Deep Neural Networks with Random Gaussian Weights: A Universal Classification Strategy?

Raja Giryes*, Guillermo Sapiro*, and Alex M. Bronstein**

*Department of Electrical and Computer Engineering, Duke University, Durham, North Carolina, 27708, USA.
{raja.giryes,guillermo.sapiro}@duke.edu.

**School of Electrical Engineering, Faculty of Engineering, Tel-Aviv University, Ramat Aviv 69978, Israel.
bron@eng.tau.ac.il

***Abstract*—Three important properties of a classification machinery are: (i) the system preserves the important information of the input data; (ii) the training examples convey information for unseen data; and (iii) the system is able to treat differently points from different classes. In this work we show that these fundamental properties are inherited by the architecture of deep neural networks. We formally prove that these networks with random Gaussian weights perform a distance-preserving embedding of the data, with a special treatment for in-class and out-of-class data. Similar points at the input of the network are likely to have the same The theoretical analysis of deep networks here presented exploits tools used in the compressed sensing and dictionary learning literature, thereby making a formal connection between these important topics. The derived results allow drawing conclusions on the metric learning properties of the network and their relation to its structure; and provide bounds on the required size of the training set such that the training examples would represent faithfully the unseen data. The results are validated with state-of-the-art trained networks.**

I. Introduction

Deep neural networks (DNN) have led to a revolution in the areas of machine learning, audio analysis, and computer vision, achieving state-of-the-art results in numerous applications [1], [2], [3]. In this work we formally study the properties of deep network architectures with random weights and data that reside in a low dimensional manifold. Our results provide insights into the outstanding empirically observed performance of DNN, the role of training, and the size of the training data.

Our motivation for studying networks with random weights is twofold. First, a series of works [4], [5], [6] empirically showed successful DNN learning techniques based on randomization. Second, studying a system with random weights rather than learned deterministic ones may lead to a better understanding of the system even in the deterministic case. For example, in the compressed sensing arena, where the goal is to recover a signal from a small number of its measurements, the study of random sampling operators led to breakthroughs in the understanding of the number of measurements required for achieving a stable reconstruction [7]. While the bounds provided in this case are universally optimal, the introduction of a learning phase provides a better reconstruction performance as it adapts the system to the particular data at hand [8], [9], [10].

Notice that the technique of proving results for deep learning with assumptions on some random distribution and then showing that the same holds in the more general case is not unique only to our work. On the contrary, some of the stronger recent theoretical results on DNN follow this approach. For example, Arora et al. analyzed the learning of autoencoders with random weights in the range $[-1, 1]$, showing that it is possible to learn them in polynomial time under some restrictions on the depth of the network [11]. Another example is the series of works [12], [13], [14] that study the optimization perspective of DNN.

Another example of the importance of analyzing random weights comes in the context of phase retrieval, where the goal is to recover the phase of a signal from the magnitudes of complex inner-products. In standard applications, the inner-products are performed between the signal and complex exponentials with different frequencies. Analyzing this problem under the assumption of inner products with random vectors led to a fundamental and novel understanding of the problem [15].

Therefore, in this work we study the properties of deep networks under the assumption of random weights. Before we turn to describe our contribution, we survey previous studies that formally analyzed the role of deep networks.

Hornik et al. proved that neural networks serve as a universal approximation for any measurable Borel functions [16]. However, finding the network's weights for a given function was shown to be NP-hard.

Montúfar and Morton showed that the depth of DNN allows representing restricted Boltzmann machines with a number of parameters exponentially greater than the number of the network parameters [17]. Montúfar et al. suggest that each layer divides the space by a hyper-plane [18]. Therefore, the depth of the network divides the space into an exponential number of sets, which is not achievable with a single layer with the same number of parameters.

Bruna et al. showed that the pooling stage in DNN result in shift-invariance [19]. In [20] they interpret this step as the removal of phase from a complex signal and show how the signal may be recovered after a pooling stage using phase retrieval methods. This work also calculates the Lipschitz constants of the pooling and the rectified linear unit (ReLU) stages, showing that they perform a stable embedding of

the data under the assumption that the filters applied in the network are frames, i.e., for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we have

$$A \left\| \mathbf{x} - \mathbf{y} \right\|_2 \leq \left\| \rho(\mathbf{M}\mathbf{x}) - \rho(\mathbf{M}\mathbf{y}) \right\|_2 \leq B \left\| \mathbf{x} - \mathbf{y} \right\|_2, \quad (1)$$

where $\mathbf{M} \in \mathbb{R}^{m \times n}$ is linear operator at a given layer in the network and $\rho(x) = \max(0, x)$ is the ReLU operator (applied element-wise). However, it is not clear what the values of these Lipschitz constants A and B are in real networks and how they scale with the data dimension. To see why such a bound may be very loose, consider the output of a fully connected layer with random i.i.d Gaussian $N(0, \frac{1}{m})$ weights, which is a standard initialization in deep learning, followed by a ReLU. Then A and B scale like $\sqrt{\frac{2n}{m}}\left(1 \pm \sqrt{\frac{m}{2n}}\right)$ respectively [21]. Note that this undesired behavior is not unique to $\mathbf{M}$ with Gaussian distribution, characterizing any distribution with a bounded fourth moment.

A. Contributions

Such a scaling of the distances may drastically deform the distances throughout each layer even in the case where m is very close to n, and it is unknown whether there is a correspondence between the metric of the input and output data of each layer. To have such a correspondence in the Lipschitz sense we need the constants A and B to be approximately equal.

In this work, the main question we focus on is: What happens to the metric of the input data throughout the network? As mentioned above, throughout the work we assume that the network has random i.i.d. Gaussian weights. We prove that DNN preserve the metric structure of the data as it propagates along the layers, allowing for the stable recovery of the original data from the features calculated by the network. However, unlike random projections that preserve the Euclidean distances up to a small distortion [22], each layer of DNN with random weights distorts these distances proportionally to the angles between its input points: the smaller the angle at the input the stronger the shrinkage of the distances. Therefore, the deeper the network the stronger the shrinkage we get.

As random projection is a universal sampling strategy for any low dimensional data [7], [23], [24], deep networks with random weights are a universal system that separates any data (belonging to a low dimensional model) according to the angles between its points, where the general assumption is that there are large angles between different classes [25], [26], [27], [28]. As training adapts the sampling matrix to better preserve specific distances over others, the training of a network prioritizes intra-class angles over inter-class ones. This is alluded by our proof techniques and observed by looking at the angles and Euclidean distances at the output of trained networks, as demonstrated in this paper.

In Section II, we start by utilizing the recent theory of 1-bit compressed sensing to show that each DNN layer 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 K.

In Section III, we continue by analyzing the behavior of the Euclidean distances and angles in the data throughout the network. This section reveals an important effect of the ReLU. Without the ReLU, we would just have random projections and Euclidean distances preservation. Our theory shows that the addition of ReLU turns the system to become sensitive to the angles between points. We prove that networks tend to keep the Euclidean and angular distances between points with a small angle between them ("same class"), and to increase these distances between points with large angles between them ("different classes").

