\_in}}}$, where fan_in is the number of input features for that layer.

Batch functional dimension computation: At specified epochs during training, we compute the batch functional dimension using the inputs for the entire data set $\mathcal{D}_i$ as the batch. To do this, we first compute the matrix

$$\mathbf{J}E_{\mathcal{D}_i}(\theta) \coloneqq \begin{bmatrix} \nabla_\theta f(x_1;\theta) \\ \nabla_\theta f(x_2;\theta) \\ \vdots \\ \nabla_\theta f(x_m;\theta) \end{bmatrix}$$

where each x_j is one of the inputs of a point $(x_j, y_j) \in \mathcal{D}_i$ and $f(\cdot,\theta)$ is the function determined by the parameter θ. We then compute the batch functional dimension as

$$\dim_{\text{ba.fun}}(\theta) = \text{torch.linalg.matrix\_rank}(\mathbf{J}E_{\mathcal{D}_i}(\theta)).$$

Note that the command torch.linalg.matrix_rank computes the number of "non-zero" singular values, where a singular value is considered non-zero if it is greater than the default error tolerance for floating point calculations. This default threshold is defined for a $n \times m$ matrix A as

$$\text{threshold}(A) \coloneqq \max(n,m) \cdot \sigma_{max} \cdot \epsilon$$

where σ_{max} is the largest singular value of A and ϵ represents machine precision, which is roughly $1.19 * 10^{-7}$ for dtype float32. Thus, our computation of batch functional dimension will set to 0 any singular values sufficiently small with respect to the maximum possible rank of the Jacobian matrix (100 for our 1D experiments and 400 for our 2D experiments), and the largest singular value, which empirically were $\approx 10^3$ for our 1D experiments and $\approx 10^4$ for our 2D experiments. The calculation of the rank is therefore setting to 0 singular values below (roughly) 0.01 for our 1D experiments and 0.1 for our 2D experiments (both typically 10^{-5} times σ_{max}).

6 Experimental results

The empirical NTK is typically rank deficient at initialization. This observation is supported by Figures 4, 8, 5, 7, 9, 10, 12, 13, 14.

Weight decay suppresses batch functional dimension of trained networks. See Figure 4.

Evolution of functional dimension before and after approximate function stabilization. Figures 3 and 6 show, for each univariate dataset $\mathcal{D}_i$, the functions found during a training run. Visual investigation of these and similar plots for other random initializations suggests that, in our experimental

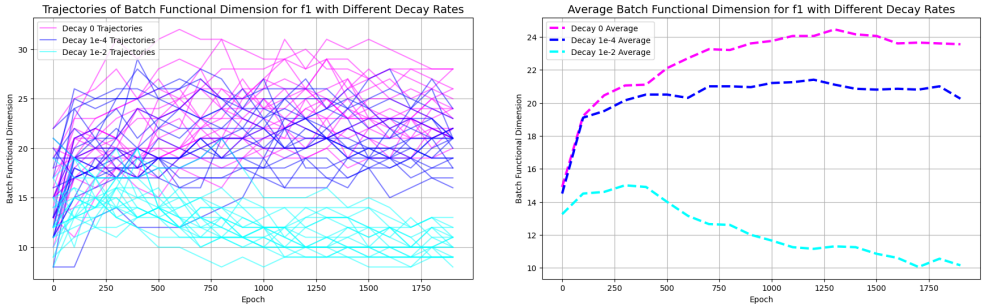

Figure 4: Comparing the impact of the weight decay rate on functional dimension evolution. Left: We track functional dimension (trained on $\mathcal{D}_1$) during 20 randomly initialized training runs for each of 3 weight decay rates: 0, 1e-4, and 1e-2. Right: Averages at each epoch over the 20 training runs per group. (Functional dimension was computed every 100 epochs.)

setup, the functions found by training on $\mathcal{D}_1$ typically appear to be approximately stable by epoch 100, on $\mathcal{D}_2$ by epoch 1000-1500, and on $\mathcal{D}_3$ by epoch 2000-4000. Continuing to train beyond the point when the function approximately stabilizes does not appear to lead to overfitting (Figure 6 in the Appendix visualizes function evolution over 3000 epochs for f_1.)

Figure 8 depicts trajectories of functional dimension evolution across training epochs until approximate functional stabilization for multiple random initializations. The trend shown in each of the plots in Figure 8 is that, on average, functional dimension increases until the epoch at which we stopped training (chosen to correspond to roughly when the functions tend to stabilize).

Batch functional dimension tracks dataset complexity. Figure 2 supports the conclusion that for the univariate datasets, on average, training tends to select a function whose complexity, as measured by functional dimension, reflects the complexity of the data set. That is, the rank of the empirical NTK after 3000 epochs of training correlates positively with the complexity of the function the model is learning. Figure 5 demonstrates same trend holds for the bivariate datasets.

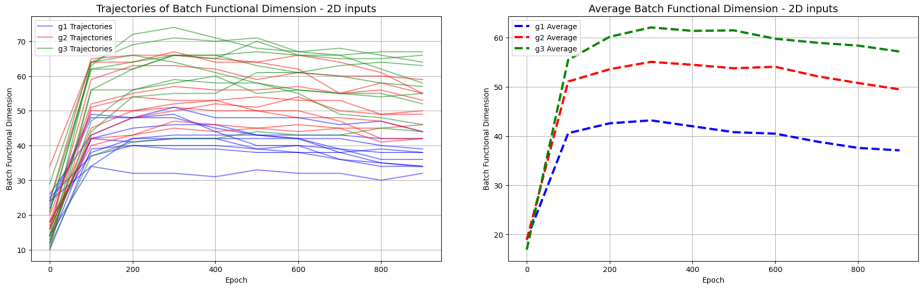

Figure 5: Left: Trajectories of batch functional dimension over 10 randomly initialized training runs for each bivariate dataset (using weight decay 0, functional dimension computed every 100 epochs). Right: Averages. (All trained functions had r^2 scores ≥ 0.95.)

