Mathematics of Deep Learning

René Vidal Joan Bruna Raja Giryes Stefano Soatto

***Abstract*— Recently there has been a dramatic increase in the performance of recognition systems due to the introduction of deep architectures for representation learning and classification. However, the mathematical reasons for this success remain elusive. This tutorial will review recent work that aims to provide a mathematical justification for several properties of deep networks, such as global optimality, geometric stability, and invariance of the learned representations.**

I. Introduction

Deep networks [1] are parametric models that perform sequential operations on their input data. Each such operation, colloquially called a "layer", consists of a linear transformation, say, a convolution of its input, followed by a point-wise nonlinear "activation function", e.g., a sigmoid. Deep networks have recently led to dramatic improvements in classification performance in various applications in speech and natural language processing, and computer vision. The crucial property of deep networks that is believed to be the root of their performance is that they have a large number of layers as compared to classical neural networks; but there are other architectural modifications such as rectified linear activations (ReLUs) [2] and residual "shortcut" connections [3]. Other major factors in their success is the availability of massive datasets, say, millions of images in datasets like ImageNet [4], and efficient GPU computing hardware for solving the resultant high-dimensional optimization problem which may have up to 100 million parameters.

The empirical success of deep learning, especially convolutional neural networks (CNNs) for image-based tasks, presents numerous puzzles to theoreticians. In particular, there are three key factors in deep learning, namely the architectures, regularization techniques and optimization algorithms, which are critical to train well-performing deep networks and understanding their necessity and interplay is essential if we are to unravel the secrets of their success.

A. *Approximation, depth, width and invariance properties*

An important property in the design of a neural network architecture is its ability to approximate arbitrary functions of the input. But how does this ability depend on parameters of the architecture, such as its depth and width? Earlier work shows that neural networks with a single hidden layer and sigmoidal activations are universal function approximators [5], [6], [7], [8]. However, the capacity of a wide and shallow network can be replicated by a deep network with significant improvements in performance. One possible explanation is that deeper architectures are able to better capture invariant properties of the data compared to their shallow counterparts. In computer vision, for example, the category of an object is invariant to changes in viewpoint, illumination, etc. While a mathematical analysis of why deep networks are able to capture such invariances remains elusive, recent progress has shed some light on this issue for certain sub-classes of deep networks. In particular, scattering networks [9] are a class of deep networks whose convolutional filter banks are given by complex, **multi-resolution wavelet families.** As a result of this extra structure, they are provably stable and locally invariant signal representations, and reveal the fundamental role of geometry and stability that underpins the generalization performance of modern deep convolutional network architectures; see Section IV.

B. *Generalization and regularization properties*

Another critical property of a neural network architecture is its ability to generalize from a small number of training examples. Traditional results from statistical learning theory [10] show that the number of training examples needed to achieve good generalization grows polynomially with the size of the network. In practice, however, deep networks are trained with much fewer data than the number of parameters ($N \ll D$ regime) and yet they can be prevented from overfitting using very simple (and seemingly counter-productive) regularization techniques like Dropout [11], which simply freezes a random subset of the parameters at each iteration.

One possible explanation for this conundrum is that deeper architectures produce an embedding of the input data that approximately preserves the distance between data points in the same class, while increasing the separation between classes. This tutorial will overview the recent work of [12], which uses tools from **compressed sensing and dictionary learning** to prove that deep networks with random Gaussian weights perform a distance-preserving embedding of the data in which similar inputs are likely to have a similar output. These results provide insights into the metric learning properties of the network and lead to bounds on the generalization error that are informed by the structure of the input data.

C. *Information-theoretic properties*

Another key property of a network architecture is its ability to produce a good "representation of the data". Roughly speaking, a representation is any function of the input data

R. Vidal is with the Center for Imaging Science, Biomedical Engineering, Johns Hopkins University, Baltimore, USA rvidal@cis.jhu.edu
J. Bruna is with the Courant Institute of Mathematical Sciences, Center for Data Science, New York University, USA bruna@cims.nyu.edu
R. Giryes is with the School of Electrical Engineering, Tel-Aviv University, Tel Aviv, Israel raja@tauex.tau.ac.il
S. Soatto is with the Department of Computer Science, University of California, Los Angeles, USA soatto@ucla.edu

that is useful for a task. An optimal representation would be one that is "most useful" as quantified, for instance, by information-theoretic, complexity or invariance criteria [13]. This is akin to the "state" of the system and is what an agent would store in its memory in lieu of the data to predict future observations. For example, the state of a Kalman filter is an optimal representation for predicting data generated by a linear dynamical system with Gaussian noise; in other words, it is a minimal sufficient statistic for prediction. For complex tasks where the data may be corrupted by "nuisances" that do not contain information about the task, one may also wish for this representation to be "invariant" to such nuisances so as not to affect future predictions. In general, optimal representations for a task can be defined as sufficient statistics (of past or "training" data) that are also minimal, and invariant to nuisance variability affecting future ("test") data [14]. Despite a large interest in representation learning, a comprehensive theory that explains the performance of deep networks as constructing optimal representations does not yet exist. Indeed, even basic concepts such as sufficiency and invariance have received diverse treatment [9], [14], [15].

Recent work [16], [17], [18] has begun to establish information-theoretic foundations for representations learned by deep networks. These include the observation that the information bottleneck loss [13], which defines a relaxed notion of minimal sufficiency, can be used to compute *optimal* representations. The information bottleneck loss can be re-written as the sum of a cross-entropy term, which is precisely the most commonly used loss in deep learning, with an additional regularization term. The latter can be implemented by introducing noise, similar to adaptive dropout noise, in the learned representation [17]. The resulting form of regularization, called *information dropout* in [17], shows improved learning under resource constraints, and can be shown to lead to "maximally disentangled" representations, i.e., the (total) correlation among components of the representation is minimal, thereby making the features indicators of independent characteristics of data. Moreover, similar techniques show improved robustness to adversarial perturbations [18]. **Information theory** is hence expected to play a key role in formalizing and analyzing the properties of deep representations and suggesting new classes of regularizers.

