A Hessian Based Complexity Measure for Deep Networks

Hamid Javadi
ECE Department
Rice University

Randall Balestriero
ECE Department
Rice University

Richard Baraniuk
ECE Department
Rice University

Abstract

Deep (neural) networks have been applied productively in a wide range of supervised and unsupervised learning tasks. Unlike classical machine learning algorithms, deep networks typically operate in the overparameterized regime, where the number of parameters is larger than the number of training data points. Consequently, understanding the generalization properties and role of (explicit or implicit) regularization in these networks is of great importance. Inspired by the seminal work of Donoho and Grimes in manifold learning, we develop a new measure for the complexity of the function generated by a deep network based on the integral of the norm of the tangent Hessian. This complexity measure can be used to quantify the irregularity of the function a deep network fits to training data or as a regularization penalty for deep network learning. Indeed, we show that the oft-used heuristic of data augmentation imposes an implicit Hessian regularization during learning. We demonstrate the utility of our new complexity measure through a range of learning experiments.

1 Introduction

Deep (neural) networks are being profitably applied in a large and growing number of areas, from signal processing to computer vision and artificial intelligence. The expressive power of these networks has been demonstrated both in theory and practice [Cyb89, Bar94, Tel15, Yar17, HS17, DDF+19]. In fact, it has been shown that deep networks can even perfectly fit pure noise [ZBH+16]. Surprisingly, highly overparameterized deep networks – where the number of network parameters exceeds the number of training data points – can be trained for a range of different classification and regression tasks and perform extremely well on unobserved data. Understanding why these networks generalize so well has been the subject of great interest in recent years. But, to date, classical approaches to bounding the generalization error have failed to provide much insight into deep networks.

The ability of overparameterized deep networks to overfit noise while generalizing well suggests the existence of some kind of (explicit or implicit) regularization in the learning process. In order to both understand and improve regularization in training deep networks, one key question to address is: What is the correct measure to evaluate the complexity of a deep network? As we discuss in detail below, both classical and recent measures have come up short on insights. In this paper, we take a different tack. Let \( f(x) \) represent the mapping from the input to output of a deep network constructed using piecewise-affine activations (e.g., ReLU, leaky ReLU, absolute value). When activations are also convex, \( f \) can be written as composition of Max Affine Spline Operators (MASOs) and using the

Preprint. Under review.
framework provided in [BB18a, BB18b], we can write $f$ as the continuous, piecewise affine operator

$$f(x) = A[x]x + b[x].$$

(1)

Below, we propose a new complexity measure for $f$ based on the $A$ matrices.

The main intuition behind our measure can be described as follows. We aim to quantify how far the mapping $f$ is from a locally linear mapping on the data. Motivated by the concept of Hessian eigenmaps introduced by Donoho and Grimes [DG03] for manifold learning, we propose the tangent Hessian norm integral as a new complexity measure for deep networks.

Two main features of our measure distinguish it from other proposed regularization penalties: 1) **Distance from a linear mapping**: Most regularization penalties proposed in the literature focus on the behavior of the mapping $f$ on regions where $f$ is linear. For example, Tikhonov regularization on the weights of the network bounds the Lipschitz constant of the mapping in individual regions. In contrast, our Hessian measure quantifies how much the mapping differs from an affine mapping over the entire input space. 2) **Local geometrical structure of the input data**: In most applications, for example when the input data consists of images, the training data points $x_i \in \mathbb{R}^D$ lie on a lower-dimensional manifold $M$ of dimension $d \ll D$. We can exploit the local geometrical structure of the data to evaluate the mapping $f$ as a function of the manifold local coordinates.

Our main contributions can be summarized as follows:

[C1] **A new complexity measure for deep networks.** In Section 2, we propose and justify the tangent Hessian norm integral as a new complexity measure for deep networks.

In Section 4, we present two methods to compute the measure efficiently.

[C2] **Understanding the role of deep network parameters on complexity.** In Section 3, we study the growth in complexity of functions generated by the units (units) in each layer of a deep network. This provides an upper bound on the complexity of the network output in terms of network parameters.

[C3] **Data augmentation as implicit Hessian regularization.** In Section 5, we study data augmentation and show that using this technique while training a deep network decreases the Hessian complexity measure. Hence, we can consider this technique as an implicit regularization method with the tangent Hessian norm integral as the penalty.

More broadly, our Hessian complexity measure can open up new directions in understanding the role of optimization methods such as stochastic gradient descent in training deep networks as an implicit regularization. In addition, it can inspire new training frameworks for deep networks that are robust to a range of adversarial attacks. Proofs of all of our results appear in the Appendix.

**Related work.** There is a growing literature studying the generalization properties of overparameterized neural networks [BM02, BFT17, AGNZ18, DR17, NBMS17]. In these papers, the authors obtained sharper bounds than naive parameter counting by using the stability of the deep network around the weights achieved after training. Recently, it has been shown that by having sufficient overparameterization, the weights of a trained network will be close to its random initialization [AZLL18, ADH+19b, CG19]. Using this fact, these papers have achieved better generalization bounds for deep networks. In [HR19], the authors observe that the expected number of linear regions in a trained deep network grows polynomially (and not exponentially) with the number of units. All of the above works suggest that, in order to understand generalization in deep nets, one must measure the complexity of the deep network mapping.

