Appendix A: Detailed Algorithms for Section 4

Steadiness and Cohesiveness (Section 4.2)

Algorithm 1 Computing Steadiness and Cohesiveness

Input The set of points $P = \{p_1, p_2, \cdots, p_n\}$ in dataset where $|P| = n$
Input Each point p_i's original coordinate h_i and projected coordinate l_i
Input Number of the nearest neighbors k, walking number w, iteration number *iter*
Input Hyperparameter functions `dist`, `dist_cluster`, `extract_cluster`, `clustering`
Output Steadiness or Cohesiveness
$D^+, D^- \leftarrow$ COMPUTEDISSIMILARITYMATRIX(n, $\{h_1, \cdots, h_n\}$, $\{l_1, \cdots l_n\}$, `dist`) ▷ Step 1
distortions = [] ▷ initialize empty array
for $i \leftarrow 1$ to *iter* **do** ▷ Step 2
    $s \leftarrow$ **random**($[1, n]$) ▷ the index of random seed point
    distortions.**concat**(COMPUTEPARTIALDISTORTION(s, P, D^+, D^-, `clustering`, `extract_cluster`))
distortionSum $\leftarrow 0$, *weightSum* $\leftarrow 0$
for (m_i, w_i) in *distortions* **do** ▷ Step 3
    distortionSum $\leftarrow$ *distortionSum* $+ m_i \cdot w_i$
    weightSum $\leftarrow$ *weightSum* $+ w_i$
return $1 -$ *distortionSum*/*weightSum*

procedure COMPUTEDISSIMILARITYMATRIX(n, $\{h_1, \cdots, h_n\}$, $\{l_1, \cdots l_n\}$, `dist`)
    Initialize $n \times n$ matrix H, L
    for $i \leftarrow 1$ to n **do**
        for $j \leftarrow 1$ to n **do**
            $H_{ij} \leftarrow$ `dist`(h_i, h_j)
            $L_{ij} \leftarrow$ `dist`(l_i, l_j)
    $H \leftarrow H/H_{max}$, $L \leftarrow L/L_{max}$ ▷ normalize H, L by their max elements
    $D = H - L$
    Initialize $n \times n$ matrix D^+, D^-
    for $i \leftarrow 1$ to n **do**
        for $j \leftarrow 1$ to n **do**
            $D^+_{ij} \leftarrow D_{ij}$ if $D_{ij} > 0$ else 0
            $D^-_{ij} \leftarrow -D_{ij}$ if $D_{ij} < 0$ else 0
    return D^+, D^-

procedure COMPUTEPARTIALDISTORTION(s, P, D^+, D^-, `clustering`, `extract_cluster`)
    $P_s \leftarrow$ `extract_cluster`(p_s, P) ▷ $p_s \in P_s$, $P_s \subset P$
    $\mathscr{C} \leftarrow$ `clustering`(P_s)
    distortions $\leftarrow$ [] ▷ initialize empty array
    for C_i in $\mathscr{C}$ **do**
        for C_j in $\mathscr{C}$ **do**
            $\delta_h, \delta_l =$ `dist_cluster`(C_i, C_j) ▷ distances between C_i and C_j in the original and projected space
            if Steadiness Case **then**
                $\mu_{C_i, C_j} \leftarrow -(\delta_h - \delta_l)$ if $-(\delta_h - \delta_l) > 0$ else 0
                $m_{ij} \leftarrow \frac{\mu^{stretch}_{C_i, C_j} - \min D^-}{\max D^- - \min D^-}$
            else ▷ Cohesiveness Case
                $\mu_{C_i, C_j} \leftarrow \delta_h - \delta_l$ if $\delta_h - \delta_l > 0$ else 0
                $m_{ij} \leftarrow \frac{\mu_{C_i, C_j} - \min D^+}{\max D^+ - \min D^+}$
            $w_{ij} \leftarrow |C_i| \cdot |C_j|$
            distortions.**append**$((m_{ij}, w_{ij}))$ ▷ add new element
    return *distortions*

Pointwise Distortion Measurement for the Reliability Map (Section 4.4)

