SIAM J. MATH. DATA SCI.
Vol. 1, No. 3, pp. 476–496

Stop Memorizing: A Data-Dependent Regularization Framework for Intrinsic Pattern Learning[*]

Wei Zhu[†], Qiang Qiu[‡], Bao Wang[§], Jianfeng Lu[†], Guillermo Sapiro[‡], and Ingrid Daubechies[¶]

Abstract. Deep neural networks (DNNs) typically have enough capacity to fit random data by brute force even when conventional data-dependent regularizations focusing on the geometry of the features are imposed. We find out that the reason for this is the inconsistency between the enforced geometry and the standard softmax cross entropy loss. To resolve this, we propose a new framework for data-dependent DNN regularization, the Geometrically-Regularized-Self-Validating neural Networks (GRSVNet). During training, the geometry enforced on one batch of features is simultaneously validated on a separate batch using a validation loss consistent with the geometry. We study a particular case of GRSVNet, the Orthogonal-Low-rank Embedding (OLE)-GRSVNet, which is capable of producing highly discriminative features residing in orthogonal low-rank subspaces. Numerical experiments show that OLE-GRSVNet outperforms DNNs with conventional regularization when trained on real data, especially when the training samples are scarce. More importantly, unlike conventional DNNs, OLE-GRSVNet refuses to memorize random data or random labels, suggesting that it only learns intrinsic patterns by reducing the memorizing capacity of the baseline DNN.

Key words. deep neural networks, data-dependent regularization, network memorization, intrinsic pattern learning

AMS subject classifications. 68T45, 68T10

DOI. 10.1137/19M1236886

1. Introduction.

It remains an open question why deep neural networks (DNNs), typically with far more model parameters than training samples, can achieve such small generalization error. Previous work used various complexity measures from statistical learning theory, such as VC dimension [24], Radamacher complexity [2], and uniform stability [3, 18], to provide an upper bound for the generalization error, suggesting that the effective capacity of DNNs, possibly with some regularization techniques, is usually limited.

However, the experiments by Zhang et al. [28] showed that, even with data-independent regularization, DNNs can perfectly fit the training data when the true labels are replaced by

[*]Received by the editors January 4, 2019; accepted for publication (in revised form) June 10, 2019; published electronically August 27, 2019.
https://doi.org/10.1137/19M1236886
Funding: This work was partially supported by the NSF, the DoD, the NIH, and Google.
[†]Department of Mathematics, Duke University, Durham, NC 27708 (zhu@math.duke.edu, jianfeng@math.duke.edu).
[‡]Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 (qiang.qiu@duke.edu, guillermo.sapiro@duke.edu).
[§]Department of Mathematics, University of California Los Angeles, Los Angeles, CA 90095 (wangbao@math.ucla.edu).
[¶]Department of Mathematics and Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 (ingrid@math.duke.edu).

random labels or when the training data are replaced by Gaussian noise. This suggests that DNNs with data-independent regularization have enough capacity to "memorize" the training data. This poses an interesting question for network regularization design: Is there a way for DNNs to refuse to (over)fit training samples with random labels, while exhibiting better generalization power than conventional DNNs when trained with true labels? Such networks are very important because they will extract only intrinsic patterns from the training data instead of memorizing miscellaneous details.

One would expect that data-dependent regularizations should be a better choice for reducing the memorizing capacity of DNNs. Such regularizations are typically enforced by penalizing the standard softmax cross entropy loss with an extra *geometric loss* which regularizes the feature geometry [15, 26, 27, 29]. However, regularizing DNNs with an extra geometric loss has two disadvantages: First, the output of the softmax layer, usually viewed as a probability distribution, is typically not derived based on the feature geometry enforced by the geometric loss. Thus minimizing the geometric loss would not necessarily benefit the validation, and these two losses are therefore inconsistent. As a result, the geometric loss typically has a small weight to avoid jeopardizing the minimization of the softmax loss. Second, we find that DNNs with such regularization can still perfectly (over)fit random training samples or random labels. The reason is that the geometric loss (because of its small weight) is ignored and only the softmax loss is minimized.

This suggests that simply penalizing the softmax loss with a geometric loss is not sufficient to regularize DNNs. Instead, the softmax loss should be replaced by a *validation loss* that is consistent with the enforced geometry. More specifically, every training batch B is split into two sub-batches, the geometry batch B^g and the validation batch B^v. The geometric loss l_g is imposed on the features of B^g for them to exhibit a desired geometric structure. A semisupervised learning algorithm based on the proposed feature geometry is then used to generate a predicted label distribution for the validation batch, which combined with the true labels defines a validation loss on B^v. The total loss on the training batch B is then defined as the weighted sum $l = l_g + \lambda l_v$. Because the predicted label distribution on B^v is based on the enforced geometry, the geometric loss l_g can no longer be neglected. Therefore, l_g and l_v will be minimized simultaneously; i.e., the geometry is correctly enforced (small l_g) and it can be used to predict validation samples (small l_v). We call such DNNs Geometrically-Regularized-Self-Validating neural Networks (GRSVNets). See Figure 1a for a visual illustration of the network architecture.

GRSVNet is a general architecture because every consistent geometry/validation pair can fit into this framework as long as the loss functions are differentiable. In this paper, we focus on a particular type of GRSVNet, the Orthogonal-Low-rank-Embedding-GRSVNet (OLE-GRSVNet). More specifically, we impose the OLE loss [19] on the geometry batch to produce features residing in orthogonal subspaces, and we use the distances between the validation features and those subspaces to define a predicted label distribution on the validation batch. We prove that the loss function obtains its minimum if and only if the subspaces of different classes spanned by the features in the geometry batch are orthogonal, and the features in the validation batch reside perfectly in the subspaces corresponding to their labels (see Figure 1f). We show in our experiments that OLE-GRSVNet has better generalization performance when trained on real data, but it refuses to memorize the training samples when given random training data or random labels, which suggests that OLE-GRSVNet effectively learns intrinsic patterns.

Our contributions can be summarized as follows:

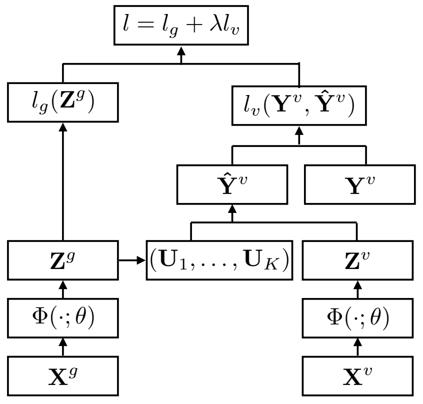
- We proposed a potential framework, GRSVNet, to effectively impose data-dependent DNN regularization. The core idea is the self-validation of the enforced geometry with a consistent validation loss on a separate batch of features.
- We study a particular case of GRSVNet, OLE-GRSVNet, that can produce highly discriminative features: Samples from the same class belong to a low-dimensional subspace, and the subspaces for different classes are orthogonal.
- OLE-GRSVNet achieves better generalization performance when compared to DNNs with conventional regularizers. This is especially the case when the training data are scarce. More importantly, unlike conventional DNNs, OLE-GRSVNet refuses to fit the training data (i.e., with a training error close to random guess) when the training data or the training labels are randomly generated. This implies that OLE-GRSVNet tends not to memorize the training samples and instead learns intrinsic patterns.

2. Related work.

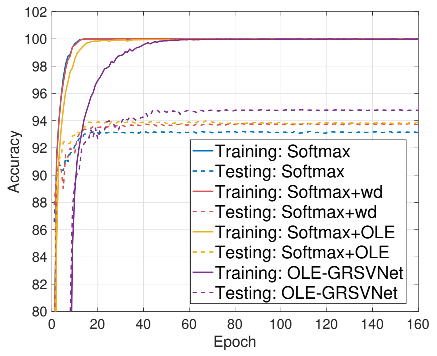
Many data-dependent regularizations focusing on feature geometry have been proposed for deep learning [15, 26, 27, 29]. The center loss [26] produces compact clusters by minimizing the Euclidean distance between features and their class centers. Weston et al. proposed to recover a low-dimensional feature structure via semisupervised manifold embedding [27]. LDMNet [29] extracts features sampling a collection of low-dimensional manifolds by explicitly minimizing the manifold dimension. The OLE loss [15, 19] increases interclass separation and intraclass similarity by embedding inputs into orthogonal low-dimensional subspaces. However, as mentioned in section 1, these regularizations are imposed by adding the geometric loss to the softmax loss, which, when viewed as a probability distribution, is typically not consistent with the desired geometry. Our proposed GRSVNet instead uses a validation loss based on the regularized geometry so that the predicted label distribution has a meaningful geometric interpretation.

The way in which GRSVNets impose geometric loss and validation loss on two separate batches of features extracted with two identical baseline DNNs bears a certain resemblance to the siamese network architecture [5] used extensively in metric learning [4, 9, 11, 21, 23]. The difference is, unlike contrastive loss [9] and triplet loss [21] in metric learning, the feature geometry is explicitly regularized in GRSVNets, and a representation of the geometry, e.g., basis of the low-dimensional subspace, can be later used directly for the classification of test data.

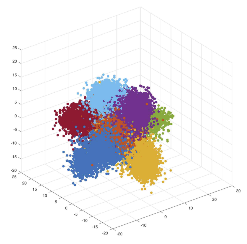
Our work is also related to two recent papers [1, 28] addressing the memorization of DNNs. Zhang et al. [28] empirically showed that conventional DNNs, even with data-independent regularization, are fully capable of memorizing random labels or random data. Arpit et al. [1] argued that DNNs trained with stochastic gradient descent (SGD) tend to fit patterns first before memorizing miscellaneous details, suggesting that memorization of DNNs depends also on the data itself, and SGD with early stopping is a valid strategy in conventional DNN training. We empirically demonstrate in our paper that when data-dependent regularization is imposed in accordance with the validation, GRSVNets have the potential to refuse memorizing random labels or random data, and only extract intrinsic patterns. A conjecture to explain



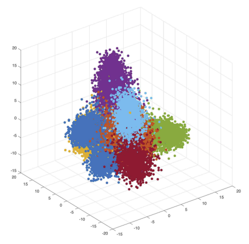
(a) GRSVNet architecture



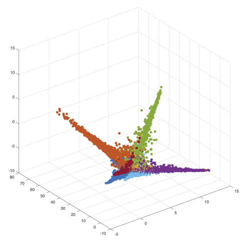
(b) Learning curves



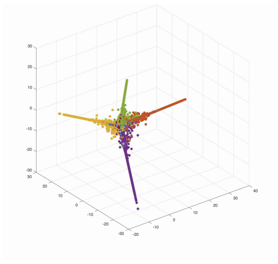
(c) Softmax
Accuracy = 93.10%



(d) Weight decay
Accuracy = 93.73%



(e) Softmax + OLE
Accuracy = 94.04%



(f) OLE-GRSVNet
Accuracy = 94.75%

Figure 1. *GRSVNet architecture and the results of different networks with the same VGG-11 baseline architecture on the SVHN dataset with real data and real labels.* (a) *GRSVNet architecture (better understood in its special case OLE-GRSVNet detailed in section 3).* (b) *Training/testing accuracy.* (c)–(f) *Features of the test data learned by different networks visualized in three dimensions using PCA. Note that for OLE-GRSVNet, only four classes (out of 10) have nonzero 3D embedding (Theorem 3.2).*

this phenomenon is provided in section 4.