D. Optimization properties

The classical approach to training neural networks is to minimize a (regularized) loss using backpropagation [19], a gradient descent method specialized to neural networks. Modern versions of backpropagation rely on stochastic gradient descent (SGD) to efficiently approximate the gradient for massive datasets. While SGD has been rigorously analyzed only for convex loss functions [20], in deep learning the loss is a non-convex function of the network parameters, hence there are no guarantees that SGD finds the global minimizer.

In practice, there is overwhelming evidence that SGD routinely yields good solutions for deep networks. Recent work on understanding the quality of training argues that critical points are more likely to be saddle points rather than spurious local minima [21] and that local minima concentrate near the global optimum [22]. Recent work has also revealed that local minima discovered by SGD that lead to good generalization error belong to very flat regions of the parameter space [23]. This motivates algorithms like Entropy-SGD that are specialized to find such regions and draw from similar results in the analysis of binary perceptrons in statistical physics [24]; they have been shown to perform well on deep networks [25]. Surprisingly, these techniques from statistical physics are intimately connected to the regularization properties of partial differential equations (PDEs) [26]. For instance, local entropy, the loss that Entropy-SGD minimizes, is the solution of the **Hamilton-Jacobi-Bellman PDE** and can therefore be written as a **stochastic optimal control** problem, which penalizes greedy gradient descent. This direction further leads to connections between variants of SGD with good empirical performance and standard methods in convex optimization such as inf-convolutions and proximal methods. Researchers are only now beginning to unravel the loss functions of deep networks in terms of their topology, which dictates the complexity of optimization, and their geometry, which seems to be related to generalization properties of the classifiers [27], [28], [29].

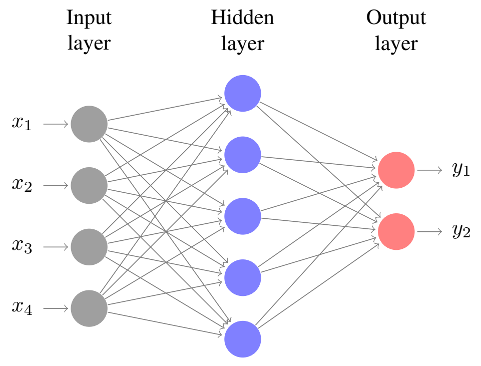

Fig. 1. Illustration of a neural neural network with $D = d_1 = 4$ inputs, $d_2 = 5$ hidden layers, and $C = d_3 = 2$ outputs. Here, the output can be written as $y = (y_1, y_2) = \psi_2(\psi_1(xW^1)W^2)$, where $x = (x_1, \ldots, x_4)$ is the input, $W^1 \in \mathbb{R}^{4\times5}$ is the matrix of weights from the input layer to the hidden layer, $W^2 \in \mathbb{R}^{5\times2}$ is the matrix of weights from the hidden to the output layer, and ψ_1 and ψ_2 are activation functions.

This tutorial will overview recent work showing that the error surface for high-dimensional non-convex optimization problems such as deep learning has some benign properties. For example, the work of [30], [31] shows that for certain classes of neural networks for which both the loss function and the regularizer are sums of positively homogeneous functions of the same degree, a local optimum such that many of its entries are zero is also a global optimum. These results will also provide a possible explanation for the success of RELUs, which are positively homogeneous functions. Particular cases of this framework include, in addition to deep learning, matrix factorization and tensor factorization [32].

E. Paper outline

The remainder of this paper is organized as follows. Section II describes the input-output map of a deep network.

Section III studies the problem of training deep networks and establishes conditions for global optimality. Section IV studies invariance and stability properties of scattering networks. Section V studies structural properties of deep networks, such as metric properties of the embedding as well as bounds on the generalization error. Section VI studies information-theoretic properties of deep representations.

II. Preliminaries

A deep network is a hierarchical model where each layer applies a linear transformation followed by a non-linearity to the preceding layer. Specifically, let $X \in \mathbb{R}^{N\times D}$ be the input data,[1] where each row of X is a D-dimensional data point (e.g., a grayscale image with D pixels) and N is the number of training examples. Let $W^k \in \mathbb{R}^{d_{k-1}\times d_k}$ be a matrix representing a *linear transformation* applied to the output of layer $k-1$, $X_{k-1} \in \mathbb{R}^{N\times d_{k-1}}$, to obtain a d_k-dimensional representation $X_{k-1}W^k \in \mathbb{R}^{N\times d_k}$ at layer k. For example, each column of W^k could represent a convolution with some filter (as in convolutional neural networks) or the application of a linear classifier (as in fully connected networks). Let $\psi_k : \mathbb{R} \to \mathbb{R}$ be a non-linear *activation function*, e.g., a hyperbolic tangent $\psi_k(x) = \tanh(x)$, a sigmoid $\psi_k(x) = (1+e^{-x})^{-1}$, or a rectified linear unit (ReLU) $\psi_k(x) = \max\{0, x\}$.[2] This non-linearity is applied to each entry of $X_{k-1}W^k$ to generate the kth layer of a neural network as $X_k = \psi_k(X_{k-1}W^k)$. The output X_K of the network is thus given by (see Fig. 1):

$$\begin{aligned}\Phi(X, W^1, \ldots, W^K) = \psi_K(\psi_{K-1}(\cdots \\ \psi_2(\psi_1(XW^1)W^2)\cdots W^{K-1})W^K).\end{aligned} \tag{1}$$

Note that Φ is an $N \times C$ matrix, where $C = d_K$ is the dimension of the output of the network, which is equal to the number of classes in the case of a classification problem. Notice also that the map Φ can be seen as a function of the network weights $W = \{W^k\}_{k=1}^K$ with a fixed input X, as will be the case when we discuss the training problem in III. Alternatively, we can view the map Φ as a function of the input data X with fixed weights W, as will be the case when we discuss properties of this map in Sections IV-VI.

