Subspace Interpolation and Indexing on Stiefel and Grassmann Manifolds as a Lightweight Inference Engine

Wenqing Hu
Department of Mathematics and Statistics
Missouri University of Science and Technology
Rolla, Missouri, USA, 65409
Email: huwen@mst.edu

Tiefeng Jiang
School of Statistics
University of Minnesota, Twin Cities
Minneapolis, MN, USA, 55414
Email: jiang040@umn.edu

Birendra Kathariya
Department of Computer Science & Electrical Engineering
University of Missouri, Kansas City
Kansas City, MO, USA, 64110

Vikram Abrol
Department of Computer Science & Electrical Engineering
University of Missouri, Kansas City
Kansas City, MO, USA, 64110

Jiali Zhang
Department of Mathematics and Statistics
Missouri University of Science and Technology
Rolla, Missouri, USA, 65409

Zhu Li
Department of Computer Science & Electrical Engineering
University of Missouri, Kansas City
Kansas City, MO, USA, 64110
Email: lizhu@umkc.edu

***Abstract*—Subspace Indexing with Interpolation (SIM-I) on Stiefel and Grassmann manifolds is proposed in this work. Given a partition of some original high-dimensional data set, SIM-I is constructed via two steps: in the first step we build linear affinity-aware subspace models based on each partition; in the second step we interpolate between several adjacent linear subspace models constructed in the first step using the "center of mass" calculation on Stiefel and Grassmann manifolds. Through these two steps, SIM-I builds a globally nonlinear and smoothly regularized low-dimensional embedding model of the original data set. Furthermore, given sufficiently many training samples on the data manifold either labelled by some pre-trained learning model such as Deep Neural Networks (DNNs) or provided with original natural labels, we first apply SIM-I on this data set and then perform nearest-neighbor classification on the resulting low-dimensional embedding. This helps us to build a Lightweight Inference Engine (LIE) carrying similar level of feature extraction by the pre-trained learning model. For DNNs, such LIE can be interpreted as some (nonstandard) shallow neural network with a wide first hidden layer. From this perspective, SIM-I provides a way to exchange deep network for wide but shallow ones and may provide some new insights to interpret DNNs.**

1. Introduction

Subspace selection algorithms have been successful in many application problems related to dimension reduction ([1], [2], [3], [4]), with applications including, e.g., human face recognition ([5]), speech and gait recognition ([6]), etc.. Classical approaches, such as Principle Component Analysis (PCA, see [7]) and Linear Discriminant Analysis (LDA, see [8], [9]), are looking for globally linear subspace models and are thus failing to estimate the nonlinearity of the intrinsic data manifold, and they also ignore the local variation of the data ([10], [11]). Consequently, these globally linear models are often ineffective for search problems on large scale image data sets. To remedy this, nonlinear algorithms such as kernel algorithms ([12]), manifold learning ([13], [14]) and Deep Neural Networks (DNNs, [15]) are proposed. Typically, nonlinear methods are subject to heavy computational cost and needs techniques that enable efficient processing ([16]).

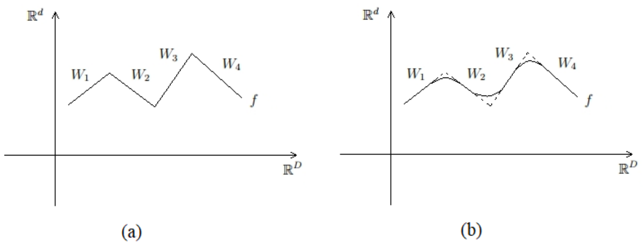

Figure 1: The idea of "smoothing" a piecewise linear low-dimensional embedding model: (a) The piecewise linear low-dimensional embedding model built from LPP; (b) The regularized low-dimensional embedding by taking Stiefel/Grassmann manifold center-of-mass among adjacent linear pieces.

Here we propose Subspace Indexing Model with Interpolation (SIM-I), that produces from a given high-dimensional data set a piecewise linear, locality aware and globally nonlinear model of low-dimensional embedding. SIM-I is constructed via two steps: in the first step we build a piecewise linear affinity-aware subspace model under a given partition of the data set; in the second step we interpolate between several adjacent linear subspace models constructed in the first step using the "center of mass" calculation on Stiefel and Grassmann manifolds ([17], [18], [19]). The interpolation step outputs a "smoothed" version (Figure 1) of the original piecewise linear model, and can be regarded as a regularization process. Compared to previously mentioned subspace methods, SIM-I enjoys the following advantages: (1) it captures the *global nonlinearity* and thus the local fluctuations of the data set; (2) it is *computationally feasible* to large-scale data sets since it avoids heavy computations in training; (3) it includes a *regularization step* via interpolating between several adjacent pieces of subspace models.

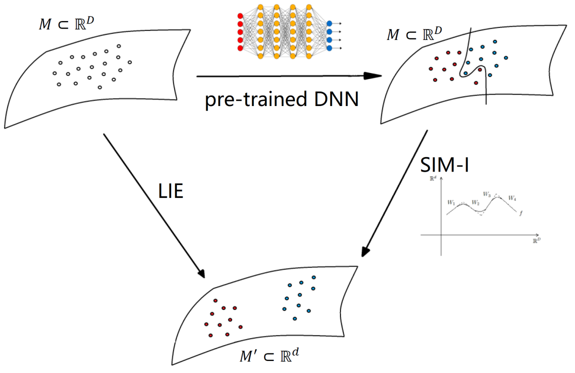

Figure 2: SIM-I combined with a pre-trained learning model and the data manifold enables Lightweight Inference Engine (LIE).

Our construction of SIM-I further enables us to build a Lightweight Inference Engine (LIE). Actually, given sufficiently many training samples on the high-dimensional data manifold, labelled by some pre-trained learning model such as Deep Neural Networks (DNNs), we can first apply SIM-I on this data set and then we perform k-Nearest-Neighbor (kNN) classification on the resulting low-dimensional embedding (Figure 2). Viewing this procedure as a direct mapping from the original high-dimensional data manifold to the classification labels, we obtain an inference engine that carries similar level of feature extraction by the pre-trained learning model. Since SIM-I is a low-dimensional embedding that does not require many hierarchies of function compositions (like in DNNs), our inference engine is light weighted (the name LIE thus follows).

The above LIE construction is also applicable when the training samples are given their original natural labels instead of being labelled by pre-trained learning models. In this case, following the same procedure, LIE can be regarded as a new learning model that can produce high level of classification accuracy as some standard learning models such as DNNs.

It is worth mentioning here that at a high level, our SIM-I based LIE can be interpreted as a shallow neural network with very wide first hidden layer activated by ReLU-like nonlinearity. From this perspective, when the pre-trained learning model is a DNN, SIM-I provides a way to exchange deep network for wide but shallow ones. However, we will illustrate in Section 5 that such a shallow network is different from the usual feed-forward networks with explicit activation functions. Rather, it is equipped with non-standard, implicitly defined activation functions as well as feedback circuits ([20]) in the network architecture. In this respect LIE is different from Neural Tangent Kernel (NTK, see [21]) type wide networks. Although theoretical upper bounds for classification accuracy have been derived for the NTK model in many recent literature ([22], [23], [24], [25], [26]), they are still far away from fully explaining the actual behavior of deep networks ([27]). We hope our LIE as an alternative model from NTK can provide some further insight in understanding intepretable DNNs.

The paper is organized as follows: In Section 2 we introduce the piecewise linear embedding; In Section 3 we discuss the "center of mass" calculation on Stiefel and Grassmann manifolds; In Sections 4 and 5 we introduce SIM-I and LIE; in Section 6 we provide experiments that further illustrate the theoretical framework. We conclude the paper in Section 7.

