Received: 8 December 2019 | Revised: 21 January 2020 | Accepted: 21 January 2020
DOI: 10.1002/sta4.273

SPECIAL ISSUE PAPER

Statistical insights into deep neural network learning in subspace classification

Hao Wu[1] | Yingying Fan[2] | Jinchi Lv[2]

[1]Department of Mathematics, Dornsife College of Letters, Arts and Sciences, University of Southern California, Los Angeles, 90089-0001, CA, USA
[2]Data Sciences and Operations Department, Marshall School of Business, University of Southern California, Los Angeles, 90089-0001, CA, USA

Correspondence
Hao Wu, Department of Mathematics, Dornsife College of Letters, Arts and Sciences, University of Southern California, Los Angeles, CA 90089-0001, USA.
Email: hwu409@usc.edu

Deep learning has benefited almost every aspect of modern big data applications. Yet its statistical properties still remain largely unexplored. It is commonly believed nowadays that deep neural networks (DNNs) benefit from representational learning. To gain some statistical insights into this, we design a simple simulation setting where we generate data from some latent subspace structure with each subspace regarded as a cluster. We empirically demonstrate that the performance of DNN is very similar to that of the two-step procedure of clustering followed by classification (unsupervised plus supervised). This motivates us to ask: Does DNN indeed mimic the two-step procedure statistically? That is, do bottom layers in DNN try to cluster first and then top layers classify within each cluster? To answer this question, we conduct a series of simulation studies, and to our surprise, none of the hidden layers in DNN conduct successful clustering. In some sense, our results provide an important complement to the common belief of representational learning, suggesting that at least in some model settings, although the performance of DNN is comparable with that of the ideal two-step procedure knowing the true latent cluster information a priori, it does not really do clustering in any of its layers. We also provide some statistical insights and heuristic arguments to support our empirical discoveries and further demonstrate the revealed phenomenon on the real data application of traffic sign recognition.

KEYWORDS
clustering, DNN, statistical insights, subspace classification

1 | INTRODUCTION

Deep learning is a popular machine learning method that has gained a lot of interest in recent years. It has dramatically improved the state-of-the-art in image processing, speech recognition, text analysis, and many other domains such as health care and finance; see, e.g., He, Zhang, Ren, and Sun (2016), Amodei et al. (2016), Majumder, Poria, Gelbukh, and Cambria (2017), Litjens et al. (2017), and Heaton, Polson, and Witte (2017). In particular, there has been a significant amount of research on classifications of different objects by deep learning; see, e.g., Chen, Lin, Zhao, Wang, and Gu (2014), Socher, Huval, Bath, Manning, and Ng (2012), Cireşan, Meier, and Schmidhuber (2012), Glorot, Bordes, and Bengio (2011), and Lee, Pham, Largman, and Ng (2009). To name a few, Kussul, Lavreniuk, Skakun, and Shelestov (2017) proposed a multilevel deep learning approach for land cover and crop type classification with multitemporal multisource satellite imagery. Dolz et al. (2017) created a deep learning-based classification scheme by stacking denoising auto-encoders to segment organs at risk in the optic region in brain cancer. Esteva et al. (2017) demonstrated the effectiveness of deep learning in dermatology, a technique that we apply to both general skin conditions and specific cancers.

Despite their popularity, most deep learning methods are regarded as black-box procedures in the sense that their statistical properties are largely unknown. In recent years, researchers attempt to unveil the statistical properties behind deep learning. Existing theoretical developments focus mainly on algorithms, probabilistic understanding, and approximation theory in deep learning. For example, Mhaskar and Poggio (2016) demonstrated how different smoothness classes lead to satisfactory results for approximation by rectified linear unit networks and Gaussian networks on the entire Euclidean space. Patel, Nguyen, and Baraniuk (2015) developed a generative probabilistic model for deep learning that explicitly captures latent nuisance variation. Poggio et al. (2017) used the classical theory of ordinary differential equations and replaced a potentially fundamental puzzle about generalization in deep learning with elementary properties of gradient optimization techniques. Jacot, Gabriel, and Hongler (2018) introduced a new perspective to analyse the converging dynamics of neural networks via the neural tangent kernel. Saxe et al. (2018) suggested that compression dynamics in the information plane are not a general feature of deep networks but are critically influenced by the non-linearities employed by the network. Lu, Fan, Lv, and Noble (2018) introduced a general framework of DeepPINK for reproducible feature selection in deep neural networks (DNNs). Nevertheless, the statistical insights into deep learning methods are still mostly unexcavated.

In this paper, we attempt to provide some understanding on the statistical performance of basic deep learning methods through numerically and theoretically studying a simple statistical model for two-class classification in a high-dimensional space. Our model assumes that observations are independently and identically distributed in some high-dimensional space, which is the union of several latent lower dimensional subspaces. On each subspace, we further assume that the observations are independent and follow a mixture of two Gaussian distributions with known labels. Note that the subspace structure is completely latent and unavailable to us. This model is remotely motivated from some real-life problems such as classifying cats and dogs using image data, where different pictures can correspond to different activities (e.g., eating, playing, and sleeping). Roughly speaking, each lower dimensional subspace in our model can be regarded as one type of activities of animals, and the union of all these subspaces mimics the reality that pictures for each animal species can consist of different activities. Another example is skin cancer detection, which aims at successfully classifying cancer cells and noncancer ones. Skin cancer is among the most common of all human cancers, with 1 million people in the United States diagnosed each year with some type of the disease. There are three major types of skin cancers: basal cell carcinoma, squamous cell carcinoma, and melanoma. They are likely to grow in different areas, which could be thought of as clusters. A further example is the recognition of traffic signs that lie in different groups on the basis of the level of brightness and the angle of view, which plays an important role in self-driving cars. These examples motivate us to consider the high-dimensional classification problem under the latent structural assumption of lower dimensional subspaces. We acknowledge that our statistical model is definitely an oversimplified version of the previously discussed real-life examples. However, because our intension is to provide some statistical insights into deep learning methods, studying this oversimplified model should not be a big issue for our specific purpose.

It is popularly believed nowadays that DNNs benefit from representational learning, meaning that the bottom layers of DNNs try to extract representations of different clusters (subspaces in our statistical model), and then the top layers try to use these representations to help with classifications. However, through applying deep learning methods to a simulated data set from our statistical model discussed above, we discover some surprising facts that provide an important complement to the common belief of representational learning. To better understand our message, let us temporarily walk away from the deep learning framework and think about the ideal procedure for classification under our model setting. Ideally, if the clusters (or subspaces) are known to us, then the best we can do is to first separate the data according to subspaces and then conduct classification within each subspace. In the following, we will name this two-step procedure relying on the oracle subspace information the *ideal procedure* and use it as a benchmark to evaluate the performance of DNNs. It is worth mentioning that clustering methods are popular in the data mining literature. For example, Soltanolkotabi, Elhamifar, and Candes (2014) formulated a great solution to recover the union of subspaces from a high-dimensional setting with elegant theoretical guarantees. Many adapted algorithms have also been developed. Aljalbout, Golkov, Siddiqui, Strobel, and Cremers (2018) designed customized neural networks coupled with K-means clustering and added the clustering hardening loss to guide the process of updating the network and encourage clustering of feature spaces, which is also supporting evidence for our paper that the regular DNN does not efficiently conduct clustering. In this paper, we intend to study this phenomenon of regular DNNs through a simple statistical model, provide some statistical insights, and demonstrate the revealed phenomenon on the real data application. For face recognition, Liu et al. (2017), Deng, Guo, and Zafeiriou (2018), Wang, Cheng, Liu, and Liu (2018), Wang et al. (2018), and Wen, Zhang, Li, and Qiao (2019) have dedicated a lot of efforts to the design of new loss functions to minimize the distances between the deep features of the same class, thereby improving face identification and verification accuracy empirically. Because it is impractical to precollect all possible class identities for training, those interesting works focus on compressing the learned deep features of the same class to their centres and enhancing their discriminative power for verifying new unseen classes without label prediction. In contrast, in our paper, the data have a latent subspace structure on top of the classes, where the subspaces (clusters) can be viewed as the superclasses at a higher level. Under such a model structure, we explicitly study how the DNN learns without clustering in advance and why it achieves comparable classification performance to the ideal two-step procedure, that is, unsupervised clustering to partition data into subspaces (superclasses) followed by supervised classification for further recognition within each subspace.

Through simulation studies, we discover that the classification error rate of DNNs is very close to that of the ideal procedure, motivating us to ask whether the bottom layers of DNNs indeed conduct clustering first and then top layers classify within each learned cluster, which can be regarded as some type of representational learning. However, our simulation studies suggest otherwise. More specifically, we discover that none of the layers in the DNN successfully recover the cluster/subspace structure. This message suggests that at least in some high-dimensional classifications, the way a DNN learns is different from the commonly believed representational learning.

To strengthen our empirical evidence, we further provide some theoretical results and insights under our statistical model. Our theoretical findings are consistent with our empirical ones—the ideal procedure can have very low classification error rate and greatly outperform the naive procedure of blind classification, which completely ignores the subspace structure in the data. We also provide some heuristic statistical arguments on the performance of DNNs. Furthermore, we demonstrate the revealed phenomenon on the real data application of traffic sign recognition, which plays a crucial role in self-driving vehicles.

2 | EXPERIMENTS

2.1 | Model setting and some initial results

Consider a sample of independent observations in $\mathbb{R}^D$ from a union of three lower dimensional subspaces S_1, S_2, S_3, where each of the three subspaces is of dimensionality p with $p < D$. For each subspace S_k (for $k \in \{1, 2, 3\}$), denote by $A_k \in \mathbb{R}^{D\times p}$ the basis matrix whose columns are orthonormal vectors spanning the subspace. For an observation from subspace S_k, suppose it can be represented as $\mathrm{x} = A_k\widetilde{\mathrm{x}}$, where $\widetilde{\mathrm{x}} \in \mathbb{R}^p$ follows the mixture Gaussian distribution $\widetilde{\mathrm{x}} \sim (1 - Y)\mathcal{N}_p(\boldsymbol{\mu}_0^{(k)}, \Sigma) + Y\mathcal{N}_p(\boldsymbol{\mu}_1^{(k)}, \Sigma)$ with $\boldsymbol{\mu}_j^{(k)} \in \mathbb{R}^p$ (for $j \in \{1, 2\}$) the mean vectors and $\Sigma \in \mathbb{R}^{p\times p}$ the covariance matrix. Here, Y is a Bernoulli random variable with probability of success 1/2 representing the class label. Note that the latent subspace structure, that is, the basis matrices (A_1, A_2, A_3), is completely unavailable to us.

