

A APPENDIX

A.1 PROOF OF THEOREM 1

Let $L \in \mathbb{R}^{N \times N}$ denote the Laplacian matrix of an undirected graph, and u_i denote the i -th eigenvector of L corresponding to the i -th eigenvalue λ_i that satisfies:

$$Lu_i = \lambda_i u_i, \quad (15)$$

then we have the following eigenvalue perturbation analysis:

$$(L + \delta L)(u_i + \delta u_i) = (\lambda_i + \delta \lambda_i)(u_i + \delta u_i), \quad (16)$$

where a perturbation $\delta L = \delta w_{p,q} e_{p,q} e_{p,q}^\top$ that implies a new edge connection is applied to L , resulting in perturbed eigenvalues and eigenvectors $\lambda_i + \delta \lambda_i$ and $u_i + \delta u_i$ for $i = 1, \dots, N$, respectively.

Keeping only the first-order terms leads to:

$$L\delta u_i + \delta L u_i = \lambda_i \delta u_i + \delta \lambda_i u_i. \quad (17)$$

Write δu_i in terms of the original eigenvectors u_i for $i = 1, \dots, N$:

$$\delta u_i = \sum_{i=1}^N \alpha_i u_i. \quad (18)$$

Substituting (18) into (17) leads to:

$$L \sum_{i=1}^N \alpha_i u_i + \delta L u_i = \lambda_i \sum_{i=1}^N \alpha_i u_i + \delta \lambda_i u_i. \quad (19)$$

Multiplying u_i^\top to both sides of (19) results in:

$$u_i^\top L \sum_{i=1}^N \alpha_i u_i + u_i^\top \delta L u_i = \lambda_i u_i^\top \sum_{i=1}^N \alpha_i u_i + \delta \lambda_i u_i^\top u_i. \quad (20)$$

Since u_i for $i = 1, \dots, N$ are unit-length, mutually-orthogonal eigenvectors, we have:

$$u_i^\top L \sum_{i=1}^N \alpha_i u_i = \alpha_i u_i^\top L u_i, \quad \lambda_i u_i^\top \sum_{i=1}^N \alpha_i u_i = \alpha_i u_i^\top \lambda_i u_i. \quad (21)$$

Substituting (15) into (21), we have:

$$\alpha_i u_i^\top L u_i = \alpha_i u_i^\top \lambda_i u_i. \quad (22)$$

According to (21), we have:

$$u_i^\top \sum_{i=1}^N \alpha_i u_i = \lambda_i u_i^\top \sum_{i=1}^N \alpha_i u_i. \quad (23)$$

Substituting (23) into (20) leads to:

$$u_i^\top \delta L u_i = \delta \lambda_i u_i^\top u_i = \delta \lambda_i. \quad (24)$$

Then the eigenvalue perturbation due to δL is given by:

$$\delta \lambda_i = \delta w_{p,q} (u_i^\top e_{p,q})^2. \quad (25)$$

Assume each edge weight is computed by $w_{p,q} = \frac{1}{z_{p,q}^{data}}$ where $z_{p,q}^{data} = \|X^\top e_{p,q}\|_2^2$. Since $(u_i^\top e_{p,q})^2 \propto z_{p,q}^{emb} = \|U_N^\top e_{p,q}\|_2^2 \approx \|U_r^\top e_{p,q}\|_2^2$, as long as we can find an edge with large $w_{p,q} (u_i^\top e_{p,q})^2$ or $\eta_{p,q} = \frac{M z_{p,q}^{emb}}{z_{p,q}^{data}}$, including this edge into the current graph will significantly perturb the Laplacian eigenvalue λ_i and eigenvector u_i . When $\sigma^2 \rightarrow +\infty$, the total (relative) spectral perturbation of the first r eigenvalues due to the inclusion of edge (p, q) becomes:

$$\Delta_r(e_{p,q}) = \sum_{i=2}^r \frac{\delta \lambda_i}{\lambda_i} = \delta w_{p,q} \|U_r^\top e_{p,q}\|_2^2. \quad (26)$$

A.2 ALGORITHM FLOW

Algorithm 1 The GRASPEL Algorithm Flow

Input: A data matrix $(X = [x_1, \dots, x_M] \in \mathbb{R}^{N \times M})$ with N data points in M -dimensional, embedding distortion tolerance ($1 \leq tol$), window size for edge sampling ($0 < \epsilon \leq 50\%$), edge sampling ratio ($0 < \zeta \leq 1$), and the number of edges to be selected in each iteration ($0 < s$).
Output: The spectrally-learned graph G .