Summary of Contributions. We have proposed Subspace Indexing Model with Interpolation (SIM-I), which is a low-dimensional embedding model of high-dimensional data based on piecewise linear embedding and smooth regularization via "center of mass" calculation on Stiefel and Grassmann manifolds. Our SIM-I enjoys 3 advantages including (1) global nonlinearity; (2) computationally feasible; (3) smooth regularization. Based on SIM-I, we further build a Lightweight Inference Engine (LIE) that carries similar level of feature extraction by a given pre-trained learning model on the data manifold. We further interpret the LIE as a wide and shallow network and propose the new perspective that Deep Neural Networks (DNNs) can be transformed by our procedure to wide and shallow networks.

2. Piecewise linear Locality Preserving Projection (LPP) model

Let some image data point $x \in \mathbb{R}^D$ be represented as a vector in a very high-dimensional space. We want to find a low-dimensional embedding $y = f(x) \in \mathbb{R}^d$, $d \ll D$ such that the embedding function f retains some meaningful properties of the original image data set, ideally close to its intrinsic dimension. If we restrict ourselves to linear maps of the form $y = W^T x \in \mathbb{R}^d$, where the $D \times d$ projection matrix $W = (w_{ij})_{1 \leq i \leq D, 1 \leq j \leq d}$ (assuming full rank), then such a procedure is called a locally linear low-dimensional embedding (see [28], [29]). The target is to search for a "good" projection matrix W, such that the projection $x \mapsto y = W^T x$ must preserve certain *locality* in the data set (this is called a *Locality Preserving Projection*, or LPP projection, see [30]). The locality is interpreted as a kind of intrinsic relative geometric relations between the data points in the original high-dimensional space, usually represented by some affinity matrix $S = (s_{ij})$ (which is a symmetric matrix with non-negative terms). As an example, given unlabelled data points $x_1, ..., x_n \in \mathbb{R}^D$, we can take $s_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ when $\|x_i - x_j\| < \varepsilon$ and $s_{ij} = 0$ otherwise, where $1 \leq i, j \leq n$. Here $\sigma > 0$ and $\varepsilon > 0$ is a small threshold parameter, and $\|x_i - x_j\|$ is the Euclidean norm in $\mathbb{R}^D$. Based on the affinity matrix $S = (s_{ij})_{1 \leq i,j \leq n}$, the search for the projection matrix W can be formulated as the following optimization problem

$$\min_W \phi(W) = \frac{1}{2} \sum_{i,j=1}^{n} s_{ij} \|y_i - y_j\|^2 \ , \qquad (1)$$

in which $y_i = W^T x_i$ and $y_j = W^T x_j$ and the norm $\|y_i - y_j\|$ is taken in the projected space $\mathbb{R}^d$. Usually when $\|x_i - x_j\|$ is large, the affinity s_{ij} will be small, and vice versa. Thus (1) is seeking for the embedding matrix W such that close pairs of image points x_i and x_j will be mapped to close pairs of embeddings $y_i = W^T x_i$ and $y_j = W^T x_j$, and vice versa. This helps to preserve the local geometry of the data set, a.k.a the locality. To solve (1), we introduce a weighted fully-connected graph G where the vertex set consists of all data points $x_1, ..., x_n$ and the weight on the edge connecting x_i and x_j is given by $s_{ij} \geq 0$. Consider the diagonal matrix $\mathcal{D} = \text{diag}(\mathcal{D}_{11}, ..., \mathcal{D}_{nn})$ where $\mathcal{D}_{ii} = \sum\limits_{j=1}^{n} s_{ij}$, and we then introduce the *graph Laplacian* $L = \mathcal{D} - S$. Then the minimization problem (1), together with the normalization constraint $\sum\limits_{i=1}^{n} \mathcal{D}_{ii} y_i^2 = 1$, reduces to the following *generalized eigenvalue problem* (see [30])

$$XLX^T w = \lambda X \mathcal{D} X^T w \ , \qquad (2)$$

where $X = [x_1, ..., x_n] \in \mathbb{R}^{D \times n}$.

Assume we have obtained an increasing family of eigenvalues $0 = \lambda_0 < \lambda_1 \leq ... \leq \lambda_{n-1}$. Let the corresponding eigenvectors be $\boldsymbol{w}_0, \boldsymbol{w}_1, ..., \boldsymbol{w}_{n-1}$. Then the low-dimensional embedding matrix can be taken as $W = [\boldsymbol{w}_1, ..., \boldsymbol{w}_d]$ (see [31]). By choosing different affinity matrices $S = (s_{ij})$, the above LPP framework includes many commonly seen practical examples. For example, if the data $x_1, ..., x_n$ are not labelled, then we can take $s_{ij} = \dfrac{1}{n}$ and (2) produces the classical Principle Component Analysis (PCA). For labelled data forming subsets $\mathcal{X}_1, ..., \mathcal{X}_m$ with same labels in each subset, we can take $s_{ij} = \dfrac{1}{n_k}$ when $x_i, x_j \in \mathcal{X}_k$, and $s_{ij} = 0$ other-wise. Here n_k is the cardinality of $\mathcal{X}_k$. This will produce Linear Discriminant Analysis (LDA). The detailed justifications of these connections can be found in [31].

Given an input data set $\mathcal{X} = \{x_1, ..., x_n\}$ where each $x_i \in \mathbb{R}^D$, either labelled or unlabelled, we can apply a k-d tree ([32], [33]) based partition scheme to divide the whole data set $\mathcal{X}$ into non-overlapping subsets $C_1, ..., C_{2^h}$ where h is the depth of the tree. Conventional subspace selection algorithms could be applied on the whole sample space before the whole space is partitioned and indexed. For example, we can first apply a PCA to $\mathcal{X}$, which selects the first d bases $[a_1, ..., a_d]$ with largest variance. Based on these bases, the covariance information obtained from global PCA is utilized in the indexing as follows: (1) we project all sample points $x_1, ..., x_n$ onto the maximum variance basis a_1, then we find the median value $\mathfrak{m}_1$ of the projected samples, and split the whole collection of data along a_1 at $\mathfrak{m}_1$, i.e., split the current node into left and right children; (2) starting from level $i = 2$, for each left and right child, project the whole collection of data along the i-th maximum variance basis a_i, find the median value $\mathfrak{m}_i$, and split all the children at $\mathfrak{m}_i$; (3) increase the level from i to $i+1$ and repeat (2) until $i = h$ reaches the bottom of the tree. We collect all the 2^h children at level $i = h$ and obtain the disjoint subsets $C_1, ..., C_{2^h}$. Each subset C_k, $k = 1, 2, ..., 2^h$ consists of a family of input data in $\mathbb{R}^D$. Based on them, using the above LPP framework, for each C_k, a low-dimensional embedding matrix $W_k \in \mathbb{R}^{D \times d}$ can be constructed. In this way, over the whole data set $\mathcal{X}$, we have constructed a piecewise linear low-dimensional embedding model $f(x) : \mathbb{R}^D \to \mathbb{R}^d$ (see Figure 1(a)) where $x \in \mathcal{X}$. This model is given by the linear embedding matrices $W_1, ..., W_{2^h} \in \mathbb{R}^{D \times d}$.