Then, in Section IV we prove that if the data are low-dimensional at the input of the network, then the same also holds throughout the whole net, i.e., DNN (almost) do not increase the intrinsic dimension of the data. This property is used in Section V to deduce the size of data needed for training DNN.

We conclude by studying the role of training in Section VI. As random networks are blind to the labels of the data, 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. We demonstrate this in several simulations, some of which with networks that recently showed state-of-the-art performance in challenging datasets, e.g., the network by [29] for the ImageNet dataset [30]. Section VII concludes the paper.

For the sake of simplicity of the discussion and presentation clarity, we focus only on the role of the ReLU operator [31], assuming that our data are properly aligned. Utilizing recent results for the phase retrieval problem, together with the proof techniques in this paper, can lead to theory that covers also the pooling operation. In addition, with the strategy in [32], [33], [34], [35], it is possible to generalize our guarantees to subgaussian distributions and to random convolutional filters. We defer these natural extensions to a future work.

II. Stable Embedding of a Single Layer

In this section we consider the Gromov-Hausdorff distance between the input and output of a given DNN layer of the form $f(\mathbf{M} \ \cdot \ )$, mapping an input vector $\mathbf{x}$ to the output vector $f(\mathbf{M}\mathbf{x})$, where $\mathbf{M}$ is a random Gaussian matrix and $f : \mathbb{R} \to \mathbb{R}$ is a semi-truncated linear function. f is such if it is linear on some (possibly, semi-infinite) interval and constant outside of it, $f(0) = 0$, $0 < f(x) \leq x, \forall x > 0$, and $0 \geq f(x) \geq x, \forall x < 0$. The ReLU, henceforth denoted by ρ, is an example of such a function, while the sigmoid functions satisfy this property approximately.

We assume the input data to belong to a manifold K with Gaussian mean width

$$\omega(K) := E[\sup_{\mathbf{x},\mathbf{y} \in K} \langle \mathbf{g}, \mathbf{x} - \mathbf{y} \rangle], \quad (2)$$

where $\mathbf{g}$ is a random vector with normal i.i.d. elements. In Section IV we will illustrate this concept and exemplify the results with the Gaussian mixture models (GMM) and the standard sparsity model. For more details about $\omega(K)$ and its properties we refer the reader to [24], [36].

We now show that each standard DNN layer performs a stable embedding of the data in the Gromov-Hausdorff sense, i.e., it is a δ-isometry between (K, d_K) and $(K', d_{K'})$, where

K and K' are the manifolds of the input and output data and d_K and $d_{K'}$ are metrics induced on them. A function $h : K \to K'$ is a δ-isometry if

$$|d_{K'}(h(\mathbf{x}), h(\mathbf{y})) - d_K(\mathbf{x}, \mathbf{y})| \leq \delta, \forall \mathbf{x}, \mathbf{y} \in K, \tag{3}$$

and for every $\mathbf{x}' \in K'$ there exists $\mathbf{x} \in K$ such that $d_Y(\mathbf{x}', h(\mathbf{x})) \leq \delta$ (the latter property is sometimes called δ-surjectivity). In the following theorem and throughout the paper C denotes a given constant (not necessarily the same one) and $\mathbb{S}^{n-1}$ the unit sphere in $\mathbb{R}^n$.

Theorem 1: Let $\mathbf{M}$ be the linear operator applied at the i-th layer, f a semi-truncated linear activation function, and $K \subset \mathbb{S}^{n-1}$ the manifold of the input data for the i-th layer. If $\sqrt{m}\mathbf{M} \in \mathbb{R}^{m \times n}$ is a random matrix with i.i.d normally distributed entries, with $m \geq C\delta^{-6} w(K)^2$ being the output dimension, then with high probability $g(\cdot) = f(\mathbf{M}\cdot)$ is a δ-isometry.

The proof follows from the proof of Theorem 1.5 in [36], setting d_K to be the geodesic distance on $\mathbb{S}^{n-1}$ and $d_{K'}$ to be the Hamming distance in $\mathrm{sgn}(K')$, where the sign function is applied elementwise, and is defined as $\mathrm{sgn}(x) = 1$ if $x > 0$ and $\mathrm{sgn}(x) = -1$ if $x \leq 0$.

This result stands in line with [18] that suggested that each layer in the network creates a tessellation of the input data by the different hyper-planes imposed by the rows in $\mathbf{M}$. However, it implies more than that. By Corollary 1.9 in [36], each cell in the tessellation has a diameter of at most δ, i.e., if $\mathbf{x}$ and $\mathbf{y}$ fall to the same side of all the hyperplanes, then $\|\mathbf{x} - \mathbf{y}\|_2 \leq \delta$. In addition, the number of hyperplanes separating two points $\mathbf{x}$ and $\mathbf{y}$ in K contains enough information to deduce their distance.

Having a stable embedding of the data, it is natural to assume that it is possible to recover the input of a layer from its output. Indeed, Mahendran and Vedaldi demonstrate that it is achievable through the whole network [37]. The next result provides a theoretical justification for this, showing that it is possible to recover the input of each layer from its output:

Theorem 2: Under the assumptions of Theorem 1 there exists a program $\mathcal{A}$ such that $\|\mathbf{x} - \mathcal{A}(f(\mathbf{M}\mathbf{x}))\|_2 \leq \epsilon$, where $\epsilon = O\left(\frac{\omega(K)}{\sqrt{m}}\right)$.

The proof follows from Theorem 1.3 in [24]. If K is a cone then one may use also Theorem 2.1 in [38] to get a similar result.

III. Distance and Angle Distortion

So far we have focused on the metric preservation of the deep networks in terms of the Gromov-Hausdorff distance. In this section we turn to look at how the Euclidean distances and angles change within the layers. We focus on the case of ReLU as the activation function. A similar analysis can also be applied for pooling. For the simplicity of the discussion we defer it to a longer version of this paper.

Note that so far we assumed that K is basically normalized and lies on the sphere $\mathbb{S}^{n-1}$. Given that the data at the input of the network lie on the sphere and we use the ReLU ρ as the activation function, the transformation $\rho(\mathbf{M}\cdot)$ keeps the output data (approximately) on a sphere (with half the diameter, see (15) in the proof of Theorem 3). Therefore, in this case the normalization requirement holds up to a small distortion throughout the layers. This adds a motivation for having a normalization stage at the output of each layer, which was shown to provide some gain in several DNN [1], [39].

The normalization, which is also useful in shallow representations [40], can be interpreted as a transformation making the inner products between the data vectors coincide with the cosines of the corresponding angles. While the bounds we provide in this section do not require normalization, they show that the operations of each layer rely on the angles between the data points.

The following results relate the Euclidean and angular distances in the input of the i-th layer to the ones in its output. We denote by $\mathbb{B}_r^n \subset \mathbb{R}^n$ the Euclidean ball of radius r.