III. Global Optimality in Deep Learning

This section studies the problem of learning the parameters $W = \{W^k\}_{k=1}^K$ of a deep network from N training examples (X, Y). In a classification setting, each row of $X \in \mathbb{R}^{N\times D}$ denotes a data point in $\mathbb{R}^D$ and each row of $Y \in \{0,1\}^{N\times C}$ denotes the membership of each data point to one out of C classes, i.e., $Y_{jc} = 1$ if the jth row of X belongs to class $c \in \{1, \ldots, C\}$ and $Y_{jc} = 0$ otherwise. In a regression setting, the rows of $Y \in \mathbb{R}^{N\times C}$ denote the dependent variables for the rows of X. The problem of learning the network weights W is formulated as the following optimization problem:

$$\min_{\{W^k\}_{k=1}^K} \ell(Y, \Phi(X, W^1, \ldots, W^K)) + \lambda\Theta(W^1, \ldots, W^K), \tag{2}$$

[1]For the sake of simplicity, we assume that inputs lie in $\mathbb{R}^D$, but many of the results in this paper apply also to more general inputs such as tensors (e.g., RGB data), in which case the weights become tensors as well.

[2]More broadly, ψ_k could be a many to one function, such as max pooling.

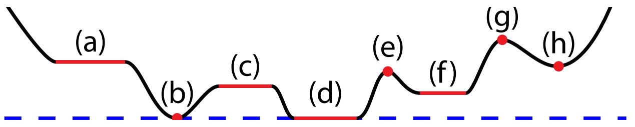

Fig. 2. Example critical points of a non-convex function (shown in red). (a,c) Plateaus. (b,d) Global minima. (e,g) Local maxima. (f,h) Local minima.

where $\ell(Y, \Phi)$ is a *loss function* that measures the agreement between the true output, Y, and the predicted output, $\Phi(X, W)$ in (1), Θ is a *regularization function* designed to prevent overfitting, e.g., weight decay via ℓ_2 regularization $\Theta(W) = \sum_{k=1}^K \|W^k\|_F^2$, and $\lambda > 0$ is a balancing parameter.

A. The challenge of non-convexity in neural network training

A key challenge in neural network training is that the optimization problem in (2) is *non-convex* because, even if the loss $\ell(Y, \Phi)$ is typically a convex function of Φ, e.g., the squared loss $\ell(Y, \Phi) = \|Y - \Phi\|_F^2$, the map $\Phi(X, W)$ is usually a non-convex function of W due to the product of the W^k variables and the nonlinearities ψ_k in (1). This presents significant challenges to existing optimization algorithms – including (but certainly not limited to) gradient descent, stochastic gradient descent, alternating minimization, block coordinate descent, back-propagation, and quasi-Newton methods – which are typically only guaranteed to converge to a critical point of the objective function [33], [34], [35], [36]. However, for non-convex problems, the set of critical points includes not only global minima, but also local minima, local maxima, saddle points and saddle plateaus, as illustrated in Fig. 2. As a result, the non-convexity of the problem leaves the model somewhat ill-posed in the sense that it is not just the model formulation that is important but also implementation details, such as how the model is initialized and particulars of the optimization algorithm, which can have a significant impact on the performance of the model.

To address the issue of non-convexity, a common strategy used in deep learning is to initialize the network weights at random, update these weights using local descent, check if the training error decreases sufficiently fast, and if not, choose another initialization. In practice, it has been observed that if the size of the network is large enough, this strategy can lead to markedly different solutions for the network weights, which give nearly the same objective values and classification performance [22]. It has also been observed that when the size of the network is large enough and the non-linearity is chosen to be a ReLU, many weights are zero, a phenomenon known as *dead neurons*, and the classification performance significantly improves [37], [38], [39], [40]. While this empirically suggests that when the size of the network is large enough and ReLU non-linearities are used *all local minima could be global*, there is currently no rigorous theory that provides a precise mathematical explanation for these experimentally observed phenomena.

B. Optimality for neural networks with a single hidden layer

Earlier work on global optimality for neural networks [41] showed that for networks with a single hidden layer

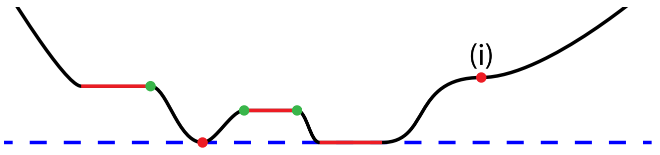

Fig. 3. Illustration of the properties of the framework of [30], [31]. Starting from any initialization, a non-increasing path exists to a global minimizer. Starting from points on a plateau, a simple "sliding" method exists to find the edge of the plateau (green points).

and linear activations, the squared loss has a unique global minimum and all other critical points are saddle points. However, when the activations are nonlinear, [42] gives examples of networks where the backpropagation algorithm [19] fails even with separable data. The examples are, however, not generic, and [43], [44] show that backpropagation generally finds a global minimizer for linearly separable data.

Later work [45] showed that for neural networks with a single hidden layer, if the number of neurons in the hidden layer is not fixed, but instead fit to the data through a sparsity inducing regularization, then the process of training a globally optimal neural network is analogous to selecting a finite number of hidden units from a potentially infinite dimensional space of all possible hidden units. A weighted sum of the selected hidden units is then taken to produce the output. The specific optimization problem is of the form

$$\min_{w} \ell(Y, \sum_{i} h_i(X) w_i) + \lambda \|w\|_1, \tag{3}$$

where $h_i(X)$ represents one possible hidden unit activation in response to the training data X from an infinite dimensional space $h_i(X) \in \mathcal{H}$ of all possible hidden unit activations. Clearly (3) is a convex optimization problem on w (assuming $\ell(Y, w)$ is convex on w) and straightforward to solve for a finite set of $h_i(X)$ activations. However, since $\mathcal{H}$ is an infinite dimensional space the primary difficulty lies in how to select the appropriate hidden unit activations. Nonetheless, by using arguments from gradient boosting, it is possible to show that problem (3) can be globally optimized by sequentially adding hidden units to the network until one can no longer find a hidden unit whose addition will decrease the objective function [45], [46], [47].