Suppose we have n labelled observations (x_i, Y_i), $i = 1, \dots, n$, independently sampled from the above latent subspace model. Denote by $\mathrm{X} = [\mathrm{x}_1^T, \dots, x_n^T]^T \in \mathbb{R}^{n\times D}$ the matrix whose columns are feature vectors, and $\mathrm{Y} = (Y_1, \dots, Y_n)^T$ the vector collecting the labels. Let $(\mathrm{X}^{(k)}, \mathrm{Y}^{(k)})$ be the observations corresponding to the kth subspace and $\widetilde{\mathrm{X}}^{(k)}$ the corresponding coefficient matrix, that is, $\mathrm{X}^{(k)} = \widetilde{\mathrm{X}}^{(k)}A_k^T$. Note that for each $k \in \{1, 2, 3\}$, $\mathrm{X}^{(k)}, \widetilde{\mathrm{X}}^{(k)}, \mathrm{Y}^{(k)}$, and A_k are unobservable to us. Our goal is to train a classifier $\widehat{\mathrm{C}}(\mathrm{x})$ such that for a new unlabelled observation x, the trained classifier $\widehat{\mathrm{C}}(\mathrm{x})$ can predict the class label of x with high success rate.

We simulate $n_k = 800$ observations from subspace S_k, and we set $D = 100$ and $p = 40$. The augmented basis matrix $[A_1, A_2, A_3] \in \mathbb{R}^{D\times(3p)}$ is nondegenerate with rank D. So we end up with $n = 2,400$ observations (x_i, Y_i) in total, where each feature vector x_i has dimensionality $D = 100$. Let $A = (0.317, 0.318, 0.684)^T$ and $\mathrm{d} = (1.098, 1.092, -1.104)^T$. We set $\boldsymbol{\mu}_0^{(k)} = A_k^T A(k)1$ and $\boldsymbol{\mu}_1^{(k)} = A_k^T(A(k) + \mathrm{d}(k))1$, where 1 is a vector of 1's, for $k = 1, 2, 3$, where $A(k)$ and $\mathrm{d}(k)$ stand for the kth entry of the vectors A and d, respectively.

As discussed in Section 1, if the oracle information on each observation's subspace identity is known, the best one can do is the ideal procedure. We apply the ideal procedure to the simulated data and use the result as a benchmark to evaluate the performance of other methods. We also apply neural networks with two hidden layers to the simulated data set (x_i, Y_i). More details about our neural network can be found in a later section. The third comparison method is the Fisher linear discriminant analysis (LDA) applied to (x_i, Y_i). Note that for the last two methods, we do not use the oracle information on observations' subspace identity.

Table 1 shows the distance correlations (DCs; Reshef et al., 2011; Szkely, Rizzo, & Bakirov, 2007) between various pairs of variables, which is a specific case of maximum mean discrepancy (MMD; Arbel, Sutherland, Bińkowski, & Gretton, 2018; Dziugaite, Roy, & Ghahramani, 2015; Gao, Lv, & Shao, 2019; Jones and Forrest, 1995; Kong, Li, Fan, & Lv, 2017; Li, Zhong, & Zhu, 2012) with the distance kernel. DC provides an effective measure of the non-linear dependency between two random variables of potentially different dimensions and data types defined as

$$dCor(u, v) = \frac{dCov(u, v)}{\sqrt{dVar(u)dVar(v)}} \tag{1}$$

with

$$dCov^2(u, v) = \frac{1}{c_p c_q} \int_{\mathbb{R}^{p+q}} \frac{|\phi_{u,v}(x, y) - \phi_u(x)\phi_v(y)|^2}{||x||^{p+1}||y||^{q+1}}\, dxdy. \tag{2}$$

Here, u and v are two random vectors of arbitrary dimensions p and q, respectively; $\phi_u(x)$, $\phi_v(y)$, and $\phi_{u,v}(x, y)$ are the characteristic functions of u, v, and (u, v), respectively; and the constant $c_k = \pi^{(1+k)/2}/\Gamma((1 + k)/2)$ is half of the surface area of the unit sphere in $\mathbb{R}^{k+1}$. In particular, DC is zero if and only if u and v are independent. See Székely and Rizzo (2013) for details of the bias-corrected version of DC. Several messages can be drawn. First, the classification problem is challenging as reflected from the very low DC between feature matrix X and label vector Y. This will be further confirmed later by Figure 3. Second, knowing the subspace/cluster identity can help with classification, as evidenced from the much higher DCs between the class labels $\mathrm{Y}^{(k)}$ and submatrices $\mathrm{X}^{(k)}$ or $\widetilde{\mathrm{X}}^{(k)}$, and the high DC between Y and Y_F with Y_F denoting the predicted training labels by the ideal procedure. In fact, the classification error rate by the ideal procedure is 4.34%. Third, the predicted training labels by DNN (the structure is shown in Figure 1) $\widehat{\mathrm{Y}}$ mimic Y_F very closely, with DC as high as 0.8829. Fourth, none of the hidden layers cluster the data well, as shown by the low DCs in the last three rows in Table 1. These results suggest a surprising fact that DNN does not learn the cluster representations in data well but still manages to classify well. We next provide more empirical evidence in the following sections.

2.2 | Classification methods without clustering versus with clustering

In the last section, we have seen that the ideal procedure based on oracle subspace information performs well in classification. We further demonstrate here that even with empirically learned cluster information, the performance of various popular classifiers can be very close to that of the ideal procedure. Specifically, we directly apply popularly used classification methods including LDA, lasso logistic regression, KNN, decision tree, random forest, and support vector machine (SVM) with linear and non-linear kernels, to the simulated data. To compare, we also implement a two-step procedure where in the first step, we use subspace clustering algorithm (Park, Caramanis, & Sanghavi, 2014) to cluster the data first, and

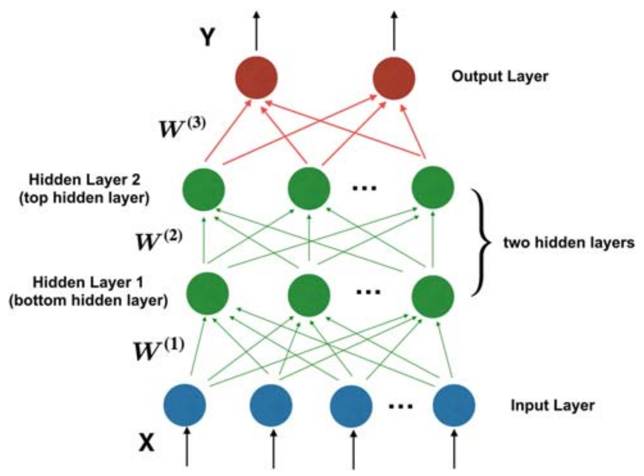

FIGURE 1 Deep neural network (DNN) structure

TABLE 1 Bias-corrected distance correlations (DCs) between different pairs of variables

Variable 1	Variable 2	DC
X	Y	0.0779
$X^{(1)}$	$Y^{(1)}$	0.3817
$X^{(2)}$	$Y^{(2)}$	0.2826
$X^{(3)}$	$Y^{(3)}$	0.4004
$\widetilde{X}^{(1)}$	$Y^{(1)}$	0.4740
$\widetilde{X}^{(2)}$	$Y^{(2)}$	0.3345
$\widetilde{X}^{(3)}$	$Y^{(3)}$	0.4038
Y	Y_F	**0.8341**
$\hat{Y}$	Y_F	**0.8829**
X_c	Z_{subspace}	0.8925
$X_{\text{bottomLayer}}$	Z_{subspace}	0.3428
X_{topLayer}	Z_{subspace}	0.2420
$X_{\text{twoLayers}}$	Z_{subspace}	0.2965

Note. X: $n \times D$ feature matrix; Y: vector of true class labels; Y_F: predicted class labels by the ideal procedure; $\hat{Y}$: predicted class labels by a DNN with two hidden layers; $\widetilde{X}^{(k)}$: $n_k \times p$ coefficient matrix corresponding to the kth subspace; $X^{(k)} = \widetilde{X}^{(k)} A_k^T$: $n_k \times D$ feature matrix corresponding to subspace k; $Y^{(k)}$: labels for observations on subspace k; X_c : transformed X via clustering; $X_{\text{bottomLayer}}$: n-vector representing the hidden layer after the input layer; X_{topLayer}: n-vector representing the hidden layer before the output layer; $X_{\text{twoLayers}}$: the augmented matrix $[X_{\text{bottomLayer}}, X_{\text{topLayer}}]$; Z_{subspace}: the latent subspace label for each observation. Bold data emphasize the trend.

then in the second step, we apply each of the aforementioned classification methods within each identified cluster. The number of repetitions is 100, with the mean error for clustering being 0.0008. The very low clustering error also suggests that the latent subspace structure is easy to learn.

It is shown that all methods have improved performance after we incorporate the clustering step, with clustering followed by lasso logistic regression with the best overall classification error rate of 5.46%. In addition, with clustering, almost all methods (except for KNN because of possible curse of dimensionality) perform similarly and are comparable with the ideal procedure of classification error rate 4.34%. For illustration purposes, the specific plot for lasso logistic regression is shown in Figure 2.

Similar to Table 1, we calculate the DC between the predicted class labels by various methods in Figures 3 and 4. The results are presented in Table 2. The last column corresponds to the DCs for test data. These results confirm again that DNN has performance very close to that of these two-step procedures.