Theorem 3: Let $\mathbf{M}$ be the linear operator applied at the i-th layer, the activation function be ρ (the ReLU), and $K \subset \mathbb{B}_1^n$ the manifold of the data in the input of the i-th layer. If $\sqrt{m}\mathbf{M} \in \mathbb{R}^{m \times n}$ is a random matrix with i.i.d. normally distributed entries and $m \geq C\delta^{-4} w(K)^4$, then with high probability

$$\left| \|\rho(\mathbf{M}\mathbf{x}) - \rho(\mathbf{M}\mathbf{y})\|_2^2 - \left(\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + \frac{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}{\pi}\Big(\sin\angle(\mathbf{x}, \mathbf{y}) - \angle(\mathbf{x}, \mathbf{y}) \cos\angle(\mathbf{x}, \mathbf{y}) \Big) \right) \right| \leq \delta, \tag{4}$$

where $0 \leq \angle(\mathbf{x}, \mathbf{y}) \triangleq \cos^{-1}\left(\frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}\right) \leq \pi$.

Corollary 4: Under the same conditions of Theorem 3, with high probability

$$\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 - \delta \leq \|\rho(\mathbf{M}\mathbf{x}) - \rho(\mathbf{M}\mathbf{y})\|_2^2 \leq \|\mathbf{x} - \mathbf{y}\|_2^2 + \delta. \tag{5}$$

Theorem 5: Under the same conditions of Theorem 3 and $K \subset \mathbb{B}_1^n \setminus \mathbb{B}_\beta^n$, where $\delta \ll \beta^2 < 1$, with high probability

$$\left| \cos\angle\left(\rho(\mathbf{M}\mathbf{x}), \rho(\mathbf{M}\mathbf{y})\right) - \left(\cos\angle(\mathbf{x}, \mathbf{y}) + \frac{1}{\pi}\Big(\sin\angle(\mathbf{x}, \mathbf{y}) - \angle(\mathbf{x}, \mathbf{y}) \cos\angle(\mathbf{x}, \mathbf{y}) \Big) \right) \right| \leq \frac{8\delta}{\beta^2}. \tag{6}$$

Remark 1 *Similar to what we have seen in Section IV, the assumption $m \geq C\delta^{-4} w(K)^2$ implies $m = O(k^2)$ if K is a GMM and $m = O(k^2 \log L)$ if K is generated by k-sparse vectors and a dictionary $\mathbf{D} \in \mathbb{R}^{n \times L}$. By Theorem 6 it is enough to have the model assumption only at the data in the input of the DNN. Finally, the quadratic relationship between m and $w(K)^2$ might be improved.*

The proof of Corollary 4 follows from the inequality of arithmetic and geometric means and the behavior of (7) (see Fig. 1). We leave the proof of these theorems to Appendices A and B, and dwell on their implications. Note that if the term

$$\frac{1}{\pi}\Big(\sin\angle(\mathbf{x}, \mathbf{y}) - \angle(\mathbf{x}, \mathbf{y}) \cos\angle(\mathbf{x}, \mathbf{y}) \Big) \tag{7}$$

were equal to zero, then Theorems 3 and 5 would have stated that the distances and angles are preserved throughout the

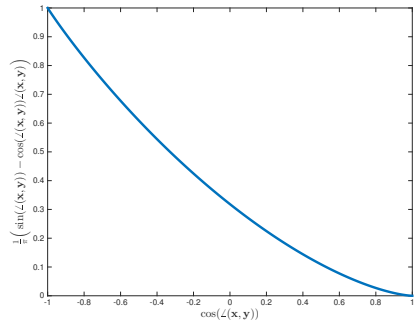

Fig. 1. Behavior of (7) as a function of $\cos\angle(\mathbf{x},\mathbf{y})$. $\cos\angle(\mathbf{x},\mathbf{y})=\pm 1$ correspond to the angles zero and π, respectively, between $\mathbf{x}$ and y . Notice that the larger is the angle between the two, the larger is the value of the term. In addition, it vanishes for small angles between the two vectors.

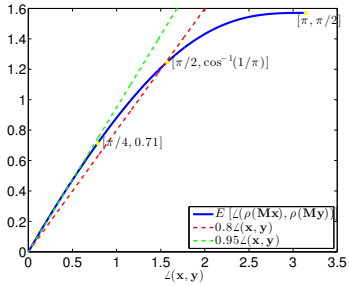

Fig. 2. Behavior of the angle between two points $\mathbf{x}$ and $\mathbf{y}$ in the output of a DNN layer as a function of their angle in the input. In the ranges $[0,\pi/4]$ and $[\pi/4,\pi/2]$ the output behaves approximately like $0.95\angle(\mathbf{x},\mathbf{y})$ and $0.21+\frac{2}{3}\angle(\mathbf{x},\mathbf{y})$, respectively.

network.[1] As can be seen in Fig. 1, the term (7) behaves approximately like $0.5(1-\cos\angle(\mathbf{x},\mathbf{y}))$. The larger is the angle between the two, the larger is the value of the term. For highly correlated $\mathbf{x}$ and $\mathbf{y}$ (i.e., $\cos\angle(\mathbf{x},\mathbf{y})$ close to 1), this term vanishes and therefore their Euclidean and angular distances are preserved throughout the layers of the network.

Considering this effect on the Euclidean distance, the larger is the angle between $\mathbf{x}$ and $\mathbf{y}$, the larger is the distortion to this distance at the output of the layer, and the larger it turns to be. We conclude that DNN preserve local structures in the manifold K and increase the distances between points away from it, a property much desired for classification.

The influence of the entire network on the angles is slightly different. Note that starting from the input of the second layer, all the vectors reside in the non-negative orthant. The cosine of the angles is translated from the range $[-1,1]$ in the first layer to the range $[0,1]$ in the subsequent second layers. In particular, the range $[-1,0]$ is translated to the range $[0,1/\pi]$, and in terms of the angle $\angle(\mathbf{x},\mathbf{y})$ from the range $[\pi/2,\pi]$ to $[\cos^{-1}(1/\pi),\pi/2]$. These angles shrink approximately by half, while the ones that are initially small remain approximately unchanged.

The action of the network preserves the order between the angles. Generally speaking, the network affects the angles in the range $[0,\pi/2]$ in the same way. In particular, in the range $[0,\pi/4]$ the angles in the output of the layer behave like $0.95\angle(\mathbf{x},\mathbf{y})$ and in the wider range $[0,\pi/2]$ they are bounded from below by $0.8\angle(\mathbf{x},\mathbf{y})$ (see Figure 2). Therefore, we conclude that the DNN distort the angles in the input manifold K similarly and keep the general constellation of angles between the points.

To see that our theory captures the behavior of the DNN, also with the presence of pooling, we test how the angles change through the state-of-the-art 19-layers deep network trained in [29] for the ImagNet dataset. We randomly select 30000 angles (pairs of data points) in the input of the network, partitioned to three equally-sized groups, each corresponds to a one of the ranges $[0,\pi/4]$, $[\pi/4,\pi/2]$ and $[\pi/2,\pi]$. We test their behavior after applying eight and sixteen non-linear layers. The latter case corresponds to the part of the network excluding the fully connected layers. We denote by $\bar{\mathbf{x}}$ the vector in the output of a layer corresponding to the input vector $\mathbf{x}$. Fig. 3 presents a histogram of the values of the angles $\angle(\bar{\mathbf{x}},\bar{\mathbf{y}})$ at the output of each of the layers for each of the three groups. It shows also the ratio $\angle(\bar{\mathbf{x}},\bar{\mathbf{y}})/\angle(\mathbf{x},\mathbf{y})$, between the angles at the output of the layers and their original value at the input of the network.

