Self Organizing Nebulous Growths for Robust and Incremental Data Visualization

Damith A. Senanayake, Wei Wang, Shalin H. Naik, Saman Halgamuge

Abstract—Non-parametric dimensionality reduction techniques, such as t-SNE and UMAP, are proficient in providing visualizations for fixed or static datasets, but they cannot incrementally map and insert new data points into existing data visualizations. We present Self-Organizing Nebulous Growths (SONG), a parametric nonlinear dimensionality reduction technique that supports incremental data visualization, i.e., incremental addition of new data while preserving the structure of the existing visualization. In addition, SONG is capable of handling new data increments no matter whether they are similar or heterogeneous to the existing observations in distribution. We test SONG on a variety of real and simulated datasets. The results show that SONG is superior to Parametric t-SNE, t-SNE and UMAP in incremental data visualization. Specifically, for heterogeneous increments, SONG improves over Parametric t-SNE by 14.98% on the Fashion MNIST dataset and 49.73% on the MNIST dataset regarding the cluster quality measured by the Adjusted Mutual Information scores. On similar or homogeneous increments, the improvements are 8.36% and 42.26% respectively. Furthermore, even in static cases, SONG performs better or comparable to UMAP, and superior to t-SNE. We also demonstrate that the algorithmic foundations of SONG render it more tolerant to noise compared to UMAP and t-SNE, thus providing greater utility for data with high variance or high mixing of clusters or noise.

I. INTRODUCTION

In data analysis today, we often encounter high-dimensional datasets with each dimension representing a variable or feature. For example, in experimental biology, one typically records the expression levels of thousands of genes per cell [1] or per population [2]. When analysing such datasets, reducing the data dimensionality is highly useful to gain insights into the structure of the data. Visualization of high-dimensional data is achieved by reducing the data down to 2 or 3 dimensions.

In practice, we often assume static data visualization, i.e., the data are presented to the dimensionality reduction methods all at once. However, with the advent of big data, the data may be presented incrementally for the following two main reasons. First, the dataset may be extremely large and has to be divided and processed sequentially [3]. Second, there are scenarios where data is incrementally acquired through a series of experiments, such as the continuous acquisition of Geographical Data [4] or data gathered by mining social media [5]. In Fig. 1 we show how data can be augmented with either homogeneous data (new data that has a structure similar to the already observed structure) or heterogeneous data (new data that has a structure unlike the already observed structure). In real-world situations, both scenarios may be present indistinguishably, and these necessitate incremental data visualization, where we either directly use or continually train a pre-trained model to visualize the data increments. In addition, it is often required that the visualization on existing data do not change drastically after the new data is added for consistency in data interpretation. In the following, we explore the applicability of existing dimensionality reduction techniques for incremental data visualization.

In static data visualization scenarios, t-distributed Stochastic Nonlinear Embedding (t-SNE) [6] and Uniform Manifold Approximation and Projection (UMAP) [7] are two state-of-the-art methods frequently used for dimensionality reduction. Both t-SNE and UMAP first create a graph in the input space, where
the vertices are the input points and the edge-weights represent the pseudo-probability of two inputs being in the same local neighborhood. t-SNE calculates the edge-weights by assuming Gaussian distributions in local neighborhoods. These pseudo-probabilities are then replicated on an output map of low-dimensionality (typically 2 or 3). However, to allow for clear cluster separation, t-SNE assumes the output probabilities to be sampled from a student’s t-distribution. UMAP follows a different strategy to calculate the edge weights in the input space, by assuming that the local neighborhoods lie on a Riemannian manifold. Using this assumption, UMAP normalizes the local pairwise distances to obtain a fuzzy simplicial set that represents a weighted graph similar to that of t-SNE. UMAP then uses a suitable rational quadratic kernel function in low-dimensional output space to approximate the edge probabilities of the weighted graph.

Being non-parametric models, t-SNE and UMAP do not retain a mapping from the inputs to the outputs, and thus cannot be directly applied for incremental data visualization. Instead, t-SNE and UMAP need to be reinitialized and retrained at each increment of data. In UMAP, the heuristic initialization using spectral embedding (Laplacian Eigenmaps) provides some degree of stability in visualizations of datasets from the same distribution as similar datasets have similar nearest neighbor graphs that provide similar graph laplacians. However, it remains to be answered how well such heuristic initialization performs when the new data have heterogeneous structure to the existing data.