The above model construction can be regarded as a training process from the data set $\mathcal{X}$. For a given test data point $x \in \mathbb{R}^D$, not included in $\mathcal{X}$, we can find the closest subset $C_{k(x)}$ to it, by selecting the index $k = k(x) \in \{1, ..., 2^h\}$ with the smallest distance $\|x - m_{k(x)}\|$. Here m_k is the mean among all data points in the subset C_k. With the subset $C_{k(x)}$ chosen, we map the test point $x \in \mathbb{R}^D$ to its low-dimensional embedding $f(x) = W_{k(x)}^T x \in \mathbb{R}^d$. Such a procedure extended the piecewise linear embedding model $f(x) : \mathbb{R}^D \to \mathbb{R}^d$ to all testing data points in $\mathbb{R}^D$.

3. Calculating the "center of mass" on Stiefel and Grassmann manifolds

Subspace Indexing ([33]) provides a d-dimensional representation of the data set $\{x_1, ..., x_n\}$ by the subspace $\text{span}(w_1, ..., w_d) = \{W^T x, x \in \mathbb{R}^D\}$ generated from the linear embedding matrix $W \in \mathbb{R}^{D \times d}$. In this case, we are only interested in the column space of W, so we can assume that $\boldsymbol{w}_0, \boldsymbol{w}_1, ..., \boldsymbol{w}_{n-1}$ is an orthonormal basis [1]. Such a matrix W belongs to the *Stiefel manifold*, defined by

Definition 1 (Stiefel manifold)**.** *The compact* Stiefel manifold $\text{St}(d, D)$ *is a submanifold of the Euclidean space* $\mathbb{R}^{D \times d}$

[1] If this is not the case, we can replace the matrix $[\boldsymbol{w}_0 \ \boldsymbol{w}_1 \ ... \ \boldsymbol{w}_{n-1}]$ by the Q matrix of the QR-decomposition of itself, without changing the corresponding subspace.

such that

$$\mathrm{St}(d, D) = \{X \in \mathbb{R}^{D\times d} : X^T X = I_d\} \ . \tag{3}$$

As an example, if we are interested in signal recovery using low-dimensional PCA embedding, the projections we calculated from PCA analysis will be on Stiefel manifolds. However, for classification tasks, the exact distance information is less important than label information. In this case, two such Stiefel matrices W_1 and W_2 produce the same embedding if $W_1 = W_2 O_d$ for some $O_d \in O(d)$, where $O(d)$ is the group of orthogonal matrices in dimension d. In this case, the relevant embedding we obtained is a point on the *Grassmann manifold*, defined by

Definition 2 (Grassmann manifold)**.** *The Grassmann manifold* $\mathrm{Gr}(d, D)$ *is defined to be the quotient manifold* $\mathrm{Gr}(d, D) = \mathrm{St}(d, D)/O(d)$*. A point on* $\mathrm{Gr}(d, D)$ *is defined by an equivalence class* $[W] = \{WO_d, O_d \in O(d)\}$ *where* $W \in \mathrm{St}(d, D)$*.*

Given a family of elements on the Stiefel or Grassmann manifold, the *center-of-mass* is defined as an element on the same manifold that minimizes the functional given by the weighted sum of square distances. To be precise, we have

Definition 3 (Stiefel and Grassmann center-of-masses)**.** *Given a sequence of matrices* $W_1, ..., W_l \in \mathrm{St}(d, D)$ *and a sequence of weights* $w_1, ..., w_l > 0$*, the* Stiefel center-of-mass *with respect to the distance* $\mathrm{d}(W_1, W_2)$ *on* $\mathrm{St}(d, D)$ *is defined as a matrix* $W_c = W_c^{\mathrm{St}}(W_1, ..., W_l; w_1, ..., w_l) \in \mathrm{St}(d, D)$ *such that*

$$\begin{aligned} W_c &= W_c^{\mathrm{St}}(W_1, ..., W_l; w_1, ..., w_l) \\ &\equiv \arg \min_{W\in \mathrm{St}(d,D)} \sum_{j=1}^{l} w_j \mathrm{d}^2(W, W_j) \ . \end{aligned} \tag{4}$$

Similarly, if the corresponding equivalent classes are $[W_1], ..., [W_l] \in \mathrm{Gr}(d, D)$*, then the* Grassmann center-of-mass *with respect to the distance* $\mathrm{d}([W_1], [W_2])$ *on* $\mathrm{Gr}(d, D)$ *is defined as the equivalence class* $[W_c]$*, where* $W_c = W_c^{\mathrm{Gr}}(W_1, ..., W_l; w_1, ..., w_l) \in \mathrm{St}(d, D)$ *is such that*

$$\begin{aligned} W_c &= W_c^{\mathrm{Gr}}(W_1, ..., W_l; w_1, ..., w_l) \\ &\equiv \arg \min_{W\in \mathrm{St}(d,D)} \sum_{j=1}^{l} w_j \mathrm{d}^2([W], [W_j]) \ . \end{aligned} \tag{5}$$

The distances $\mathrm{d}(W_1, W_2)$ or $\mathrm{d}([W_1], [W_2])$ can be taken in different ways. For example, for $W_1, W_2 \in \mathrm{St}(d, D)$, one way is to consider $\mathrm{d}(W_1, W_2) = \mathrm{d}_F(W_1, W_2) = \|W_1 - W_2\|_F$, the matrix Frobenius norm of $W_1 - W_2$. One can also take a more intrinsic distance, such as the geodesic distance between W_1 and W_2 on the manifold $\mathrm{St}(d, D)$ with the metric given by embedded geometry (see [17]). For $[W_1], [W_2] \in \mathrm{Gr}(d, D)$, one way is to consider the projected Frobenius norm $\mathrm{d}([W_1], [W_2]) = \mathrm{d}_{pF}([W_1], [W_2]) = 2^{-1/2}\|W_1 W_1^T - W_2 W_2^T\|_F$. There are also many other choices, such as using the principle angles between the subspaces, chordal norms, or other types of Frobenius norms (see [17, Section 4.3]).

With respect to matrix Frobenius norm and projected Frobenius norm, the Stiefel and Grassmann center-of-masses can be calculated explicitly in the following theorems:

Theorem 1 (Stiefel center-of-mass)**.** *We consider the singular value decomposition of the matrix* $\sum_{j=1}^{l} w_j W_j = O_1 \Delta O_2$*, where* $O_1 \in O(D)$ *and* $O_2 \in O(d)$*,* $\Delta = \begin{pmatrix} diag(\lambda_1, ..., \lambda_d)_{d\times d} \\ 0_{(D-d)\times d} \end{pmatrix}$ *and* $\lambda_1 \geq ... \geq \lambda_d \geq 0$ *are the singular values. Then the Stiefel center-of-mass with respect to the distance given by Frobenius norm* $\mathrm{d}(W_1, W_2) = \|W_1 - W_2\|_F$ *is given by* $W_c = O_1 \Lambda O_2$ *where* $\Lambda = \begin{pmatrix} diag(1, ..., 1)_{d\times d} \\ 0_{(D-d)\times d} \end{pmatrix}$*.*

Theorem 2 (Grassmann center-of-mass)**.** *Set* $\Omega_j = \left(\sum_{j=1}^{l} w_j\right)^{-1} w_j$*. We consider the singular value decomposition of the symmetric matrix* $\sum_{j=1}^{l} \Omega_j W_j W_j^T = Q\Delta Q^T$ *where* $Q \in O(D)$ *and* $\Delta = diag(\sigma_1^2, ..., \sigma_D^2)$*,* $\sigma_1^2 \geq ... \geq \sigma_D^2 \geq 0$*. Then the Grassmann center-of-mass with respect to the distance given by projected Frobenius norm* $\mathrm{d}_{pF}([W_1], [W_2]) = 2^{-1/2}\|W_1 W_1^T - W_2 W_2^T\|_F$ *is the equivalence class* $[W_c]$ *determined by* $W_c = Q\Lambda$*, where* $\Lambda = \begin{pmatrix} diag(1, ..., 1)_{d\times d} \\ 0_{(D-d)\times d} \end{pmatrix}$*.*

