Robust Large Margin Deep Neural Networks

Jure Sokolić, *Student Member, IEEE*, Raja Giryes, *Member, IEEE*, Guillermo Sapiro, *Fellow, IEEE*, and Miguel R. D. Rodrigues, *Senior Member, IEEE*

***Abstract*—The generalization error of deep neural networks via their classification margin is studied in this paper. Our approach is based on the Jacobian matrix of a deep neural network and can be applied to networks with arbitrary nonlinearities and pooling layers, and to networks with different architectures such as feed forward networks and residual networks. Our analysis leads to the conclusion that a bounded spectral norm of the network's Jacobian matrix in the neighbourhood of the training samples is crucial for a deep neural network of arbitrary depth and width to generalize well. This is a significant improvement over the current bounds in the literature, which imply that the generalization error grows with either the width or the depth of the network. Moreover, it shows that the recently proposed batch normalization and weight normalization reparametrizations enjoy good generalization properties, and leads to a novel network regularizer based on the network's Jacobian matrix. The analysis is supported with experimental results on the MNIST, CIFAR-10, LaRED, and ImageNet datasets.**

***Index Terms*—Deep learning, deep neural networks, generalization error, robustness.**

I. Introduction

In RECENT years, deep neural networks (DNNs) achieved state-of-the-art results in image recognition, speech recognition and many other applications [1]–[4]. DNNs are constructed as a series of non-linear signal transformations that are applied sequentially, where the parameters of each layer are estimated from the data [3]. Typically, each layer applies on its input a linear (or affine) transformation followed by a pointwise non-linearity such as the sigmoid function, the hyperbolic tangent function or the Rectified Linear Unit (ReLU) [5]. Many DNNs also include pooling layers, which act as down-sampling operators and may also provide invariance to various input transformations such as translation [6], [7]. They may be linear, as in average pooling, or non-linear, as in max-pooling.

There were various attempts to provide a theoretical foundation for the representation power, optimization and generalization of DNNs. For example, the works in [8], [9] showed that neural networks with a single hidden layer—shallow networks—can approximate any measurable Borel function. On the other hand, it was shown in [10] that a deep network can divide the space into an exponential number of sets, which can not be achieved by shallow networks that use the same number of parameters. Similarly, the authors in [11] conclude that functions implemented by DNNs are exponentially more expressive than functions implemented by shallow networks. The work in [12] shows that for a given number of parameters and a given depth, there always exists a DNN that can be approximated by a shallower network only if the number of parameters in the shallow network is exponential in the number of layers of the deep network.

Scattering transform—a convolutional DNN like transform, which is based on the wavelet transform and pointwise non-linearities—provides insights into translation invariance and stability to deformations of convolutional DNNs [13]–[15].

DNNs with random weights are studied in [16], where it is shown that such networks perform distance preserving embedding of low-dimensional data manifolds. The authors in [17] model a loss function of DNN with a spin-glass model and show that for large networks the local optima of the loss function are close to the global optima. Optimization aspects of DNNs are studied from the perspective of tensor factorization in [18] where it is shown that if a network is large, then it is possible to find the global minima from any initialization with a gradient descent algorithm. The role of DNNs in improving convergence speed of various iterative algorithms is studied in [19].

Optimization dynamics of a deep linear network is studied in [20], where it is shown that the learning speed of deep networks may be independent of their depth. Reparametrization of DNN for more efficient learning is studied in depth in [21]. A modified version of stochastic gradient descent for optimization of DNNs that are invariant to weight rescaling in different layers is proposed in [22], where it is shown that such an optimization may lead to a smaller generalization error (GE)—the difference between the empirical error and the expected error, than the one achieved with the classical stochastic gradient descent. The authors in [23] propose the batch normalization—a technique that normalizes the output of each layer and leads to faster training and also a smaller GE. A similar technique based on normalization of the weight matrix rows is proposed in [24].

Manuscript received October 3, 2016; revised February 12, 2017 and April 1, 2017; accepted May 9, 2017. Date of publication May 25, 2017; date of current version June 16, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gwo Giun Lee. The work of J. Sokolić and M. R. D. Rodrigues was supported in part by Engineering and Physical Sciences Research Council under Grant Ep/K033166/1. The work of R. Giryes was supported in part by the German-Israeli Foundation for Scientific Research and Development (GIF). The work of G. Sapiro was supported in part by National Science Foundation, Office of Naval Research, ARO, and NGA. *(Corresponding author: Jure Sokolić.)*

J. Sokolić and M. R. D. Rodrigues are with the Department of Electronic and Electrical Engineering, Univeristy College London, London WC1E 6BT, U.K. (e-mail: jure.sokolic.13@ucl.ac.uk; m.rodrigues@ucl.ac.uk).

R. Giryes is with the School of Electrical Engineering, Faculty of Engineering, Tel-Aviv University, Tel Aviv 6997801, Israel (e-mail: raja@tauex.tau.ac.il).

G. Sapiro is with the Department of Electrical and Computer Engineering, Duke University, NC 27708 USA (e-mail: guillermo.sapiro@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2017.2708039

It is shown empirically that such reparametrization leads to a faster training and a smaller GE. Learning of DNN by bounding the spectral norm of the weight matrices is proposed in [25]. Other methods for DNN regularization include weight decay, dropout [26], constraining the Jacobian matrix of encoder for regularization of auto-encoders [27], and enforcing a DNN to be a partial isometry [28].

An important theoretical aspect of DNNs is the effect of their architecture, e.g. depth and width, on their GE. Various measures such as the VC-dimension [29], [30], the Rademacher or Gaussian complexities [31] and algorithmic robustness [32] have been used to bound the GE in the context of DNNs. For example, the VC-dimension of DNN with the hard-threshold non-linearity is equal to the number of parameters in the network, which implies that the sample complexity is linear in the number of parameters of the network. The GE can also be bounded independently of the number of parameters, provided that the norms of the weight matrices (the network's linear components) are constrained appropriately. Such constraints are usually enforced by training networks with weight decay regularization, which is simply the ℓ_1- or ℓ_2-norm of all the weights in the network. For example, the work [33] studies the GE of DNN with ReLUs and constraints on the norms of the weight matrices. However, it provides GE bounds that scale exponentially with the network depth. Similar behaviour is also depicted in [34]. The authors in [32] show that DNNs are robust provided that the ℓ_1-norm of the weights in each layer is bounded. The bounds are exponential in the ℓ_1-norm of the weights if the norm is greater than 1.

The GE bounds in [30], [32], [33] suggest that the GE of a DNN is bounded only if the number of training samples grows with the DNN depth or size. However, in practice increasing network's depth or size often leads to a lower GE [4], [35]. Moreover, recent work in [36] shows that a 2 layer DNN with ReLUs may fit any function of n samples in d dimensions provided that it has $2n + d$ parameters, which is often the case in practice. They show that the nature of the GE depends more on the nature of the data than on the architecture of the network as the same network is able to fit both structured data and random data, where for the first the GE is very low and for the latter it is very large. The authors conclude that data agnostic measures such as the Rademacher complexity or VC-dimension are not adequate to explain the good generalization properties of modern DNN.

Our work complements the previous works on the GE of DNNs by bounding the GE in terms of the DNN classification margin, which is independent of the DNN depth and size, but takes into account the structure of the data (considering its covering number) and therefore avoids the issues presented above. The extension of our results to invariant DNN is provided in [37].

A. Contributions

In this work we focus on the GE of a multi-class DNN classifier with general non-linearities. We establish new GE bounds of DNN classifiers via their classification margin, i.e. the distance between the training sample and the non-linear decision boundary induced by the DNN classifier in the sample space. The work capitalizes on the algorithmic robustness framework in [32] to cast insight onto the generalization properties of DNNs. In particular, the use of this framework to understand the operation of DNNs involves various innovations, which include:

- We derive bounds for the GE of DNNs by lower bounding their classification margin. The lower bound of the classification margin is expressed as a function of the network's Jacobian matrix.
- Our approach includes a large class of DNNs. For example, we consider DNNs with the softmax layer at the network output; DNNs with various non-linearities such as the Rectified Linear Unit (ReLU), the sigmoid and the hyperbolic tangent; DNNs with pooling, such as down-sampling, average pooling and max-pooling; and networks with shortcut connections such as Residual Networks [4].
- Our analysis shows that the GE of a DNN can be bounded independently of its depth or width provided that the spectral norm of the Jacobian matrix in the neighbourhood of the training samples is bounded. We argue that this result gives a justification for a low GE of DNNs in practice. Moreover, it also provides an explanation for why training with the recently proposed weight normalization or batch normalization can lead to a small GE. In such networks the ℓ_2-norm of the weight matrices is fixed and ℓ_2-norm regularization does not apply. The analysis also leads to a novel Jacobian matrix-based regularizer, which can be applied to weight normalized or batch normalized networks.
- We provide a series of examples on the MNIST, CIFAR-10, LaRED and ImageNet datasets that validate our analysis and demonstrate the effectiveness of the Jacobian regularizer.

Our contributions differ from the existing works in many ways. In particular, the GE of DNNs has been studied via the algorithmic robustness framework in [32]. Their bounds are based on the per-unit ℓ_1-norm of the weight matrices, and the studied loss is not relevant for classification. Our analysis is much broader, as it aims at bounding the GE of 0-1 loss directly and also considers DNNs with pooling. Moreover, our bounds are a function of the network's Jacobian matrix and are tighter than the bounds based on the norms of the weight matrices.

The work in [28] shows that learning transformations that are locally isometric is robust and leads to a small GE. Though they apply the proposed technique to DNNs they do not show how the DNN architecture affects the GE as our work does.

The authors in [25] have observed that contractive DNNs with ReLUs trained with the hinge loss lead to a large classification margin. However, they do not provide any GE bounds. Moreover, their results are limited to DNNs with ReLUs, whereas our analysis holds for arbitrary non-linearities, DNNs with pooling and DNNs with the softmax layer.

The work in [27] is related to ours in the sense that it proposes to regularize auto-encoders by constraining the Frobenious norm of the encoder's Jacobian matrix. However, their work is more empirical and is less concerned with the classification margin or GE bounds. They use the Jacobian matrix to regularize the

encoder whereas we use the Jacobian matrix to regularize the entire DNN.

Finally, our DNN analysis, which is based on the network's Jacobian matrix, is also related to the concept of sensitivity analysis that has been applied to feature selection for SVM and neural networks [38], [39], and for the construction of radial basis function networks [40], since the spectral norm of the Jacobian matrix quantifies the sensitivity of DNN output with respect to the input perturbation.

B. Paper Organization

Section II introduces the problem of generalization error, including elements of the algorithmic robustness framework, and introduces DNN classifiers. Properties of DNNs are described in Section III. The bounds on the classification margin of DNNs and their implication for the GE of DNNs are discussed in Section IV. Generalizations of our results are discussed in Section V. Section VI presents experimental results. The paper is concluded in Section VII. The proofs are deferred to the Appendix.

C. Notation

We use the following notation in the sequel: matrices, column vectors, scalars and sets are denoted by boldface upper-case letters ($\mathbf{X}$), boldface lower-case letters ($\mathbf{x}$), italic letters (x) and calligraphic upper-case letters ($\mathcal{X}$), respectively. The convex hull of $\mathcal{X}$ is denoted by $\text{conv}(\mathcal{X})$. $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ denotes the identity matrix, $\mathbf{0}_{M\times N} \in \mathbb{R}^{M\times N}$ denotes the zero matrix and $\mathbf{1}_N \in \mathbb{R}^N$ denotes the vector of ones. The subscripts are omitted when the dimensions are clear from the context. $\mathbf{e}_k$ denotes the k-th basis vector of the standard basis in $\mathbb{R}^N$. $\|\mathbf{x}\|_2$ denotes the Euclidean norm of $\mathbf{x}$, $\|\mathbf{X}\|_2$ denotes the spectral norm of $\mathbf{X}$, and $\|\mathbf{X}\|_F$ denotes the Frobenious norm of $\mathbf{X}$. The i-th element of the vector $\mathbf{x}$ is denoted by $(\mathbf{x})_i$, and the element of the i-th row and j-th column of $\mathbf{X}$ is denoted by $(\mathbf{X})_{ij}$. The covering number of $\mathcal{X}$ with d-metric balls of radius ρ is denoted by $\mathcal{N}(\mathcal{X}; d, \rho)$.

II. Problem Statement

We start by describing the GE in the framework of statistical learning. Then, we dwell on the GE bounds based on the robustness framework by Xu and Manor [32]. Finally, we present the DNN architectures studied in this paper.

A. The Classification Problem and Its GE

We consider a classification problem, where we observe a vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^N$ that has a corresponding class label $y \in \mathcal{Y}$. The set $\mathcal{X}$ is called the input space, $\mathcal{Y} = \{1, 2, \ldots, N_\mathcal{Y}\}$ is called the label space and $N_\mathcal{Y}$ denotes the number of classes. The samples space is denoted by $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$ and an element of $\mathcal{S}$ is denoted by $s = (\mathbf{x}, y)$. We assume that samples from $\mathcal{S}$ are drawn according to a probability distribution P defined on $\mathcal{S}$. A training set of m samples drawn from P is denoted by $S_m = \{s_i\}_{i=1}^m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$. The goal of learning is to leverage the training set S_m to find a classifier $g(\mathbf{x})$ that provides a label estimate $\hat{y}$ given the input vector $\mathbf{x}$. In this work the classifier is a DNN, which is described in detail in Section II-C.