Previous studies on t-SNE and UMAP have attempted to retain a parametric model for new data. One approach is to train a neural network regressor on visualizations obtained by t-SNE or UMAP. However, to our best knowledge, there is no existing work with a generally applicable version of this approach. Another approach is to use deep Latent Variable Models, e.g., Variational Autoencoders [9] and parametric t-SNE [10], to represent the low-dimensional visualizations. However, it is questionable whether such approaches provide visualizations with comparable quality of cluster separation to the non-parametric counterparts of the respective algorithms. Additionally, these methods suffer from issues commonly associated with deep neural networks such as requiring a large amount of training data [11], high computational complexity [12] and lack of model interpretability [13].

Self Organizing Map (SOM) [14] and its variants are arguably the only dimensionality reduction methods that retain a parametric graph on the input space to approximate the input data distribution locally. SOMs obtain the graph by vector encoding, i.e., SOMs partition the input space into Voronoi regions by mapping each input to the closest element in a set of representative vectors called Coding Vectors. These Coding Vectors, now representing the centroids of the voronoi regions, are mapped onto a low-dimensional (typically 2 or 3) uniform output grid. Such a uniform (or regular) grid can have either a square, triangular or hexagonal topology of locally connected output vectors. The topology preservation of SOMs is achieved by moving the set of coding vectors in the input space such that the coding vectors corresponding to neighbors in the low-dimensional output grid are placed close together.

There are two main problems associated with the visualizations provided by SOMs. First, the SOM visualizations have poor cluster separation possibly due to the uniform nature of the output grid. Since all edges in the output graph is of equal length, the difference between the edge lengths in the input space are not preserved in the output grid [13]. Second, the size of the map (the number of coding vectors ) needs to be known a priori. Growing Cell Structures (GCS), presented by Fritzke [15], uses a non-uniform triangulation (where the coding vectors represent the vertices of the triangles) of the input space to tackle these two problems. However, the graph inferred in GCS is planar thus non-planar graph structures in local neighborhoods cannot be preserved. Fritzke’s later work of Growing Neural Gas (GNG) [16] approximates non-planar graph structures on the coding vectors (by having more densely connected local subgraphs than GCS) in local neighborhoods. However, due to this non-planarity, GNG cannot provide a visualization of the inferred graph by triangulating a set of low-dimensional points as GCS does as such triangulation relies on the planarity of the graph. A possible approach for visualizing the coding vectors of a static Neural Gas using a Cross Entropy measure (NG-CE) has been investigated by Estvez et al. [17], however NG-CE has not been extended to support the dynamically growing nature of GNGs. Furthermore, NG-CE relies having a fully trained set of coding vectors prior to the visualization, and does not support simultaneous approximation of the input topology and the simultaneous projection of the said topology to a low dimensional embedding. Growing SOM (GSOM) [18] provides another approach to overcome SOM’s problem with unknown map size by using a uniform but progressively growing grid. This is different to the GNGs where the grids were non-uniform and non-planar. However, similar to SOM, GSOM still suffers from poor cluster separation due to the uniform output grid. Additionally the topology of localities in the output map of GSOM is limited to be 2 or 3 dimensional.

Inspired by GNG and NG-CE, we propose Self Organizing Nebulous Growths (SONG), a nonlinear dimensionality reduction algorithm with two main advantages over t-SNE, UMAP and their variants. First, SONG retains a parametric model for incremental data visualization, while being significantly more efficient than the parametric variants of t-SNE and UMAP. Second, SONG solves the visualization problems of SOM and produces comparable or better visualizations than t-SNE and UMAP in the static data visualization scenarios. In the following section, we describe our proposed method in detail.