2.3 | DNNs via TensorFlow

We implemented deep learning by TensorFlow on the hardware 8-core 2.4-GHz Intel E5-2665 processors and NVIDIA K20 Kepler graphics processing unit (GPU) accelerators. To speed up the training process of DNN, we use parallel computing with 24 threads/central processing units (CPUs) (2 sockets, 6 cores per socket, and 2 threads per core) and 2 GPUs.

Figure 5 shows the average error rate over 100 test data sets. It can be seen from Figure 5 that the classification error rate follows a U-shaped curve, regardless of whether dropout is used or not on both CPU and GPU. For the same number of layers, the error rate decreases as the

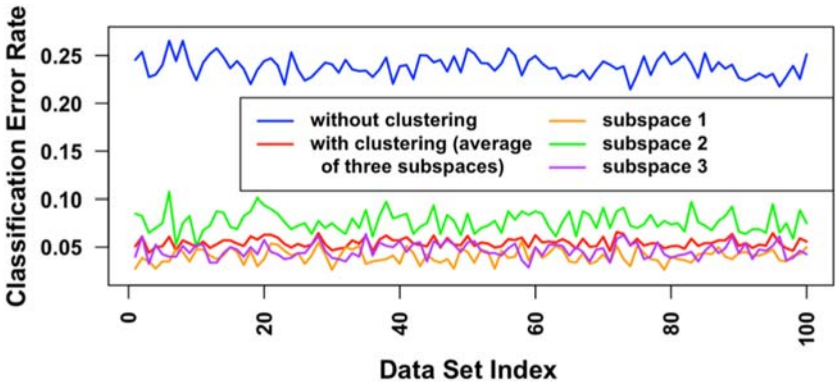

FIGURE 2 Classification error rate for lasso logistic regression

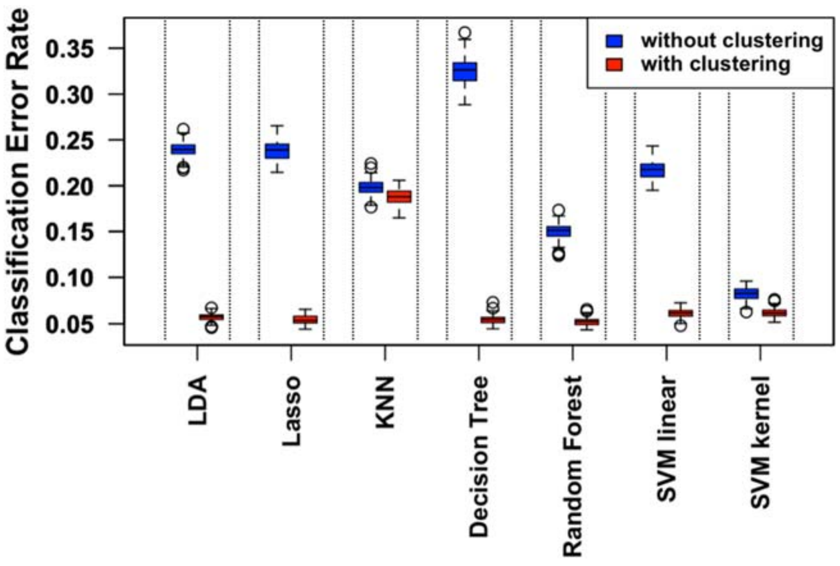

FIGURE 3 Classification error rates by different methods

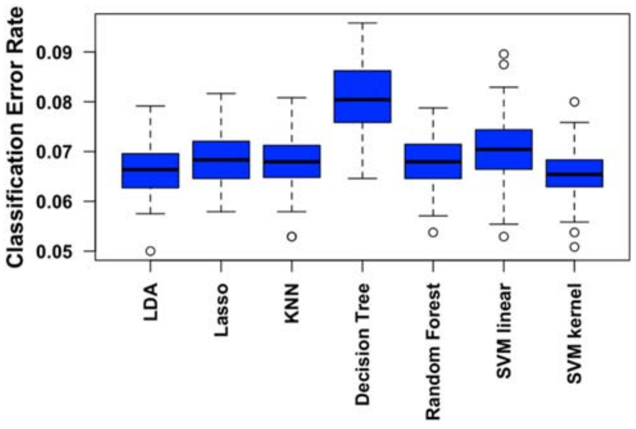

FIGURE 4 Classification error rates by different methods

number of hidden units grows. Additionally, for the same number of hidden units in each hidden layer, the error rate declines first and then rises afterward. This explains why we focus on DNN with two hidden layers and 100 hidden units in each layer (dropout = 0.5) (see Tables A1–A4 for detailed numerical results for various numbers of layers with dropout = 0.5). Later, we add dropout = 0.2 for the input layer and apply centerization and normalization to achieve lower error rate 0.0640 (CPU) and 0.0644(GPU).

To examine whether any layers of DNN carry the latent cluster information, we apply various popularly used clustering methods to the bottom hidden layer $X_{\text{bottomLayer}}$, top hidden layer X_{topLayer}, and the combination of the two hidden layers $X_{\text{bottomLayer}} + X_{\text{topLayer}}$. The methods we experimented with include the greedy subspace clustering (Park et al., 2014), the robust subspace clustering via thresholding (Heckel & Bölcskei, 2015), the K-means clustering Lloyd (1982), the hierarchical clustering (Rokach & Maimon, 2005), and the singular value decomposition coupled with the aforementioned two methods, where the clustering error rates are shown in Table 3. The clustering error rate is the mismatch ratio of cluster labels relative to the true labels up to permutation of classes. To get some visualization, we provide the T-distributed Stochastic Neighbor Embedding (t-SNE) plots (van der Maaten & Hinton, 2008) in Figure 6. We can see from the plots that DNN did not preserve the clustering

TABLE 2 Bias-corrected distance correlations (DCs) for the top hidden layer of deep neural network

Variable 1	Variable 2	Training DC	Test DC
$\hat{Y}$	Y_F	0.8829	0.8354
Y_{lasso}	Y_F	0.8717	0.8175
Y_{knn}	Y_F	0.8707	0.8197
$Y_{\text{linearSVM}}$	Y_F	0.8650	0.8081
$Y_{\text{kernelSVM}}$	Y_F	0.8746	0.8310
$Y_{\text{randomForest}}$	Y_F	0.8345	0.8217

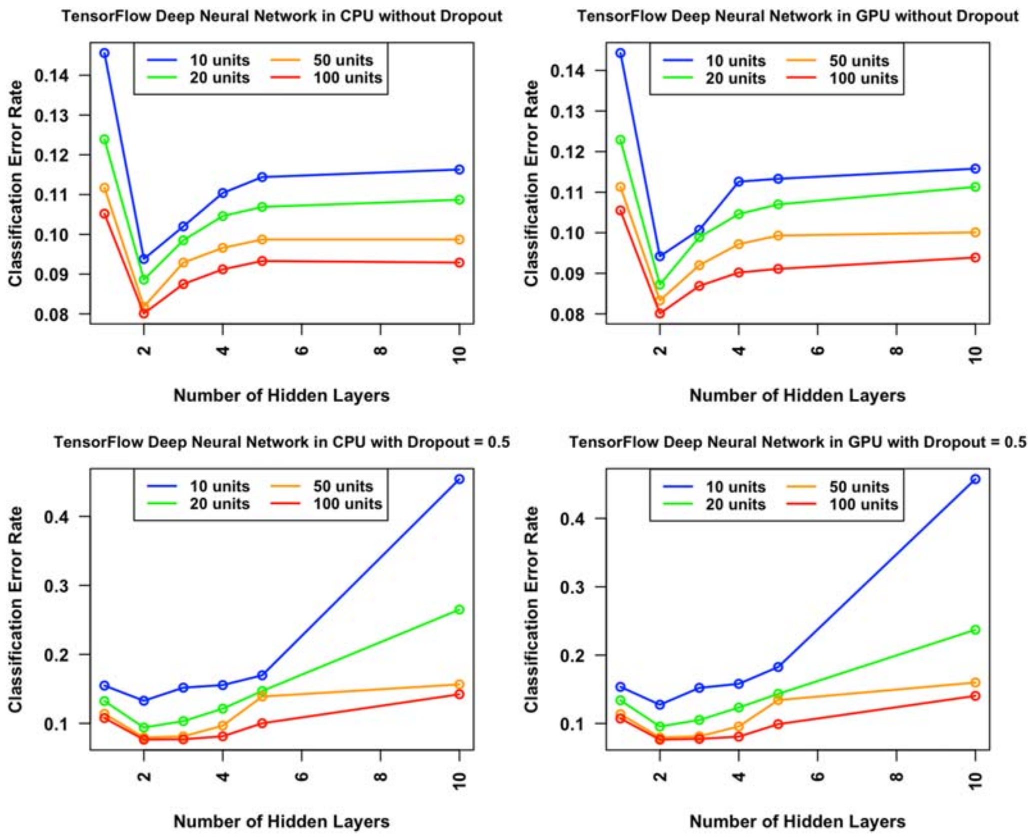

FIGURE 5 Classification error rate of deep neural network (DNN)

TABLE 3 Clustering error rates on hidden layers

Method	$X_{\text{bottomLayer}}$	X_{topLayer}	$X_{\text{bottomLayer}} + X_{\text{topLayer}}$
GreedySC	0.431913	0.470013	0.470083
RSCT	0.403967	0.430333	0.421375
K-means	0.448750	0.474450	0.460292
SVD + *K*-means	0.444742	0.470392	0.464654
Hierarchical	0.666054	0.666050	0.666079
SVD + Hierarchical	0.666075	0.666025	0.666071

Abbreviations: GreedySC, greedy subspace clustering; RSCT, robust subspace clustering via thresholding; SVD, singular value decomposition.

pattern and learned features in a different mechanism. The patterns shown in the three plots also support the trend in the DC values that we calculated in Table 1. These high clustering error rates here with the low clustering error rate in Section 2.2, as well as the t-SNE visualization in Figure 6, suggest that DNN does not conduct efficient clustering in any of its layers.

We also apply the classification methods discussed in Section 2.2 to just the top hidden layer. As seen from Figure 4, the classification performance is highly similar to that in Figure 3 with the exception of the decision tree method. These results reinforce our previous statement that DNN has comparable classification results to two-step procedures without performing clustering first.