C. *Optimality for networks with random inputs and weights*

Recent work has analyzed the problem of training neural networks with random inputs and weights. For example, the authors of [48] study random networks with a single hidden layer. Rather than training the network by solving an optimization problem such as (2), the authors use tensor decomposition methods to estimate the network weights from high order statistical moments of the network mapping. Moreover, the authors show that under certain assumptions on the loss and data distribution, polynomial-time training is possible [48]. Further, the authors of [49] study the problem when a given initialization of a neural network is likely to be within the basin of attraction of a global minimizer and provide conditions that ensure a random initialization will be within the basin of a global minimizer with high probability.

Several recent works have also analyzed the error surface of multilayer neural networks using tools from random matrix theory and statistical physics. For example, the authors of [21] argue that, under certain assumptions, it is vastly more likely that a critical point of a high-dimensional optimization problem be a saddle point rather than a local minimizer. Therefore, avoiding saddle points is the key difficulty in high-dimensional, non-convex optimization. Likewise, the authors of [22] show that, under certain assumptions on the distributions of the training data and the network weight parameters, as the number of hidden units in a network increases, the distribution of local minima becomes increasingly concentrated in a small band of objective function values near the global optimum (and thus all local minima become increasingly close to being global minima).

D. *Global optimality for positively homogeneous networks*

Recent work [30], [31] largely echoes ideas from the above work, but takes a markedly different approach. Specifically, [30], [31] analyzes the optimization problem in (2) using a purely deterministic approach which does not make any assumptions regarding the distribution of the input data, the network weight parameter statistics, or the network initialization. With this approach, [30], [31] shows that saddle points and plateaus are the *only* critical points that one needs to be concerned with due to the fact that for networks of sufficient size, local minima that require one to climb the objective surface to escape from, such as (f) and (h) in Fig. 2, are guaranteed not to exist.

More specifically, [30] studies conditions under which the optimization landscape for the non-convex optimization problem in (2) is such that *all critical points are either global minimizers or saddle points/plateaus*, as shown in Fig. 3. The authors show that if the network size is large enough and the functions Φ and Θ are *sums of positively homogeneous functions of the same degree*, any local minimizer such that *some of its entries are zero* is also a global minimizer. Interestingly, ReLU and max-pooling non-linearities are positively homogeneous, while sigmoids are not, which could provide a possible explanation for the improved performance of ReLU and max pooling. Furthermore, many state-of-the-art networks are not trained with *classical regularization*, such as an ℓ_1 or ℓ_2 norm penalty on the weight parameters but instead rely on techniques such as dropout [11]. The results of [30], [31] also provide strong guidance on the design of network regularization to ensure the non-existence of spurious local minima, showing that traditional weight decay is not appropriate for deep networks. However, more recently proposed forms of regularization such as Path-SGD [50] or batch normalization [51] can be easily incorporated into the analysis framework of [30], [31], and stochastic regularization methods, such as dropout [11], have strong similarities to their framework.

IV. Geometric Stability in Deep Learning

An important question in the path towards understanding deep learning models is to mathematically characterize its inductive bias; i.e., define the class of regression/classification tasks for which they are predesigned to perform well, or at least better than classical alternatives.

In the particular case of computer vision tasks, convolutional archictectures provide a fundamental inductive bias at the origin of most successful deep learning vision models. As we explain next, the notion of geometric stability provides a possible framework to understand its success.

Let $\Omega = [0,1]^d \subset \mathbb{R}^d$ be a compact d-dimensional Euclidean domain on which square-integrable functions $X \in L^2(\Omega)$ are defined (for example, in image analysis applications, images can be thought of as functions on the unit square $\Omega = [0,1]^2$). In a supervised learning task, an unknown function $f : L^2(\Omega) \to \mathcal{Y}$ is observed on a training set

$$\{X_i \in L^2(\Omega), Y_i = f(X_i)\}_{i \in \mathcal{I}} \ , \tag{4}$$

where the target space $\mathcal{Y}$ can be thought as being discrete in a standard classification setup (with $C = |\mathcal{Y}|$ being the number of classes), or $\mathcal{Y} = \mathbb{R}^C$ in a regression task.

In the vast majority of computer vision and speech analysis tasks, the unknown function f typically satisfies the following crucial assumptions:

1) ***Stationarity:*** Consider a *translation operator*[3]

$$\mathcal{T}_v X(u) = X(u - v), \quad u, v \in \Omega, \tag{5}$$

acting on functions $X \in L^2(\Omega)$. Depending on the task, we assume that the function f is either *invariant* or *equivariant* with respect to translations. In the former case, we have $f(\mathcal{T}_v X) = f(X)$ for any $X \in L^2(\Omega)$ and $v \in \Omega$. This is typically the case in object classification tasks. In the latter, we have $f(\mathcal{T}_v X) = \mathcal{T}_v f(X)$, which is well-defined when the output of the model is a space in which translations can act upon (for example, in problems of object localization, semantic segmentation, or motion estimation). Our definition of invariance should not be confused with the traditional notion of *translation invariant systems* in signal processing, which corresponds to translation equivariance in our language (since the output translates whenever the input translates).