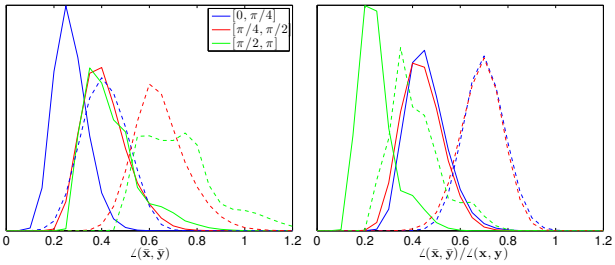

Fig. 3. Left: Histogram of the angles in the output of the 8-th (dashed-lines) and 16-th (continuous-lines) layers of the ImageNet deep network for different input angle ranges. Right: Histogram of the ratio between the angles in the output of the 8-th (dashed-line) and 16-th (continuous-line) layers of this network and the angles in its input for different input angles ranges.

As our theorem predicts, the ratio for the range $[\pi/2,\pi]$ is half the ratio in the range $[0,\pi/2]$. Furthermore, the ratios in the ranges $[0,\pi/4]$ and $[\pi/4,\pi/2]$ are approximately the same, where in the range $[\pi/4,\pi/2]$ they are slightly larger. This is in line with Theorem 6 that claim that the angles in this range decrease approximately in the same rate, where for larger angles the shrink is slightly larger. Note also that according to our theory the ratio in the range $[0,\pi/4]$ should behave on average like 0.95^i, where i is the number of layers. Indeed, for $i=8$, $0.95^8=0.66$ and for $i=16$, $0.95^{16}=0.44$ and the centers of the histograms for the range $[0,\pi/4]$ is very close to these values. Notice that we have a similar behavior also for the range $[\pi,\pi/2]$. This is not surprising, as by looking at Fig. 2 one may observe that these angles also turn to be in the range that has the ratio 0.95. Remarkably, Fig. 3 demonstrates that the network keeps the order between the angles as Theorem 5 suggests. Small angles in the input stay small in the output.

Ideally, we would like points belonging to the same class to stay closer to each other in the output of the network, compared to points from different classes. However, random

[1]More specifically, we would have $\left\|\sqrt{2}\rho(\mathbf{Mx})-\sqrt{2}\rho(\mathbf{My})\right\|_2^2 \cong \|\mathbf{x}-\mathbf{y}\|_2^2$. Notice that this is the outcome we would expect for the distance $\|\rho(\mathbf{Mx}-\mathbf{My})\|_2^2$.

networks are not selective in this sense: if a point $\mathbf{x}$ forms the same angle with a point $\mathbf{z}$ from its class and as that with a point $\mathbf{y}$ from another class, then their angle will be distorted approximately by an equal amount.

IV. Stable Embedding of the Network

In order to show that the results in Sections II and III apply also to *entire* network and not only to one layer; we need to show that the Gaussian mean width does not grow significantly as the data propagate through the layers of the network Instead of bounding the variation of the Gaussian mean width throughout the network, we bound the change in the covering number $N_\epsilon(K)$, i.e., the lowest number of ℓ_2-balls of radius ϵ that cover K. Having the bound on the covering number, we use Dudley's inequality [41], $\omega(K) \leq C\int_0^\infty \sqrt{\log N_\epsilon(K)}d\epsilon$, to bound the Gaussian mean width.

Theorem 6: Under the assumptions of Theorem 1,

$$N_\epsilon(f(\mathbf{M}K)) \leq N_{\frac{\epsilon}{1+\frac{\omega(K)}{\sqrt{m}}}}(K). \tag{8}$$

Proof: It is not hard to see that since a semi-truncated linear activation function shrinks the data, it can not increase the size of the covering; therefore we focus on the linear part. Following [42, Theorem 1.4], the distances in $\mathbf{M}K$ are the same as the ones in K up to a $1+\frac{\omega(K)}{\sqrt{m}}$ factor. This is sufficient to complete the proof. □

We now demonstrate the implication of the above theorem for two popular data models: GMM, assuming K consisting of L Gaussians of dimension k in the ℓ_2-ball; and sparsely representable signals, assuming that the data can be approximated by a sparse linear combination of atoms of a dictionary. Similar results can be shown for other models such as union of subspaces and low dimensional manifolds.

a) Gaussian mixture models -: For GMM, the covering number is $N_\epsilon(K) = L\left(1+\frac{2}{\epsilon}\right)^k$ for $\epsilon < 1$ and 1 otherwise (see [23]). Therefore we have that $\omega(K) \leq C\sqrt{k+\log L}$ and that at each layer the Gaussian mean width grows at most with an order of $1+\frac{\sqrt{k}+\log L}{\sqrt{m}}$.

b) Sparsely representable signals -: In the sparsity case $K = \{\mathbf{x} = \mathbf{D}\boldsymbol{\alpha} : \|\boldsymbol{\alpha}\|_0 \leq k,\}$, where $\|\cdot\|_0$ is the pseudo-norm that counts the number of non-zeros in a vector and $\mathbf{D} \in \mathbb{R}^{n\times L}$ is a given dictionary. For this model, $N_\epsilon(K) = \binom{L}{k}\left(1+\frac{2}{\epsilon}\right)^k$. By Stirling's approximation we have $\binom{L}{k} \leq \left(\frac{eL}{k}\right)^k$ and therefore $\omega(K) \leq C\sqrt{k\log(L/k)}$. Thus, at each layer the Gaussian mean width grows at most by an order of $1+\frac{\sqrt{k\log(L/k)}}{\sqrt{m}}$.

Remark 7: Theorem 6 generalizes the results in Theorems 2, 3 and 5 to the whole network, but not the one of Theorem 1. In order to do that for the later, we need also a version of Theorem 1 that guarantees a stable embedding from the Hamming cube to the Hamming cube. However, we did not add that as we are more interested in the fact that it is possible to recover the input of the network from its output, i.e., the generalization of Theorem 2. Moreover, as Corollary 4 implies stability in the Lipschitz sense with constants 0.5 and 1 for each layer, we do have stability for the whole network in the Lipschitz sense, which is even stronger than stability in the Gromov-Hausdorff sense that we would get by having the generalization of Theorem 1.

V. Training Set Size

An important question in deep learning is what is the amount of labeled samples needed at training. Using Sudakov minoration [41], one can get an upper bound on the size of an ϵ-net in K. We have demonstrated that networks with random Gaussian weights realize a stable embedding; consequently, if a network is trained using the screening technique by selecting the best among many networks generated with random weights as suggested in [4], [5], [6], then the number of data points needed in order to guarantee that the network represents all the data is $O(\exp(\omega(K)^2/\epsilon^2))$. Since $\omega(K)^2$ is a proxy for the data dimension as we have seen in the previous section (see [24] for more details), this bound formally predicts that the number of training points grows exponentially with the intrinsic dimension of the data.