The quality of the classifier output is measured by the loss function $\ell(g(\mathbf{x}), y)$, which measures the discrepancy between the true label y and the estimated label $\hat{y} = g(\mathbf{x})$ provided by the classifier. Here we take the loss to be the 0-1 indicator function. Other losses such as the hinge loss or the categorical cross entropy loss are possible. The empirical loss of the classifier $g(\mathbf{x})$ associated with the training set and the expected loss of the classifier $g(\mathbf{x})$ are defined as

$$\ell_{\text{emp}}(g) = 1/m \sum_{s_i \in S_m} \ell\left(g(\mathbf{x}_i), y_i\right) \tag{1}$$

and

$$\ell_{\text{exp}}(g) = \mathbb{E}_{s\sim P}\left[\ell\left(g(\mathbf{x}), y\right)\right], \tag{2}$$

respectively. An important question, which occupies us throughout this work, is how well $l_{\text{emp}}(g)$ predicts $l_{\text{exp}}(g)$. The measure we use for quantifying the prediction quality is the difference between $l_{\text{exp}}(g)$ and $l_{\text{emp}}(g)$, which is called the *generalization error*:

$$\text{GE}(g) = \left|\ell_{\text{exp}}(g) - \ell_{\text{emp}}(g)\right|. \tag{3}$$

B. The Algorithmic Robustness Framework

In order to provide bounds to the GE for DNN classifiers we leverage the robustness framework [32], which is described next.

The algorithmic robustness framework provides bounds for the GE based on the robustness of a learning algorithm that learns a classifier g leveraging the training set S_m:

Definition 1 ([32]): Let S_m be a training set and $\mathcal{S}$ the sample space. A learning algorithm is $(K, \epsilon(S_m))$-robust if the sample space $\mathcal{S}$ can be partitioned into K disjoint sets denoted by $\mathcal{K}_k$, $k = 1, \ldots, K$,

$$\mathcal{K}_k \subseteq \mathcal{S}, \quad k = 1, \ldots, K, \tag{4}$$

$$\mathcal{S} = \cup_{k=1}^K \mathcal{K}_k, \tag{5}$$

$$\mathcal{K}_k \cap \mathcal{K}_{k'} = \emptyset, \quad \forall k \neq k', \tag{6}$$

such that for all $s_i \in S_m$ and all $s \in \mathcal{S}$

$$\begin{aligned} s_i &= (\mathbf{x}_i, y_i) \in \mathcal{K}_k \wedge s = (\mathbf{x}, y) \in \mathcal{K}_k \\ &\Longrightarrow \left|\ell(g(\mathbf{x}_i), y_i) - \ell(g(\mathbf{x}), y)\right| \leq \epsilon(S_m). \end{aligned} \tag{7}$$

Note that s_i is an element of the training set and s is an arbitrary element of the sample space $\mathcal{S}$. Therefore, a robust learning algorithm chooses a classifier g for which the losses of any s and s_i in the same partition $\mathcal{K}_k$ are close. The following theorem provides the GE bound for robust algorithms.[1]

Theorem 1 (Theorem 3 in [32]): If a learning algorithm is $(K, \epsilon(S_m))$-robust and $\ell(g(\mathbf{x}), y) \leq M$ for all $s = (\mathbf{x}, y) \in \mathcal{S}$, then for any $\delta > 0$, with probability at least $1 - \delta$,

$$\text{GE}(g) \leq \epsilon(S_m) + M\sqrt{\frac{2K\log(2) + 2\log(1/\delta)}{m}}. \tag{8}$$

[1] Additional variants of this theorem are provided in [32].

The first term in the GE bound in (8) is constant and depends on the training set S_m. The second term behaves as $\mathcal{O}(1/\sqrt{m})$ and vanishes as the size of the training set S_m approaches infinity. $M = 1$ in the case of 0-1 loss, and K corresponds to the number of partitions of the samples space $\mathcal{S}$.

A bound on the number of partitions K can be found by the covering number of the samples space $\mathcal{S}$. The covering number is the smallest number of (pseudo-)metric balls of radius ρ needed to cover $\mathcal{S}$, and it is denoted by $\mathcal{N}(\mathcal{S}; d, \rho)$, where d denotes the (pseudo-)metric.[2] The space $\mathcal{S}$ is the Cartesian product of a continuous input space $\mathcal{X}$ and a discrete label space $\mathcal{Y}$, and we can write $\mathcal{N}(\mathcal{S}; d, \rho) \leq N_{\mathcal{Y}} \cdot \mathcal{N}(\mathcal{X}; d, \rho)$, where $N_{\mathcal{Y}}$ corresponds to the number of classes. The choice of metric d determines how efficiently one may cover $\mathcal{X}$. A common choice is the Euclidean metric

$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2, , \quad \mathbf{x}, \mathbf{x}' \in \mathcal{X}\,, \tag{9}$$

which we also use in this paper. The covering number of many structured low-dimensional data models can be bounded in terms of their "intrinsic" properties, for example:

- a Gaussian mixture model (GMM) with L Gaussians and covariance matrices of rank at most k leads to a covering number $\mathcal{N}(\mathcal{X}; d, \rho) = L(1 + 2/\rho)^k$ [41];
- k-sparse signals in a dictionary with L atoms have a covering number $\mathcal{N}(\mathcal{X}; d, \rho) = \binom{L}{k}(1 + 2/\rho)^k$ [16];
- C_M regular k-dimensional manifold, where C_M is a constant that captures its "intrinsic" properties, has a covering number $\mathcal{N}(\mathcal{X}; d, \rho) = (\frac{C_M}{\rho})^k$ [42].

1) Large Margin Classifier: An example of a robust learning algorithm is the large margin classifiers, which we consider in this work. The classification margin is defined as follows:

Definition 2 (Classification margin): The classification margin of a training sample $s_i = (\mathbf{x}_i, y_i)$ measured by a metric d is defined as

$$\gamma^d(s_i) = \sup\{a : d(\mathbf{x}_i, \mathbf{x}) \leq a \Longrightarrow g(\mathbf{x}) = y_i \,\forall \mathbf{x}\}\,. \tag{10}$$

The classification margin of a training sample s_i is the radius of the largest metric ball (induced by d) in $\mathcal{X}$ centered at $\mathbf{x}_i$ that is contained in the decision region associated with class label y_i. The robustness of large margin classifiers is given by the following Theorem.

Theorem 2 (Adapted from Example 9 in [32]): If there exists γ such that

$$\gamma^d(s_i) > \gamma > 0 \quad \forall s_i \in S_m\,, \tag{11}$$

then the classifier $g(\mathbf{x})$ is $(N_{\mathcal{Y}} \cdot \mathcal{N}(\mathcal{X}; d, \gamma/2), 0)$-robust.

Theorems 1 and 2 imply that the GE of a classifier with margin γ is upper bounded by (neglecting the $\log(1/\delta)$ term in (8))

$$\mathrm{GE}(g) \lesssim \frac{1}{\sqrt{m}}\sqrt{2\log(2) \cdot N_{\mathcal{Y}} \cdot \mathcal{N}(\mathcal{X}; d, \gamma/2)}\,. \tag{12}$$

Note that in case of a large margin classifier the constant $\epsilon(S_m)$ in (8) is equal to 0, and the GE approaches zero at a rate $\sqrt{m}$ as the number of training samples grows. The GE also increases sub-linearly with the number of classes $N_{\mathcal{Y}}$. Finally, the GE depends on the complexity of the input space $\mathcal{X}$ and the classification margin via the covering number $\mathcal{N}(\mathcal{X}; d, \gamma/2)$.

For example, if we take $\mathcal{X}$ to be a C_M regular k-dimensional manifold then the upper bound to the GE behaves as:

Corollary 1: Assume that $\mathcal{X}$ is a (subset of) C_M regular k-dimensional manifold, where $\mathcal{N}(\mathcal{X}; d, \rho) \leq (\frac{C_M}{\rho})^k$. Assume also that classifier $g(\mathbf{x})$ achieves a classification margin γ and take $\ell(g(\mathbf{x}_i), y_i)$ to be the 0-1 loss. Then for any $\delta > 0$, with probability at least $1 - \delta$,

$$\mathrm{GE}(g) \leq \sqrt{\frac{\log(2) \cdot N_{\mathcal{Y}} \cdot 2^{k+1} \cdot (C_M)^k}{\gamma^k m}} + \sqrt{\frac{2\log(1/\delta)}{m}}\,. \tag{13}$$

Proof: The proof follows directly from Theorems 1 and 2. ∎

Note that the role of the classifier is captured via the achieved classification margin γ. If we can always ensure a classification margin $\gamma = 1$, then the GE bound only depends on the dimension of the manifold k and the manifold constant C_M. We relate this bound, in the context of DNNs, to other bounds in the literature in Section IV.

C. Deep Neural Network Classifier

The DNN classifier is defined as

$$g(\mathbf{x}) = \underset{i \in [N_{\mathcal{Y}}]}{\arg\max} (f(\mathbf{x}))_i\,, \tag{14}$$

where $(f(\mathbf{x}))_i$ is the i-th element of the $N_{\mathcal{Y}}$ dimensional output of a DNN $f : \mathbb{R}^N \to \mathbb{R}^{N_{\mathcal{Y}}}$. We assume that $f(\mathbf{x})$ is composed of L layers:

$$f(\mathbf{x}) = \phi_L(\phi_{L-1}(\cdots \phi_1(\mathbf{x}, \theta_1), \cdots \theta_{L-1}), \theta_L)\,, \tag{15}$$

where $\phi_l(\cdot, \theta_l)$ represents the l-th layer with parameters θ_l, $l = 1, \ldots, L$. The output of the l-th layer is denoted by $\mathbf{z}^l$, i.e. $\mathbf{z}^l = \phi_l(\mathbf{z}^{l-1}, \theta_l)$, $\mathbf{z}^l \in \mathbb{R}^{M_l}$; the input layer corresponds to $\mathbf{z}^0 = \mathbf{x}$; and the output of the last layer is denoted by $\mathbf{z} = f(\mathbf{x})$. Such a DNN is visualized in Fig. 1. Next, we define various layers $\phi_l(\cdot, \theta_l)$ that are used in the modern state-of-the-art DNNs.

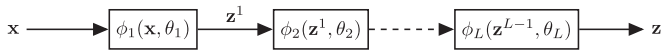


Fig. 1. DNN transforms the input vector $\mathbf{x}$ to the feature vector $\mathbf{z}$ by a series of (non-linear) transforms.

1) Linear and Softmax Layers: We start by describing the last layer of a DNN that maps the output of previous layer into $\mathbb{R}^{N_{\mathcal{Y}}}$, where $N_{\mathcal{Y}}$ corresponds to the number of classes.[3] This layer can be linear:

$$\mathbf{z} = \hat{\mathbf{z}}, \quad \hat{\mathbf{z}} = \mathbf{W}_L \mathbf{z}^{L-1} + \mathbf{b}_L\,, \tag{16}$$

where $\mathbf{W}_L \in \mathbb{R}^{N_{\mathcal{Y}} \times M_{L-1}}$ is the weight matrix associated with the last layer and $\mathbf{b} \in \mathbb{R}^{N_{\mathcal{Y}}}$ is the bias vector associated with the last layer. Note that according to (14), the i-th row of $\mathbf{W}_L$ can

[2] Note that we can always obtain a set of disjoint partitions from the set of metric balls used to construct the covering.

[3] Assuming that there are $N_{\mathcal{Y}}$ one-vs.-all classifiers.

TABLE I
POINT-WISE NON-LINEARITIES

Name	Function: $\sigma(x)$	Derivative: $\frac{d}{dx}\sigma(x)$	Derivative bound: $\sup_x \left\lvert \frac{d}{dx}\sigma(x) \right\rvert$
ReLU	$\max(x, 0)$	$\{1 \text{ if } x > 0; 0 \text{ if } x \leq 0\}$	≤ 1
Sigmoid	$\frac{1}{1+e^{-x}}$	$\sigma(x)(1-\sigma(x)) = \frac{e^{-x}}{(1+e^{-x})^2}$	$\leq \frac{1}{4}$
Hyperbolic tangent	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$1-\sigma(x)^2$	≤ 1

be interpreted as a normal to the hyperplane that separates class i from the others. If the last layer is linear the usual choice of learning objective is the hinge loss.

A more common choice for the last layer is the softmax layer:

$$\mathbf{z} = \zeta(\hat{\mathbf{z}}) = e^{\hat{\mathbf{z}}} / \left(\mathbf{1}^T e^{\hat{\mathbf{z}}}\right), \quad \hat{\mathbf{z}} = \mathbf{W}_L \mathbf{z}^{L-1} + \mathbf{b}_L, \tag{17}$$

where $\zeta(\cdot)$ is the softmax function and $\mathbf{W}_L$ and $\mathbf{b}_L$ are the same as in (16). Note that the exponential is applied element-wise. The elements of $\mathbf{z}$ are in range $(0, 1)$ and are often interpreted as "probabilites" associated with the corresponding class labels. The decision boundary between class y_1 and class y_2 corresponds to the hyperplane $\{\mathbf{z} : (\mathbf{z})_{y_1} = (\mathbf{z})_{y_2}\}$. The softmax layer is usually coupled with categorical cross-entropy training objective.

For the remainder of this work we will take the softmax layer as the last layer of DNN, but note that all results still apply if the linear layer is used.