Batch functional dimension is correlated with number of regions sampled. See Figures 9, 10. Proposition C.3 in Appendix C proves that, except for a measure 0 situation, the functional dimension is bounded by the number of linear regions sampled by the dataset times $(n_0 + 1)$, where n_0 is the input dimension.

7 Conclusions

We have performed both a theoretical and empirical investigation of the behavior of the empirical NTK during training of deep ReLU networks on toy datasets, assembling empirical evidence that the rank of the empirical NTK in this setting is (i) lower-than-expected at initialization, and (ii) tracks task complexity. We provide a possible theoretical explanation for this phenomenon by relating the rank of the empirical NTK to the functional dimension, whose behavior for ReLU networks has been related to a growing body of literature on parameter space symmetries.

REFERENCES

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 242–252. PMLR, 2019. URL http://proceedings.mlr.press/v97/allen-zhu19a.html.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 322–332. PMLR, 2019. URL http://proceedings.mlr.press/v97/arora19a.html.

Joachim Bona-Pellissier, Francois Malgouyres, and Francois Bachoc. Local identifiability of deep ReLU neural networks: The theory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b0ae046e198a5e43141519868a959c74-Abstract-Conference.html.

Joachim Bona-Pellissier, Fran cois Malgouyres, and Fran cois Bachoc. Geometry-induced implicit regularization in deep ReLU neural networks. *Preprint arXiv:2402.08269*, 2024.

Phuong Bui Thi Mai and Christoph Lampert. Functional vs. parametric equivalence of ReLU networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta (eds.), *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pp. 342–350. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/hash/5751ec3e9a4feab575962e78e006250d-Abstract.html.

Alex Damian, Tengyu Ma, and Jason D. Lee. Label noise SGD provably prefers flat global minimizers. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 27449–27461, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e6af401c28c1790eaef7d55c92ab6ab6-Abstract.html.

Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Matthieu Wyart, and Clément Hongler. Disentangling feature and lazy training in deep neural networks. *arXiv preprint arXiv:1906.08034*, 2019.

Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *NeurIPS 2022, New Orleans, LA*, 2022.

J. Elisenda Grigsby and Kathryn Lindsey. On transversality of bent hyperplane arrangements and the topological expressiveness of ReLU neural networks. *SIAM Journal on Applied Algebra and Geometry*, 6(2):216–242, 2022.

J. Elisenda Grigsby and Kathryn Lindsey. On functional dimension and persistent pseudodimension. *Preprint arXiv:2410.17191*, 2024.

J. Elisenda Grigsby, Kathryn Lindsey, Robert Meyerhoff, and Chenxi Wu. Functional dimension of feedforward ReLU neural networks. *Preprint arXiv:2209.04036*, 2022.

J. Elisenda Grigsby, Kathryn Lindsey, and David Rolnick. Hidden symmetries of ReLU networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML*

2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of *Proceedings of Machine Learning Research*, pp. 11734–11760. PMLR, 2023. URL https://proceedings.mlr.press/v202/grigsby23a.html.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJgndT4KwB.

Boris Hanin and David Rolnick. Deep ReLU networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4542–4551. PMLR, 2020. URL http://proceedings.mlr.press/v119/huang20l.html.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. 2018.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8570–8581, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0d1a9651497a38d8b1c3871c84528bd4-Abstract.html.

Jaehoon Lee, Samuel S. Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/ad086f59924fffe0773f8d0ca22ea712-Abstract.html.

John M. Lee. *Introduction to smooth manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer, New York, second edition, 2013. ISBN 978-1-4419-9981-8.

Giovanni Luca Marchetti, Vahid Shahverdi, Stefano Mereta, Matthew Trager, and Kathlén Kohn. An invitation to neuroalgebraic geometry. *CoRR*, abs/2501.18915, 2025. doi: 10.48550/ARXIV.2501.18915. URL https://doi.org/10.48550/arXiv.2501.18915.

Marissa Masden. Algorithmic determination of the combinatorial structure of the linear regions of ReLU neural networks. *Preprint arXiv:2207.07696*, 2022.

Michael Murray, Hui Jin, Benjamin Bowman, and Guido Montúfar. Characterizing the spectrum of the NTK via a power series expansion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=Tvms8xrZHyR.

Quynh Nguyen and Marco Mondelli. Global convergence of deep networks with one wide layer followed by pyramidal topology. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/8abfe8ac9ec214d68541fcb888c0b4c3-Abstract.html.

Quynh Nguyen, Marco Mondelli, and Guido F. Montúfar. Tight bounds on the smallest eigenvalue of the neural tangent kernel for deep relu networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8119–8129. PMLR, 2021. URL http://proceedings.mlr.press/v139/nguyen21g.html.

David Rolnick and Konrad P. Kording. Reverse-engineering deep ReLU networks. In *International Conference on Machine Learning (ICML)*, 2020.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

Thiago Serra, Abhinav Kumar, and Srikumar Ramalingam. Lossless compression of deep neural networks. In Emmanuel Hebrard and Nysret Musliu (eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21-24, 2020, Proceedings*, volume 12296 of *Lecture Notes in Computer Science*, pp. 417–430. Springer, 2020. doi: 10.1007/978-3-030-58942-4_27. URL https://doi.org/10.1007/978-3-030-58942-4_27.

Gribonval Rémi Stock, Pierre. An embedding of ReLU networks and an analysis of their identifiability. *Constructive Approximation*, 57:853–899, 2023.