- 1: Construct an initial 2NN graph G using approximate kNN algorithms.
 - 2: **while** $\eta_{max} \geq tol$ **do**
 - 3: Embed the latest graph G using its Fiedler vector and sort the nodes into a 1D array I_{node} ;
 - 4: Obtain node set N_{top} (N_{bot}) by including only the top (bottom) $\lceil \epsilon N \rceil$ nodes in I_{node} ;
 - 5: Sample each of the $\lceil s/\zeta \rceil$ edges by randomly choosing one node from N_{top} and another node from N_{bot} ;
 - 6: Form an edge set E_{sel} using edges with large distortions ($\eta \geq tol$) and set the largest edge embedding distortion as η_{max} .
 - 7: **if** $|E_{sel}| \geq s$ **then**
 - 8: Add the top s edges with largest η from E_{sel} into G ;
 - 9: **else**
 - 10: Add all the edges in E_{sel} into G ;
 - 11: **end if**
 - 12: **end while**
 - 13: Return the learned graph G .
-

Algorithm 2 Spectral Clustering Algorithm

Input: A graph $G = (V, E, w)$ and the number of clusters r .
Output: Clusters $C_1 \dots C_r$.

- 1: Compute the adjacency matrix A , and diagonal matrix D ;
 - 2: Obtain the unnormalized Laplacian matrix $L = D - A$;
 - 3: Compute the eigenvectors u_1, \dots, u_r that correspond to the bottom r nonzero eigenvalues of L ;
 - 4: Construct $U_r \in \mathbb{R}^{N \times r}$, with r eigenvectors of L stored as columns;
 - 5: Perform k-means algorithm to partition the rows of U_r into r clusters and return the result.
-

A.3 DATA SETS DESCRIPTION

COIL20: the data set contains 1,440 gray-scale images of 20 objects, and each object on a turntable has 72 normalized gray-scale images taken from different degrees. The image size is 32x 32 pixels.

PenDigits: the data set consists of 7,494 images of handwritten digits from 44 writers, using the sampled coordination information. Each digit is represented by 16 attributes.

USPS: the data set includes 9,298 scanned hand-written digits on the envelopes from U.S. Postal Service with 256 attributes.

MNIST: the data set consists of 70,000 images of handwritten digits. Each image has 28-by-28 pixels in size. This database can be found at website (<http://yann.lecun.com/exdb/mnist/>).

A.4 COMPARED ALGORITHMS

Standard kNN: the most widely used affinity graph construction method. Each node is connected to its k nearest neighbors.

Consensus of kNN (cons-kNN) (Premachandran & Kakarala, 2013): adopts the state-of-the-art neighborhood selection methods to construct the affinity graphs. It selects strong neighborhoods to improve the robustness of the graph by using the consensus information from different neighborhoods in a given kNN graph.

LSGL (Kalofolias & Perraudin, 2019): a method to automatically select the parameters of the model introduced in (Kalofolias, 2016) given a desired graph sparsity level. The default settings have been used in our experiments.

A.5 EVALUATION METRIC

(1) **The ACC metric** measures the agreement between the clustering results generated by clustering algorithms and the ground-truth labels. A higher value of ACC indicates better clustering quality. The ACC can be computed by:

$$ACC = \frac{\sum_{j=1}^n \delta(y_i, \text{map}(c_i))}{n}, \quad (27)$$

where n is the number of samples in the data set, y_i is the ground-truth label provided by the data sets, and c_i is clustering result obtained from the algorithm. $\delta(x, y)$ is a delta function defined as: $\delta(x, y)=1$ for $x = y$, and $\delta(x, y)=0$, otherwise. $\text{map}(\bullet)$ is a permutation function that maps each cluster index c_i to a ground truth label, which can be realized using the Hungarian algorithm (Papadimitrou & Steiglitz, 1982).