2) Non-Linear Layers: A non-linear layer is defined as

$$\mathbf{z}^l = [\hat{\mathbf{z}}^l]_\sigma = [\mathbf{W}_l \mathbf{z}^{l-1} + \mathbf{b}_l]_\sigma, \tag{18}$$

where $[\hat{\mathbf{z}}^l]_\sigma$ represents the element-wise non-linearity applied to each element of $\hat{\mathbf{z}}^l \in \mathbb{R}^{M_l}$, and $\hat{\mathbf{z}}^l$ represents the linear transformation of the layer input: $\hat{\mathbf{z}}^l = \mathbf{W}_l \mathbf{z}^{l-1} + \mathbf{b}_l$. $\mathbf{W}_l \in \mathbb{R}^{M_l \times M_{l-1}}$ is the weight matrix and $\mathbf{b}_l \in \mathbb{R}^{M_l}$ is the bias vector. The typical non-linearities are the ReLU, the sigmoid and the hyperbolic tangent. They are listed in Table I. The choice of non-linearity σ is usually the same for all the layers in the network.

Note that the non-linear layer in (18) includes the convolutional layers which are used in the convolutional neural networks. In that case the weight matrix is block-cyclic.

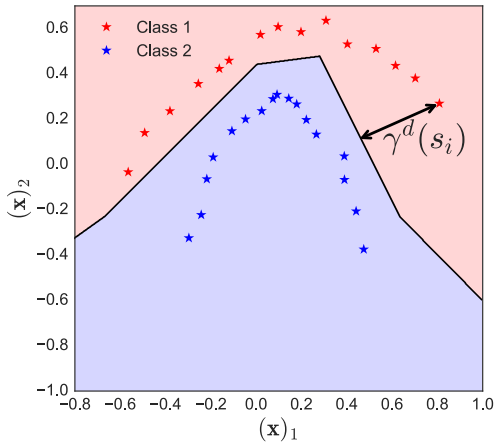
3) Pooling Layers: A pooling layer reduces the dimension of intermediate representation and is defined as

$$\mathbf{z}^l = \mathbf{P}^l(\mathbf{z}^{l-1})\mathbf{z}^{l-1}, \tag{19}$$

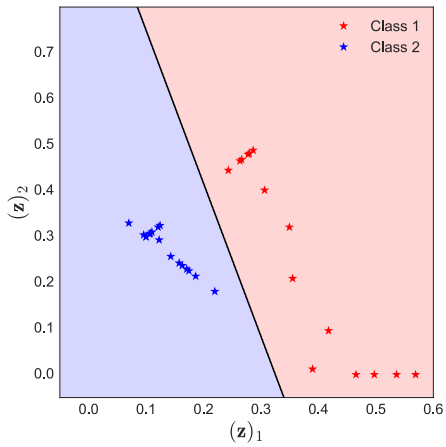
where $\mathbf{P}^l(\mathbf{z}^{l-1})$ is the pooling matrix. The usual choices of pooling are down-sampling, max-pooling and average pooling. We denote by $\mathbf{p}_i^l(\mathbf{z}^{l-1})$ the i-th row of $\mathbf{P}^l(\mathbf{z}^{l-1})$ and assume that there are M_l pooling regions $\mathcal{P}_i, i = 1, \ldots, M_l$. In the case of down-sampling $\mathbf{p}_i^l(\mathbf{z}^{l-1}) = \mathbf{e}_{\mathcal{P}_i(1)}$, where $\mathcal{P}_i(1)$ is the first element of the pooling region $\mathcal{P}_i$; in the case of max-pooling $\mathbf{p}_i^l(\mathbf{z}^{l-1}) = \mathbf{e}_{j^\star}$, where $j^\star = \arg\max_{j' \in \mathcal{P}_i} |(\mathbf{z}^{l-1})_{j'}|$; and in the case of average pooling $\mathbf{p}_i^l(\mathbf{z}^{l-1}) = \frac{1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} \mathbf{e}_j$.

III. THE GEOMETRICAL PROPERTIES OF DEEP NEURAL NETWORKS

The classification margin introduced in Section II-A is a function of the decision boundary in the input space. This is visualized in Fig. 2(a). However, a training algorithm usually optimizes the decision boundary at the network output (Fig. 2(b)),



(a) Input space.



(b) Output space.

Fig. 2. Decision boundaries in the input space and in the output space. Plot (a) shows samples of class 1 and 2 and the decision regions produced by a two-layer network projected into the input space. Plot (b) shows the samples transformed by the network and the corresponding decision boundary at the network output.

which does not necessarily imply a large classification margin. In this section we introduce a general approach that allows us to bound the expansion of distances between the network input and its output. In Section IV we use this to establish bounds of the classification margin and the GE bounds that are independent of the network depth or width.

We start by defining the Jacobian matrix (JM) of the DNN $f(\mathbf{x})$:

$$\mathbf{J}(\mathbf{x}) = \frac{df(\mathbf{x})}{d\mathbf{x}} = \prod_{l=1}^{L} \frac{d\phi_l(\mathbf{z}^{l-1})}{d\mathbf{z}^{l-1}} \cdot \frac{d\phi_1(\mathbf{x})}{d\mathbf{x}}. \tag{20}$$

Note that by the properties of the chain rule, the JM is computed as the product of the JMs of the individual network layers,

evaluated at the appropriate values of the layer inputs $\mathbf{x}, \mathbf{z}^1, \ldots, \mathbf{z}^{L-1}$. We use the JM to establish a relation between a pair of vectors in the input space and the output space.

Theorem 3: For any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and a DNN $f(\cdot)$, we have

$$f(\mathbf{x}') - f(\mathbf{x}) = \int_0^1 \mathbf{J}(\mathbf{x} + t(\mathbf{x}' - \mathbf{x}))\, dt\, (\mathbf{x}' - \mathbf{x}) \quad (21)$$

$$= \mathbf{J}_{\mathbf{x},\mathbf{x}'}(\mathbf{x}' - \mathbf{x}), \quad (22)$$

where

$$\mathbf{J}_{\mathbf{x},\mathbf{x}'} = \int_0^1 \mathbf{J}(\mathbf{x} + t(\mathbf{x}' - \mathbf{x}))\, dt \quad (23)$$

is the average Jacobian on the line segment between $\mathbf{x}$ and $\mathbf{x}'$.

Proof: The proof appears in Appendix A. ■

As a direct consequence of Theorem 3 we can bound the distance expansion between $\mathbf{x}$ and $\mathbf{x}'$ at the output of the network $f(\cdot)$:

Corollary 2: For any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and a DNN $f(\cdot)$, we have

$$\|f(\mathbf{x}') - f(\mathbf{x})\|_2 = \|\mathbf{J}_{\mathbf{x},\mathbf{x}'}(\mathbf{x}' - \mathbf{x})\|_2 \leq \sup_{\mathbf{x}'' \in \text{conv}(\mathcal{X})} \|\mathbf{J}(\mathbf{x}'')\|_2 \|\mathbf{x}' - \mathbf{x}\|_2 . \quad (24)$$

Proof: The proof appears in Appendix B. ■

Note that we have established that $\mathbf{J}_{\mathbf{x},\mathbf{x}'}$ corresponds to a linear operator that maps the vector $\mathbf{x}' - \mathbf{x}$ to the vector $f(\mathbf{x}') - f(\mathbf{x})$. This implies that the maximum distance expansion of the network $f(\mathbf{x})$ is bounded by the maximum spectral norm of the network's JM. Moreover, the JM of $f(\mathbf{x})$ corresponds to the product of JMs of all the layers of $f(\mathbf{x})$ as shown in (20). It is possible to calculate the JMs of all the layers defined in Section II-C:

1) Jacobian Matrix of Linear and Softmax Layers: The JM of the linear layer defined in (16) is equal to the weight matrix

$$\frac{d\mathbf{z}}{d\mathbf{z}^{L-1}} = \mathbf{W}_L . \quad (25)$$

Similarly, in the case of softmax layer defined in (17) the JM is

$$\frac{d\mathbf{z}}{d\mathbf{z}^{L-1}} = \frac{d\mathbf{z}}{d\hat{\mathbf{z}}} \cdot \frac{d\hat{\mathbf{z}}}{d\mathbf{z}^{L-1}} = \left(-\zeta(\hat{\mathbf{z}})\zeta(\hat{\mathbf{z}})^T + \text{diag}(\zeta(\hat{\mathbf{z}})\right) \cdot \mathbf{W}_L . \quad (26)$$

Note that $\left(-\zeta(\hat{\mathbf{z}})\zeta(\hat{\mathbf{z}})^T + \text{diag}(\zeta(\hat{\mathbf{z}})\right)$ corresponds to the JM of the softmax function $\zeta(\hat{\mathbf{z}})$.

2) Jacobian Matrix of Non-Linear Layers: The JM of the non-linear layer (18) can be derived in the same way as the JM of the softmax layer. We first define the JM of the point-wise non-linearity, which is a diagonal matrix[4]

$$\left(\frac{d\mathbf{z}^l}{d\hat{\mathbf{z}}^l}\right)_{ii} = \frac{d\sigma\left((\hat{\mathbf{z}}^l)_i\right)}{d(\hat{\mathbf{z}}^l)_i}, \quad i = 1, \ldots, M_l . \quad (27)$$

The derivatives associated with various non-linearities are provided in Table I. The JM of the non-linear layer can be expressed as

$$\frac{d\mathbf{z}^l}{d\hat{\mathbf{z}}^{l-1}} = \frac{d\mathbf{z}^l}{d\hat{\mathbf{z}}^l} \cdot \mathbf{W}_l . \quad (28)$$

3) Jacobian Matrix of Pooling Layers: The pooling operator defined in (19) is linear or a piece-wise linear operator. The corresponding JM is therefore also linear or piece-wise linear and is equal to:

$$\mathbf{P}^l(\mathbf{z}^{l-1}) . \quad (29)$$

The following Lemma collects the bounds on the spectral norm of the JMs for all the layers defined in Section II-C.

Lemma 1: The following statements hold:

(1) The spectral norm of JMs of the linear layer in (16), the softmax layer in (17) and non-linear layer in (18) with the ReLU, Sigmoid or Hyperbolic tangent non-linearities is upper bounded by

$$\left\| \frac{d\mathbf{z}^l}{d\hat{\mathbf{z}}^{l-1}} \right\|_2 \leq \|\mathbf{W}_l\|_2 \leq \|\mathbf{W}_l\|_F . \quad (30)$$

(2) Assume that the pooling regions of the down-sampling, max-pooling and average pooling operators are non-overlapping. Then the spectral norm of their JMs can be upper bounded by

$$\left\| \frac{d\mathbf{z}^l}{d\hat{\mathbf{z}}^{l-1}} \right\|_2 \leq 1 . \quad (31)$$

Proof: The proof appears in Appendix C. ■

Lemma 1 shows that the spectral norms of all layers can be bounded in terms of their weight matrices. As a consequence, the spectral norm of the JM is bounded by the product of the spectral norms of the weight matrices. We leverage this facts to provide GE bounds in the next section.

We also briefly explore a relationship between the Jacobian matrix and the Fisher information matrix. To simplify the derivations we assume $M = 1$, $N = 1$, $\mathbf{x}' = \mathbf{x} + \theta\mathbf{n}$ and $\mathbf{n} \sim \mathcal{N}(0, 1)$, where θ is the model parameter and $\mathbf{x}$ is deterministic. The Fisher information $F(\theta)$ measures how much information about the parameter θ is contained in the random variable $\mathbf{y} = f(\mathbf{x}')$, where f represents a DNN. In this particular case the Fisher information is given as

$$F(\theta) = \mathbb{E}_{\mathbf{n}}\left[\left(\frac{d\log f(\mathbf{x}')}{d\theta}\right)^2\right] = \mathbb{E}_{\mathbf{n}}\left[\left(\frac{d\log f(\mathbf{x}')}{df(\mathbf{x}')}\frac{df(\mathbf{x}')}{d\mathbf{x}'}\frac{d\mathbf{x}'}{\theta}\right)^2\right] = \mathbb{E}_{\mathbf{n}}\left[\left(\frac{d\log f(\mathbf{x}')}{df(\mathbf{x}')}\mathbf{J}(\mathbf{x}')\mathbf{n}\right)^2\right] . \quad (32)$$

In our setup the parameter θ can be interpreted as a magnitude of the input perturbation. It is clear from (32) that a small norm of the Jacobian matrix leads to a small Fisher information, which indicates that the distribution of $\mathbf{y}$ is not very informative about the parameters θ. By ensuring that the norm of the Jacobian is small we then naturally endow the network with robustness against perturbations of the input.

[4]Note that in case of ReLU the derivative of $max(x, 0)$ is not defined for $x = 0$, and we need to use subderivatives (or subgradients) to define the JM. We avoid this technical complication and simply take the derivative of $max(x, 0)$ to be 0 when $x = 0$. Note that this does not change the results in any way because the subset of $\mathcal{X}$ for which the derivatives are not defined has zero measure.

IV. Generalization Error of a Deep Neural Network Classifier

In this section we provide the classification margin bounds for DNN classifiers that allow us to bound the GE. We follow the common practice and assume that the networks are trained by a loss that promotes separation of different classes at the network output, e.g. categorical cross entropy loss or the hinge loss. In other words, the training aims at maximizing the score of each training sample, where the score is defined as follows.

