Orthogonal Deep Neural Networks

Kui Jia*, Shuai Li*, Yuxin Wen, Tongliang Liu, and Dacheng Tao

Abstract—In this paper, we introduce the algorithms of Orthogonal Deep Neural Networks (OrthDNNs) to connect with recent interest of spectrally regularized deep learning methods. OrthDNNs are theoretically motivated by generalization analysis of modern DNNs, with the aim to find solution properties of network weights that guarantee better generalization. To this end, we first prove that DNNs are of local isometry on data distributions of practical interest; by using a new covering of the sample space and introducing the local isometry property of DNNs into generalization analysis, we establish a new generalization error bound that is both scale- and range-sensitive to singular value spectrum of each of networks' weight matrices. We prove that the optimal bound w.r.t. the degree of isometry is attained when each weight matrix has a spectrum of equal singular values, among which orthogonal weight matrix or a non-square one with orthonormal rows or columns is the most straightforward choice, suggesting the algorithms of OrthDNNs. We present both algorithms of strict and approximate OrthDNNs, and for the later ones we propose a simple yet effective algorithm called Singular Value Bounding (SVB), which performs as well as strict OrthDNNs, but at a much lower computational cost. We also propose Bounded Batch Normalization (BBN) to make compatible use of batch normalization with OrthDNNs. We conduct extensive comparative studies by using modern architectures on benchmark image classification. Experiments show the efficacy of OrthDNNs.

Index Terms—Deep neural networks, generalization error, robustness, spectral regularization, image classification

1 Introduction

Deep learning or deep neural networks (DNNs) have been achieving great success on many machine learning tasks, with image classification [47] as one of the prominent examples. Key design that supports success of deep learning can date at least back to Neocognitron [17] and Convolutional neural networks (CNNs) [38], which employ hierarchial, compositional design to facilitate learning target functions that approximately capture statistical properties of natural signals. Modern DNNs are usually over-parameterized and have very high model capacities, yet practically meaningful solutions can be obtained via simple back-propagation training of stochastic gradient descent (SGD) [39], where regularization methods such as early stopping, weight decay, and data augmentation are commonly used to alleviate the issue of overfitting.

Over the years, new technical innovations have been introduced to improve DNNs in terms of architectural design [22], [27], optimization [14], [19], [34], and also regularization [25], [29], which altogether make efficient and effective training of extremely over-parameterized models possible. While many of these innovations are empirically proposed, some of them are justified by subsequent theoretical studies that explain their practical effectiveness. For example, dropout training [25] is explained as an approximate regularization of adaptive weight decay in [3], [55]. Theoretically characterizing global optimality conditions of DNNs are also presented in [31], [61].

The above optimization and regularization methods aim to explain and address the generic difficulties of training DNNs, and to improve efficient use of network parameters; they do not have designs on properties of solutions to which network training should converge. In contrast, there exist other deep learning methods that have favored solution properties of network parameters, and expect such properties to guarantee *good generalization at inference time*. In this work, we specially focus on DNN methods that impose explicit regularization on weight matrices of network layers [51], [56]. For example, Sokolic *et al.* [51] propose by theoretical analysis a soft regularizer that penalizes Frobenius norm of the Jacobian. More recently, methods that regularize the whole spectrum of singular values and its range for each of networks' weight matrices are also proposed [2], [5], [13], [30], [57], [58]. They achieve clearly improved performance over those without imposing such a regularization. However, many of these methods are empirically motivated, with no theoretical justification on its effect on generalization. We aim to study this theoretical issue in this work.

Motivated by geometric intuitions from isometric mappings [28], we introduce a term of *local isometry* into the framework of generalization analysis via algorithmic robustness [60]. We use an intuitive and also formal definition of *instance-wise variation space* to characterize data distributions of practical interest, and prove that DNNs are of local isometry on such data distributions. More specifically, we prove that for a DNN trained on such a data distribution, a covering based on a linear partition (induced by the DNN) of the input space can be found such that DNN is *locally* linear in each covering ball, where we give bound on the diameters of covering balls in terms of spectral norms of the DNN's weight matrices. Based on a further proof that for a mapping induced by a linear DNN, degree of isometry is fully controlled by singular value spectrum of each of its weight matrices, we establish our generalization error (GE) bound for (nonlinear) DNNs, and show that it is *both scale- and range-sensitive to singular value spectrum of each of their weight matrices*. An illustration of our proofs is given in fig. 1.

- *K. Jia, S. Li, and Y. Wen are with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China. E-mails: kuijia@scut.edu.cn, lishuai918@gmail.com, wen.yuxin@mail.scut.edu.cn*
- *T. Liu and D. Tao are with the Faculty of Engineering and Information Technologies, The University of Sydney, Darlington, NSW, Australia. E-mails: tliang.liu@gmail.com, dacheng.tao@sydney.edu.au*

** indicates equal contribution.*

Derivation of our bound is based on a new covering of the sample space, as illustrated in fig. 2, which enables explicit characterization of GEs caused by both the *distance expansion* and *distance contraction* of locally isometric mappings.

To attain an optimal GE bound w.r.t. the degree of isometry, we prove that the optimum is achieved when each weight matrix of a DNN has a spectrum of equal singular values, among which orthogonal weight matrix or a non-square one with orthonormal rows or columns is the most straightforward choice, suggesting the algorithms of *Orthogonal Deep Neural Networks (OrthDNNs)*. Training to obtain a *strict OrthDNN* amounts to optimizing the weight matrices over their respective Stiefel manifolds, which, however, is very costly for large-sized DNNs. To achieve efficient learning, we propose a simple yet effective algorithm of *approximate OrthDNNs* called *Singular Value Bounding (SVB)*. SVB periodically bounds, in the SGD based training iterations, all singular values of each weight matrix in a narrow band around the value of 1, thus achieving near orthogonality (row- or column-wise orthonormality) of weight matrices. In this work, we also discuss alternative schemes of soft regularization [4], [13], [58] to achieve approximate OrthDNNs, and compare with our proposed SVB. Batch Normalization (BN) [29] is commonly used in modern DNNs, yet it has a potential risk of ill-conditioned layer transform, making it incompatible with OrthDNNs. We propose *Degenerate Batch Normalization (DBN)* and *Bounded Batch Normalization (BBN)* to remove such a potential risk, and to enable its use with strict and approximate OrthDNNs respectively.

To investigate the efficacy of OrthDNNs, we conduct extensive experiments of benchmark image classification [36], [47] on modern architectures [23], [27], [50], [59], [62]. These experiments show that OrthDNNs consistently improve generalization by providing regularization to training of these architectures. Interestingly, approximate OrthDNNs perform as well as strict ones, but at a much lower computational cost. For approximate OrthDNNs, we also compare hard regularization via our proposed SVB and BBN with the alternatives of soft regularization; our results are better than or comparable to those of these alternatives on modern architectures. In some of these studies, we investigate behaviors of our method under learning regimes from small to large sizes of training samples; results confirm the empirical strength of our method, especially for learning problems of smaller sample sizes. We also investigate robustness of our method against corruptions that are commonly encountered in natural images; our results demonstrate better robustness against such corruptions, and the robustness stands gracefully with increase of corruption severity levels.

1.1 Relations with existing works

1.1.1 Generalization analysis of DNNs

Classical theories of DNNs show that they are universal approximators [6], [26]. However, recent results from Zhang *et al.* [63] show an apparent puzzle that over-parameterized DNNs are able to shatter randomly labeled training data, suggesting worst-case generalization since test performance can only be at a chance level, while at the same time they perform well on practical learning tasks (e.g., ImageNet classification); the puzzle suggests that traditional analysis of data-independent generalization does not readily apply. They further conjecture [64] that over-parameterized DNNs, when trained via SGD, tend to find local solutions that fall in, with high probability, flat regions in the high-dimensional solution space, which is even obvious when learning tasks are on natural signals; flat-region solutions imply robustness in the parameter space of DNNs, which may further implies robustness in the input data space. Similar argument of flat-region solutions is also presented in [33], although Dinh *et al.* [15] argue that these flat minima can be equivalently converted as sharp minima without affecting network prediction. Generalization of DNNs is also explained by stochastic optimization. In [21], the notion of uniform stability [10] is extended to characterize the randomness of SGD, and a generalization bound in expectation is established for learning with SGD. The distribution-free stability bound of [21] is improved in [37] via the notion of on-average stability, revealing data-dependent behavior of SGD. To understand practical generalization of DNNs, Kawaguchi *et al.* [32] argue that independent of the hypothesis set and algorithms used, the learned model itself, possibly selected via a validation set, is the most important factor that accounts for good generalization; a generalization bound w.r.t. validation error is also presented in [32].

To further characterize generalization of DNNs with their weight matrices, Sokolic *et al.* [51] study DNNs as robust large-margin classifiers via the algorithmic robustness framework [60]. They introduce a notion of average Jacobian, and use spectral norm of the Jacobian matrix to locally bound the distance expansion from the input to the output space of a DNN; spectral norm of the Jacobian is further relaxed as the product of spectral norms of the network's weight matrices, which is used to establish the robustness based generalization bound. Bartlett *et al.* [7] use a scale-sensitive measure of complexity to establish a generalization bound. They derive a margin-normalized spectral complexity, i.e., the product of spectral norms of weight matrices divided by the margin, via covering number approximation of Rademacher complexity; they further show empirically that such a bound is task-dependent, suggesting that SGD training learns parameters of a DNN whose complexity scales with the difficulty of the learning task.

While both of our bound and that of [51] are developed under the framework of algorithmic robustness [60], our bound is controlled by the whole spectrum of singular values, rather than spectral norm (i.e., the largest singular value) of each of the network's weight matrices, by introducing a term of local isometry into the framework. This also means that in contrast to [7], our bound is both scale- and range-sensitive to singular values of weight matrices. The fact that our bound is scale-sensitive in the sense of [7] implies that for difficult learning tasks, e.g., randomly labeled CIFAR10 [63], spectral norms of weight matrices would go extremely large, causing the diameters of covering balls go extremely small and correspondingly the second term of our bound (cf. theorem 3.2) that characterizes distribution mismatch between training and test samples dominates, and that the bound becomes vacuous. In contrast, for learning tasks of practical interest, e.g., standard CIFAR10 [36], the spectra of singular values of weight matrices are potentially in a benign range, and the bound is of practical use to inspire

design of improved learning algorithms.

1.1.2 Compositional computations and isometries of DNNs

Montúfar *et al.* [43] characterize complexity of functions computable by DNNs and establish a lower bound on the maximal number of linear regions into which a DNN (with ReLU activation) can partition the input space, where the bound is derived by compositional replication of layer-wise space partitioning and grows exponentially with depth of the DNN. Our derivation of the analytic form of region-wise linear mapping (cf. lemma 3.2) borrows ideas from [43]. Similar compositional derivations for the number of computational paths from the network input to a hidden unit are also presented in [31], [32].

Geometric intuition of isometric mappings has been introduced to improve robustness of deep feature transformation [28], where DNNs are studied as a form of transformation functions. However, their development of robustness bound only uses explicitly the distance expansion constraint of isometric mappings; moreover, their studies are in the context of metric learning and for DNNs, they stay on a general function form, with no indications on how layer-wise weight matrices affect generalization.

1.1.3 Optimization benefits of isometry/orthogonality

Previous works [30], [48], [58] show that orthogonality helps the optimization of DNNs by preventing explosion or vanishing of back-propagated gradients. More specifically, a property of dynamic isometry is studied in [48] to understand learning dynamics of deep linear networks. Pennington *et al.* [46] extend such studies to DNNs by employing powerful tools from free probability theory; they show that with orthogonal weight initialization, sigmoid activation functions can keep the maximum singular value to be 1 as layers go deeper, and isometry of DNNs can be preserved for a large amount of time during training. However, the analysis on optimization benefits does not explain the gain in test accuracy, i.e., the generalization.

1.1.4 Regularization on weight matrices

Wang *et al.* [56] propose Extended Data Jacobian Matrix (EDJM) as a network analyzing tool, and study how the spectrum of EDJM affects performance of different networks of varying depths, architectures, and training methods. Based on these observations, they propose a spectral soft regularizer that encourages major singular values of EDJM to be closer to the largest one (practically implemented on weight matrix of each layer). As discussed above, a related notion of average Jacobian is used in [51] to motivate a soft regularizer that penalizes spectral norms of weight matrices.

There exist other recent methods [5], [13], [30], [58] that improve empirical performance of DNNs by regularizing the whole spectrum of singular values for each of networks' weight matrices. This is implemented in [13], [58] as soft regularizers that encourage the product between each weight matrix and its transpose to be close to an identity one. Different from [13], [58], we propose a hard regularization method termed Singular Value Bounding (SVB), which periodically bounds in the training process all singular values of each weight matrix in a narrow band around the value of 1, so that orthonormality of rows or columns of weight matrices can be approximately achieved.

1.2 Contributions

There exists a growing recent interest on using spectral regularization to improve training of DNNs. These methods impose explicit regularization on weight matrices of network layers by penalizing either their spectral norms [51], [56] or the whole spectrums of their singular values [5], [13], [30], [58]. The SVB algorithm proposed in our preliminary work [30] is among the later approach. Most of these methods are empirically motivated with no theoretical guarantees. In the present paper, we focus on theoretical analysis of these methods from the perspective of generalization analysis, and prove a novel GE bound for data distributions of practical interest. We also intensively compare empirical performance of these methods, and present their empirical strengths under various learning scenarios. We summarize our technical contributions as follows.

- We present in this paper a new generalization error bound for DNNs. We first prove that DNNs are of *local isometry* on data distributions of practical interest, where the degree of isometry is fully controlled by singular value spectrum of each of their weight matrices. By using a new covering of the sample space and introducing the local isometry property of DNNs into an algorithmic robustness framework, we establish our GE bound and show that it is *both scale- and range-sensitive to singular value spectrum of each of networks' weight matrices*.
- We prove that the optimal bound w.r.t. the degree of isometry is attained when each weight matrix of a DNN has a spectrum of equal singular values, among which orthogonal weight matrix or a non-square one with orthonormal rows or columns is the most straightforward choice, suggesting the algorithms of *Orthogonal Deep Neural Networks (OrthDNNs)*. In this paper, we also present the algorithmic details of OrthDNNs.
- To address the heavy computation of *strict OrthDNNs*, we propose a novel algorithm called *Singular Value Bounding (SVB)*, which achieves *approximate OrthDNNs* via a simple scheme of hard regularization. We discuss alternative schemes of soft regularization, and compare with our proposed SVB. Batch normalization has a potential risk of ill-conditioned layer transform, making it incompatible with OrthDNNs. We propose *Degenerate Batch Normalization (DBN)* and *Bounded Batch Normalization (BBN)* to remove such a potential risk, and to enable its use with strict and approximate OrthDNNs.

2 Problem Statement

We start by describing the formalism of classification problems that jointly learn a representation and a classifier, e.g., via Deep Neural Networks (DNNs).

2.1 The classification-representation-learning problem and its generalization error

Assume a sample space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the instance space and $\mathcal{Y}$ is the label space. We restrict ourselves to classification problems in this paper, and have $\boldsymbol{x} \in \mathcal{X}$ as vectors in $\mathbb{R}^n$ and $y \in \mathcal{Y}$ as a positive integer less than $|\mathcal{Y}| \in \mathbb{N}$. We use $S_m = \{s_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^m$ to denote the training set of size m whose examples are drawn independent and identically distributed (i.i.d.) according to an unknown distribution P. We also denote $S_m^{(x)} = \{\boldsymbol{x}_i\}_{i=1}^m$. Given a loss function $\mathcal{L}$, the goal of learning is to identify a function $f_{S_m} : \mathcal{X} \mapsto \mathcal{Y}$ in a hypothesis space (a class $\mathcal{F}$ of functions) that minimizes the expected risk

$$R(f) = \mathbb{E}_{z\sim P}\left[\mathcal{L}\left(f(\boldsymbol{x}), y\right)\right],$$

where $z = (\boldsymbol{x}, y) \in \mathcal{Z}$ is sampled i.i.d. according to P. Since P is unknown, the observable quantity serving as a proxy to the expected risk $R(f)$ is the empirical risk

$$R_m(f) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(f(\boldsymbol{x}_i), y_i\right).$$

One of the primary goals in statistical learning theory is to characterize the discrepancy between $R(f_{S_m})$ and $R_m(f_{S_m})$, which is termed as *generalization error* — it is sometimes termed as generalization gap in the literature

$$\mathrm{GE}(f_{S_m}) = |R(f_{S_m}) - R_m(f_{S_m})|.$$

In this paper, we are interested in using DNNs to solve classification problems. It amounts to learning a map T, which extracts feature characteristic to a classification task, and minimizing R_m simultaneously. We denote classification with this approach as a *Classification-Representation-Learning* (CRL) problem. We single out the map T because most of the theoretical analysis in this paper resolves around it. Rewriting the two risks by incorporating a map T (we write $\boldsymbol{T}$ when it is instantiated by a DNN), we have

$$R(f, T) = \mathbb{E}_{z\sim P}\left[\mathcal{L}\left(f(T\boldsymbol{x}), y\right)\right], \tag{1}$$

$$R_m(f, T) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(f(T\boldsymbol{x}_i), y_i\right). \tag{2}$$

2.2 Generalization analysis for robust algorithms with isometric mapping

The upper bounds of GE are generally established by leveraging on certain measures related to the capacity of hypothesis space $\mathcal{F}$, such as Rademacher complexity and VC-dimension [42]. These complexity measures capture global properties of $\mathcal{F}$; however, GE bounds based on them ignore the specifically used learning algorithms. To establish a finer bound, one may resort to algorithm-dependent analysis [41], [60]. Our analysis of GE bound in this work is based on the algorithmic robustness framework [60] that has the advantage of conveying information of local geometry. We begin with the definition of robustness used in [60].

Definition 1 ($(K, \epsilon(\cdot))$-robustness). *An algorithm is $(K, \epsilon(\cdot))$-robust for $K \in \mathbb{N}$ and $\epsilon(\cdot) : \mathcal{Z}^m \mapsto \mathbb{R}$, if $\mathcal{Z}$ can be partitioned into K disjoint sets, denoted by $\mathcal{C} = \{C_k\}_{k=1}^K$, such that the following holds for all $s_i = (\boldsymbol{x}_i, y_i) \in S_m, z = (\boldsymbol{x}, y) \in \mathcal{Z}, C_k \in \mathcal{C}$:*

$$\begin{aligned} &\forall s_i = (\boldsymbol{x}_i, y_i) \in C_k, \forall z = (\boldsymbol{x}, y) \in C_k \\ \Longrightarrow\ &|\mathcal{L}(f(\boldsymbol{x}_i), y_i) - \mathcal{L}(f(\boldsymbol{x}), y)| \le \epsilon(S_m). \end{aligned}$$

The gist of the definition is to constrain the variation of loss values on test examples w.r.t. those of training ones through local property of the algorithmically learned function. Intuitively, if $s \in S_m$ and $z \in \mathcal{Z}$ are "close" (e.g., in the same partition C_k), their loss should also be close, due to the intrinsic constraint imposed by f.