The exponential dependency is too restrictive, and it is often possible to achieve a better bound on the needed number of measurements. Indeed, [11] require much less samples. As the authors study the recovery of an autoencoder, they assume that there exists a 'ground-truth autoencoder' generating the data. A combination of the data dimension bound here provided, with a prior on the relation of the data to a deep network, should lead to a better prediction of the number of needed samples. The use of a combination of the properties of the system with those of the input data led to works in the sparsity arena providing bounds that are better than just using the manifold covering number (see [43] and references therein).

The following section presents such a combination. It empirically shows that a purpose of training in DNN is to treat boundary points. This observation is likely to lead to a significant reduction in the required size of the training data and to perform active learning.

VI. The Role of Training

The proof of the theorems provides us with an insight on the role of training. One key property of the Gaussian distribution, which allows it to keep the ratio between the angles in the data, is its rotational invariance. The phase of a random Gaussian vector with i.i.d. entries is a random vector with a uniform distribution. Therefore, it does not prioritize one direction in the manifold over the other but treats all the same, leading to the behavior of the angles and distances throughout the net that we have described above.

In general, points within the same class would have small angles within them and points from distinct classes would have larger ones. If this was the case for all the points, then Gaussian weights would have been the ideal thing to use. However, as this is not the case, the role of training would be to select in a smart way the separating hyper-planes induced by $\mathbf{M}$ in such a way that the angles between points from different classes are 'penalized more' than the angles between the points in the same class. This also explains why random initialization is a good choice for random network. As it is hard to find the optimal constellation that separates the classes on the manifold, it is desirable to start with a universal one that treats most of the angles and distances well, and then

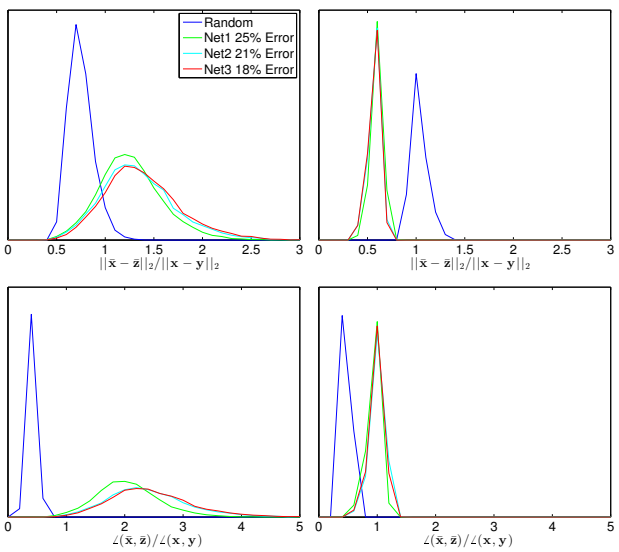

Fig. 4. Ratios of closest inter (left) and farthest intra (right) class Euclidean (top) and angular (bottom) distances for CIFAR-10.

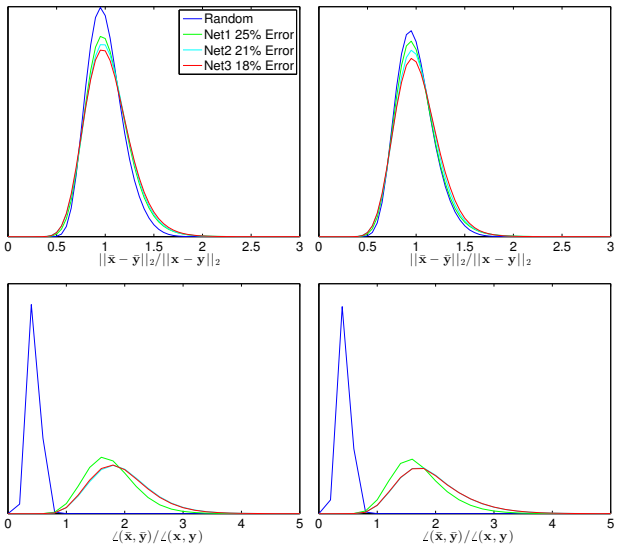

Fig. 5. Ratios of inter (left) and intra (right) class Euclidean (top) and angular (bottom) distances for CIFAR-10.

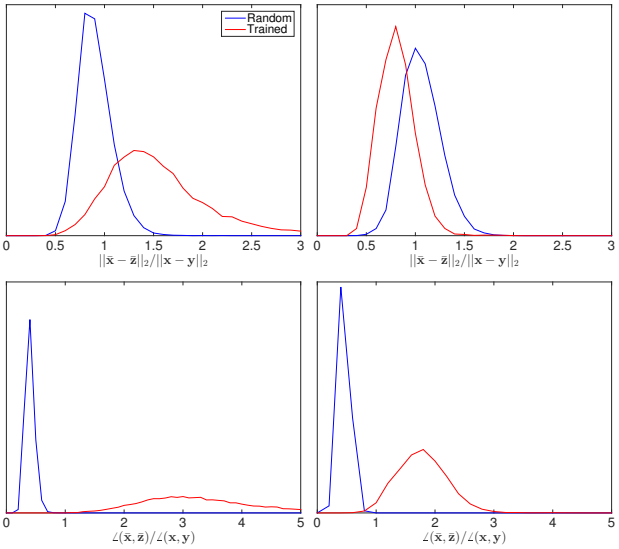

Fig. 6. Ratios of closest inter (left) and farthest intra (right) class Euclidean (top) and angular (bottom) distances for ImagNet.

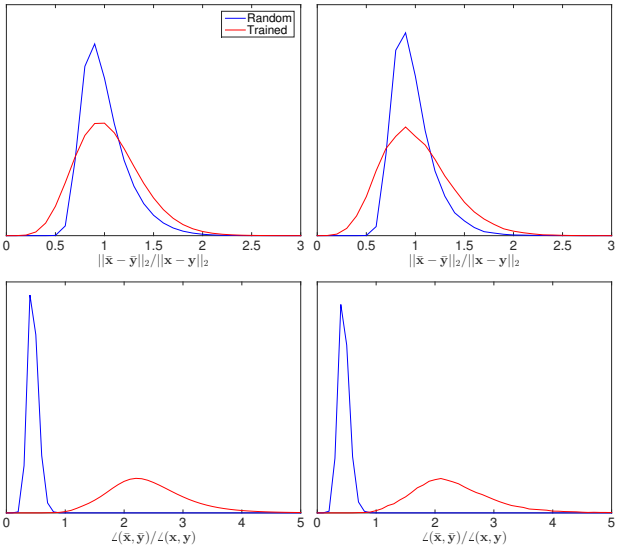

Fig. 7. Ratios of inter (left) and intra (right) class Euclidean (top) and angular (bottom) distances for ImagNet.

just to adjust it in the locations that create ambiguity in the classification.

To test this behavior, we trained two DNN on the MNIST and CIFAR-10 datasets, each containing 10 classes. The training of the networks was done using the *matcovnet* toolbox [44]. The MNIST and CIFAR-10 networks were trained with four and five layers, respectively, followed by a softmax operation. We used the default settings provided by the toolbox for each dataset, where with CIFAR we also used horizontal mirroring and 64 filters in the first two layers instead of 32 (which is the default in the example provided with the package) to improve performance. The trained DNN achieve 1% and 18% errors for MNIST and CIFAR-10 respectively.