(2) **The NMI metric** is in the range of $[0, 1]$, while a higher NMI value indicates a better matching between the algorithm generated result and ground truth result. For two random variables P and Q , normalized mutual information is defined as (Strehl & Ghosh, 2002):

$$NMI = \frac{I(P, Q)}{\sqrt{H(P)H(Q)}}, \quad (28)$$

where $I(P, Q)$ denotes the mutual information between P and Q , while $H(P)$ and $H(Q)$ are entropies of P and Q . In practice, the NMI metric can be calculated as follows (Strehl & Ghosh, 2002):

$$NMI = \frac{\sum_{i=1}^k \sum_{j=1}^k n_{i,j} \log(\frac{n \cdot n_{i,j}}{n_i \cdot n_j})}{\sqrt{(\sum_{i=1}^k n_i \log \frac{n_i}{n})(\sum_{j=1}^k n_j \log \frac{n_j}{n})}}, \quad (29)$$

where n is the number of data points in the data set, k is the number of clusters, n_i is the number of data points in cluster C_i according to the clustering result generated by algorithm, n_j is the number of data points in class C_j according to the ground truth labels provided by the data set, and $n_{i,j}$ is the number of data points in cluster C_i according to the clustering result as well as in class C_j according to the ground truth labels.

A.6 ADDITIONAL EXPERIMENTAL RESULTS

A.6.1 GRAPH LEARNING FOR SPECTRAL CLUSTERING (SC)

The classical spectral clustering (SC) algorithm (see Algorithm 2 in the Appendix) first constructs a kNN graph where each edge weight encodes similarities between different data points (entities); then SC calculates the eigenvectors of the graph Laplacian matrix and embeds data points into low-dimensional space (Belkin & Niyogi, 2003); in the last, k-means algorithms are used to partition the data points into multiple clusters. The performance of SC strongly depends on the quality of the underlying graph (Guo, 2015). In this section, we apply GRASPEL for graph construction, and show the learned graphs can result in drastically improved efficiency and accuracy in SC tasks.

Table 1: Spectral Clustering Results

Data Set	ACC(%) / NMI / Time-C (seconds) / Time-S (seconds)			
	Standard KNN ($k = 10$)	ConskNN (Premachandran & Kakarala, 2013)	LSGL (Kalofolias & Perraudin, 2019)	GRASPEL
COIL-20	78.80/ 0.86/ 0.36 /0.37	79.86/ 0.86/ 0.54/0.28	65.48/0.77/60.22/1.02	90.27/ 0.96/ 0.40/0.19
PenDigits	81.12/ 0.80/ 1.25 /0.47	84.17/ 0.81/ 8.59/46.41	75.01/0.72/1.622/15.64	85.96/ 0.82/ 4.51/0.27
USPS	68.22/ 0.77/ 2.66 /1.02	78.94/ 0.82/ 19.82/74.57	72.35/0.71/2.598/29.37	92.59/ 0.87/ 5.19/0.21
MNIST	71.95/ 0.72/ 242.38/6,785	-	-	81.67/ 0.75/ 59.27/2.90

- indicates that the method is not capable for handling data sets of this scale.

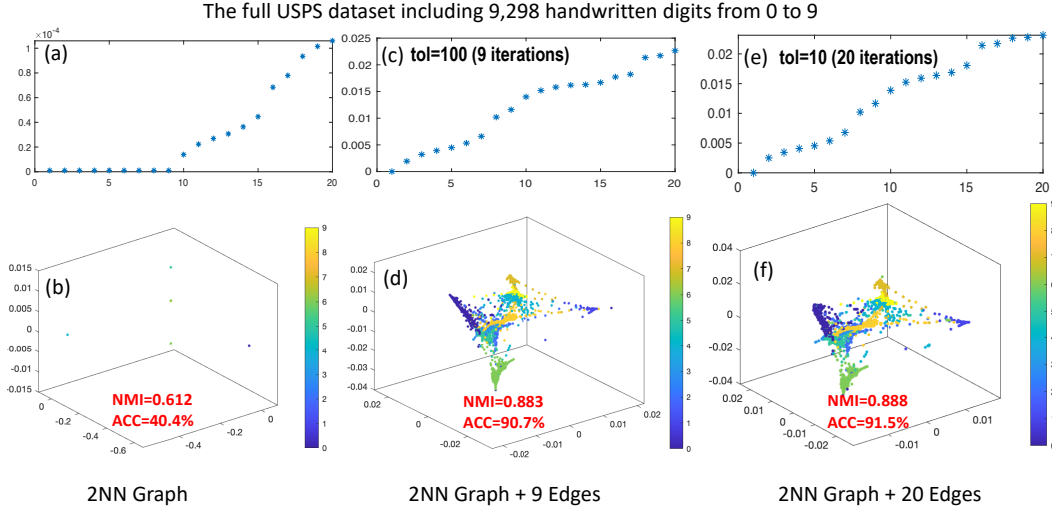


Figure 4: The first 20 Laplacian eigenvalues (top) and 3D spectral drawings (bottom) of the 2NN graph in figures (a) and (b), and the GRASPEL-learned graphs in figures (c) to (f).