3. GRSVNet and its special case: OLE-GRSVNet. As pointed out in section 1, the core idea of GRSVNet is to self-validate the geometry using a consistent validation loss. To contextualize this idea, we study a particular case, the OLE-GRSVNet, where the regularized feature geometry is orthogonal low-dimensional subspaces, and the validation loss is defined by the distances between the validation features and the subspaces.

3.1. OLE loss. The OLE loss was originally proposed by Qiu and Sapiro in [19]. Consider a K-way classification problem. Let $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{d \times N}$ be a collection of data points $\{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^d$. Let $\mathbf{X}_c$ denote the submatrix of $\mathbf{X}$ formed by inputs of the cth class. The authors in [19] proposed to learn a linear transformation $\mathbf{T} : \mathbb{R}^d \to \mathbb{R}^d$ that maps data from the same class $\mathbf{X}_c$ into a low-dimensional subspace, while mapping the entire data $\mathbf{X}$ into a high-dimensional linear space. This is achieved by solving

$$\min_{\mathbf{T}:\mathbb{R}^d \to \mathbb{R}^d} \sum_{c=1}^{K} \|\mathbf{T}\mathbf{X}_c\|_* - \|\mathbf{T}\mathbf{X}\|_*, \quad \text{s.t. } \|\mathbf{T}\|_2 = 1, \tag{3.1}$$

where $\|\cdot\|_*$ is the matrix nuclear norm, which is a convex lower bound of the rank function on the unit ball in the operator norm [20]. The norm constraint $\|\mathbf{T}\|_2 = 1$ is imposed to avoid the trivial solution $\mathbf{T} = \mathbf{0}$. It is proved in [19] that the OLE loss (3.1) is always nonnegative, and the global optimum value 0 is obtained if $\mathbf{T}\mathbf{X}_c \perp \mathbf{T}\mathbf{X}_{c'}\ \forall c \neq c'$.

Lezama et al. [15] later used OLE loss as a data-dependent regularization for deep learning. Let $\Phi(\cdot;\theta) : x \in \mathbb{R}^d \mapsto z = \Phi(x,\theta) \in \mathbb{R}^D$ be a baseline DNN feature extractor, where D is the dimension of the ambient feature space, which is typically chosen to be larger than the number of classes. Given a batch of labeled inputs $(\mathbf{X}, \mathbf{Y})$ and their corresponding features $\mathbf{Z} = \Phi(\mathbf{X};\theta)$, the OLE loss on $\mathbf{Z}$ is

$$l_g(\mathbf{Z}) = \sum_{c=1}^{K} \|\mathbf{Z}_c\|_* - \|\mathbf{Z}\|_* = \sum_{c=1}^{K} \|\Phi(\mathbf{X}_c;\theta)\|_* - \|\Phi(\mathbf{X};\theta)\|_*. \tag{3.2}$$

The OLE loss is later combined with the standard softmax loss for training. More specifically, let $W \in \mathbb{R}^{D\times K}$ be the weights of the last fully connected layer; then the total loss on the input batch is defined as

$$l(\mathbf{X},\mathbf{Y}) = l_g(\mathbf{Z}) + \lambda l_{\text{softmax}}(W^T\mathbf{Z}, \mathbf{Y}), \tag{3.3}$$

where l_{softmax} is the standard softmax cross entropy loss. We will henceforth call such a network "softmax+OLE." Softmax+OLE significantly improves the generalization performance, but it suffers from two problems because of the inconsistency between the softmax loss and the OLE loss: First, the learned features no longer exhibit the desired geometry of orthogonal low-dimensional subspaces. Second, as will be shown in section 4, softmax+OLE is still capable of memorizing random data or random labels; i.e., it does not reduce the memorizing capacity of DNNs.

3.2. OLE-GRSVNet. We will now explain how to incorporate OLE loss into the proposed GRSVNet framework. First, let us better understand the geometry enforced by the OLE loss by stating the following theorem.

Theorem 3.1. *Let $\mathbf{Z} = [\mathbf{Z}_1, \dots, \mathbf{Z}_c]$ be a horizontal concatenation of matrices $\{\mathbf{Z}_c\}_{c=1}^K$. The OLE loss $l_g(\mathbf{Z})$ defined in (3.2) is always nonnegative. Moreover, $l_g(\mathbf{Z}) = 0$ if and only if $\mathbf{Z}_c^*\mathbf{Z}_{c'} = \mathbf{0}\ \forall c \neq c'$, i.e., the column spaces of $\mathbf{Z}_c$ and $\mathbf{Z}_{c'}$ are orthogonal.*

The proofs of Theorem 3.1 and those of the remaining theorems are detailed in the appendices. Note that Theorem 3.1 is stronger than the one in [19], which only showed one direction of the result. We then need to define a validation loss l_v that is consistent with the geometry enforced by l_g. A natural choice would be the distances between the validation features and the subspaces spanned by $\{\mathbf{Z}_c\}_{c=1}^K$.

Now we detail the architecture for OLE-GRSVNet. Given a baseline DNN, we split every training batch $\mathbf{X} \in \mathbb{R}^{d\times|B|}$ into two sub-batches, the geometry batch $\mathbf{X}^g \in \mathbb{R}^{d\times|B_g|}$ and the validation batch $\mathbf{X}^v \in \mathbb{R}^{d\times|B_v|}$, both of which are mapped by the same baseline DNN into features $\mathbf{Z}^g = \Phi(\mathbf{X}^g;\theta)$ and $\mathbf{Z}^v = \Phi(\mathbf{X}^v;\theta)$. Assume for now that both $\mathbf{X}^g$ and $\mathbf{X}^v$ contain samples from all K classes (more details will be explained in Remark 3.5). The OLE loss $l_g(\mathbf{Z}^g)$ is imposed on the geometry batch to ensure $\operatorname{span}(\mathbf{Z}^g_c)$ are orthogonal low-dimensional subspaces, where $\operatorname{span}(\mathbf{Z}^g_c)$ is the column space of $\mathbf{Z}^g_c$. Let $\mathbf{Z}^g_c = \mathbf{U}_c\mathbf{\Sigma}_c\mathbf{V}^*_c$ be the (compact) singular value decomposition (SVD) of $\mathbf{Z}^g_c$; then the columns of $\mathbf{U}_c$ form an orthonormal basis of $\operatorname{span}(\mathbf{Z}^g_c)$. For any feature $\boldsymbol{z} = \Phi(\boldsymbol{x};\theta) \in \mathbf{Z}^v$ in the validation batch, its projection onto the subspace $\operatorname{span}(\mathbf{Z}^g_c)$ is $\operatorname{proj}_c(\boldsymbol{z}) = \mathbf{U}_c\mathbf{U}^*_c\boldsymbol{z}$. The cosine similarity between $\boldsymbol{z}$ and $\operatorname{proj}_c(\boldsymbol{z})$ is then defined as the (unnormalized) probability of $\boldsymbol{x}$ belonging to class c, i.e.,

$$\begin{aligned}\hat{y}_c(\boldsymbol{x}) &= \mathbf{P}(\boldsymbol{x}\in c)\\ &\triangleq \begin{cases} \left\langle \boldsymbol{z}, \dfrac{\operatorname{proj}_c(\boldsymbol{z})}{\max(\|\operatorname{proj}_c(\boldsymbol{z})\|,\varepsilon)}\right\rangle \Big/ \displaystyle\sum_{c'=1}^{K}\left\langle \boldsymbol{z}, \dfrac{\operatorname{proj}_{c'}(\boldsymbol{z})}{\max(\|\operatorname{proj}_{c'}(\boldsymbol{z})\|,\varepsilon)}\right\rangle & \text{if } \|z\|\ge\epsilon,\\ 1/K & \text{if } \|z\|<\epsilon,\end{cases}\end{aligned} \tag{3.4}$$

where a small ε is chosen for numerical stability. The validation loss for $\boldsymbol{x}$ is then defined as the cross entropy between the predicted distribution $\hat{\boldsymbol{y}} = (\hat{y}_1,\ldots,\hat{y}_K)^T \in \mathbb{R}^K$ and the true label $y \in \{1,\ldots,K\}$. More specifically, let $\mathbf{Y}^v \in \mathbb{R}^{1\times|B_v|}$ and $\hat{\mathbf{Y}}^v \in \mathbb{R}^{K\times|B_v|}$ be the collection of true labels and predicted label distributions on the validation batch; then the validation loss is defined as

$$l_v(\mathbf{Y}^v,\hat{\mathbf{Y}}^v) = \frac{1}{|B_v|}\sum_{\boldsymbol{x}\in\mathbf{X}^v} H(\delta_{y(\boldsymbol{x})},\hat{\boldsymbol{y}}(\boldsymbol{x})) = -\frac{1}{|B_v|}\sum_{\boldsymbol{x}\in\mathbf{X}^v}\log\hat{y}(\boldsymbol{x})_{y(\boldsymbol{x})}, \tag{3.5}$$

where δ_y is the Dirac distribution at label y, and $H(\cdot,\cdot)$ is the cross entropy between two distributions. The empirical loss l on the training batch $\mathbf{X}$ is then defined as

$$l(\mathbf{X},\mathbf{Y}) = l([\mathbf{X}^g,\mathbf{X}^v],[\mathbf{Y}^g,\mathbf{Y}^v]) = l_g(\mathbf{Z}^g) + \lambda l_v(\mathbf{Y}^v,\hat{\mathbf{Y}}^v). \tag{3.6}$$

See Figure 1a for a visual illustration of the OLE-GRSVNet architecture. Because of the consistency between l_g and l_v, we have the following theorem.