Recently, in [SESS19], the authors show that for one-dimensional, one hidden layer, ReLU networks, an $\ell_2$ penalty on the weights (Tikhonov regularization) is equivalent to a penalty on the integral of the absolute value of the second derivative of the network output. Moreover, in [BHX19, BHMM18, HMRT19], the authors show that in some linear and nonlinear inference problems, properly regularized overparameterized models can generalize well. This again indicates the importance of complexity regularization for understanding generalization in deep networks.

In [RMV+11], the authors propose an auto-encoder whose regularization penalizes the change in the encoder Jacobian. Also, in [DGR+18], the effects of data augmentation have been studied by modeling data augmentation as a kernel.
2 Formulating the Hessian Based Complexity Measure

Network with smooth activations. Let $f : \mathbb{R}^D \to \mathbb{R}$ be the prediction function of a deep network whose nonlinear activation functions are smooth functions. For regression, we can take $f$ as the mapping from the input to the output of the network. For classification, we can take $f$ as the mapping from the input to one of the inputs of the final softmax operation. We assume that the training data $\{x_i\}_{i=1}^N$ lies close to a $d$-dimensional smooth manifold $M \subseteq \mathbb{R}^D$. This assumption has been studied in an extensive literature on unsupervised learning, e.g. [TDLS00, BN03, DG03], and holds at least approximately for many practical datasets, including images.

For $p \geq 1$, inspired by the Hessian eigenmaps approach of Donoho and Grimes [DG03], we propose the following complexity measure $C_p(f)$,

$$C_p(f) \equiv \left( \int_M \left\| \nabla^2 f(x) \right\|_F^p \, dx \right)^{1/p}, \tag{2}$$

where $\nabla^2 f(x)$ is the Hessian of $f$ at $x$ in the coordinates of $d$-dimensional affine space tangent to manifold at $x \in M$.

From [DG03] we know that $C_p(f)$ measures the average curviness of $f$ over the manifold $M$ and that $C_p(f)$ is zero if and only if $f$ is an affine function on $M$. In the simplistic case of one-dimensional data and one hidden layer networks, [SESS19] have related $C_1(f)$ to the sum of the squared Frobenius norms of the weight matrices.

While the manifold assumption is highly recommended for exploiting the data’s geometrical structure in computing the complexity measure (2), it is not essential. We can take $d = D$ and by let $M$ be the entire input space to obtain

$$C_p(f) = \left( \int_{\mathbb{R}^D} \left\| \nabla^2 f(x) \right\|_F^p \, dx \right)^{1/p}. \tag{3}$$

This measure is easier to compute, but it does not exploit the geometrical structure of the data and therefore might not be as revealing as (2).

Network with continuous, piecewise affine activations. Our focus in this paper is on a complexity measure for deep networks constructed using piecewise affine activations (e.g., ReLU, leaky ReLU, absolute value). In this case, $f$ is piecewise affine and thus not continuously differentiable. Therefore, the Hessian is not well defined in its usual form.

Note that a network with continuous, piecewise affine activations partitions the input space $\mathbb{R}^D$ based on the activation pattern of the network’s units (neurons). We call these partitions, $\mathcal{Q}_j \subset \mathbb{R}^D$, the vector quantization (VQ) regions of the network. Note that inside one VQ region, $f$ is simply an affine mapping. As a result, $f$ can be written as a continuous, piecewise affine operator as in (1).

Note that in (1), $A[x], b[x]$ are in fact functions of the network activation pattern and therefore the VQ region $\mathcal{Q}_j[x] \in \mathbb{R}^D$ containing $x$. However, for the sake of brevity, we will use the simplified notation in (1).

We can now define our complexity measure for a network with continuous, piecewise affine activations. Let $\varepsilon > 0$. For $x$ not on the boundaries of VQ partitions and $u$ an arbitrary unit vector, we define $H_\varepsilon(x)$ as

$$H_\varepsilon(x)[u] \equiv \frac{1}{\varepsilon} (A[x + \varepsilon u] - A[x]) \tag{4}$$

if $u$ is on the $d$-dimensional affine space tangent to the data manifold $M$ at $x$ and $H_\varepsilon(x)[u] = 0$, otherwise. Note that $A[x]$ is a (weak) gradient of $f$ at $x$, and therefore this definition agrees with the finite element definition of the Hessian. For smooth $f$ and $\varepsilon \to 0$ recovers the Hessian [MT00, JJ65].