Algorithm 2 Computing Pointwise Distortions

Input The set $\mathbb{C}$ which contains every pair of groups (C_i, C_j) with distortion m_{ij} and weight w_{ij}
Input The set of points $P = \{p_1, p_2, \cdots, p_n\}$ in dataset where $|P| = n$
Output The set of pointwise distortion $DIST = \{dist_1, dist_j, \cdots, dist_n\}$ where $dist_i$ is the pointwise distortion of p_i

for (C_i, C_j) in $\mathbb{C}$ **do** ▷ registering points and corresponding distortion strengths
  for point p_{i,k_i} in C_i **do**
    for point p_{j,k_j} in C_j **do**
      $d \leftarrow m_{ij} \cdot w_{ij}$ ▷ distortion strengths
      Register (p_{j,k_j}, d) to p_{i,k_i}
      Register (p_{i,k_i}, d) to p_{j,k_j}
for data point p_i in P **do**
  $dup \leftarrow$ the points that have been registered multiple times to p_i
  for q in dup **do**
    Remove the duplicated registration of q by averaging distortion strengths
  $dist_i \leftarrow 0$
  for each registered point and distortion strengths (q, d) of p_i **do**
    $dist_i \leftarrow dist_i + d$
return $\{dist_1, dist_2, \cdots, dist_n\}$

Appendix B: Additional Reliability Maps for MNIST

Among the *t*-SNE, UMAP, PCA, Isomap, and LLE projections that we used for the MNIST exploration with ML engineers (Section 7.1), we depicted the projections and the reliability map of UMAP, LLE, and Isomap in Figure 5. Here, we present the remaining projections in which generated by PCA and *t*-SNE.