Table 1 shows the ACC and NMI results of SC with graphs constructed by different methods with the best numbers highlighted, where graph construction time (Time-C) and spectral clustering time (Time-S) that involves eigendecomposition and kmeans clustering have also been reported. Note that the high computational and memory cost of recent GSP-based graph learning methods, such as GL-SigRep (Dong et al., 2016), GL-Logdet (Dong et al., 2016) and GLSC (Egilmez et al., 2017) do not allow for processing data sets with more than a few thousands data points, thus can not be used for real-world SC tasks. As observed, GRASPEL can consistently lead to dramatic performance improvement in SC, beating all competitors in clustering accuracy (ACC) across all data sets. Note that when starting with uNN graphs, the graphs learned by GRASPEL are ultra sparse, thereby allowing much faster eigendecompositions in SC when comparing with other methods (Wang & Feng, 2017): the SC of the MNIST data set with standard kNN ($k = 10$) takes over 6,000 seconds, which will be dramatically improved to require less than three seconds (over 2,000X speedup) using the graph learned by our method (GRASPEL).

In Figures 4 and 5, additional SC results have been provided for the USPS and the Pendigit data sets by comparing two embedding distortion tolerance levels ($tol = 100$ and $tol = 10$). Not surprisingly, when starting with initial 2NN graphs a few GRASPEL iterations have already dramatically improved SC results: the normalized mutual information (NMI) has been improved from 0.612 to 0.888 for the USPS data set, and from 0.04 to 0.840 for the Pendigit data set; The spectral clustering accuracy (ACC) has been improved from 40.4% to 91.5% for the USPS data set, and from 15.2% to 89.4% for the Pendigit data set.

Since the number of zero Laplacian eigenvalues equals to the connected components in the learned graph, we observe that GRASPEL can always identify $O(q)$ spectrally-critical edges added to the initial 2NN graph so that its q connected components immediately get stitched into a connected graph for both test cases.

Discussion. The superior performance of GRASPEL is due to the following reasons: 1) In traditional kNN graphs, all the nodes have the same degrees; as a result, the clustering may strongly favor balanced cut, which may lead to improper cuts in high-density regions of the graph. In con-

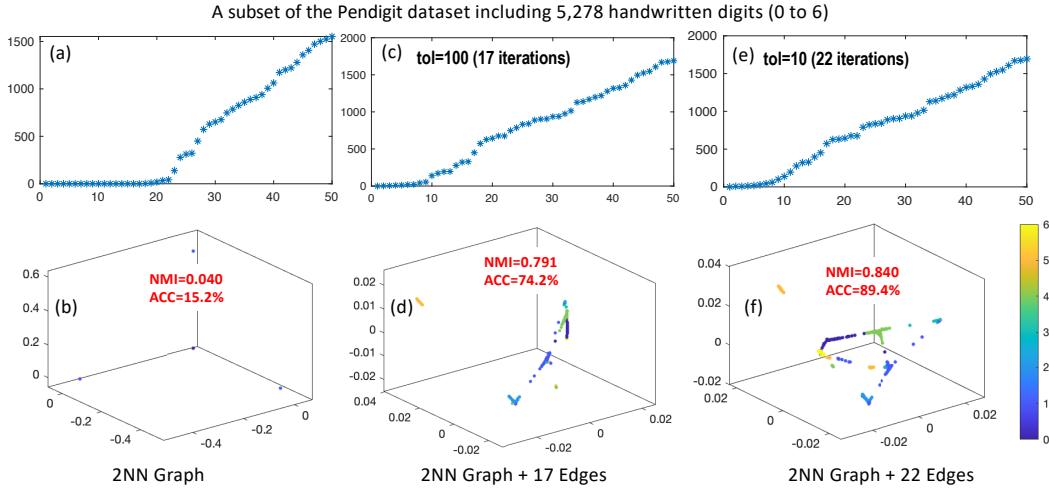


Figure 5: The first 50 Laplacian eigenvalues (top) and 3D spectral drawings (bottom) of the 2NN graph in figures (a) and (b), and the GRASPEL-learned graphs in figures (c) to (f).

trast, GRASPEL always learns ultra-sparse graphs that only include edges with the largest impact to graph spectral (structural) properties; as a result, the corresponding cuts will always occur in proper regions of the graph, which enables to handle even unbalanced data. **2)** Recent work (Garg et al., 2018) shows the fundamental connections between spectral properties of graphs associated with data and the inherent robustness to adversarial examples. Since GRASPEL identifies candidate edges by leveraging spectral graph properties, the learned graph structure will also be robust to input noises (perturbations).