In addition, we have also conducted a larger scale simulation study in which both the sample size and dimensionality are enlarged by 10 times. Indeed, the implementation of such a large-scale study was very time-consuming. However, the patterns and phenomenon observed previously stay the same in the large scale. A short summary of the main results for this additional large-scale simulation study is presented in Section A.5 (the full detailed results are available upon request).

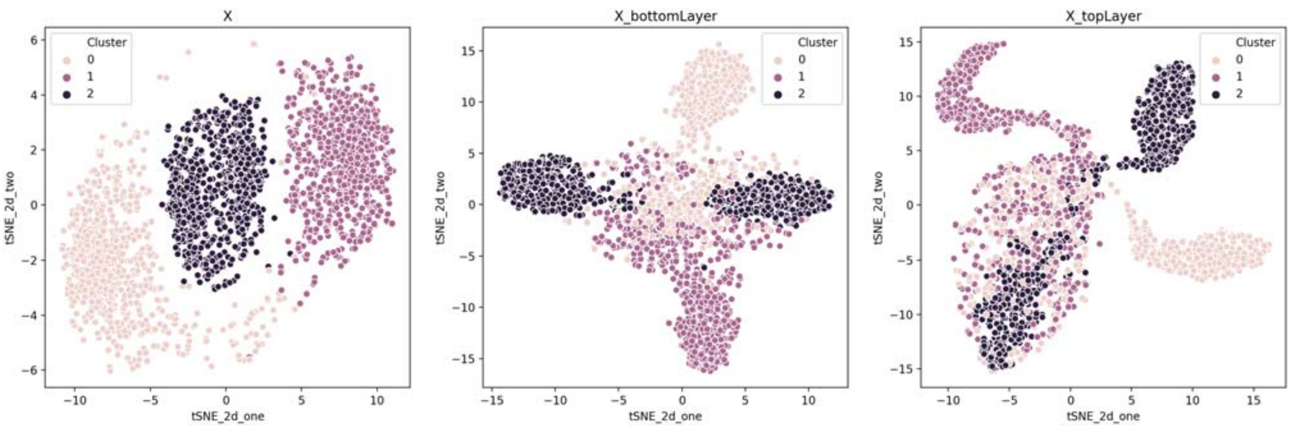

FIGURE 6 T-distributed Stochastic Neighbor Embedding (t-SNE) visualization

3 | SOME STATISTICAL INSIGHTS AND HEURISTIC ARGUMENTS

We attempt to provide some theoretical justifications on our empirical findings in this section. Our results have three components: classification within each cluster, classification without learning the cluster information, and some heuristic statistical arguments on how DNN learns.

3.1 | Gaussian classification in one subspace

We focus on classification within one subspace in this section. For simplicity, we drop all the superscripts corresponding to the subspace, and thus the two class distributions are $\mathcal{N}_p(\mu_0, \Sigma)$ and $\mathcal{N}_p(\mu_1, \Sigma)$, and the observations are (x_i, Y_i), $i = 1, \ldots, n$. We restrict ourselves to the lower dimensional subspace, so the dimensionality is p. The class prior probabilities are assumed to be equal.

In this two-class Gaussian classification, the Bayes rule takes the form

$$\delta_{\mathrm{B}}(\mathrm{x}) = \mathbb{1}\{\Delta^T \Sigma^{-1}(\mathrm{x} - \mu) > 0\},$$

where $\Delta = \mu_1 - \mu_0$ and $\mu = \frac{1}{2}(\mu_0 + \mu_1)$. With training data, we can plug in the sample estimates of $\boldsymbol{\mu}_i$ and Σ, denoted by $\hat{\boldsymbol{\mu}}_i$ and $\hat{\Sigma}$, and the corresponding LDA classifier is

$$\delta_F(\mathrm{x}) = \mathbb{1}\{\hat{\Delta}^T \hat{\Sigma}^{-1}(\mathrm{x} - \hat{\mu}) > 0\} \text{with } \hat{\Delta} = \hat{\mu}_1 - \hat{\mu}_0.$$

It is well known that the Bayes risk, or the classification error rate of the Bayes rule $\delta_{\mathrm{B}}(\mathrm{x})$, takes the form

$$R_{\mathrm{B}} = \bar{\Phi}(m/2),$$

where $m^2 = \Delta^T \Sigma^{-1} \Delta$ is the Mahalanobis distance between two classes. For the LDA $\delta_F(\mathrm{x})$, conditional on the training data, the classification error rate is

$$R_n(\delta_F) = \frac{1}{2} \sum_{k=0,1} \bar{\Phi}\left(\frac{(-1)^{k+1} \Delta^T \hat{\Sigma}^{-1}(\boldsymbol{\mu}_k - \hat{\boldsymbol{\mu}}_k) + \hat{\Delta}\hat{\Sigma}^{-1}\hat{\Delta}/2}{(\hat{\Delta}^T \hat{\Sigma}^{-1} \Sigma \hat{\Sigma}^{-1} \hat{\Delta})^{1/2}} \right),$$

where $\bar{\Phi} = 1 - \Phi$ with Φ the standard Gaussian cumulative distribution function.

The following result is a direct consequence from Li and Shao, 2015 (Theorem 1 (ii)). We include it for completeness.

Theorem 1. *Assume that $p/n \to 0$ as $n \to \infty$. Then for large enough n,*

$$R_n(\delta_F) = R_{\mathrm{B}} + o_p(1). \tag{3}$$

The above result suggests that when the latent subspace structure is known, classification can be highly accurate as long as the Mahalanobis distance m^2 is large enough. This is consistent with our results in Section 2.2. See, for example, Fan and Fan (2008), Fan, Jin, and Yao (2013), and Fan, Kong, Li, and Zheng (2015) for more developments on high-dimensional classification.

3.2 | Gaussian classification in two subspaces

We now consider the case where data are from the union of some lower dimensional subspaces. To simplify the proof, we consider two subspaces S_1 and S_2 with $p = D/2$, and we assume that the basis matrices take the forms $A_1^T = [I_p, 0] \in \mathbb{R}^{p \times D}$ and $A_2^T = [0, I_p] \in \mathbb{R}^{p \times D}$. In addition, on each subspace S_k, we assume that the common covariance matrix for the mixture Gaussian is I_p. We also assume that conditional on the class label Y, the observations have equal chance of coming from each of these two subspaces. These oversimplifications allow us to provide an explicit bound for the overall classification error rate.

TABLE 4 Bias-corrected distance correlations (DCs) between different pairs of variables

Clustering strategy	Variable 1	Variable 2	DC
(1) LBFV, LBSV, HBFV, HBSV	$X_{\text{bottomLayer}}$	Z^1	0.0558
	X_{topLayer}	Z^1	0.0508
	$X_{\text{twoLayers}}$	Z^1	0.0516
(2) LB, HB	$X_{\text{bottomLayer}}$	Z^2	0.0631
	X_{topLayer}	Z^2	0.0583
	$X_{\text{twoLayers}}$	Z^2	0.0591
(3) FV, SV	$X_{\text{bottomLayer}}$	Z^3	0.0311
	X_{topLayer}	Z^3	0.0276
	$X_{\text{twoLayers}}$	Z^3	0.0281

Note. $X_{\text{bottomLayer}}$: deep feature by the hidden layer after the input layer; X_{topLayer}: deep feature by the hidden layer before the output layer; $X_{\text{twoLayers}}$: the augmented matrix $[X_{\text{bottomLayer}}, X_{\text{topLayer}}]$; Z^i: the one-hot cluster identities of the strategy i). Abbreviations: B, brightness; F, front; H, high; L, low; S, side; V, view.

With the above model structure, class $Y = 0$ observations have feature vectors independently drawn from the mixture of two degenerate Gaussian distributions $\mathcal{N}(A_1\boldsymbol{\mu}_0^{(1)}, A_1A_1^T)$ and $\mathcal{N}(A_2\boldsymbol{\mu}_0^{(2)}, A_2A_2^T)$ with equal mixing probability. Similarly, class $Y = 1$ observations have feature vectors independently drawn from the mixture of two degenerate Gaussian distributions $\mathcal{N}(A_1\boldsymbol{\mu}_1^{(1)}, A_1A_1^T)$ and $\mathcal{N}(A_2\boldsymbol{\mu}_1^{(2)}, A_2A_2^T)$ with equal mixing probability.

Note that the subspace structure is completely latent to us, and we only observe data pairs (x_i, Y_i). We will focus on the LDA classification rule by pretending that observations were from two class Gaussian classifications. This corresponds to the first boxplot labelled as LDA in Figure 3.

Theorem 2. *Assume that $\boldsymbol{\mu}_0^{(1)} = \boldsymbol{\mu}_1^{(2)} = 0$ and $\|\boldsymbol{\mu}_1^{(1)}\|_2 = \|\boldsymbol{\mu}_0^{(2)}\|_2$. Then the Bayes rule has risk*

$$R_B = \frac{1}{4} + \bar{\Phi}(\|\boldsymbol{\mu}_1^{(1)}\|_2). \tag{4}$$

Assume further that $p/n \to 0$. Then for large enough n, it holds that

$$R_n(\delta_F) = R_B + o_p(1).$$

The full proof is provided in the Appendix. It is interesting to see from Equation (4) that as long as $\|\boldsymbol{\mu}_1^{(1)}\|_2$ is large enough, the overall classification error rate is close to 1/4. In view of our simulation results in Section 2.2, it is seen that our theory is consistent with the result in Figure 3 about LDA. Moreover, using our notation in this section, the asymptotic classification error within one subspace (as shown in Theorem 1) can be written as $\bar{\Phi}(\|\boldsymbol{\mu}_1^{(1)}\|_2/2)$. Combining this result with Theorem 2, we can see that the latent subspace structure does impose additional difficulties in classification, and the extra difficulty is mainly reflected in the term 1/4.

In addition, we also provide in Section A.3 some heuristic statistical arguments on how DNN learns in our simple subspace classification model introduced earlier, which support our empirical findings.