Thus, for a network with continuous, piecewise affine activations, we define

$$C_{p,\varepsilon}(f) \equiv \sqrt{d} \left( \int_M \left\| \nabla H_\varepsilon(x)[u] \right\|_F^2 \, dx \right)^{1/p} \tag{5}$$

where $u$ is uniform over the unit sphere. Comparing with (2), this definition is consistent with the definition of the distributional derivative for piecewise constant functions and can be seen as measuring the changes in the local slopes of the piecewise affine spline realized by the network.
It is worthwhile to compare the Hessian complexity measure with the notion of “number of VQ regions” [HR19] in a ReLU network. We believe that the Hessian measure provides a more useful quantification of the network output complexity, because it explicitly takes into account the changes in the output function across the VQ regions. For instance, consider the analysis of infinitely wide networks, which have been used to help understand the convergence properties and performance of deep networks [LBN+17, MMIT9, ADH+19]. The number of VQ regions can be infinite in such networks; however, the Hessian measure remains bounded as long as the network weight matrices have rows with bounded norm. We will discuss this in more detail in Section 3.

When a network has more than one hidden layer, it is not straightforward to obtain an explicit formula for $C_p(f)$ or $C_{p,x}(f)$ in terms of the network parameters (weight matrices and biases). However, it is possible to efficiently approximate the Hessian measure (see Section 4).

3 Hessian Complexity Growth Through a Deep Network

In this section, we study the outputs of different units (neurons) in a ReLU deep network as functions of the input in order to shed light on how the Hessian complexity increases through the network. The scalar ReLU activation function ReLU$(v) = \max\{0, v\}$ is applied elementwise to an input vector to create a thresholded output. A ReLU network intersperses affine transformations $x \mapsto Wx + b$ with ReLU thresholding. We call $W$ the weights and $b$ the biases of the network.

We focus on the case of a network with $L$ layers processes a scalar, one-dimensional input $x \in \mathbb{R}$. For $\ell \in \{1, 2, \ldots, L\}$, let $D^{(\ell)}$ be the number of units in the $\ell$-th layer of the network. In this case, the output of $r$-th unit in the $\ell$-th layer of the network, $1 \leq r^{(\ell)} \leq D^{(\ell)}$, is a continuous, piecewise affine function of the input $x$ that we denote by $f^{(\ell)}_r$. Such a function can be written as

$$f^{(\ell)}_r(x) = \begin{cases} a_0x + b_0 & x \leq x_1 \\ a_qx + b_q & x_q < x \leq x_{q+1}, \ q = 1, 2, \ldots, Q - 1 \\ a_Qx + b_Q & x \geq x_Q \end{cases}$$

where $Q + 1$ is the number of linear pieces of $f^{(\ell)}_r(x)$ and $x_1, x_2, \cdots, x_Q$ are the spline break points (or knots). We can compute

$$C_1(f^{(\ell)}_r) = \lim_{\varepsilon \to 0} C_1,\varepsilon(f^{(\ell)}_r)$$

for $f^{(\ell)}_r$ as in (6) via

$$C_1(f^{(\ell)}_r) = \sum_{q=0}^{Q-1} |a_q - a_{q+1}|.$$  

In order to understand the complexity of the output functions of different units of the network, i.e., $f^{(\ell)}_r$ for $1 \leq \ell \leq L$ and $1 \leq r \leq D^{(\ell)}$, we must compute the complexity of a unit output function in terms of the complexities of its input functions. In a ReLU network, each unit performs two operations on its input functions: a linear combination of the input functions followed by ReLU thresholding. The following lemma studies the linear combination process and finds that $C_1(f)$ is a seminorm on the space of one-dimensional, continuous, piecewise affine functions $f$.

**Lemma 1.** Let $f^{(\ell)}_r(x) : \mathbb{R} \to \mathbb{R}$ be piecewise affine functions for $r = 1, 2, \ldots, D^{(\ell)}$ and let $w \in \mathbb{R}^{D^{(\ell)}}$. If $f(x) = \sum_{r=1}^{D^{(\ell)}} w_rf^{(\ell)}_r(x)$, then

$$C_1(f(x)) \leq \sum_{r=1}^{D^{(\ell)}} |w_r|C_1(f^{(\ell)}_r(x)) \leq \|w\|_1 \max_{1 \leq r \leq D^{(\ell)}} C_1(f^{(\ell)}_r(x)).$$

The following theorem bounds the complexity of ReLU$(f)$ in terms of the complexity of $f$ and its Lipschitz constant.

**Theorem 1.** Let $f : \mathbb{R} \to \mathbb{R}$ be a piecewise affine function as in (6) with Lipschitz constant $K$. Then

$$C_1(\text{ReLU}(f)) \leq C_1(f) + 2K.$$
We can also apply the Monte Carlo method to estimate smooth activations. When the network has bounded weight matrices, we can bound the complexity of the layers for networks with bounded weight matrices. This suggests that, in the case of one-dimensional networks with weight matrices of bounded norm, the maximum complexity of the output is a linear function of number of units in the network.

Define the complexity of the $\ell$-th layer of the network as

$$C_1^{(\ell)} = \max_{1 \leq r \leq D^{(\ell)}} C_1(f^{(\ell)}_r(x)).$$

We can bound the complexity of the layers for networks with bounded weight matrices.