2) ***Local deformations and scale separation*:** Similarly, a deformation $\mathcal{L}_\tau$, where $\tau : \Omega \to \Omega$ is a smooth vector field, acts on $L^2(\Omega)$ as $\mathcal{L}_\tau X(u) = X(u - \tau(u))$. Deformations can model local translations, changes in viewpoint, rotations and frequency transpositions [9]. Most tasks studied in computer vision are not only translation invariant/equivariant, but, more importantly, also stable with respect to local deformations [52], [9]. In tasks that are translation invariant we have

$$|f(\mathcal{L}_\tau X) - f(X)| \approx \|\nabla \tau\| \ , \tag{6}$$

for all X, τ, where $\|\nabla\tau\|$ measures the smoothness of a given deformation field. In other words, the quantity to be predicted does not change much if the input image is slightly deformed. In tasks that are translation equivariant, we have

$$|f(\mathcal{L}_\tau X) - \mathcal{L}_\tau f(X)| \approx \|\nabla \tau\|. \tag{7}$$

This property is much stronger than stationarity, since the space of local deformations has high dimensionality – of the order of $\mathbb{R}^D$ when we discretize images with D pixels, as opposed to the d-dimensional translation group, which has only $d = 2$ dimensions in the case of images.

Assumptions (6)–(7) can be leveraged to approximate f from features $\Phi(X)$ that progressively reduce the spatial resolution. Indeed, extracting, demodulating and downsampling localized filter responses creates local summaries that are insensitive to local translations, but this loss of loss of sensitivity does not affect our ability to approximate f, thanks to (6)–(7). To illustrate this principle, denote by

$$Z(a_1, a_2; v) = \text{Prob}(X(u) = a_1 \text{ and } X(u+v) = a_2) \tag{8}$$

the joint distribution of two image pixels at an offset v from each other. In the presence of long-range statistical dependencies, this joint distribution will not be separable for any v. However, the deformation stability prior states that $Z(a_1, a_2; v) \approx Z(a_1, a_2; v(1+\epsilon))$ for small ϵ. In other words, whereas long-range dependencies indeed exist in natural images and are critical to object recognition, they can be captured and down-sampled at different scales. Although this principle of stability to local deformations has been exploited in the computer vision community in models other than CNNs, for instance, deformable parts models [53], CNNs strike a good balance in terms of approximation power, optimization, and invariance.

Indeed, stationarity and stability to local translations are both leveraged in convolutional neural networks (CNN). A CNN consists of several *convolutional layers* of the form $\tilde{X} = C_W(X)$, acting on a p-dimensional input $X(u) = (X_1(u), \ldots, X_p(u))$ by applying a bank of filters $W = (w_{l,l'})$, $l = 1, \ldots, q, l' = 1, \ldots, p$ and point-wise non-linearity ψ,

$$\tilde{X}_l(u) \quad = \quad \psi\left(\sum_{l'=1}^{p} (X_{l'} \star w_{l,l'})(u)\right), \tag{9}$$

producing a q-dimensional output $\tilde{X}(u) = (\tilde{X}_1(u), \ldots, \tilde{X}_q(u))$ often referred to as the *feature maps*. Here,

$$(X \star w)(u) = \int_\Omega X(u - u') w(u') du' \tag{10}$$

denotes the standard convolution. According to the local deformation prior, the filters W have compact spatial support.

Additionally, a downsampling or *pooling* layer $\tilde{X} = P(X)$ may be used, defined as

$$\tilde{X}_l(u) = P(\{X_l(u') \ : \ u' \in \mathcal{N}(u)\}), \ \ l = 1, \ldots, q, \tag{11}$$

where $\mathcal{N}(u) \subset \Omega$ is a neighborhood around u and P is a permutation-invariant function such as an *average-*, *energy-*, or *max-pooling*).

[3] We assume periodic boundary conditions to ensure that the operation is well-defined over $L^2(\Omega)$.

A convolutional network is constructed by composing several convolutional and optionally pooling layers, obtaining a generic hierarchical representation

$$\Phi_{\boldsymbol{W}}(X) = (C_{W^{(K)}} \cdots P \cdots \circ C_{W^{(2)}} \circ C_{W^{(1)}})(X), \quad (12)$$

where $\boldsymbol{W} = \{W^{(1)}, \ldots, W^{(K)}\}$ is the hyper-vector of the network parameters (all the filter coefficients). The output features enjoy translation invariance/covariance depending on whether spatial resolution is progressively lost by means of pooling or kept fixed. Moreover, if one specifies the convolutional tensors to be complex wavelet decomposition operators and uses complex modulus as point-wise nonlinearities, one can provably obtain stability to local deformations [54]. Although this stability is not rigorously proved for generic compactly supported convolutional tensors, it underpins the empirical success of CNN architectures across a variety of computer vision applications [1].

A key advantage of CNNs explaining their success in numerous tasks is that the geometric priors on which CNNs are based result in a sample complexity that avoids the curse of dimensionality. Thanks to the stationarity and local deformation priors, the linear operators at each layer have a constant number of parameters, independent of the input size D (number of pixels in an image). Moreover, thanks to the multiscale hierarchical property, the number of layers grows at a rate $\mathcal{O}(\log D)$, resulting in a total learning complexity of $\mathcal{O}(\log D)$ parameters.

Finally, recently there has been an effort to extend the geometric stability priors to data that is not defined over an Euclidean domain, where the translation group is generally not defined. In particular, researchers are exploiting geometry in general graphs via the spectrum of graph Laplacians and its spatial counterparts; see [55] for a recent survey on those advances.

V. Structure Based Theory for Deep Learning

A. *Structure of the data throughout a neural network*

An important aspect for understanding better deep learning is the relationship between the structure of the data and the deep network. For a formal analysis, consider the case of a network that has random i.i.d. Gaussian weights, which is a common initialization in training deep networks. Recent work [56] shows that such networks with random weights preserve the metric structure of the data as they propagate along the layers, allowing for stable recovery of the original data from the features calculated by the network – a property that is often encountered in general deep networks [57], [58].