4 | REAL DATA APPLICATION

In addition to the simulation studies, we also demonstrate the revealed phenomenon on the real data application of traffic sign recognition, which is a crucial task in self-driving vehicles. The real data set (cropped images) is composed of 62 classes labelled as 0 to 61. The sizes of the images are around 128 ∗ 128 pixels, and each one contains a tuple for RGB colour ranging from 0 to 255. An overview of the images is shown in Figure A2 (Mathias, Timofte, Benenson, & Van Gool, 2013). We transform the images to 32 ∗ 32 pixels, which is beneficial for recognition. Thus, the number of features for each image is $32 * 32 * 3 = 3,072$. See also Cao, Song, Peng, Xiao, and Song (2019) and Shao et al. (2018).

Each image of a traffic sign may be captured by the camera from the front view or the side view, and brightness of the image could be low or high. We apply three clustering strategies: (a) four clusters with low brightness and front view (LBFV), low brightness and side view (LBSV), high brightness and front view (HBFV), and high brightness and side view (HBSV); (b) two clusters with low brightness (LB) and high brightness (HB); and (c) two clusters with front view (FV) and side view (SV). These strategies are motivated by the intermediate results by clustering algorithms (the images are clustered by brightness) and different aspect ratios of the images (the images are clustered by view angles). For instance, the images of label 21 (stop sign) can be partitioned into four clusters as shown in Figure A3.

Without applying the cluster structure a priori, the classification error rates by LDA, random forest, SVM, and deep learning are 17.38%, 12.86%, 12.42%, and 4.73%, respectively.

If we utilize the cluster identities on the basis of the above three strategies, then the classification error rates can be reduced for the clusters, in which images have low brightness and/or front view (LBFV of strategy 1, LB of strategy 2, and FV of strategy 3), which matches our intuition on the difficulty level of detecting the traffic signs. The sample sizes for these four clusters HBFV, HBSV, LBFV, and LBSV in the training data set are 1706, 799, 1380, and 690, respectively. As a result, random forest can lower the classification error rate by 1% using clustering strategies

1 and 2, SVM benefits from strategy 3 and also decreases the classification error rate by 1%, and the four-cluster strategy 1 facilitates LDA to obtain a much improved classification error rate of 9.56% error rate. So does deep learning capture the cluster structure internally over some hidden layers? As in our analysis for the simulation studies, we calculate the DCs of the deep features by the first hidden layer, second hidden layer, and their combinations with the one-hot cluster identities of the three clustering strategies; see Table 4. We see that the DCs are quite low. These real data results are consistent with our simulation experiment and theoretical interpretation that DNN does not learn the cluster structure but yet is capable of achieving comparable or even better classification performance in comparison with other machine leaning methods with clustering added on top.

5 | CONCLUSION

The main purpose of this paper is to provide some statistical insights into how DNN learns in the subspace classification model. We have designed a simple simulation study on two-class classification with data generated from a latent subspace structure. We compared DNN with the two-step procedures of clustering followed by classification. We discovered that although DNN has comparable classification performance to the two-step procedure, it does not conduct efficient clustering in any of its layers. This is a surprising result and in some sense provides an important complement to the common belief of representational learning for DNN. We also provided theoretical results to support our empirical findings and presented some heuristic statistical arguments on how DNN learns. The real data application of traffic sign recognition for self-driving vehicles further showcased the revealed phenomenon.

DATA ACCESSIBILITY

The real data that support the findings of this study are openly available at https://btsd.ethz.ch/shareddata/.
The simulation data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Hao Wu https://orcid.org/0000-0001-8899-2734

REFERENCES

Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., & Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. arXiv preprint arXiv:1801.07648.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., …, & Chen, J. (2016). Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International conference on machine learning* (pp. 173–182). New York City.

Arbel, M., Sutherland, D., Bińkowski, M., & Gretton, A. (2018). On gradient regularizers for MMD GANs, *Advances in neural information processing systems* (pp. 6700–6710). MIT Press: Montréal Canada.

Cao, J., Song, C., Peng, S., Xiao, F., & Song, S. (2019). Improved traffic sign detection and recognition algorithm for intelligent vehicles. *Sensors*, *19*(18), 4021.

Chen, Y., Lin, Z., Zhao, X., Wang, G., & Gu, Y. (2014). Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, *7*(6), 2094–2107.

Cireşan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745.

Deng, J., Guo, J., & Zafeiriou, S. (2018). Arcface: Additive angular margin loss for deep face recognition. arXiv preprint arXiv:1801.07698.

Dolz, J., Reyns, N., Betrouni, N., Kharroubi, D., Quidet, M., Massoptier, L., & Vermandel, M. (2017). A deep learning classification scheme based on augmented-enhanced features to segment organs at risk on the optic region in brain cancer patients. arXiv preprint arXiv:1703.10480.

Dziugaite, G. K., Roy, D. M., & Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. arXiv preprint arXiv:1505.03906.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118.

Fan, J., & Fan, Y. (2008). High dimensional classification using features annealed independence rules. *The Annals of statistics*, *36*(6), 2605–2637.

Fan, Y., Jin, J., & Yao, Z. (2013). Optimal classification in sparse Gaussian graphic model. *The Annals of Statistics*, *41*(5), 2537–2571.

Fan, Y., Kong, Y., Li, D., & Zheng, Z. (2015). Innovated interaction screening for high-dimensional nonlinear classification. *The Annals of Statistics*, *43*(3), 1243–1272.

Fan, J., & Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *70*(5), 849–911.

Fan, Y., & Lv, J. (2013). Asymptotic equivalence of regularization methods in thresholded parameter space. *Journal of the American Statistical Association*, *108*(503), 1044–1061.

Fan, Y., & Tang, C. Y. (2013). Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *75*(3), 531–552.

Gao, L., Lv, J., & Shao, Q. (2019). Asymptotic distributions of high-dimensional nonparametric inference with distance correlation. arXiv preprint arXiv:1910.12970.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 513–520). Bellevue, Washington, USA.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). Las Vegas, Nevada, USA.

Heaton, J., Polson, N., & Witte, J. H. (2017). Deep learning for finance: Deep portfolios. *Applied Stochastic Models in Business and Industry*, *33*(1), 3–12.

Heckel, R., & Bölcskei, H. (2015). Robust subspace clustering via thresholding. *IEEE Transactions on Information Theory*, *61*(11), 6320–6342.

Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks, *Advances in neural information processing systems*, pp. 8571–8580.

Jones, T., & Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proc. 6th Internat. Conf. on Genetic Algorithms* (pp. 184–192). Pittsburgh, PA.

Kong, Y., Li, D., Fan, Y., & Lv, J. (2017). Interaction pursuit in high-dimensional multi-response regression via distance correlation. *The Annals of Statistics*, *45*(2), 897–922.

Kussul, N., Lavreniuk, M., Skakun, S., & Shelestov, A. (2017). Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, *14*(5), 778–782.

Lee, H., Pham, P., Largman, Y., & Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks, *Advances in neural information processing systems* (pp. 1096–1104). Vancouver, B.C., Canada: MIT Press.

Li, Q., & Shao, J. (2015). Sparse quadratic discriminant analysis for high dimensional data. *Statistica Sinica*, *25*(2), 457–473.

Li, R., Zhong, W., & Zhu, L. (2012). Feature screening via distance correlation learning. *Journal of the American Statistical Association*, *107*(499), 1129–1139.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ..., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, *42*, 60–88.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). SphereFace: Deep hypersphere embedding for face recognition. arXiv preprint arXiv:1704.08063.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, *28*(2), 129–137.

Lu, Y., Fan, Y., Lv, J., & Noble, W. S. (2018). DeepPINK: reproducible feature selection in deep neural networks, pp. 8689–8699.

Lv, J. (2013). Impacts of high dimensionality in finite samples. *Ann Stat*, *41*(4), 2236–2262.

van der Maaten, L, & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Majumder, N., Poria, S., Gelbukh, A., & Cambria, E. (2017). Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, *32*(2), 74–79.

Mathias, M., Timofte, R., Benenson, R., & Van Gool, L. (2013). Traffic sign recognition—how far are we from the solution? In *The 2013 international joint conference on neural networks (ijcnn)* (pp. 1–8). Dallas, Texas, USA: IEEE.

Mhaskar, H. N., & Poggio, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, *14*(06), 829–848.

Park, D., Caramanis, C., & Sanghavi, S. (2014). Greedy subspace clustering. In *Advances in neural information processing systems* (pp. 2753–2761). Montréal Canada.

Patel, A. B., Nguyen, T., & Baraniuk, R. G. (2015). A probabilistic theory of deep learning. arXiv preprint arXiv:1504.00641.

Poggio, T., Kawaguchi, K., Liao, Q., Miranda, B., Rosasco, L., Boix, X., ..., & Mhaskar, H. (2017). Theory of deep learning III: Explaining the non-overfitting puzzle. arXiv preprint arXiv:1801.00173.

Reshef, D. N., Reshef, Y. A., Finucane, H. K., Grossman, S. R., McVean, G., Turnbaugh, P. J., ..., & Sabeti, P. C. (2011). Detecting novel associations in large data sets. *science*, *334*(6062), 1518–1524.

Rokach, L., & Maimon, O. (2005). Clustering methods, *Data mining and knowledge discovery handbook*, pp. 321–352.

Saxe, A., Bansal, Y, Dapello, J, Advani, M, Kolchinsky, A, Tracey, B., & Cox, D. (2018). On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, IOP Publishing, 124020.

Shao, F., Wang, X., Meng, F., Rui, T., Wang, D., & Tang, J. (2018). Real-time traffic sign detection and recognition method based on simplified Gabor wavelets and CNNs. *Sensors*, *18*(10), 3192.

Socher, R., Huval, B., Bath, B., Manning, C. D., & Ng, A. Y. (2012). Convolutional-recursive deep learning for 3D object classification. In *Advances in neural information processing systems* (pp. 656–664). Lake Tahoe, Nevada, USA.

Soltanolkotabi, M., Elhamifar, E., & Candes, E. J. (2014). Robust subspace clustering. *The Annals of Statistics*, *42*(2), 669–699.

Székely, G. J, & Rizzo, M. L. (2013). The distance correlation t-test of independence in high dimension. *Journal of Multivariate Analysis*, *117*, 193–213.