A.6.2 GRAPH LEARNING FOR DIMENSIONALITY REDUCTION (DR)

The t-Distributed Stochastic Neighbor Embedding (t-SNE) has become one of the most popular visualization tools for high-dimensional data analytic tasks (Maaten & Hinton, 2008; Linderman & Steinerberger, 2017). However, its high computational cost limits its applicability to large scale problems. An substantially improved t-SNE algorithm has been introduced based on tree approximation (Van Der Maaten, 2014). However, for large data set the computational cost can still be very high.

A multilevel t-SNE algorithm has been proposed in (Zhao et al., 2018) leveraging spectral graph coarsening as a pre-processing step applied to the original kNN graph. A much smaller set of representative data points can be then selected from the coarsened graph for t-SNE visualization. In this work, we use GRASPEL to learn sparse graphs that can be further reduced into much smaller ones using spectral graph coarsening. Then more efficient t-SNE visualization can be achieved based on the sampled data points corresponding to the nodes in the coarsened graphs.

In our experiments, we first construct initial graphs by applying a spectral sparsification procedure to the 5NN graphs of both the MNIST and USPS data sets. Then a spectral graph coarsening procedure (Zhao et al., 2018) has been applied to create a hierarchy of coarse-level graphs. The t-SNE visualization can be obtained by directly using the data points corresponding to the nodes on the coarsest graph. Figure 6 shows the visualization and runtime results of the standard t-SNE (with tree-based acceleration) (Van Der Maaten, 2014) and the multilevel t-SNE algorithm (Zhao et al., 2018) based on graphs learned by GRASPEL. When using a 5X graph reduction ratio, t-SNE can be dramatically accelerated (12.8X and 7X speedups for MNIST and USPS data sets, respectively) without loss of visualization quality.

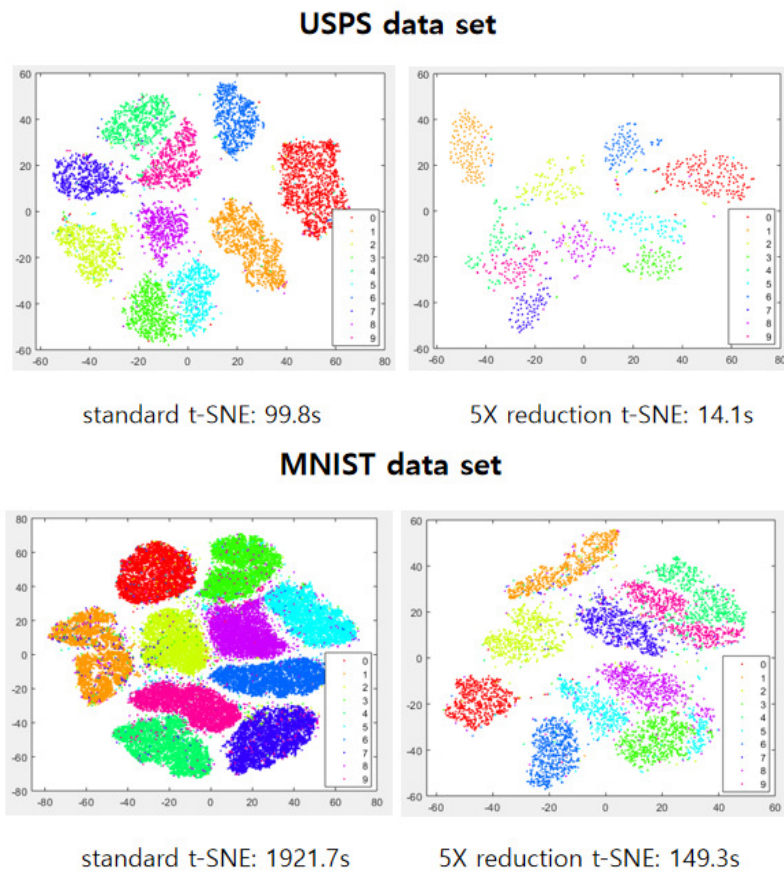


Figure 6: Multilevel t-SNE visualization results.