More precisely, the work of [56] shows that the input to the network can be recovered from the network's features at a certain layer if their size is proportional to the intrinsic dimension of the input data. This is similar to data reconstruction from a small number of random projections [59], [60]. However, while random projections preserve the Euclidean distance between two inputs up to a small distortion, each layer of a deep network with random weights distorts this distance proportionally to the angle between the two inputs: the smaller the angle, the stronger the shrinkage of the distance. Therefore, the deeper the network, the stronger the shrinkage achieved. Note that this does not contradict the fact that it is possible to recover the input from the output; even when properties such as lighting, pose and location are removed from an image (up to certain extent), the resemblance to the original image is still maintained.

As random projection is a universal sampling strategy for low-dimensional data [59], [60], [61], deep networks with random weights are a universal system that separates any data (belonging to a low-dimensional model) according to the angles between the data points, where the general assumption is that there are large angles between different classes [62], [63]. As the training of the projection matrix adapts it to better preserve specific distances over others, the training of a network prioritizes intra-class angles over inter-class ones. This relation is alluded to by the proof techniques in [56] and is empirically manifested by observing the angles and Euclidean distances at the output of trained networks.

By using the theory of 1-bit compressed sensing, [64] shows that each layer of a network preserves the metric of its input data in the Gromov-Hausdorff sense up to a small constant δ, under the assumption that these data reside in a low-dimensional manifold denoted by K. This allows drawing conclusions on the tessellation of the space created by each layer and the relationship between the operation of these layers and local sensitive hashing (LSH). It also implies that it is possible to retrieve the input of a layer, up to certain accuracy, from its output. This shows that every layer preserves the important information of the data.

An analysis of the behavior of the Euclidean distances and angles in the data along the network reveals an important effect of the ReLU. Without a non-linearity, each layer is simply a random projection for which Euclidean distances are approximately preserved. The addition of a ReLU makes the system sensitive to the angles between points. In this case, the network tends to decrease the Euclidean distances between points with small angles between them ("same class"), more than the distances between points with large angles between them ("different classes"). Still, low-dimensional data at the input remain such throughout the entire network, i.e., deep networks (almost) do not increase the intrinsic dimension of the data [56]. This is related to the recent work in [65] that claims that deep networks may be viewed a sparse coding procedure leading to guarantees on the uniqueness of the representation calculated by the network and its stability.

As random networks are blind to the data labels, training may select discriminatively the angles that cause the distance deformation. Therefore, it will cause distances between different classes to increase more than the distances within the same class. This is demonstrated in several simulations for different deep networks. It is observed that a potential main goal of the training of the network is to treat the class boundary points while keeping the other distances approximately the same.

B. Generalization error

The above suggests that there is a relation between the structure of the data and the error the network achieves in training, which leads to study the relationship between the generalization error in deep networks and the data structure. Generalization error – the difference between the empirical error and the expected error – is a fundamental concept in statistical learning theory. Generalization error bounds offer insight into why learning from training samples is possible.

Consider a classification problem with a data point $X \in \mathcal{X} \subseteq \mathbb{R}^D$ that has a corresponding class label $Y \in \mathcal{Y}$, where C is the number of classes. A training set of N samples drawn from a distribution P is denoted by $\Upsilon_N = \{(X_i, Y_i)\}_{i=1}^N$ and the loss function is denoted by $\ell(Y, \Phi(X, W))$, which measures the discrepancy between the true label Y and the estimated label $\Phi(X, W)$ provided by the classifier. The empirical loss of a network $\Phi(\cdot, W)$ associated with the training set Υ_N is defined as

$$\ell_{\text{emp}}(\Phi) = \frac{1}{N} \sum_{X_i \in \Upsilon_N} \ell\left(Y_i, \Phi(X_i, W)\right), \tag{13}$$

and the expected loss as

$$\ell_{\text{exp}}(\Phi) = \mathbb{E}_{(X,Y)\sim P}\left[\ell\left(Y, \Phi(X, W)\right)\right]. \tag{14}$$

The *generalization error* is then given as:

$$\text{GE}(\Phi) = \left|\ell_{\text{exp}}(\Phi) - \ell_{\text{emp}}(\Phi)\right|. \tag{15}$$

Various measures such as the the VC-dimension [66], [67], the Rademacher or Gaussian complexities [68] and algorithm robustness [69] have been used to bound the GE in the context of deep networks. However, these measures do not offer an explanation for good deep network generalization in practice where the number of parameters can often be larger than the number of training samples [70] or the networks are very deep [71]. For example, the VC-dimension of a deep network 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. On the other hand, the work [72] bounds the generalization error independently of the number of parameters. Yet, in its bound the generalization error of deep network with ReLUs scales exponentially with the network depth. Similarly, the authors in [69] show that deep networks 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.

An alternative route followed by [28], [29], [69], [73] bounds the generalization error in terms of the networks' classification margin, which is independent of the depth and size but takes into account the structure of the data (considering its covering number) and therefore avoids the above issues. As it is hard to calculate the input margin directly, in [28] it is tied to the Jacobian matrix and the loss of the deep networks showing that bounding the spectral norm of the Jacobian matrix reduces the generalization error. This analysis is general to arbitrary network structures, types of non-linearities and pooling operations. Furthermore, a relationship between the generalization error, the invariance in the data and the network is formally characterized in [29].

TABLE I
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**

Using the relationship between the generalization error and the network Jacobian matrix, a new Jacobian based regularization strategy is developed in [29] and its advantage is demonstrated for several networks and datasets. For example, Table I shows the improvement achieved when using this regularization with the Wide ResNet architecture [74] for CIFAR-10 with different numbers of training examples.

A related theory to the one presented above is the one in [50], [75]. These works study the relationship between the generalization error and the minimization of the network loss using SGD. They provide modifications for SGD that improves the error achieved by the network.