4. SIM-I: Interpolating the LPP model family

Recall that we have developed a piecewise linear embedding model $f(x) : \mathbb{R}^D \to \mathbb{R}^d$ over the data set $\mathcal{X} = \{x_1, ..., x_n\}$. The embedding $f(x)$ corresponds to a family of subspace indexing models $W_1, ..., W_{2^h} \in \mathrm{St}(d, D)$ (e.g. for PCA signal recovery tasks) or $[W_1], ..., [W_{2^h}] \in \mathrm{Gr}(d, D)$ (e.g. for classification tasks). Each subspace model W_k is built from LPP embedding using the subset $C_k \subset \mathcal{X}$ developed from k-d tree and h is the depth tree. Given a test point $x \in \mathbb{R}^D$ that does not lie in $\mathcal{X}$, we can map it to the low-dimensional embedding $f(x) = W_{k(x)}^T x \in \mathbb{R}^d$. The index $k(x)$ corresponds to the subset $C_{k(x)}$ that lie closest to x. In practice, we can first compute the means m_k over all the data points in the subset C_k for each $k = 1, 2, ..., 2^h$ and sort the distances $\|x - m_k\|$ in ascending order $\|x - m_{k_1(x)}\| \leq ... \leq \|x - m_{k_{2^h}(x)}\|$, $\{k_1(x), ..., k_{2^h}(x)\} = \{1, ..., 2^h\}$. We then take $k(x) = k_1(x)$ to be the index k corresponding to the shortest distance. This is effective when the test point x lies significantly close to one of the subsets $C_{k(x)}$, see Figure 3(a).

However, for a general test point $x \in \mathbb{R}^D$, it might happen that this point lies at approximately the same distances to the centers of each of the several different subsets adjacent to x (see Figure 3(b)). In this case, we aim to *interpolate* between several subspace indexing models $W_{j_1(x)}, ..., W_{j_I(x)}$. To do this, we first find the subspace indexes $j_1(x), ..., j_I(x)$ from the first I subsets $C_{j_1(x)}, ..., C_{j_I(x)}$ closest to x, i.e., $j_1(x) = k_1(x), ..., j_I(x) = k_I(x)$ given the sorted dis-

Algorithm 1 SIM-I: Subspace Indexing Model with Interpolation

```
Input: Data set $\mathcal{X} = \{x_1, ..., x_n \in \mathbb{R}^D\}$ and its corresponding affinity matrix $S = (s_{ij})_{1 \leq i,j \leq n}$; test point $x \in \mathbb{R}^D$; threshold ratio $r_{\text{thr}} > 1$; tree depth $h$; parameter $K > 0$
Using an initial PCA and a k-d tree based partition scheme, decompose the data set $\mathcal{X}$ into subsets $C_1, ..., C_{2^h}$, where $h$ is the depth of the tree
For each subset $C_k$, calculate its mean (center) $m_k \in \mathbb{R}^D$ and its LPP embedding matrix $W_k \in \text{St}(d, D)$ based on the affinity matrix $S$
Sort the distances $\|x - m_k\|$ in ascending order $\|x - m_{k_1(x)}\| \leq ... \leq \|x - m_{k_{2^h}(x)}\|$, $\{k_1(x), ..., k_{2^h}(x)\} = \{1, ..., 2^h\}$
Determine $I$, which is the first sub-index $i$ of $k_i(x)$ such that $\|x - m_{k_{I+1}(x)}\| > r_{\text{thr}} \|x - m_{k_1(x)}\|$
Set $j_i(x) = k_i(x)$ for $i = 1, 2, ..., I$ and obtain the embedding matrices $W_{j_1(x)}, ..., W_{j_I(x)} \in \text{St}(d, D)$ or their corresponding subspaces $[W_{j_1(x)}], ..., [W_{j_I(x)}] \in \text{Gr}(d, D)$, together with the choice of weights $w_i \equiv 1$ or $w_i = \exp(-K\|x - m_{j_i(x)}\|^2) > 0$ for $i = 1, ..., I$
Find a center-of-mass $W_c = W_c^{\text{St}}(W_{j_1(x)}, ..., W_{j_I(x)}; w_1, ..., w_I)$ (Stiefel case) or $[W_c] = [W_c^{\text{Gr}}(W_{j_1(x)}, ..., W_{j_I(x)}; w_1, ..., w_I)]$ (Grassmann case) according to Definition 3 and Theorems 1 and 2.
Output: The low-dimensional embedding $f(x) = W_c^T x \in \mathbb{R}^d$
```

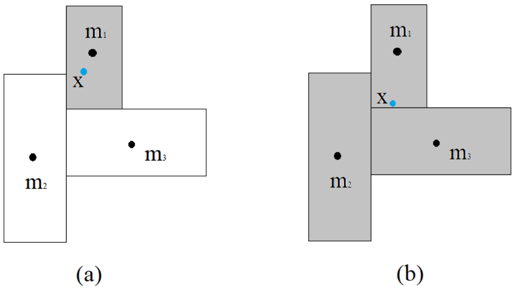

Figure 3: There are $I = 3$ nearby subsets C_1, C_2, C_3 with means m_1, m_2, m_3 in the training set. (a) Test point x is apparently close to m_1, and thus the low-dimensional embedding $f(x)$ is taken as $f(x) = W_1^T x$, where W_1 is the LPP subspace based on C_1; (b) Test point x has approximately the same distances to m_1, m_2, m_3, and thus the embedding $f(x)$ is taken as $f(x) = W_c^T x$, where W_c is the Stiefel/Grassmann center-of-mass for the LPP subspace models W_1, W_2, W_3 based on C_1, C_2, C_3.

tances $\|x - m_k\|$ mentioned above. In practice, the number $I = I(x)$ is depending on x and can be chosen in the following way: I is the first sub-index i of $k_i(x)$ such that $\|x - m_{k_{I+1}(x)}\| > r_{\text{thr}} \|x - m_{k_1(x)}\|$, where $r_{\text{thr}} > 1$ is a threshold ratio that can be tuned. We then pick the weights as $w_i \equiv 1$ or $w_i = \exp(-K\|x - m_{j_i(x)}\|^2)$ for some $K > 0$ and $i = 1, 2, ..., I$. The first choice of the weights w_i is just a standard arithmetic average while the second is indicating that the closer x is to $C_{j_i(x)}$, the heavier weights we assign to $W_{j_i(x)}$ in the interpolation process. Given the embedding matrices $W_{j_1(x)}, ..., W_{j_I(x)} \in \text{St}(d, D)$ or their corresponding subspaces $[W_{j_1(x)}], ..., [W_{j_I(x)}] \in \text{Gr}(d, D)$, together with the weights $w_1, ..., w_I > 0$, we find a center-of-mass $W_c = W_c^{\text{St}}(W_{j_1(x)}, ..., W_{j_I(x)}; w_1, ..., w_I)$ (Stiefel case) or $[W_c] = [W_c^{\text{Gr}}(W_{j_1(x)}, ..., W_{j_I(x)}; w_1, ..., w_I)]$ (Grassmann case) according to Definition 3 and Theorems 1 and 2. Finally, we map the test point x to the low-dimensional embedding $f(x) = W_c^T x \in \mathbb{R}^d$. Notice that when $I = 1$, the interpolation procedure reduces to projecting x using $W_{k_1(x)}$ calculated from LPP analysis on the closest subset only. In general, the whole interpolation procedure can be regarded as providing a regularized version of the piecewise linear embedding we discussed in Section 2 (also see Figure 1). We summarize our interpolation method as the SIM-I Algorithm 1.