For any algorithm that is robust, [60] proves

Theorem 2.1 ([60]). *If a learning algorithm is $(K, \epsilon(\cdot))$-robust and $\mathcal{L}$ is bounded, a.k.a. $\mathcal{L}(f(\boldsymbol{x}), y) \le M\ \forall z \in \mathcal{Z}$, for any $\nu > 0$, with probability at least $1 - \nu$ we have*

$$GE(f_{S_m}) \le \epsilon(S_m) + M\sqrt{\frac{2K\log(2) + 2\log(1/\nu)}{m}}. \tag{3}$$

To control the first term, a natural approach is to constrain the variation of the loss function. Covering number [35] provides a way to bound the variation of the loss function, and more importantly, it conceptually realizes the actual number K of disjoint partitions.

Definition 2 (Covering number). *Given a metric space $(\mathcal{S}, \rho)$, we say that a subset $\hat{\mathcal{S}}$ of $\mathcal{S}$ is a γ-cover of $\mathcal{S}$, if $\forall s \in \mathcal{S}, \exists \hat{s} \in \hat{\mathcal{S}}$ such that $\rho(s, \hat{s}) \le \gamma$. The γ-covering number of $\mathcal{S}$ is*

$$\mathcal{N}_\gamma(\mathcal{S}, \rho) = \min\{|\hat{\mathcal{S}}| : \hat{\mathcal{S}} \text{ is a } \gamma\text{-covering of } \mathcal{S}\}.$$

In [28], they propose δ-isometry as a desirable property in CRL problem to help control the variation, where δ-isometry is a geometric property of mapping functions.

Definition 3 (δ-isometry). *Given a map T that maps a metric space $(\mathcal{P}, \rho_P)$ to another metric space $(\mathcal{Q}, \rho_Q)$, it is called δ-isometry if the following inequality holds*

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{P}, |\rho_Q(T\boldsymbol{x}, T\boldsymbol{x}') - \rho_P(\boldsymbol{x}, \boldsymbol{x}')| \le \delta.$$

When T in eq. (1) and eq. (2) is of δ-isometry, by using $\rho_Q(T\boldsymbol{x}, T\boldsymbol{x}') \le \rho_P(\boldsymbol{x}, \boldsymbol{x}') + \delta$ a realization of algorithmic robustness (or GE bound in the form of Theorem 2.1) similar to [28] can be established for DNNs as follows.

Theorem 2.2. *Given an algorithm in a CRL problem, if the Lipschtiz constant of $\mathcal{L} \circ f$ w.r.t. $\boldsymbol{T}\boldsymbol{x}$ is bounded by A, T is of δ-isometry, and $\mathcal{X}$ is compact with a covering number $\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho)$, then it is $(|\mathcal{Y}|\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho), A(\gamma + \delta))$-robust.*

Remark. *The result in [28] is $(|\mathcal{Y}|\mathcal{N}_{\gamma/2}(\mathcal{X}, \rho), 2A(\gamma + \delta))$-robust; the factor of 2 in the second term is dropped here due to the fact that in CRL problems, we are not doing metric learning as in [28], which involves two pairs of examples, and we only compare one pair of examples. Its proof under the context of DNN, i.e., the proof of theorem 3.1, is given in Appendix E.*

Remark. *Denote $\rho_Q(T\boldsymbol{x}, T\boldsymbol{x}') \le \rho_P(\boldsymbol{x}, \boldsymbol{x}') + \delta$ as the* expansion property *of the δ-isometry, and $\rho_Q(T\boldsymbol{x}, T\boldsymbol{x}') \ge \rho_P(\boldsymbol{x}, \boldsymbol{x}') - \delta$ as its* contraction property. *We note that the above theorem is established by only exploiting the expansion property. After proving that DNNs achieve locally isometric mappings in section 3.1, we will show that a better generalization can be derived by considering both the properties.*

2.3 Notations of deep neural networks

We study the map $\boldsymbol{T}$ as a neural network. We present the definition of Multi-Layer Perceptron (MLP) here, which captures all ingredients for theoretical analysis and enables us to convey the analysis without unnecessary complications, though we note that the analysis extends to Convolutional Neural Networks (CNNs) almost equally.

A MLP is a map that takes an input $\boldsymbol{x} \in \mathbb{R}^n$ from the space $\mathcal{X}$, and builds its output by recursively applying a linear map $\boldsymbol{W}_l$ followed by a pointwise non-linearity g

$$\boldsymbol{x}_l = g(\boldsymbol{W}_l \boldsymbol{x}_{l-1}), \tag{4}$$

where $l \in \{1, \ldots, L\}$ indexes the layer, $\boldsymbol{x}_l \in \mathbb{R}^{n_l}$, $\boldsymbol{x}_0 = \boldsymbol{x}$, $\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$, and g denotes the activation function, which throughout the paper is the Rectifier Linear Unit (ReLU) [20]. Optionally g may include max pooling operator [8] after applying ReLU. We also denote the intermediate feature space $g(\boldsymbol{W}_l \boldsymbol{x}_{l-1})$ as $\mathcal{X}_l$ and $\mathcal{X}_0 = \mathcal{X}$. Each $\mathcal{X}_l$ is a metric space, and throughout the paper the metric is taken as the ℓ_2 norm $||\cdot||_2$, shortened as $||\cdot||$. We compactly write the map of a MLP as

$$\boldsymbol{T}\boldsymbol{x} = \boldsymbol{W}_L g(\boldsymbol{W}_{L-1} \ldots g(\boldsymbol{W}_1 \boldsymbol{x})).$$

We denote the spectrum of singular values of a matrix $\boldsymbol{W}$ by $\sigma(\boldsymbol{W})$, and $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum (nonzero) singular values of $\boldsymbol{W}$ respectively. We denote the rank of a matrix $\boldsymbol{W}$ by $r(\boldsymbol{W})$, and the null space of $\boldsymbol{W}$ by $\mathcal{N}(\boldsymbol{W})$. We write the complement of $\mathcal{N}(\boldsymbol{W})$ as $\mathcal{X} - \mathcal{N}(\boldsymbol{W})$.

3 Generalization Bounds of Deep Neural Networks

In this section, we develop GE bounds for CRL problems instantiated by DNNs. We identify two quantities that help control a bound, i.e., δ-isometry of $\boldsymbol{T}$ and the diameter γ of covering balls of $\mathcal{X}$. We show that both of the two quantities can be controlled by constraining the spectrum of singular values of the weight matrix associated with each network layer, i.e., spectrums of singular values of $\{\boldsymbol{W}_i\}_{i=1,\ldots,L}$.

To proceed, we consider in this paper variations of instances in $\mathcal{X}$ that are of practical interest — more specifically, those that output nonzero vectors after passing through a DNN. We first prove in lemma 3.1 that in such a variation subspace, mapping induced by a *linear* neural network is of δ-isometry, where δ is specified by the *maximum and minimum* singular values of weight matrices of all the network layers. For a *nonlinear* neural network, where we assume ReLU activation and optionally with max pooling, we consider the fact that it divides the input space $\mathcal{X}$ into a set of regions and within each region, it induces a linear mapping. In lemma 3.2, we specify the explicit form of region-wise mapping $\boldsymbol{T}_q$, associated with any linear region q, with submatrices of $\{\boldsymbol{W}_i^q\}_{i=1,\ldots,L}$, based on which we prove in lemma 3.3 that a covering set for $\mathcal{X}$ can be found with a diameter γ of covering balls that is upper bounded by a quantity inversely proportional to the product of maximum singular values of weight matrices of some network layers. With the δ and γ specified in lemma 3.1 and lemma 3.3, we further prove in lemma 3.4 that in the instance-wise variation subspaces considered in this paper, a nonlinear neural network $\boldsymbol{T}$ is of local δ-isometry within each covering ball. The proofs are illustrated in fig. 1.

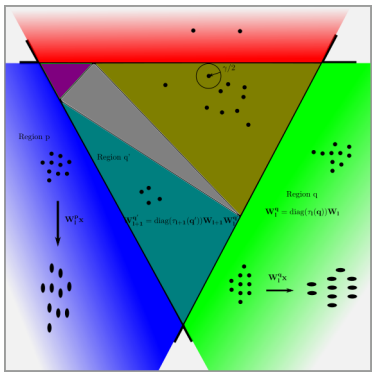

Fig. 1. Illustration for the local δ-isometry of a Deep Neural Network (DNN) with ReLU activation and max pooling functions. Black dots represent training instances in the sample space, and regions coded with different colors represent regions in the instance space that are hierarchically specified by layers of the DNN. The illustration depicts proofs of lemmas 3.1, 3.2, 3.3, and 3.4. Lemma 3.1 proves that the mapping induced by a linear DNN is of δ-isometry, where δ specified the expansion and contraction properties of the mapping and is determined by singular value spectrums of weight matrices of all the layers. Lemma 3.2 proves that a nonlinear DNN partitions the instance space into increasingly refined regions. As illustrated in the figure, the space is firstly partitioned into coarser regions (i.e., the center triangle and other three regions color coded as red, blue, and green), an additional layer further partitions some of the coarser regions into sets of smaller regions (e.g., those inside the region of center triangle), and the process goes recursively. Suppose that a region q is created by layer l of the DNN, and in region q, the nonlinear mapping defined by the matrix $\boldsymbol{W}_l$ and activation function reduces to a linear mapping of $\boldsymbol{W}_l^q = \text{diag}(\tau_l(q))\boldsymbol{W}_l$, where $\tau_l(q)$ is a binary vector roughly indicating active neurons, and $\text{diag}(\cdot)$ diagonalizes $\tau_l(q)$. Suppose q' is created by layer $l+1$ at the bottom part of the center triangle, and in region q', the nonlinear mappings of layer l and layer $l+1$ are reduced to a linear mapping of $\boldsymbol{W}_{l+1}^{q'} = \text{diag}(\tau_{l+1}(q'))\boldsymbol{W}_{l+1}\boldsymbol{W}_l^{q'}$, where symbols have similar meanings as described above. The phenomenon enables to find a covering for the sample space, such that in each covering ball that contains training instances, e.g., $\boldsymbol{x}$, the DNN $\boldsymbol{T}$ defines a transformation that can be characterized as a linear mapping, e.g., $\boldsymbol{T}_{|\boldsymbol{x}}\boldsymbol{x}$. This is proved in lemma 3.3. Radius of the covering ball is illustrated as $\gamma/2$ in the figure — a radius is acceptable as long as it is less than the smallest distance from any of the training instances to their respective region boundaries. The behaviors of δ-isometry of $\boldsymbol{T}_{|\boldsymbol{x}}$ are also visualized in regions p and q respectively. The transformation of $\boldsymbol{T}$ applied on instances $\boldsymbol{x}$ is different in different regions. As a demonstration, in region p, the transformation vertically elongates the distance between instances, while in region q, it horizontally elongates the distance instead. Lastly, in lemma 3.4, we prove that by Cauchy interlacing law, a nonlinear DNN is of local δ-isometry within each covering ball specified above.

To develop a GE bound, we propose a covering scheme that includes instances of different labels into the same balls, thus reducing the size of covering set when compared with that in theorem 2.1. The covering scheme is illustrated in fig. 2. We correspondingly characterize both the errors caused by *distance contraction* between instances of different labels and those by *distance expansion* between instances of the same labels. Based on such characterization, we come with our *main result of theorem 3.2*.

Given the bound in theorem 3.2, we prove in lemma 3.5 that the optimal bound w.r.t. δ is obtained when *all singular*

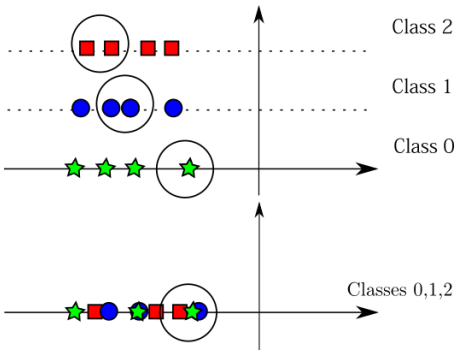

Fig. 2. Illustration of our used covering scheme that includes instances of different labels into same balls. Colored squares, circles, and stars represent instances of different labels, and unfilled (large) circles represent covering balls. *Top:* existing works [51], [60] usually separate instances of different labels into different covering balls, either by assuming that distances between instances of different labels in the sample space are infinite, or by using covering balls that are small enough not to contain instances of different labels; such a scheme can characterize the generalization errors caused by improper expansion of intra-class variations, but it cannot characterize the errors caused by improper contraction of inter-class differences. *Bottom:* we use a covering scheme that includes instances into covering balls regardless of their labels; it enables characterization of both types of the aforementioned errors, and leads to a possibly tighter generalization error bound in some cases.

values of the weight matrix of each network layer are of equal ones, which inspires a straightforward choice of enforcing all singular values to have the value of 1, and thus the algorithms of OrthDNNs.

3.1 δ-isometry in deep neural networks

We begin with a few definitions necessary for the subsequent analysis.

Definition 4 (Variation subspace of an instance)**.** *Given an instance $\boldsymbol{x} \in \mathcal{X}$, suppose we are interested in the variation of a set $\mathcal{X}' \subseteq \mathcal{X}$ w.r.t. $\boldsymbol{x}$. We call the linear vector space*

$$span(\{\boldsymbol{x}' - \boldsymbol{x} | \boldsymbol{x}' \in \mathcal{X}'\})$$

the variation subspace w.r.t. the instance $\boldsymbol{x}$ of $\mathcal{X}$, shorten as variation subspace of instance $\boldsymbol{x}$.

The definition is to formalize variations of interest of particular instances, thus enabling us to discuss what variations a DNN is able to constrain. Correspondingly, we have the following definition of isometry.

Definition 5 (δ-isometry w.r.t. variation subspace of an instance)**.** *Given a map $\boldsymbol{T}$ that maps a metric space $(\mathcal{P}, \rho_P)$ to another metric space $(\mathcal{Q}, \rho_Q)$, it is called δ-isometry w.r.t. the variation subspace $\mathcal{P}_x$ of instance $\boldsymbol{x}$, if the following holds*

$$\forall \boldsymbol{x}' \in \{\boldsymbol{z} \mid \boldsymbol{z} - \boldsymbol{x} \in \mathcal{P}_x\}, |\rho_Q(\boldsymbol{T}\boldsymbol{x}, \boldsymbol{T}\boldsymbol{x}') - \rho_P(\boldsymbol{x}, \boldsymbol{x}')| \leq \delta.$$

We provide an example here to describe δ-isometry w.r.t. the variation subspace of a linear DNN.

Example 1. *In a linear DNN $\boldsymbol{T}$, given an instance $\boldsymbol{x}$, it is δ-isometry w.r.t. variation subspace $\mathcal{X} - \mathcal{N}(\boldsymbol{T})$. This is proved in lemma 3.1. Thus, for any variation $\delta\boldsymbol{x} \in \mathcal{X} - \mathcal{N}(\boldsymbol{T})$, we have $|\rho(\boldsymbol{T}\boldsymbol{x}, \boldsymbol{T}(\boldsymbol{x} + \delta\boldsymbol{x})) - \rho(\boldsymbol{x}, \boldsymbol{x} + \delta\boldsymbol{x})| \leq \delta$. In this case, the variation subspace $\mathcal{X} - \mathcal{N}(\boldsymbol{T})$ is the same for any instance $\boldsymbol{x}$.*

The following lemma specifies for a *linear* DNN the δ-isometry w.r.t. data variations of practical interest.

Lemma 3.1. *Given a linear neural network $\boldsymbol{T}$ and an instance $\boldsymbol{x} \in \mathcal{X}$, if $||\boldsymbol{x}|| \leq b$, i.e., instances are norm bounded, then $\boldsymbol{T}$ is of $2b\max(|\prod_{i=1}^{L}\sigma_{\max}^i - 1|, |\prod_{i=1}^{L}\sigma_{\min}^i - 1|)$-isometry w.r.t. the variation subspace $\mathcal{X} - \mathcal{N}(\boldsymbol{T})$ of the instance $\boldsymbol{x}$.* [1]. *We also have $\forall \boldsymbol{x}' \in \{\boldsymbol{z} \mid \boldsymbol{z} - \boldsymbol{x} \in \mathcal{P}_x\}$, $\rho_Q(\boldsymbol{T}\boldsymbol{x}, \boldsymbol{T}\boldsymbol{x}') \leq \rho_P(\boldsymbol{x}, \boldsymbol{x}') + 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|$ and $\rho_Q(\boldsymbol{T}\boldsymbol{x}, \boldsymbol{T}\boldsymbol{x}') \geq \rho_P(\boldsymbol{x}, \boldsymbol{x}') + \rho_P(\boldsymbol{x}, \boldsymbol{x}')(\prod_{i=1}^{L}\sigma_{\min}^i - 1)$.*

See the proof in appendix A.

The above lemma shows that as long as $\boldsymbol{x}$ varies within the complement of the null space of a linear DNN $\boldsymbol{T}$, we can constrain variations induced by the mapping by the specified δ. Outside the space, it is unlikely of practical interest since $\boldsymbol{T}$ discards all the information about the variations.

To further proceed for nonlinear DNNs, we introduce some terminologies from hyperplane arrangement [44], and give definitions to describe the objects of interest exactly.

Definition 6 ((Finite) Hyperplane arrangement)**.** *A finite hyperplane arrangement $\mathcal{A}$ is a finite set of affine hyperplanes in some vector space $\mathcal{X} \equiv \mathbb{K}^n$, where $\mathbb{K}$ is a field and is taken as $\mathbb{R}$ in this paper.*

Definition 7 (Region)**.** *Denote $H \in \mathcal{A}$ an element of the arrangement, a region of the arrangement is a connected component of the complement $\mathbb{R}^n - \bigcup_{H \in \mathcal{A}} H$. The set of all regions is denoted as $\mathcal{R}(\mathcal{A})$, shortened as $\mathcal{R}$ when no confusion exists.*

We set up a labeling scheme for $r \in \mathcal{R}$. Choosing a linear order in $\mathcal{A}$, we write $\mathcal{A} = \{H_i\}_{i=1,\ldots,n}$ and $H_i = \ker \alpha_i$, where ker denotes the kernel $\{\boldsymbol{x} \in \mathcal{X} | \alpha_i(\boldsymbol{x}) = \langle \boldsymbol{\alpha}_i, \boldsymbol{x} \rangle = 0\}$ and $\boldsymbol{\alpha}_i$ is the normal of hyperplane H_i. Let $\mathbb{J} = \{1, 0\}$, and $\pi_i : \mathbb{J}^n \to \mathbb{J}$ be the projection onto the i-th coordinate. Define a map $\tau : \mathcal{X} \to \mathbb{J}^n$ by

$$\pi_i \tau(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \alpha_i(\boldsymbol{x}) > 0 \\ 0 & \text{if } \alpha_i(\boldsymbol{x}) \leq 0. \end{cases}$$

With the scheme, for any $\mathcal{A}$, we would have an index set $\mathcal{T}(\mathcal{A}, \tau)$, shortened as $\mathcal{T}$, such that for any $r \in \mathcal{R}$, it corresponds to a unique element in $\mathbb{J}^n$, denoted as $\tau(r)$. We will use $\tau(\boldsymbol{x})$ —- labeling on elements, and $\tau(r)$ — labeling on regions, interchangely.