**Theorem 2.** Let $W^{(\ell-1)}$ with rows $w^{(\ell-1)}_r$ be the weight matrix of the $(\ell-1)$-th layer. If the output functions of the units in the $(\ell-1)$-th layer have Lipschitz constant $K$, then

$$C_1^{(\ell)} \leq \left( C_1^{(\ell-1)} + 2K \right) \max_{r} \|w^{(\ell-1)}_r\|_1.$$  

Using the fact that $C_1^{(0)} = 0$, we have the following.

**Corollary 1.** If the weight matrices of all layers have rows with $\ell_1$ norm less than one, then the complexity of the network output function is at most $2L$, where $L$ is the number of layers.

### 4 Computing the Hessian Complexity Measure

In this section, we discuss two efficient methods to approximate $C_p(f)$ in practical deep networks.

**Finite Differences Method.** First note that we can find the $d$-dimensional subspace tangent to the data manifold $M$ at neighborhood around data point $x_i$ as the $d$-dimensional principal subspace of $X_i \in \mathbb{R}^{d+k+1}$ defined by $X_i = [x_i - x_{i_1}, x_{i_2} - x_{i_1}, \ldots, x_{i_k} - x_{i_1}]$, where $\{ x_{i_1}, x_{i_2}, \ldots, x_{i_k} \}$ are the set of $k$ nearest neighbors of $x_i$ and

$$x_i = \frac{1}{k+1} \left( x_i + \sum_{\ell=1}^{k} x_{i_{\ell}} \right).$$  

In this method, we estimate $C_p(f)$ using its Monte Carlo approximation $\bar{C}_p(f)$ based on the training data $x_i$.

When the network has smooth activations, we have

$$\bar{C}_p(f) \equiv \left( \frac{1}{n} \sum_{i=1}^{n} \left\| \nabla^2_{(\tan)} f(x_i) \right\|_F^p \right)^{1/p}.$$  

We can also apply the Monte Carlo method to estimate $\| \nabla^2_{(\tan)} f(x_i) \|_F^p$. If $u \in \mathbb{R}^d$ is chosen uniformly at random on the unit sphere, then for $A \in \mathbb{R}^{d \times d}$ we have

$$\mathbb{E} \| Au \|_2^2 = \mathbb{E} \sum_{i,j} u_i u_j (A^T A)_{ij} = \frac{1}{d} \| A \|_F^2.$$  

For smooth $f$, we have

$$\nabla^2 f(x) u = \lim_{\delta \to 0} \frac{1}{\delta} \left[ \nabla f(x + \delta u) - \nabla f(x) \right] .$$

Therefore, choosing $u_j$ uniformly at random with $\| u_j \|_2 = 1$ on the $d$-dimensional subspace tangent to the manifold at $x$, plugging (15), (16) into (14) yields the following approximation for $C_p(f)$ for small $\delta > 0$

$$\bar{C}_p(f, \delta) \equiv \left( \frac{d^{p/2}}{n^p \delta^{p/2}} \sum_{i=1}^{n} \left( \sum_{j=1}^{m} \left\| \nabla f(x_i + \delta u_j) - \nabla f(x_i) \right\|_2^2 \right)^{p/2} \right)^{1/p} .$$  

5
When the network has *continuous, piecewise affine activations*, Monte Carlo approximation of $C_{p,ε}(f)$ based on the training data $x_i$ yields

$$
\tilde{C}_{p,ε}(f) ≜ \left( \frac{d^{p/2}}{m} \sum_{i=1}^{n} \left( \mathbb{E}_u \| H_s(x_i)[u]\|_2^2 \right)^{p/2} \right)^{1/p}.
$$

(18)

For such a network, using $[\text{4}]$ yields

$$
\tilde{C}_p(f,ε) = \left( \frac{d^{p/2}}{n^p m} \sum_{i=1}^{n} \left( \sum_{j=1}^{m} \| A[x_i + εu_j] - A[x_i]\|_2^2 \right)^{p/2} \right)^{1/p}
$$

(19)

as our approximation.

In a ReLU network with $L$ layers, for a given $x$, $A[x]$ can be computed very efficiently via

$$
A[x] = W^{(L)} \left( \prod_{\ell=1}^{L} D^{(L-\ell)}[x] W^{(L-\ell)} \right),
$$

(20)

where $W^{(\ell)}$ is the weight matrix of the $\ell$-th layer and $D^{(\ell)}[x]$ is a diagonal matrix with $(D^{(\ell)}[x])_{rr} = 1$ if the output of the $r$-th ReLU unit in the $\ell$-th layer is nonzero (aka “active”) with $x$ as input and $(D^{(\ell)}[x])_{rr} = 0$ if the ReLU output is zero. This enables us to use $\tilde{C}_{p,ε}(f,δ)$ as a regularization penalty in training real networks.

**Discretization Method.** When the manifold dimension $d < D$, we can use the discretization technique of [DG03] to write the penalty in (2) as a quadratic form of the values of $f$ on the data points.