Theorem 3.2. *For any $\lambda > 0$, and any geometry/validation splitting of $\mathbf{X} = [\mathbf{X}^g,\mathbf{X}^v]$ satisfying $\mathbf{X}^v$ containing at least one sample for each class, the empirical loss function defined in (3.6) is always nonnegative. Moreover, $l(\mathbf{X},\mathbf{Y}) = 0$ if and only if both of the following conditions hold:*
- *The features of the geometry batch belonging to different classes are orthogonal, i.e., $\operatorname{span}(\mathbf{Z}^g_c)\perp\operatorname{span}(\mathbf{Z}^g_{c'})\ \forall c\neq c'$.*
- *For every datum $\boldsymbol{x}\in\mathbf{X}^v_c$, i.e., $\boldsymbol{x}$ belongs to class c in the validation batch, its feature $\boldsymbol{z} = \Phi(\boldsymbol{x};\theta)$ belongs to $\operatorname{span}(\mathbf{Z}^g_c)$.*

Moreover, if $l<\infty$, then $\operatorname{rank}(\operatorname{span}(\mathbf{Z}^g_c)) \ge 1\ \forall c$; i.e., $\Phi(\cdot;\theta)$ does not trivially map data to $\mathbf{0}$.

Remark 3.3. The requirement that $\lambda > 0$ is crucial in Theorem 3.2 because otherwise the network can map every input into $\mathbf{0}$ and achieve the minimum. This is validated in our numerical experiments.

Remark 3.4. Since the low-dimensional subspaces spanned by features of different classes are orthogonal, the dimension of the ambient feature space has to be at least larger than the number of classes K. Empirically, we find out that the dimension of each subspace is typically less than three after convergence. Thus the minimally required feature dimension scales linearly with respect to K.

Remark 3.5. When the number of classes K is large, requiring $\mathbf{X}^g$ and $\mathbf{X}^v$ to contain samples from all K classes will result in a large batch size. If this is the case, a training batch $\mathbf{X}$ is sampled in the following way: We first randomly choose L out of K classes, where $L \ll K$. The training batch $\mathbf{X}$ is then sampled from these L classes, satisfying that both $\mathbf{X}^g$ and $\mathbf{X}^v$ contain at least one sample from each of the L classes.

After the training process has finished, we can then map the entire training data $\mathbf{X}^{\text{all}} = [\mathbf{X}_1^{\text{all}}, \ldots, \mathbf{X}_K^{\text{all}}]$ (or a random portion of $\mathbf{X}^{\text{all}}$) into their features $\mathbf{Z}^{\text{all}} = \Phi(\mathbf{X}^{\text{all}}; \theta^*)$, where θ^* is the learned parameter. The low-dimensional subspace $\text{span}(\mathbf{Z}_c^{\text{all}})$ for class c can be obtained via the SVD of $\mathbf{Z}_c^{\text{all}}$. The label of a test datum $\boldsymbol{x}$ is then determined by the distances between $\boldsymbol{z} = \Phi(\boldsymbol{x}; \theta^*)$ and $\{\text{span}(\mathbf{Z}_c^{\text{all}})\}_{c=1}^K$.

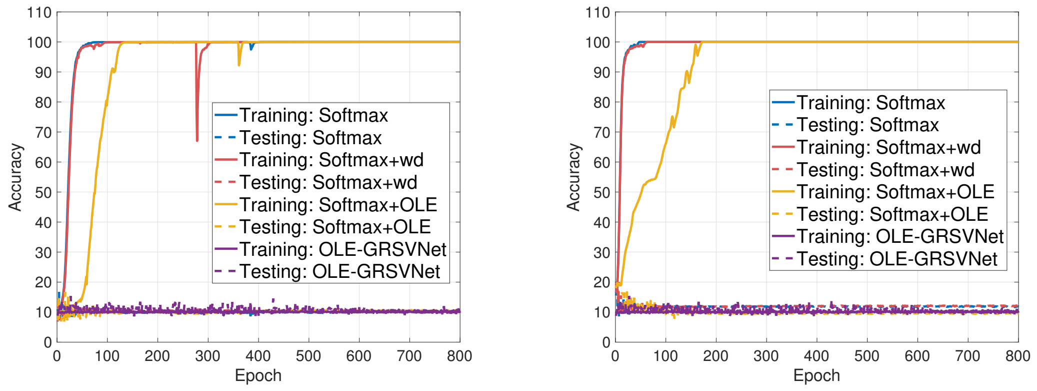
4. Two toy experiments.

Before stating the implementation details of OLE-GRSVNet, we first present two toy experiments to illustrate our proposed framework. We use VGG-11 [22] as the baseline architecture and compare the performance of the following four DNNs: (a) the baseline network with a softmax classifier (softmax), (b) VGG-11 with weight decay (softmax+wd), (c) VGG-11 regularized by penalizing the softmax loss with the OLE loss (softmax+OLE), and (d) OLE-GRSVNet.

We first train these four DNNs on the Street View House Numbers (SVHN) dataset [16] with the original data and labels without data augmentation. The test accuracy and the PCA embedding of the learned test features are shown in Figure 1. OLE-GRSVNet has the highest test accuracy among the comparing DNNs. Moreover, because of the consistency between the geometric loss and the validation loss, the test features produced by OLE-GRSVNet are even more discriminative than softmax+OLE: Features of the same class reside in a low-dimensional subspace, and different subspaces are (almost) orthogonal. Note that in Figure 1f, features of only four classes out of 10 have nonzero 3D embedding, although ideally it should be at most three because features of at least seven classes are orthogonal to the three leading principal components due to Theorem 3.2.

Next, we train the same networks, without changing hyperparameters, on the SVHN dataset with either (a) randomly generated labels or (b) random training data (Gaussian noise). We train the DNNs for 800 epochs to ensure their convergence, and the learning curves of training/testing accuracy are shown in Figure 2. Note that the baseline DNN, with either data-independent or conventional data-dependent regularization, can perfectly (over)fit the training data, while OLE-GRSVNet refuses to memorize the training data when there are no intrinsically learnable patterns.

In another experiment, we generate three classes of one-dimensional data in $\mathbb{R}^{10}$: The data points in the ith class are i.i.d. samples from the Gaussian distribution with the standard deviation in the ith coordinate 50 times larger than other coordinates. Each class has 500 data points, and we randomly shuffle the class labels after generation. We then train a



(a) Training/testing accuracy with random labels

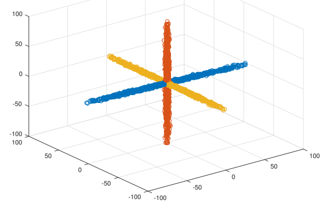
(b) Training/testing accuracy with random data

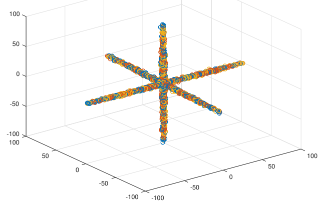
Figure 2. *Training and testing accuracy of different networks on the SVHN dataset with random labels or random data (Gaussian noise). Note that softmax, sotmax+wd, and softmax+OLE can all perfectly (over)fit the random training data or training data with random labels. Only the proposed OLE-GRSVNet refuses to fit the training data when there are no intrinsically learnable patterns.*

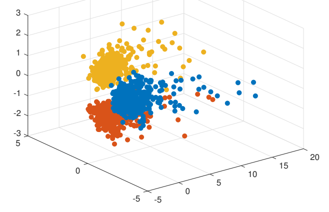
multilayer perceptron (MLP) with 128 neurons in each layer for 2000 epochs to classify these low-dimensional data with random labels. We found out that only three layers are needed to perfectly classify these data when using a softmax classifier. However, after incrementally adding more layers to the baseline MLP, we found out that OLE-GRSVNet still refuses to memorize the random labels even for a 100-layer MLP. This further suggests that OLE-GRSVNet refuses to memorize training data by brute force when there are no intrinsic patterns in the data. A visual illustration of this experiment is shown in Figure 3.

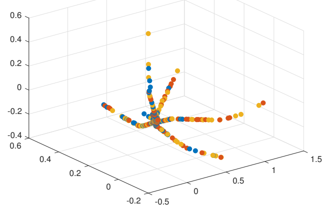
We provide an intuitive explanation for why OLE-GRSVNet can generalize significantly better than conventional DNNs when given true labeled data but refuses to memorize random data or random labels. For conventional DNNs with softmax activations, the class distribution of a single input datum is determined solely by itself, even if a minibatch $|B|$ of input data is used during training. On the other hand, in OLE-GRSVNets, the class distribution of an input datum is determined instead by the geometry of the entire training batch. Thus the training object in OLE-GRSVNets is no longer a single datum but the entire random batch of size $|B|$. Hence we conjecture that OLE-GRSVNets are implicitly conducting $O(N^{|B|})$-fold data augmentation, where N is the number of training data, while conventional data augmentation by the manipulation of the inputs, e.g., random cropping, flipping, etc., is typically $O(N)$. This poses a very interesting question: Does it mean that OLE-GRSVNets can also memorize random data if the baseline DNN has exponentially many model parameters? Or is it because of the learning algorithm (SGD) that prevents OLE-GRSVNets from learning a decision boundary too complicated for classifying random data? Answering this question will be the focus of our future research.

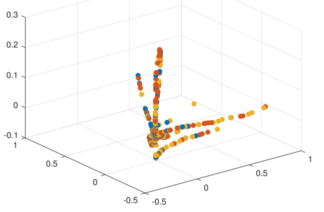
5. Implementation details of OLE-GRSVNet. Most of the operations in the computational graph of OLE-GRSVNet (Figure 1a) explained in section 3 are basic matrix operations.

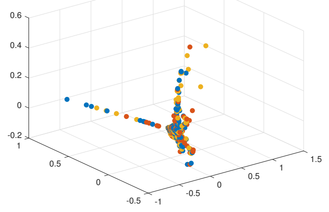

(a) Original three classes of one-dimensional data in $\mathbb{R}^{10}$

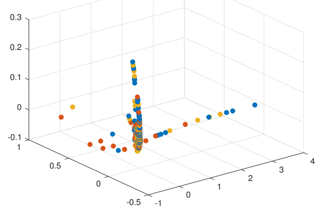

(b) Labels randomly shuffled

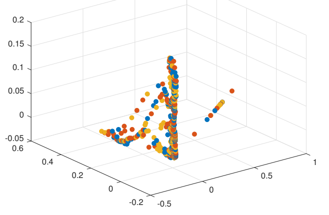

(c) Softmax with 3-layer MLP (accuracy = 100.0%)


(d) OLE-GRSVNet with 3-layer MLP (accuracy = 33.4%)


(e) OLE-GRSVNet with 5-layer MLP (accuracy = 33.4%)


(f) OLE-GRSVNet with 10-layer MLP (accuracy = 34.2%)


(g) OLE-GRSVNet with 20-layer MLP (accuracy = 35.0%)


(h) OLE-GRSVNet with 50-layer MLP (accuracy = 30.6%)

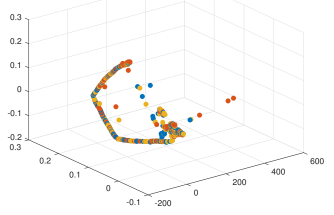

(i) OLE-GRSVNet with 100-layer MLP (accuracy = 29.8%)

Figure 3. *Visual illustration of the second toy experiment in section 4.* (a) *Three classes of one-dimensional data in* $\mathbb{R}^{10}$. (b) *Labels randomly shuffled.* (c)–(i) *Features extracted by the baseline MLP with a softmax classifier or OLE-GRSVNet. Only three layers of MLP are needed for conventional DNNs to perfectly memorize random labels. But even with* 100 *layers of MLP, OLE-GRSVNet still refuses to memorize the random labels because there are no intrinsically learnable patterns.*

The only two exceptions are the OLE loss ($\mathbf{Z}_g \to l^g((\mathbf{Z}^g))$) and the SVD ($\mathbf{Z}^g \to (\mathbf{U}_1, \dots, \mathbf{U}_K)$). We hereby specify their forward and backward propagations.