Definition 8 (Neuron)**.** *A neuron a_{lk} of a neural network $\boldsymbol{T}$ is a functional defined by*

$$a_{lk}(\boldsymbol{x}) = \pi_k g(\boldsymbol{W}_l g(\boldsymbol{W}_{l-1} \ldots g(\boldsymbol{W}_1 \boldsymbol{x}))),$$

where $l \in \{1, \ldots, L\}$ and $k \in \{1, \ldots, n_l\}$. All the neurons at layer l define a map, denoted as

$$a_l(\boldsymbol{x}) = g(\boldsymbol{W}_l g(\boldsymbol{W}_{l-1} \ldots g(\boldsymbol{W}_1 \boldsymbol{x}))).$$

The lemma that follows is mostly an analysis of the domain of a DNN $\boldsymbol{T}$. We begin with the following definition.

1. Note that the space $\mathcal{X} - \mathcal{N}(\boldsymbol{T})$ does not depend on $\boldsymbol{x}$, and this is a trivial case where the variation subspaces of all instances are the same. We will see nontrivial cases later when dealing with nonlinear neural networks.

Definition 9 (Support of Neuron/DNN). *Given a neuron a_{lk}, the support of a_{lk} is the set of instances in $\mathcal{X}$ that satisfy*

$$supp(a_{lk}) = \{\boldsymbol{x} \in \mathcal{X} | a_{lk}(\boldsymbol{x}) \neq 0\}.$$

Similarly, the support of a neural network $\boldsymbol{T}$ is the set of instances in $\mathcal{X}$ that satisfy

$$supp(\boldsymbol{T}) = \{\boldsymbol{x} \in \mathcal{X} | \boldsymbol{T}\boldsymbol{x} \neq 0\}.$$

Lemma 3.2. *A nonlinear neural network $\boldsymbol{T}$ divides $\mathcal{X}$ into a set of regions $\mathcal{Q}$, and within each region $q \in \mathcal{Q}$, $\boldsymbol{T}$ is linear w.r.t. variations of instances as long as they vary within q. We denote the linear mapping at q as $\boldsymbol{T}_q$ and have*

$$\boldsymbol{T}_q = \prod_{i=1}^{L} \boldsymbol{W}_i^q$$
$$\boldsymbol{W}_l^q = diag(\tau_l(q))\boldsymbol{W}_l,$$

where when layer l does not contain max pooling, and $\tau_l(q)$ is defined as

$$\pi_k \tau_l(q) = \pi_k \tau_l^{relu}(q) = \begin{cases} 1 & \text{if } a_{lk}(\boldsymbol{x}) > 0, \forall \boldsymbol{x} \in q \\ 0 & \text{if } a_{lk}(\boldsymbol{x}) \leq 0, \forall \boldsymbol{x} \in q. \end{cases}$$

When layer l does contain max pooling, we define $\boldsymbol{W}_l^q$ as

$$\boldsymbol{W}_l^q = \boldsymbol{P}_l \, diag(\tau_l(q))\boldsymbol{W}_l$$

, and $\tau_l(q)$ is defined as

$$\tau_l(q) = \tau_l^{\max}(q)\tau_l^{relu}(q).$$

In this above definition τ^{relu} is defined as before, and

$$\pi_k \tau_l^{\max}(q) = \begin{cases} 1 & \text{if } k = \operatorname{argmax}_{k \in K} a_{lk}(\boldsymbol{x}), \forall \boldsymbol{x} \in q \\ 0 & \text{otherwise}, \end{cases}$$

where K is the set of indices of neurons being pooled over; $\boldsymbol{P}_l$ is defined as (layer index suppressed)

$$\boldsymbol{P}_{ik} = \begin{cases} 1 & \text{if } k \text{ is the index that is pooled over by } i^{th} \text{ pooling area} \\ 0 & \text{otherwise}. \end{cases}$$

$\boldsymbol{P}_l$ could be understood as a matrix that for each pooling area, it sums over the dimension/area being pooled, and since only one dimension of the area is nonzero (due to $\tau_l(q)$), it outputs the maximal value. For clarity and convenience, we would use the definition without max pooling in discussion, and note that all the results present are proved for both definitions.

See the proof in appendix B.

Remark. *The function τ_l is intuitively a selection function that sets some rows of $\boldsymbol{W}_l$ to zero, and selects a submatrix from it.*

Remark. *For $q \in \mathcal{Q}$ and $q \notin supp(\boldsymbol{T})$, by the definition of linearity, $\boldsymbol{T}$ is still linear over q, i.e., the special case of $\boldsymbol{T}\boldsymbol{x} = 0, \forall \boldsymbol{x} \in q$. In this case, $\boldsymbol{x} \in \mathcal{N}(\boldsymbol{T}_q)$.*

In the proof of appendix B for lemma 3.2, we prove that within each region $q \in Q$, each neuron of a DNN is a linear functional. We summarize the result in the following corollary.

Corollary 3.1. *A nonlinear neural network $\boldsymbol{T}$ divides $\mathcal{X}$ into a set of regions $\mathcal{Q}$, and within each region $q \in \mathcal{Q}$, the k^{th} neuron a_{lk} of the layer l is linear w.r.t. variations of $\boldsymbol{x}$ within q, and we have*

$$a_{lk}^q = \pi_k \prod_{i=1}^{l} \boldsymbol{W}_i^q.$$

With the above lemma, we define the behavior of a DNN at a local area around $\boldsymbol{x} \in \mathcal{X}$ or a local area around a set $B \subset \mathcal{X}$ as below.

Definition 10 (Linear neural network and neuron induced at $\boldsymbol{x} \in \mathcal{X}$ from a nonlinear neural network). *For any given $\boldsymbol{x} \in \mathcal{X}$ with $\boldsymbol{x} \in q \in \mathcal{Q}$, we call the linear neural network $\boldsymbol{T}_q$ the linear neural network induced by a nonlinear neural network $\boldsymbol{T}$ at $\boldsymbol{x}$ — denoting it as $\boldsymbol{T}_{|\boldsymbol{x}}$, the linear neuron a_{lk}^q the linear neuron induced by the nonlinear neuron a_{lk} at $\boldsymbol{x}$ — denoting it as $a_{lk|\boldsymbol{x}}$, and the submatrix of weight matrix of each layer l the submatrix induced by nonlinearity — denoting it as $\boldsymbol{W}_{l|\boldsymbol{x}}$.*

Definition 11 (Linear neural network and neuron induced at subset $B \subset \mathcal{X}$ from a nonlinear neural network). *For any given $B \subset \mathcal{X}$ with $B \subset q \subset \mathcal{Q}$, we call the linear neural network $\boldsymbol{T}_q$ the linear neural network induced by a nonlinear neural network $\boldsymbol{T}$ at B — denoting it as $\boldsymbol{T}_{|B}$, the linear neuron a_{lk}^q the linear neuron induced by the nonlinear neuron a_{lk} at B — denoting it as $a_{lk|B}$, and the submatrix of weight matrix of each layer l the submatrix induced by nonlinearity — denoting it as $\boldsymbol{W}_{l|B}$.*

Lemma 3.3. *For any nonlinear neural network $\boldsymbol{T}$ of L layers, a covering set for $\mathcal{X}$ can be found with a diameter $\gamma = o(S_m, \boldsymbol{T}) / \left(\prod_{i=1}^{l(S_m, \boldsymbol{T})} \sigma_{\max}^i \right) > 0$, such that for any given $\boldsymbol{x} \in S_m^{(x)}$ and $\{\boldsymbol{x}' \in \mathcal{X} | \, \|\boldsymbol{x} - \boldsymbol{x}'\| \leq \gamma\}$, $\boldsymbol{T}\boldsymbol{x} - \boldsymbol{T}\boldsymbol{x}' = \boldsymbol{T}_{|\boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}')$, where $o(S_m, \boldsymbol{T})$ is a value depending on the training data and network weights, so is $l(S_m, \boldsymbol{T})$ with $1 \leq l(S_m, \boldsymbol{T}) \leq L$ (the dependence is specified in the proof), and $\sigma_{\max}^i$ is the maximum singular value of weight matrix $\boldsymbol{W}_i$ of the i^{th} layer.*

See the proof in appendix C.

We come with the local isometry property of DNNs after one more definition.

Definition 12 (γ-cover δ-isometry w.r.t. variation subspace of instance). *Given a map $\boldsymbol{T}$ that maps a metric space $(\mathcal{P}, \rho_P)$ to another metric space $(\mathcal{Q}, \rho_Q)$, and an instance $\boldsymbol{x} \in \mathcal{P}$, it is called γ-cover δ-isometry w.r.t. variation space $\mathcal{P}_{\boldsymbol{x}}$ of $\boldsymbol{x}$, if a γ-cover exists such that the following inequality holds*

$$\forall \boldsymbol{x}' \in \{\boldsymbol{z} \mid \boldsymbol{z} - \boldsymbol{x} \in \mathcal{P}_{\boldsymbol{x}}, \boldsymbol{z} \in B\}, |\rho_Q(\boldsymbol{T}\boldsymbol{x}, \boldsymbol{T}\boldsymbol{x}') - \rho_P(\boldsymbol{x}, \boldsymbol{x}')| \leq \delta,$$

where B denotes a ball given by the γ-cover.

Lemma 3.4. *Given a nonlinear neural network $\boldsymbol{T}$, if $||\boldsymbol{x}|| \leq b$ $\forall \boldsymbol{x} \in \mathcal{X}$, i.e., instances are norm bounded, then $\boldsymbol{T}$ is of γ-cover δ-isometry w.r.t. $\mathcal{X} - \mathcal{N}(\boldsymbol{T}_{|\boldsymbol{x}})$ of $\boldsymbol{x} \in S_m^{(x)}$, where δ and γ are respectively specified in lemma 3.1 and lemma 3.3.*

See the proof in appendix D.

Example 2. *In a nonlinear DNN $\boldsymbol{T}$ as defined in section 2.3, given an instance $\boldsymbol{x}$, it is δ-isometry w.r.t. variation subspace $\mathcal{X} - \mathcal{N}(T_{|\boldsymbol{x}})$. Note that for $\boldsymbol{x}$ of different instances, $T_{|\boldsymbol{x}}$ is potentially different. In practice, the singular values of weight matrices of the induced linear DNN $\boldsymbol{T}_{|\boldsymbol{x}}$ only relate to the variation in the space $\mathcal{X} - \mathcal{N}(T_{|\boldsymbol{x}})$. The bound given in the following will establish the*

relationship between generalization errors and singular values of weight matrices of DNNs by characterizing the constraints that DNNs impose on the variation in this space.

3.2 Main results of generalization bound

With the local δ-isometry property of DNNs established, we derive in this section our main results of generalization bound.

Theorem 3.1. *Given a CRL problem, the algorithm to learn is a nonlinear neural network of L layers, denoted as $\boldsymbol{T}$. Suppose the following assumptions hold: 1) $||\boldsymbol{x}|| \leq b \; \forall \; \boldsymbol{x} \in \mathcal{X}$, i.e., instances are norm bounded; 2) the loss function $\mathcal{L}$ is bounded, a.k.a. $\forall z \in \mathcal{Z}, \mathcal{L}(f(\boldsymbol{x}), y) \leq M$, and the Lipschitz constant of $\mathcal{L} \circ f$ w.r.t $\boldsymbol{T}\boldsymbol{x}$ is bounded by A; 3) $\mathcal{X}$ is a regular k-dimensional manifold with a covering number $(\frac{C_{\mathcal{X}}}{\gamma/2})^k$; 4) within each covering ball B of $\mathcal{X}$ that contains $\boldsymbol{x} \in S_m^{(x)}$, $\boldsymbol{x} - \boldsymbol{x}' \in \mathcal{X} - \mathcal{N}(T_{|B}) \; \forall \; \boldsymbol{x}, \boldsymbol{x}' \in B$. Then, for any $\nu > 0$, with probability at least $1 - \nu$ we have*

$$GE(f_{S_m}) \leq A(\gamma + \delta') + M\sqrt{\frac{\log(2)2^{k+1}|\mathcal{Y}|C_{\mathcal{X}}^k}{\gamma^k m} + \frac{2\log(1/\nu)}{m}}$$

with

$$\delta' = 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|, \gamma = \frac{o(S_m, \boldsymbol{T})}{\prod_{i=1}^{l(S_m,\boldsymbol{T})}\sigma_{\max}^i},$$

where $o(S_m, \boldsymbol{T})$ and $1 \leq l(S_m, \boldsymbol{T}) \leq L$ are values depending on the training set S_m and learned network $\boldsymbol{T}$.

A proof sketch is provided below, and the full proof is given in Appendix E.

Proof. By lemma 3.4, we have that $\boldsymbol{T}$ is of γ-cover δ-isometry w.r.t. variation space of each training instance. The expansion property of δ-isometry gives $\rho_Q(\boldsymbol{T}\boldsymbol{x}, \boldsymbol{T}\boldsymbol{x}') \leq \rho_P(\boldsymbol{x}, \boldsymbol{x}') + 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|$. By theorem 2.2, we have that DNNs are $(|\mathcal{Y}|2^k C_{\mathcal{X}}^k/\gamma^k, A(\gamma + 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|))$-robust. Note that the proof differs from theorem 2.2 subtly, for that the loss difference needs to stay in the variation space of each covering ball. Since the tricky part is also present in the proof of theorem 3.2 (the condition $\boldsymbol{x}' - \boldsymbol{x}_j \in \mathcal{P}_{\boldsymbol{x}_j}$ in eq. (7)), to avoid tautology, we do not write the full proof here. The proof is finished by applying the robustness conclusion into theorem 2.1. □

Regarding the 4^{th} assumption in theorem 3.1, it intuitively states that the local variation of interest w.r.t. each $\boldsymbol{x} \in S_m^{(x)}$ falls in the space $\mathcal{X} - \mathcal{N}(\boldsymbol{T}_{|B})$. Denote $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{x}'$, we have $\boldsymbol{T}\boldsymbol{v} = \boldsymbol{T}_{|B}\boldsymbol{v} = 0$ if $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{T}_{|B})$, i.e., the variation vanishes after passing through the network. In practice, we are not interested in such a trivial case of vanishing local variations. Instead, it is the variation in the complement $\mathcal{X} - \mathcal{N}(\boldsymbol{T}_{|B})$ that we want to constrain.

We have assumed that $\mathcal{X}$ is a regular k-dimensional manifold, whose covering number is $(\frac{C_{\mathcal{X}}}{\gamma/2})^k$, where $C_{\mathcal{X}}$ is a constant that captures the "intrinsic" properties of $\mathcal{X}$, and γ is the diameter of the covering ball. Such an assumption is general enough to accommodate at least visual data such as natural images and has been widely used [54].

In theorem 3.1, $|\mathcal{Y}|(\frac{C_{\mathcal{X}}}{\gamma/2})^k$ corresponds the covering number of the joint space $\mathcal{X} \times \mathcal{Y}$. We now show that with proper use of the contraction property of isometric mapping of DNNs, the constant $|\mathcal{Y}|$ in the second term of the upper bound can be removed while a modified first term has the potential to be small as well, indicating a better bound. Our idea is to directly exploit the covering number of the instance space $\mathcal{X}$ and measure the differences of the loss function $\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}), y)$ w.r.t. both arguments. Such a measure deals with instances of different labels but are close enough in $\mathcal{X}$, thus characterizing both the errors that are caused by erroneously contracting the distance between instances from different classes and erroneously expanding the distance between instances from the same classes, instead of those of the same classes alone. However, it will cause the issue of infinite Lipschitz constant of loss function $\mathcal{L}(f(\boldsymbol{x}), y)$. To see this, consider a binary classification problem whose loss function $\mathcal{L}$ is

$$\mathcal{L}(f(\boldsymbol{x}), y) = -\mathbf{1}_{y=1}\log f(\boldsymbol{x}) - \mathbf{1}_{y=0}\log(1 - f(\boldsymbol{x})),$$

where $(\boldsymbol{x}, y)$ is an example, $f(\boldsymbol{x})$ is a function that maps $\boldsymbol{x}$ to probability, and $\mathbf{1}$ is an indicator function. We provide an example case to illustrate the influence of metrics on $\mathcal{Z}$.
Case. Let the metric on $\mathcal{Z}$ be $\rho((\boldsymbol{x}, y), (\boldsymbol{x}', y')) = ||(\boldsymbol{x} - \boldsymbol{x}', y - y')|| = ||\boldsymbol{x} - \boldsymbol{x}'|| + |y - y'|$. Suppose that we have a pair of examples $(\boldsymbol{x}, y = 1)$ and $(\boldsymbol{x}, y' = 0)$ that only differ in labels, we have

$$A \geq \frac{|\mathcal{L}(f(\boldsymbol{x}), y) - \mathcal{L}(f(\boldsymbol{x}), y')|}{||(\boldsymbol{x}, y) - (\boldsymbol{x}, y')||} = \frac{\log(f(\boldsymbol{x})/(1 - f(\boldsymbol{x})))}{1}.$$

When there exists an $\boldsymbol{x}$ such that $f(\boldsymbol{x}) \to 1$, we have $|\mathcal{L}(f(\boldsymbol{x}), y) - \mathcal{L}(f(\boldsymbol{x}), y')| \to +\infty$. This happens because as y changes values, due to its discreteness, it could induce a jump discontinuity on $\mathcal{L}(f(\boldsymbol{x}), y)$, even though $\mathcal{L}(f(\boldsymbol{x}), y)$ is Lipschitz continuous w.r.t. $\boldsymbol{x}$. To avoid this, [60] and [28] employ a large covering number to ensure that examples in the same ball have the same label.

We note that derivation of generalization bounds for robust algorithms concerns with the loss difference $|\mathcal{L}(f(\boldsymbol{x}), y) - \mathcal{L}(f(\boldsymbol{x}'), y')|$ between example pairs. To address the aforementioned issue, we consider two separate cases for the loss difference: the cases of $y = y'$ and $y \neq y'$. For the case $y = y'$, we exploit the bounded Lipschitz constant of $\mathcal{L} \circ f$ w.r.t. $\boldsymbol{T}\boldsymbol{x}$. For the case $y \neq y'$, we introduce the following *pairwise error function* to characterize the loss difference.

Definition 13 (Pairwise error function)**.** *Given a CRL problem, of which $\mathcal{L}$ is bounded for any compact set in $\mathcal{Z}$, a.k.a. for z in any compact subset of $\mathcal{Z}$, $\mathcal{L}(f(\boldsymbol{x}), y) \leq M$, $\mathcal{X}$ is a regular k-dimensional manifold with a γ-cover, and $\boldsymbol{T}$ is of γ-cover δ-isometry, a pairwise error function (PE) of the tuple $(\mathcal{L}, f, \boldsymbol{T}, \mathcal{Z}, \gamma)$ is defined as*

$$PE(\delta) = \max_{z=(\boldsymbol{x},y)\in\mathcal{Z}} \max_{z'\in D} |\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}), y) - \mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}'), y')|$$

with $D = \{z' = (\boldsymbol{x}', y') \in \mathcal{Z} \mid \gamma - \delta \leq ||\boldsymbol{T}\boldsymbol{x}' - \boldsymbol{T}\boldsymbol{x}|| \leq \gamma + \delta, ||\boldsymbol{x} - \boldsymbol{x}'|| \leq \gamma\}$.
It characterizes the largest loss difference for examples in $\mathcal{Z}$ that may arise due to the contraction and expansion properties of δ-isometry mapping. Note that $PE(\delta)$ is a monotonously increasing function of δ — a larger δ means more feasible examples in $\mathcal{X}$ and possibly larger distance contraction/expansion, leading to a possibly larger value of $PE(\delta)$.