Let $M_i \subset M$ be the manifold neighborhood of $x_i$ that contains $x_i$ and its $k$ nearest neighbors $\mathcal{N}(i) = \{x_i, x_{j_1}, x_{j_2}, \ldots, x_{j_k}\}$. Taking $\{\phi_i\}_{i=1}^{n}$ as a partition of unity on $M$ with supp$\{\phi_i\} \subset M_i$, we can write $C_{p}(f)$ as

$$
C_p(f) = \left( \sum_{i=1}^{n} \int_{M_i} \phi_i \left\| \nabla^2_{\text{tan}} f(m) \right\|_F^{p} \,dm \right)^{1/p}.
$$

(21)

Let $f_i$ be the vector of samples of $f$ over $x_i$

$$
f_i = [f(x_i), f(x_{j_1}), f(x_{j_2}), \ldots, f(x_{j_k})].
$$

(22)

We use the approximation

$$
\tilde{C}_{p,\text{sampled}}(f) ≜ \left( \sum_{i=1}^{n} (f_i, \mathcal{H}_i f_i)^{p/2} \right)^{1/p},
$$

(23)

where $\mathcal{H}_i$ is the tangent Hessian operator on $M_i$, $\mathcal{H}_i$ is a positive semidefinite matrix whose null space contains the constant and linear functions on the tangent space at $M_i$; it can be constructed in the following way [Wan12]. Let $V^i = [v^i_1, v^i_2, \ldots, v^i_d]$ be top $d$ right singular vectors of $X_i = [x_i - \bar{x}_i, x_{j_1} - \bar{x}_i, x_{j_2} - \bar{x}_i, \ldots, x_{j_k} - \bar{x}_i]$ and construct the matrix

$$
Q^i = [v^i_s \circ v^i_{s'}]_{1 \leq s \leq s' \leq d},
$$

(24)

where $\circ$ is the Hadamard product of the vectors $x_1, x_2$. By performing Gram-Schmidt orthogonalization on the columns of $[1, V^i, Q^i]$, we obtain $[1, \tilde{V}^i, \tilde{Q}^i]$, which forms an orthonormal basis for quadratic functions on $\mathcal{N}_i$. Then

$$
\mathcal{H}_i = \tilde{Q}^i \tilde{Q}^{iT}
$$

(25)

is the tangent Hessian operator on $M_i$.

For $p = 2$, this approximation can be further simplified to

$$
\tilde{C}_{2,\text{sampled}}(f) = (f, \mathcal{H} f),
$$

(26)
We can make a few additional observations from the results in Table 1.

1) Impact of network

\[ \text{WGSM16, PW17} \] is an oft-used, yet poorly understood, heuristic applied in Data augmentation where

\[ \text{Consider a deep network with continuous, piecewise affine activations and thus pre-} \]

\[ \text{Theorem 3. The following result establishes the relationship between data augmentation and Hessian complexity.} \]

\[ \text{Theorem 3. Consider a deep network with continuous, piecewise affine activations and thus prediction function } f \text{ as in [1]. Assume that } \| x \|_2 \leq R, \text{ and } f \text{ has Lipschitz constant } K_1, \text{ i.e., } \| A x \|_2 \leq K_1, \text{ for all } x. \text{ Further, assume that the loss function } \ell \text{ has Lipschitz constant } K_2. \text{ Then, for } \| u_{ij} \|_2 \leq \varepsilon \text{ and small enough } \varepsilon, \text{ } L^{\text{aug}} \text{ in (29) can be approximated by} \]

\[ L^{\text{aug}} \leq L + \frac{R K_2}{m+1} \sum_{i=1}^{n} \sum_{j=1}^{m} \| A [x_i + u_{ij}] - A [x_i] \|_2 
+ \frac{K_2}{m+1} \sum_{i=1}^{n} \sum_{j=1}^{m} \| b [x_i + u_{ij}] - b [x_i] \| + \frac{K_1 K_2 m n \varepsilon}{m+1}. \]

From (30), we can note the close relationship between data augmentation and the Hessian complexity measure. Indeed, the second term on the right-hand side of the inequality is very similar in form to the Hessian complexity measure in (12). This, suggests that adding a Hessian complexity penalty term as a regularizer to the loss should decrease the resulting \( L^{\text{aug}} \) of the network. Moreover, in the experimental validation in Figure 4 and Table 1, we observe that the converse is also true; data augmentation also decreases the Hessian complexity.

We can make a few additional observations from the results in Table 1.  

1) Impact of network architecture on complexity: We observe that the convolutional network (CNN) results in smaller \( C_2 \) than the residual network (ResNet). This sheds light on the advantages of a convolutional architecture for image classification. We also observe that the \( C_2 \) measured using the training data is almost zero for CNNs trained on all four datasets. This suggests that the prediction function is almost linear within \( \varepsilon \) of the training data. However, the complexity is significantly higher when measured using the test data. This is an interesting property of purely convolutional networks that begs investigation.

2) Impact of overparameterization on complexity: Surprisingly, the Large ResNet (with more parameters) results in smaller \( C_2 \) than the smaller ResNet. This might be a result
of the implicit regularization that results from training overparameterized networks via stochastic gradient descent (SGD) \cite{ADH19b, AZLL18}.