II. Method

In the proposed SONG, we use a set of coding vectors $C = \{ c \in \mathbb{R}^D \}$ to partition and represent the input dataset $X = \{ x \in \mathbb{R}^D \}$. For an input $x_i \in X$, we define an index set $I^{(k)} = \{ ij | l = 1, \ldots, k \}$ for a user-defined $k$, where $c_{ij}$ is the $l$-th closest coding vector to $x_i$. Moreover, we define a set of directional edges between coding vectors $C$ and an adjacency matrix $E$, such that if a coding vector $c_l$ is one of the closest neighbors to another coding vector $c_i$, they are connected by an edge with edge strength $E(l,m) > 0$. We organize the graph $\{ C, E \}$ to approximate the input topology.
We also define a set of low-dimensional vectors \( Y = \{ y \in \mathbb{R}^d \} \), \( d \ll D \) which has a bijection correspondence with the set of coding vectors \( C \). When \( d = 2 \) or \( 3 \), \( Y \) represents the visualization of the input space, i.e., the input \( x_i \in X \) is visualized as \( y_i \in Y \). We preserve the topology of \( C \) given by \( E \) in \( Y \) by positioning \( Y \) such that, if \( E(i, j, m) > 0 \) or \( E(m, l) > 0 \) , \( y_i \) and \( y_m \) will be close to each other in the visualization. Typically, the number of \( c \in C \) and corresponding low dimensional vectors \( y \in Y \) is far less than the number of input data points \( X \). By retaining the parameters \( C \), \( E \) and \( Y \), SONG obtains a parametric mapping from input data to visualization.

We initialize a SONG model by randomly placing \( d + 1 \) coding vectors \( C \) in the input space, since \( d + 1 \) is the minimum number for coding vectors to obtain a topology preserving visualization in a \( d \)-dimensional visualization space (see Supplement Section 1.1 for proof). No edge connection is assumed at initialization (i.e. \( E = 0 \)). The corresponding \( Y \) are also randomly placed in the \( d \)-dimensional output space. Next, we approximate local topology of any given \( x \in X \) using \( C \), and project this approximated topology to \( Y \) in the visualization space. To be specific, SONG randomly samples an input point \( x_i \in X \) and performs the following steps at each iteration until terminated:

1) **Updating the Directional Edges in \( E \) between Coding Vectors based on \( x_i \):** This step modifies the adjacency matrix \( E \) to add or remove the edges between coding vectors based on local density information at \( x_i \). Eventually, if no edge is added or removed for repeated sampling of all input \( x_i \)'s (i.e. the whole dataset is sampled with no addition or removal of edges), the graph is considered stable and SONG has finished its training. We describe this step in detail in Section [II-A]

2) **Self Organization of Coding Vectors \( C \):** This step moves \( x_i \)'s closest coding vector \( c_{i_1} \) closer to \( x_i \), along with any coding vectors \( c_j \) if \( c_{i_1} \) and \( c_j \) are connected by an edge as indicated by \( E(i_1, j) > 0 \). This movement enforces the closeness of coding vectors connected by edges. We describe this in detail in Section [II-B]

3) **Topology Preservation of the Low-dimensional Points \( Y \):** Given that \( c_{i_1} \) encodes \( x_i \) and corresponds to \( y_{i_1} \) in the output space, we organize low-dimensional points \( y_j \in Y \) in the locality of \( y_{i_1} \) such that the coding vector topology at \( c_{i_1} \) is preserved in the output space. This step is described in Section [II-C]

4) **Growing \( C \) and \( Y \) to Refine the Inferred Topology:** There may be cases where \( c_{i_1} \) and its neighboring coding vectors are insufficient to capture the local fine topology at \( x_i \), e.g., inputs from multiple subclusters may have the same \( c_{i_1} \). In such cases, we place new coding vectors close to \( c_{i_1} \), and new corresponding low-dimensional vectors close to \( y_{i_1} \), without reintializing the parametric model \{X, C, E\}. Details of this are in Section [II-D]

For a given epoch, we randomly sample (without replacement) a new input \( x_i \in X \) and repeat the four steps, until all \( x_i \in X \) are sampled. The algorithm is terminated if the graph becomes stable in Step 1) or we have executed the maximum number of epochs. When new data \( X' \) are presented, we simply allow \( x_i \) to be sampled from \( X' \) at the next iteration, and continue training without reinitializing the parameters \( C \), \( Y \) and \( E \).