Bo Xie, Yingyu Liang, and Le Song. Diverse neural network learns true target functions. In Aarti Singh and Xiaojin (Jerry) Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1216–1224. PMLR, 2017. URL http://proceedings.mlr.press/v54/xie17a.html.

Bo Zhao, Iordan Ganev, Robin Walters, Rose Yu, and Nima Dehmamy. Symmetries, flat minima, and the conserved quantities of gradient flow. In *ICLR 2023, Kigali, Rwanda*. OpenReview.net.

Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *CoRR*, abs/2506.13018, 2025. doi: 10.48550/ARXIV.2506.13018. URL https://doi.org/10.48550/arXiv.2506.13018.

A Lie group actions, homogeneous spaces, orbits, and stabilizers

We recall some classical results about Lie group actions on smooth manifolds, following the treatment in Lee (2013).

Definition A.1. *A* Lie group *is a smooth manifold G that is also a group, for which the group operations are all smooth maps. That is, G is endowed with a multiplication map*

$$m : G \times G \to G \qquad\qquad m(g, h) = gh$$

and an inversion map

$$i : G \to G \qquad\qquad i(g) = g^{-1},$$

and both m and i are smooth (derivatives of all orders are well-defined).

Definition A.2. *If G is a Lie group with identity element Id, and Ω is a smooth manifold, a smooth left action of G on Ω is a map*

$$\psi : G \times \Omega \to \Omega \qquad\qquad (g, \theta) \mapsto g \cdot \theta$$

satisfying:

- *ψ is a smooth map.*
- *$g_1 \cdot (g_2 \cdot \theta) = (g_1 g_2) \cdot \theta$ for all $g_1, g_2 \in G, \theta \in \Omega$,*
- *$Id \cdot \theta = \theta$ for all $\theta \in \Omega$.*

Definition A.3. *Let G be a Lie group acting smoothly on a smooth manifold Ω.*

- *The action of G on Ω is said to be* transitive *if for all $\theta, \theta' \in \Omega$ there exists $g \in G$ such that $g \cdot \theta = \theta'$.*
- *For $\theta \in \Omega$, the* orbit *of θ under the action of G is the set of points in Ω obtainable by applying an element $g \in G$ to θ:*

$$G\theta := \{g \cdot \theta \mid g \in G\}.$$

- *For $\theta \in \Omega$, the* stabilizer *of θ is the set*

$$Stab_G(\theta) := \{g \in G \mid g \cdot \theta = \theta\}.$$

Definition A.4. *A* homogeneous G–space *is a smooth manifold M equipped with a* transitive *action of a Lie group G.*

It is immediate from the definitions that if a Lie group G acts smoothly on a smooth manifold Ω, then for every $\theta \in \Omega$, every G–orbit $G\theta$ is a homogeneous space.[5]

Proofs of the following results can be found in the section on *Homogeneous spaces* in Chapter 9 of Lee (2013), which mostly rely on the equivariant rank theorem for Lie groups (Theorem 9.7 in Lee (2013)).

Lemma A.5. *(Lemma 9.23 of Lee (2013)) If G is a Lie group acting smoothly on Ω, $Stab_G(\theta)$ is a closed Lie subgroup of G for every $\theta \in \Omega$.*

Theorem 2. *(Theorem 9.22 of Lee (2013) Let G be a Lie group, and let $H \subseteq G$ be a closed Lie subgroup of G. The left coset space G/H has a unique smooth manifold structure such that the quotient map $\pi : G \to G/H$ is a smooth submersion. Moreover, G/H is also a homogeneous G–space with respect to the natural G–action on the quotient group.*

Theorem 3. *(Theorem 9.24 of Lee (2013) Let G be a Lie group and M a homogeneous G–space. Then the coset space (quotient space) $G/Stab_G(\theta)$ is diffeomorphic to M.*

Corollary A.6. *(Orbit-stabilizer theorem) Let G be a Lie group acting smoothly on a manifold Ω, and let $G\theta$ be the G–orbit of $\theta \in \Omega$. Then $G/Stab_G(\theta)$ is diffeomorphic to $G\theta$. In particular,*

$$dim(G\theta) = dim(G) - dim(Stab_G(\theta).$$

The following lemma is immediate from the definitions.

Lemma A.7. *Let X be a smooth manifold, G a Lie group acting on X, and $H \subseteq G$ a Lie subgroup of G. For $\theta \in X$, if $Stab_G(\theta)$ is trivial, then so is $Stab_H(\theta)$.*

Proof. $\text{Stab}_H(\theta)$ is by definition a subgroup of $\text{Stab}_G(\theta)$, so if $\text{Stab}_G(\theta)$ is the trivial subgroup, then so is $\text{Stab}_H(\theta)$. □

We will sometimes need the following architecture restriction in order to deduce properties of the functional dimension from our knowledge of a particular Lie group action:

Assumption A.1. *We say that an architecture $(n_0, \ldots, n_{d-1} \mid n_d)$ is* roughly monotonic *if:*

(i) For all $\ell \in \{1, \ldots, d\}$, we have $n_\ell - 1 \leq n_{\ell-1}$, or

(ii) For all $\ell \in \{1, \ldots, d\}$, we have $n_{\ell-1} - 1 \leq n_\ell$.

In other words, the dimensions of the layers either grow or shrink (up to a linear error) through the network. In practice, architectures are typically roughly rectangular, so this restriction is mild.

Proposition A.8. *Let Ω be the parameter space for a ReLU neural network of architecture $(n_0, n_1, \ldots, n_{d-1} \mid n_d)$. If Assumption A.1 holds, then for almost all $\theta \in \Omega$, $Stab_{G_{hid}}(\theta)$ is trivial.*

In other words, as long as the dimensions of the layers don't shrink too fast (condition (i)) or grow too fast (condition (ii)) as you move through the network, every parameter away from a Lebesgue measure 0 set in parameter space has trivial stabilizer under the action of the hidden symmetry group.