3) Impact of dataset on complexity: As expected, training the same network for a more complex task (e.g., classification with CIFAR100 vs. CIFAR10) results in larger $\tilde{C}_2$.

6 Discussion

In this paper we have introduced a new Hessian-based measure for the complexity of a deep network and its prediction. An attractive property of our measure compared to previously proposed measures such as the number of linear regions (VQ partitions) is that it captures the amount by which the network’s output changes not just locally but across the entire input space. Further, our measure
Table 1: Additional experimental validation that data augmentation reduces the Hessian complexity measure \( C^2 \) in a classification task with a range of deep networks and datasets. We tabulate the converged values for \( C^2, \varepsilon (f, \delta) \) on the training and test data without and with data augmentation (denoted by DA). The fourth and sixth sets of rows summarizes the experiment in Figure 1. Generally speaking, learning with data augmentation reduces the Hessian complexity. (See the Appendix for the experimental details.)

|                     | \( \tilde{C}^2 \) on training data | \( \tilde{C}^2 \) on test data | Test accuracy (%) |
|---------------------|-----------------------------------|---------------------------------|-------------------|
| CNN (MNIST)         | 1.57e-09                          | 0.036                           | 99.6              |
| CNN+DA (MNIST)      | 1.56e-09                          | 0.074                           | 99.6              |
| ResNet (MNIST)      | 0.086                             | 0.20                            | 99.4              |
| ResNet+DA (MNIST)   | 0.021                             | 0.016                           | 99.4              |
| Large ResNet (MNIST)| 0.061                             | 0.079                           | 99.5              |
| Large ResNet+DA (MNIST)| 0.019                        | 0.025                           | 99.5              |
| ResNet (CIFAR10)    | 0.43                              | 0.50                            | 84.9              |
| ResNet+DA (CIFAR10) | 0.10                              | 0.12                            | 91.0              |
| CNN (CIFAR10)       | 1.54e-09                          | 0.10                            | 87.4              |
| CNN+DA (CIFAR10)    | 2.58e-09                          | 0.11                            | 91.7              |
| ResNet (SVHN)       | 0.08                              | 0.09                            | 93.9              |
| ResNet+DA (SVHN)    | 0.01                              | 0.01                            | 94.1              |
| CNN (SVHN)          | 8.3e-10                           | 0.022                           | 95.6              |
| CNN+DA (SVHN)       | 6.32e-10                          | 0.019                           | 95.5              |
| ResNet (CIFAR100)   | 4.32                              | 4.39                            | 49.3              |
| ResNet+DA (CIFAR100)| 1.08                              | 1.14                            | 64.5              |
| CNN (CIFAR100)      | 6.3e-08                           | 1.51                            | 61.1              |
| CNN+DA (CIFAR100)   | 7.0e-08                           | 1.40                            | 68.3              |

explicitly exploits the geometrical structure of the training data. We have demonstrated a direct link between the heuristic of data augmentation and an implicit Hessian complexity penalty during learning. There are many potential applications for our new measure, including new ways to study generalization and optimization in deep networks and new more powerful regularization penalties.

References

[ADH+19a] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang, On exact computation with an infinitely wide neural net, arXiv preprint arXiv:1904.11955 (2019).

[ADH+19b] 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, arXiv preprint arXiv:1901.08584 (2019).

[AGNZ18] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang, Stronger generalization bounds for deep nets via a compression approach, arXiv preprint arXiv:1802.05296 (2018).

[AZLL18] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang, Learning and generalization in overparameterized neural networks, going beyond two layers, arXiv preprint arXiv:1811.04918 (2018).

[Bar94] Andrew R Barron, Approximation and estimation bounds for artificial neural networks, Machine learning 14 (1994), no. 1, 115–133.

[BB18a] Randall Balestriero and Richard Baraniuk, Mad max: Affine spline insights into deep learning, arXiv preprint arXiv:1805.06576 (2018).

[BB18b] Randall Balestriero and Richard G. Baraniuk, A spline theory of deep networks, International Conference on Machine Learning, 2018, pp. 383–392.

[BFT17] Peter L. Bartlett, Dylan J Foster, and Matus J Telgarsky, Spectrally-normalized margin bounds for neural networks, Advances in Neural Information Processing Systems, 2017, pp. 6240–6249.
[BHMM18] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal, *Reconciling modern machine learning and the bias-variance trade-off*, arXiv preprint arXiv:1812.11118 (2018).

[BHX19] Mikhail Belkin, Daniel Hsu, and Ji Xu, *Two models of double descent for weak features*, arXiv preprint arXiv:1903.07571 (2019).

[BM02] Peter L. Bartlett and Shahar Mendelson, *Rademacher and gaussian complexities: Risk bounds and structural results*, Journal of Machine Learning Research 3 (2002), no. Nov, 463–482.

[BN03] Mikhail Belkin and Partha Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation 15 (2003), no. 6, 1373–1396.

[CG19] Yuan Cao and Quanquan Gu, *A generalization theory of gradient descent for learning over-parameterized deep relu networks*, arXiv preprint arXiv:1905.02199 (2019).