Szkely, G. J., Rizzo, M. L., & Bakirov, N. K. (2007). Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, *35*(6), 2769–2794.

Wang, F., Cheng, J., Liu, W., & Liu, H. (2018). Additive margin softmax for face verification. *IEEE Signal Processing Letters*, *25*(7), 926–930.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Li, Z., Gong, D., ..., & Liu, W. (2018). CosFace: Large margin cosine loss for deep face recognition. arXiv preprint arXiv:1801.09414.

Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2019). A comprehensive study on center loss for deep face recognition. *International Journal of Computer Vision*, *127*, 1–16.

How to cite this article: Wu H, Fan Y, Lv J. Statistical insights into deep neural network learning in subspace classification. *Stat*. 2020;9:e273. https://doi.org/10.1002/sta4.273

APPENDIX A

One real-life problem that motivates our study is shown in Figure A1, where we classify cats and dogs from the a collection of images capturing different activities (e.g., eating, playing, and sleeping). In our model, we mimic the set of images through a high-dimensional space, which is the union of three subspaces, and each low-dimensional subspace corresponds to one type of activities.

A.1 | Building deep neural networks

We train and test different structures of DNNs over 100 training and test data sets with the same sample size, which are randomly generated according to our model setting. We choose the optimal hyper parameters on the validation set (10% of the training set). DNNs are run on both CPUs and GPUs; we discover that when the network is shallow or has small number of hidden units, CPUs outperform GPUs in terms of computational speed. However, as the number of hidden layers and units increases, GPUs definitely run faster.

A.2 | Proof of Theorem 2

Proof. Recall that under our model assumption, class $Y = 0$ observations have feature vectors independently drawn from the mixture of two degenerate Gaussian distributions $\mathcal{N}(A_1\boldsymbol{\mu}_0^{(1)}, A_1A_1^T)$ and $\mathcal{N}(A_2\boldsymbol{\mu}_0^{(2)}, A_2A_2^T)$ with equal mixing probability. Similarly, class $Y = 1$ observations have feature vectors independently drawn from the mixture of two degenerate Gaussian distributions $\mathcal{N}(A_1\boldsymbol{\mu}_1^{(1)}, A_1A_1^T)$ and $\mathcal{N}(A_2\boldsymbol{\mu}_1^{(2)}, A_2A_2^T)$

FIGURE A1 Classification of cats and dogs (copyright by Google Images)

TABLE A1 DNN with hidden units [10]

Layer	Training error		Test error		Run time (s)	
	CPU	GPU	CPU	GPU	CPU	GPU
1	0.1282	0.1283	0.1548	0.1536	8.1665	14.4766
2	0.0923	0.0875	0.1328	0.1274	9.5305	21.9774
3	0.1016	0.1262	0.1518	0.1521	11.0447	19.4718
4	0.0993	0.0961	0.1555	0.1580	12.7921	26.6293
5	0.1104	0.1263	0.1697	0.1826	14.3730	28.7950
10	0.4439	0.4483	0.4540	0.4575	22.6595	38.1027

Abbreviations: CPU, central processing unit; DNN, deep neural network; GPU, graphics processing unit.

TABLE A2 DNN with hidden units [20]

Layer	Training error		Test error		Run time (s)	
	CPU	GPU	CPU	GPU	CPU	GPU
1	0.1051	0.1085	0.1324	0.1339	9.5807	19.5555
2	0.0500	0.0510	0.0940	0.0956	12.1759	21.2431
3	0.0391	0.0399	0.1033	0.1051	14.8497	23.4055
4	0.0330	0.0330	0.1213	0.1233	18.1428	26.5479
5	0.0342	0.0318	0.1469	0.1433	21.0214	37.0104
10	0.2165	0.1794	0.2649	0.2371	35.0329	50.8225

Abbreviations: CPU, central processing unit; DNN, deep neural network; GPU, graphics processing unit.

TABLE A3 DNN with hidden units [50]

Layer	Training error		Test error		Run time (s)	
	CPU	GPU	CPU	GPU	CPU	GPU
1	0.0848	0.0847	0.1139	0.1137	10.1360	19.5282
2	0.0306	0.0300	0.0795	0.0795	15.1895	17.0301
3	0.0105	0.0105	0.0812	0.0812	20.1912	16.5940
4	0.0057	0.0054	0.0967	0.0956	25.2243	17.9423
5	0.0033	0.0030	0.1389	0.1342	30.0675	29.2289
10	0.0170	0.0255	0.1565	0.1599	53.2125	40.1224

Abbreviations: CPU, central processing unit; DNN, deep neural network; GPU, graphics processing unit.

TABLE A4 DNN with hidden units [100]

Layer	Training error		Test error		Run time (s)	
	CPU	GPU	CPU	GPU	CPU	GPU
1	0.0781	0.0777	0.1077	0.1073	12.8185	18.7767
2	0.0201	0.0206	0.0765	0.0764	21.2060	21.5077
3	0.0030	0.0031	0.0770	0.0775	29.3675	24.0765
4	0.0013	0.0013	0.0813	0.0807	37.3774	26.7721
5	0.0009	0.0008	0.1002	0.0990	45.5618	23.3326
10	0.0021	0.0045	0.1422	0.1404	84.7540	42.6131

Abbreviations: CPU, central processing unit; DNN, deep neural network; GPU, graphics processing unit.

with equal mixing probability. Therefore, for a random observation $\mathrm{x} \in \mathbb{R}^D$ from class k, we have the condition mean

$$m_k \equiv \mathrm{E}[\mathrm{x}|Y = k] = \begin{pmatrix} \frac{1}{2}\mu_k^{(1)} \\ \frac{1}{2}\mu_k^{(2)} \end{pmatrix},$$

and the conditional covariance matrix

$$\mathrm{Var}(\mathrm{x}|Y = k) = \begin{pmatrix} \frac{1}{2}I_p + \frac{1}{4}\mu_k^{(1)}(\mu_k^{(1)})^T & 0 \\ 0 & \frac{1}{2}I_p + \frac{1}{4}\mu_k^{(2)}(\mu_k^{(2)})^T \end{pmatrix}.$$

Therefore, for an observation randomly drawn, the unconditional covariance matrix can be calculated as

$$\mathrm{Var}(\mathrm{x}) = \mathrm{E}[\mathrm{Var}(\mathrm{x}|Y)] + \mathrm{Var}(\mathrm{E}[\mathrm{x}|Y]) = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

where

$$C_{11} = \frac{1}{2}I_p + \frac{3}{16}(\mu_0^{(1)}(\mu_0^{(1)})^T + \mu_1^{(1)}(\mu_1^{(1)})^T),$$

$$C_{12} = \frac{1}{16}(\mu_0^{(1)}(\mu_0^{(2)})^T + \mu_1^{(1)}(\mu_1^{(2)})^T),$$

$$C_{21} = \frac{1}{16}(\mu_0^{(2)}(\mu_0^{(1)})^T + \mu_1^{(2)}(\mu_1^{(1)})^T),$$

$$C_{22} = \frac{1}{2}I_p + \frac{3}{16}(\mu_0^{(2)}(\mu_0^{(2)})^T + \mu_1^{(2)}(\mu_1^{(2)})^T).$$

Recall that we assume $\boldsymbol{\mu}_0^{(1)} = 0$ and $\boldsymbol{\mu}_1^{(2)} = 0$. Thus, the above conditional mean and unconditional covariance matrix can be simplified as

$$m_0 = \mathrm{E}[\mathrm{x}|Y = 0] = \begin{pmatrix} 0 \\ \frac{1}{2}\mu_0^{(2)} \end{pmatrix},$$

$$m_1 = \mathrm{E}[\mathrm{x}|Y = 1] = \begin{pmatrix} \frac{1}{2}\mu_1^{(1)} \\ 0 \end{pmatrix},$$

and

$$\Sigma_0 \equiv \mathrm{Var}(\mathrm{x}) = \begin{pmatrix} \frac{1}{2}I_p + \frac{3}{16}\mu_1^{(1)}(\mu_1^{(1)})^T & 0 \\ 0 & \frac{1}{2}I_p + \frac{3}{16}\mu_0^{(2)}(\mu_0^{(2)})^T \end{pmatrix},$$

respectively.

Now, suppose we completely ignore the subspace structure and use the Bayes rule from two class Gaussian classifications, that is, by pretending that the data were from Gaussian distributions

$$\mathcal{N}(m_0, \Sigma_0) \text{ and } \mathcal{N}(m_1, \Sigma_0),$$

we obtain the following decision rule

$$\delta(\mathrm{x}) = \mathbb{1}\{r(\mathrm{x}) := \left(\mathrm{x} - \frac{m_0 + m_1}{2}\right)^T \Sigma_0{}^{-1}(m_1 - m_0) > 0\}. \tag{A1}$$

We next calculate the classification error of the above decision rule (A1). For a new observation x with true class label Y, by conditional probability,

$$P(\delta(\mathrm{x}) \neq Y) = P(\delta(\mathrm{x}) = 0 | Y = 1)\frac{1}{2} + P(\delta(\mathrm{x}) = 1 | Y = 0)\frac{1}{2}. \tag{A2}$$

We first calculate $P(\delta(\mathrm{x}) = 0|Y = 1)$. Because conditional on class 1, a random observation has equal probability of coming from the two subspaces, we have

$$P(\delta(\mathrm{x}) = 0 | Y = 1) = \frac{1}{2}P(\delta(\mathrm{x}) = 0 | Y = 1, \mathrm{x} \in S_1) + \frac{1}{2}P(\delta(\mathrm{x}) = 0 | Y = 1, \mathrm{x} \in S_2). \tag{A3}$$