5. Building a Lightweight Inference Engine (LIE) from SIM-I

Here we demonstrate how we can use our SIM-I construction to build a Lightweight Inference Engine (LIE). Let us assume that $M \subset \mathbb{R}^D$ is the high-dimensional data manifold from which we sample many training samples $x_1, ..., x_n \in M$. We apply SIM-I on $\{x_1, ..., x_n\}$ and map each x_i into a low-dimensional embedding $x_i' \in M' \subset \mathbb{R}^d$, $1 \leq i \leq n$. Assume some pre-trained learning model, such as Deep Neural Networks (DNNs), is given to label the training data samples from M: $y_i = y(x_i), i = 1, 2, ..., n$. For another situation that can be included here, we can assume that the labels y_i of the training data x_i are the natural labels originally given. With the training samples and their labels given, i.e., $\mathcal{T} = \{(x_1, y_1), ..., (x_n, y_n)\}$, the SIM-I embedding gives us a low-dimensional labelled dataset $\mathcal{T}' = \{(x_1', y_1), ..., (x_n', y_n)\}$. For a test data x on M, we use our obtained SIM-I model to map it to some $x' \in M'$. We then perform k-Nearest-Neighbor (kNN) classification (see Figure 2) for x' with respect to $\mathcal{T}'$ in the low-dimensional Euclidean space $\mathbb{R}^d$ and we obtain the classification label y. Viewing the whole procedure as a direct mapping from the high-dimensional data x to the classification label y, we obtain an inference engine. When the original labels are from a pre-trained learning model, our inference engine so obtained carries similar level of feature extraction as this pre-trained model. When the original labels for the training data are natural, our inference engine can be regarded as a new type of learning model and we can compare its accuracy with some known models such as DNNs. Moreover, as SIM-I is a low-dimensional embedding that does not require many hierarchies of function compositions (like in DNNs), our inference engine is light weighted. We thus name this as a Lightweight Inference Engine (LIE, see Figure 2).

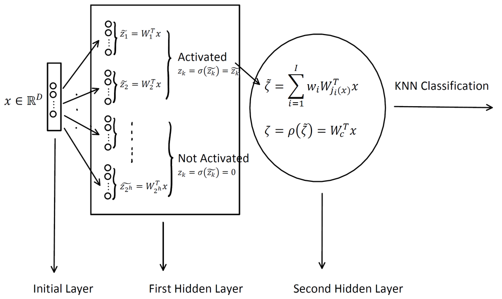
$x \in \mathbb{R}^D$
$\widetilde{z}_1 = W_1^T x$
$\widetilde{z}_2 = W_2^T x$
$\widetilde{z_{2^h}} = W_{2^h}^T x$
Activated
$z_k = \sigma(\widetilde{z_k}) = \widetilde{z_k}$
Not Activated
$z_k = \sigma(\widetilde{z_k}) = 0$
$\tilde{\zeta} = \sum_{i=1}^{I} w_i W_{j_i(x)}^T x$
$\zeta = \rho(\tilde{\zeta}) = W_c^T x$
KNN Classification
Label y
Initial Layer
First Hidden Layer
Second Hidden Layer

Figure 4: LIE build from SIM-I interpreted as a shallow network with a wide first hidden layer (using the Stiefel center-of-mass case).

It is worth mentioning here that at a high level, using Theorems 1 and 2, our SIM-I based LIE can be interpreted as a shallow neural network with very wide first hidden layer activated by ReLU-like nonlinearity (Figure 4). Given the input data $x \in \mathbb{R}^D$, and assume that SIM-I at the first step builds 2^h piecewise linear LPP models $W_1, ..., W_{2^h}$, where h is the k-d tree height and each LPP matrix W_i is a $D \times d$ projection matrix, $i = 1, ..., 2^h$. Then the first layer of the neural network maps x into $W_1^T x, ..., W_{2^h}^T x$, each of which is a vector in $\mathbb{R}^d$. This can be regarded as a very wide first hidden layer (Figure 4) with $d \times 2^h$ neurons and the linear pre-output mapping $x \mapsto \widetilde{z} = (\widetilde{z}_1, ..., \widetilde{z}_{2^h}) = (W_1^T x, ..., W_{2^h}^T x)$. At the next step, recalling that in Step 6 of the SIM-I Algorithm 1 we picked only several embedding matrices $W_{j_1(x)}, ..., W_{j_I(x)}$, we see that the nonlinear activation function at the first hidden layer can be regarded as some ReLU-like activation, i.e.,

$$z_i = \sigma(\widetilde{z}_i) = \begin{cases} \widetilde{z}_i & \text{if } i \in \{j_1(x), ..., j_I(x)\} \ , \\ 0 & \text{otherwise } \ . \end{cases}$$

The mapping $x \mapsto z = (z_1, ..., z_{2^h})$ is thus the neural network output mapping of the first hidden layer.

A crucial step in our SIM-I construction is to calculate the center-of-masses for $W_{j_1(x)}, ..., W_{j_I(x)}$ on either the Stiefel or Grassmann manifolds, i.e. Step 7 in Algorithm 1. In Theorems 1 and 2 we provided explicit formulas for calculating these center-of-masses. Based on these formulas, we can build the second hidden layer of the neural network. We take the Stiefel center-of-mass as an example (Figure 4). Here we assume that the weights $w_1, ..., w_I > 0$ have been a-priori assigned, say $w_1 = ... = w_I = 1$ [2]. Given the output $z = (z_1, ..., z_{2^h})$ from the first hidden layer, it is easy to see from our construction of the first hidden layer that the linear combination $\sum_{k=1}^{2^h} \omega_k z_k = \sum_{i=1}^{I} w_i W_{j_i(x)}^T x$ for some numerical sequence $\{\omega_k\}_{k=1}^{2^h}$, such that $\omega_k = w_i$ when $k = j_i(x)$ and $\omega_k = 0$ otherwise. The pre-output of the second hidden network layer is thus defined by the linear map $z \mapsto \widetilde{\zeta} = \sum_{k=1}^{2^h} \omega_k z_k$. Recall that $\sum_{i=1}^{I} w_i W_{j_i(x)} = O_1 \Delta O_2$ is the singular value decomposition as in Theorem 1. Moreover, by Theorem 1, the embedding of x via the Stiefel center-of-mass can be represented as $\zeta = W_c^T x = O_2^T \Lambda O_1^T x$. Here Δ and Λ are defined as in Theorem 1. We regard ζ as the output from the second hidden layer. By this, the nonlinear mapping $\widetilde{\zeta} \mapsto \zeta = \rho(\widetilde{\zeta})$ can be regarded as the activation function of the second hidden layer. Actually, $\widetilde{\zeta} = \sum_{i=1}^{I} w_i W_{j_i(x)}^T x = O_2^T \Delta O_1^T x$ and $\zeta = O_2^T \Lambda O_1^T x$, so the nonlinear mapping is just $\rho(O_2^T \Delta O_1^T x) = O_2^T \Lambda O_1^T x$.

We have obtained the output $\zeta \in \mathbb{R}^d$ of the second hidden layer, which is just the SIM-I embedding of $x \in \mathbb{R}^D$, i.e., $\zeta = x'$. The predicted label of x is obtained by doing a nearest-neighbor classification based on x'. This can also be achieved using a 3-layer neural network with feedback circuits ([20]). Thus we have realized our LIE built from SIM-I as a shallow neural network with a wide first hidden layer (see Figure 4). From this perspective, when the pre-trained learning model is a DNN, SIM-I provides a way to exchange deep network for wide but shallow ones. However, such a shallow network is different from the usual feed-forward networks with explicit activation functions. Rather, it is equipped with non-standard, implicitly defined activation functions as well as feedback circuits. In this respect LIE is different from Neural Tangent Kernel (NTK, see [21]) type wide networks, as we have discussed in the Introductory Section 1. We hope our LIE as an alternative model from NTK can provide some further insight in understanding intepretable DNNs.