Theorem 3.2. *Given a CRL problem, the algorithm to learn is a nonlinear neural network of L layers, denoted as $\boldsymbol{T}$. Suppose the following assumptions hold: 1) $||\boldsymbol{x}|| \leq b \; \forall \; \boldsymbol{x} \in \mathcal{X}$, i.e., instances are norm bounded; 2) the loss function $\mathcal{L}$ is bounded, a.k.a. $\forall z \in \mathcal{Z}, \mathcal{L}(f(\boldsymbol{Tx}), y) \leq M$, and the Lipschitz constant of $\mathcal{L} \circ f$ w.r.t $\boldsymbol{Tx}$ is bounded by A; 3) $\mathcal{X}$ is a regular k-dimensional manifold with a covering number $(\frac{C_{\mathcal{X}}}{\gamma/2})^k$; 4) within each covering ball B of $\mathcal{X}$ that contains $\boldsymbol{x} \in S_m^{(x)}$, $\boldsymbol{x} - \boldsymbol{x}' \in \mathcal{X} - \mathcal{N}(\boldsymbol{T}_{|B}) \; \forall \; \boldsymbol{x}, \boldsymbol{x}' \in B$. Then, for any $\nu > 0$, with probability at least $1 - \nu$ we have*

$$GE(f_{S_m}) \leq \max\{A(\gamma + \delta), PE(\delta)\} \tag{5}$$

$$+ M\sqrt{\frac{\log(2)2^{k+1}C_{\mathcal{X}}^k}{\gamma^k m} + \frac{2\log(1/\nu)}{m}}, \tag{6}$$

where $\delta = 2b\max(|\prod_{i=1}^{L}\sigma_{\max}^i - 1|, |\prod_{i=1}^{L}\sigma_{\min}^i - 1|)$, and γ is the same as that of theorem 3.1.

Proof. Similar to the proof of theorem 2.1, we partition the space $\mathcal{Z}$ via the assumed γ-cover. Since $\mathcal{X}$ is a k-dimensional manifold, its covering number is upper bounded by $C_{\mathcal{X}}^k/(\gamma/2)^k$. Let K be the overall number of covering set, which is upper bounded by $C_{\mathcal{X}}^k/(\gamma/2)^k$. Denote C_i the i^{th} covering ball and let N_i be the set of indices of training examples that fall into C_i. Note that $(|N_i|)_{i=1,\ldots,K}$ is an IDD multimonial random variable with parameters m and $(|\mu(C_i)|)_{i=1,\ldots,K}$. Then

$$\begin{aligned}
&|R(f\boldsymbol{T}) - R_m(f\boldsymbol{T})| \\
=&|\sum_{i=1}^{K}\mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{Tx}), y)|z \in C_i]\mu(C_i) - \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(f(\boldsymbol{Tx}_i), y_i)| \\
\leq&|\sum_{i=1}^{K}\mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{Tx}), y)|z \in C_i]\frac{|N_i|}{m} - \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(f(\boldsymbol{Tx}_i), y_i)| \\
&+|\sum_{i=1}^{K}\mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{Tx}), y)|z \in C_i]\mu(C_i) \\
&\qquad - \sum_{i=1}^{K}\mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{Tx}), y)|z \in C_i]\frac{|N_i|}{m}| \\
\leq&|\frac{1}{m}\sum_{i=1}^{K}\sum_{j\in N_i}\max_{z'\in C_i, \boldsymbol{x}'-\boldsymbol{x}_j\in\mathcal{P}_{\boldsymbol{x}_j}}|\mathcal{L}(f(\boldsymbol{Tx}'), y') - \mathcal{L}(f(\boldsymbol{Tx}_j), y_j)|| \qquad (7) \\
&+|\max_{z\in\mathcal{Z}}|\mathcal{L}(f(\boldsymbol{Tx}), y)|\sum_{i=1}^{K}|\frac{|N_i|}{m} - \mu(C_i)||. \qquad (8)
\end{aligned}$$

Remember that $z = (\boldsymbol{x}, y)$. We consider the two cases of $y' = y_j$ and $y' \neq y_j$ to bound eq. (7).

When $y' = y_j$, by the assumption that $\boldsymbol{T}$ is of γ-cover δ-isometry w.r.t. $\mathcal{P}_{\boldsymbol{x}}$ of $\boldsymbol{x} \in S_m^{(x)}$ and the Lipschitz constant of $\mathcal{L} \circ f$ w.r.t. $\boldsymbol{Tx}$ is A, suppose the maximum is achieved at $\boldsymbol{x}_k$ and $\boldsymbol{x}_k \in C_p$, we have

$$\begin{aligned}
&\max_{z'\in C_p, \boldsymbol{x}'-\boldsymbol{x}_k\in\mathcal{P}_{\boldsymbol{x}_k}}|\mathcal{L}(f(\boldsymbol{Tx}'), y') - \mathcal{L}(f(\boldsymbol{Tx}_k), y_k)| \\
\leq&A\max_{z'\in C_p, \boldsymbol{x}'-\boldsymbol{x}_k\in\mathcal{P}_{\boldsymbol{x}_k}}||\boldsymbol{T}_{|\boldsymbol{x}_k}(\boldsymbol{x}' - \boldsymbol{x}_k)|| \qquad (9) \\
\leq&A\max_{z'\in C_p, \boldsymbol{x}'-\boldsymbol{x}_k\in\mathcal{P}_{\boldsymbol{x}_k}}(||\boldsymbol{x}' - \boldsymbol{x}_k|| + 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|) \qquad (10) \\
\leq&A(\gamma + 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|) \\
\leq&A(\gamma + 2b\max(|\prod_{i=1}^{L}\sigma_{\max}^i - 1|, |\prod_{i=1}^{L}\sigma_{\min}^i - 1|)),
\end{aligned}$$

where the second inequality holds since the γ-cover δ-isometry of $\boldsymbol{T}$ also gives $\rho_Q(\boldsymbol{Tx}, \boldsymbol{Tx}') \leq \rho_P(\boldsymbol{x}, \boldsymbol{x}') + 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|$.

When $y' \neq y_j$, given any training $\boldsymbol{x}_k$, we have

$$\begin{aligned}
&\max_{z'\in C_p, \boldsymbol{x}'-\boldsymbol{x}_k\in\mathcal{P}_{\boldsymbol{x}_k}}|\mathcal{L}(f(\boldsymbol{Tx}'), y') - \mathcal{L}(f(\boldsymbol{Tx}_k), y_k)| \\
=&\max_{z'\in D_{z_k}}|\mathcal{L}(f(\boldsymbol{Tx}'), y') - \mathcal{L}(f(\boldsymbol{Tx}_k), y_k)| \qquad (11) \\
\leq&\text{PE}(2b\max(|\prod_{i=1}^{L}\sigma_{\max}^i - 1|, |\prod_{i=1}^{L}\sigma_{\min}^i - 1|)),
\end{aligned}$$

where the inequality holds since by γ-cover δ-isometry of $\boldsymbol{T}$, we have $\rho_Q(\boldsymbol{Tx}, \boldsymbol{Tx}') \leq \rho_P(\boldsymbol{x}, \boldsymbol{x}') + \delta$ and $\rho_Q(\boldsymbol{Tx}, \boldsymbol{Tx}') \geq \rho_P(\boldsymbol{x}, \boldsymbol{x}') - \delta$, with

$$\delta = 2b\max(|\prod_{i=1}^{L}\sigma_{\max}^i - 1|, |\prod_{i=1}^{L}\sigma_{\min}^i - 1|).$$

Thus eq. (7) is less than or equal to $\max\{A(\gamma+\delta), \text{PE}(\delta)\}$. By Breteganolle-Huber-Carol inequality, eq. (8) is less than or equal to $M\sqrt{\frac{\log(2)2^{k+1}C_{\mathcal{X}}^k}{\gamma^k m} + \frac{2\log(1/\nu)}{m}}$.

The proof is finished. □

We now specify a case where the obtained bound in theorem 3.2 is tighter than that in theorem 3.1; for example, in the ball that covers $(\boldsymbol{x}_k, y_k)$, few examples with $y \neq y_k$ are misclassified. In this case, $\text{PE}(2b\max(|\prod_{i=1}^{L}\sigma_{\max}^i - 1|, |\prod_{i=1}^{L}\sigma_{\min}^i - 1|) \leq A(\gamma + 2b|\prod_{i=1}^{L}\sigma_{\max}^i - 1|)$, and the covering number is shrunken by a factor of $\sqrt{|\mathcal{Y}|}$. Our result only incrementally improves the generalization bound. However, it clearly shows that the contraction property of isometric mapping plays an important role in bounding the generalization error.

3.3 Suggestion of new algorithms

Many quantities exist in the GE bound established in theorem 3.2. Except γ and δ, all others are independent of the neural network. Although both γ and δ are controlled by singular values of weight matrices, we note that γ, which specifies the size of covering balls for a covering of $\mathcal{X}$, is more of a trade-off parameter that balances between the first and second term of the GE bound, than of a variable used to control GE, as long as its values satisfy the condition of $\gamma \leq o(S_m, \boldsymbol{T})/\left(\prod_{i=1}^{l(S_m,\boldsymbol{T})}\sigma_{\max}^i\right)$ established in lemma 3.3.

To control the bound via δ, we note that the minimum value of the bound w.r.t. δ is achieved when $\delta = 0$, which implies $\prod_{i=1}^{L} \sigma_{\max}^i = 1$ and $\prod_{i=1}^{L} \sigma_{\min}^i = 1$. In the following lemma, we show that the condition is achieved only when $\sigma_{\max}^i = \sigma_{\min}^i, \forall i = 1, \ldots, L$.

Lemma 3.5. *In theorem 3.2, $\delta = 0$ is achieved only when*

$$\sigma_{\max}^i = \sigma_{\min}^i, \ \forall i = 1, \ldots, L,$$

$$\prod_{i=1}^{L} \sigma_{\max}^i = 1, \prod_{i=1}^{L} \sigma_{\min}^i = 1.$$

Proof. It is straightforward to see that $\delta = 0$ i.f.f. $\prod_{i=1}^{L} \sigma_{\max}^i = 1$ and $\prod_{i=1}^{L} \sigma_{\min}^i = 1$. In the following, we show the two conditions hold only when $\sigma_{\max}^i = \sigma_{\min}^i$, $\forall i = 1, \ldots, L$.

For any $i \in \{1, \ldots, L\}$, we reparameterize $\sigma_{\min}^i$ as $\sigma_{\min}^i = \alpha_i \sigma_{\max}^i$. It is clear that $\alpha_i \in (0, 1]$.

Since $\prod_{i=1}^{L} \sigma_{\min}^i = \prod_{i=1}^{L} \sigma_{\max}^i = 1$, we have

$$1 = \prod_{i=1}^{L} \sigma_{\min}^i = \prod_{i=1}^{L} \alpha_i \sigma_{\max}^i = \prod_{i=1}^{L} \alpha_i. \tag{12}$$

Notice that $\alpha \in (0, 1]$, thus, we have $0 < \prod_{i=1}^{L} \alpha_i \leq 1$. To have eq. (12), we need $\alpha_i = 1, \forall i = 1, \ldots, L$, indicating $\sigma_{\min}^i = \sigma_{\max}^i, \forall i = 1, \ldots, L$. □

We show in lemma 3.5 that the optimal GE bound of theorem 3.2 w.r.t. δ is achieved only when all singular values of each of weight matrices of a DNN are equal. Among various solutions, the most straightforward one is that all singular values are equal to 1; in other words, each weight matrix has orthonormal rows or columns. This inspires a new set of algorithms that we generally term as *Orthogonal Deep Neural Networks (OrthDNNs)*.

4 Algorithms of Orthogonal Deep Neural Networks

In this section, we first present the algorithm of *strict OrthDNNs* by enforcing strict orthogonality of weight matrices during network training. It amounts to optimizing weight matrices on their respective Stiefel manifolds, which however, is computationally prohibitive for large-sized networks. To achieve efficient OrthDNNs, we propose a novel algorithm called Singular Value Bounding (SVB), which achieves *approximate OrthDNNs* via a simple scheme of hard regularization. We discuss alternative schemes of soft regularization for approximate OrthDNNs, and compare with our proposed SVB. Batch Normalization [29] is commonly used to accelerate training of modern DNNs, yet it has a potential risk of ill-conditioned layer transform, causing its incompatibility with OrthDNNs. In fact, direct use of BN in OrthDNNs makes it ineffective to enforce strict orthogonality of weight matrices. We propose Degenerate Batch Normalization (DBN) to enable its use with strict OrthDNNs. We also propose Bounded Batch Normalization (BBN) to remove the potential risk of ill-conditioned layer transform. We finally explain how OrthDNNs are used for convolutional kernels.

Denote parameters of a DNN collectively as $\Theta = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L}$, where $\{b_l\}_{l=1}^{L}$ are bias terms. We discuss algorithms of strict or approximate OrthDNNs in the following context. Given a training set $\{\boldsymbol{x}_i, y_i\}_{i=1}^{m}$, we write the training objective as $\mathcal{L}\left(\{\boldsymbol{x}_i, y_i\}_{i=1}^{m}; \Theta\right)$. Training is based on SGD (or its variants [52]), which updates Θ via a simple rule of $\Theta^{t+1} \leftarrow \Theta^t - \eta \frac{\partial \mathcal{L}}{\partial \Theta^t}$, where η is the learning rate, and the gradient $\frac{\partial \mathcal{L}}{\partial \Theta^t}$ is usually computed from a mini-batch of training examples. Network training proceeds by sampling for each iteration t a mini-batch from $\{\boldsymbol{x}_i, y_i\}_{i=1}^{m}$, until a specified number of iterations or the training loss plateaus.

4.1 The case of strict orthogonality

Enforcing orthogonality of weight matrices during network training amounts to solving the following constrained optimization problem

$$\begin{aligned} \min_{\Theta = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L}} & \ \mathcal{L}\left(\{\boldsymbol{x}_i, y_i\}_{i=1}^{m}; \Theta\right) \\ \text{s.t. } & \boldsymbol{W}_l \in \mathcal{O} \ \forall \ l \in \{1, \ldots, L\}, \end{aligned} \tag{13}$$

where $\mathcal{O}$ stands for the set of matrices whose row or column vectors are orthonormal. For $\boldsymbol{W}_l$ of any l^{th} layer, problem (13) in fact constrains its solution set as a Riemannian manifold called Stiefel manifold, which is defined as $\mathcal{M}_l = \{\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}} | \boldsymbol{W}_l^\top \boldsymbol{W}_l = \boldsymbol{I}\}$ assuming $n_l \geq n_{l-1}$, and is an embedded submanifold of the space $\mathbb{R}^{n_l \times n_{l-1}}$, where $\boldsymbol{I}$ is an identity matrix. In literature, optimization of a differentiable cost function on such a matrix manifold and its convergence analysis have been intensively studied [1], [9]. For completeness, we briefly present the solving algorithm of (13) as follows.

Denote $T_{\boldsymbol{W}_l}\mathcal{M}_l$ as the tangent space to $\mathcal{M}_l$ at the current $\boldsymbol{W}_l \in \mathcal{M}_l$. First-order methods such as SGD first find a tangent vector $\Omega_{\boldsymbol{W}_l} \in T_{\boldsymbol{W}_l}\mathcal{M}_l$ that describes the steepest descent direction for the cost, and update $\boldsymbol{W}_l$ as $\boldsymbol{W}_l - \eta\Omega_{\boldsymbol{W}_l}$ with the step size η that satisfies conditions of convergence, and then perform a retraction $\mathcal{R}_{\boldsymbol{W}_l}(-\eta\Omega_{\boldsymbol{W}_l})$ that defines a mapping from the tangent space to the Stiefel manifold, which can be achieved by $\mathcal{R}_{\boldsymbol{W}_l}(-\eta\Omega_{\boldsymbol{W}_l}) = \mathcal{Q}(\boldsymbol{W}_l - \eta\Omega_{\boldsymbol{W}_l})$, where the operator $\mathcal{Q}$ denotes the Q factor of QR matrix decomposition. To obtain the tangent vector $\Omega_{\boldsymbol{W}_l}$, one may project the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l}$ in the embedding space $\mathbb{R}^{n_l \times n_{l-1}}$ (or its momentum version [52]) onto the tangent space $T_{\boldsymbol{W}_l}\mathcal{M}_l$ by $\mathcal{P}_{\boldsymbol{W}_l}\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l}$, where $\mathcal{P}_{\boldsymbol{W}_l}$ defines a projection operator according to the local geometry of $\boldsymbol{W}_l \in \mathcal{M}_l$. Convergence analysis for such a scheme to obtain the tangent vector is presented in [45]. In Appendix F, we present the algorithmic details for optimization of weight matrices on the Stiefel manifolds.

4.2 Achieving near orthogonality via Singular Value Bounding

Constraining solutions of weight matrices of a DNN on their Stiefel manifolds is an interesting direction of research. It also supports analysis of theoretical properties as in section 3 and the related works [45], [48]. However, it arguably has the following shortcomings concerned with computation, empirical performance, and also compatibility with existing deep learning methods, which motivate us to

address these shortcomings by developing new algorithms of approximate OrthDNNs.

- Strict constraining of weight matrices on the Stiefel manifolds requires expensive computations — in particular, the operations of projecting the Euclidean gradient onto the tangent space and retraction onto the Stiefel manifold (Steps 2 and 4 in Appendix F) dominate the costs in each iteration. If we allow the solutions slightly away from the manifolds, the expensive projection and retraction operations are not necessary to be performed in each iteration. Instead, similar pulling-back operations can be performed less frequently, e.g., in every a certain number of iterations, and consequently such a burden of pulling back is amortized.
- Theorem 3.2 gives a bound $GE(f_{S_m})$ of the expected error $R(f_{S_m})$ w.r.t the training error $R_m(f_{S_m})$. To achieve good performance on practical problems, both $R_m(f_{S_m})$ and $GE(f_{S_m})$ should be small. However, optimization of DNNs is characterized by proliferation of local optima/critical points [14], [31]. When we are motivated to optimize weight matrices on their Stiefel manifolds, obtaining $\boldsymbol{W}_l \in \mathcal{M}_l$, $l \in \{1, \ldots, L\}$, with $\Omega_{\boldsymbol{W}_l} = 0$, it is very likely that for a $\boldsymbol{W}_l$, there exists a better local optimum in the embedding Euclidean space that is slightly away from the manifold (e.g., $\Omega_{\boldsymbol{W}_l} = 0$ while $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l} \neq 0$, or the Euclidean gradient is in the complement null space of the current tangent space), and has a smaller $R_m(f_{S_m})$. If we allow the optimization to step away from, but still pivot around, the manifold, better solutions could be obtained by escaping from local optima on the manifold.
- Successful training of modern DNNs depends heavily on BN [29], a technique that can greatly improve training convergence and empirical results. However, as analyzed shortly in section 4.4, BN would change the spectrum of singular values of each layer transform (i.e., the combined linear transform of each layer achieved by weight mapping and BN, as specified in (17)). Consequently, the efforts spending on enforcing strict orthogonality of weight matrices become ineffectual. Algorithms of approximate OrthDNNs seem more compatible with BN transform.