Definition 3 (Score): Take score of a training sample $s_i = (\mathbf{x}_i, y_i)$

$$o(s_i) = \min_{j \neq y_i} \sqrt{2}(\boldsymbol{\delta}_{y_i} - \boldsymbol{\delta}_j)^T f(\mathbf{x}_i)\,, \tag{33}$$

where $\boldsymbol{\delta}_i \in \mathbb{R}^{N_\mathcal{Y}}$ is the Kronecker delta vector with $(\boldsymbol{\delta}_i)_i = 1$.

Recall the definition of the classifier $g(\mathbf{x})$ in (14) and note that the decision boundary between class i and class j in the feature space $\mathcal{Z}$ is given by the hyperplane $\{\mathbf{z} : (\mathbf{z})_i = (\mathbf{z}_j)\}$. A positive score indicates that at the network output, classes are separated by a margin that corresponds to the score. However, a large score $o(s_i)$ does not necessarily imply a large classification margin $\gamma^d(s_i)$. Theorem 4 provides classification margin bounds expressed as a function of the score and the properties of the network.

Theorem 4: Assume that a DNN classifier $g(\mathbf{x})$, as defined in (14), classifies a training sample $\mathbf{x}_i$ with the score $o(s_i) > 0$. Then the classification margin can be bounded as

$$\gamma^d(s_i) \geq \frac{o(s_i)}{\sup_{\mathbf{x}:\|\mathbf{x}-\mathbf{x}_i\|_2 \leq \gamma^d(s_i)} \|\mathbf{J}(\mathbf{x})\|_2} \triangleq \gamma_1^d(s_i) \tag{34}$$

$$\geq \frac{o(s_i)}{\sup_{\mathbf{x} \in \text{conv}(\mathcal{X})} \|\mathbf{J}(\mathbf{x})\|_2} \triangleq \gamma_2^d(s_i) \tag{35}$$

$$\geq \frac{o(s_i)}{\prod_{\mathbf{W}_l \in \mathcal{W}} \|\mathbf{W}_l\|_2} \triangleq \gamma_3^d(s_i) \tag{36}$$

$$\geq \frac{o(s_i)}{\prod_{\mathbf{W}_l \in \mathcal{W}} \|\mathbf{W}_l\|_F} \triangleq \gamma_4^d(s_i)\,, \tag{37}$$

where $\mathcal{W}$ is the set of all weight matrices of $f(\mathbf{x})$.

Proof: The proof appears in Appendix D. ■

Given the bounds of the classification margin we can specialize Corollary 1 to DNN classifiers.

Corollary 3: Assume that $\mathcal{X}$ is a (subset of) C_M regular k-dimensional manifold, where $\mathcal{N}(\mathcal{X}; d, \rho) \leq (\frac{C_M}{\rho})^k$. Assume also that DNN classifier $g(\mathbf{x})$ achieves a lower bound to the classification margin $\gamma_b^d(s_i) > \gamma_b$ for $b \in \{1,2,3,4\}, \forall s_i \in S_m$ and take $\ell(g(\mathbf{x}_i), y_i)$ to be the 0-1 loss. Then for any $\delta > 0$, with probability at least $1 - \delta$,

$$\text{GE}(g) \leq \sqrt{\frac{\log(2) \cdot N_\mathcal{Y} \cdot 2^{k+1} \cdot (C_M)^k}{\gamma_b^k m}} + \sqrt{\frac{2\log(1/\delta)}{m}}. \tag{38}$$

Proof: The proof follows from Theorems 1, 2 and 4. ■

Corollary 3 suggests that the GE will be bounded by $C\frac{1}{\sqrt{m}}\gamma^{-k/2}$, where $C = \sqrt{\log(2) \cdot N_\mathcal{Y} 2^{k+1} (C_M)^k}$, provided that the classification margin bounds satisfy $\gamma_b^d(s_i) > \gamma$ for some $b \in \{1,2,3,4\}, \forall s_i \in S_m$.

We now leverage the classification margin bounds in Theorem 4 to construct constraint sets $\mathcal{W}_b = \{\mathbf{W}_l \in \mathcal{W} : \gamma_b^d(s_i) > \gamma \forall s_i\}$, $b \in 1,2,3,4$ such that $\mathcal{W} \in \mathcal{W}_b$ ensures that the GE is bounded by $C\frac{1}{\sqrt{m}}\gamma^{-k/2}$. Using (34)-(37) we obtain

$$\mathcal{W}_1 = \Bigg\{ \mathbf{W}_l \in \mathcal{W} : \sup_{\mathbf{x}:\|\mathbf{x}-\mathbf{x}_i\|_2 \leq \gamma^d(s_i)} \|\mathbf{J}(\mathbf{x})\|_2 < \gamma \cdot o(s_i) \quad \forall s_i = (\mathbf{x}_i, y_i) \Bigg\}, \tag{39}$$

$$\mathcal{W}_2 = \Bigg\{ \mathbf{W}_l \in \mathcal{W} : \sup_{\mathbf{x} \in \text{conv}(\mathcal{X})} \|\mathbf{J}(\mathbf{x})\|_2 < \gamma \cdot o(s_i) \quad \forall s_i = (\mathbf{x}_i, y_i) \Bigg\}, \tag{40}$$

$$\mathcal{W}_3 = \Bigg\{ \mathbf{W}_l \in \mathcal{W} : \prod_{\mathbf{W}_l \in \mathcal{W}} \|\mathbf{W}_l\|_2 < \gamma \cdot o(s_i) \quad \forall s_i = (\mathbf{x}_i, y_i) \Bigg\}, \tag{41}$$

$$\mathcal{W}_4 = \Bigg\{ \mathbf{W}_l \in \mathcal{W} : \prod_{\mathbf{W}_l \in \mathcal{W}} \|\mathbf{W}_l\|_F < \gamma \cdot o(s_i) \quad \forall s_i = (\mathbf{x}_i, y_i) \Bigg\}. \tag{42}$$

Note that while we want to maximize the score $o(s_i)$, we also need to constrain the network's Jacobian matrix $\mathbf{J}(\mathbf{x})$ (following $\mathcal{W}_1$ and $\mathcal{W}_2$) or the weight matrices $\mathbf{W}_l \in \mathcal{W}$ (following $\mathcal{W}_3$ and $\mathcal{W}_4$). This stands in line with the common training rationale of DNN in which we do not only aim at maximizing the score of the training samples to ensure a correct classification of the training set, but also have a regularization that constrains the network parameters, where this combination eventually leads to a lower GE. The constraint sets in (39)–(42) impose different regularization techniques:

- The term $\sup_{\mathbf{x}:\|\mathbf{x}-\mathbf{x}_i\|_2 \leq \gamma^d(s_i)} \|\mathbf{J}(\mathbf{x})\|_2 < \gamma \cdot o(s_i)$ in (39) considers only the supremum of the spectral norm of the Jacobian matrix evaluated at the points within $\mathcal{N}_i = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_i\|_2 \leq \gamma^d(s_i)\}$, where $\gamma^d(s_i)$ is the classification margin of training sample s_i (see Definition 2). We can not compute the margin $\gamma^d(s_i)$, but can still obtain a rationale for regularization: as long as the spectral norm of the Jacobian matrix is bounded in the neighbourhood of a training sample $\mathbf{x}_i$ given by $\mathcal{N}_i$ we will have the GE guarantees.
- The constraint on the Jacobian matrix $\sup_{\mathbf{x} \in \text{conv}(\mathcal{X})} \|\mathbf{J}(\mathbf{x})\|_2 < \gamma \cdot o(s_i)$ in (40) is more restrictive as it requires bounded spectral norm for all samples $\mathbf{x}$ in the convex hull of the input space $\mathcal{X}$.
- The constraints in (41) and (42) are of similar form, $\prod_{\mathbf{W}_l \in \mathcal{W}} \|\mathbf{W}_l\|_2 < \gamma \cdot o(s_i)$ and $\prod_{\mathbf{W}_l \in \mathcal{W}} \|\mathbf{W}_l\|_F < \gamma \cdot o(s_i)$, respectively. Note that the weight decay, which aims at bounding the Frobenious norms of the weight matrices

might be used to satisfy the constrains in (42). However, note also that the bound based on the spectral norm in (41) is tighter than one based on the Frobenious norm in (42). For example, take $\mathbf{W}_l \in \mathcal{W}$ to have orthonormal rows and be of dimension $M \times M$. Then the constraint in (41), which is based on the spectral norm, is of the form $1 < \gamma o(s_i)$ and the constraint in (42), which is based on the Frobenious norm, is $M^{L/2} < \gamma o(s_i)$. In the former case we have a constraint on the score, which is independent of the network width or depth. In the latter the constraint on the output score is exponential in network depth and polynomial in network width. The difference is that the Frobenious norm does not take into account the correlation (angles) between the rows of the weight matrix $\mathbf{W}_l$, while the spectral norm does. Therefore, the bound based on the Frobenious norm corresponds to the worst case when all the rows of $\mathbf{W}_l$ are aligned. In that case $\|\mathbf{W}_l\|_F = \|\mathbf{W}_l\|_2 = \sqrt{M}$. On the other hand, if the rows of $\mathbf{W}_l$ are orthonormal $\|\mathbf{W}_l\|_F = \sqrt{M}$, but $\|\mathbf{W}_l\|_2 = 1$.

Remark 1: To put results into perspective we compare our GE bounds to the GE bounds based on the Rademacher complexity in [33], which hold for DNNs with ReLUs. The work in [33] shows that if

$$\mathcal{W} \in \mathcal{W}_F = \left\{ \mathbf{W}_i \in \mathcal{W} : \prod_{i=1}^{L} \|\mathbf{W}_i\|_F < C_F \right\} \tag{43}$$

and the energy of training samples is bounded then:

$$GE(g) \lessapprox \frac{1}{\sqrt{m}} 2^{L-1} C_F \,. \tag{44}$$

Although the bounds (38) and (44) are not directly comparable, since the bounds based on the robustness framework rely on an underlying assumption on the data (covering number), there is still a remarkable difference between them. The behaviour in (44) suggests that the GE grows exponentially with the network depth even if the product of the Frobenious norms of all the weight matrices is fixed, which is due to the term 2^L. The bound in (34) and the constraint sets in (39)–(42), on the other hand, imply that the GE does not increase with the number of layers provided that the spectral/Frobenious norms of the weight matrices are bounded. Moreover, if we take the DNN to have weight matrices with orthonormal rows then the GE behaves as $\frac{1}{\sqrt{m}}(C_M)^{k/2}$ (assuming $o(s_i) \geq 1, i = 1, \ldots, m$), and therefore relies only on the complexity of the underlying data manifold and not on the network depth. This provides a possible answer to the open question of [33] that depth independent capacity control is possible in DNNs with ReLUs.

Remark 2: An important value of our bounds is that they provide an additional explanation to the success of state-of-the-art DNN training techniques such as batch normalization [23] and eight normalization [24].

Weight normalized DNNs have weight matrices with normalized rows, i.e.

$$\mathbf{W}_l = \operatorname{diag}(\hat{\mathbf{W}}_l^T \hat{\mathbf{W}}_l)^{-1} \hat{\mathbf{W}}_l \,, \tag{45}$$

where $\operatorname{diag}(\cdot)$ denotes the diagonal part of the matrix. While the main motivation for this method is a faster training, the authors also show empirically that such networks achieve good generalization. Note that for row-normalized weight matrices $\|\mathbf{W}_l\|_F = \sqrt{M_l}$ and therefore the bounds based on the Frobenious norm can not explain the good generalization of such networks as adding layers or making $\mathbf{W}_l$ larger will lead to a larger GE bound. However, our bound in (34) and the constraint sets in (39)–(41) show that a small Frobenious norm of the weight matrices is not crucial for a small GE. A supporting experiment is presented in Section VI-A2.

We also note that batch normalization also leads to row-normalized weight matrices in DNNs with ReLUs:[5]

Theorem 5: Assume that the non-linear layers of a DNN with ReLUs are batch normalized as:

$$\mathbf{z}^{l+1} = [\mathbf{N}\left(\{\mathbf{z}_i^l\}_{i=1}^m, \mathbf{W}_l\right) \hat{\mathbf{z}^l}]_\sigma, \quad \hat{\mathbf{z}}^l = \mathbf{W}_l \mathbf{z}^l \,, \tag{46}$$

where σ denotes the ReLU non-linearity and

$$\mathbf{N}\left(\{\mathbf{z}_i\}_{i=1}^m, \mathbf{W}\right) = \operatorname{diag}\left(\sum_{i=1}^{m} \mathbf{W}\mathbf{z}_i\mathbf{z}_i^T\mathbf{W}^T\right)^{-\frac{1}{2}} \tag{47}$$

is the normalization matrix. Then all the weight matrices are row normalized. The exception is the weight matrix of the last layer, which is of the form $\mathbf{N}(\{\mathbf{z}_i^{L-1}\}_{i=1}^m, \mathbf{W}_L)\mathbf{W}_L$.

Proof: The proof appears in Appendix E. ■

A. Jacobian Regularizer

The constraint set (39) suggests that we can regularize the DNN by bounding the norm of the network's JM for the inputs close to $\mathbf{x}_i$. Therefore, we propose to penalize the norm of the network's JM evaluated at each training sample $\mathbf{x}_i$,

$$R_J(\mathcal{W}) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{J}(\mathbf{x}_i)\|_2^2 \,. \tag{48}$$

The implementation of such regularizer requires computation of its gradients or subgradients. In this case the computation of the subgradient of the spectral norm requires the calculation of a SVD decomposition [44], which makes the proposed regularizer inefficient. To circumvent this, we propose a surrogate regularizer based on the Frobenious norm of the Jacobian matrix:

$$R_F(\mathcal{W}) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{J}(\mathbf{x}_i)\|_F^2. \tag{49}$$

Note that the Frobenious norm and the spectral norm are related as follows: $1/\operatorname{rank}(\mathbf{J}(\mathbf{x}_i))\|\mathbf{J}(\mathbf{x}_i)\|_F^2 \leq \|\mathbf{J}(\mathbf{x}_i)\|_2^2 \leq \|\mathbf{J}(\mathbf{x}_i)\|_F^2$, which justifies using the surrogate regularizer. We will refer to $R_F(\mathcal{W})$ as the Jacobian regularizer.