[Cyb89] George Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems 2 (1989), no. 4, 303–314.

[DDF+19] Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova, *Nonlinear approximation and (deep) relu networks*, arXiv preprint arXiv:1905.02199 (2019).

[DG03] David L. Donoho and Carrie Grimes, *Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data*, Proceedings of the National Academy of Sciences 100 (2003), no. 10, 5591–5596.

[DGR+18] Tri Dao, Albert Gu, Alexander J Ratner, Virginia Smith, Christopher De Sa, and Christopher Ré, *A kernel theory of modern data augmentation*, arXiv preprint arXiv:1803.06084 (2018).

[DR17] Gintare Karolina Dziugaite and Daniel M. Roy, *Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data*, arXiv preprint arXiv:1703.11008 (2017).

[HMRT19] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani, *Surprises in high-dimensional ridgeless least squares interpolation*, arXiv preprint arXiv:1903.08560 (2019).

[HR19] Boris Hanin and David Rolnick, *Complexity of linear regions in deep networks*, arXiv preprint arXiv:1901.09021 (2019).

[HS17] Boris Hanin and Mark Sellke, *Approximating continuous functions by relu nets of minimal width*, arXiv preprint arXiv:1710.11278 (2017).

[JJ65] Charles Jordan and Károly Jordán, *Calculus of finite differences*, vol. 33, American Mathematical Soc., 1965.

[LBN+17] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein, *Deep neural networks as gaussian processes*, arXiv preprint arXiv:1711.00165 (2017).

[MMM19] Song Mei, Theodor Misiakiewicz, and Andrea Montanari, *Mean-field theory of two-layer neural networks: dimension-free bounds and kernel limit*, arXiv preprint arXiv:1902.06015 (2019).

[MT00] Louis Melville Milne-Thomson, *The calculus of finite differences*, American Mathematical Soc., 2000.

[NBMS17] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro, *Exploring generalization in deep learning*, Advances in Neural Information Processing Systems, 2017, pp. 5947–5956.

[PW17] Luis Perez and Jason Wang, *The effectiveness of data augmentation in image classification using deep learning*, arXiv preprint arXiv:1712.04621 (2017).

[RMV+11] Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot, *Higher order contractive auto-encoder*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2011, pp. 645–660.
Appendices

A Proofs

A.1 Proof of Lemma 1

Let $f^{(\ell)}_1(x)$ be as in (6). We have

$$C_1(\alpha f^{(\ell)}_1) = \sum_{q=0}^{Q-1} |\alpha a_q - \alpha a_{q+1}| = |\alpha| C_1(f^{(\ell)}_1). \quad (31)$$

Further, let

$$f_2(x) = \begin{cases} a'_0x + b'_0 & x \leq x'_1 \\ a'_q x + b'_q & x'_q \leq x \leq x'_{q+1}, \ q' = 1, \ldots, Q'-1 \\ a'_{Q'} x + b'_{Q'} & x \geq x'_{Q'}. \end{cases} \quad (32)$$

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_Q\}$, $\mathcal{X}' = \{x'_1, x'_2, \ldots, x'_{Q'}\}$. We have

$$C_1(f_1 + f_2) = \sum_{0 \leq q < Q, x_q \notin \mathcal{X}'} |a_q - a_{q+1}| + \sum_{0 \leq q' < Q', x'_{q+1} \notin \mathcal{X}} |a'_q - a'_{q+1}| \quad (33)$$

$$+ \sum_{0 \leq q \leq Q, 0 \leq q' \leq Q'} |a_q + a'_q - a_{q+1} - a'_{q+1}| \quad (34)$$

$$\leq \sum_{0 \leq q < Q} |a_q - a_{q+1}| + \sum_{0 \leq q' < Q'} |a'_q - a'_{q+1}| \quad (35)$$

$$= C_1(f_1) + C_1(f_2). \quad (36)$$

Combining this with (31), the first inequality in (9) is proved. The second inequality is the result of applying the Cauchy-Schwarz inequality.
A.2 Proof of Theorem 1

Let $f$ be as in (3). If at the most negative point where $f$ crosses zero, the function changes from negative to positive, we call this root $\eta_0$. Also, if at the most positive point where $f$ crosses zero, it changes from positive to negative, we call this root $\lambda_{M+1}$. Let all other zero crossings of $f$ be $\lambda_1 < \eta_1 < \lambda_2 < \eta_2 < \cdots < \lambda_M < \eta_M$, and let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_M\}$ and $H = \{\eta_1, \eta_2, \ldots, \eta_M\}$ be the set of points where $f$ changes from positive to negative, and from negative to positive, respectively. Note that we have

$$C_1(\text{ReLU}(f)) = \sum_{0 \leq q \leq Q-1; f(x_{q+1}) > 0} |a_q - a_{q+1}|$$

$$+ \sum_{0 \leq q \leq Q-1; f(x_{q+1}) = 0, a_q \leq 0 < a_{q+1}} |a_q - a_{q+1}| + \sum_{j=1}^M \left( |f'_-(\lambda_j)| + |f'_+(\eta_j)| \right)$$

and that

$$|f'_-(\lambda_j)| + |f'_+(\eta_j)| \leq \sum_{0 \leq q \leq Q-1; \lambda_j \leq x_{q+1} \leq \eta_j} |a_q - a_{q+1}|. \tag{40}$$