To develop an algorithm of approximate OrthDNNs, we propose a simple yet effective network training method called Singular Value Bounding (SVB). SVB is a sort of projected SGD method and can be summarized as follows: SVB simply bounds, after every T_{svb} iterations of SGD training, all the singular values of each $\boldsymbol{W}_l$, for $l = 1, \ldots, L$, in a narrow band $[1/(1+\epsilon), (1+\epsilon)]$ around the value of 1, where $\epsilon \geq 0$ is a specified small constant. Algorithm 1 presents the details.

After each bounding step, optimization of SVB in fact proceeds in the embedding Euclidean space, to search for potentially better solutions, before next bounding step that pulls the solutions back onto ($\epsilon = 0$) or near ($\epsilon > 0$) the Stiefel manifolds. With annealed learning rate schedules, we observe empirical convergence of SVB. Compared with manifold optimization in section 4.1, SVB is more efficient

Algorithm 1: Singular Value Bounding

input : A network of L layers with trainable parameters $\Theta = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^L$, training loss $\mathcal{L}$, learning rate η, the maximal number T of training iterations, a specified number T_{svb} of iteration steps, a small constant ϵ

```
Initialize Θ such that W_l^⊤ W_l = I or W_l W_l^⊤ = I for l = 1, ..., L
for t = 0, ..., T − 1 do
    Update Θ^{t+1} ← Θ^t − η ∂L/∂Θ^t using SGD based methods
    while training proceeds for every T_svb iterations do
        for l = 1, ..., L do
            Perform [U_l, Σ_l, V_l] = svd(W_l)
            Let {σ_i^l}_{i=1}^{n_l} be the diagonal entries of Σ_l
            for i = 1, ..., n_l do
                σ_i^l = 1 + ε  if  σ_i^l > 1 + ε
                σ_i^l = 1/(1 + ε)  if  σ_i^l < 1/(1 + ε)
            end
            Update W_l ← U_l Σ_l V_l^⊤ with the bounded
              diagonal entries {σ_i^l}_{i=1}^{n_l} of Σ_l
        end
    end
end
```

output: Trained network with parameters Θ^T for inference

since the dominating computation of SVD is invoked only every a certain number of iterations. Experiments of image classification in section 5 show that SVB sometimes outperforms the algorithm of strict OrthDNNs in section 4.1, both of which outperform the commonly used SGD based methods, and in many cases with a large margin.

4.3 Alternative algorithms for approximate OrthDNNs

To achieve approximate OrthDNNs, one may alternatively penalize the main objective $\mathcal{L}\left(\{\boldsymbol{x}_i, y_i\}_{i=1}^m; \Theta\right)$ with an augmented term that encourages orthonormality of columns or rows of weight matrices, resulting in the following unconstrained optimization problem

$$\min_{\Theta=\{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^L} \mathcal{L}\left(\{\boldsymbol{x}_i, y_i\}_{i=1}^m; \Theta\right) + \lambda \sum_{l=1}^{L} \|\boldsymbol{W}_l^\top \boldsymbol{W}_l - \boldsymbol{I}\|_F^2, \quad (14)$$

where $\|\cdot\|_F$ denotes Frobenius norm, λ is the penalty parameter, and we have assumed $n_l \geq n_{l-1}$ for a certain layer l. By using increasingly larger values of λ, the problem (14) approaches to achieve strict OrthDNNs. One can use SGD based methods to solve (14), where the additional computation cost incurred by the regularizer is marginal. Soft regularization of the type (14) is used in the related works [13], [58].

To relax the requirement of $n_l \geq n_{l-1}$ assumed in (14), an algorithm termed Spectral Restricted Isometry Property (SRIP) regularization, which leverages the matrix RIP condition [11], is proposed in [4], whose objective is written as

$$\min_{\Theta=\{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^L} \mathcal{L}\left(\{\boldsymbol{x}_i, y_i\}_{i=1}^m; \Theta\right) + \kappa \sum_{l=1}^{L} \sigma_{\max}(\boldsymbol{W}_l^\top \boldsymbol{W}_l - \boldsymbol{I}), \quad (15)$$

where κ is a penalty parameter, and $\sigma_{\max}(\cdot)$ denotes the spectral norm of a matrix. Although computation of (15) involves expensive eigen-decomposition, it can be efficiently approximated via power iteration method. One may refer to [4] for the solving equation. In this work, we compare the alternative (14) and (15) with our proposed SVB.

4.4 Compatibility with Batch Normalization

We start this section by showing that the original design of Batch Normalization [29] is incompatible with our proposed OrthDNNs. Technically, for a network layer that computes, before the nonlinear activation, $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}^n$, BN inserts a normalization denoted as $\text{BN}(\boldsymbol{h}) = \text{BN}(\boldsymbol{W}\boldsymbol{x})$, where we have ignored the bias term for simplicity. BN in fact applies the following linear transformation to $\boldsymbol{h}$

$$\text{BN}(\boldsymbol{h}) = \boldsymbol{\Upsilon}\boldsymbol{\Phi}(\boldsymbol{h} - \boldsymbol{\mu}) + \boldsymbol{\beta}, \tag{16}$$

where each entry of $\boldsymbol{\mu} \in \mathbb{R}^n$ is the output mean at each of the n neurons of the layer, the diagonal matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$ contains entries $\{1/\phi_i\}_{i=1}^n$ that is the inverse of the neuron-wise output standard deviation ϕ_i (obtained by adding a small constant to the variance for numerical stability), $\boldsymbol{\Upsilon} \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing trainable scalar parameters $\{\upsilon_i\}_{i=1}^n$, and $\boldsymbol{\beta} \in \mathbb{R}^n$ is a trainable bias term. Note that during training, μ and ϕ for each neuron are computed using mini-batch examples, and during inference they are fixed representing the statistics of all the training population, which are usually obtained by running average. Thus the computation (16) for each example is deterministic after network training.

Inserting $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x}$ into (16) we get

$$\text{BN}(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}\boldsymbol{x} + \tilde{\boldsymbol{b}} \text{ s.t. } \widetilde{\boldsymbol{W}} = \boldsymbol{\Upsilon}\boldsymbol{\Phi}\boldsymbol{W} \;\; \tilde{\boldsymbol{b}} = \boldsymbol{\beta} - \boldsymbol{\Upsilon}\boldsymbol{\Phi}\boldsymbol{\mu}, \tag{17}$$

which is simply a standard layer with change of variables. The following lemma suggests that BN is incompatible with OrthDNNs: even though $\boldsymbol{W}$ is enforced to have orthonormal rows or columns, BN would change the conditioning of layer transform, i.e., spectrum of singular values of $\widetilde{\boldsymbol{W}}$, by learning $\boldsymbol{\Upsilon}$ and $\boldsymbol{\Phi}$ whose product does not necessarily contain diagonal entries $\{\upsilon_i/\phi_i\}_{i=1}^n$ of equal value.

Lemma 4.1. *For a matrix $\boldsymbol{W} \in \mathbb{R}^{M\times N}$ with singular values of 1, and a diagonal matrix $\boldsymbol{G} \in \mathbb{R}^{M\times M}$ with nonzero entries $\{g_i\}_{i=1}^M$, let $g_{\max} = \max(|g_1|, \ldots, |g_M|)$ and $g_{\min} = \min(|g_1|, \ldots, |g_M|)$, the singular values of $\widetilde{\boldsymbol{W}} = \boldsymbol{G}\boldsymbol{W}$ is bounded in $[g_{min}, g_{\max}]$. When $\boldsymbol{W}$ is fat, i.e., $M \leq N$, and rank$(\boldsymbol{W}) = M$, singular values of $\widetilde{\boldsymbol{W}}$ are exactly $\{|g_i|\}_{i=1}^M$.*

See the proof in Appendix G.

To make BN compatible with *strict OrthDNNs*, we propose *Degenerate Batch Normalization (DBN)* that learns layer-wise $\bar{\upsilon}$ and $\bar{\phi}$ instead of neuron-wise $\{\upsilon_i\}_{i=1}^n$ and $\{\phi_i\}_{i=1}^n$, so that the learned $\boldsymbol{\Upsilon}$ and $\boldsymbol{\Phi}$ respectively contain diagonal entries of equal value. Such a $\boldsymbol{\Upsilon}$ is learned simply by using a single trainable parameter $\bar{\upsilon}$ shared by all n neurons of the layer. To learn such a $\boldsymbol{\Phi}$, DBN still computes neuron-wise $\{\phi_i\}_{i=1}^n$ in each iteration of training, but uses running average to learn $\bar{\phi} = \frac{1}{n}\sum_{i=1}^n \phi_i$ that would be shared by the n neurons. The proposed DBN enjoys the benefit of neuron-wise normalization in BN, and when training converges, it learns a layer transform $\widetilde{\boldsymbol{W}} = \boldsymbol{\Upsilon}\boldsymbol{\Phi}\boldsymbol{W} = \bar{\upsilon}/\bar{\phi}\boldsymbol{W}$ whose conditioning is the same as that of $\boldsymbol{W}$, thus achieving compatibility with strict OrthDNNs.

Section 4.2 discusses the potential advantages of approximate OrthDNNs over the strict ones. To make BN compatible with *approximate OrthDNNs*, especially with our proposed SVB method, we propose *Bounded Batch Normalization (BBN)* that controls the variations among $\{\upsilon_i/\phi_i\}_{i=1}^n$, so that the conditioning of layer transform is not severely affected by $\boldsymbol{\Upsilon}\boldsymbol{\Phi}$. More specifically, BBN computes $\alpha = \frac{1}{n}\sum_{i=1}^n \upsilon_i/\phi_i$ in each iteration of training, and bounds each of $\{\frac{1}{\alpha}\upsilon_i/\phi_i\}_{i=1}^n$ in a narrow band $[1/(1+\tilde{\epsilon}), (1+\tilde{\epsilon})]$ around the value of 1, where $\tilde{\epsilon} \geq 0$ is a scalar parameter. Algorithm 2 presents details of the proposed BBN.

Algorithm 2: Bounded Batch Normalization

input : A network with L BN layers, trainable parameters $\{\boldsymbol{\Upsilon}_l^t\}_{l=1}^L$, $\{\boldsymbol{\beta}_l^t\}_{l=1}^L$, and statistics $\{\boldsymbol{\mu}_l^t\}_{l=1}^L$, $\{\boldsymbol{\Phi}_l^t\}_{l=1}^L$ of BN layers at iteration t, a small constant $\tilde{\epsilon}$

Update to get $\{\boldsymbol{\Upsilon}_l^{t+1}\}_{l=1}^L$ from $\{\boldsymbol{\Upsilon}_l^t\}_{l=1}^L$ (and $\{\boldsymbol{\beta}_l^{t+1}\}_{l=1}^L$ from $\{\boldsymbol{\beta}_l^t\}_{l=1}^L$), using SGD based methods

Update to get $\{\boldsymbol{\Phi}_l^{t+1}\}_{l=1}^L$ from $\{\boldsymbol{\Phi}_l^t\}_{l=1}^L$ (and $\{\boldsymbol{\mu}_l^{t+1}\}_{l=1}^L$ from $\{\boldsymbol{\mu}_l^t\}_{l=1}^L$), using running average over statistics of mini-batch examples

for $l = 1, \ldots, L$ **do**

  Let $\{\upsilon_i\}_{i=1}^{n_l}$ and $\{1/\phi_i\}_{i=1}^{n_l}$ be respectively the diagonal entries of $\boldsymbol{\Upsilon}_l^{t+1}$ and $\boldsymbol{\Phi}_l^{t+1}$

  Let $\alpha = \frac{1}{n_l}\sum_{i=1}^{n_l} \upsilon_i/\phi_i$

  for $i = 1, \ldots, n_l$ **do**

    $\upsilon_i = \alpha\phi_i(1+\tilde{\epsilon})$ if $\frac{1}{\alpha}\upsilon_i/\phi_i > 1+\tilde{\epsilon}$

    $\upsilon_i = \alpha\phi_i/(1+\tilde{\epsilon})$ if $\frac{1}{\alpha}\upsilon_i/\phi_i < 1/(1+\tilde{\epsilon})$

  end

end

output: Updated BN parameters and statistics at iteration $t+1$

The introduction of DBN makes it possible to empirically compare strict and approximate OrthDNNs in the context of modern architectures, which is presented in section 5. Experiments in section 5 also show that performance is improved when using BBN instead of BN, confirming the benefit by resolving BN's compatibility with OrthDNNs.

4.5 Orthogonal Convolutional Neural Networks

In previous sections, we present theories and algorithms of OrthDNNs by writing their layer-wise weights in matrix forms. When applying DNNs to image data, one is actually using networks with convolutional layers. For an l^{th} convolutional layer with weight tensor of the size $n_l \times n_{l-1} \times n_h \times n_w$, where n_h and n_w denote the height and width of the convolutional kernel, we choose to convert the tensor as a matrix of the size $n_l \times n_{l-1}n_hn_w$ based on the following rational. Natural images are usually modeled by first learning filters from (densely overlapped) local patches, and then applying the thus learned filters to images to aggregate the corresponding local statistics. The convolutional layer in fact linearly transforms the n_{l-1} input feature maps in the same way, by applying each of n_l filters of the size $n_{l-1} \times n_h \times n_w$ to n_{l-1} patches of the size $n_h \times n_w$ in a sliding window fashion, resulting in local responses that are arrayed in the form of n_l feature maps, which have the same size as that of input feature maps when padding the boundaries. In other words, the convolutional layer applies linear transformation, using n_l filters, to $n_{l-1}n_hn_w$-dimensional instances that are collected from local patches of input feature maps. Correspondingly, we choose to convert the weight tensor containing the n_l filters of dimension $n_{l-1}n_hn_w$ to its matrix form, and apply to it specific algorithms of OrthDNNs.

However, we note that our way of forming the weight matrices does not exactly specify linear transformations of convolutional layers. For the l^{th} layer, its exact weight

matrix is in fact a function of the kernel tensor and contains doubly block circulant submatrices [49]. Our preliminary experiments show that OrthDNNs based on such forms are effective to regularize network training as well. We would conduct further investigations with additional experiments in future research.

5 Experiments

We present in this section extensive experiments of image classification to verify the efficacy of OrthDNNs. We are particularly interested in how algorithms of strict or approximate OrthDNNs provide regularization to various architectures of modern DNNs, such as ConvNets [38], [50], ResNets [22], [23], DenseNet [27], and ResNeXt [59]. We use the benchmark datasets of CIFAR10, CIFAR100 [36], and ImageNet [47] for these experiments. We compare empirical performance and efficiency among different algorithms of strict and approximate OrthDNNs. For some of these comparisons, we also investigate behaviours of OrthDNNs under regimes of both small and large sizes of training samples, and robustness of OrthDNNs against corruptions that are commonly encountered in natural images, in order to better understand the empirical strength of OrthDNNs. For network training, we use SGD with momentum and initialize networks using orthogonal weight matrices, where the momentum is set as 0.9 with a weight decay of 0.0001. When our proposed SVB is turned on, we apply it to weight matrices of all layers after every epoch of training.

5.1 Comparative studies on algorithms of strict and approximate OrthDNNs

In this section, we use architectures of ConvNet and ResNet on CIFAR10 to study the behaviours of strict and approximate OrthDNNs. The CIFAR10 dataset consists of $60,000$ 32×32 color images of 10 object categories ($50,000$ training and $10,000$ testing ones). We use raw images without preprocessing. Data augmentation follows the standard manner in [40]: during training, we zero-pad 4 pixels along each image side, and sample a 32×32 region crop from the padded image or its horizontal flip; during testing, we use the original non-padded image. Our ConvNet architectures follow [22], [50]. Each network starts with a conv layer of 16 3×3 filters, and then sequentially stacks three types of $2X$ conv layers of 3×3 filters, each of which has the feature map sizes of 32, 16, and 8, and filter numbers of 16, 32, and 64, respectively; spatial sub-sampling of feature maps is achieved by conv layers of stride 2; the network ends with a global average pooling and a fully-connected layer. The ResNet construction is based on the ConvNets presented above, where we use an "identity shortcut" to connect every two conv layers of 3×3 filters and use a "projection shortcut" when sub-sampling of feature maps is needed; we adopt the pre-activation version [23]. Thus, for both types of networks, we have $6X + 2$ weight layers in total. We set $X = 3$ for experiments in this section, giving networks of 20 weight layers.

The DBN proposed in section 4.4 is designed to be compatible with strict OrthDNNs. We use DBN to enable training and comparison of strict and approximate OrthDNNs on the networks constructed above. We implement strict OrthDNNs as the algorithm presented in section 4.1. We use our proposed SVB and those in [4], [13], [58] (i.e., the problems (14) and (15)) for approximate OrthDNNs. The learning rates start at 0.1 and end at 0.001, and decay every two epochs until the end of 160 epochs of training, where we set the mini-batch size as 128. We fix the parameter ϵ of SVB as 0.05, while both λ of soft regularization in (14) and κ of SRIP in (15) are optimally tuned as 0.1.

TABLE 1
Comparison of strict and approximate OrthDNNs on the CIFAR10 dataset [36], using ConvNet and ResNet architectures respectively of 20 weight layers (referring to the main text for their specifics). Degenerate batch normalization is used due to its compatibility with strict OrthDNNs.

Network	Training method	Error rate (%)	Averaged time per iter. (sec.)
ConvNet	SGD with momentum	16.68	0.0702
	Strict OrthDNNs via Manifold Opt.	10.85	0.2034
	Approx. OrthDNNs via Soft Regu.	10.39	0.0930
	Approx. OrthDNNs via SRIP	10.67	0.0760
	Approx. OrthDNNs via SVB	11.41	0.0718
ResNet	SGD with momentum	10.27	0.0754
	Strict OrthDNNs via Manifold Opt.	9.03	0.2042
	Approx. OrthDNNs via Soft Regu.	8.91	0.0844
	Approx. OrthDNNs via SRIP	8.72	0.0832
	Approx. OrthDNNs via SVB	8.76	0.0768

Table 1 gives the results with the curves of training convergence plotted in fig. 3. Table 1 shows that on both of the two networks, algorithms of strict and approximate OrthDNNs outperform standard SGD based method, confirming the improved generalization by their regularization of network training. Moreover, approximate OrthDNNs via either SVB, soft regularization, or SRIP perform as well as strict ones, but at a much lower computational cost [2], suggesting their advantage in practical use. Due to the prohibitive computation of strict OrthDNNs on modern architectures of larger sizes, we choose to use approximate OrthDNNs in subsequent experiments, and correspondingly use BN or our proposed BBN to replace DBN.

5.2 Comparison of hard and soft regularization for approximate OrthDNNs

In this section, we study algorithms of approximate OrthDNNs by comparing our proposed SVB with soft reg-

2. In table 1, the respective dominating computations of QR decomposition for manifold optimization and singular value decomposition for SVB are based on CUDA implementation.