5.1. Backward propagation of the OLE loss. According to the definition of the OLE loss in (3.2), we only need to find a (sub)gradient of the nuclear norm to back-propagate the OLE loss. The characterization of the subdifferential of the nuclear norm is explained in [25]. More specifically, assuming $m \geq n$ for simplicity, let $\mathbf{U} \in \mathbb{R}^{m\times m}, \mathbf{\Sigma} \in \mathbb{R}^{m\times n}$, $\mathbf{V} \in \mathbb{R}^{n\times n}$ be the SVD of a rank-s matrix $\mathbf{A}$. Let $\mathbf{U} = [\mathbf{U}^{(1)}, \mathbf{U}^{(2)}]$, $\mathbf{V} = [\mathbf{V}^{(1)}, \mathbf{V}^{(2)}]$ be the partition of $\mathbf{U}$,

$\mathbf{V}$, respectively, where $\mathbf{U}^{(1)} \in \mathbb{R}^{m\times s}$ and $\mathbf{V}^{(1)} \in \mathbb{R}^{n\times s}$; then the subdifferential of the nuclear norm at $\mathbf{A}$ is

$$\partial\|\mathbf{A}\|_* = \left\{\mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \mathbf{U}^{(2)}\mathbf{W}\mathbf{V}^{(2)*} \quad \forall \mathbf{W} \in \mathbb{R}^{(m-s)\times(n-s)} \text{ with } \|\mathbf{W}\|_2 \le 1\right\}, \tag{5.1}$$

where $\|\cdot\|_2$ is the spectral norm. Note that to use (5.1), one needs to identify the rank-s column space of $\mathbf{A}$, i.e., $\mathrm{span}(\mathbf{U}^{(1)})$ to find a subgradient, which is not necessarily easy because of the existence of numerical error. The authors in [15] intuitively truncated the numerical SVD with a small parameter chosen a priori to ensure the numerical stability. We show in the following theorem using the backward stability of SVD [6, 8] that such concern is, in theory, not necessary.

Theorem 5.1. *Let $\mathbf{U}^\varepsilon, \mathbf{\Sigma}^\varepsilon, \mathbf{V}^\epsilon$ be the numerically computed reduced SVD of $\mathbf{A} \in \mathbb{R}^{m\times n}$, i.e., $\mathbf{U}^\varepsilon \in \mathbb{R}^{m\times n}$, $\mathbf{V}^\varepsilon \in \mathbb{R}^{n\times n}$, $(\mathbf{U}^\varepsilon + \delta\mathbf{U}^\varepsilon)\mathbf{\Sigma}^\varepsilon(\mathbf{V}^\varepsilon + \delta\mathbf{V}^\varepsilon)^* = \mathbf{A} + \delta\mathbf{A} = \mathbf{A}^\varepsilon$, and $\|\delta\mathbf{U}\|_2$, $\|\delta\mathbf{V}\|_2$, $\|\delta\mathbf{A}\|_2$ are all $O(\varepsilon)$, where ε is the machine error. If $\mathrm{rank}(\mathbf{A}) = s \le n$, and the smallest singular value $\sigma_s(\mathbf{A})$ of $\mathbf{A}$ satisfies $\sigma_s(\mathbf{A}) \ge \eta > 0$, we have*

$$\mathrm{d}(\mathbf{U}^\varepsilon\mathbf{V}^{\varepsilon*}, \partial\|\mathbf{A}\|_*) = O(\varepsilon/\eta). \tag{5.2}$$

However, in practice we did observe that using a small threshold (10^{-6} in this work) to truncate the numerical SVD can speed up the convergence, especially in the first few epochs of training. With the help of Theorem 5.1, we can easily find a stable subgradient of the OLE loss in (3.2).

5.2. Forward and backward propagation of $\mathbf{Z}^g \to (\mathbf{U_1}, \dots, \mathbf{U}_K)$. Unlike the computation of the subgradient in Theorem 5.1, we have to threshold the singular vectors of $\mathbf{Z}_c^g$ because the desired output $\mathbf{U}_c$ should be an orthonormal basis of the low-dimensional subspace $\mathrm{span}(\mathbf{Z}_c^g)$. In the forward propagation, we threshold the singular vectors $\mathbf{U}_c$ such that the smallest singular value is at least $1/10$ of the largest singular value.

As for the backward propagation, one needs to know the Jacobian of the SVD, which has been explained in [17]. Typically, for a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, computing the Jacobian of the SVD of $\mathbf{A}$ involves solving a total of $O(n^4)$ 2×2 linear systems. We have not implemented the backward propagation of the SVD in this work because this involves technical implementation with CUDA API. In our current implementation, the node $(\mathbf{U}_1, \dots, \mathbf{U}_K)$ is detached from the computational graph during back propagation; i.e., the validation loss l_v is only propagated back through the path $l_v \to \hat{\mathbf{Y}}^v \to \mathbf{Z}^v \to \theta$. Our rational is this: The validation loss l_v can be propagated back through two paths: $l_v \to \hat{\mathbf{Y}}^v \to \mathbf{Z}^v \to \theta$ and $l_v \to \hat{\mathbf{Y}}^v \to (\mathbf{U_1}, \dots, \mathbf{U}_K) \to \mathbf{Z}^g \to \theta$. The first path will modify θ so that $\mathbf{Z}_c^v$ moves closer to $\mathbf{U}_c$, while the second path will move $\mathbf{U}_c$ closer to $\mathbf{Z}_c^v$. Cutting off the second path when computing the gradient might decrease the speed of convergence, but numerical experiments suggest that the training process is still well behaved under such simplification. With such simplification, the only extra computation is the SVD of a minibatch of features, which is negligible (<5%) when compared to the time of training the baseline network.

6. Experimental results.

In this section, we show the superiority of OLE-GRSVNet when compared to conventional DNNs in two aspects: (a) It has greater generalization power when trained on true data with real labels, and the improvement of the test accuracy is especially significant when the number of training examples is small. (b) Unlike conventionally regularized DNNs, OLE-GRSVNet refuses to memorize the training samples when given random training data or random labels. This gives OLE-GRSVNets a significant advantage when dealing with training data with corrupted labels.

The following benchmark datasets are chosen to evaluate the effectiveness of the different regularizations:

- **MNIST**. The MNIST dataset contains 28×28 grayscale images of digits from 0 to 9. There are 60,000 training samples and 10,000 testing samples. No data augmentation was used.
- **SVHN**. The Street View House Numbers (SVHN) dataset contains 32×32 RGB images of digits from 0 to 9. The training and testing set contain 73,257 and 26,032 images, respectively. No data augmentation was used.
- **CIFAR**. The CIFAR dataset [13] contains 32×32 RGB images of 10 classes, with 50,000 images for training and 10,000 images for testing. We use "**CIFAR+**" to denote experiments on CIFAR with data augmentation: 4 pixel padding, 32×32 random cropping and horizontal flipping.
- **Tiny ImageNet**. The Tiny ImageNet (tImageNet) dataset.[1]

We use an experimental setup similar to that in section 4; i.e., the same four modifications to the baseline DNNs are considered: (a) **Softmax**, (b) **Softmax+wd**, (c) **Softmax+OLE**, and (d) **OLE-GRSVNet**. The performance of the different regularizations is evaluated on the following baseline architectures:[2]

6.1. Training details.

All networks are trained from scratch with the "Xavier" initialization [7]. SGD with Nesterov momentum 0.9 is used for the optimization, and the batch size is set to 200 (a 100/100 split for geometry/validation batch is used in OLE-GRSVNet). Without explicitly mentioning it, we set the initial learning rate to 0.01, and decrease it tenfold at 50% and 75% of the total training epochs. The numbers of training epochs for the experiments with true labels are reported in Table 2. In order to ensure the convergence of

[1]The dataset is available at https://tiny-imagenet.herokuapp.com/ contains 64×64 RGB images from 200 classes. Each class has 500 training and 50 test data.

[2]The code is available at https://services.math.duke.edu/~zhu/software.html:

- **VGG** [22]. The VGG-11, 16, 19 architectures are composed of five blocks of convolutional layers with ReLU activations and Batch Normalization (BN). Five Max-Pooling layers are used to gradually decrease the spatial dimension of the input images.
- **LeNet** [14]. This is a simple 5-layer network consisting of two convolutional layers, two Max-Pooling layers, and a final fully connected layer. We use this architecture mainly for the MNIST dataset.
- **ResNet** [10]. The basic building block, i.e., the residual block, of the ResNet consists of a concatenation of seven operations: *conv.-BN-conv.-BN-conv.-BN-ReLU*. The input of each residual block is added to the output through a short connection. The ResNet architecture with 27 residual blocks is used for the experiments.
- **DenseNet** [12]. DenseNets are composed of three DenseNet blocks, each of which contains multiple densely connected convolutional blocks with a small number of output channels. The DenseNet with 40 layers of operations and a growth rate of 12 is used for the experiments.
- **CNN-5**. This is a plain convolutional neural network consisting of five convolution and Max-Pooling layers. This network is mainly used for the experiments in subsection 6.4.

The detailed network architectures are summarized in Table 1.