6. Experiments

6.1. PCA Recovery for SIFT data set

SIFT (Scale Invariant Feature Transform, see [34]) data set is a data set that computes for each keypoint a real-valued descriptor, based on the content of the surrounding patch in terms of local intensity gradients. Given its remarkable performance, SIFT has been often used as starting point for the creation of other descriptors. The final SIFT descriptor consists of 128 elements. This means that each data point in SIFT data set has dimension $D = 128$, and we pick the embedding dimension $d = 16$. The original SIFT data set has 10068850 data samples, that form the data set `sift_sample`. We randomly collect $n_{\text{train}} = 200 \times 2^{13}$ elements from these sample points as our target data set $\mathcal{X} = \texttt{sift\_train} = \{x_1, ..., x_{n_{\text{train}}} \in \mathbb{R}^D\}$. We consider the recovery efficiency of PCA embedding of the SIFT data set. Let x be a point in `sift_sample` and let W be a Stiefel matrix in $\text{St}(16, 128)$. The projection of x onto the

[2] Notice that in our second choice of the weights from Step 6 of Algorithm 1, i.e., $w_i = \exp(-K\|x - m_{j_i(x)}\|^2)$ for some $K > 0$ and $i = 1, 2, ..., I$, we calculate the weights from the input x. The latter case can be regarded as a further modification of the constant weight case, but it will cause nonlinearity.

16-dimensional space is then denoted by $y = W^T x$. By recovery we meant to consider the point $\widehat{x} = (W^T)^- y$ where $(W^T)^-$ is the Moore-Penrose pseudo-inverse of W^T. The recovery efficiency can then be quantified by the recovery error $\|x - \widehat{x}\|$, where the Euclidean norm is computed in $\mathbb{R}^{128}$.

We pick $h = 13$ so that $2^h = 8192$. Then we decompose `sift_train` into 8192 subsets $C_1, ..., C_{8192}$ using k-d tree based partition. For each subset C_k, we calculate the mean $m_k \in \mathbb{R}^{128}$ and we obtain a PCA embedding matrix $W_k \in \mathrm{St}(16, 128)$. We sort the distances $\|x - m_k\|$, $k = 1, 2, ..., 8192$ in ascending order so that $\|x - m_{k_1}\| \leq ... \leq \|x - m_{k_{8192}}\|$, where $\{k_1, ..., k_{8192}\} = \{1, ..., 8192\}$. Then we find among the subset means m_k, $k = 1, 2, ..., 8192$ the first I nearest to x, with their indexes denoted by $k_i = k_i(x), i = 1, .., I$. The number $I = I(x)$ is depending on x and is chosen in the following way: I is the first subindex i of k_i such that $\|x - m_{k_{I+1}}\| > r_{\text{thr}} \|x - m_{k_1}\|$, where $r_{\text{thr}} > 1$ is a threshold ratio. We pick $r_{\text{thr}} = 2$.

For test data set, we randomly pick from `sift_sample\sift_train` a subset of size $n_{\text{test}} = 500$, and we denote the data set as `sift_test`. For each test point $x \in$ `sift_test`, we consider sending it to the nearest subset C_{k_1} and the corresponding Stiefel matrix is $W_{k_1} \in \mathrm{St}(16, 128)$. We can then consider the recovery point $\widehat{x} = (W_{k_1}^T)^- W_{k_1}^T x$ and the benchmark recovery error Error_bm $= \|x - \widehat{x}\|$.

Consider the alternative recovery scheme using our method SIM-I. We calculate the weights $w_j = \exp(-K\|x - m_{k_j}\|^2)$ for $j = 1, ..., I$ and we choose the constant $K = 10^{-8}$. We then find the Stiefel center-of-mass $W_c = W_c^{\mathrm{St}}(W_{k_1(x)}, ..., W_{k_I(x)}; w_1, ..., w_I)$ using Theorem 1 and taking the distance function to be the matrix Frobenius norm $\mathrm{d}(W_1, W_2) = \|W_1 - W_2\|_F$. We consider the recovery point $\widehat{x}_c = ((W_c)^T)^-(W_c)^T x$ and the recovery error Error_c $= \|x - \widehat{x}_c\|$.

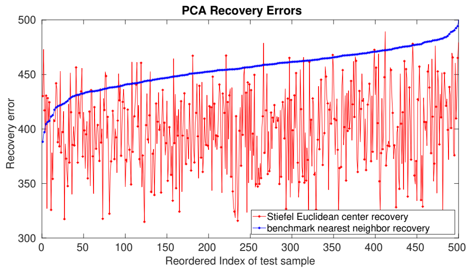

Figure 5: Comparison of the PCA Recovery Errors: Blue = benchmark case using closest subset PCA recovery, with the error sorted from low to high; Red = using SIM-I based on Stiefel center-of-mass and $\mathrm{d}(W_1, W_2) = \|W_1 - W_2\|_F$.

Over the test set `sift_test`, we find that for about 93% test points, Error_c < Error_bm, which implies that SIM-I improved the efficiency of recovery. The empirical average Error_c is 399.786223, and the empirical average Error_bm is 455.537462. Figure 5 plots the Error_c and Error_bm (vertical axis) as functions of the test sample indexes. The red curve is for Error_c and blue curve is for Error_bm, where we have sorted Error_bm in ascending order and reordered the test indexes correspondingly. Figure 6 gives the differences Error_c − Error_bm in descending order. It can be apparently seen that Error_c − Error_bm < 0 for most of test samples.

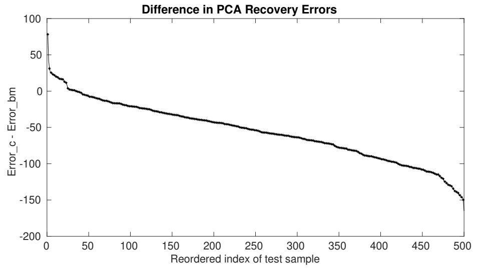

Figure 6: Differences Error_c − Error_bm in descending order.

6.2. Nearest-neighbor classification with SIM-I embedding and building the corresponding LIE on MNIST dataset

Here we consider a labeled data set `data` $= \{(x_i, y_i), i = 1, 2, ..., N\}$ where $x_i \in \mathbb{R}^D$ are the inputs and $y_i \in \mathbb{N}$ are the labels. We randomly select the training set `data_train` $= \{(x_i, Y_i), i = 1, 2, ..., n_{\text{train}}\}$ and the test set `data_test` $= \{(a_i, b_i), i = 1, 2, ..., n_{\text{test}}\}$, and we make them disjoint. We first project the training set onto a `kd_PCA`-dimensional subspace via a standard PCA. Based on this initial embedding, using a kd-tree with height h, we divide `data_train` into 2^h clusters C_k, $k = 1, 2, ..., 2^h$. For each C_k, we find an LPP embedding matrix $W_k \in \mathrm{St}(\texttt{kd\_LPP}, D)$ by setting the affinity matrix to be $s_{ij} = \exp(-\|x_i - x_j\|^2)$ if $x_i, x_j \in C_k$ are in the same class and $s_{ij} = 0$ otherwise. Since we are having a classification problem, we can identify each W_k by the subspace it spans, i.e., we consider $[W_k]$ which is the equivalence class of W_k in $\mathrm{Gr}(\texttt{kd\_LPP}, D)$.