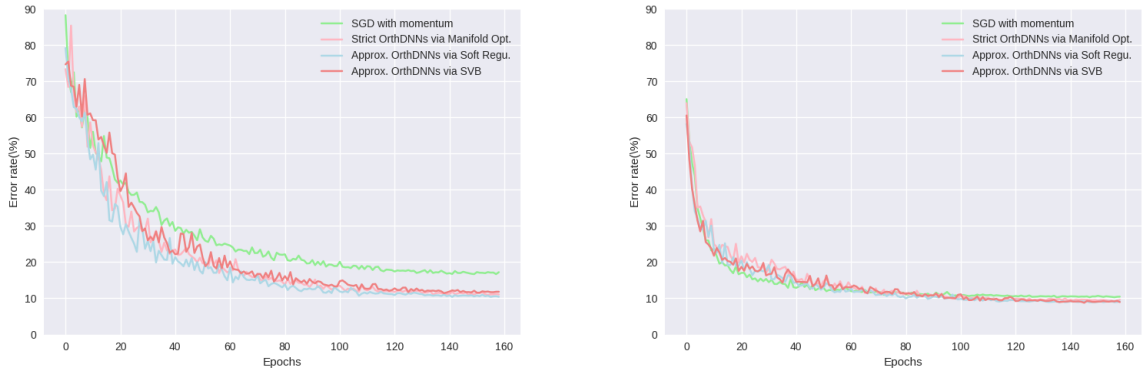

Fig. 3. Validation curves of strict and approximate OrthDNNs on the CIFAR10 dataset [36] using architectures of a ConvNet (left) and a ResNet (right) respectively of 20 weight layers.

TABLE 2
Comparison of approximate OrthDNNs via hard and soft regularization on the CIFAR10 dataset [36], using a ResNet of 68 weight layers. Each setting is run for five times, and results are reported in the format of best (mean $\pm$ standard deviation).

Training method	Error rate (%)
SGD with momentum + BN	6.25 (6.43 $\pm$ 0.15)
Soft Regularization + BN	6.12 (6.28 $\pm$ 0.12)
SRIP + BN	5.86 (5.95 $\pm$ 0.08)
SVB + BN	5.84 (5.96 $\pm$ 0.17)
Soft Regularization + BBN	6.22 (6.30 $\pm$ 0.07)
SRIP + BBN	5.99 (6.10 $\pm$ 0.11)
SVB + BBN	5.79 (5.88 $\pm$ 0.07)

ularization [13], [58] and SRIP [4]. The experiments are conducted on CIFAR10 using a pre-activation version of ResNet constructed in the same way as in section 5.1. Setting $X = 11$ gives a total of 68 weight layers. To train the network, we use learning rates that start at 0.5 and end at 0.001, and decay every two epochs until the end of 160 epochs of training, where we set the mini-batch size as 128. Comparison is made with the baseline of standard SGD with momentum. We also switch BBN on or off to verify its effectiveness. We fix ϵ of SVB as 0.5, while the penalty λ of soft regularization and κ of SRIP are optimally tuned as 0.005 and 0.01 respectively. We fix $\tilde{\epsilon}$ of BBN as 0.2. We run each setting of experiments for five times, and report results in the format of best (mean $\pm$ standard deviation).

Table 2 shows that approximate OrthDNNs via SVB, soft regularization, and SRIP provide effective regularization to network training, and SVB and SRIP outperform soft regularization with a noticeable margin. Compared with BN, our proposed BBN can better regularize training and give slightly improved performance. Note that algorithmic design of BBN may not be compatible with soft regularization and RRIP, which explains the degraded performance when using them together.

5.3 Experiments with Modern Architectures

In this section, we investigate how our proposed SVB and BBN methods provide regularization to modern architectures of ResNet [23], Wide ResNet [62], DenseNet [27], and ResNeXt [59]. We use CIFAR10, CIFAR100 [36], and ImageNet [47] for these experiments. The CIFAR100 dataset has the same number of 32×32 color images as CIFAR10 does, but it has 100 object categories where each category contains one-tenth images of those of CIFAR10. We use data augmentation in the same way as for CIFAR10. The ImageNet dataset contains 1.28 million images of $1,000$ categories for training, and $50,000$ images for validation. We use data augmentation as in [59].

For experiments on CIFAR10 and CIFAR100, we use the following specific architectures. ResNet is constructed in the same way as in section 5.1; we set $X = 9$ here giving a total of 56 weight layers. Wide ResNet is the same as "WRN-28-10" in [62]. ResNeXt is the same as "ResNeXt-29 (16×64d)" in [59], i.e., the depth $L = 29$, cardinality $C = 16$, and the feature width in each cardinal branch $d = 64$. We use consistent hyper-parameters to train these architectures. The learning rates start at 0.5 and end at 0.001, and decay every two epochs until the end of 300 epochs of training, where we set the mini-batch size as 64 — note that this schedule with more training epochs and smaller mini-batch size usually gives better empirical performance than the training schedule used in section 5.2 does. We fix ϵ and $\tilde{\epsilon}$ of SVB and BBN as 0.5 and 0.2 respectively. Table 3 confirms that SVB and BBN improve generalization of various architectures. We also observe that improvements on CIFAR100 are generally greater than those on CIFAR10, which may be due to the problem nature of smaller sample size for CIFAR100. We will investigate how our methods perform with varying sample sizes more thoroughly in the subsequent section.

For experiments on ImageNet, we use top-performing models of the following architectures: "ResNet-152" of [23], "DenseNet-264" of [27], and "ResNeXt-101 (64×4d)" of [59]. To train these models, we use the same hyper-parameters as respectively reported in these methods. The parameters ϵ and $\tilde{\epsilon}$ of SVB and BBN are fixed as 0.5 and 0.5 respectively. Results in table 4 confirm that approximate OrthDNNs via our proposed methods improve generalization by providing effective regularization to large-scale learning.

TABLE 3
Error rates (%) on the CIFAR10 and CIFAR100 [36] datasets when applying our proposed SVB and BBN to various modern architectures (referring to the main text for their specifics).

Method	CIFAR10	CIFAR100
ResNet W/O SVB+BBN	5.68	27.71
ResNet WITH SVB+BBN	5.28	26.47
Wide ResNet W/O SVB+BBN	3.78	20.02
Wide ResNet WITH SVB+BBN	3.24	18.75
ResNeXt W/O SVB+BBN	4.12	20.65
ResNeXt WITH SVB+BBN	3.33	16.94

TABLE 4
Error rates (%) on the validation set of ImageNet [47] when applying our proposed SVB and BBN to various modern architectures (referring to the main text for their specifics). Results are based on single-crop testing of the size 320×320.

Method	Top-1 error	Top-5 error
ResNet W/O SVB+BBN	21.00	5.72
ResNet WITH SVB+BBN	20.74	5.35
DenseNet W/O SVB+BBN	22.32	6.33
DenseNet WITH SVB+BBN	21.80	5.83
ResNeXt W/O SVB+BBN	19.39	4.43
ResNeXt WITH SVB+BBN	18.89	4.27

5.4 Effects of Varying Sample Sizes

We are also interested in the efficacy of SVB and BBN for problems with varying sizes of training samples. To this end, we respectively sample $1/10$, $1/5$, $1/2$, or all of training images per category from ImageNet [47], which constitute our ImageNet training subsets of varying sizes. We train the "ResNeXt-101 (64×4d)" model of [59] in the same way as in section 5.3 for this investigation. Fig. 4 shows that SVB and BBN consistently improve classification across the regimes from small to large sizes of training samples, and the improvements are more obvious for the smaller ones.

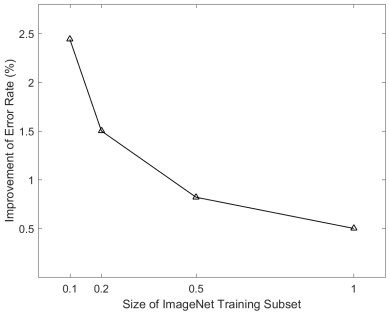

Fig. 4. Regularization effects of our proposed SVB and BBN for varying sizes of training samples. Results in terms of top-1 error rate improvement (%) are obtained by training ResNeXt-101 [59] on ImageNet subsets that are constructed by respectively sampling $1/10$, $1/5$, $1/2$, or all of training images per category from ImageNet. Our methods regularize network training and achieve improved results over the respective baselines of 44.10%, 34.21%, 25.32%, and 19.39%. Results are based on single-crop testing of the size 320×320.

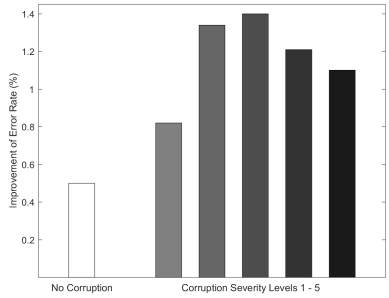

Fig. 5. Robustness test of 5 severity levels on the ImageNet-C dataset [24]. Results in terms of top-1 error rate improvement (%) are obtained by applying ResNeXt-101 [59] models, which are trained without or with regularization of our proposed SVB and BBN, to either clean or corrupted validation images of ImageNet. Our methods give improved results over the respective baselines of 19.39%, 30.56%, 39.26%, 47.09%, 58.50%, and 70.65%.

5.5 Robustness against Common Corruptions

We have shown in previous experiments that OrthDNNs, particularly our proposed SVB and BBN, have better generalization to testing samples that are drawn from the same distributions of training ones. In this section, we investigate the robustness of OrthDNNs when testing samples are corrupted such that they are getting away from the distributions of training ones. We focus on corruptions that are frequently encountered in natural images, e.g., the "common" corruptions of noise, blur, weather, or digitization [24]. Existing research suggests fragility of deep learning models to corruptions of such kinds [16], and that fine-tuning on specific corruption types would help, but cannot well generalize to other types of corruptions [18], [53]. To this end, we use the ImageNet-C dataset [24] that is produced by applying 15 corruption types of 5 severity levels to validation images of ImageNet [47] [3]. As indicated in [24], networks should not be trained or fine-tuned on this dataset for testing of their robustness. We again use the trained models of ResNeXt-101 as described in section 5.3, with or without the regularization of SVB and BBN. Performance improvements of top-1 error rates are plotted in fig. 5, where result for each severity level is an average over the 15 types of corruptions. Compared with the result on clean images of ImageNet validation set, fig. 5 demonstrates better robustness of our proposed methods against common corruptions, and the robustness stands gracefully with the increase of severity levels.

3. The 15 types of corruptions include Gaussian Noise, Shot Noise, Impulse Noise, Defocus Blur, Frosted Glass Blur, Motion Blur, Zoom Blur, Snow, Frost, Fog, Brightness, Contrast, Elastic, Pixelate, and JPEG. Refer to [24] for examples of corrupted ImageNet validation images.

6 Conclusion

In this paper, we present theoretical analysis to connect with the recent interest of spectrally regularized deep learning methods. Technically, we prove a new generalization error bound for DNNs, which is both scale- and range-sensitive to singular value spectrum of each of networks' weight matrices. The bound is established by first proving that DNNs are of local isometry on data distributions of practical

interest, and then introducing the local isometry property of DNNs into a PAC based generalization analysis. We further prove that the optimal bound w.r.t. the degree of isometry is attained when each weight matrix has a spectrum of equal singular values — OrthDNNs with weight matrices of orthonormal rows or columns are thus the most straightforward choice. Based on such analysis, we present algorithms of strict and approximate OrthDNNs, and propose a simple yet effective algorithm called Singular Value Bounding. We also propose Bounded Batch Normalization to make compatible use of batch normalization with OrthDNNs. Experiments on benchmark image classification show the efficacy and robustness of OrthDNNs and our proposed SVB and BNN methods.

REFERENCES

[1] P. A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, USA, 2007. 10, 21

[2] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. *CoRR*, arXiv:1511.06464, 2016. 1

[3] Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems 26*, pages 2814–2822. 2013. 1

[4] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep cnns? In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4266–4276, 2018. 2, 11, 13, 14

[5] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems 31*, pages 4261–4271. 2018. 1, 3

[6] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993. 2

[7] Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6241–6250, 2017. 2

[8] Gary Bécigneul. On the effect of pooling on the geometry of representations. Technical report, 2017. 5

[9] S. Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Autom. Control*, 58(9):2217–2229, 2013. 10, 21

[10] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, March 2002. 2

[11] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Inf. Theor.*, 51(12):4203–4215, December 2005. 11

[12] Djalil Chafaï, Djalil Chafä, Olivier Guédon, Guillaume Lecue, and Alain Pajor. Singular values of random matrices. https://pdfs.semanticscholar.org/37f9/fc9b8cb7a04c7863a0d53c4c3a84a8a7da64.pdf. 19, 20

[13] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, pages 854–863, 2017. 1, 2, 3, 11, 13, 14

[14] Yann N. Dauphin, Razvan Pascanu, Çaglar Gülçehre, KyungHyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 27*, pages 2933–2941, 2014. 1, 11

[15] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028, 2017. 2

[16] Samuel Dodge and Lina Karam. A study and comparison of human and deep learning recognition performance under visual distortions. *arXiv preprint arXiv:1705.02498*, 2017. 15

[17] K. Fukushima. Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. 1

[18] R. Geirhos, D. H. J. Janssen, H. H. Schtt, J. Rauber, M. Bethge, and F. A. Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*, 2017. 15

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010. 1

[20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011. 5

[21] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1225–1234, 2016. 2

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *arXiv prepring arXiv:1506.01497*, 2015. 1, 13

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016. 2, 13, 14

[24] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 15

[25] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. 1

[26] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. 2

[27] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. 1, 2, 13, 14

[28] Jiaji Huang, Qiang Qiu, Guillermo Sapiro, and Robert Calderbank. Discriminative robust transformation learning. In *Advances in Neural Information Processing Systems 28*, pages 1333–1341. 2015. 1, 3, 4, 8

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456, 2015. 1, 2, 10, 11, 12

[30] Kui Jia, Dacheng Tao, Shenghua Gao, and Xiangmin Xu. Improving training of deep neural networks via singular value bounding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4002, 2017. 1, 3

[31] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, 2016. 1, 3, 11

[32] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *CoRR*, abs/1710.05468, 2017. 2, 3

[33] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. 2

[34] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 1

[35] A. N. Kolmogorov and V. Tihomirov. ϵ-entropy and ϵ-capacity of sets in functional spaces. *American Mathematical Society Translations (2)*, 17:227–364, 2002. 4

[36] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Tech. Report*, 2009. 2, 13, 14, 15

[37] Ilja Kuzborskij and Christoph H. Lampert. Data-dependent stability of stochastic gradient descent. *CoRR*, 1703.01678, 2017. 2

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1, 13

[39] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50, 1998. 1

[40] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015. 13

[41] Tongliang Liu, Gábor Lugosi, Gergely Neu, and Dacheng Tao. Algorithmic stability and hypothesis complexity. In *International Conference on Machine Learning*, pages 2159–2167, 2017. 4

[42] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. 4

[43] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014. 3

[44] P. Orlik and H. Terao. *Arrangements of Hyperplanes*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1992. 6

[45] Mete Ozay and Takayuki Okatani. Optimization on submanifolds of convolution kernels in cnns. *CoRR*, abs/1610.07008, 2016. 10, 21

[46] Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in Neural Information Processing Systems*, pages 4788–4798, 2017. 3

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 2, 13, 14, 15

[48] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014. 3, 10

[49] Hanie Sedghi, Vineet Gupta, and Philip M. Long. The singular values of convolutional layers. In *Proceedings of the International Conference on Learning and Representation (ICLR)*, 2019. 13

[50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2, 13

[51] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Trans. Signal Processing*, 65(16):4265–4280, 2017. 1, 2, 3, 6

[52] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1139–1147, May 2013. 10, 21

[53] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760*, 2016. 15

[54] Nakul Verma. Distance Preserving Embeddings for General n-Dimensional Manifolds. *Journal of Machine Learning Research*, 14:2415–2448, 2013. 8

[55] Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems 26*, pages 351–359. 2013. 1

[56] Shengjie Wang, Abdel-rahman Mohamed, Rich Caruana, Jeff A. Bilmes, Matthai Philipose, Matthew Richardson, Krzysztof Geras, Gregor Urban, and Özlem Aslan. Analysis of deep neural networks with extended data jacobian matrix. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 718–726, 2016. 1, 3

[57] Scott Wisdom, Thomas Powers, John R. Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. *CoRR*, arXiv:1611.00035, 2016. 1

[58] Di Xie and Jiang Xiongand Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 11, 13, 14

[59] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5987–5995, 2017. 2, 13, 14, 15

[60] Huan Xu and Shie Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012. 1, 2, 4, 6, 8

[61] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Global optimality conditions for deep neural networks. In *International Conference on Learning Representations*, 2018. 1

[62] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. 2, 14

[63] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. 2

[64] Chiyuan Zhang, Qianli Liao, Alexander Rakhlin, Brando Miranda, Noah Golowich, and Tomaso A. Poggio. Theory of deep learning iib: Optimization properties of SGD. *CoRR*, 1801.02254, 2018. 2

APPENDIX A
PROOF OF LEMMA 3.1.

To prove lemma 3.1, we begin with the following lemma regarding matrix pseudo-inverse.

Lemma A.1. *Given a matrix* $\boldsymbol{W} \in \mathbb{R}^{M\times N}$ *and* $\boldsymbol{x} \in \mathcal{X} - \mathcal{N}(\boldsymbol{W})$, *where* $\mathcal{X}$ *is* $\mathbb{R}^N$, *we have*

$$\boldsymbol{W}^\dagger \boldsymbol{W}\boldsymbol{x} = \boldsymbol{x}$$

where $\boldsymbol{W}^\dagger$ *is the pseudo-inverse of* $\boldsymbol{W}$, *given as* $\boldsymbol{W}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T$ *when* $\boldsymbol{W}$ *has the singular value decomposition* $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, *and* $\boldsymbol{\Sigma}^\dagger$ *is the matrix obtained by first taking the transpose of* $\boldsymbol{\Sigma}$, *and then the inverse of its non-zero elements.*

Proof. Let $\mathcal{R}$ be the index set such that $\boldsymbol{\Sigma}_{rr} \neq 0, \forall r \in \mathcal{R}$. Given any $\boldsymbol{x} \in \mathcal{X} - \mathcal{N}(\boldsymbol{W})$, $\boldsymbol{x}$ can be represented as

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\alpha},$$

where entries of $\boldsymbol{\alpha}$ have $\alpha_r = 0$ when $r \notin \mathcal{R}$. Then

$$\begin{aligned}\boldsymbol{W}^\dagger\boldsymbol{W}\boldsymbol{x} &= \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\alpha}\\ &= \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{\Sigma}\boldsymbol{\alpha}.\end{aligned}$$

Since $\forall\alpha_r \neq 0, \boldsymbol{\Sigma}_{rr} \neq 0, \boldsymbol{\Sigma}^\dagger_{rr} \neq 0$, we have $\boldsymbol{\Sigma}^\dagger\boldsymbol{\Sigma}\boldsymbol{\alpha} = \boldsymbol{\alpha}$, which is to say $\boldsymbol{W}^\dagger\boldsymbol{W}\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\alpha} = \boldsymbol{x}$. □

Now, we are going to prove lemma 3.1.

Proof. For any $\boldsymbol{W}_i$ with $i \in \{1, \ldots, L\}$, performing singular value decomposition (SVD) upon it, we have

$$\boldsymbol{W}_i = \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T,$$

where $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ are both orthogonal matrices.