[5]To simplify the derivation we omit the bias vectors and therefore also the centering applied by the batch normalization. This does not affect the generality of the result. We also follow [43] and omit the batch normalization scaling, as it can be included into the weight matrix of the layer following the batch normalization. We also omit the regularization term and assume that the matrices are invertible.

1) Computation of Gradients and Efficient Implementation: Note that the k-th row of $\mathbf{J}(\mathbf{x}_i)$ corresponds to the gradient of $(f(\mathbf{x}))_k$ with respect to the input $\mathbf{x}$ evaluated at $\mathbf{x}_i$. It is denoted by $\mathbf{g}_k(\mathbf{x}_i) = \frac{d(f(\mathbf{x}))_k}{d\mathbf{x}}|_{\mathbf{x}=\mathbf{x}_i}$. Now we can write

$$R_F(\mathcal{W}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{M}\mathbf{g}_k(\mathbf{x}_i)\mathbf{g}_k(\mathbf{x}_i)^T. \tag{50}$$

As the regularizer will be minimized by a gradient descent algorithm we need to compute its gradient with respect to the DNN parameters. First, we express $\mathbf{g}_k(\mathbf{x}_i)$ as

$$\mathbf{g}_k(\mathbf{x}_i) = \mathbf{g}_k^l(\mathbf{x}_i)\mathbf{W}_l\mathbf{J}^{l-1}(\mathbf{x}_i) \tag{51}$$

where $\mathbf{g}_k^l(\mathbf{x}_i) = \frac{d(f(\mathbf{x}))_k}{d\hat{\mathbf{z}}^l}|_{\mathbf{x}=\mathbf{x}_i}$ is the gradient of $(f(\mathbf{x}))_k$ with respect to $\hat{\mathbf{z}}^l$ evaluated at the input $\mathbf{x}_i$ and $\mathbf{J}^{l-1}(\mathbf{x}_i) = \frac{d\mathbf{z}^{l-1}}{d\mathbf{x}}|_{\mathbf{x}=\mathbf{x}_i}$ is the JM of $l-1$-th layer output $\mathbf{z}^{l-1}$ evaluated at the input $\mathbf{x}_i$. The gradient of $\mathbf{g}_k(\mathbf{x}_i)\mathbf{g}_k(\mathbf{x}_i)^T$ with respect to $\mathbf{W}_l$ is then given as [45]

$$\nabla_{\mathbf{W}_l}\left(\mathbf{g}_k(\mathbf{x}_i)\mathbf{g}_k(\mathbf{x}_i)^T\right) = 2\mathbf{g}_k^l(\mathbf{x}_i)^T\mathbf{g}_k^l(\mathbf{x}_i)\mathbf{W}_l\mathbf{J}^{l-1}(\mathbf{x}_i).$$

The computation of the gradient of the regularizer at layer l requires the computation of gradients $\mathbf{g}_k^l(\mathbf{x}_i)$, $k=1,\ldots,M$, $i=1,\ldots,m$ and the computation of the Jacobian matrices $\mathbf{J}^{l-1}(\mathbf{x}_i)$, $i=1,\ldots,m$. The computation of the gradient of a typical loss used for training DNN usually involves a computation of m gradients with computational complexity similar to the computational complexity of $\mathbf{g}_k^l(\mathbf{x}_i)$. Therefore, the computation of gradients required for an implementation of the Jacobian regularizer can be very expensive.

To avoid excessive computational complexity we propose a simplified version of the regularizer (49), which we name per-layer Jacobian regularizer. The per-layer Jacobian regularizer at layer l is defined as

$$R_F^l(\mathbf{W}_l) = \frac{1}{m}\sum_{i=1}^{m}\tilde{\mathbf{g}}_{\pi(i)}^{l-1}(\mathbf{x}_i)(\tilde{\mathbf{g}}_{\pi(i)}^{l-1}(\mathbf{x}_i))^T, \tag{52}$$

where $\tilde{\mathbf{g}}_{\pi(i)}^{l-1}(\mathbf{x}_i) = \frac{d(f(\mathbf{x}))_{\pi(i)}}{d\mathbf{z}^{l-1}}|_{\mathbf{x}=\mathbf{x}_i}$, and $\pi(i)\in\{1,\ldots,M\}$ is a random index. Compared to (49) we have made two simplifications. First, we assumed that input of layer l is fixed. This way we do not need to compute the JM $\mathbf{J}^{l-1}(\mathbf{x}_i)$ between the output of the layer $l-1$ and the input. Second, by choosing only one index $\pi(i)$ per training sample we have to compute only one additional gradient per training sample. This significantly reduces the computational complexity. The gradient of $\tilde{\mathbf{g}}_{\pi(i)}^{l-1}(\mathbf{x}_i)(\tilde{\mathbf{g}}_{\pi(i)}^{l-1}(\mathbf{x}_i))^T$ is simply

$$\nabla_{\mathbf{W}_l}\left(\tilde{\mathbf{g}}_{\pi(i)}^{l-1}(\mathbf{x}_i)(\tilde{\mathbf{g}}_{\pi(i)}^{l-1}(\mathbf{x}_i))^T\right) = 2\mathbf{g}_{\pi(i)}^{l}(\mathbf{x}_i)^T\mathbf{g}_{\pi(i)}^{l}(\mathbf{x}_i)\mathbf{W}_l.$$

We demonstrate the effectiveness of this regularizers in Section VI.

V. Discussion

In the preceding sections we analysed the standard feed-forward DNNs and their classification margin measured in the Euclidean norm. We now briefly discuss how our results extend to other DNN architectures and different margin metrics.

A. Beyond Feed Forward DNN

There are various DNN architectures such as Residual Networks (ResNets) [4], [46], Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [47], and Auto-encoders [48] that are used frequently in practice. It turns out that our analysis—which is based on the network's JM—can also be easily extended to such DNN architectures. In fact, the proposed framework encompasses all DNN architectures for which the JM can be computed. Below we compute the JM of a ResNet.

The ResNets introduce shortcut connection between layers. In particular, let $\phi(\cdot,\theta_l)$ denote a concatenation of several non-linear layers (see (18)). The l-th block of a Residual Network is then given as

$$\mathbf{z}^l = \mathbf{z}^{l-1} + \phi(\mathbf{z}^{l-1},\theta_l). \tag{53}$$

We denote by $\mathbf{J}_l(\mathbf{z}^{l-1})$ the JM of $\phi(\mathbf{z}^{l-1},\theta_l)$. Then the JM of the l-th block is

$$\frac{d\mathbf{z}^l}{d\mathbf{z}^{l-1}} = \mathbf{I} + \mathbf{J}_l(\mathbf{z}^{l-1}), \tag{54}$$

and the JM of a ResNet is of the form

$$\mathbf{J}_{SM}(\mathbf{z}^{L-1})\cdot\left(\mathbf{I}+\sum_{l=1}^{L}\mathbf{J}_l(\mathbf{z}^{l-1})\left(\prod_{i=1}^{l-1}(\mathbf{I}+\mathbf{J}_{l-i}(\mathbf{z}^{l-2}))\right)\right), \tag{55}$$

where $\mathbf{J}_{SM}(\mathbf{z}^{L-1})$ denotes the JM of the soft-max layer. In particular, the right element of the product in (55) can be expanded as

$$\begin{aligned}&\mathbf{I}+\mathbf{J}_1(\mathbf{x})\\&+\mathbf{J}_2(\mathbf{z}^1)+\mathbf{J}_2(\mathbf{z}^1)\mathbf{J}_1(\mathbf{x})\\&+\mathbf{J}_3(\mathbf{z}^2)+\mathbf{J}_3(\mathbf{z}^2)\mathbf{J}_2(\mathbf{z}^1)\mathbf{J}_1(\mathbf{x})+\mathbf{J}_3(\mathbf{z}^2)\mathbf{J}_2(\mathbf{x})\\&+\mathbf{J}_3(\mathbf{x})\mathbf{J}_1(\mathbf{x})\\&+\ldots\end{aligned}$$

This is a sum of JMs of all the possible sub-networks of a ResNet. In particular, there are L elements of the sum consisting only of one 1-layer sub-networks and there is only one element of the sum consisting of L-layer sub-network. This observation is consistent with the claims in [49], which states that ResNets resemble an ensemble of relatively shallow networks.

B. Beyond the Euclidean Metric

Moreover, we can also consider the geodesic distance on a manifold as a measure for margin instead of the Euclidean distance. The geodesic distance can be more appropriate than the Euclidean distance since it is a natural metric on the manifold. Moreover, the covering number of the manifold $\mathcal{X}$ may be smaller if we use the covering based on the geodesic metric balls, which will lead to tighter GE bounds. We outline the approach below.

Assume that $\mathcal{X}$ is a Riemannian manifold and $\mathbf{x},\mathbf{x}'\in\mathcal{X}$. Take a continuous, piecewise continuously differentiable curve

$c(t), t = [0,1]$ such that $c(0) = \mathbf{x}, c(1) = \mathbf{x}'$ and $c(t) \in \mathcal{X}\ \forall t \in [0,1]$. The set of all such curves $c(\cdot)$ is denoted by $\mathcal{C}$. Then the geodesic distance between $\mathbf{x}$ and $\mathbf{x}'$ is defined as

$$d_G(\mathbf{x}, \mathbf{x}') = \inf_{c(t)\in\mathcal{C}} \int_0^1 \left\| \frac{dc(t)}{dt} \right\|_2 dt\,. \tag{56}$$

Similarly as in Section III, we can show that the JM of DNN is central to bounding the distance expansion between the signals at the DNN input and the signals at the DNN output.

Theorem 6: Take $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, where $\mathcal{X}$ is the Riemmanian manifold and take $c(t)$, $t = [0,1]$ to be a continuous, piecewise continuously differentiable curve connecting $\mathbf{x}$ and $\mathbf{x}'$ such that $d_G(\mathbf{x}, \mathbf{x}') = \int_0^1 \left\| \frac{dc(t)}{dt} \right\|_2 dt$. Then

$$\|f(\mathbf{x}') - f(\mathbf{x})\|_2 \leq \sup_{t\in[0,1]} \|\mathbf{J}(c(t))\|_2 d_G(\mathbf{x}', \mathbf{x}) \tag{57}$$

Proof: The proof appears in Appendix F. ■

Note that we have established a relationship between the Euclidean distance of two points in the output space and the corresponding geodesic distance in the input space. This is important because it implies that promoting a large Euclidean distance between points can lead to a large geodesic distance between the points in the input space. Moreover, the ratio between $\|f(\mathbf{x}') - f(\mathbf{x})\|_2$ and $d_G(\mathbf{x}, \mathbf{x}')$ is upper bounded by the maximum value of the spectral norm of the network's JM evaluated on the line $c(t)$. This result is analogous to the results of Theorem 3 and Corollary 2. It also implies that regularizing the network's JM as proposed in Section IV is beneficial also in the case when the classification margin is not measured in the Euclidean metric.

Finally, note that in practice the training data may not be balanced. The provided GE bounds are still valid in such cases. However, the classification error may not the best measure of performance in such cases as it is dominated by the classification error of the class with the highest prior probability. Therefore, alternative performance measures need to be considered. We leave a detailed study of training DNN with unbalanced training sets for possible future work.

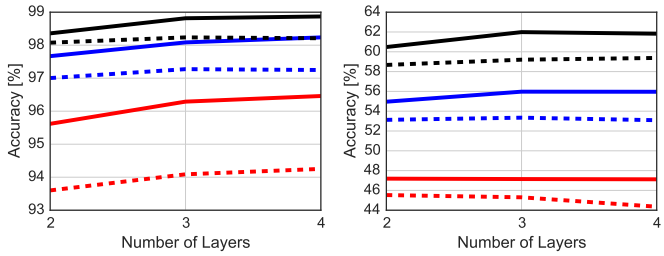
VI. Experiments

We now validate the theory with a series of experiments on the MNIST [50], CIFAR-10 [51], LaRED [52] and ImageNet (ILSVRC2012) [53] datasets. The Jacobian regularizer is applied to various DNN architectures such as DNN with fully connected layers, convolutional DNN and ResNet [4]. We use the ReLUs in all considered DNNs as this is currently the most popular non-linearity.

A. Fully Connected DNN

In this section we compare the performance of fully connected DNNs regularized with Jacobian Regularization or with the weight decay. Then we analyse the behaviour of the JM of a fully connected DNNs of various depth and width.