VI. TOWARDS AN INFORMATION-THEORETIC FRAMEWORK

The loss function of choice for training deep networks to solve supervised classification problems is the *empirical cross-entropy*

$$\tilde{\ell}(W) = \mathbb{E}_{P(X,Y)}(-\log \Phi(X, W)), \tag{16}$$

This loss function is prone to overfitting, as the network could trivially memorize the training data instead of learning the underlying distribution. This problem is usually addressed by regularization, which can be explicit (e.g., the norm of W, known as *weight decay*), or implicit in stochastic gradient descent. It was suggested by [76] almost a quarter century ago that better regularization, hence less overfitting, might be achieved by limiting the information stored in the weight, $\text{KL}(\, p(W|X,Y) \,\|\, p(W)\,)$, where $p(W)$ is a prior on the weights. Choosing this regularizer leads to the loss function

$$\ell(W) = H(Y|X, W) + \lambda\, \text{KL}(\, p(W|X,Y) \,\|\, p(W)\,) \tag{17}$$

where the first term denotes the empirical conditional cross-entropy obtained from $\tilde{\ell}(W)$. For $\lambda = 1$, this is the variational lower bound on the observed data distribution $p_\theta(Y|X)$, and can therefore be seen as a form of **Bayesian inference** of the weights. More generally, this is equivalent to the information bottleneck Lagrangian. The first term is the same as the empirical cross-entropy and ensures that information stored in the weights is *sufficient* for the task Y, while the second term minimizes the amount of information stored. Thus, the weights learned by minimizing cross-entropy, with a KL regularizer, approximate a minimal sufficient statistic of the training set. Computing the KL term and optimizing ℓ was considered a show-stopper until recently, when advances in Stochastic Gradient Variational Bayes [77], [78] made efficient optimization possible.

But for a representation to be useful, it should not just efficiently memorize past (training) data. It should also reduce the effect of nuisance variability affecting future (test) data. Indeed, most of the variability in imaging data can be ascribed to the effects of illumination and viewpoint, quotienting out which leaves a thin set [79]. As we have already pointed out, deep networks are known to reduce the effects of nuisance variability in test data. This can be partly achieved through the architecture, in particular multi-scale convolutional structures. Some may be ascribed to the optimization procedure, that converges to "flat minima." But the choice of regularizer is also responsible for the networks' ability to discount nuisance variability. Denoting by X be the input sample, Y the target variable, and $Z \sim p(Z|X)$ the (stochastic) representation of X learned by a layer in the network, the tradeoff between sufficiency an minimality of Z is formalized by the information bottleneck Lagrangian

$$\ell(W) = H(Y \mid Z, W) + \lambda I(Z; X), \tag{18}$$

where the first term ensures that the representation Z is sufficient for Z, while the second term ensures its information content remains minimal, i.e. that nuisances are filtered out.

Notice that, while formally equivalent, the losses in (17) and (18) are conceptually different: In the former, the weights are a representation of the *training set* that is minimal and sufficient. In the latter, the activations are a minimal representation of the *test sample*. We conjecture that the two are related, and that the relation informs the generalization properties of the network, but specific bounds have yet been shown.

Different choices of the noise distribution, and different models of the marginal distribution $p(Z)$ lead to slightly different analyses. For example, [18] considers the case of additive Gaussian noise and Gaussian marginals, while [17] study multiplicative noise distributions, and considers both a scale invariant log-uniform marginal (the only one compatible with the fact that networks with ReLU activations are scale invariant), and a log-normal marginal distribution. Interestingly, the special case in which the multiplicative noise is chosen from a Bernoulli distribution reduces to the well-known practice of Dropout [76], while choosing from a multiplicative Gaussian distribution leads to Gaussian Dropout [78].

In order to compute the term $I(Z;X)$, it is commonly assumed that the activations are mutually independent, that is, that the marginal $p(Z)$ is factorized; [17] shows that making this assumption corresponds to minimizing a modified loss function, which also reduces the total correlation $\mathrm{TC}(Z)$ of the activations. Therefore, a choice dictated by convenience in order to explicitly compute the information bottleneck Lagrangian, yields a disentangled representation, with entanglement measured by total correlation.

It is remarkable that empirical practice has managed to converge to the use of the cross-entropy loss with dropout, that happens to be what would have been prescribed by first principles, since for certain choices of distribution, training is equivalent to minimizing the information bottleneck Lagrangian, that yields an approximation of a minimal sufficient invariant statistic, which define an optimal representation. It is only recently that developments in theory have made this association possible, and developments in optimization and hardware have made deep neural networks a sensational success.

VII. Acknowledgments

René Vidal acknowledges grant NSF 1618485. Raja Giryes acknowledges the Global Innovation Fund (GIF). Stefano Soatto acknowledges grants ONR N00014-15-1-2261, ARO W911NF-15-1-0564/66731-CS, and AFOSR FA9550-15-1-0229.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
[2] V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International conference on machine learning*, pages 807–814, 2010.
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.
[4] Jia Deng, Wei Dong, R Socher, Li-Jia Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, June 2009.
[5] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems*, 2(4):303–314, 1989.
[6] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
[7] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.
[8] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14(1):115–133, 1994.
[9] J. Bruna and S. Mallat. Invariant scattering convolution networks. *Trans. PAMI*, 35(8):1872–1886, 2013.
[10] P. Bartlett and W. Maass. Vapnik-Chervonenkis dimension of neural nets. *The handbook of brain theory and neural networks*, pages 1188–1192, 2003.
[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[12] R. Giryes, G. Sapiro, and A. Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.
[13] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the Allerton Conf.*, 2000.
[14] S. Soatto and A. Chiuso. Visual representations: Defining properties and deep approximations. *Proc. of the Intl. Conf. on Learning Representations (ICLR); ArXiv: 1411.7676*, May 2016.
[15] F. Anselmi, L. Rosasco, and T. Poggio. On invariance and selectivity in representation learning. *arXiv preprint arXiv:1503.05938*, 2015.
[16] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.
[17] A. Achille and S. Soatto. Information dropout: Learning optimal representations through noisy computation. *arXiv:1611.01353*, 2016.
[18] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.
[19] PJ Werbos. *Beyond regression: New tools for predictions and analysis in the behavioral science. Cambridge, MA, itd.* PhD thesis, Harvard University, 1974.