Now, recall that for a random observation x from class $Y = 1$ and subspace S_1, we have

$$\mathrm{x} \sim \mathcal{N}(A_1\boldsymbol{\mu}_1^{(1)}, A_1A_1^T).$$

Therefore, the classification rule $\delta(\mathrm{x})$ has Gaussian distribution with mean and variance

$$\begin{aligned}
\mathrm{E}[r(\mathrm{x})|Y = 1, \mathrm{x} \in S_1] &= \frac{3}{8}(\boldsymbol{\mu}_1^{(1)})^T\left(\frac{1}{2}I_p + \frac{3}{16}\mu_1^{(1)}(\mu_1^{(1)})^T\right)^{-1}\boldsymbol{\mu}_1^{(1)} + \frac{1}{8}(\mu_0^{(2)})^T\left(\frac{1}{2}I_p + \frac{3}{16}\mu_0^{(2)}(\mu_0^{(2)})^T\right)^{-1}\boldsymbol{\mu}_0^{(2)} \\
&= \frac{3}{4}\frac{\|\boldsymbol{\mu}_1^{(1)}\|_2^2}{1 + \frac{3}{8}\|\boldsymbol{\mu}_1^{(1)}\|_2^2} + \frac{1}{4}\frac{\|\boldsymbol{\mu}_0^{(2)}\|_2^2}{1 + \frac{3}{8}\|\boldsymbol{\mu}_0^{(2)}\|_2^2} = \frac{\|\boldsymbol{\mu}_1^{(1)}\|_2^2}{1 + \frac{3}{8}\|\boldsymbol{\mu}_1^{(1)}\|_2^2}
\end{aligned}$$

$$\mathrm{Var}(r(\mathrm{x})|Y = 1, \mathrm{x} \in S_1) = \frac{1}{4}(\boldsymbol{\mu}_1^{(1)})^T\left(\frac{1}{2}I_p + \frac{3}{16}\mu_1^{(1)}(\mu_1^{(1)})^T\right)^{-2}\boldsymbol{\mu}_1^{(1)} = \frac{\|\boldsymbol{\mu}_1^{(1)}\|_2^2}{(1 + \frac{3}{8}\|\boldsymbol{\mu}_1^{(1)}\|_2^2)^2},$$

respectively, where we have used the assumption that $\|\boldsymbol{\mu}_0^{(2)}\|_2^2 = \|\boldsymbol{\mu}_1^{(1)}\|_2^2$. Thus, it can be calculated that

$$P(\delta(\mathrm{x}) = 0 | Y = 1, \mathrm{x} \in S_1) = \bar{\Phi}\left(\|\boldsymbol{\mu}_1^{(1)}\|_2\right).$$

Similarly, it can be derived that

$$\begin{aligned}
P(\delta(\mathrm{x}) = 0 | Y = 1, \mathrm{x} \in S_2) &= \Phi\left(\|\boldsymbol{\mu}_1^{(2)}\|_2\right) = 1/2, \\
P(\delta(\mathrm{x}) = 1 | Y = 0, \mathrm{x} \in S_1) &= \Phi\left(\|\boldsymbol{\mu}_0^{(1)}\|_2\right) = 1/2, \\
P(\delta(\mathrm{x}) = 1 | Y = 0, \mathrm{x} \in S_2) &= \bar{\Phi}\left(\|\boldsymbol{\mu}_0^{(2)}\|_2\right).
\end{aligned}$$

This proves the first result of the theorem.

Combining the above result with Equations (A2) and (A3), we arrive at

$$R(\delta_{\mathrm{B}}) = P(\delta(\mathrm{x}) \neq Y) = \frac{1}{4} + \frac{1}{2}\bar{\Phi}\left(\|\boldsymbol{\mu}_0^{(2)}\|_2\right).$$

When the population parameters are estimated from sample, as long as the sample size satisfying that $p/n \to 0$, then using similar arguments as in Li and Shao (2015), we can prove that conditional on training data, the classification error rate of the plug-in classifier $\delta_F(\mathrm{x})$ satisfies that with asymptotic probability 1,

$$R_n(\delta_F) \to R(\delta_{\mathrm{B}}).$$

This completes the proof of the theorem. □

A.3 | Some heuristic arguments on how DNN learns in subspace classification

In this section, we intend to provide some heuristic statistical arguments on how DNN learns in our simple subspace classification model introduced earlier. In such a model, there are k subspaces S_j's of the ambient feature space $\mathbb{R}^D$ with $1 \leq j \leq k$; and within each subspace, there are two classes that are labelled as 0 and 1. For simplicity, we assume that the k subspaces S_j's are generated independently and that each subspace S_j has dimensionality $p = p_j$ and is uniformly distributed on the Grassmann manifold $G_{D,p}$, which is composed of all p-dimensional subspaces of $\mathbb{R}^D$. See, for example, Lv (2013) for some brief background on the geometry and invariant measure of Grassmann manifold. To simplify the technical exposition, we further assume that for each subspace S_j, the data distributions corresponding to the two classes 0 and 1 are $\mathcal{N}(0, I_{p_j})$ and $\mathcal{N}(\boldsymbol{\mu}_j, I_{p_j})$, respectively, and that each class on each subspace has sample size n_0. Thus, the total sample size is $n = 2kn_0$. Moreover, assume that the k vectors $\boldsymbol{\mu}_j$'s are also statistically independent of each other. Denote by H_j the $(p_j - 1)$-dimensional affine hyperplane passing through point $2^{-1}\boldsymbol{\mu}_j$ and with normal vector $\boldsymbol{\mu}_j$. The half space of subspace S_j containing point $\boldsymbol{\mu}_j$ is referred to as the positive half space for brevity. Then

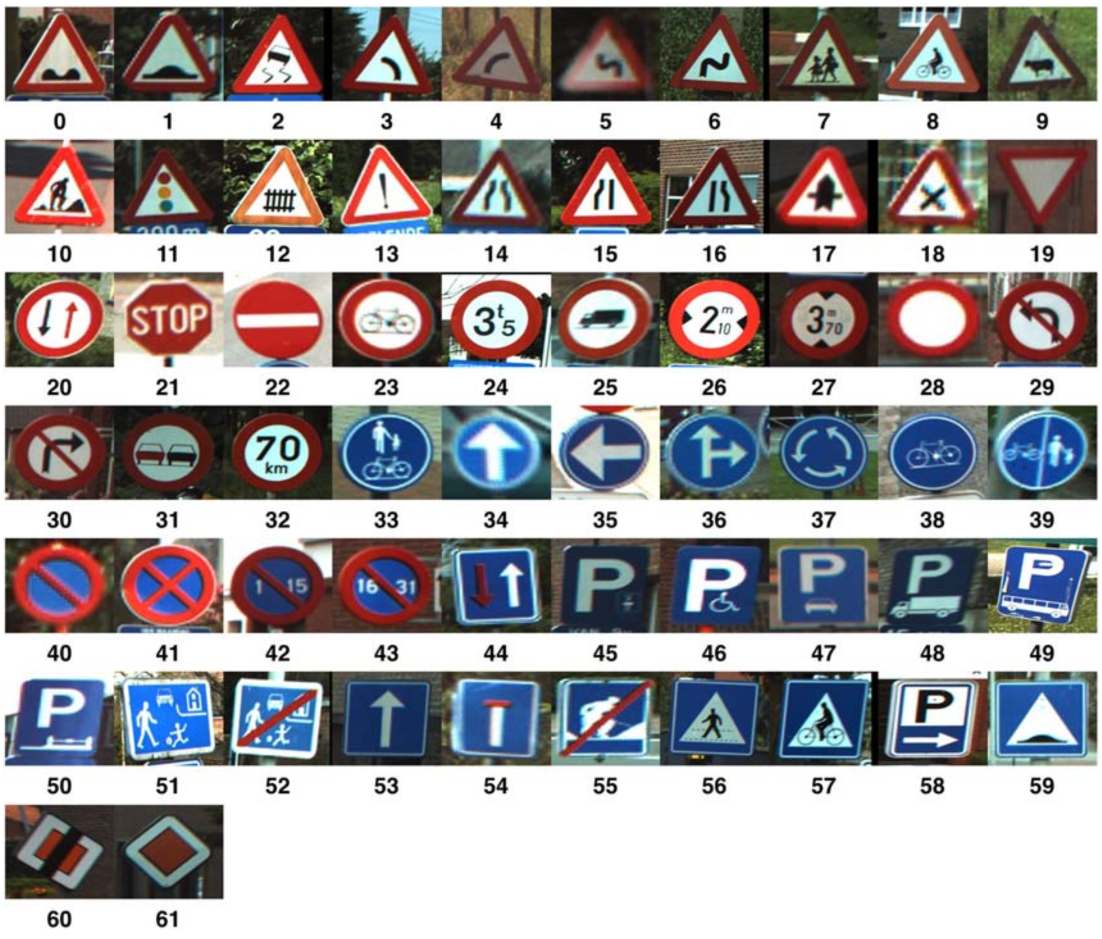

FIGURE A2 Traffic signs of labels 0 to 61

an ideal classification procedure is assigning the true subspace cluster label to each data point and classifying it as class 1 or 0 according to whether or not the data point lies in the positive half space of the corresponding subspace. Such an ideal classification procedure is an optimal classifier because it knows the true subspace cluster label and the underlying classifier given each subspace is the Bayes classifier.

We next gain some statistical insights into how DNN tries to mimic closely the above ideal classification procedure without recovering the latent subspace cluster label. Although the k subspaces S_j's lie in the same ambient Euclidean space $\mathbb{R}^D$, it is more useful to view them as points on the Grassmann manifolds, which are compact Riemannian homogeneous spaces. To ease the technical presentation of the underlying geometry, we further assume that all the k subspaces S_j's have the same dimensionality p. Thus, $(S_j)_{1\le j\le k}$ is simply an independent and identically distributed (i.i.d.) sample from the uniform (Haar) distribution on the Grassmann manifold $G_{D,p}$. By the classical manifold embedding theory from geometry and topology, the Grassmann manifold $G_{D,p}$, which is a curved space of dimension $p(D-p)$, can be embedded into a higher dimensional Euclidean space where the distance on the Grassmann manifold is preserved. Here, in a similar spirit, DNN tries to embed the points on each subspace S_j in a high-dimensional latent Euclidean space that is different than the original ambient Euclidean space $\mathbb{R}^D$.