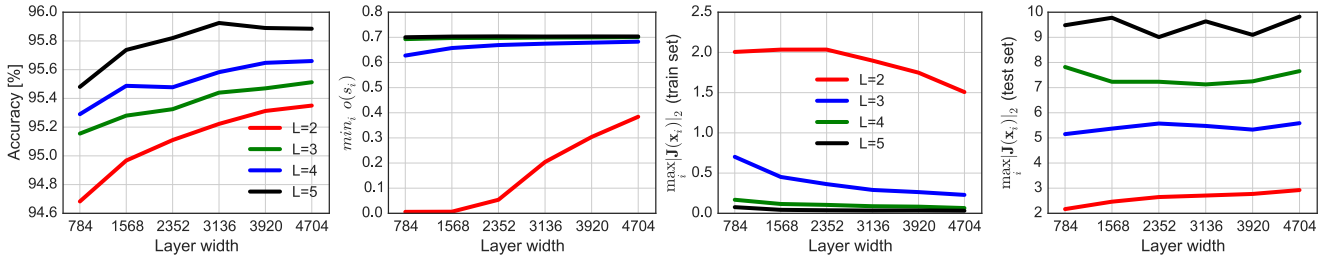
1) Comparison of Jacobian Regularization and Weight Decay: First, we compare standard DNN with fully connected layers trained with the weight decay and the Jacobian regularization (49) on the MNIST and CIFAR-10 datasets. Different number of training samples are used (5000, 20000, 50000). We consider DNNs with 2, 3 and 4 fully connected layers where all layers, except the last one, have dimension equal to the input signal dimension, which is 784 in case of MNIST and 3072 in case of CIFAR-10. The last layer is always the softmax layer and the objective is the CEE loss. The networks were trained using the stochastic gradient descent (SGD) with momentum, which was set to 0.9. Batch size was set to 128 and learning rate was set to 0.01 and reduced by factor 10 after every 40 epochs. The networks were trained for 120 epochs in total. The weight decay and the Jacobian regularization factors were chosen on a separate validation set. The experiments were repeated with the same regularization parameters on 5 random draws of training sets and weight matrix initializations. Classification accuracies averaged over different experimental runs are shown in Fig. 3. We observe that the proposed Jacobian regularization always outperforms the weight decay. This validates our theoretical results in Section IV, which predict that the Jacobian matrix is crucial for the control of (the bound to) the GE. Interestingly, in the case of MNIST, a 4 layer DNN trained with 20000 training samples and Jacobian regularization (solid blue line if Fig. 3(a)) performs on par with DNN trained with 50000 training samples and weight decay (dashed black line Fig. 3(a)), which means that the Jacobian regularization can lead to the same performance with significantly less training samples.



(a) MNIST (b) CIFAR-10

Fig. 3. Classification accuracy of DNNs trained with the jacobian regularization (solid lines) and the weight decay (dashed lines). Different numbers of training samples are used: 5000 (red), 20000 (blue) and 50000 (black).

2) Analysis of Weight Normalized Deep Neural Networks: Next, we explore weight normalized DNNs, which are described in Section IV. We use the MNIST dataset and train DNNs with a different number of fully connected layers ($L = 2, 3, 4, 5$) and different sizes of weight matrices ($M_l = 784, 2 \cdot 784, 3 \cdot 784, 4 \cdot 784, 5 \cdot 784, 6 \cdot 784,\ l = 1, \ldots, L-1$). The last layer is always the softmax layer and the objective is the CCE loss. The networks were trained using the stochastic gradient descent (SGD) with momentum, which was set to 0.9. Batch size was set to 128 and learning rate was set to 0.1 and reduced by factor 10 after every 40 epochs. The networks were trained for 120 epochs in total. All experiments are repeated 5 times with different random draws of a training set and different random weight initializations. We did not employ any additional regularization as our goal here is to explore the effects of the weight



(a) Classification accuracy. (b) Smallest score $o(s_i)$ on the training set. (c) Largest $\|\mathbf{J}(\mathbf{x}_i)\|_2$ on the training set. (d) Largest $\|\mathbf{J}(\mathbf{x}_i)\|_2$ on the test set.

Fig. 4. Weight normalized DNN with $L = 2, 3, 4, 5$ layers and different sizes of weight matrices (layer width). Plot (a) shows classification accuracy, plot (b) shows the smallest score of training samples, plot (c) shows the largest spectral norm of the network's JM evaluated on the training set and plot (d) shows the largest spectral norm of the network's JM evaluated on the testing set.

normalization on the DNN behaviour. We always use 5000 training samples.

The classification accuracies are shown in Fig. 4(a) and the smallest classification score obtained on the training set is shown in Fig. 4(b). We have observed for all configurations that the training accuracies were 100% (only exception is the case $L = 2, M_l = 784$ where the training accuracy was 99.6%). Therefore, the (testing set) classification accuracies increasing with the network depth and the weight matrix size directly imply that the GE is smaller for deeper and wider DNNs. Note also that the score increases with the network depth and width. This is most obvious for the 2 and 3 layer DNNs, whereas for the 3 and 4 layer DNNs the score is close to $\sqrt{2}$ for all network widths.

Since the DNNs are weight normalized, the Frobenious norms of the weight matrices are equal to the square root of the weight matrix dimension, and the product of Frobenious norms of the weight matrices grows with the network depth and the weight matrix size. The increase of score with the network depth and network width does not offset the product of Frobenious norms, and clearly, the bound in (38) based on the margin bound in (37) and the bound in (44), which leverage the Frobenious norms of the weight matrices, predict that the GE will increase with the network depth and weight matrix size in this scenario. Therefore, the experiment indicates that these bounds are too pessimistic.

We have also inspected the spectral norms of the weight matrices of the trained networks. In all cases the spectral norms were greater than one. We can argue that the bound in (38) based on the margin bound in (36) predicts that the GE will increase with network depth, as the product of the spectral norms grows with the network depth in a similar way than in previous paragraph. We note however, that the spectral norms of the weight matrices are much smaller than the Frobenious norms of the weight matrices.

Finally, we look for a possible explanation for the success of the weight normalization in the bounds in (38) based on the margin bounds in (34) and (35), which are a function on the JM. The largest value of the spectral norm of the network's JM evaluated on the training set is shown in Fig. 4(c) and the largest value of the spectral norm of the network's JM evaluated on the testing set is shown in Fig. 4(d).

We can observe an interesting phenomena. The maximum value of the JM's spectral norm on the training set decreases with the network depth and width. On the other hand, the maximum value of the JM's spectral norm on the testing set increases with network depth (and slightly with network width). From the perspective of the constraint sets in (39) and (40) we note that in the case of the latter we have to take into account the worst case spectral norm of the JM for inputs in conv($\mathcal{X}$). The maximum value of the spectral norm on the testing set indicates that this value increases with the network depth and implies that the bound based on (35) is still loose. On the other hand, the bound in (34) implies that we have to consider the JM in the neighbourhood of the training samples. As an approximation, we can take the spectral norms of the JMs evaluated at the training set. As it is shown in Fig. 4(c) this values decrease with the network depth and width. We argue that this provides a reasonable explanation for the good generalization of deeper and wider weight normalized DNNs.

B. *Convolutional DNN*

In this section we compare the performance of convolutional DNNs regularized with the Jacobian regularizer or with the weight decay. We also show that Jacobian Regularization can be applied to batch normalized DNNs. We will use the standard MNIST and CIFAR-10 dataset and the LaRED dataset which is briefly described below.

The LaRED dataset contains depth images of 81 distinct hand gestures performed by 10 subjects with approximately 300 images of each gesture per subject. We extracted the depth images of the hands using the masks provided in [52] and resized the images to 32×32. The images of the first 6 subjects were used to create non-overlapping training and testing sets. In addition we also constructed a testing set composed from the images of the last 4 subjects in the dataset in order to test generalization across different subjects. The goal is classification of gestures based on the depth image.

1) Comparison of Jacobian Regularization and Weight Decay: We use a 4 layer convolutional DNN with the following architecture: $(32, 5, 5)$-conv, $(2, 2)$-max-pool, $(32, 5, 5)$-conv, $(2, 2)$-max-pool followed by a softmax layer, where (k, u, v)-conv denotes the convolutional layer with k filters of size

TABLE II
CLASSIFICATION ACC. [%] OF CONVOLUTIONAL DNN ON MINST AND LARED

MNIST			LaRED (same subject)			LaRED (different subject)		
# train samples	Weight dec.	Jac. reg.	# train samples	Weight dec.	Jac. reg.	# train samples	Weight dec.	Jac. reg.
1000	94.00	**96.03**	2000	61.40	**63.56**	2000	31.53	**32.62**
5000	97.59	**98.20**	5000	76.59	**79.14**	5000	38.11	**39.62**
20000	98.60	**99.00**	10000	87.01	**88.24**	10000	41.18	**42.85**
50000	99.10	**99.35**	50000	97.18	**97.54**	50000	45.12	**46.78**

TABLE III
CLASSIFICATION ACC. [%] OF CONVOLUTIONAL DNN ON CIFAR-10

# train samples	Batch norm.	Batch norm. + Jac. reg.
2500	60.86	**66.15**
10000	76.35	**80.57**
50000	87.44	**88.95**

TABLE IV
CLASSIFICATION ACC. [%] OF RESNET CIFAR-10

# train samples	ResNet	ResNet + Jac. reg.
2500	55.69	**62.79**
10000	71.79	**78.70**
50000 + aug.	93.34	**94.32**

$u \times v$ and (p, p)-max-pool denotes max-pooling with pooling regions of size $p \times p$. The training procedure follows the one described in the previous paragraphs. The results are reported in Table II.

We observe that training with the Jacobian regularization outperforms the weight decay in all cases. This is most obvious at smaller training set sizes. For example, on the MNIST dataset, the DNN trained using 1000 training samples and regularized with the weight decay achieves classification accuracy of 94% and the DNN trained with the Jacobian regularization achieves classification accuracy of 96.3%.

Similarly, on the LaRED dataset the Jacobian regularization outperforms the weight decay with the difference most obvious at the smallest number of training samples. Note also that the generalization of the network to the subjects outside the training set is not very good; i.e., using 50000 training samples the classification accuracy on the testing set containing the same subjects is higher than 97% whereas the classification accuracy on the testing set containing different subjects is only 46%. Nevertheless, the Jacobian regularization outperforms the weight decay also on this testing set by a small margin.

2) Batch Normalization and Jacobian Regularization: Now we show that the Jacobian regularization (49) can also be applied to a batch normalized DNN. Note that we have shown in Section IV that the batch normalization has an effect of normalizing the rows of the weight matrices.

We us the CIFAR-10 dataset and use the All-convolutional-DNN proposed in [54] (All-CNN-C) with 9 convolutional layers, an average pooling layer and a softmax layer. All the convolutional layers are batch normalized and the softmax layer is weight normalized. The networks were trained using the stochastic gradient descent (SGD) with momentum, which was set to 0.9. Batch size was set to 64 and the learning rate was set to 0.1 and reduced by a factor 10 after every 25 epochs. The networks were trained for 75 epochs in total. The classification accuracy results are presented in Table III for different sizes of training sets (2500, 10000, 50000).

We can observe that the Jacobian regularization also leads to a smaller GE in this case.

C. Residual Networks

Now we demonstrate that the Jacobian regularizer is also effective when applied to ResNets. We use the CIFAR-10 and ImageNet datasets. We use the per-layer Jacobian regularization (52) for experiments in this section.

1) CIFAR-10: The Wide ResNet architecture proposed in [35], which follows [46], but proposes wider and shallower networks which leads to the same or better performance than deeper and thinner networks is used here. In particular, we use the ResNet with 22 layers of width 5.

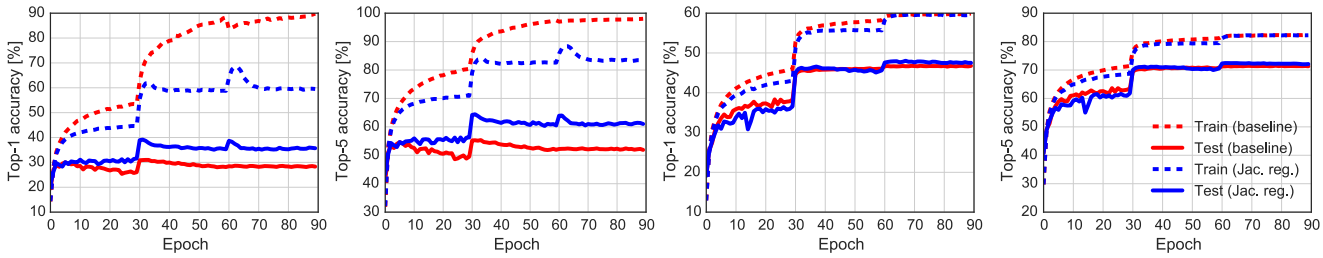
We follow the data normalization process of [35]. We also follow the training procedure of [35] except for the learning rate and use the learning rate sequence: (0.01, 5), (0.05, 20), (0.005, 40), (0.0005, 40), (0.00005, 20), where the first number in parenthesis corresponds to the learning rate and the second number corresponds to the number of epochs. We train ResNet on small training sets (2500 and 10000 training samples) without augmentation and on the full training set with the data augmentation as in [35]. The regularization factor were set to 1 and 0.1 for the smaller training sets (2500 and 10000) and the full augmented training set, respectively.

The results are presented in Table IV. In all cases the ResNet with Jacobian regularization outperforms the standard ResNet. The effect of regularization is the strongest with the smaller number of training samples, as expected.

2) ImageNet: We use the 18 layer ResNet [4] with identity connection [46]. The training procedure follows [4] with the learning rate sequence: (0.1, 30), (0.01, 30), (0.001, 30). The Jacobian regularization factor is set to 1.