### A. Updating the Directional Edges in \( E \) between Coding Vectors based on \( x_i \)

For each input \( x_i \) randomly sampled from \( X \), we conduct three operations to any edge-strength \( e_{i_1j} = E(i_1, j) \) at current iteration \( t \):

- **Renewal:** we reset \( e_{i_1j}^t \) to 1 if \( j \in I^{(k)} \).
- **Decay:** if \( e_{i_1j}^{t-1} > 0 \), we decay it by a constant multiplier \( \epsilon \), i.e., \( e_{i_1j}^t = e_{i_1j}^{t-1} \cdot \epsilon \) and \( \epsilon \in (0, 1) \).
- **Pruning:** we set \( e_{i_1j}^t \) to 0 if \( e_{i_1j}^{t-1} < \epsilon_{\text{min}} \). This helps to obtain a sparse graph. Note that \( \epsilon_{\text{min}} \) is predefined to obtain the desired degree of sparseness in the graph.

The edge strength \( e_{i_1j} \) reflects the rate the edge is renewed and is proportional to \( p_{i_1j} \), the probability of \( i_1 \) and \( j \) being close neighbors to the input \( x_i \). Since the edge strengths \( e_{i_1j} \in [0, 1] \), larger edge strengths can be interpreted as \( c_{i_1} \) and neighboring coding vectors representing finer topologies (shorter distances between \( C \)). Conversely, the smaller the edge strengths, the coarser the topology represented by such edges. Note that here we define finer and coarser topologies in their conventional sense [19].

Note that edge strength \( e_{i_1j} \) obtained above is directional and thus the adjacency matrix \( E \) is asymmetric. We observe faster convergence in subsequent optimization with a symmetric adjacency matrix, which is simply calculated as:

\[
E_s = \frac{E + E^T}{2}
\]

Next, we use the coding vector graph to approximate the topology of the input through self organization.

### B. Self-Organization of Coding Vectors \( C \)

To ensure the coding vectors \( C \) are located at the centers of input regions with high probability densities (such as cluster centers), we move the coding vectors \( c_{i_1} \) towards \( X \) by a small amount to minimize the following Quantization Error (QE):

\[
QE(x) = \frac{1}{2} \| x_i - c_{i_1} \|^2
\]

However, moving \( c_{i_1} \) independent of other coding vectors may cause the coding vectors sharing an edge with \( c_{i_1} \) to be no longer close to \( c_{i_1} \), which disorganizes the graph. To avoid this, we also move \( c_{i_1} \)'s neighboring coding vectors \( c_j \) (as indicated by \( E_s(i_1, j) > 0 \)) towards \( x_i \) by a smaller amount than that of \( c_{i_1} \). Moreover, the more distant \( c_j \) is from \( c_{i_1} \), the smaller the movement of \( c_j \) should be. This ensures the organization of distant neighbors is proportionately preserved by this movement. Therefore, we define a loss function that monotonically decreases when the distance from the coding vectors to \( x_i \) increases. In addition, to penalize large neighborhoods (and thereby large edge lengths), we scale the loss function by \( \| x_i - c_{i_1} \|^2 \). Note that we treat the distance to the \( k^{th} \) coding vector from \( x \) as a constant for the considered
neighborhood, and the final loss gradient of $\|x_i - c_{i_k}\|$ w.r.t. $c_{i_k}$ is not calculated. The final loss function is:

$$L'(x_i) = -\frac{\|x_i - c_{i_k}\|^2}{2} \sum_{c_{i_k} \in \mathcal{N}_{i_k}} \exp(-\frac{\|x_i - c_{i_k}\|^2 \mathcal{N}_{i_k}}{2})$$

(3)

where $\mathcal{N}_{i_k} = \{c_{i_k} \mid E_s(i, j) > 0\}$ is the set of $c_{i_k}$’s neighboring coding vectors. Using stochastic gradient descent to minimize this loss, we calculate the partial derivatives of the loss w.r.t. a given $c$ for the sampled $x_i$ as:

$$\frac{\partial L'(x_i)}{\partial c} = (x_i - c) \times \exp(-\frac{\|x_i - c\|^2 \mathcal{N}_{i_k}}{2})$$

(4)

Next, we describe how we optimize the output embedding (the placement of $Y$) to reflect the topology inferred in the input space.