Therefore,

$$\sum_{j=1}^M \left( |f'_-(\lambda_j)| + |f'_+(\eta_j)| \right) \leq \sum_{0 \leq q \leq Q-1; f(x_{q+1}) < 0} |a_q - a_{q+1}|$$

$$+ \sum_{0 \leq q \leq Q-1; f(x_{q+1}) = 0, a_{q+1} \leq 0 < a_q} |a_q - a_{q+1}| \tag{42}$$

$$+ \sum_{0 \leq q \leq Q-1; f(x_{q+1}) = 0, 0 < a_q < a_{q+1}} |a_q - a_{q+1}| \tag{43}$$

$$+ \sum_{0 \leq q \leq Q-1; f(x_{q+1}) = 0, 0 > a_q, a_{q+1}} |a_q - a_{q+1}| \tag{44}$$

Therefore,

$$C_1(\text{ReLU}(f)) \leq 2K + \sum_{q=0}^{Q-1} |a_q - a_{q+1}| = 2K + C_1(f), \tag{45}$$

which completes the proof.

A.3 Proof of Theorem 2

The proof follows from Lemma 1 and Theorem 1. Note that under the theorem’s assumptions, all pre-activation functions in the $\ell$-th layer have Lipschitz constant of at most $\max_{1 \leq r \leq D(\ell)} \|w_r^{(\ell-1)}\|_1 K$. Further, by Lemma 1 their complexity is bounded by $\max_{1 \leq r \leq D(\ell)} \|w_r^{(\ell-1)}\|_1 C_1^{(\ell-1)}$. Applying Theorem 1 the proof is complete.

A.4 Proof of Theorem 3

Let $\ell_{ij} \equiv \ell(f(x_i + u_{ij}), y_i)$. Using (1), we have

$$\ell_{ij} = \ell(A[x_i + u_{ij}](x_i + u_{ij}) + b[x_i + u_{ij}], y_i)$$

where

$$= \ell(A[x_i]x_i + b[x_i]$$

$$+ (A[x_i + u_{ij}] - A[x_i])x_i + (b[x_i + u_{ij}] - b[x_i]) + A[x_i + u_{ij}]u_{ij}, y_i) \tag{47}.$$
Under the conditions of the theorem, we obtain
\[
\tilde{\ell}_{ij} \leq \ell_i + RK_2 \| A[x_i + u_{ij}] - A[x_i] \|_2 + K_2 \| b[x_i + u_{ij}] - b[x_i] \| + \varepsilon K_1 K_2.
\] (49)

Summing up over \( x_i \) and \( u_{ij} \) yields the following bound for \( \tilde{L}^{\text{aug}} \), the first order approximation of \( L^{\text{aug}} \) for small \( \varepsilon \),
\[
\tilde{L}^{\text{aug}} = \frac{1}{m+1} \sum_{i=1}^{n} \left( \ell_i + \sum_{j=1}^{m} \tilde{\ell}_{ij} \right)
\leq L + \frac{RK_2}{m+1} \sum_{i=1}^{n} \sum_{j=1}^{m} \| A[x_i + u_{ij}] - A[x_i] \|_2
\]
\[
+ \frac{K_2}{m+1} \sum_{i=1}^{n} \sum_{j=1}^{m} | b[x_i + u_{ij}] - b[x_i] | + \frac{K_1 K_2 m n \varepsilon}{m+1}
\] (51)

which completes the proof.

B Experimental Details

All experiments used the following parameters: batch size of 16, Adam optimizer with learning scheduled at 0.005 (initial), 0.0015 (epoch 100) and 0.001 (epoch 150). The default training/test split was used for all datasets. The validation set consists of 15% of the training set sampled randomly.

B.1 CNN Architecture

Conv2D(Number Filters=96, size=3x3, Leakiness=0.01))
Conv2D(Number Filters=96, size=3x3, Leakiness=0.01))
Conv2D(Number Filters=96, size=3x3, Leakiness=0.01))
Pool2D(2x2)
Conv2D(Number Filters=192, size=3x3, Leakiness=0.01))
Conv2D(Number Filters=192, size=3x3, Leakiness=0.01))
Conv2D(Number Filters=192, size=3x3, Leakiness=0.01))
Pool2D(2x2)
Conv2D(Number Filters=192, size=3x3, Leakiness=0.01))
Conv2D(Number Filters=192, size=1x1, Leakiness=0.01))
Conv2D(Number Filters=Number Classes, size=1x1, Leakiness=0.01))
GlobalPool2D(pool_type='AVG'))

B.2 ResNet and Large ResNet Architectures

The ResNets follow the original architecture \[ZK16\] with depth 2, width 1 for the ResNet and depth 4, width 2 for the Large ResNet.