For each data point we calculate its Euclidean and angular distances to the farthest point from its class and to the closest point not in its class. We do this both at the input of the DNN and at the output of the last convolutional layers (the input of the fully connected layers). Then we compute the ratio between the two, i.e., if $\mathbf{x}$ is the point at input, $\mathbf{y}$ is its farthest point in class, $\bar{\mathbf{x}}$ is the point at the output, and $\bar{\mathbf{z}}$ is its farthest point from the same class (it should not necessarily be the output of $\mathbf{y}$), then we calculate $\frac{\|\bar{\mathbf{x}}-\bar{\mathbf{z}}\|_2^2}{\|\mathbf{x}-\mathbf{y}\|_2^2}$ for Euclidean distances and $\cos\angle(\bar{\mathbf{x}},\bar{\mathbf{z}})/\cos\angle(\mathbf{x},\mathbf{y})$ for the angles. We do the same for the distances between different classes, comparing the shortest ones. Fig. 4 presents histogram of these distance ratios for CIFAR-10. We also compare the behavior of all the inter and intra-class distances by computing the above ratios for all pairs of points $(\mathbf{x},\mathbf{y})$ in the input with respect to their corresponding points $(\bar{\mathbf{x}},\bar{\mathbf{y}})$ at the output. These ratios are presented in Fig. 5. Different amounts of training are shown in the figures and referred to as Net1, Net2 and Net3.

Considering the Euclidean distance ratios, the histograms over all data point pairs change only slightly due to training. Remark that all histograms are centered around 1, implying that the network preserves most of the distances as our theorems predict for a network with random weights. Observe that

the trained networks behave like the random one in keeping the distance of a randomly picked pair of points. However, they distort the distances between points on class boundaries better than the random network: The farthest intra class distances are shrunk with a larger factor than the ones of the random network, and the closest inter class distances are set farther apart by the training. Notice that the shrinking of the distances within the class and enlargement of the distances between the classes improves as the training proceeds. This confirms our prediction above that a goal of training is to treat the boundary points.

A similar behavior can be observed for the angles. The closest angles are enlarged more in the trained network compared to the random one. However, enlarging the angles between classes also causes the enlargement of the angles within the classes. Notice though that these are enlarged less than the ones which are outside the class. Finally, observe that the enlargement of the angles, as we have seen in our theorems, causes a larger distortion in the Euclidean distances. Therefore, we may explain the enlargement of the distances in within the class as a means for shrinking the intra-class distances.

Similar behavior is observed for the MNIST dataset. However, the gaps between the random network and the trained network are smaller as the MNIST dataset contains data which are initially well separated. As we have argued above, for such manifolds the random network is already a good choice.

We also compared the behavior of the validation data, of the ImagNet dataset, in the network provided by [29] and in the same network but with random weights. The results are presented in Figs. 6, and 7. Behavior similar to the one we observed in the case of CIFAR-10, is also manifested by the ImageNet network.

VII. Discussion and Conclusion

We have shown that DNN with random Gaussian weights perform a stable embedding of the data, drawing a connection between the dimension of the features produced by the network, which still keep the metric information of the original manifold, and the complexity of the data. In addition, our result provides a formal relationship between the complexity of the input data and the size of the required training set. Moreover, we proved that the Euclidean distances of the input data are distorted throughout the networks based on the angles between the data points. Our results lead to the conclusion that DNN are universal classifiers for data based on the principal angles between the classes in the data.

Our work implies that it is possible to view DNN as a stagewise metric learning process, suggesting that it might be possible to replace the currently used layers with other metric learning algorithms, opening a new venue for semi-supervised DNN. This also stands in line with the recent literature on convolutional kernel methods (see [45], [46]).

In addition, we 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. This may lead to a new active learning strategy for deep learning [47].

Acknowledgments- Work partially supported by NSF, ONR, NGA, NSSEFF, and ARO. A.B. is supported by ERC StG 335491 (RAPID)

Appendix A
Proof of Theorem 3

Before we turn to prove our Theorem, we present a proposition that will aid us in its proof.

Proposition 8: Let $\mathbf{g} \in \mathbb{R}^n$ be a random vector with i.i.d. normally distributed entries, and $K \subset \mathbb{B}_1^n$ be a set with a Gaussian mean width $w(K)$. Then with probability exceeding $1 - 2\exp(-\alpha^2/2)$

$$\sup_{\mathbf{x},\mathbf{y}\in K} \left(\rho(\mathbf{g}^T\mathbf{x}) - \rho(\mathbf{g}^T\mathbf{y})\right)^2 < (2w(K) + \alpha)^2. \tag{9}$$

Proof: From the triangle inequality we have $\left|\rho(\mathbf{g}^T\mathbf{x}) - \rho(\mathbf{g}^T\mathbf{y})\right| \leq \left|\mathbf{g}^T\mathbf{x}\right| + \left|\rho(\mathbf{g}^T\mathbf{y})^2\right|$. From Proposition 2.1 in [24] we have that $E\sup_{\mathbf{x}\in K}\left|\mathbf{g}^T\mathbf{x}\right| \leq w(K)$. Using the Gaussian concentration bound [41, Equation 1.6], with the fact that $\left|\mathbf{g}^T\mathbf{x}\right|$ is Lipschitz-continuous with the constant 1 (since $K \subset \mathbb{B}_1^n$), leads to (9). □

Proof of Theorem 3: Our proof of Theorem 3 consists of three keys steps. In the first one, we show that the bound in (4) holds with high probability for any two points $\mathbf{x}, \mathbf{y} \in K$. In the second, we pick an ϵ-cover for K and show that the same holds for each pair in the cover. The last generalizes the bound for any point in K.

Bound for a pair $\mathbf{x}, \mathbf{y} \in K$: Denoting by $\mathbf{m}_i$ the i-th column of $\mathbf{M}$ we rewrite $\|\rho(\mathbf{M}\mathbf{x}) - \rho(\mathbf{M}\mathbf{y})\|_2^2 = \sum_i^m \left(\rho(\mathbf{m}_i^T\mathbf{x}) - \rho(\mathbf{m}_i^T\mathbf{y})\right)^2$. Let

$$\mathbf{z}_i \triangleq \left(\rho(\mathbf{m}_i^T\mathbf{x}) - \rho(\mathbf{m}_i^T\mathbf{y})\right)^2 - \frac{1}{m}\left(\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \frac{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}{\pi}\Big(\sin(\angle(\mathbf{x},\mathbf{y})) - \cos(\angle(\mathbf{x},\mathbf{y}))\angle(\mathbf{x},\mathbf{y})\Big)\right). \tag{10}$$