Given any $\boldsymbol{\Delta} = \boldsymbol{x} - \boldsymbol{x}' \in \mathcal{X} - \mathcal{N}(T)$, let $\boldsymbol{\Delta}_{i-1} = \prod_{j=1}^{i-1}\boldsymbol{W}_j\boldsymbol{\Delta}$, we have

$$\boldsymbol{W}_i\boldsymbol{\Delta}_{i-1} = \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T\boldsymbol{\Delta}_{i-1}.$$

Let $\boldsymbol{\Delta}'_{i-1} = \boldsymbol{V}_i^T\boldsymbol{\Delta}_{i-1}$. We show that if $\boldsymbol{\Delta}'_{i-1,k} \neq 0$, then $\boldsymbol{\Sigma}_{i,kk} \neq 0$, which implies that $\boldsymbol{\Delta}_{i-1}$ lies in the subspace spanned by right singular vectors of $\boldsymbol{W}_i$ that have nonzero singular values, where $\boldsymbol{\Delta}'_{i-1,k}$ and $\boldsymbol{\Sigma}_{i,kk}$ are respectively the k^{th} element of $\boldsymbol{\Delta}'_{i-1}$ and k^{th} diagonal element of $\boldsymbol{\Sigma}_i$.

Suppose otherwise, for a set $\mathcal{K}$, $\boldsymbol{\Delta}'_{i-1,k} \neq 0$ and $\boldsymbol{\Sigma}_{i,kk} = 0, \forall k \in \mathcal{K}$. Let $\boldsymbol{v}_k$ denote the k^{th} column of $\boldsymbol{V}_i$, we reparameterize $\boldsymbol{\Delta}_{i-1}$ as

$$\boldsymbol{\Delta}_{i-1} = \sum_{k\notin\mathcal{K}}\boldsymbol{\Delta}'_{i-1,k}\boldsymbol{v}_k + \sum_{k\in\mathcal{K}}\boldsymbol{\Delta}'_{i-1,k}\boldsymbol{v}_k,$$

where the terms having $\boldsymbol{\Delta}'_{i-1,k} = 0$ are omitted.

Denote $\boldsymbol{W} = \prod_{j=1}^{i-1}\boldsymbol{W}_j$. Since $\boldsymbol{\Delta} \in \mathcal{X} - \mathcal{N}(\boldsymbol{T})$, we have $\boldsymbol{\Delta} \in \mathcal{X} - \mathcal{N}(\boldsymbol{W})$. By lemma A.1, we have $\boldsymbol{W}^\dagger\boldsymbol{\Delta}_{\boldsymbol{i-1}} = \boldsymbol{W}^\dagger\boldsymbol{W}\boldsymbol{\Delta} = \boldsymbol{\Delta} \in \mathcal{X} - \mathcal{N}(\boldsymbol{T})$. Since

$$\begin{aligned}\boldsymbol{W}^\dagger\boldsymbol{\Delta}_{\boldsymbol{i-1}} &= \boldsymbol{W}^\dagger\Big(\sum_{k\notin\mathcal{K}}\boldsymbol{\Delta}'_{i-1,k}\boldsymbol{v}_k + \sum_{k\in\mathcal{K}}\boldsymbol{\Delta}'_{i-1,k}\boldsymbol{v}_k\Big)\\ &= \sum_{k\notin\mathcal{K}}\boldsymbol{\Delta}'_{i-1,k}\boldsymbol{W}^\dagger\boldsymbol{v}_k + \sum_{k\in\mathcal{K}}\boldsymbol{\Delta}'_{i-1,k}\boldsymbol{W}^\dagger\boldsymbol{v}_k,\end{aligned}$$

which is to say

$$\boldsymbol{\Delta} = \sum_{k\notin\mathcal{K}} \boldsymbol{\Delta}'_{i-1,k}\boldsymbol{W}^\dagger\boldsymbol{v}_k + \sum_{k\in\mathcal{K}} \boldsymbol{\Delta}'_{i-1,k}\boldsymbol{W}^\dagger\boldsymbol{v}_k.$$

By assumption we have $\boldsymbol{\Delta}'_{i-1,k} \neq 0$; we also have $\boldsymbol{W}^\dagger\boldsymbol{v}_k \neq \boldsymbol{0}$ — otherwise $\boldsymbol{\Delta}'_{i-1,k}$ would be zero, and $\boldsymbol{W}^\dagger\boldsymbol{v}_k \perp \boldsymbol{W}^\dagger\boldsymbol{v}_{k'}$, for $k \neq k'$; Considering that $\boldsymbol{\Delta} \in \mathcal{X} - \mathcal{N}(\boldsymbol{T})$, we have $\boldsymbol{W}^\dagger\boldsymbol{v}_k \in \mathcal{X} - \mathcal{N}(\boldsymbol{T})$. However, we assume $\boldsymbol{\Sigma}_{i,kk} = 0$ for $k \in \mathcal{K}$, and have

$$\boldsymbol{T}(\sum_{k\in\mathcal{K}} \boldsymbol{\Delta}'_{i-1,k}\boldsymbol{W}^\dagger\boldsymbol{v}_k) = \boldsymbol{0},$$

which implies $\boldsymbol{W}^\dagger\boldsymbol{v}_k \in \mathcal{N}(\boldsymbol{T})$ and leads to a contradiction. We thus prove that if $\boldsymbol{\Delta}'_{i-1,k} \neq 0$, then $\boldsymbol{\Sigma}_{i,kk} \neq 0$.

With the above result, we can constrain the sample variation more precisely through singular values of weight matrices. To be specific, for any pair of $\boldsymbol{x}_{i-1} \in \mathcal{X}_{i-1}$ and $\boldsymbol{x}'_{i-1} \in \mathcal{X}_{i-1}$, we have

$$\begin{aligned}
||\boldsymbol{W}_i\boldsymbol{x}_{i-1} - \boldsymbol{W}_i\boldsymbol{x}'_{i-1}|| &= ||\boldsymbol{W}_i(\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1})|| \\
&= ||\boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T(\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1})|| \\
&= ||\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T(\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1})|| \\
&\geq ||\sigma^i_{\min}\boldsymbol{I}\boldsymbol{V}_i^T(\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1})|| \\
&\geq \sigma^i_{\min}||\boldsymbol{V}_i^T(\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1})|| \\
&= \sigma^i_{\min}||(\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1})||,
\end{aligned}$$

where the second and last equalities use the fact that an orthogonal matrix does not change the norm of operated vectors, and the two inequalities are derived based on our result that for $\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1} \in \boldsymbol{\Delta}_{i-1}$, it lies in the subspace spanned by the right singular vectors of $\boldsymbol{W}_i$ whose corresponding singular values are great than or equal to the nonzero $\sigma^i_{\min}$.

Similarly, we have

$$||\boldsymbol{W}_i\boldsymbol{x}_{i-1} - \boldsymbol{W}_i\boldsymbol{x}'_{i-1}|| \leq \sigma^i_{\max}||(\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1})||.$$

Denote $||\boldsymbol{x}_{i-1} - \boldsymbol{x}'_{i-1}||$ as d_i with $d = ||\boldsymbol{x} - \boldsymbol{x}'||$, we have

$$\sigma^i_{\min}d_i \leq ||\boldsymbol{W}_i\boldsymbol{x}_{i-1} - \boldsymbol{W}_i\boldsymbol{x}'_{i-1}|| \leq \sigma^i_{\max}d_i.$$

Cascading on all layers, we have

$$\begin{aligned}
&\prod_{i=1}^{L}\sigma^i_{\min}d \leq ||\prod_{i=1}^{L}\boldsymbol{W}_i\boldsymbol{x} - \prod_{i=1}^{L}\boldsymbol{W}_i\boldsymbol{x}'|| \leq \prod_{i=1}^{L}\sigma^i_{\max}d \\
\iff &\prod_{i=1}^{L}\sigma^i_{\min}d \leq ||\boldsymbol{T}\boldsymbol{x} - \boldsymbol{T}\boldsymbol{x}'|| \leq \prod_{i=1}^{L}\sigma^i_{\max}d.
\end{aligned}$$

Thus

$$\begin{aligned}
&|||\boldsymbol{T}\boldsymbol{x} - \boldsymbol{T}\boldsymbol{x}'|| - ||\boldsymbol{x} - \boldsymbol{x}'|| | \\
\leq &\max(|\prod_{i=1}^{L}\sigma^i_{\max}d - d|, |\prod_{i=1}^{L}\sigma^i_{\min}d - d|) \\
\leq &\max(|\prod_{i=1}^{L}\sigma^i_{\max} - 1|2b, |\prod_{i=1}^{L}\sigma^i_{\min} - 1|2b).
\end{aligned}$$

We conclude the proof by showing $\boldsymbol{T}$ is of $2b\max(|\prod_{i=1}^{L}\sigma^i_{\max} - 1|, |\prod_{i=1}^{L}\sigma^i_{\min} - 1|)$-isometry. □

APPENDIX B
PROOF OF LEMMA 3.2.

Proof. We proceed by induction on layer l.

For $l = 1$, each row in $\boldsymbol{W}_l$ corresponds to a hyperplane in $\mathcal{X}$. Thus, $\boldsymbol{W}_l$ imposes a hyperplane arrangement $\mathcal{A} = \{\boldsymbol{W}_{l,i}\}_{i=1,\dots,n_l}$ on $\mathcal{X}$, and is associated with an index set $\mathcal{T}(\mathcal{A},\tau)$ of the region set $\mathcal{R}(\mathcal{A})$, where $\boldsymbol{W}_{l,i}$ denotes the i^{th} row of $\boldsymbol{W}_l$. Denote a_{li} the neuron corresponding to a hyperplane $\boldsymbol{W}_{l,i} \in \mathcal{A}$, we have

$$a_{li}(\boldsymbol{x}) = \begin{cases} \boldsymbol{W}_{l,i}\boldsymbol{x} & \text{if } \boldsymbol{x} \in r\ \forall\ r \in \{q \in \mathcal{R} | \pi_i\tau(q) = 1\} \\ 0 & \text{otherwise,} \end{cases}$$

i.e., a_{li} is linear over regions of $\mathcal{R}$ that are active on the i^{th} neuron of the layer. We then have $\mathcal{Q}_{li} = \{q \in \mathcal{R} | \pi_i\tau(q) = 1\}$ as the support of a_{li}.

To present the effect in the form of $\boldsymbol{W}_l$, we have

$$\boldsymbol{W}_l^q = \text{ diag}(\tau(q))\boldsymbol{W}_l,$$

which is a linear map over each $q \in \mathcal{Q}_l = \bigcup_{i=1}^{n_l}\mathcal{Q}_{li}$. The case $l = 1$ is proved, with $\tau_1 = \tau$.

Assume now for all the neurons a_{lj}, $j = 1,\dots,n_l$, of layer l, a_{lj} is a linear functional over its support Q_{lj}, and is 0-valued otherwise. We proceed by building a new set of regions $\mathcal{Q}_{(l+1)i}, i = 1,\dots,n_{l+1}$, for layer $l+1$, whose neurons are linear functionals over regions of $\mathcal{Q}_{(l+1)i}$.

We separately discuss the cases of g with or without max pooling.

First, when g does not include max pooling, for a neuron $a_{(l+1)i}$ of layer $l+1$, it is a functional of the form

$$\begin{aligned}
a_{(l+1)i} &= g\sum_{j=1}^{n_l}\boldsymbol{W}_{l+1,ij}a_{lj} \\
&= g\ \text{pre}(a_{(l+1)i}),
\end{aligned}$$

where $\boldsymbol{W}_{l+1,ij}$ is the (i,j)-entry of $\boldsymbol{W}_{l+1}$. Since $\forall q \in \bigcup_{j=1}^{n_l}\mathcal{Q}_{lj}$, $a_{lj}, j = 1,\dots,n_l$, is a linear functional over q, so is the linear combination $\text{pre}(a_{(l+1)i})$ of them.

When $\text{pre}(a_{(l+1)i})(\boldsymbol{x}) > 0\ \forall \boldsymbol{x} \in q$, $g\ \text{pre}(a_{(l+1)i}) = \text{pre}(a_{(l+1)i})$, and q is not further divided by neuron i. When $\text{pre}(a_{(l+1)i})(\boldsymbol{x}) > 0$ for some of $\boldsymbol{x} \in q$, by the fact that $\text{pre}(a_{(l+1)i})$ is a monotonous function, it splits q into two regions q_+ and q_-, where $\forall \boldsymbol{x} \in q_+$, $\text{pre}(a_{(l+1)i})(\boldsymbol{x}) > 0$ and $\forall \boldsymbol{x} \in q_-$, $\text{pre}(a_{(l+1)i})(\boldsymbol{x}) \leq 0$. Since g sets $a_{(l+1)i} = 0$ $\forall \boldsymbol{x} \in q_-$, $a_{(l+1)i}$ is a linear functional over q_+. When $\text{pre}(a_{(l+1)i})(\boldsymbol{x}) \leq 0\ \forall \boldsymbol{x} \in q$, q does not provide support for neuron i, but it may support other neurons.

Consequently, neurons a_{l+1} further divide the region q into sub-regions, where for each region, a_{l+1} is a linear map. We say the boundaries of the new set of regions as the hyperplane arrangement induced by neurons, with a alight abuse of terminology. For the newly created set of regions $\mathcal{Q}_{l+1}$, we define the labeling function τ_{l+1} for layer $l+1$ such that for $\boldsymbol{x} \in q' \in \mathcal{Q}_{l+1}$, we have

$$\pi_i\tau^{\text{relu}}_{l+1}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } a_{(l+1)i}(\boldsymbol{x}) > 0 \\ 0 & \text{if } a_{(l+1)i}(\boldsymbol{x}) \leq 0. \end{cases}$$

Since for each $q' \in \mathcal{Q}_{l+1}$, it is a sub-region of $q \in \mathcal{Q}_l$, $q' \in \mathcal{Q}_l$ holds true as well. In this case, the new set of regions are built.

For the case that g includes max pooling, the neuron is of the form

$$a_{(l+1)i} = \max_{k\in K}(\ \text{ReLU}\sum_{j=1}^{n_l} \boldsymbol{W}_{l+1,(si+k)j}a_{lj})$$
$$= \max_{k\in K}(\ \text{pre}(a_{(l+1)i})_k),$$

where K is index set of neurons being pooled with $|K| = s$, and $\text{pre}(a_{(l+1)i})_k$ denotes $\text{ReLU}\sum_{j=1}^{n_l} \boldsymbol{W}_{l+1,(si+k)j}a_{lj}$.

Similarly, for each $k \in K$, $\text{pre}(a_{(l+1)i})_k$ may split q into two sub-regions. Denote the boundary as H_k if it indeed splits. $\{H_k\}_{k\in K}$, together with the boundary of q, forms a hyperplane arrangement $\mathcal{A}^*_{(l+1)i}$ induced by neurons within q. Denote the set of regions in this new arrangement as $\mathcal{Q}^*_{(l+1)i}$, for each $q^* \in \mathcal{Q}^*_{(l+1)i}$, consider the set of hyperplanes $\mathcal{A}'_{(l+1)i} = \{\ \text{pre}(a_{(l+1)i})_k - \text{pre}(a_{(l+1)i})_{k'}\}_{k<k',k,k'\in K}$. With a similar argument, they will create another hyperplane arrangement $\mathcal{A}'_{(l+1)i}$ within q^*. For $q' \in \mathcal{Q}'_{(l+1)i}$, $H \in \mathcal{A}'_{(l+1)i}$ does not have discontinuity in derivative — does not suddenly switch from constant function 0 to non-zero linear function. Now within each q', we impose an order on $\mathcal{A}^*_{(l+1)i}$, if $\text{pre}(a_{(l+1)i})_k - \text{pre}(a_{(l+1)i})_{k'} \geq 0$, we say $H_k \geq H'_k$. Given that K is a finite totally ordered set, the maximum element w.r.t. the defined order exists, and we denote its index as $k_{\max}$. Thus, for each q', $a_{(l+1)i} = \text{pre}(a_{(l+1)i})_{k_{\max}}$, which we have proved to be a linear function over q' in the ReLU case, so is a_{li}. Thus similar to the ReLU only case, Max Pooling with ReLU divides q into a new set of regions as well.

For the newly created set of regions $\mathcal{Q}_{l+1}$, we have a composed labeling function

$$\tau_{l+1}(q) = \tau^{\max}_{l+1}(q)\tau^{\text{relu}}_{l+1}(q),$$

where τ^{relu} is defined as before, and

$$\pi_k\tau^{\max}_{l+1}(q) = \begin{cases} 1 & \text{if } k = \text{argmax}_{k\in K}\ a_{(l+1)k}(\boldsymbol{x}), \forall \boldsymbol{x} \in q \\ 0 & \text{otherwise.} \end{cases}$$

Since for each $q' \in \mathcal{Q}_{l+1}$, it is a sub-region of $q \in \mathcal{Q}_l$, $q' \in \mathcal{Q}_l$ holds true as well. In this case, the new set of regions are built. The max pooling case is proved.

The same with the case $l = 1$, to present the effect in the form of $\boldsymbol{W}_l$, denoting $\tau_{l+1}(\boldsymbol{x}) = \tau^{\text{relu}}_{l+1}$ for ReLU only case, and $\tau_{l+1}(\boldsymbol{x}) = \tau^{\max}_{l+1}\tau^{\text{relu}}_{l+1}$ for ReLU with Max Pooling case, we have in the ReLU only case

$$\boldsymbol{W}^q_{l+1} = \ \text{diag}(\tau_{l+1}(q))\boldsymbol{W}_{l+1},$$

and in the ReLU and Max Pooling case,

$$\boldsymbol{W}^q_{l+1} = \boldsymbol{P}_{l+1}\ \text{diag}(\tau_{l+1}(q))\boldsymbol{W}_{l+1},$$

which both are linear maps over $q \in \mathcal{Q}_{l+1} = \bigcup_{i=1}^{n_{l+1}} \mathcal{Q}_{(l+1)i}$(note that $\boldsymbol{P}_{l+1}$ is a also a linear mapping/matrix).

By induction, $\forall i \leq l+1$, $\boldsymbol{W}^q_i$ is a linear map. Cascading the result, we have the neural network $\boldsymbol{T}$ as a linear map over $q \in \mathcal{Q}_{(l+1)}$

$$\boldsymbol{T}^{l+1}_q = \prod_{i=1}^{l+1} \boldsymbol{W}^q_i.$$

We now finish the induction and prove that for $0 < l \leq L$, there exists a set $\mathcal{Q}_l$ such that $\forall q \in \mathcal{Q}_l$, $\boldsymbol{T}^l_q$ is linear over q. Given that we are interested in $\boldsymbol{T}^L_q$ in this paper, we drop the upper index, and denote it as

$$\boldsymbol{T}_q = \prod_{i=1}^{L} \boldsymbol{W}^q_i,$$

and the corresponding region set Q_L is the set of regions over which $\boldsymbol{T}$ is linear. We also drop the index, and denote it as Q. □

APPENDIX C
PROOF OF LEMMA 3.3.

Proof. By lemma 3.2, a set of regions $\mathcal{Q}$ exists such that for $q \in \mathcal{Q}$, $\boldsymbol{T}_q$ is a linear mapping induced by $\boldsymbol{T}$ over q.