Table 1

Summary of the network architectures. The last fully connected layer, the size of which depends on the number of classes, is omitted from the table. **C***X: Convolutional block with the kernel size set to* 3×3*.* **MP***: Max Pooling with kernel size* 2×2 *and stride* 2*.* **FC***X: Fully connected layer.* **R***X/Y: Residual block.* **AP***: Global Average Pooling.* **D***X/G: DenseNet block. For all the modules, X is the number of output channels, Y is the number of inner channels for* **R***, and G is the growth rate for* **D** *blocks. See the text for detailed block definitions.*

VGG-11	**C**64-**MP**-**C**128-**MP**-**C**256(×2)-**MP**-**C**512(×2)-**MP**-**C**512(×2)-**MP**-**FC**512
VGG-16	**C**64(×2)-**MP**-**C**128(×2)-**MP**-**C**256(×3)-**MP**-**C**512(×3)-**MP**-**C**512(×3)-**MP**-**FC**512
VGG-19	**C**64(×2)-**MP**-**C**128(×2)-**MP**-**C**256(×4)-**MP**-**C**512(×4)-**MP**-**C**512(×4)-**MP**-**FC**512
LeNet	**C**6-**MP**-**C**16-**MP**-**FC**120
ResNet	**C**16-**R**64/16(×9)-**R**128/32(×9)-**R**256/64(×9)-**BN**-**ReLU**-**AP**
DenseNet	**C**32-**MP**-**C**64-**MP**-**C**128-**MP**-**C**256-**C**256-**MP**
CNN-5	**C**64-**MP**-**C**128-**MP**-**C**256-**MP**-**C**512-**MP**-**C**512-**MP**-**FC**512

Table 2

Hyperparameters and numbers of the training epochs used for the experiments with true labels on the entire data. For every entry $\nu/\lambda/N$*,* ν *is the weight for the OLE loss in "softmax+OLE,"* λ *is the weight of the validation loss in OLE-GRSVNets* (3.6)*, and* N *is the number of training epochs.*

Dataset	LeNet	VGG-11	VGG-16	VGG-19	ResNet	DenseNet	CNN-5
MNIST	0.1/5/100	0.5/10/100	-	-	-	-	0.1/5/100
SVHN	-	0.5/5/160	-	-	-	0.5/10/300	0.25/5/100
CIFAR	-	0.5/5/200	0.5/5/300	0.25/1/400	0.25/10/300	-	0.1/5/100
CIFAR+	-	0.5/5/200	0.5/5/300	0.25/1/400	-	-	-
tImageNet	-	-	-	0.1/1/400	-	-	-

SGD, all networks are trained for 800 epochs for the experiments with random labels. The mean accuracy after five independent trials is reported.

As for the hyperparameters, the weight decay parameter, if used, is always set to $\mu = 10^{-4}$. The weight for the OLE loss in "softmax+OLE" and the parameter λ in (3.6) are determined by cross-validation and reported in Table 2.

6.2. Testing/training performance when trained on the entire datasets with real or random labels. Table 3 reports the performance of the networks trained on the entire datasets with real or randomly generated labels. The numbers in the upper block are the percentage accuracies on the *testing data* when networks are trained with *real labels*. The numbers in the lower block are the accuracies on the *training data* when networks are trained with *random labels*. Accuracies on the training data with real labels (always 100%) and accuracies on the test data with random labels (always close to random guess) are omitted from the table. As we can see, similar to the experiment in section 4, when trained with real labels, OLE-GRSVNet exhibits better generalization performance than the competing networks. This is because the feature geometry is better enforced through consistent self-validation in each step

Table 3

Testing or training accuracies when trained on the entire datasets with real or random labels. The numbers in the upper block are the percentage accuracies on the testing data *when networks are trained with* real labels. *The numbers in the lower block are the accuracies on the* training data *when networks are trained with* random labels. *The means and standard deviations after five independent trials are reported. ResNet*-0.01 *and ResNet*-0.1*, respectively, denote the results of the DNNs with the ResNet baseline architecture trained with the initial learning rates* 0.01 *and* 0.1.

		Testing accuracy (%) when trained with real label			
Dataset	Architecture	Softmax	Softmax+wd	Softmax+OLE	OLE-GRSVNet
MNIST	LeNet	99.25 ± 0.09	99.31 ± 0.07	99.37 ± 0.11	$\mathbf{99.41 \pm 0.08}$
MNIST	VGG-11	99.40 ± 0.03	99.47 ± 0.03	99.49 ± 0.02	$\mathbf{99.57 \pm 0.02}$
SVHN	VGG-11	93.10 ± 0.04	93.73 ± 0.05	94.04 ± 0.08	$\mathbf{94.75 \pm 0.06}$
SVHN	DenseNet	93.24 ± 0.26	93.89 ± 0.11	93.35 ± 0.16	$\mathbf{95.08 \pm 0.26}$
CIFAR	VGG-11	81.81 ± 0.12	81.87 ± 0.10	82.04 ± 0.14	$\mathbf{85.29 \pm 0.10}$
CIFAR	VGG-16	83.37 ± 0.13	83.97 ± 0.13	84.35 ± 0.14	$\mathbf{87.44 \pm 0.11}$
CIFAR	VGG-19	83.56 ± 0.19	84.21 ± 0.17	84.71 ± 0.19	$\mathbf{86.69 \pm 0.20}$
CIFAR	ResNet-0.01	75.79 ± 0.18	77.90 ± 0.19	78.53 ± 0.23	$\mathbf{84.40 \pm 0.16}$
CIFAR	ResNet-0.1	81.93 ± 0.29	$\mathbf{85.38 \pm 0.27}$	82.60 ± 0.77	85.16 ± 0.21
CIFAR+	VGG-11	89.52 ± 0.15	89.68 ± 0.16	90.04 ± 0.20	$\mathbf{90.58 \pm 0.17}$
CIFAR+	VGG-16	91.21 ± 0.16	91.29 ± 0.19	91.40 ± 0.11	$\mathbf{92.15 \pm 0.12}$
CIFAR+	VGG-19	91.19 ± 0.22	91.53 ± 0.19	$\mathbf{91.67 \pm 0.24}$	91.65 ± 0.21
tImageNet	VGG-19	45.28 ± 0.21	45.89 ± 0.18	46.36 ± 0.50	$\mathbf{47.51 \pm 0.47}$

		Training accuracy (%) when trained with random label			
Dataset	VGG	Softmax	Softmax+wd	Softmax+OLE	OLE-GRSVNet
MNIST	LeNet	99.99 ± 0.01	100.00 ± 0.00	99.97 ± 0.01	$\mathbf{9.83 \pm 0.51}$
MNIST	VGG-11	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	$\mathbf{9.93 \pm 0.23}$
SVHN	VGG-11	99.99 ± 0.01	100.00 ± 0.00	99.99 ± 0.01	$\mathbf{9.75 \pm 0.47}$
SVHN	DenseNet	100.00 ± 0.00	99.99 ± 0.01	99.96 ± 0.02	$\mathbf{10.26 \pm 0.43}$
CIFAR	VGG-11	100.00 ± 0.00	100.00 ± 0.00	99.95 ± 0.02	$\mathbf{9.97 \pm 0.21}$
CIFAR	VGG-16	100.00 ± 0.00	99.99 ± 0.01	99.96 ± 0.02	$\mathbf{10.13 \pm 0.33}$
CIFAR	VGG-19	99.99 ± 0.00	99.97 ± 0.01	99.96 ± 0.02	$\mathbf{9.86 \pm 0.45}$
CIFAR	ResNet-0.01	99.96 ± 0.02	99.97 ± 0.01	99.96 ± 0.02	$\mathbf{9.67 \pm 0.51}$
CIFAR+	VGG-11	99.98 ± 0.01	99.98 ± 0.01	99.93 ± 0.03	$\mathbf{10.05 \pm 0.39}$
CIFAR+	VGG-16	99.96 ± 0.02	99.96 ± 0.02	99.92 ± 0.04	$\mathbf{9.94 \pm 0.45}$
CIFAR+	VGG-19	99.96 ± 0.02	99.95 ± 0.03	99.91 ± 0.04	$\mathbf{10.07 \pm 0.32}$
tImageNet	VGG-19	99.97 ± 0.02	99.96 ± 0.02	99.95 ± 0.02	$\mathbf{0.48 \pm 0.04}$

Table 4

The effect of the training batch size $|B|$ on the test accuracy of the OLE-GRSVNet. The mean accuracies after three independent trials are reported.

		Testing accuracy (%) of the OLE-GRSVNet with different $|B|$					
Dataset	Architecture	$|B| = 60$	$|B| = 80$	$|B| = 140$	$|B| = 200$	$|B| = 300$	$|B| = 400$
MNIST	LeNet	99.27	99.36	99.35	99.41	99.42	99.39
SVHN	VGG-11	93.96	94.48	94.65	94.75	94.94	94.87
CIFAR	VGG-11	84.01	84.57	85.16	85.29	85.21	85.27

of the training (see Figure 1 and Theorem 3.2.) However, when trained with random labels, OLE-GRSVNet refuses to memorize the training samples like the other networks because there are no intrinsically learnable patterns. This is still the case even if we increase the number of training epochs to 2000. We did not report the results of "softmax+centerloss" [26] and "softmax+LDMNet" [29] because they have entirely different geometric constraint, but the story is the same: they are both capable of perfectly (over)fitting random labels. As mentioned in section 5, the extra computational time during the training of the OLE-GRSVNet, i.e., the SVD of the features of a random batch, is negligible (<5%) when compared to that of training the VGG-11 baseline, and even more so for more complicated baseline DNNs.

It is worth mentioning that the performance of the proposed OLE-GRSVNet seems to be more stable with respect to the choice of the initial learning rate. As can be seen from Table 3, when ResNets are trained on the CIFAR dataset with the initial learning rate set to 0.1, the proposed OLE-GRSVNet (test accuracy = 85.16%) slightly underperforms the data-independent weight decay regularization (test accuracy = 85.38%.) However, if we change the initial learning rate to 0.01, all networks except for the OLE-GRSVNet experience a significant decrease in the test accuracy. This might suggest that the OLE-GRSVNet is less susceptible to converging to a "nongeneralizable" minimum when a small initial learning rate is chosen. Later in subsection 6.3, we will show that when ResNets are trained on a small fraction (10%, 20%, and 40%) of the CIFAR training data, the OLE-GRSVNet always significantly outperforms other competing networks, no matter which initial learning rate is chosen.

The effect of the training batch size $|B|$ on the performance of the OLE-GRSVNet is shown in Table 4. As we can see, the test accuracy generally increases before reaching a plateau as $|B|$ gets larger. This is because more training data are used to resolve the low-dimensional subspaces in each training batch as the batch size increases. This empirical finding also corroborates our claim in section 4 that OLE-GRSVNets are implicitly conducting $O(N^{|B|})$-fold data augmentation.