Proof. Recall (Equation 4) that

$$g = (g_1, \ldots, g_d) \in G_{hid} := GL_{n_1} \times \ldots \times GL_{n_{d-1}}$$

acts on $\theta = (W^1, b^1, \ldots, W^d, b^d) \in \Omega$ by:

$$(W^\ell, b^\ell) \quad \to \quad (g_\ell W^\ell g_{\ell-1}^{-1}, g_\ell b^\ell),$$

[5] Beware that this does *not* imply that every G–orbit is a *smoothly imbedded submanifold* of Ω. See the examples in the section on Proper Actions in Chapter 9 of Lee (2013).

so if g is in the stabilizer of θ, then (recalling that for convenience we set $g_0 = g_d = \text{Id}$) we have:

$$\begin{array}{rcll} g_1 W^1 & = & W^1 & g_1 b^1 = b^1 \\ g_2 W^2 g_1^{-1} & = & W^2 & g_2 b^2 = b^2 \\ & \cdots & & \cdots \\ g_{d-1} W^{d-1} g_{d-2}^{-1} & = & W^{d-1} & g_{d-1} b^{d-1} = b^{d-1} \\ W^d g_{d-1}^{-1} & = & W^d & \end{array}$$

Now note that by the first line of equations above, 1 must be an eigenvalue of g_1, and the dimension of the eigenspace of 1 must be at least the the rank of the matrix, $(W^1\ b^1)$, whose columns are all in the in the eigenspace of 1 for g_1. But if we assume that $g_1 \neq \text{Id}$, then by the fact that each square matrix has a unique Jordan canonical form (up to reordering the blocks), and the number of Jordan blocks associated to each eigenvalue is equal to the dimension of the eigenspace for that eigenvalue, the dimension of the eigenspace of 1 is bounded above by $n_1 - 1$. Since generically (away from a Lebesgue measure 0 set) $(W^1\ b^1)$ has rank = $\min\{n_1, n_0 + 1\}$, we conclude that $n_0 + 1 \leq n_1 - 1$. But if the dimensions of the layers satisfy condition (i), then we have $n_0 \geq n_1 - 1$, so $g_1 = \text{Id}$. Applying this logic to each equation in turn, working from the top to the bottom in the list of equations above and using condition (i), we conclude that $g_\ell = \text{Id}$ for all ℓ and hence the stabilizer of a generic θ is trivial.

We arrive at the same conclusion by applying the same reasoning to the transpose of g_ℓ, working from the final equation to the first. In this case, we will need to restrict to symmetric matrices g_ℓ in order for it to be possible for the rows of a generic W^ℓ to be in the 1-eigenspace of $(g_\ell)^T$ and simultaneously have a generic b^ℓ be in the 1–eigenspace of g^ℓ. But if we restrict to symmetric g_ℓ (that is, $g_\ell = g_\ell^T$) and use assumption (ii), then the same argument as in the previous paragraph tells us that for generic θ, $g_\ell = \text{Id}$ for all ℓ. □

Definition A.9. *For $0 \leq k \leq n$ let $\sigma_{[1:k]} : \mathbb{R}^n \to \mathbb{R}^n$ denote the function that applies the activation function ReLU(x) (resp., $Id(x)$) to the first k components (resp., to the last $n - k$ components).*

A mixed ReLU-linear *neural network of architecture $(n_0, (n_1^R, n_1^L), \ldots, (n_{d-1}^R, n_{d-1}^L)|n_d)$ is a neural network of architecture $(n_0, n_1^R + n_1^I, \ldots, n_{d-1}^R + n_1^L \mid n_d)$ whose ℓth layer map is*

$$F^\ell(x) := \begin{cases} \sigma_{[1:n_\ell^R]}(W^\ell x + b^\ell) & \text{for } 1 \leq \ell < d \\ W^\ell x + b^\ell & \text{for } \ell = d. \end{cases} \tag{6}$$

Proposition A.10. *Given any parameter $\theta \in \Omega$ in a mixed ReLU-linear neural network of architecture*

$$(n_0, n_1^R + n_1^I, \ldots, n_{d-1}^R + n_{d-1}^L \mid n_d),$$

all parameters in the H-orbit of θ for the hidden symmetry subgroup

$$H := \{D_+(n_1^R) \times GL(n_1^L)\} \times \ldots \times \{D_+(n_{d-1}^R) \times GL(n_{d-1}^L)\}$$

are equivalent to θ. That is:

$$F_{h\theta}(x) = F_\theta(x) \ \ \forall\ x \in \mathbb{R}^{n_0}, h \in H.$$

Moreover, if the architecture is roughly monotonic (Definition A.1), then for almost all parameters θ,

$$dim(H\theta) = dim(H) = \sum_{\ell=0}^{d-1} n_\ell^R + \sum_{\ell=1}^{d-1} (n_\ell^L)^2.$$

Proof. Let $D_+(n_\ell^R) \times GL(n_\ell^L) \subseteq GL(n_\ell)$ be the subgroup of $GL(n_\ell)$ consisting of 2-block matrices with an $n_\ell^R \times n_\ell^R$ upper-left block of diagonal matrices with positive entries on the diagonal, and an $n_\ell^L \times n_\ell^L$ lower-right block of invertible matrices. The subgroup

$$H = \{D_+(n_1^R) \times GL(n_1^L)\} \times \ldots \times \{D_+(n_{d-1}^R) \times GL(n_{d-1}^L)\}$$

acts on $\theta \in \Omega$ as in Equation 4. dand since $D_+(n_\ell^R)$ commutes with component-wise application of ReLU on the first n_ℓ^R neurons, and $GL(n_\ell^L)$ commutes with the component-wise application of

the Id activation function on the last n_ℓ^L neurons, the action of H on Ω leaves the overall function invariant. Since the architecture is roughly monotonic, for almost all parameters $\theta \in \Omega$, the stabilizer of the H action is trivial by Lemma A.7 and Proposition A.8, so the orbit-stabilizer theorem tells us:

$$\dim(H\theta) = \dim(H) - 0 = \sum_{\ell=1}^{d-1} n_\ell^R + \sum_{\ell=1}^{d-1} (n_\ell^L)^2.$$

□

The following theorem says that if there are any neurons for a parameter θ that have fixed activation status (always-active or always-inactive) on an entire batch X of data, then the dimension of the space of local data-dependent parameter space symmetries matches the dimension of the hidden equivalence group for a mixed ReLU-linear network where all of the fixed-status neurons are treated as linear.