C. Topology Preservation of the Low-dimensional Points $Y$

Similar to UMAP and inspired by NG-CE, SONG optimizes the embedding $Y$ by minimizing the Cross Entropy ($CE$) between the probability distribution $p$ in the input space and a predefined low-dimensional probability distribution $q$ in the output space. We define the local cross entropy for a given $x_i \in X$ as:

$$CE(x_i) = \sum_{j} -p_{i,j} \log(q_{i,j}) - (1-p_{i,j}) \log(1-q_{i,j})$$

(5)

where $q_{i,j}$ is the probability that output points $y_i$ and $y_j$ are located close together, and is calculated using the following rational quadratic function:

$$q_{i,j} = \frac{1}{1 + a\|y_i - y_j\|^2b}$$

(6)

See Supplement Section 2 for how to calculate the hyper-parameters $a$ and $b$.

The cross entropy (see Eq. 5) can be interpreted as two sub-components: an attraction component $CE_{attr} = \sum_{j} -p_{i,j} \log(q_{i,j})$ that attracts $y_i$ towards $y_j$, and a repulsion component $CE_{rep} = \sum_{j} (1-p_{i,j}) \log(1-q_{i,j})$ that repels $y_i$ from $y_j$. The attraction component heavily influences local arrangement at $y_i$, since distant $y_j$ results in $p_{i,j} = 0$. The repulsion component, on the other hand, influences the global arrangement of neighborhoods, as $1-p_{i,j} = 1$ for such distant $y_j$. Due to the difference of influences, we derive the gradients of these two components separately. The attraction component has a gradient as defined by:

$$\frac{\partial CE_{attr}}{\partial y} = (y_i - y_j) \cdot \frac{2ab \cdot p_{i,j}}{1 + a\|y_j - y_i\|^2b}$$

(7)

The gradient for the repulsion component is:

$$\frac{\partial CE_{rep}}{\partial y} = (y_i - y_j) \cdot \frac{2b \cdot (1-p_{i,j})}{\|y_j - y_i\|^2(1 + a\|y_j - y_i\|^2b)}$$

(8)

We use stochastic gradient descent to minimize $CE_{attr}$ and $CE_{rep}$ w.r.t $y_i, y_j, Y$. We select stochastic gradient descent over batch gradient descent to avoid convergence on sub-optimal organizations [20]. Moreover, since the edge renewal rate is proportional to the $p_{i,j}$, we propose to use the symmetric edge strengths $\tilde{e}_{i,j} \in E_s$ as an approximation of $p_{i,j}$ in $CE_{attr}$. This avoids the explicit assumptions on $p_{i,j}$ as made by t-SNE and UMAP. In a similar fashion, we use the negative sampling of edges (i.e. sampling of $j$ where $E_s(i, j) = 0$) to approximate $(1-p_{i,j})$ in $CE_{rep}$. In this negative sampling, for a very large dataset, we randomly sample a set of non-edges, such that for each $x_i$, the number of sampled non-edges $n_{i,n}$ equals the number of edges connected to $c_{i_k}$ by a constant rate. Similar ideas have been used in Word2Vec [21] and UMAP [7]. The algorithmic summary of this step is shown in Algorithm 3.

D. Growing C and Y to Refine the Inferred Topology

By iterating the above three steps from Sections II-B - II-C, the topology of $x$ can be approximated using $\{C, E\}$ and preserved onto $Y$ in the visualization space. However, since the optimal number of coding vectors $C$ is unknown a priori, SONG starts with a small number of $C$ which may be insufficient to capture all the structures in $X$ such as clusters and sub-clusters. Therefore, we grow the sizes of $C$ and $Y$ as needed during training. Additionally, such growth can accommodate structural changes, e.g., addition of new clusters, when new data are presented in the incremental data visualization scenarios. Note that this growth of coding vectors and low-dimensional vectors is conditional, so this step may not be done for certain iterations.