As before, we compute in $\mathbb{R}^D$ the means m_k of each cluster C_k. For a test point $x \in$ `data_test`, we sort the distances $\|x - m_k\|$, $k = 1, 2, ..., 2^h$ in ascending order so that $\|x - m_{k_1}\| \leq ... \leq \|x - m_{k_{2^h}}\|$, where $\{k_1, ..., k_{2^h}\} = \{1, ..., 2^h\}$. Then we find among the cluster means m_k, $k = 1, 2, ..., 2^h$ the first I nearest to x, with their indexes denoted by $k_i = k_i(x), i = 1, ..., I$. The number $I = I(x)$ is depending on x and is chosen in the following way: I is the first sub-index i of k_i such that $\|x - m_{k_{I+1}}\| > r_{\text{thr}} \|x - m_{k_1}\|$, where $r_{\text{thr}} > 1$ is a threshold ratio. The parameter r_{thr} can be treated as a hyper-parameter that we can tune here.

For baseline method, we do a nearest-neighbor classification for x on a low-dimensional embedding of the cluster C_{k_1}, and we pick the number of nearest-neighbors to be `knn` ≥ 1. Indeed we project x and all training data points in C_{k_1} using W_{k_1}, and perform nearest-neighbor classification on the resulting projection.

For our method SIM-I, we take the union $C_{k_1} \cup ... \cup C_{k_I}$. Recall that each C_{k_i} corresponds to an LPP embedding projection matrix W_{k_i}. We set the weights $w_i = \exp(-K\|x - m_{k_i}\|^2)$ for a $K > 0$ to be determined. We compute a center-of-mass W_c of the projection matrices W_{k_i} with weights w_i, $i = 1, 2, ..., I$ using the Grassmann center-of-mass method, where the distance is taken as the projected Frobenius norm of Grassmann matrices, i.e., $\mathrm{d}(W_1, W_2) = 2^{-1/2}\|W_1W_1^T - W_2W_2^T\|_F$ and $\| \bullet \|_F$ is the matrix Frobenius norm. We obtain $W_c = W_c^{\mathrm{Gr}}(W_{k_1(x)}, ..., W_{k_I(x)}; w_1, ..., w_I)$ and we project x and all training data points in the union $C_{k_1} \cup ... \cup C_{k_I}$ using W_c. We then perform a nearest-neighbor classification for x on this low-dimensional embedding with the number of nearest-neighbors being equal to $\texttt{knn}$.

We first experiment on the original MNIST data set with its original labels in the training samples, so $N = 70000$. We pick $n_{\text{train}} = 60000, n_{\text{test}} = 10000$ and the original dimension $D = 784$. We pick the embedding dimensions $\texttt{kd\_PCA} = 128$, $\texttt{kd\_LPP} = 100$ and we take $K = 10^{-8}$. The k-d tree height is taken to be 8 and $r_{\text{thr}} = 1.2$. We performed two experiments (a) take $\texttt{knn} = 1$: for baseline method, the classification rate is 93.58%, for SIM-I, the classification rate is 95.55%; (b) take $\texttt{knn} = 75$, for baseline method, the classification rate is 87.52%, for SIM-I, the classification rate is 94.21%.

We then experiment on the MNIST dataset with the labels of the training samples given by a pre-trained LeNet-5 model (slighly modified, see https://github.com/guptajay/Kaggle-Digit-Recognizer). We try to build an LIE from this model. As in the previous experiment, our $N = 70000$. We first pick $n_{\text{train}} = 60000$, $n_{\text{test}} = 10000$, and the original dimension $D = 784$. We pick the embedding dimensions $\texttt{kd\_PCA} = 512$ and $\texttt{kd\_LPP} = 256$. For each training data we take its x component and label it using the LeNet-5 model. We build a k-d tree of depth 8 from decomposing the training data set. Since $256^2 > 50000$, we do not have enough points in each k-d tree cluster to build the LPP embedding model. To remedy this, we build a GMM model with 10 components at each of the $2^8 = 256$ clusters and use this GMM model to generate 500 new training data points for each cluster. We label these newly augmented data points by the pre-trained LeNet-5 model. For each of the clusters in the k-d tree decomposition, we then have enough points to build the LPP model and proceed with our SIM-I construction. As we described in Section 5, working with this augmented training data set, our SIM-I procedure leads to an LIE that can classify each of the test data (in the nearest-neighbor classification step we use $\texttt{knn} = 1$). We use the same baseline method as in the previous experiment to compare the classification rates. Our experiment shows that the baseline method has a classification rate of 88.76% and our SIM-I based LIE has a classification rate of 96.14%. While the classification rate of LeNet-5 at the original 784 dimensions is given by 99.79%, we can see that our SIM-I based LIE has an approximate rate though our nearest-neighbor classification happens at an only 128 dimensional LPP embedding. On the other hand, since we first projected the training dataset onto 512 dimensional space using a PCA, to play a fair comparison, we actually should use a pseudo-inverse to map the 512-dimensional projection of each test data back to dimension 784 and we then use LeNet-5 to classifiy them. This leads to a classification rate of 85.06%, which is lower than our SIM-I based LIE.

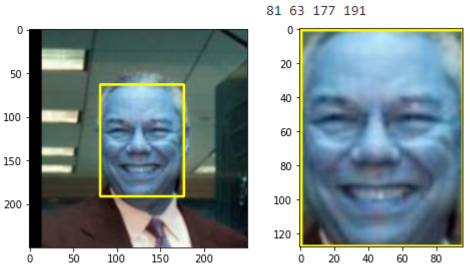

Figure 7: An example with calculated face coordinates from LFW image set.

6.3. Nearest-neighbor classification with SIM-I embedding and building the coreesponding LIE on the PCA embedded face images dataset

We carried another experiment on a large-size image set with over 400K face images with 900 subjects for this test. Image set is downloaded using the url page at [35]. Face bounding box parameters for each subject is given. Even though the original images were focused on faces with some extra area, images were further cropped to just the faces of the subjects. Face detection DNN libraries from OpenCV ([36]) were used to achieve this. The OpenCV DNN program provides the coordinates around the face that are used to crop just the faces out of images. Figure 7 shows an example with calculated face coordinates from LFW (Labelled Faces in the Wild) image set.

The face cropped images were resized into $32 \times 32 \times 3$ size to help reduce the dimensionality as well as standardize the size of images. All of these resized images' raw pixels were appended into an array.

Considering it's a huge dataset and takes a lot of computer resources to run classification algorithms, we decided to reduce dimensionality of the data through PCA. PCA was applied to obtain projection of data on to a lower dimension size. Before applying PCA, data was standardized using standard scalar in python. In first test, PCA dimensions were reduced to 256 dimensions and for the second test PCA dimensions were set at 128.

Reducing the dimensionality to 256 dimensions had an explained variance of 95.4%. Figure 8 shows a graph of explained variance with principal components for 256 components.

Reducing the dimensionality to 128 dimensions had an explained variance of 91%. Figure 9 shows a graph of explained variance with principal components for 128 components.

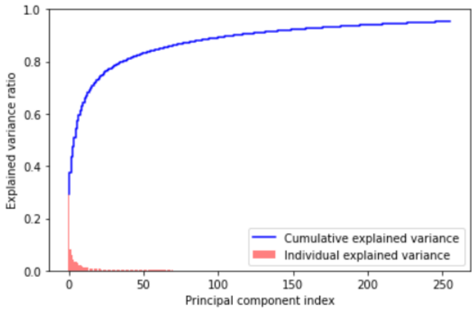

Figure 8: The Explained Variance of 256 dimensional PCA embedding for face dataset.