The images in the dataset are resized to 128×128. We run an experiment without data augmentation and with data augmentation following [1], which includes random cropping of images of size 112×112 from the original image and color augmentation. The classification accuracies during training are shown in Fig. 5 and the final results are reported in Table V.

We first focus on training without data augmentation. The ResNet trained using the Jacobian regularization has a much smaller GE (23.83%) compared to the baseline ResNet



(a) No data augmentation. (b) No data augmentation. (c) Data augmentation. (d) Data augmentation.

Fig. 5. Training set (dashed) and testing set (solid) classification accuracies during training. Blue curves correspond to the ResNet with Jacobian regularization and red curves correspond to the baseline ResNet. Top-1 and top-5 classification accuracies are reported for training without data augmentation (a), (b) and for training with data augmentation (c), (d).

TABLE V
CLASSIFICATION ACC. [%] AND GE [%] OF RESNET ON IMAGENET

Setup	Train	Test	GE
Baseline	89.82	28.29	61.53
Baseline + Jac. reg.	59.52	35.69	23.83
Baseline + aug.	59.89	46.75	13.14
Baseline + aug + Jac. reg.	59.54	**47.51**	**12.03**

TABLE VI
AVERAGE COMPUTATION TIME [*s*/BATCH]

Experiment	no reg.	Jac. reg.	Increase factor
MNIST (Sec.VI-B1)	0.003	0.030	10.00
ImageNet (Sec. VI-C2)	0.120	0.190	1.580

(61.53%). This again demonstrates that the Jacobian regularization decreases the GE, as our theory predicts. Note that the smaller GE of Jacobian regularized ResNet partially transfers to a higher classification accuracy on the testing set. However, in practice DNNs are often trained with data augmentation. In this case the GE of a baseline ResNet is much lower (13.14%) and is very close to the GE of the ResNet with the Jacobian regularization (12.03%). It is clear that data augmentation reduces the need for strong regularization. Nevertheless, note that the ResNet trained with the Jacobian regularization achieves a slightly higher testing set accuracy (47.51%) compared to the baseline ResNet (46.75%).

D. Computational Time

Finally, we measure how the use of Jacobian regularization affects training time of DNNs. We have implemented DNNs in Theano [55], which includes automatic differentiation and computation graph optimization. The experiments are run on the Titan X GPU. The average computational time per batch for the convolutional DNN on the MNIST dataset in Section VI-B1 and for the ResNet on the ImageNet dataset in Section VI-C2 are reported in Table VI. Note that in the case of MNIST the regularizer in (49) is used and in the case of ImageNet the per-layer regularizer in (52) is used. These results are also representative of the other datasets and network architectures.

We can observe that using the Jacobian regularizer in (49) introduces additional computational time. This may not be critical if the number of training samples is small and training computational time is not too critical. On the other hand, the per-layer Jacobian regularizer in (52) has a much smaller cost. As shown in the experiments this regularizer is still effective and leads to only 58% increase in computation time on the ImageNet dataset. Due to its efficiency the per-layer Jacobian regularizer might be more appropriate for large scale experiments where computational time is important.

VII. CONCLUSION

This paper studies the GE of DNNs based on their classification margin. In particular, our bounds express the generalization error as a function of the classification margin, which is bounded in terms of the achieved separation between the training samples at the network output and the network's JM.

One of the hallmarks of our bounds relates to the fact that the characterization of the behaviour of the generalization error is tighter than that associated with other bounds in the literature. Our bounds predict that the generalization error of deep neural networks can be independent of their depth and size whereas other bounds say that the generalization error is exponential in the network width or size.

Our bounds also suggest new regularization strategies such as the regularization of the network's Jacobian matrix, which can be applied on top of other modern DNN training strategies such as the weight normalization and the batch normalization, where the standard weight decay can not be applied. These regularization strategies are especially effective in the limited training data regime in comparison to other approaches, with moderate increase in computational complexity.

APPENDIX

A. Proof of Theorem 3

We first note that the line between $\mathbf{x}$ and $\mathbf{x}'$ is given by $\mathbf{x}+t(\mathbf{x}'-\mathbf{x}),\quad t\in[0,1]$. We define the function $F(t)=f(\mathbf{x}+t(\mathbf{x}'-\mathbf{x}))$, and observe that $\frac{dF(t)}{dt}=\mathbf{J}(\mathbf{x}+t(\mathbf{x}'-\mathbf{x}))(\mathbf{x}'-\mathbf{x})$. By the generalized fundamental theorem

of calculus or the Lebesgue differentiation theorem we write

$$f(\mathbf{x}') - f(\mathbf{x}) = F(1) - F(0) = \int_0^1 \frac{dF(t)}{dt}\, dt = \int_0^1 \mathbf{J}(\mathbf{x} + t(\mathbf{x}' - \mathbf{x}))\, dt\, (\mathbf{x}' - \mathbf{x}). \quad (58)$$

This concludes the proof.

B. Proof of Corollary 2

First note that $\|\mathbf{J}_{\mathbf{x},\mathbf{x}'}(\mathbf{x}' - \mathbf{x})\|_2 \leq \|\mathbf{J}_{\mathbf{x},\mathbf{x}'}\|_2 \|\mathbf{x}' - \mathbf{x}\|_2$ and that $\mathbf{J}_{\mathbf{x},\mathbf{x}'}$ is an integral of $\mathbf{J}(\mathbf{x} + t(\mathbf{x}' - \mathbf{x}))$. In addition, notice that we may always apply the following upper bound:

$$\|\mathbf{J}_{\mathbf{x},\mathbf{x}'}\|_2 \leq \sup_{\mathbf{x},\mathbf{x}' \in \mathcal{X}, t \in [0,1]} \|\mathbf{J}(\mathbf{x} + t(\mathbf{x}' - \mathbf{x}))\|_2 . \quad (59)$$

Since $\mathbf{x} + t(\mathbf{x}' - \mathbf{x}) \in \text{conv}(\mathcal{X})\ \forall t \in [0,1]$, we get (24).

C. Proof of Lemma 1

In all proofs we leverage the fact that for any two matrices $\mathbf{A}, \mathbf{B}$ of appropriate dimensions it holds $\|\mathbf{AB}\|_2 \leq \|\mathbf{A}\|_2 \|\mathbf{B}\|_2$. We also leverage the bound $\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$.

We start with the proof of statement 1). For the non-linear layer (18), we note that the JM is a product of a diagonal matrix (27) and the weight matrix $\mathbf{W}_l$. Note that for all the considered non-linearities the diagonal elements of (27) are bounded by 1 (see derivatives in Table I), which implies that the spectral norm of this matrix is bounded by 1. Therefore the spectral norm of the JM is upper bounded by $\|\mathbf{W}_l\|_2$. The proof for the linear layer is trivial. In the case of the softmax layer (17) we have to show that the spectral norm of the softmax function $\left(-\zeta(\hat{\mathbf{z}})\zeta(\hat{\mathbf{z}})^T + \text{diag}(\zeta(\hat{\mathbf{z}}))\right)$ is bounded by 1. We use the Gershgorin disc theorem, which states that the eigenvalues of $\left(-\zeta(\hat{\mathbf{z}})\zeta(\hat{\mathbf{z}})^T + \text{diag}(\zeta(\hat{\mathbf{z}}))\right)$ are bounded by

$$\max_i (\zeta(\hat{\mathbf{z}}))_i (1 - (\zeta(\hat{\mathbf{z}}))_i) + (\zeta(\hat{\mathbf{z}}))_i \sum_{j \neq i} (\zeta(\hat{\mathbf{z}}))_j . \quad (60)$$

Noticing that $\sum_{j \neq i} (\zeta(\hat{\mathbf{z}}))_j \leq 1$ leads to the upper bound

$$\max_i (\zeta(\hat{\mathbf{z}}))_i (2 - (\zeta(\hat{\mathbf{z}}))_i) . \quad (61)$$

Since $(\zeta(\hat{\mathbf{z}}))_i \in [0,1]$ it is trivial to show that (61) is upper bounded by 1.

The proof of statement 2) is straightforward. Because the pooling regions are non-overlapping it is straightforward to verify that the rows of all the defined pooling operators $\mathbf{P}^l(\mathbf{z}^{l-1})$ are orthonormal. Therefore, the spectral norm of the JM is equal to 1.

D. Proof of Theorem 4

Throughout the proof we will use the notation $o(s_i) = o(\mathbf{x}_i, y_i)$ and $\mathbf{v}_{ij} = \sqrt{2}(\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)$. We start by proving the inequality in (34). Assume that the classification margin $\gamma^d(s_i)$ of training sample $(\mathbf{x}_i, y_i)$ is given and take $j^\star = \arg\min_{j \neq y_i} \min \mathbf{v}_{y_i j}^T f(\mathbf{x}_i)$. We then take a point $\mathbf{x}^\star$ that lies on the decision boundary between y_i and $j^\star$ such that $o(\mathbf{x}^\star, y_i) = 0$. Then

$$o(\mathbf{x}_i, y_i) = o(\mathbf{x}, y_i) - o(\mathbf{x}^\star, y_i) = \mathbf{v}_{y_i j^\star}^T (f(\mathbf{x}_i) - f(\mathbf{x}^\star)) = \mathbf{v}_{y_i j^\star}^T \mathbf{J}_{\mathbf{x}_i, \mathbf{x}^\star} (\mathbf{x}_i - \mathbf{x}^\star) \leq \|\mathbf{J}_{\mathbf{x}_i, \mathbf{x}^\star}\|_2 \|\mathbf{x}_i - \mathbf{x}^\star\|_2 .$$

Note that by the choice of $\mathbf{x}^\star$, $\|\mathbf{x}_i - \mathbf{x}^\star\|_2 = \gamma^d(s_i)$ and similarly $\|\mathbf{J}_{\mathbf{x}_i, \mathbf{x}^\star}\|_2 \leq \sup_{\mathbf{x}: \|\mathbf{x} - \mathbf{x}_i\|_2 \leq \gamma^d(s_i)} \|\mathbf{J}(\mathbf{x})\|_2$. Therefore, we can write

$$o(s_i) \leq \sup_{\mathbf{x}: \|\mathbf{x} - \mathbf{x}_i\|_2 \leq \gamma^d(s_i)} \|\mathbf{J}(\mathbf{x})\|_2\ \gamma^d(s_i), \quad (62)$$

which leads to (34).

Next, we prove (35). Recall the definition of the classification margin in (10):

$$\gamma^d(s_i) = \sup\{a : \|\mathbf{x}_i - \mathbf{x}\|_2 \leq a \implies g(\mathbf{x}) = y_i\ \forall \mathbf{x}\} = \sup\{a : \|\mathbf{x}_i - \mathbf{x}\|_2 \leq a \implies o(\mathbf{x}, y_i) > 0\, \forall \mathbf{x}\},$$

where we leverage the definition in (33). We observe $o(\mathbf{x}, y_i) > 0 \iff \min_{j \neq y_i} \mathbf{v}_{y_i j}^T f(\mathbf{x}) > 0$ and

$$\min_{j \neq y_i} \mathbf{v}_{y_i j}^T f(\mathbf{x}) = \min_{j \neq y_i} \left(\mathbf{v}_{y_i j}^T f(\mathbf{x}_i) + \mathbf{v}_{y_i j}^T (f(\mathbf{x}) - f(\mathbf{x}_i))\right) .$$

Note that

$$\min_{j \neq y_i} \left(\mathbf{v}_{y_i j}^T f(\mathbf{x}_i) + \mathbf{v}_{y_i j}^T (f(\mathbf{x}) - f(\mathbf{x}_i))\right) \quad (63)$$

$$\geq \min_{j \neq y_i} \mathbf{v}_{y_i j}^T f(\mathbf{x}_i) + \min_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}) - f(\mathbf{x}_i))$$

$$= o(\mathbf{x}_i, y_i) + \min_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}) - f(\mathbf{x}_i)) . \quad (64)$$

Therefore,

$$o(\mathbf{x}_i, y_i) + \min_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}) - f(\mathbf{x}_i)) > 0 \implies o(\mathbf{x}, y_i) > 0 .$$

This leads to the bound of the classification margin

$$\gamma^d(s_i) \geq \sup\{a : \|\mathbf{x}_i - \mathbf{x}\|_2 \leq a \implies o(\mathbf{x}_i, y_i) + \min_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}) - f(\mathbf{x}_i)) > 0\, \forall \mathbf{x}\} .$$

Note now that

$$o(\mathbf{x}_i, y_i) + \min_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}) - f(\mathbf{x}_i)) > 0 \quad (65)$$

$$\iff$$

$$o(\mathbf{x}_i, y_i) - \max_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}_i) - f(\mathbf{x})) > 0 \quad (66)$$

$$\iff$$

$$o(\mathbf{x}_i, y_i) > \max_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}_i) - f(\mathbf{x})). \quad (67)$$

Moreover,

$$\max_{j \neq y_i} \mathbf{v}_{y_i j}^T (f(\mathbf{x}_i) - f(\mathbf{x})) \leq \sup_{\mathbf{x} \in \text{conv}(\mathcal{X})} \|\mathbf{J}(\mathbf{x})\|_2 \|\mathbf{x}_i - \mathbf{x}\|_2 ,$$

where we have leveraged the fact that $\|\mathbf{v}_{ij}\|_2 = 1$ and the inequality (24) in Corollary 2. We may write

$$\gamma^d(s_i) \geq \sup\{a : \|\mathbf{x}_i - \mathbf{x}\|_2 \leq a \implies o(\mathbf{x}_i, y_i) > \sup_{\mathbf{x} \in \text{conv}(\mathcal{X})} \|\mathbf{J}(\mathbf{x})\|_2 \|\mathbf{x}_i - \mathbf{x}\|_2\ \forall \mathbf{x}\}.$$

a that attains the supremum can be obtain easily and we get:

$$\gamma^d(s_i) \geq \frac{o(\mathbf{x}_i, y_i)}{\sup_{\mathbf{x} \in \text{conv}(\mathcal{X})} \|\mathbf{J}(\mathbf{x})\|_2}\,, \tag{68}$$

which proves (35). The bounds in (36) and (37) follow from the bounds provided in Lemma 1 and the fact that the spectral norm of a matrix product is upper bounded by the product of the spectral norms. This concludes the proof.