Then, by using symmetry and the rotational invariance property of $\mathbf{m}_i$ that leads to an integral similar to the one in (12), we have that $E\mathbf{z}_i = 0$. To prove the bound, it remains to show that the sum $\sum_{i=1}^m \mathbf{z}_i$ does not deviate much from its mean. By the symmetry of the Gaussian distribution and Bernstein inequality we have

$$P\left(\sum_{i=1}^m |\mathbf{z}_i| > t\right) \leq 2\exp\left(-\frac{t^2}{\sum_{i=1}^m E\mathbf{z}_i^2 + M}\right) \tag{11}$$

where M is an upper bound for $|\mathbf{z}_i|$. To calculate $E\mathbf{z}_i^2$, one needs to calculate the fourth moments $E(\rho(\mathbf{M}\mathbf{x}))^4$ and $E(\rho(\mathbf{M}\mathbf{y}))^4$, which is easy to compute by using the symmetry of Gaussian vectors, and the correlations $E\rho(\mathbf{M}\mathbf{x})^3\rho(\mathbf{M}\mathbf{y})$, $E\rho(\mathbf{M}\mathbf{x})^2\rho(\mathbf{M}\mathbf{y})^2$ and $E\rho(\mathbf{M}\mathbf{x})^2\rho(\mathbf{M}\mathbf{y})^2$. For calculating the later, we use the fact that a Gaussian vector is uniformly distributed on the sphere and calculate an integral on an interval which is dependent on the angle between $\mathbf{x}$ and $\mathbf{y}$. For example

$$E\rho(\mathbf{M}\mathbf{x})^3\rho(\mathbf{M}\mathbf{y}) = \frac{4\|\mathbf{x}\|_2^3\|y\|_2}{m^2\pi} \cdot \int_0^{\pi-\angle(\mathbf{x},\mathbf{y})} \sin^3(\theta)\sin(\theta+\angle(\mathbf{x},\mathbf{y}))\theta, \tag{12}$$

where θ is the angle between $\mathbf{m}_i$ and $\mathbf{x}$. We have a similar formula for the other terms. This type of a formula, a variant of which is also used to show that $E\mathbf{z}_i = 0$, provides an insight into the role of training. As random layers 'integrate uniformly' on the interval $[0, \pi - \angle(\mathbf{x}, \mathbf{y})]$, learning picks the angle θ that maximizes/minimizes the inner product based on whether $\mathbf{x}$ and $\mathbf{y}$ belong to the same class or to distinct classes.

By simple arithmetics and using the fact that $K \in \mathbb{B}_1^n$, we have that $E\mathbf{z}_i^2 \leq 2.1/m$. Plugging this and (9) into the Bernstein inequality, and using the fact that $K \subset B_1^n$, leads to

$$P\left(\sum_{i=1}^{m} |\mathbf{z}_i| > \frac{\delta}{2}\right) \leq C \exp\left(-\frac{m\delta^2}{4w(K)^2}\right), \tag{13}$$

where we set $\alpha = w(K)$ and merge the probability of Proposition 8 in the above bound.

Bound for all $\mathbf{x}, \mathbf{y} \in N_\epsilon(K)$: By using a union bound we have that (13) holds for every pair in $N_\epsilon(K)$ with probability $C\left|N_\epsilon(K)\right|^2 \exp\left(-\frac{mt^2}{4w(K)^2}\right)$. Let $\epsilon \leq \delta$. Then by Sudakov's inequality we have $\log|N_\epsilon(K)| \leq C\epsilon^{-2}w(K)^2 \leq C\epsilon^{-4}w(K)^2$. Then by the assumptions of the Theorem we have (13) holds for $N_\epsilon(K)$ with probability $C\exp(-\sqrt{m})$.

Bound for all K: Let $\tilde{\mathbf{x}}, \tilde{\mathbf{y}} \in K$ then we can rewrite them as $\mathbf{x}_0 + \mathbf{x}_\epsilon$ and $\mathbf{y}_0 + \mathbf{y}_\epsilon$, where $\mathbf{x}_0, \mathbf{y}_0 \in N_\epsilon(K)$ and $\mathbf{x}_\epsilon, \mathbf{y}_\epsilon \in (K-K) \cap \mathbb{B}_\epsilon^n$, where $K - K = \{\mathbf{x} - \mathbf{y} : \mathbf{x}, \mathbf{y} \in K\}$. . By similar steps to the one used above, together with Proposition 8 to control $\rho(\mathbf{M}\mathbf{x}_\epsilon)$ and $\rho(\mathbf{M}\mathbf{y}_\epsilon)$, and using the fact that $w(K-K) \leq 2w(K)$ leads to the desired result by picking $\epsilon < \frac{1}{10}\delta$. □

APPENDIX B
PROOF OF THEOREM 5

Proof: Instead of proving Theorem 5 directly, we deduce it from Theorem 3. First we notice that (4) is equivalent to

$$\Bigg|\frac{1}{2}\left\|\rho(\mathbf{M}\mathbf{x})\right\|_2^2 + \frac{1}{2}\left\|\rho(\mathbf{M}\mathbf{y})\right\|_2^2 - \frac{1}{4}\left\|\mathbf{x}\right\|_2^2 - \frac{1}{4}\left\|\mathbf{y}\right\|_2^2 \tag{14}$$
$$-\Big(\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y}) - \frac{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2}{2}\cos(\angle(\mathbf{x},\mathbf{y})) -$$
$$\frac{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2}{2\pi}\big(\sin(\angle(\mathbf{x},\mathbf{y})) - \cos(\angle(\mathbf{x},\mathbf{y}))\angle(\mathbf{x},\mathbf{y})\big)\Big)\Bigg| \leq \frac{\delta}{2}.$$

As $\mathbf{0} \in \mathbb{B}_1^n$, Theorem 3 implies that

$$\left|\left\|\rho(\mathbf{M}\mathbf{x})\right\|_2^2 - \frac{1}{2}\left\|\mathbf{x}\right\|_2^2\right| \leq \delta, \quad \forall \mathbf{x} \in K. \tag{15}$$

Using the reverse triangle inequality on (14) together with (15), followed by dividing both sides by $\frac{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2}{2}$, leads to

$$\Bigg|\Big(\frac{2\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y})}{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2} - \cos(\angle(\mathbf{x},\mathbf{y})) - \tag{16}$$
$$\frac{1}{\pi}\big(\sin(\angle(\mathbf{x},\mathbf{y})) - \cos(\angle(\mathbf{x},\mathbf{y}))\angle(\mathbf{x},\mathbf{y})\big)\Big)\Bigg| \leq \frac{3\delta}{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2}.$$

To end the proof we turn to bound

$$\left|\frac{2\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y})}{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2} - \frac{\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y})}{\left\|\rho(\mathbf{M}\mathbf{x})\right\|_2\left\|\rho(\mathbf{M}\mathbf{y})\right\|_2}\right|. \tag{17}$$