6.3. Testing performance with limited training data. We next examine the efficacy of different regularizations when the training data are scarce. Table 5 displays the test accuracies of the different networks when trained on a small fraction (10%, 20%, and 40%) of the entire training data. We use the same hyperparameters as those in subsection 6.2. It is clear from Table 5 that OLE-GRSVNet significantly outperforms other competing networks when trained with very limited samples. We note that, unlike in subsection 6.2, when the networks with the

Table 5

*Testing accuracies when the networks are trained on a small fraction of the entire datasets. The means and standard deviations after three independent trials are reported. ResNet-*0.01 *and ResNet-*0.1*, respectively, denote the results of the DNNs with the ResNet baseline architecture trained with the initial learning rates* 0.01 *and* 0.1*.*

		Testing accuracy (%) when trained on 10% of the entire data			
Dataset	Architecture	Softmax	Softmax+wd	Softmax+OLE	OLE-GRSVNet
MNIST	LeNet	98.00 ± 0.06	98.10 ± 0.09	98.39 ± 0.09	$\mathbf{98.44 \pm 0.10}$
SVHN	VGG-11	85.75 ± 0.26	85.99 ± 0.25	86.72 ± 0.29	$\mathbf{88.16 \pm 0.27}$
CIFAR	VGG-11	61.67 ± 0.24	62.54 ± 0.20	62.45 ± 0.17	$\mathbf{66.66 \pm 0.22}$
CIFAR	ResNet-0.01	44.44 ± 0.23	47.73 ± 0.26	46.55 ± 0.32	$\mathbf{57.23 \pm 0.21}$
CIFAR	ResNet-0.1	49.21 ± 0.24	50.39 ± 0.29	49.31 ± 0.32	$\mathbf{60.50 \pm 0.33}$
		Testing accuracy (%) when trained on 20% of the entire data			
Dataset	Architecture	Softmax	Softmax+wd	Softmax+OLE	OLE-GRSVNet
MNIST	LeNet	98.48 ± 0.02	98.56 ± 0.07	98.81 ± 0.12	$\mathbf{98.84 \pm 0.04}$
SVHN	VGG-11	88.72 ± 0.20	88.83 ± 0.19	89.84 ± 0.21	$\mathbf{90.78 \pm 0.23}$
CIFAR	VGG-11	68.70 ± 0.14	68.99 ± 0.16	68.84 ± 0.21	$\mathbf{74.50 \pm 0.17}$
CIFAR	ResNet-0.01	55.20 ± 0.24	55.87 ± 0.26	55.73 ± 0.27	$\mathbf{66.28 \pm 0.24}$
CIFAR	ResNet-0.1	59.50 ± 0.31	63.30 ± 0.29	60.67 ± 0.38	$\mathbf{67.62 \pm 0.26}$
		Testing accuracy (%) when trained on 40% of the entire data			
Dataset	Architecture	Softmax	Softmax+wd	Softmax+OLE	OLE-GRSVNet
MNIST	LeNet	98.93 ± 0.10	98.99 ± 0.02	99.14 ± 0.06	$\mathbf{99.17 \pm 0.02}$
SVHN	VGG-11	90.84 ± 0.16	91.14 ± 0.21	91.99 ± 0.18	$\mathbf{92.83 \pm 0.25}$
CIFAR	VGG-11	74.37 ± 0.22	74.79 ± 0.17	75.21 ± 0.24	$\mathbf{79.92 \pm 0.14}$
CIFAR	ResNet-0.01	63.80 ± 0.14	64.23 ± 0.19	64.55 ± 0.19	$\mathbf{74.29 \pm 0.21}$
CIFAR	ResNet-0.1	73.20 ± 0.22	74.55 ± 0.31	73.57 ± 0.41	$\mathbf{77.88 \pm 0.38}$

ResNet baseline architecture are trained on a fraction of the CIFAR dataset, OLE-GRSVNet achieves substantially better test performance compared to other regularizations no matter which initial learning rate is chosen.

6.4. Testing performance with corrupted labels. Finally, we demonstrate the practical usage of the proposed OLE-GRSVNet in a realistic setting where part of the training samples are wrongly labeled. To achieve this, we randomly choose 40% of the training data and replace their real labels with random labels. All networks with the same CNN-5 baseline architecture are first trained on the corrupted dataset for 100 epochs, after which each network produces its own list of predicted "bad" training samples, i.e., the ones that the network refuses to fit. Each network is then trained again on the "purified" training data after discarding its predicted "bad" training samples.

Table 6

Testing accuracies when the networks are trained with 40% corrupted labels. The means and standard deviations after three independent trials are reported. See subsection 6.4 for the detailed experimental setup.

		Testing accuracy (%) when trained with 40% corrupted labels			
Dataset	Architecture	Softmax	Softmax+wd	Softmax+OLE	OLE-GRSVNet
MNIST	CNN-5	76.26 ± 0.31	77.69 ± 0.33	79.23 ± 0.29	$\mathbf{85.13 \pm 0.41}$
SVHN	CNN-5	68.21 ± 0.28	67.70 ± 0.33	68.79 ± 0.39	$\mathbf{85.24 \pm 0.52}$
CIFAR	CNN-5	54.39 ± 0.42	54.08 ± 0.28	55.64 ± 0.33	$\mathbf{61.14 \pm 0.23}$

Table 6 reports the performance of the networks in the aforementioned setting. It is clear that the OLE-GRSVNet achieves much better test performance, the reason for which is that it correctly detects part of the corrupted training data by refusing to memorize them through the first round of training. On the contrary, all other networks suffer from low test accuracy by (over)fitting the "bad" training samples.

We do want to mention that the OLE-GRSVNet is not able to perfectly detect all corrupted labels in the training data, i.e., it still can "memorize" some "bad" training samples. This is especially the case when baseline architectures with huge capacities are trained on complicated datasets, e.g., CIFAR. We believe this is unavoidable because of the rich information intrinsic in such datasets.

7. Conclusion and future work.

We proposed a potential framework, GRSVNet, for data-dependent DNN regularization. The core idea is the self-validation of the enforced geometry on a separate batch using a validation loss consistent with the geometric loss, so that the predicted label distribution has a meaningful geometric interpretation. In particular, we study a special case of GRSVNet, the OLE-GRSVNet, which is capable of producing highly discriminative features: Samples from the same class belong to a low-dimensional subspace, and the subspaces for different classes are orthogonal. When trained on benchmark datasets with real labels, OLE-GRSVNet achieves better test accuracy when compared to DNNs with different regularizations sharing the same baseline architecture. More importantly, unlike conventional DNNs, OLE-GRSVNet refuses to memorize and overfit the training data when trained on random labels or random data. This suggests that OLE-GRSVNet effectively reduces the memorizing capacity of DNNs, and it only extracts intrinsically learnable patterns from the data.

Although we have mainly focused on the special case of the GRSVNet framework where the geometric constraint is orthogonal low-dimensional subspaces, any consistent geometry/validation pair can fit into this potential framework. For example, we can enforce the features to sample low-dimensional manifolds as in [29] and choose a smooth manifold interpolation function for label propagation as the consistent validation loss. Moreover, the similar idea of imposing geometry consistent validation can be extended to other vision tasks, such as motion analysis and depth estimation.

We provided some intuitive explanation as to why GRSVNet generalizes well on real data and refuses overfitting random data, but there are still open questions to be answered. For example, what is the minimum representational capacity of the baseline DNN (i.e., number of layers and number of units) to make even GRSVNets trainable on random data? Or is it because of the learning algorithm (SGD) that prevents GRSVNets from learning a decision boundary that is too complicated for random samples? Moreover, we still have not answered why conventional DNNs, while fully capable of memorizing random data by brute force, typically find generalizable solutions on real data. These questions will be the focus of our future work.

Appendix A. Proof of Theorem 3.1.

It suffices to prove the case when $K = 2$, as the case for larger K can be proved by induction. In order to simplify the notation, we restate the original theorem for $K = 2$.

Theorem A.1. *Suppose* $\mathbf{A} \in \mathbb{R}^{N\times m}$ *and* $\mathbf{B} \in \mathbb{R}^{N\times n}$ *are two matrices of the same row dimension, and* $[\mathbf{A}, \mathbf{B}] \in \mathbb{R}^{N\times(m+n)}$ *is the concatenation of* $\mathbf{A}$ *and* $\mathbf{B}$*. Let* $\|\cdot\|_*$ *be the nuclear norm of a matrix defined as*

$$\|\mathbf{A}\|_* = \mathrm{Tr}\left(|\mathbf{A}|\right), \quad \textit{where} \ \ |\mathbf{A}| = (\mathbf{A}^*\mathbf{A})^{\frac{1}{2}}. \tag{A.1}$$

Then we have

$$\|[\mathbf{A}, \mathbf{B}]\|_* \le \|\mathbf{A}\|_* + \|\mathbf{B}\|_*. \tag{A.2}$$

Moreover, the equality holds if and only if $\mathbf{A}^*\mathbf{B} = \mathbf{0}$*, i.e., the column spaces of* $\mathbf{A}$ *and* $\mathbf{B}$ *are orthogonal:*

Proof. Let $\left[\begin{smallmatrix}\mathbf{E} & \mathbf{G}\\ \mathbf{G}^* & \mathbf{F}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{A}^*\mathbf{A} & \mathbf{A}^*\mathbf{B}\\ \mathbf{B}^*\mathbf{A} & \mathbf{B}^*\mathbf{B}\end{smallmatrix}\right]^{\frac{1}{2}} = |[\mathbf{A}, \mathbf{B}]|$ be a symmetric positive semidefinite matrix. We have

$$\begin{cases} |\mathbf{A}|^2 = \mathbf{A}^*\mathbf{A} = \mathbf{E}^2 + \mathbf{G}\mathbf{G}^*, \\ |\mathbf{B}|^2 = \mathbf{B}^*\mathbf{B} = \mathbf{F}^2 + \mathbf{G}^*\mathbf{G}, \\ \mathbf{A}^*\mathbf{B} = \mathbf{E}\mathbf{G} + \mathbf{G}\mathbf{F}. \end{cases} \tag{A.3}$$

Suppose $\{\boldsymbol{a}_i\}_{i=1}^m$, $\{\boldsymbol{b}_i\}_{i=1}^n$ are the orthonormal eigenvectors of $|\mathbf{A}|$, $|\mathbf{B}|$, respectively; then

$$\||\mathbf{A}|\boldsymbol{a}_i\|^2 = \left\langle |\mathbf{A}|^2\boldsymbol{a}_i, \boldsymbol{a}_i\right\rangle = \left\langle (\mathbf{E}^2 + \mathbf{G}\mathbf{G}^*)\boldsymbol{a}_i, \boldsymbol{a}_i\right\rangle = \|\mathbf{E}\boldsymbol{a}_i\|^2 + \|\mathbf{G}^*\boldsymbol{a}_i\|^2, \tag{A.4}$$

$$\||\mathbf{B}|\boldsymbol{b}_i\|^2 = \left\langle |\mathbf{B}|^2\boldsymbol{b}_i, \boldsymbol{b}_i\right\rangle = \left\langle (\mathbf{F}^2 + \mathbf{G}^*\mathbf{G})\boldsymbol{b}_i, \boldsymbol{b}_i\right\rangle = \|\mathbf{F}\boldsymbol{b}_i\|^2 + \|\mathbf{G}\boldsymbol{b}_i\|^2. \tag{A.5}$$