Theorem 1. *Let $\theta \in \Omega$ be almost any parameter[6] in a feedforward ReLU network architecture, $(n_0, \ldots, n_{d-1}|n_d)$, with parametric dimension D, and let $X \subseteq \mathbb{R}^{n_0}$ be a subset of the domain for which n_ℓ^{fixed} of the hidden neurons from layer ℓ are either always-active or always-inactive on all of X. Then the batch functional dimension of θ on X is at most $D - d$, where $d = \sum_{\ell=1}^{d-1} \left(n_\ell - n_\ell^{fixed}\right) + \sum_{\ell=1}^{d-1} \left(n_\ell^{fixed}\right)^2$.*

Proof of Theorem 1. By the well-known description of the piecewise polynomial structure of ReLU network functions (cf. the appendices of Hanin & Rolnick (2019),Grigsby et al. (2023) and Section 2.6 of Grigsby & Lindsey (2024)), on a neighborhood of each point of $X = \{x_1, \ldots, x_m\}$, F_θ is a polynomial function in the parameters θ, realized as a sum of monomials determined by the open paths at x_i in the computational graph for the architecture. It follows that if we delete the stably inactive neurons and replace the ReLU activation function with the Id function on the stably active neurons of the computational graph, the parameterized ReLU neural network class looks locally on X like a mixed ReLU-linear network whose ReLU neurons in layer ℓ are precisely those neurons whose activation status is *not* fixed on X. The neurons in layer ℓ whose activation status on X is *active on all of* X look locally like linear neurons (i.e., we can replace ReLU with the Id activation function) and the neurons in layer ℓ whose activation status is *inactive on all of* X can be deleted from the network. Explicitly, if we list the $k_\ell := k_{\ell,+} + k_{\ell,-}$ stably active and inactive neurons on X last in the ℓth layer, then we can replace the component-wise application of ReLU on the last k_ℓ neurons with the component-wise application of Id without impacting the overall function F_θ. Accordingly, the action of the subgroup, $H := \{(D_+(n_1 - k_1) \times GL(k_1)\} \times \ldots \times \{D_+(n_{d-1}) \times GL(k_{d-1})\}$, commutes with this mixed activation function. So H_{eq} contains H, and $\dim(H) = \sum_{\ell=1}^{d-1}(n_\ell - k_\ell) + \sum_{\ell=1}^{d-1} k_\ell^2$. The statement about the functional dimension is then an immediate consequence of Corollary 4.3 of Grigsby & Lindsey (2024) and the orbit-stabilizer theorem, once we note that there exists a non-empty open ball around θ on which an open subset of H acts with nontrivial stabilizer. □

B Additional plots and experimental results

B.1 Univariate datasets

[6] θ is in the complement of a Lebesgue measure 0 set defined in the proof of Proposition A.8 in the Appendix.

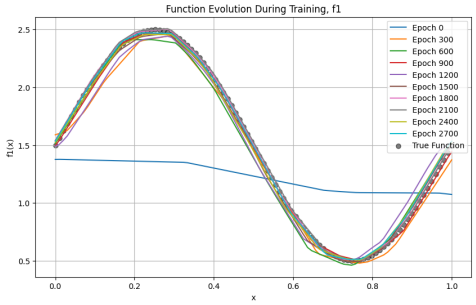

Figure 6: Training after approximate stabilization of the function does not show evidence of overfitting. This plot shows the evolution of functions found in a single training run for f_1 over 3000 epochs.

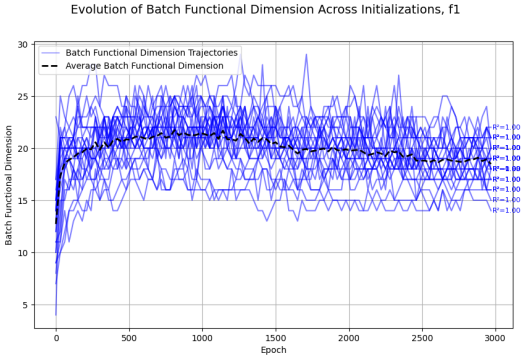

Figure 7: Functional dimension trajectories for 25 randomly initialized training runs over 3000 epochs using the $f1$ dataset.