Inspired by the GNG, we define a Growth Error associated with $c_{i_k}$ as:

$$G_{i_k}(t) \leftarrow G_{i_k}(t-1) + \|x_i - c_{i_k}\|$$

(9)

where $t$ is the index of current iteration. When any $G_{i_k}(t)$ exceeds a predefined threshold, we place a new coding vector $c$ at the centroid between $x_i$ and its $k$ nearest coding vectors, so that the regions that have high Growth Error get more populated with coding vectors. Due to the stochastic sampling, the current $x_i$ and its neighboring data may not be sampled in the next iterations, thus the newly created coding vector may not be duly connected in subsequent repetitions of Step 1 and it may eventually drift away. To avoid this, at the current iteration, we add new edges from the newly added coding vectors to all neighbors of the $c_{i_k}$. The placement of new $y$ is conducted similarly, and by placing the $y$ close to the neighborhood of $y_{i_k}$, the convergence of the new $y$ to a suitable position is made faster, than by placing the new $y$ randomly on the output map. We summarize this step in Algorithm 4. These four steps form a complete iteration of SONG algorithm, which we summarize in Algorithm 4. In the next section, we evaluate the performance of the SONG algorithm.

III. EXPERIMENTS AND RESULTS

In this section, we compare SONG against Parametric t-SNE and non-parametric methods: t-SNE [6] and UMAP [7] on a series of data visualization tasks. First, we consider incremental data visualization with heterogeneous increments of data in Section III-A and homogeneous increments of data in Section III-B. It is noteworthy that the former is more likely the case to be assumed in real problems with incremental data.
Algorithm 1: SONG Algorithm with Decaying Learning Rate

1: \( t \leftarrow \text{iteration index, initialized as 0}; \)
2: \( t_{\text{max}} \leftarrow \text{maximum number of iterations}; \)
3: \( d \leftarrow \text{output dimensionality; usually 2 or 3}; \)
4: \( k \leftarrow \text{number of neighbors to consider at a given locality, } k \geq d + 1; \)
5: \( \alpha \leftarrow \text{learning rate starting at } \alpha_0; \)
6: \( C \leftarrow \text{random matrix of size } (d + 1) \times D; \)
7: \( E \leftarrow \text{edges on } C, \text{from each } c \text{ to other } c_s, \text{all initialized as non-edges (0)}; \)
8: \( Y \leftarrow \text{random matrix of size } (d + 1) \times d; \)
9: \( r \leftarrow \text{number of negative edges to select, per positive edge for negative sampling}; \)
10: \( a, b \leftarrow \text{appropriate parameters to get desired spread and tightness as per Eq. (6)}; \)
11: \( \theta_y \leftarrow \text{user defined growth threshold}; \)
12: \( \text{while } t < t_{\text{max}} \text{ do} \)
13: \( \quad \text{for } x_i \in X \text{ do} \)
14: \( \quad \quad \text{Update } E_{x_i} \text{ as per Section II-A} \)
15: \( \quad \quad \text{Update } Y; \)
16: \( \quad \quad n_s = r \cdot n(c_{i_1}), \text{here } n(c_{i_1}) \text{ is the number of edges from or to } c_{i_1}; \)
17: \( \quad \quad \text{Record the neighbors of } c_{i_1} \text{ as } \mathcal{N}_{i_1}^{-1} = \{ j \mid E_{x_i}(i_1, j) > 0 \}; \)
18: \( \quad \quad \text{Perform Edge Curation as per Algorithm 2}; \)
19: \( \quad \quad \text{Record the new set of neighbors } \mathcal{N}_{i_1}^t; \)
20: \( \quad \quad \text{if } \mathcal{N}_{i_1}^t = \mathcal{N}_{i_1}^t; \)
21: \( \quad \quad \quad \text{End the execution of the algorithm and return;} \)
22: \( \quad \quad \text{end} \)
23: \( \quad \text{end} \)
24: \( \text{end} \)
25: \( \text{Grow } C \text{ and } Y \text{ as per Algorithm 3}; \)
26: \( \text{end} \)
27: \( t \leftarrow t + 1; \)
28: \( \alpha \leftarrow \alpha_0 \times (1 - \frac{t}{t_{\text{max}}}); \)
29: \( \text{end} \)

Algorithm 2: Updating the Directional Edges in E between Coding Vectors based on \( X \)