E. Proof of Theorem 5

We denote by $\mathbf{W}_l^N$ the row normalized matrix obtained from $\mathbf{W}_l$ (in the same way as (45)). By noting that the ReLU and diagonal non-negative matrices commute, it is straight forward to verify that

$$[\mathbf{N}\left(\{\mathbf{z}_i^l\}_{i=1}^m, \mathbf{W}_l\right)\, \mathbf{W}_l \mathbf{z}^l]_\sigma = \mathbf{N}\left(\{\mathbf{z}_i^l\}_{i=1}^m, \mathbf{W}_l^N\right)[\mathbf{W}_l^N \mathbf{z}^l]_\sigma\,.$$

Note now that we can consider $\mathbf{N}(\{\mathbf{z}_i^l\}_{i=1}^m, \mathbf{W}_l^N)$ as the part of the weight matrix $\mathbf{W}_{l+1}$. Therefore, we can conclude that layer l has row normalized weight matrix. When the batch normalization is applied to layers, all the weight matrices will be row normalized. The exception is the weight matrix of the last layer, which will be of the form $\mathbf{N}(\{\mathbf{z}_i^{L-1}\}_{i=1}^m, \mathbf{W}_L)\mathbf{W}_L$.

F. Proof of Theorem 6

We begin by noting that $f(\mathbf{x}') - f(\mathbf{x}) = f(c(1)) - f(c(0))$ and

$$f(c(1)) - f(c(0)) = \int_0^1 \frac{df(c(t))}{dt}\, dt = \int_0^1 \frac{df(c(t))}{dc(t)} \frac{dc(t)}{dt}\, dt\,,$$

where the first equality follows from the generalized fundamental theorem of calculus, following the idea presented in the proof of Theorem 3. The second equality follows from the chain rule of differentiation. Finally, we note that $\frac{df(c(t))}{c(t)} = \mathbf{J}(c(t))$ and that the norm of the integral is always smaller or equal to the integral of the norm and obtain

$$\begin{aligned}
\|f(\mathbf{x}') - f(\mathbf{x})\|_2 &= \left\| \int_0^1 \mathbf{J}(c(t)) \frac{dc(t)}{dt}\, dt \right\|_2 \\
&\leq \int_0^1 \|\mathbf{J}(c(t))\|_2 \left\| \frac{dc(t)}{dt} \right\|_2 dt \\
&\leq \sup_{t \in [0,1]} \|\mathbf{J}(c(t))\|_2 \int_0^1 \left\| \frac{dc(t)}{dt} \right\|_2 dt \\
&= \sup_{t \in [0,1]} \|\mathbf{J}(c(t))\|_2 d_G(\mathbf{x}, \mathbf{x}')\,,
\end{aligned} \tag{69}$$

where we have noted that $\int_0^1 \|\frac{dc(t)}{dt}\|_2 = d_G(\mathbf{x}, \mathbf{x}')$.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] G. Hinton *et al.*,"Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Oct. 2012.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2016, pp. 770–778.

[5] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[6] J. Bruna, A. Szlam, and Y. LeCun, "Learning stable group invariant representations with convolutional networks," *Proc. Int. Conf. Learn. Representations*, 2013.

[7] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 111–118.

[8] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[9] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[10] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2924–2932.

[11] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning: A tensor analysis," in *Proc. 29th Annu. Conf. Learn. Theory*, 2016, pp. 698–728.

[12] M. Telgarsky, "Benefits of depth in neural networks," in *Proc. 29th Annu. Conf. Learn. Theory*, 2016, pp. 1517–1539.

[13] S. Mallat, "Group invariant scattering," *Commun. Pure Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, 2012.

[14] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Mar. 2012.

[15] T. Wiatowski and H. Bölcskei, "A mathematical theory of deep convolutional neural networks for feature extraction," arXiv:1512.06293, 2015.

[16] R. Giryes, G. Sapiro, and A. M. Bronstein, "Deep neural networks with random Gaussian weights: A universal classification strategy," *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3444–3457, Jul. 2016.

[17] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2015, pp. 192–204.

[18] B. D. Haeffele and R. Vidal, "Global optimality in tensor factorization, deep learning, and beyond," arXiv:1506.07540, 2015.

[19] R. Giryes, Y. C. Eldar, A. M. Bronstein, and G. Sapiro, "Tradeoffs between convergence speed and reconstruction accuracy in inverse problems," arXiv:1605.09232, 2016.

[20] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *Proc. Int. Conf. Learn. Representations*, 2014.

[21] Y. Ollivier, "Riemannian metrics for neural networks I: Feedforward networks," *Inf. Inference*, vol. 4, no. 2, pp. 108–153, Jun. 2015.

[22] B. Neyshabur and R. Salakhutdinov, "Path-SGD: Path-normalized optimization in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2422–2430.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[24] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 901–909.

[25] S. An, M. Hayat, S. H. Khan, M. Bennamoun, F. Boussaid, and F. Sohel, "Contractive rectifier networks for nonlinear maximum margin classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2515–2523.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jun. 2014.

[27] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 833–840.

[28] J. Huang, Q. Qiu, G. Sapiro, and R. Calderbank, "Discriminative robust transformation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1333–1341.

[29] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.

[30] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[31] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *J. Mach. Learn. Res.*, vol. 3, pp. 463–482, 2002.
[32] H. Xu and S. Mannor, "Robustness and generalization," *Mach. Learn.*, vol. 86, no. 3, pp. 391–423, 2012.
[33] B. Neyshabur, R. Tomioka, and N. Srebro, "Norm-based capacity control in neural networks," *Proc. 28th Conf. Learn. Theory*, 2015, pp. 1376–1401.
[34] S. Sun, W. Chen, L. Wang, and T.-Y. Liu, "Large margin deep neural networks: Theory and algorithms," arXiv:1506.05232, 2015.
[35] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. British Mach. Vision Conf.*, 2016.
[36] C. Zhang, S. Bengio, M. Hardt, and B. Recht, "Understanding deep learning requires rethinking generalization," in *Proc. Int. Conf. Learn. Representations*, 2017.
[37] J. Sokolić, R. Giryes, G. Sapiro, and M. R. D. Rodrigues, "Generalization error of invariant classifiers," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1094–1103.
[38] K. Q. Shen, C. J. Ong, X. P. Li, and E. P. V. Wilder-Smith, "Feature selection via sensitivity analysis of SVM probabilistic outputs," *Mach. Learn.*, vol. 70, no. 1, pp. 1–20, Jan. 2008.
[39] J. B. Yang, K. Q. Shen, C. J. Ong, and X. P. Li, "Feature selection via sensitivity analysis of MLP probabilistic outputs," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2008, pp. 774–779.
[40] D. Shi, D. S. Yeung, and J. Gao, "Sensitivity analysis applied to the construction of radial basis function networks," *Neural Netw.*, vol. 18, no. 7, pp. 951–957, Mar. 2005.
[41] S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann, "Uniform uncertainty principle for Bernoulli and sub-Gaussian ensembles," *Constructive Approx.*, vol. 28, no. 3, pp. 277–289, Dec. 2008.
[42] N. Verma, "Distance preserving embeddings for general n-dimensional manifolds." *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 2415–2448, Aug. 2013.
[43] B. Neyshabur, R. Tomioka, R. Salakhutdinov, and N. Srebro,"Data-dependent path normalization in neural networks," in *Proc. Int. Conf. Learn. Representations*, 2015.
[44] G. Watson, "Characterization of the subdifferential of some matrix norms," *Linear Algebra Appl.*, vol. 170, pp. 33–45, Jun. 1992.
[45] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," *Tech. Univ.* Denmark, Lyngby, Denmark, 2012.
[46] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Europ. Conf. Comput. Vision*, 2016, pp. 630–645.
[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[48] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[49] A. Veit, M. Wilber, and S. Belongie, "Residual networks are exponential ensembles of relatively shallow networks," in *Proc. Adv. Neural Inform. Process. Syst.*, 2016, pp. 550–558.
[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[51] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Comput. Sci. Dep., Uni. Toronto, *Tech. Rep.*, Apr. 2009.
[52] Y. S. Hsiao, J. Sanchez-Riera, T. Lim, K. L. Hua, and W. H. Cheng, "LaRED: A large RGB-D extensible hand gesture dataset," *Proc. 5th ACM Multimedia Syst. Conf.*, 2014, pp. 53–58.
[53] O. Russakovsky *et al.*"ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[54] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. Int. Conf. Learn. Representations*, Dec. 2015.
[55] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," arXiv:1605.02688, 2016.

Jure Sokolić (S'14) received the Diploma degree in electrical engineering from University of Ljubljana, Ljubljana, Slovenia, in 2013. He is currently working toward the Ph.D. degree in the Department of Electrical & Electronic Engineering, University College London, London, U.K. He is visiting Duke University in 2016–2017 as a Vest Scholar.

His research interest focuses on high-dimensional data processing, machine learning, and deep learning.

Raja Giryes (M'13) received the B.Sc. degree in 2007, the M.Sc. (supervision by Prof. M. Elad and Prof. Y. C. Eldar, 2009), and Ph.D. (supervision by Prof. M. Elad 2014) degrees from the Department of Computer Science, The Technion - Israel Institute of Technology, Haifa. He was a postdoctoral at the Computer Science Department, Technion (Nov. 2013 till July 2014) and at the lab of Prof. G. Sapiro at Duke University, Durham, USA (July 2014 and Aug. 2015). He is an Assistant Professor in the School of Electrical Engineering, Tel Aviv University. His research interests lie at the intersection between signal and image processing and machine learning, and in particular, in deep learning, inverse problems, sparse representations, and signal and image modeling.

Dr. Giryes received the Maof prize for excellent young faculty (2016–2019), VATAT scholarship for excellent postdoctoral fellows (2014–2015), Intel Research and Excellence Award (2005, 2013), the Excellence in Signal Processing Award (ESPA) from Texas Instruments (2008), and was part of the Azrieli Fellows program (2010–2013).

Guillermo Sapiro (F'14) was born in Montevideo, Uruguay, on April 3, 1966. He received the B.Sc. (*summa cum laude*), M.Sc., and Ph.D. degrees from the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, Israel, in 1989, 1991, and 1993, respectively. After postdoctoral research at MIT, he became a Member of Technical Staff at the research facilities of HP Labs in Palo Alto, CA, USA. He was with the Department of Electrical and Computer Engineering, University of Minnesota, where he held the position of Distinguished McKnight University Professor and Vincentine Hermes-Luh Chair in Electrical and Computer Engineering. He is currently the Edmund T. Pratt, Jr. School Professor with Duke University.

He works on theory and applications in computer vision, computer graphics, medical imaging, image analysis, and machine learning. He has authored and coauthored more than 400 papers in these areas and has written a book published by Cambridge University Press, January 2001.

G. Sapiro received the Gutwirth Scholarship for Special Excellence in Graduate Studies in 1991, the Ollendorff Fellowship for Excellence in Vision and Image Understanding Work in 1992, the Rothschild Fellowship for Postdoctoral Studies in 1993, the Office of Naval Research Young Investigator Award in 1998, the Presidential Early Career Awards for Scientist and Engineers (PECASE) in 1998, the National Science Foundation Career Award in 1999, and the National Security Science and Engineering Faculty Fellowship in 2010. He received the test of time award at ICCV 2011. He is a Fellow of SIAM. He was the founding Editor-in-Chief of the *SIAM Journal on Imaging Sciences*.

Miguel R. D. Rodrigues (SM'15) received the Licenciatura degree in electrical and computer engineering from the University of Porto, Porto, Portugal, and the Ph.D. degree in electronic and electrical engineering from the University College London, U.K. He is currently a Reader in information theory and processing with the Department of Electronic and Electrical Engineering, University College London, London, U.K. He was with the University of Porto rising through the ranks from Assistant to Associate Professor, where he was also the Head of the Information Theory and Communications Research Group, Instituto de Telecomunicações Porto.

He has also held postdoctoral positions and visiting appointments with various institutions worldwide including University College London, Cambridge University, Princeton University, and Duke University from 2003 to 2016. His research interests include the general areas of information theory and processing with a current focus on sensing, analysis, and processing of high-dimensional data. His work, which has led to more than 150 papers in the leading international journals and conferences in the field, has also been honored with the Prestigious IEEE Communications and Information Theory Societies Joint Paper Award 2011.

Dr. Rodrigues received fellowships from the Portuguese Foundation for Science and Technology and the Foundation Calouste Gulbenkian. He has served as a Co-Chair of the Technical Program Committee of the IEEE Information Theory Workshop 2016 and also as a Co-Organizer of the Workshop on Sensing and Analysis of High-Dimensional Data 2014 and 2015. He currently serves as an Associate Editor of the IEEE COMMUNICATIONS LETTERS.