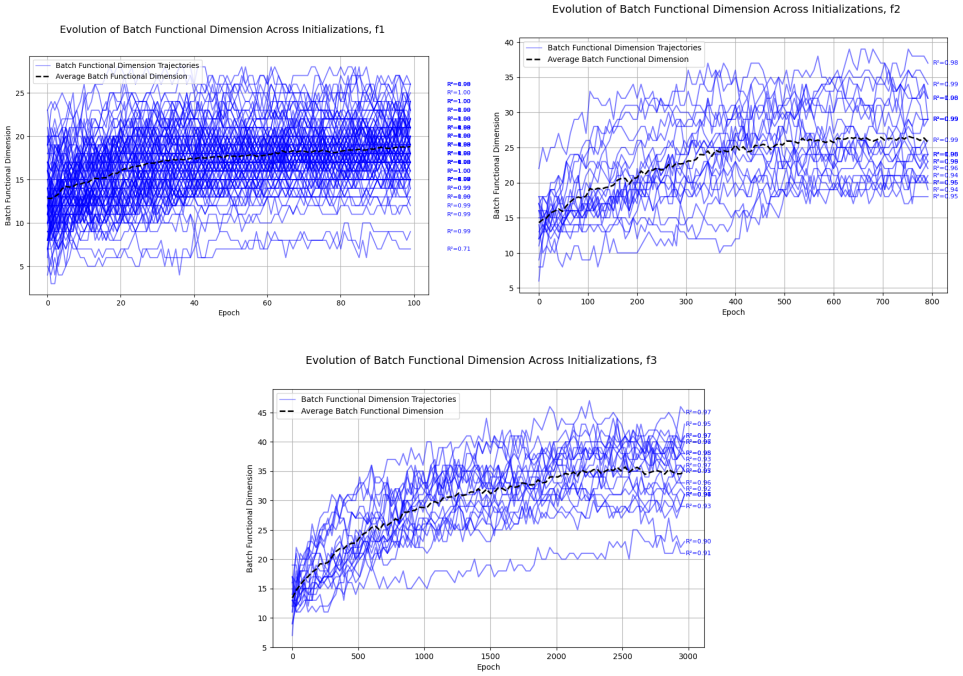

Figure 8: Evolution of functional dimension for multiple, randomly-initialized training runs until approximate stabilization. The black dotted lines indicate the average over the "successful" training runs – which we define as those with coefficient of correlation $r^2 \geq 0.9$ at the end of training. (The r^2 score is recorded to the right of each trajectory.) For $f1$ we computed the trajectory for 100 randomly initialized training runs; for each of $f2$ and $f3$ we computed 20 training run trajectories.)

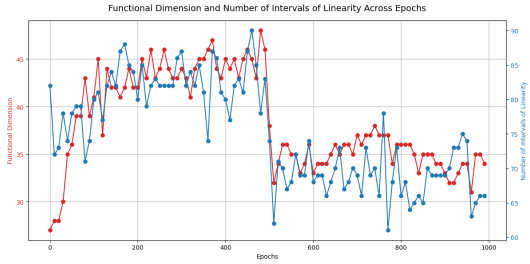

Figure 9: Tracking functional dimension and number of linear regions sampled every 100 epochs during a single training run for $f1$.

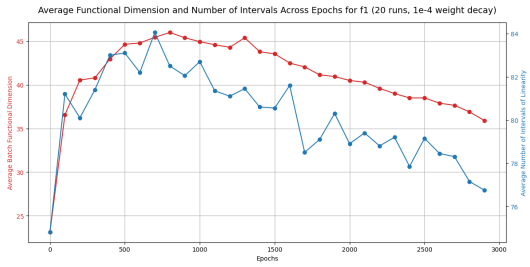

Figure 10: Averages over 20 randomly initialized training runs for $f1$ of number of linear regions sampled and functional dimension.

B.2 BIVARIATE DATASETS

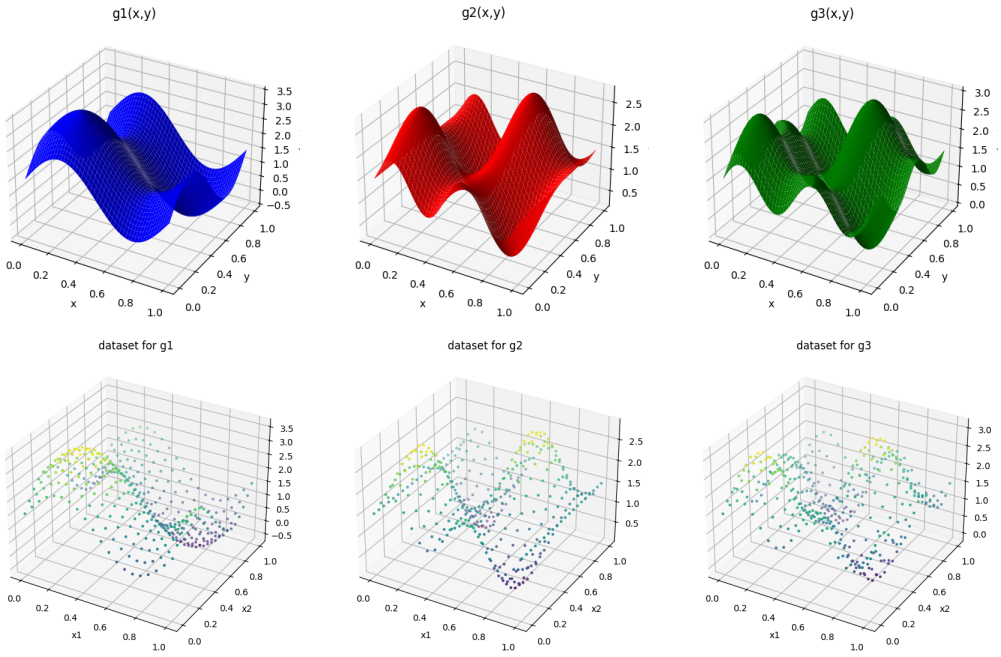

Figure 11: Graphs and datasets for the bivariate functions g_1, g_2 and g_3.