For any given $\boldsymbol{x} \in \mathcal{X}$, denote the region it belongs to as $q_{\boldsymbol{x}}$. Let $d_{\min} = \min_{\boldsymbol{x}\in S_m^{(x)}} \min_{\boldsymbol{x}'\in\partial q_{\boldsymbol{x}}} \rho(\boldsymbol{x}, \boldsymbol{x}')$, the shortest distance from $\boldsymbol{x}$ to the boundary of $q_{\boldsymbol{x}}$, denoted as $\partial q_{\boldsymbol{x}}$, among all training samples. Denote by a_{lk} the neuron that defines the hyperplane corresponding to the boundary that produces the shortest distance $d_{\min}$. Note that a_{lk} may exist in the intermediate network layers, i.e., $1 \leq l \leq L$, and the specific value of l depends on the training set S_m and the learned $\boldsymbol{T}$. By Corollary 3.1., we have $\forall\{\boldsymbol{x}' \in \mathcal{X}|\ \boldsymbol{x} + (\boldsymbol{x}' - \boldsymbol{x}) \in q_{\boldsymbol{x}}\}$, $a_{lk|\boldsymbol{x}}(\boldsymbol{x}) - a_{lk|\boldsymbol{x}}(\boldsymbol{x}') = a_{lk|\boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}')$. Thus

$$\begin{aligned} ||a_{lk|\boldsymbol{x}}(\boldsymbol{x}) - a_{lk|\boldsymbol{x}}(\boldsymbol{x}')|| &= ||a_{lk|\boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}')|| \\ &\leq ||a_{lk|\boldsymbol{x}}||||\boldsymbol{x} - \boldsymbol{x}'|| \\ &= ||a_{lk|\boldsymbol{x}}||d_{\min}. \end{aligned}$$

Given $\boldsymbol{x}' \in \partial q_{\boldsymbol{x}}$, it implies $a_{lk}(\boldsymbol{x}') = 0$, and we have a lower bound on $d_{\min}$ as

$$\begin{aligned} d_{\min} &\geq \frac{||a_{lk|\boldsymbol{x}}(\boldsymbol{x}) - a_{lk|\boldsymbol{x}}(\boldsymbol{x}')||}{||a_{lk|\boldsymbol{x}}||} \\ &= \frac{|a_{lk|\boldsymbol{x}}(\boldsymbol{x})|}{||a_{lk|\boldsymbol{x}}||} \\ &\geq \frac{|a_{lk|\boldsymbol{x}}(\boldsymbol{x})|}{\prod_{i=1}^{l} \sigma^i_{\max|\boldsymbol{x}}}, \end{aligned}$$

where $\sigma^i_{\max|\boldsymbol{x}}$ is the maximum singular value of $\boldsymbol{W}^q_i$ for $i = 1, \ldots, l$.

Since $\boldsymbol{W}^q_i$ is a submatrix of $\boldsymbol{W}_i$, by Cauchy interlacing law by rows deletion [12], we have $\sigma^i_{\max|\boldsymbol{x}} \leq \sigma^i_{\max}$, and

$$d_{\min} \geq \frac{|a_{lk|\boldsymbol{x}}(\boldsymbol{x})|}{\prod_{i=1}^{l} \sigma^i_{\max}}.$$

Denote $o(S_m, \boldsymbol{T})/2 = |a_{lk|\boldsymbol{x}}(\boldsymbol{x})|$ to stress the fact that it is a fixed value once S_m and $\boldsymbol{T}$ are given, we have a lower bound

$$r = \frac{o(S_m, \boldsymbol{T})/2}{\prod_{i=1}^{l(S_m,\boldsymbol{T})} \sigma^i_{\max}},$$

where we have explicitly write $l(S_m, \boldsymbol{T})$ to emphasize the dependence of l on S_m and $\boldsymbol{T}$. In addition, $q_{\boldsymbol{x}} \in \mathcal{Q}$, we have $\boldsymbol{T}$ is linear over $q_{\boldsymbol{x}}$. Consequently, a covering set of $\mathcal{X}$ with radius r is found, such that within each covering ball, $\boldsymbol{T}$ is

linear. Then for any given $\boldsymbol{x} \in \mathcal{X}$ and $\{\boldsymbol{x}' \in \mathcal{X} | \, ||\boldsymbol{x} - \boldsymbol{x}'|| \leq r\}$, $\boldsymbol{x}' \in q_{\boldsymbol{x}}$, thus $\boldsymbol{T}\boldsymbol{x} - \boldsymbol{T}\boldsymbol{x}' = \boldsymbol{T}_{|\boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}')$. The diameter γ of the covering ball is $2r$. □

APPENDIX D
PROOF OF LEMMA 3.4.

Proof. By lemma 3.3, there exists a covering of $\mathcal{X}$ such that $\boldsymbol{T}$ is linear over each covering ball B containing $\boldsymbol{x} \in S_m^{(x)}$, denoted as $\boldsymbol{T}_{|B}$. By lemma 3.1, within such a B, $\boldsymbol{T}_{|B}$ is $\delta_{|B}$-isometry w.r.t. variation space $\mathcal{X} - \mathcal{N}(\boldsymbol{T}_{|B})$. By Cauchy interlacing law by rows deletion [12], we have

$$\sigma^i_{\min|B} \leq \sigma^i_{\max}, \; \sigma^i_{\min|B} \geq \sigma^i_{\min},$$

where $\sigma^i_{\min}$ and $\sigma^i_{\max}$, $i = 1, \ldots, L$, are respectively the minimum and maximum singular values of weight matrices of $\boldsymbol{T}$, and $\sigma^i_{\min|B}$ and $\sigma^i_{\max|B}$ are the corresponding ones of $\boldsymbol{T}_{|B}$.

Some extra attentions need to be taken to deal with the $\boldsymbol{P}_l$ matrix introduced by max pooling. Note that in lemma 3.2, $||\boldsymbol{P}_l \operatorname{diag}(\tau_l(q))\boldsymbol{W}_l\boldsymbol{x}||$ is equivalent to $||\operatorname{diag}(\tau_l(q))\boldsymbol{W}_l\boldsymbol{x}||$ since in computing the norm, a summation is computed anyway. Thus, the Cauchy interlacing law by row deletion applies to $\boldsymbol{W}_l^q$ with $\boldsymbol{P}_l$ as well.

Denote

$$\delta^1_{|B} = \prod_{i=1}^{L} \sigma^i_{\max|B}, \; \delta^2_{|B} = \prod_{i=1}^{L} \sigma^i_{\min|B},$$
$$\delta^1 = \prod_{i=1}^{L} \sigma^i_{\max}, \; \delta^2 = \prod_{i=1}^{L} \sigma^i_{\min}.$$

Anchoring the four points $\delta^1_B, \delta^2_{|B}, \delta^1, \delta^2$ on the graph of $f(x) = |x - 1|$, we observe that $[\delta^2_{|B}, \delta^1_{|B}]$ lies between the interval $[\delta^2, \delta^1]$. Thus we have

$$\max(|\delta^1_{|B} - 1|, |\delta^2_{|B} - 1|) \leq \max(|\delta^1 - 1|, |\delta^2 - 1|).$$

Since $\boldsymbol{T}_{|B}$ means $\boldsymbol{T}$ over B is $\max(|\delta^1_{|B} - 1|, |\delta^2_{|B} - 1|)$-isometry, which implies that it is also $\max(|\delta^1 - 1|, |\delta^2 - 1|)$-isometry. □

APPENDIX E
PROOF OF THEOREM 3.1

Proof. Similar to the proof of theorem 2.1, we partition the space $\mathcal{Z}$ via the assumed γ-cover. Since $\mathcal{X}$ is a k-dimensional manifold, its covering number is upper bounded by $C^k_{\mathcal{X}}/\gamma^k$. Let K be the overall number of covering set, which is upper bounded by $|\mathcal{Y}|C^k_{\mathcal{X}}/\gamma^k$. Denote C_i the ith covering ball, and let N_i be the set of index of training samples that fall into C_i. Note that $(|N_i|)_{i=1\ldots K}$ is an IDD multimonial random variable with parameters m and $(|\mu(C_i)|)_{i=1\ldots K}$. Then

$$\begin{aligned}
&|R(f \circ \boldsymbol{T}) - R_m(f \circ \boldsymbol{T})| \\
=&|\sum_{i=1}^{K} \mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}), y)]\mu(C_i) - \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}_i), y_i)| \\
\leq&|\sum_{i=1}^{K} \mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}), y)]\frac{|N_i|}{m} - \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}_i), y_i)| \\
&+ |\sum_{i=1}^{K} \mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}), y)]\mu(C_i) - \sum_{i=1}^{K} \mathbb{E}_{z\sim\mu}[\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}), y)]\frac{|N_i|}{m}| \\
\leq&|\frac{1}{m}\sum_{i=1}^{K}\sum_{j\in N_i} \max_{z'\in C_i, \boldsymbol{x}'-\boldsymbol{x}_j\in\mathcal{P}_{\boldsymbol{x}_j}} |\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}'), y') - \mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}_j), y_j)| \qquad (18) \\
&+ |\max_{z\in\mathcal{Z}} |\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}), y)| \sum_{i=1}^{K} |\frac{|N_i|}{m} - \mu(C_i)||. \qquad (19)
\end{aligned}$$

Remember that $z = (\boldsymbol{x}, y)$.

By the assumption that $\boldsymbol{T}$ is γ-cover δ-isometry w.r.t. $\mathcal{P}_{\boldsymbol{x}}$ of $\boldsymbol{x} \in S_m^{(x)}$ and the Lipschitz constant of $\mathcal{L} \circ f$ is A, suppose the maximum is achieved at $\boldsymbol{x}_k$ and $\boldsymbol{x}_k \in C_p$, we have

$$\begin{aligned}
&\max_{z'\in C_p, \boldsymbol{x}'-\boldsymbol{x}_k\in\mathcal{P}_{\boldsymbol{x}_k}} |\mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}'), y') - \mathcal{L}(f(\boldsymbol{T}\boldsymbol{x}_k), y_k)| \\
\leq& A \max_{z'\in C_p, \boldsymbol{x}'-\boldsymbol{x}_k\in\mathcal{P}_{\boldsymbol{x}_k}} ||\boldsymbol{T}_{|\boldsymbol{x}_k}(\boldsymbol{x}' - \boldsymbol{x}_k)|| \qquad (20) \\
\leq& A \max_{z'\in C_p, \boldsymbol{x}'-\boldsymbol{x}_k\in\mathcal{P}_{\boldsymbol{x}_k}} (||\boldsymbol{x}' - \boldsymbol{x}_k|| + \delta) \qquad (21) \\
\leq& A(\gamma + \delta),
\end{aligned}$$

where $\delta = 2b|\prod_{i=1}^{L} \sigma^i_{\max} - 1|$ in (21) since we have $||\boldsymbol{T}_{|\boldsymbol{x}_k}(\boldsymbol{x}' - \boldsymbol{x}_k)|| \leq ||\boldsymbol{x}' - \boldsymbol{x}_k|| + 2b|\prod_{i=1}^{L} \sigma^i_{\max} - 1|$ by Lemma 3.1, and $\gamma = o(S_m, \boldsymbol{T})/\left(\prod_{i=1}^{l(S_m,\boldsymbol{T})} \sigma^i_{\max}\right) > 0$ by Lemma 3.3, with values of $o(S_m, \boldsymbol{T})$ and $1 \leq l(S_m, \boldsymbol{T}) \leq L$ depending on the training set S_m and learned network $\boldsymbol{T}$. Thus eq. (18) is less than or equal to $A(\gamma + \delta)$ with the specified γ and δ. By Breteganolle-Huber-Carol inequality, eq. (19) is less than or equal to $M\sqrt{\frac{\log(2)|\mathcal{Y}|2^{k+1}C^k_{\mathcal{X}}}{\gamma^k n} + \frac{2\log(1/\nu)}{m}}$.

The proof is finished. Note that given the covering number of $\mathcal{X}$ as $\mathcal{N} = (\frac{C_{\mathcal{X}}}{\gamma/2})^k$, we have also proved that the algorithm is $(|\mathcal{Y}|\mathcal{N}, A(\gamma + \delta))$-robust. □

APPENDIX F

In this section, we present a SGD based algorithm for the constrained optimization problem (i.e. problem (13)) of training a DNN of L layers with parameters $\Theta = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L}$ and objective function $\mathcal{L}$. The constraints enforce the weight matrix (kernel) $\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ of any l^{th} layer of the network staying on the Stiefel manifold defined as $\mathcal{M}_l = \{\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}} | \boldsymbol{W}_l^\top \boldsymbol{W}_l = \boldsymbol{I}\}$, assuming $n_l \geq n_{l-1}$. The presented algorithm applies directly to fully-connected network layers. For convolutional layers used in CNNs, one may refer to section 4.5 for how to convert their layer kernels as matrices.

For the t^{th} iteration of SGD, the algorithm performs the following sequential steps to update $\boldsymbol{W}_l^t \in \mathcal{M}_l$ for the l^{th} network layer with $l \in \{1, \ldots, L\}$. Updating of other

network parameters such as bias vectors $\{\boldsymbol{b}_l\}_{l=1}^L$ is the same as standard SGD based methods. The algorithm is similar to those of optimization on matrix manifolds in [1], [9], [45], where properties of convergence are also analyzed.

1) Compute the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l^t}$ in the embedding Euclidean space via back-propagation. One may alternatively use the momentum [52] to replace the gradient term in the following steps.
2) Project $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l^t}$ (or its momentum version) onto the tangent space $T_{\boldsymbol{W}_l^t}\mathcal{M}_l$ by $\mathcal{P}_{\boldsymbol{W}_l^t}\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l^t}$, to obtain the manifold gradient $\Omega_{\boldsymbol{W}_l^t}$. For the considered Stiefel manifold, the tangent space at $\boldsymbol{W}_l^t$ is defined as $T_{\boldsymbol{W}_l^t}\mathcal{M}_l = \{\boldsymbol{Z} \in \mathbb{R}^{n_l \times n_{l-1}} | \boldsymbol{W}_l^{t\top}\boldsymbol{Z} + \boldsymbol{Z}^\top \boldsymbol{W}_l^t = 0\}$, and the projection operator is defined as $\mathcal{P}_{\boldsymbol{W}_l^t}\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l^t} = \left(\boldsymbol{I} - \boldsymbol{W}_l^t \boldsymbol{W}_l^{t\top}\right)\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l^t} + \frac{1}{2}\boldsymbol{W}_l^t\left(\boldsymbol{W}_l^{t\top}\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l^t} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l^t}^\top \boldsymbol{W}_l^t\right)$, where $\boldsymbol{I}$ is the identity matrix of compatible size.
3) Update $\boldsymbol{W}_l^t$ as $\boldsymbol{W}_l^t - \eta^t \Omega_{\boldsymbol{W}_l^t}$ with the step size η^t that satisfies conditions of convergence [1], [9].
4) Perform the retraction $\mathcal{R}_{\boldsymbol{W}_l^t}(-\eta^t \Omega_{\boldsymbol{W}_l^t})$ that defines a mapping from the tangent space to the Stiefel manifold, and update $\boldsymbol{W}_l^t$ as $\boldsymbol{W}_l^{t+1} = \mathcal{R}_{\boldsymbol{W}_l^t}(-\eta^t \Omega_{\boldsymbol{W}_l^t})$. The retraction is achieved by $\mathcal{R}_{\boldsymbol{W}_l^t}(-\eta^t \Omega_{\boldsymbol{W}_l^t}) = \mathcal{Q}(\boldsymbol{W}_l^t - \eta^t \Omega_{\boldsymbol{W}_l^t})$, where the operator $\mathcal{Q}$ denotes the Q factor of the QR matrix decomposition. QR decomposition can be computed using Gram-Schmidt orthonormalization.

APPENDIX G
PROOF OF LEMMA 4.1

Proof. We first consider the general case, and let $P = \min(M, N)$. Denote singular values of $\boldsymbol{W}$ as $\sigma_1 = \cdots = \sigma_P = 1$, and singular values of $\widetilde{\boldsymbol{W}}$ as $\tilde{\sigma}_1 \geq \cdots \geq \tilde{\sigma}_P$. Based on the properties of matrix extreme singular values, we have

$$\sigma_1 = \|\boldsymbol{W}\|_2 = \max_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{W}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \min_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{W}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \sigma_P = 1.$$

Let $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\neq 0} \frac{\|\widetilde{\boldsymbol{W}}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}$, we have

$$\tilde{\sigma}_1 = \frac{\|\widetilde{\boldsymbol{W}}\boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2} = \frac{\|\boldsymbol{G}\boldsymbol{W}\boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2} \leq \frac{\|\boldsymbol{G}\|_2 \|\boldsymbol{W}\boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2},$$

where we have used the fact that $\|\boldsymbol{A}\boldsymbol{b}\|_2 \leq \|\boldsymbol{A}\|_2 \|\boldsymbol{b}\|_2$ for any $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$. We thus have

$$\tilde{\sigma}_1 \leq \|\boldsymbol{G}\|_2 \frac{\|\boldsymbol{W}\boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2} \leq \|\boldsymbol{G}\|_2 \max_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{W}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = |g_{\max}|.$$

Since $\boldsymbol{G}$ has nonzero entries, we have $\boldsymbol{W} = \boldsymbol{G}^{-1}\widetilde{\boldsymbol{G}}$. Let $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\neq 0} \frac{\|\widetilde{\boldsymbol{W}}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}$, the properties of matrix extreme singular values give $\tilde{\sigma}_P = \frac{\|\widetilde{\boldsymbol{G}}\boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2}$, and $\sigma_P = \min_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{W}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = 1$. We thus have

$$1 = \min_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{G}^{-1}\widetilde{\boldsymbol{G}}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \leq \frac{\|\boldsymbol{G}^{-1}\widetilde{\boldsymbol{G}}\boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2} \leq \|\boldsymbol{G}^{-1}\|_2 \frac{\|\widetilde{\boldsymbol{G}}\boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2},$$

which gives $\tilde{\sigma}_P \geq |g_{\min}|$. Overall, we have

$$|g_{\max}| \geq \tilde{\sigma}_1 \geq \cdots \geq \tilde{\sigma}_P \geq |g_{\min}|.$$

We next consider the special case of $M \leq N$ and $\text{rank}(\boldsymbol{W}) = M$. Without loss of generality, we assume diagonal entries $\{g_i\}_{i=1}^M$ of $\boldsymbol{G}$ are all positive and ordered. By definition we have $\widetilde{\boldsymbol{W}} = \boldsymbol{I}\boldsymbol{G}\boldsymbol{W}$, where $\boldsymbol{I}$ is an identity matrix of size $M \times M$. Let $\boldsymbol{V} = \left[\boldsymbol{W}^\top, \boldsymbol{W}^{\perp\top}\right]$, where $\boldsymbol{W}^\perp$ denotes the orthogonal complement of $\boldsymbol{W}$, we thus have the SVD of $\widetilde{\boldsymbol{W}}$ by construction as $\widetilde{\boldsymbol{W}} = \boldsymbol{I}\left[\boldsymbol{G}, \boldsymbol{0}\right]\boldsymbol{V}^\top$. When some values of $\{g_i\}_{i=1}^M$ are not positive, the SVD can be constructed by changing the signs of the corresponding columns of either $\boldsymbol{I}$ or $\boldsymbol{V}$. Since matrix singular values are uniquely determined (while singular vectors are not), singular values of $\widetilde{\boldsymbol{W}}$ are thus exactly $\{|g_i|\}_{i=1}^M$. □