[20] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, pages 1–30, 2013.

[21] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Neural Information Processing Systems*, pages 2933–2941, 2014.

[22] A. Choromanska, M. Henaff, M. Mathieu, G.B Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *International Conference on Artificial Intelligence and Statistics*, pages 192–204, 2015.

[23] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

[24] C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina. Local entropy as a measure for sampling solutions in constraint satisfaction problems. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(2):023301, 2016.

[25] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. *International Conference of Learning and Representations*, 2017.

[26] Pratik Chaudhari, Adam Oberman, Stanley Osher, Stefano Soatto, and Guillaume Carlier. Deep relaxation: partial differential equations for optimizing deep neural networks. *arXiv:1704.04932*, 2017.

[27] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. *ICLR*, 2017.

[28] J. Sokolić, R. Giryes, G. Sapiro, and M. Rodrigues. Robust large margin deep neural networks. *to appear in IEEE Transactions on Signal Processing*, 2017.

[29] J. Sokolić, R. Giryes, G. Sapiro, and M. Rodrigues. Generalization error of invariant classifiers. In *AISTATS*, 2017.

[30] B. Haeffele and R. Vidal. Global optimality in neural network training. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[31] B. Haeffele and R. Vidal. Global optimality in tensor factorization, deep learning, and beyond. *arXiv*, abs/1506.07540, 2015.

[32] B. Haeffele, E. Young, and R. Vidal. Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In *International Conference on Machine Learning*, pages 2007–2015, 2014.

[33] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[34] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5, 1988.

[35] Stephen J Wright and Jorge Nocedal. *Numerical Optimization*, volume 2. Springer New York, 1999.

[36] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

[37] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8609–8613, 2013.

[38] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, volume 30, 2013.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, pages 1097–1105, 2012.

[40] Matthew D. Zeiler, M. Ranzato, Rajat Monga, M. Mao, K. Yang, Quoc Viet Le, Patrick Nguyen, A. Senior, Vincent Vanhoucke, Jeffrey Dean, and Geoffry E. Hinton. On rectified linear units for speech processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521, 2013.

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

[42] Martin Brady, Raghu Raghavan, and Joseph Slawny. Back propagation fails to separate where perceptrons succeed. *IEEE Transactions on Circuits and Systems*, 36(5):665–674, 1989.

[43] M. Gori and A. Tesi. Backpropagation converges for multi-layered networks and linearly-separable patterns. In *International Joint Conference on Neural Networks*, volume 2, page 896. IEEE, 1991.

[44] M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

[45] Yoshua Bengio, Nicolas L Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. In *Neural Information Processing Systems*, pages 123–130, 2005.

[46] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[47] Llew Mason, Jonathan Baxter, Peter L. Bartlett, and Marcus R. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, pages 512–518, 2000.

[48] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015.

[49] Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pages 774–782, 2016.

[50] B. Neyshabur, R. Salakhutdinov, and N. Srebro. Path-SGD: Path-normalized optimization in deep neural networks. In *Neural Information Processing Systems*, pages 2422–2430, 2015.

[51] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[52] Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065), 2016.

[53] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[54] Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[55] M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond Euclidean data. *arXiv preprint arXiv:1611.08097*, 2016.

[56] R. Giryes, G. Sapiro, and A. Bronstein. Deep neural networks with random Gaussian weights: a universal classification strategy? *IEEE Trans. Sig. Proc.*, 64(13):3444–3457, July 2016.

[57] J. Bruna, A. Szlam, and Y. LeCun. Signal recovery from l_p pooling representations. In *Int. Conf. on Machine Learning (ICML)*, pages 307–315, 2014.

[58] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015.

[59] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inf. Theory*, 52(12):5406 –5425, Dec. 2006.

[60] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6):805–849, 2012.

[61] R. Giryes, Y. Eldar, A. Bronstein, and G. Sapiro. Tradeoffs between convergence speed and reconstruction accuracy in inverse problems. *arXiv:1605.09232*, 2017.

[62] L. Wolf and A. Shashua. Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, 4:913–931, Oct. 2003.

[63] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[64] Y. Plan and R. Vershynin. Dimension reduction by random hyperplane tessellations. *Discrete and Computational Geometry*, 51(2):438–461, 2014.

[65] V. Papyan, Y. Romano, and M. Elad. Convolutional Neural Networks Analyzed via Convolutional Sparse Coding. *arXiv:1607.08194*, 2016.

[66] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, September 1999.

[67] S Shalev-Shwartz and S Ben-David. *Understanding machine learning: from theory to algorithms*. Cambridge University Press, 2014.

[68] Peter L Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[69] Huan Xu and Shie Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012.

[70] C. Zhang, S. Bengio, M. Hardt, and B. Recht. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, December 2016.

[72] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401, 2015.

[73] J. Huang, Q. Qiu, G. Sapiro, and R. Calderbank. Discriminative robust transformation learning. In *Neural Information Processing Systems*, pages 1333–1341, 2015.

[74] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv:1605.07146*, 2016.

[75] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1225–1234. JMLR.org, 2016.

[76] G. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Annual Conference on Computational Learning Theory*, pages 5–13. ACM, 1993.

[77] D. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Neural Information Processing Systems*, pages 2575–2583, 2015.

[78] D. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[79] G. Sundaramoorthi, P. Petersen, V. S. Varadarajan, and S. Soatto. On the set of images modulo viewpoint and contrast changes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.