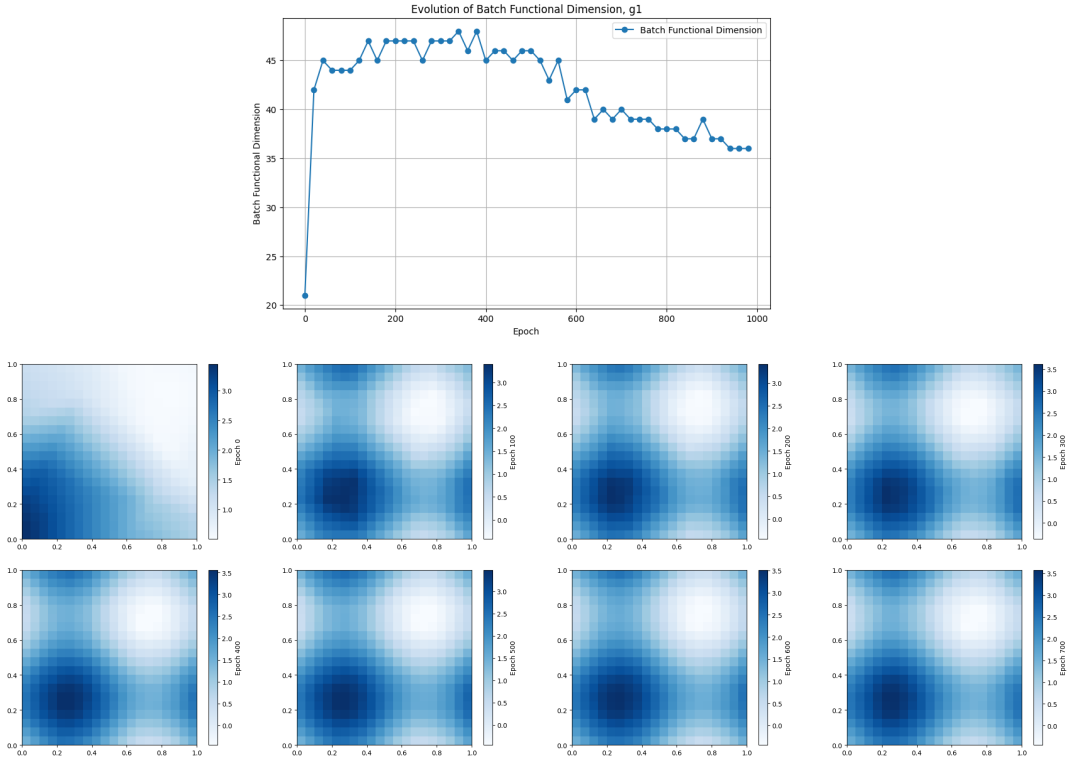

Figure 12: Functional dimension and contour maps of a single training run (with weight decay = 0) for $g1$ every 100 epochs.

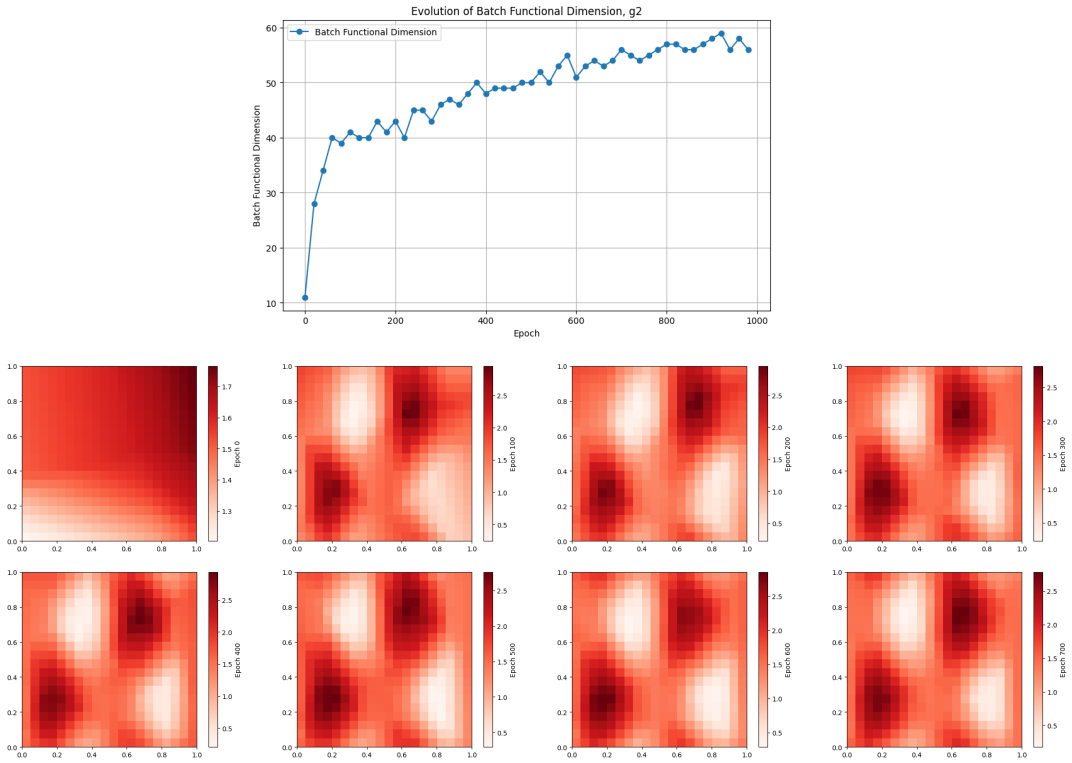

Figure 13: Functional dimension and contour maps of a single training run (with weight decay 0) for $g2$ every 100 epochs.