We thus have the following inequality chain that proves (A.2):

$$\begin{aligned} \|\mathbf{A}\|_* + \|\mathbf{B}\|_* &= \mathrm{Tr}(|\mathbf{A}|) + \mathrm{Tr}(|\mathbf{B}|) = \sum_{i=1}^m \langle |\mathbf{A}|\boldsymbol{a}_i, \boldsymbol{a}_i\rangle + \sum_{i=1}^n \langle |\mathbf{B}|\boldsymbol{b}_i, \boldsymbol{b}_i\rangle \\ &= \sum_{i=1}^m \||\mathbf{A}|\boldsymbol{a}_i\| + \sum_{i=1}^n \||\mathbf{B}|\boldsymbol{b}_i\| \\ &= \sum_{i=1}^m \left(\|\mathbf{E}\boldsymbol{a}_i\|^2 + \|\mathbf{G}^*\boldsymbol{a}_i\|^2\right)^{\frac{1}{2}} + \sum_{i=1}^n \left(\|\mathbf{F}\boldsymbol{b}_i\|^2 + \|\mathbf{G}\boldsymbol{b}_i\|^2\right)^{\frac{1}{2}} \\ &\ge \sum_{i=1}^m \|\mathbf{E}\boldsymbol{a}_i\| + \sum_{i=1}^n \|\mathbf{F}\boldsymbol{b}_i\| \ge \sum_{i=1}^m \langle \mathbf{E}\boldsymbol{a}_i, \boldsymbol{a}_i\rangle + \sum_{i=1}^n \langle \mathbf{F}\boldsymbol{b}_i, \boldsymbol{b}_i\rangle \\ &= \mathrm{Tr}(\mathbf{E}) + \mathrm{Tr}(\mathbf{F}) = \mathrm{Tr}(|[\mathbf{A}, \mathbf{B}]|) = \|[\mathbf{A}, \mathbf{B}]\|_*. \end{aligned} \tag{A.6}$$

We next show that the equality holds if and only if $\mathbf{A}^*\mathbf{B} = \mathbf{0}$:

- If $\mathbf{A}^*\mathbf{B} = \mathbf{0}$, then

$$\text{(A.7)} \qquad \|[\mathbf{A},\mathbf{B}]\|_* = \operatorname{Tr}(|[\mathbf{A},\mathbf{B}]|) = \operatorname{Tr}\left(\begin{bmatrix}\mathbf{A}^*\mathbf{A} & \mathbf{0}\\ \mathbf{0} & \mathbf{B}^*\mathbf{B}\end{bmatrix}^{\frac{1}{2}}\right) = \operatorname{Tr}\left(\begin{bmatrix}|\mathbf{A}| & \mathbf{0}\\ \mathbf{0} & |\mathbf{B}|\end{bmatrix}\right)$$
$$= \operatorname{Tr}(|\mathbf{A}|) + \operatorname{Tr}(|\mathbf{B}|) = \|\mathbf{A}\|_* + \|\mathbf{B}\|_*.$$

- If $\|[\mathbf{A},\mathbf{B}]\|_* = \|\mathbf{A}\|_* + \|\mathbf{B}\|_*$, then both of the inequalities in the chain (A.6) must be equalities. Note that the first inequality is an equality only if $\mathbf{G} = \mathbf{0}$. This combined with the last equation in (A.3) implies

$$\text{(A.8)} \qquad \mathbf{A}^*\mathbf{B} = \mathbf{E}\mathbf{G} + \mathbf{G}\mathbf{F} = \mathbf{0}. \qquad \blacksquare$$

Appendix B. Proof of Theorem 3.2.

Proof. First, l is defined in (3.6) as

$$\text{(B.1)} \qquad l(\mathbf{X},\mathbf{Y}) = l([\mathbf{X}^g,\mathbf{X}^v],[\mathbf{Y}^g,\mathbf{Y}^v]) = l_g(\mathbf{Z}^g) + \lambda l_v(\mathbf{Y}^v,\hat{\mathbf{Y}}^v).$$

The nonnegativity of $l_g(\mathbf{Z}^g)$ is guaranteed by Theorem 3.1. The validation loss $l_v(\mathbf{Y}^v,\hat{\mathbf{Y}}^v)$ is also nonnegative since it is the average (over the validation batch) of the cross entropy losses:

$$\text{(B.2)} \qquad l_v(\mathbf{Y}^v,\hat{\mathbf{Y}}^v) = \frac{1}{|B_v|}\sum_{\boldsymbol{x}\in\mathbf{X}^v} H(\delta_y,\hat{\boldsymbol{y}}) = -\frac{1}{|B_v|}\sum_{\boldsymbol{x}\in\mathbf{X}^v}\log\hat{y}_y.$$

Therefore, $l = l_g + \lambda l_v$ is also nonnegative.

Next, for a given $\lambda > 0$, $l(\mathbf{X},\mathbf{Y})$ obtains its minimum value zero if and only if both $l_g(\mathbf{Z}^g)$ and $l_v(\mathbf{Y}^v,\hat{\mathbf{Y}}^v)$ are zeros:

- By Theorem 3.1, $l_g(\mathbf{Z}^g) = 0$ if and only if $\operatorname{span}(\mathbf{Z}^g_c)\perp\operatorname{span}(\mathbf{Z}^g_{c'})\ \forall c \neq c'$.
- According to (B.2), $l_v(\mathbf{Y}^v,\hat{\mathbf{Y}}^v) = 0$ if and only if $\hat{\boldsymbol{y}}(\boldsymbol{x}) = \delta_y\ \forall\boldsymbol{x}\in\mathbf{X}^v$, i.e., for every $\boldsymbol{x}\in\mathbf{X}^v_c$, its feature $\boldsymbol{z} = \Phi(\boldsymbol{x};\theta)$ belongs to $\operatorname{span}(\mathbf{Z}^g_c)$.

Last, we want to prove that if $\lambda > 0$, and $\mathbf{X}^v$ contains at least one sample for each class, then $\operatorname{rank}(\operatorname{span}(\mathbf{Z}^g_c)) \geq 1$ for any $c \in \{1,\dots,K\}$.

If not, then there exists $c \in \{1,\dots,K\}$ such that $\operatorname{rank}(\operatorname{span}(\mathbf{Z}^g_c)) = 0$. Let $\boldsymbol{x}\in\mathbf{X}^v$ be a validation datum belonging to class $y = c$. The predicted probability of $\boldsymbol{x}$ belonging to class c is defined as in (3.4):

$$\text{(B.3)} \qquad \hat{y}_c = \mathbf{P}(\boldsymbol{x}\in c) \triangleq \left\langle \boldsymbol{z}, \frac{\operatorname{proj}_c(\boldsymbol{z})}{\max\left(\|\operatorname{proj}_c(\boldsymbol{z})\|,\varepsilon\right)}\right\rangle \Bigg/ \sum_{c'=1}^{K}\left\langle \boldsymbol{z}, \frac{\operatorname{proj}_{c'}(\boldsymbol{z})}{\max\left(\|\operatorname{proj}_{c'}(\boldsymbol{z})\|,\varepsilon\right)}\right\rangle = 0.$$

Thus we have

$$\text{(B.4)} \qquad l \geq \lambda l_v = -\frac{\lambda}{|B_v|}\sum_{\boldsymbol{x}\in\mathbf{X}^v}\log\hat{y}_y \geq -\frac{\lambda}{|B_v|}\log\hat{y}(\boldsymbol{x})_c = +\infty. \qquad \blacksquare$$

Appendix C. Proof of Theorem 5.1.

As mentioned in subsection 5.1, we assume for simplicity that $m \geq n$. We first need the following lemma.

Lemma C.1. *Let* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *be a rank-s matrix, and let* $\mathbf{A} = \mathbf{U}^{(1)}\mathbf{\Sigma}^{(1)}\mathbf{V}^{(1)*}$ *be the compact SVD of* $\mathbf{A}$*, i.e.,* $\mathbf{U}^{(1)} \in \mathbb{R}^{m\times s}, \mathbf{\Sigma}^{(1)} \in \mathbb{R}^{s\times s}, \mathbf{V}^{(1)} \in \mathbb{R}^{n\times s}$*; then the subdifferential of the nuclear norm at* $\mathbf{A}$ *is*

$$\partial\|\mathbf{A}\|_* = \left\{\mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{W}}\tilde{\mathbf{V}}^{(2)*}\right\}, \tag{C.1}$$

where $\tilde{\mathbf{U}}^{(2)} \in \mathbb{R}^{m\times(n-s)}$, $\tilde{\mathbf{V}}^{(2)} \in \mathbb{R}^{n\times(n-s)}$, $\tilde{\mathbf{W}} \in \mathbb{R}^{(n-s)\times(n-s)}$ *satisfy that the columns of* $\tilde{\mathbf{U}}^{(2)}$ *and* $\tilde{\mathbf{V}}^{(2)}$ *are orthonormal,* $\mathrm{span}(\mathbf{U}^{(1)})\perp\mathrm{span}(\tilde{\mathbf{U}}^{(2)})$*,* $\mathrm{span}(\mathbf{V}^{(1)})\perp\mathrm{span}(\tilde{\mathbf{V}}^{(2)})$*, and* $\|\tilde{\mathbf{W}}\|_2 \leq 1$.

Remark C.2. Note that $\tilde{\mathbf{U}}^{(2)} \in \mathbb{R}^{m\times(n-s)}$ and $\tilde{\mathbf{V}}^{(2)} \in \mathbb{R}^{n\times(n-s)}$ are used in Lemma C.1 instead of $\mathbf{U}^{(2)} \in \mathbb{R}^{m\times s}$ and $\mathbf{V}^{(2)} \in \mathbb{R}^{n\times s}$ in (5.1). The reason is that $\tilde{\mathbf{U}}^{(2)}$ and $\tilde{\mathbf{V}}^{(2)}$ correspond to the reduced SVD of the matrix $\mathbf{A}$, and they are thus more suitable for numerical implementation.