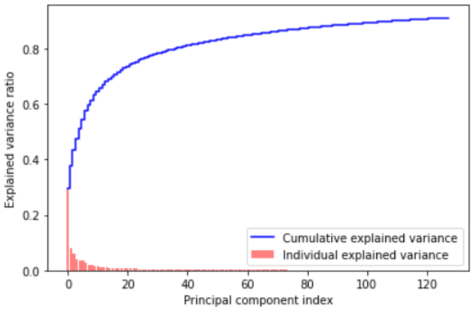

Figure 9: The Explained Variance of 128 dimensional PCA embedding for face dataset.

Data was split into 90%-10% for training and test, PCA method was fit on the training data and then obtained projection is used to transform the training and test dataset into smaller dimensionality. It goes from $400,000 \times 3072$ to $400,000 \times 256$ or $400,000 \times 128$.

A fully connected classifier neural network model is built with a couple of layers with a final softmax output of 900 units. We ran the training for 300 epochs and obtained validation accuracy of 97% with 256 PCA dimension size. With a much lower 128 dimension size, we still obtained validation accuracy of 96.5% showing that a highly accurate classification model. There're some caveats to the highly accurate results, this data is very clean and balanced among subjects and is large in size.

To compare SIM-I with the above neural network, we apply SIM-I to the 256 dimensional embedding dataset produced by PCA embedding. We choose to have the LPP embedding dimension equals to 128. We pick the k-d tree height to be equal to 8 and we apply the SIM-I construction. After SIM-I embedding, each 256 dimensional input vector is embedded into 128 dimensional space and we apply 1-nearest-neighbor classification to produce LIE. Our so-obtained LIE gives us a classification rate of 96.3% using the Grassmann center-of-mass method, compared to a rate of 81.2% using the nearest cluster projection method. This indicates that SIM-I based LIE achieves very similar level of high classification accuracy as neural network models. At the same time, it also shows that using Grassmann center-of-mass method significantly improves the classification accuracy compared with using the vanilla nearest cluster projection method.

We also apply SIM-I to the 128 dimensional embedding dataset produced by PCA embedding. We choose to have the LPP embedding dimension equals to 64. We pick the k-d tree height to be equal to 8 and we apply the SIM-I construction. After SIM-I embedding, each 128 dimensional input vector is embedded into 64 dimensional space and we apply 1-nearest-neighbor classification to produce LIE. Our so-obtained LIE gives us a classification rate of 95% using the Grassmann center-of-mass method, compared to a rate of 83.4% using the nearest cluster projection method.

7. Conclusion

Subspace Indexing Model with Interpolation (SIM-I) is proposed in this work using LPP embedding and "center-of-mass" calculations on Stiefel and Grassmann manifolds. Based on SIM-I and given a pre-trained learning model, we build a Lightweight Inference Engine (LIE). At a high level, we interpret the LIE as a wide and shallow network. Using this interpretation, we propose the new perspective that Deep Neural Networks (DNNs) can be transformed to wide and shallow networks. We hope this transformation can provide more insights on the interpretability of DNNs in the near future.

References

[1] T. Zhou, D. Tao, and X. Wu, "Manifold elastic net: A unified frame-work for sparse dimension reduction," *Data Min. Knowl. Disc.*, vol. 22, no. 3, pp. 340–371, 2010.

[2] W. Bian and D. Tao, "Max-min distance analysis by using sequential SDP relaxation for dimension reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 1037–1050, 2011.

[3] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 7, pp. 929–942, 2010.

[4] T. Zhang, D. Tao, X. Li, and J. Yang, "Patch alignment for dimensionality reduction," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1299–1313, 2009.

[5] Y. Fu and T.-S. Huang, "Image classification using correlation tensor analysis," *IEEE Trans. Image Process.*, vol. 17, no. 2, pp. 226–234, 2008.

[6] D. Tao, X. Li, X. Wu, and S. J. Maybank, "General tensor discriminant analysis and gabor features for gait recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1700–1715, 2007.

[7] I. T. Jolliffe, "Principal Component Analysis, 2nd ed. new york," 2002.

[8] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, 1997.

[9] D. Tao, X. Li, X. Wu, and S. J. Maybank, "Geometric mean for subspace selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 260–274, 2009.

[10] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," *J. Mach. Learn. Res.*, vol. 4, pp. 119–155, 2003.

[11] V. Strassen, "Gaussian elimination is not optimal," *Numer Math.*, vol. 13, pp. 54–356, 1969.

[12] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," *presented at the Int. Conf. Mach. Learning*, 2004.

[13] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 1, pp. 1–48, 2006.

[14] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Manifold regularized discriminative non-negative matrix factorization with fast gradient descent," *IEEE Trans. Image Process*, 2011.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 2016.

[16] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, 2017.

[17] A. Edleman, T. Arias, and S. Smith, "The Geometry of Algorithms with Othogonality Constraints," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 2, 1999.

[18] T. Kaneko, S. Fiori, and T. Tanaka, "Empirical Arithmetic Averaging Over the Compact Stiefel Manifold," *IEEE Transactions on Signal Processing*, vol. 61, no. 4, 2013.

[19] T. Marrinan, J. R. Beveridge, B. Draper, and M. Kirby, "Finding the Subspace Mean or Median to Fit Your Need," *CVPR*, 2014.

[20] Y. Chen, R. Damper, and M. Nixon, "On Neural-Network Implementations of Nearest Neighbor Pattern Classifiers," *IEEE Transactions on Circuits and Systems I Fundamental Theory and Applications*, 1997.

[21] A. Jacot, F. Gabriel, and C. Hongler, "Neural Tangent Kernel: Convergence and Generalization in Neural Networks," *NIPS*, 2018.

[22] S. Arora, S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," *International Conference on Machine Learning (ICML)*, 2019.

[23] Z. Chen, Y. Cao, D. Zou, and Q. Gu, "How much over-parametrization is sufficient to learn deep relu networks?" *arXiv:1911.12360*, 2019.

[24] X. Zhang, Y. Yu, L. Wang, and Q. Gu, "Learning one-hidden-layer relu networks via gradient descent," *Proc of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS), Naha, Okinawa, Japan*, 2019.

[25] Y. Cao and Q. Gu, "Generalization bounds of stochastic gradient descent for wide and deep neural networks," *Proc. of Advances in Neural Information Processing Systems (NeurIPS), Vancouver, Canada. Spotlight presentation*, vol. 32, 2019.

[26] G. Yang, "Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent derivation," *arXiv:1904.00687*, 2019.

[27] M. Seleznova and G. Kutyniok, "Analyzing Finite Neural Networks: Can We Trust Neural Tangent Kernel Theory?" *arXiv:2012.04477[cs.LG]*, 2020.

[28] S. Roweis and L. Saulm, "Nonlinear dimensionality reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[29] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *Tilburg University Technical Report, TiCC-TR 2009-005*, 2009.

[30] X. He and P. Niyogi, "Locality Preserving Projections," *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[31] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Hong-Jiang Zhang, "Face Recognition Using Laplacianfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, 2005.

[32] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[33] X. Wang, Z. Li, and D. Tao, "Subspace Indexing on Grassmann Manifold for Large Scale Multimedia Retrieval," *IEEE Trans on Image Processing*, vol. 20, no. 9, pp. 2627–2635, 2011.

[34] D. G. Lowe, "Distinctive image features from scale-invariant key points," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[35] O. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," *British Machine Vision Conference, available at https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/*, 2015.

[36] "OpenCV," *https://opencv.org/*.