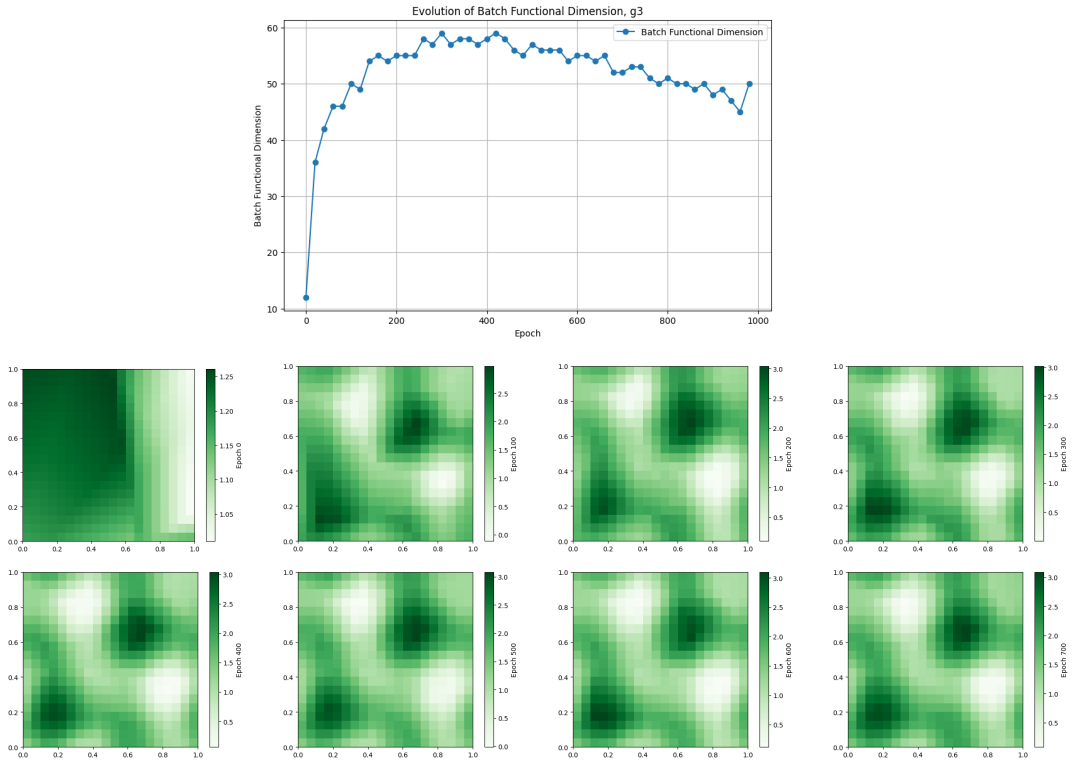

Figure 14: Functional dimension and contour maps of a single training run (with weight decay 0) for $g3$ every 100 epochs.

C NUMBER OF LINEAR REGIONS SAMPLED BY THE DATA SET INPUTS

Definition C.1. *Fix a finite set $Z \in \mathbb{R}^{n_0}$. Let θ be a parameter such that every point $z_i \in Z$ is in the interior of a top-dimensional cell of the canonical polyhedral complex $\mathcal{C}(\theta)$. Then we will say that the* number of linear regions of f_θ sampled by Z *is the number of n_0-dimensional cells of $\mathcal{C}(\theta)$ that have nonempty intersection with Z.*

Remark C.2. The reason for requiring that every point z_i is in the interior of a top-dimensional cell is to avoid ambiguity in counting caused by a point z_i being on the boundary of two or more top-dimensional cells.

Proposition C.3. *Fix an architecture $(n_0, \ldots, 1)$ of fully-connected feedforward ReLU neural networks with one-dimensional output. Fix a finite set $Z \subset \mathbb{R}^{n_0}$. Let θ be a generic, transversal, combinatorially stable parameter such that every point z_i is in the interior of a n_0-dimensional cell of the canonical polyhedral complex $\mathcal{C}(\theta)$. Then $\dim_{ba.fun}(\theta, Z)$ is at most $(n_0 + 1)$ times the number of linear regions of f_θ sampled by Z.*

In particular, for an architecture with one-dimensional input and output, $\dim_{\text{ba.fun}}(\theta, Z)$ is at most twice the number of linear regions sampled by Z.

Proof. Suppose $z_1, \ldots, z_k$ are all in the interior of the same cell $C \in \mathcal{C}(\theta)$. The assumptions on θ guarantee (from results in Grigsby et al. (2022)) that there is an open neighborhood U of θ on which the function $\{z_i\} \times U \to \mathbb{R}$ given by $(z, u) \mapsto f(\theta)(z)$ is affine-linear (in every coordinate). Consequently, if the point z_{k+1} can be expressed as a linear combination of the points $z_1, \ldots, z_k$, then the row vector $\mathbf{J}E_{z_{k+1}}(\theta)$ can be expressed as a linear combination of the row vectors $\mathbf{J}E_{z_1}(\theta), \ldots, \mathbf{J}E_{z_k}(\theta)$. Since a top-dimensional geometric simplex in $\mathbb{R}^{n_0}$ has $n_0 + 1$ points, and batch functional dimension is the number of linearly independent rows of the corresponding matrix, the result follows. □

Figure 9 tracks the batch functional dimension and number of linear regions sampled by $\mathcal{D}_1$ over a single training run. Figure 10 computes averages per epoch over 20 randomly initialized training runs.

The plots demonstrate that there is a strong correlation between the number of linear regions sampled by $\mathcal{D}_1$ and the batch functional dimension. However, the batch functional dimension is well below the upper bound imposed by the number of linear regions as in Proposition C.3.

Remark C.4. Definition C.1 and Proposition C.3 are closely related to notion of decisive sets defined in Grigsby et al. (2022). A *decisive* set for a parameter θ is a set $Z \subset \mathbb{R}^{n_0}$ consisting of precisely $n_0 + 1$ points in the interior of each top-dimensional polyhedron $C \in \mathcal{C}(\theta)$ that form an n_0-dimensional simplex. Grigsby et al. (2022) proves that for a generic, transversal, combinatorially stable parameter, functional dimension is attained on any decisive set.