Dividing both sides of (15) by $\left\|\mathbf{x}\right\|_2^2/2$ leads to

$$\frac{\sqrt{2}\left\|\rho(\mathbf{M}\mathbf{x})\right\|_2}{\left\|\mathbf{x}\right\|_2} \leq \sqrt{1 + \frac{2\delta}{\beta^2}}, \forall \mathbf{x} \in K, \tag{18}$$

as $K \subset \mathbb{B}1^n \setminus \beta\mathbb{B}_2^n$. Noticing that (17) equals to $\left|\frac{2\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y})}{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2} - \frac{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2}{2\left\|\rho(\mathbf{M}\mathbf{x})\right\|_2\left\|\rho(\mathbf{M}\mathbf{y})\right\|_2}\frac{2\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y})}{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2}\right|$ and then using the triangle inequality with (18), bounds (17) by $\frac{4\delta}{\beta^2}\frac{\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y})}{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2}$. Since the term $\cos(\angle(\mathbf{x},\mathbf{y})) + \frac{1}{\pi}\big(\sin(\angle(\mathbf{x},\mathbf{y})) - \cos(\angle(\mathbf{x},\mathbf{y}))\angle(\mathbf{x},\mathbf{y})$ is bounded by 1, (16) implies that $\frac{4\delta}{\beta^2}\frac{\rho(\mathbf{M}\mathbf{x})^T\rho(\mathbf{M}\mathbf{y})}{\left\|\mathbf{x}\right\|_2\left\|\mathbf{y}\right\|_2} \leq \frac{4\delta}{\beta^2}\left(1 + \frac{\delta}{\beta^2}\right)$. Using this bound for (17) together with applying the triangle inequality on (16), bounds (6) with $\tilde{\delta} = \frac{7\delta}{\beta^2} + \frac{\delta^2}{\beta^4}$. Since $\delta < \beta^2$ we end up having $\tilde{d} \leq \frac{8\delta}{\beta^2}$. □

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[3] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[4] N. Pinto, D. Doukhan, J. J. DiCarlo, and D. D. Cox, "A high-throughput screening approach to discovering good forms of biologically inspired visual representation," *PLoS Comput Biol*, vol. 5, no. 11, p. e1000579, 11 2009.

[5] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning," in *Int. Conf. on Machine Learning (ICML)*, 2011, pp. 1089–1096.

[6] D. Cox and N. Pinto, "Beyond simple features: A large-scale feature search approach to unconstrained face recognition," in *IEEE International Conference on Automatic Face Gesture Recognition and Workshops (FG)*, March 2011, pp. 8–15.

[7] E. J. Candès and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5406 –5425, Dec. 2006.

[8] M. Elad, "Optimized projections for compressed sensing," *IEEE Trans. Signal Process.*, vol. 55, no. 12, pp. 5695–5702, Dec 2007.

[9] J. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization," *IEEE Trans. Imag. Proc.*, vol. 18, no. 7, pp. 1395–1408, July 2009.

[10] C. Hegde, A. C. Sankaranarayanan, W. Yin, and R. G. Baraniuk, "Numax: A convex approach for learning near-isometric linear embeddings," *to appear in IEEE Transactions on Signal Processing*, 2015.

[11] S. Arora, A. Bhaskara, R. Ge, and T. Ma, "Provable bounds for learning some deep representations," in *Int. Conf. on Machine Learning (ICML)*, 2014, pp. 584–592.

[12] A. Saxe, J. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural network," in *International Conference on Learning Representations (ICLR)*, 2014.

[13] A. Choromanska, M. B. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

[14] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[15] E. J. Candès, T. Strohmer, and V. Voroninski, "Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming," *Communications on Pure and Applied Mathematics*, vol. 66, no. 8, pp. 1241–1274, 2013.

[16] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989.

[17] G. F. Montúfar and J. Morton, "When does a mixture of products contain a product of mixtures?" *to appear in SIAM Journal on Discrete Mathematics (SIDMA)*, 2014.

[18] G. F. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[19] J. Bruna, Y. LeCun, and A. Szlam, "Learning stable group invariant representations with convolutional networks," in *ICLR Workshop*, Jan. 2013.

[20] J. Bruna, A. Szlam, and Y. LeCun, "Signal recovery from l_p pooling representations," in *Int. Conf. on Machine Learning (ICML)*, 2014.

[21] M. Rudelson and R. Vershynin, "Non-asymptotic theory of random matrices: extreme singular values," in *International Congress of Mathematicans*, 2010.

[22] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," in *Conf. in Modern Analysis and Probability*, 1984, p. 189206.

[23] S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann, "Uniform uncertainty principle for Bernoulli and sub-Gaussian ensembles," *Constructive Approximation*, vol. 28, pp. 277–289, 2008.

[24] Y. Plan and R. Vershynin, "Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 482–494, Jan. 2013.

[25] O. Yamaguchi, K. Fukui, and K. Maeda, "Face recognition using temporal image sequence," in *IEEE Int. Conf. Automatic Face and Gesture Recognition*, 1998, pp. 318–323.

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

[27] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, 2013.

[28] Q. Qiu and G. Sapiro, "Learning transformations for clustering and classification," *Journal of Machine Learning Research*, vol. 16, p. 187225, Feb. 2015.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2014.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Int. Conf. on Machine Learning (ICML)*, 2010, pp. 807–814.

[32] J. Haupt, W. Bajwa, G. Raz, and R. Nowak, "Toeplitz compressed sensing matrices with applications to sparse channel estimation," *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 5862–5875, Nov. 2010.

[33] V. Saligrama, "Aperiodic sequences with uniformly decaying correlations with applications to compressed sensing and system identification," *IEEE Trans. Inf. Theory*, vol. 58, no. 9, pp. 6023–6036, Sept. 2012.

[34] H. Rauhut, J. Romberg, and J. A. Tropp, "Restricted isometries for partial random circulant matrices," *Appl. Comput. Harmon. Anal.*, vol. 32, no. 2, pp. 242–254, Mar. 2012.

[35] A. Ai, A. Lapanowski, Y. Plan, and R. Vershynin, "One-bit compressed sensing with non-gaussian measurements," *to appear in Linear Algebra and its Applications*, 2014.

[36] Y. Plan and R. Vershynin, "Dimension reduction by random hyperplane tessellations," *Discrete and Computational Geometry*, vol. 51, no. 2, pp. 438–461, 2014.

[37] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," *ArXiv*, 2014. [Online]. Available: http://arxiv.org/abs/1412.0035

[38] Y. Plan, R. Vershynin, and E. Yudovina, "High-dimensional estimation with geometric constraints," *Arxiv*, 2014. [Online]. Available: http://arxiv.org/abs/1404.3749

[39] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conference*, 2014.

[40] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*, 2010, vol. 6314, pp. 143–156.

[41] M. Ledoux and M. Talagrand, *Probability in Banach Spaces*. Springer-Verlag, 1991.

[42] B. Klartag and S. Mendelson, "Empirical processes and random projections," *Journal of Functional Analysis*, vol. 225, no. 1, pp. 229–245, Aug. 2005.

[43] R. Gribonval, R. Jenatton, and F. Bach, "Sparse and spurious: dictionary learning with noise and outliers," Sep. 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01025503

[44] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," *CoRR*, vol. abs/1412.4564, 2014.

[45] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[46] Z. Lu, A. May, K. Liu, A. B. Garakani, D. Guo, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, and F. Sha, "How to scale up kernel methods to be as good as deep neural nets," *Arxiv*, 2014. [Online]. Available: http://arxiv.org/abs/1411.4000

[47] E. Elhamifar, G. Sapiro, A. Yang, and S. Sasrty, "A convex optimization framework for active learning," in *IEEE International Conference on Computer Vision (ICCV)*, Dec 2013, pp. 209–216.