Proof. Based on (5.1), we only need to show that the following two sets are identical:

$$D_1 = \left\{\mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \mathbf{U}^{(2)}\mathbf{W}\mathbf{V}^{(2)*} : \quad \forall \mathbf{W} \in \mathbb{R}^{(m-s)\times(n-s)} \text{ with } \|\mathbf{W}\|_2 \leq 1\right\}, \tag{C.2}$$

$$D_2 = \left\{\mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{W}}\tilde{\mathbf{V}}^{(2)*} : \quad \tilde{\mathbf{U}}^{(2)}, \tilde{\mathbf{V}}^{(2)}, \tilde{\mathbf{W}} \text{ specified in the lemma}\right\}. \tag{C.3}$$

On one hand, let $\boldsymbol{d} = \mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \mathbf{U}^{(2)}\mathbf{W}\mathbf{V}^{(2)*} \in D_1$, and let $\mathbf{U}^{(2)}\mathbf{W} = \bar{\mathbf{U}}\bar{\mathbf{\Sigma}}\bar{\mathbf{V}}^*$ be the reduced SVD of $\mathbf{U}^{(2)}\mathbf{W} \in \mathbb{R}^{m\times(n-s)}$, i.e., $\bar{\mathbf{U}} \in \mathbb{R}^{m\times(n-s)}, \bar{\mathbf{\Sigma}} \in \mathbb{R}^{(n-s)\times(n-s)}, \bar{\mathbf{V}} \in \mathbb{R}^{(n-s)\times(n-s)}$. Then we can set $\tilde{\mathbf{U}}^{(2)} = \bar{\mathbf{U}}$, $\tilde{\mathbf{W}} = \bar{\mathbf{\Sigma}}\bar{\mathbf{V}}^*$, and $\tilde{\mathbf{V}}^{(2)} = \mathbf{V}^{(2)}$. It is easy to check that $\tilde{\mathbf{U}}^{(2)}, \tilde{\mathbf{V}}^{(2)}, \tilde{\mathbf{W}}$ satisfy the conditions in the lemma, and

$$\boldsymbol{d} = \mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{W}}\tilde{\mathbf{V}}^{(2)*} \in D_2. \tag{C.4}$$

On the other hand, let $\boldsymbol{d} = \mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{W}}\tilde{\mathbf{V}}^{(2)*} \in D_2$, where $\tilde{\mathbf{U}}^{(2)}, \tilde{\mathbf{V}}^{(2)}, \tilde{\mathbf{W}}$ satisfy the conditions in the lemma. Let $\tilde{\mathbf{U}}^{(2)} = \mathbf{U}^{(2)}\mathbf{P}$ and $\tilde{\mathbf{V}}^{(2)} = \mathbf{V}^{(2)}\mathbf{Q}$, where $\mathbf{P} \in \mathbb{R}^{(m-s)\times(n-s)}$ and $\mathbf{Q} \in \mathbb{R}^{(n-s)\times(n-s)}$ have orthonormal columns. After setting $\mathbf{W} = \mathbf{P}\tilde{\mathbf{W}}\mathbf{Q}^*$, we have

$$\tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{W}}\tilde{\mathbf{V}}^{(2)*} = \mathbf{U}^{(2)}\mathbf{P}\tilde{\mathbf{W}}\mathbf{Q}^*\mathbf{V}^{(2)*} = \mathbf{U}^{(2)}\mathbf{W}\mathbf{V}^{(2)*}, \tag{C.5}$$

where $\|\mathbf{W}\|_2 \leq 1$. Therefore,

$$\boldsymbol{d} = \mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{W}}\tilde{\mathbf{V}}^{(2)*} = \mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \mathbf{U}^{(2)}\mathbf{W}\mathbf{V}^{(2)*} \in D_1. \tag{C.6}$$ ■

Now we go on to prove Theorem 5.1.

Proof. Let $\mathrm{rank}(\mathbf{A}) = s$, and split the computed singular vectors into two parts, $\mathbf{U}^\varepsilon = [\mathbf{U}^{(1)\varepsilon}, \mathbf{U}^{(2)\varepsilon}]$, $\mathbf{V}^\varepsilon = [\mathbf{V}^{(1)\varepsilon}, \mathbf{V}^{(2)\varepsilon}]$, where $\mathbf{U}^{(1)\varepsilon} \in \mathbb{R}^{m\times s}$, $\mathbf{U}^{(2)\varepsilon} \in \mathbb{R}^{m\times(n-s)}$,$\mathbf{V}^{(1)\varepsilon} \in \mathbb{R}^{n\times s}$, and $\mathbf{V}^{(2)\varepsilon} \in \mathbb{R}^{n\times(n-s)}$. By the backward stability of SVD [6, 8], we have $\|\mathbf{U}^{(1)} - \mathbf{U}^{(1)\varepsilon}\|_2 = O(\varepsilon/\eta)$, $\|\mathbf{V}^{(1)} - \mathbf{V}^{(1)\varepsilon}\|_2 = O(\varepsilon/\eta)$, and there exist $\tilde{\mathbf{U}}^{(2)}, \tilde{\mathbf{V}}^{(2)}$ satisfying the condition in the lemma and $\|\tilde{\mathbf{U}}^{(2)} - \mathbf{U}^{(2)\varepsilon}\|_2 = O(\varepsilon/\eta)$, $\|\tilde{\mathbf{V}}^{(2)} - \mathbf{V}^{(2)\varepsilon}\|_2 = O(\varepsilon/\eta)$.

Because of the lemma, we have $(\mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{V}}^{(2)*}) \in \partial\|\mathbf{A}\|_*$, and

$$\begin{aligned}
\mathrm{d}(\mathbf{U}^{\varepsilon}\mathbf{V}^{\varepsilon *}, \partial\|\mathbf{A}\|_*) \leq\; & \left\|\mathbf{U}^{\varepsilon}\mathbf{V}^{\varepsilon *} - \left(\mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{V}}^{(2)*}\right)\right\|_2 \\
=\; & \left\| \left(\mathbf{U}^{(1)\varepsilon}\mathbf{V}^{(1)\varepsilon *} + \mathbf{U}^{(2)\varepsilon}\mathbf{V}^{(2)\varepsilon *}\right) - \left(\mathbf{U}^{(1)}\mathbf{V}^{(1)*} + \tilde{\mathbf{U}}^{(2)}\tilde{\mathbf{V}}^{(2)*}\right)\right\|_2 \\
\leq\; & \left\| \left(\mathbf{U}^{(1)\varepsilon} - \mathbf{U}^{(1)}\right)\mathbf{V}^{(1)\varepsilon *}\|_2 + \|\mathbf{U}^{(1)}\left(\mathbf{V}^{(1)\varepsilon *} - \mathbf{V}^{(1)*}\right)\right\|_2 \\
& + \left\| \left(\mathbf{U}^{(2)\varepsilon} - \tilde{\mathbf{U}}^{(2)}\right)\mathbf{V}^{(2)\varepsilon *}\|_2 + \|\tilde{\mathbf{U}}^{(2)}\left(\mathbf{V}^{(2)\varepsilon *} - \tilde{\mathbf{V}}^{(2)*}\right)\right\|_2 \\
=\; & O(\varepsilon/\eta). \qquad \text{(C.7)} \qquad \blacksquare
\end{aligned}$$

Acknowledgment. The authors would like to thank José Lezama for providing the code of OLE [15].

REFERENCES

[1] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, *A closer look at memorization in deep networks*, in Proceedings of the 34th International Conference on Machine Learning, D. Precup and Y. W. Teh, eds., Proc. Mach. Learn. Res. 70, International Convention Centre, Sydney, Australia, 2017, pp. 233–242.

[2] P. L. Bartlett and S. Mendelson, *Rademacher and Gaussian complexities: Risk bounds and structural results*, J. Mach. Learn. Res., 3 (2002), pp. 463–482.

[3] O. Bousquet and A. Elisseeff, *Stability and generalization*, J. Mach. Learn. Res., 2 (2002), pp. 499–526.

[4] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, *Person re-identification by multi-channel parts-based CNN with improved triplet loss function*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1335–1344.

[5] S. Chopra, R. Hadsell, and Y. LeCun, *Learning a similarity metric discriminatively, with application to face verification*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), Vol. 1, 2005, pp. 539–546.

[6] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997, https://doi.org/10.1137/1.9781611971446.

[7] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 2012.

[9] R. Hadsell, S. Chopra, and Y. LeCun, *Dimensionality reduction by learning an invariant mapping*, in Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, 2006, pp. 1735–1742.

[10] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[11] J. Hu, J. Lu, and Y.-P. Tan, *Discriminative deep metric learning for face verification in the wild*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1875–1882.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[13] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*, Technical Report TR-2009, University of Toronto, Toronto, Canada, 2009.

[14] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, *Handwritten digit recognition with a back-propagation network*, in Advances in Neural Information Processing Systems, Vol. 2, Morgan Kaufmann, San Francisco, 1990, pp. 396–404.

[15] J. Lezama, Q. Qiu, P. Musé, and G. Sapiro, *OLE: Orthogonal low-rank embedding, a plug and play geometric loss for deep learning*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[16] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, *Reading digits in natural images with unsupervised feature learning*, in NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

[17] T. Papadopoulo and M. I. A. Lourakis, *Estimating the Jacobian of the singular value decomposition: Theory and applications*, in Computer Vision – ECCV 2000, Springer, Berlin, Heidelberg, pp. 554–570.

[18] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi, *General conditions for predictivity in learning theory*, Nature, 428 (2004), pp. 419–422.

[19] Q. Qiu and G. Sapiro, *Learning transformations for clustering and classification*, J. Mach. Learn. Res., 16 (2015), pp. 187–225.

[20] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501, https://doi.org/10.1137/070697835.

[21] F. Schroff, D. Kalenichenko, and J. Philbin, *Facenet: A unified embedding for face recognition and clustering*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.

[22] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, preprint, https://arxiv.org/abs/1409.1556, 2014.

[23] Y. Sun, Y. Chen, X. Wang, and X. Tang, *Deep learning face representation by joint identification-verification*, in Proceedings of the 27th International Conference on Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., MIT Press, Cambridge, MA, 2014, pp. 1988–1996.

[24] V. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[25] G. A. Watson, *Characterization of the subdifferential of some matrix norms*, Linear Algebra Appl., 170 (1992), pp. 33–45.

[26] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, *A discriminative feature learning approach for deep face recognition*, in Computer Vision – ECCV 2016, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., Springer, Cham, 2016, pp. 499–515.

[27] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, *Deep learning via semi-supervised embedding*, in Neural Networks: Tricks of the Trade, Springer, Berlin, Heidelberg, 2012, pp. 639–655.

[28] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, *Understanding deep learning requires rethinking generalization*, in Proceedings of the International Conference on Learning Representations, 2017.

[29] W. Zhu, Q. Qiu, J. Huang, R. Calderbank, G. Sapiro, and I. Daubechies, *LDMNet: Low dimensional manifold regularized neural networks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.