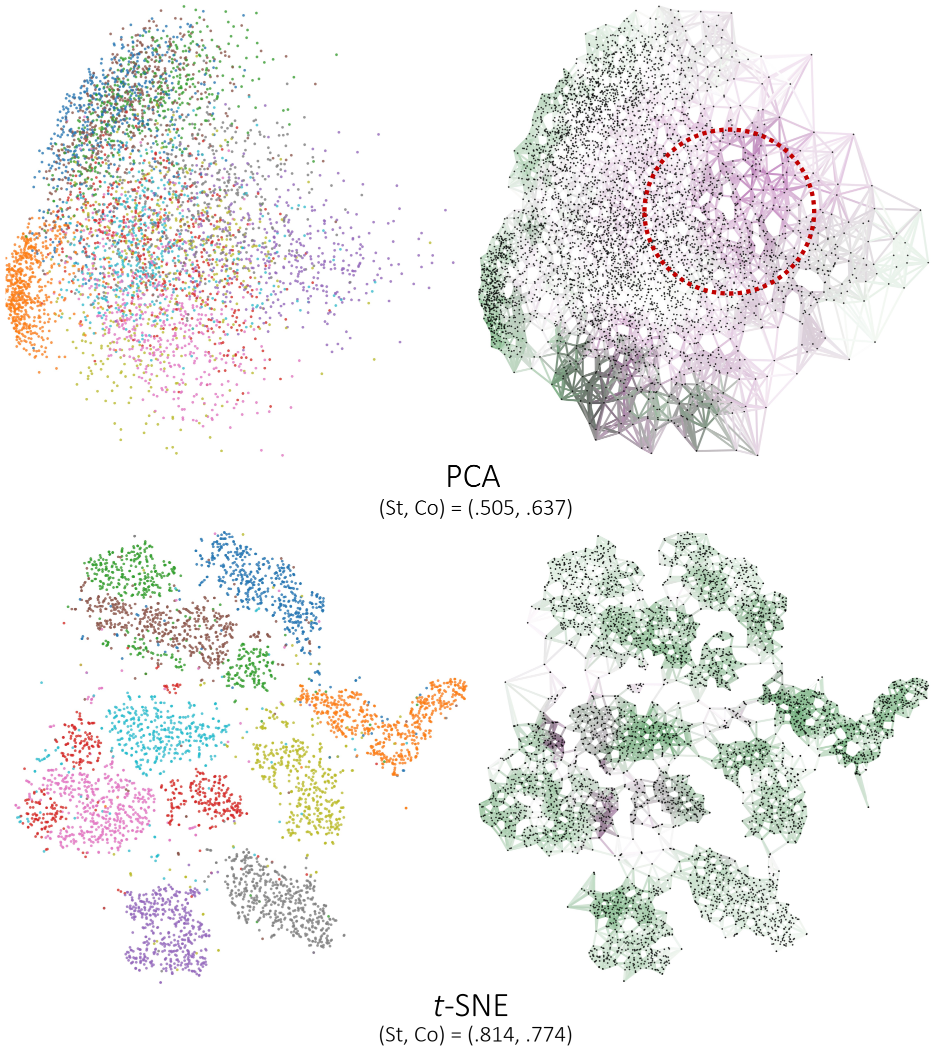

Figure 1: The PCA and *t*-SNE projections of MNIST test dataset and the reliability maps that visualize each projection' s inter-cluster distortion values. Steadiness (St) and Cohesiveness (Co) scores are depicted under the name of each technique. For each MDP technique, the left pane shows the class information, and the right pane shows the reliability map.

As aforementioned in Section 7, we found the region mainly consists of categories #4 and #7 and is highlighted as the area with high False Groups distortion (red dotted circle). The same phenomena also occurred in the Isomap projection, which explains both PCA and Isomap's low Steadiness score. Moreover, the reliability map showed that *t*-SNE suffered from Missing Groups distortion as UMAP did, which aligns to the ground truth that digits in MNIST stay much closer than they look in the projections generated by *t*-SNE or UMAP.

Appendix C: Scalability Report

We tested the metrics' scalability on a Linux server with a 40-core Intel Xeon Silver 4210R CPU and a TITAN RTX GPU. The execution time is as follows.

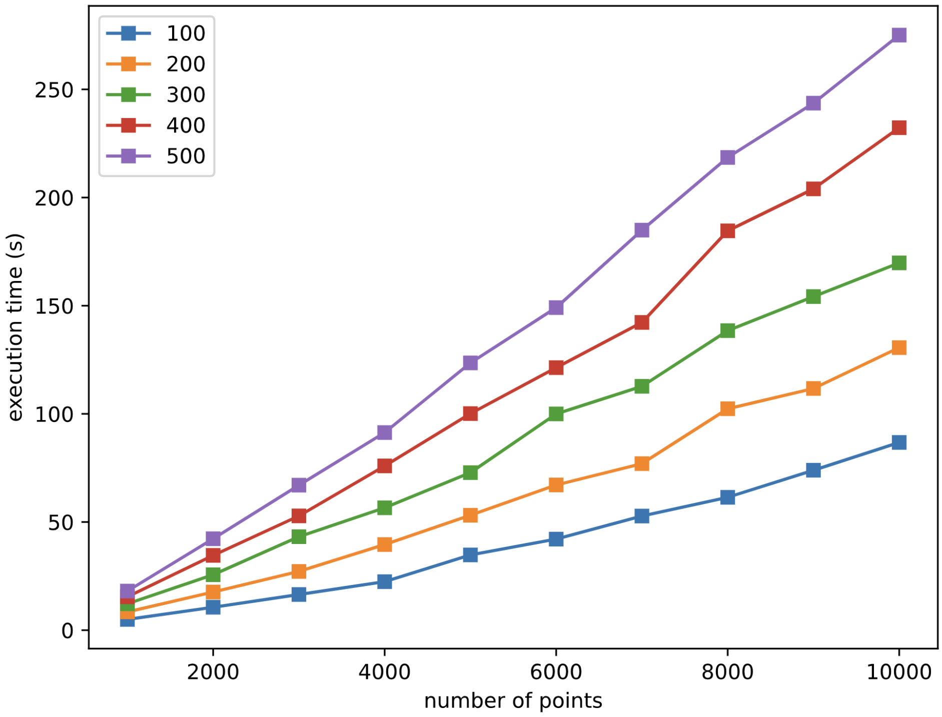

Figure 2: The execution time of computing Steadiness and Cohesiveness of the *t*-SNE projection representing MNIST dataset, where each line corresponds to the number of iterations. The execution time increases in proportion to the iteration number and number of points.

As our current implementation executes each iteration of partial distortion computation sequentially, the running time increases in proportion to the iteration number. Our future goal is to parallelize this bottleneck by utilizing multiprocessing (Section 8).