Let us consider DNN with multiple hidden layers and the sigmoid layer at the top for two-class classification. As is common in the more recent deep learning literature, we consider the rectified linear unit (ReLU) activation function for all the neurons in the hidden layers. For a given ReLU neuron l, assume that the inputs are z_i with $1 \le i \le m_l$, the bias is w_{0l}, and the network weights are w_{il} with $1 \le i \le m_l$. Denote by $\mathrm{z}_l = (1, z_1, \dots, z_{m_l})^T$ and $\mathrm{w}_l = (w_{0l}, w_{1l}, \dots, w_{m_l l})^T$ the input vector and the weight vector, respectively, associated with this particular ReLU neuron, where the bias is regarded as the weight for the constant input 1. Then the output of this neuron is given by $h_l(\mathrm{z}_l, \mathrm{w}_l) = (\mathrm{z}_l^T \mathrm{w}_l)_+$, where $a_+ = \max(0, a)$ for each scalar a. In view of such a representation, each ReLU neuron in the first (bottom) hidden layer projects the input vector $\mathrm{x} = (x_1, \dots, x_D)^T \in \mathbb{R}^D$ onto the line $\{t(w_{1l}, \dots, w_{m_l l})^T : t \in \mathbb{R}\}$ with a given centre and scale and zeros out all the negative values of this new coordinate, that is, collapsing the negative half of the real line to a single point zero.

Assume that there are at least kp ReLU neurons in the bottom hidden layer. It is easy to see that for each $1 \le j \le k$, there exist a set of p ReLU neurons with some suitable bias and weight parameters such that the resulting p ReLU coordinates as given above provide a one-to-one mapping from the positive half space associated with subspace S_j to $\mathbb{R}_+^p$ and shrink the corresponding negative half space toward the origin (by comparing each data point in the positive half space and its mirror image with respect to H_j), where $\mathbb{R}_+$ stands for the positive half of the real line. To simplify the arguments, assume that the total number of ReLU neurons in the bottom hidden layer is exactly kp; otherwise, we assign zero values to all the parameters of those remaining ReLU neurons for the moment. By our construction, the network parameters for the p ReLU neurons associated

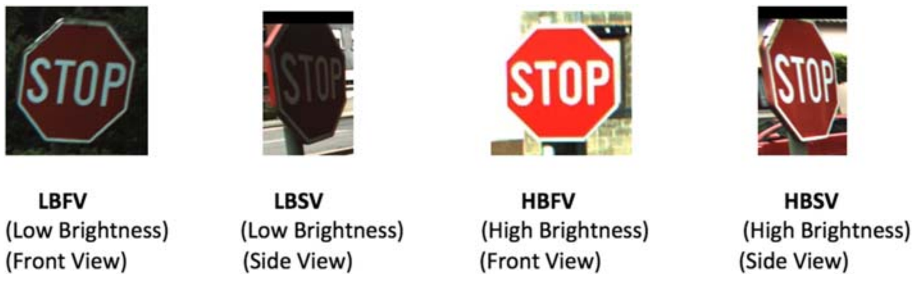

FIGURE A3 Instances of traffic signs in each cluster

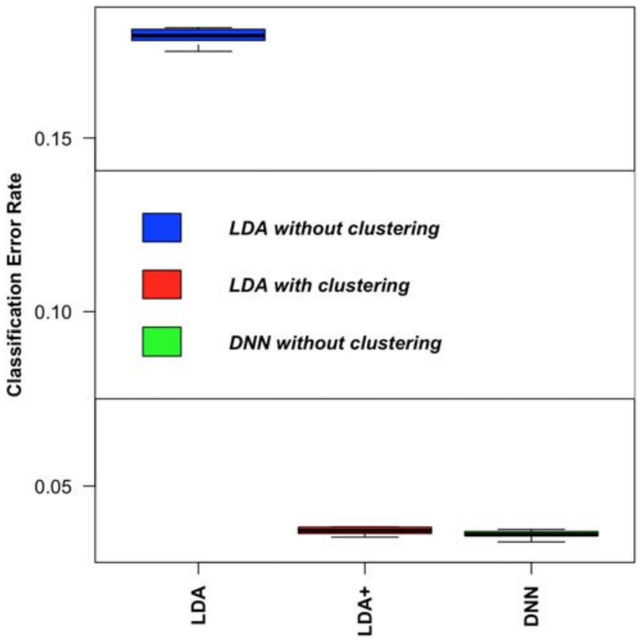

FIGURE A4 Classification error rates of linear discriminant analysis (LDA) and deep neural network (DNN)

with subspace S_j depend only on S_j and $\boldsymbol{\mu}_j$ and thus are independent of all other S_l's and $\boldsymbol{\mu}_l$'s with $1 \le l \ne j \le k$. To mimic the random initialization of network parameters in the training of DNN, let us further add some small independent Gaussian noises to the network parameters for the bottom hidden layer. Applying the classical deviation bounds for light-tailed distributions with conditioning, we can show that as the growth rates of k, p, and D relative to sample size n are properly controlled, the above choice of random network weights maps each subspace S_j into a higher dimensional space $\mathbb{R}_+^{kp}$; and with significant probability, such a mapping is a small perturbation of the aforementioned mapping from subspace S_j to $\mathbb{R}_+^{p}$; see, for example, the technical arguments in Fan and Lv (2008), Fan and Lv (2013), Fan and Tang (2013), and Lv (2013).

Observe that in our above probabilistic and geometric construction, all the data points in the positive half space associated with each subspace S_j tend to concentrate on a relatively large compact set in $\mathbb{R}_+^{kp}\backslash\{0\}$ under the latent ReLU coordinates, whereas all the remaining data points tend to concentrate around the origin due to the shrinkage effect mentioned before. Such a geometric representation of the two classes in a higher dimensional latent Euclidean space leads to the fact that the training loss of the DNN can approximate closely that of the above ideal classification procedure as the sample size n increases. Therefore, the resulting classification accuracy of the DNN can mimic closely that of the ideal classification procedure. Meanwhile, it is important to notice that by its nature, the latent representation given by the ReLU neurons in the hidden layers generally does not lead to successful recovery of the latent subspace cluster labels.

If the width of the neural network is limited or there are multiple classes within each subspace cluster, we need to use the ReLU neurons in additional hidden layers to build up effective embedding of the subspaces in a high-dimensional latent Euclidean space. A formal, rigorous theory on the above heuristic arguments and statistical insights for DNN in the subspace classification model will be presented in a separate paper.

A.4 | Figures from real data application

A.5 | Additional large-scale simulation study

Additionally, we simulate $n_k = 8,000$ observations from each subspace S_k, and we set $D = 1,000$ and $p = 40$. So we end up with $n = 24,000$ observations (x_i, Y_i) in total, where each feature vector x_i has dimensionality $D = 1,000$. The patterns and phenomenon observed previously stay the same in the large scale.

The classification error rate of LDA without clustering is 0.1795, whereas the classification error rate of DNN (without clustering) is 0.0361, which is comparable with the value 0.0371 for LDA with the knowledge of cluster identities; see Figures A4 and A5 for details.

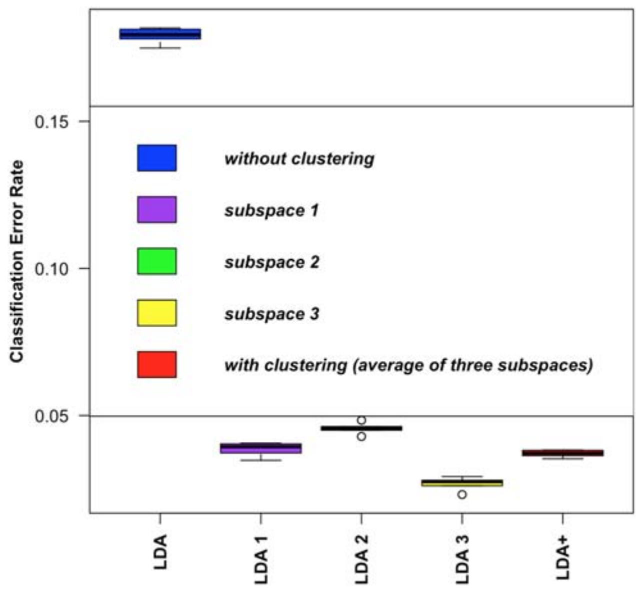

FIGURE A5 Classification error rate for linear discriminant analysis (LDA) regarding subspaces

TABLE A5 Bias-corrected distance correlations (DCs)

Variable 1	Variable 2	DC
X_c	$Z_{subspace}$	**0.8944**
$X_{bottomLayer}$	$Z_{subspace}$	**0.2014**
$X_{topLayer}$	$Z_{subspace}$	**0.1501**

Note. X_c : transformed X via clustering; $X_{bottomLayer}$: *n*-vector representing the hidden layer after the input layer; $X_{topLayer}$: *n*-vector representing the hidden layer before the output layer; $Z_{subspace}$: the latent subspace label for each observation. Bold data emphasize the trend.

TABLE A6 Clustering error rates on hidden layers

Method	$X_{bottomLayer}$	$X_{topLayer}$	$X_{bottomLayer} + X_{topLayer}$
GreedySC	0.479288	0.555154	0.540575
RSCT	0.498617	0.495413	0.497163
K-means	0.502529	0.513588	0.509113
SVD + *K*-means	0.510063	0.513175	0.506933
Hierarchical	0.666613	0.666617	0.666625
SVD + Hierarchical	0.666613	0.666617	0.666613

Abbreviations: GreedySC, greedy subspace clustering; RSCT, robust subspace clustering via thresholding; SVD, singular value decomposition.

To examine whether any layers of DNN carry the latent cluster information, again, we apply various popularly used clustering methods to the bottom hidden layer (bHidden), top hidden layer (tHidden), and the combination of the two hidden layers (bHidden + tHidden) in Table A6. We also calculate the bias-corrected distance correlation of $Z_{subspace}$ paired with X_c, $X_{bottomLayer}$, and $X_{topLayer}$ in Table A5. As seen in these tables, the high clustering error rates and decreasing trend of bias-corrected distance correlations stay the same in the large scale, suggesting that DNN does not really do clustering in any of its layers in the large-scale model setting.