1: \( \text{for } j \in \mathcal{N}_i^k \text{ do} \)
2: \( \quad \text{if } ||x_i - c_j|| \leq ||x_i - c_k|| \text{ then} \)
3: \( \quad \quad \text{Renew edges as } E(i, j) = 1; \)
4: \( \quad \text{else} \)
5: \( \quad \quad \text{Decay edges as } E(i, j) \leftarrow \varepsilon \cdot E(i, j); \)
6: \( \quad \text{end} \)
7: \( \quad \text{if } E(i, j) < \epsilon_{\text{min}} \text{ then} \)
8: \( \quad \quad \text{Prune edges as } E(i, j) \leftarrow 0; \)
9: \( \text{end} \)
10: \( \text{end} \)

Algorithm 3: Topology Preservation of the Low-dimensional Points \( Y \)

1: /\* Organization of Local Neighborhood */\*
2: \( \text{for } j \in \mathcal{N}_i^t \text{ do} \)
3: \( \quad y_j \leftarrow y_j + \alpha \cdot (y_{i_1} - y_j) \cdot \frac{2a b \epsilon_{i_1} ||y_{i_1} - y_j||^2 - 1}{||y_{i_1} - y_j||^2}; \)
4: \( \quad \text{end} \)
5: /\* Negative Sampling for Repulsion */\*
6: \( \text{Select } n_s \text{ random samples } J = \{ j_1, ..., j_{n_s} \} \text{ with } E_s(i_1, j) = 0; \)
7: \( \text{for } j \in J \text{ do} \)
8: \( \quad y_j \leftarrow y_j - \alpha \cdot (y_{i_1} - y_j) \cdot \frac{2b}{||y_{i_1} - y_j||^2}; \)
9: \( \text{end} \)

model-reinitializing methods. However, for fair comparison, we introduce a model-reinitializing version of SONG called SONG + Reinit. The 'incremental visualizations' can only be fairly assessed with model-retaining methods, but for the sake of completeness, we extend this comparison to the model-reinitializing methods as well.

We use the hyper-parameters in Table I for each method. For t-SNE [6] and UMAP [7], the recommended hyper-parameters in original papers were used as we did not observe any improvement in results by tuning these parameters. Similarly, for parametric t-SNE, we used the set of parameters provided by the GitHub implementation. The tuned hyper-parameters for SONG are noted in Table II.

Algorithm 4: Growing C and Y to Refine the Inferred Topology

1: \( \text{create new coding vector such that } w_n \leftarrow \frac{1}{k} \sum_{k=1}^{k} w_{i_k}; \)
2: \( \text{create new low-dimensional vector such that } y_n \leftarrow \frac{1}{k} \sum_{k=1}^{k} y_{i_k}; \)
3: \( \text{for } j \in \{ t_1, ..., t_k \} \text{ do} \)
4: \( \quad E(j, n) = 1; \)
5: \( \text{end} \)

1: https://github.com/jsilter/parametric_t-sne
A. Visualization of Data with Heterogenous Increments

We first evaluate SONG presented with heterogeneous increments, where new clusters or classes may be added to the existing datasets.

**Setup:** Three datasets are used: the Wong dataset [22], MNIST hand-written digit dataset [23] and the Fashion MNIST dataset [24]. Wong dataset has over 327k single human T-cells measured for expression levels for 39 different surface markers (i.e., 39 dimensions) such as CCR7 surface marker. There are many types of cells present in this dataset, such as lymphoid cells, naive T-cells, B-Cell Follicles, NK T cells etc, which we expect to be clustered separately. However, there may be some cell types that cannot be clearly separated as clusters in visualizations [25]. Since we have no ground-truth labels and UMAP provides superior qualitative cluster separation on this dataset [25], we assume that the clusters visible in the UMAP visualization of the dataset represent different cell types. This assumption allows us to do a sampling of 20k, 50k, 100k and 327k samples such that each time we add one or several cell types to the data. However, due to the lack of ground truth, we only conduct qualitative analysis on the cluster quality for each method. In addition, we conduct ‘logicle transformation’ [26] to normalize the Wong dataset as a preprocessing step. On the other hand, Fashion MNIST dataset is a collection of 60k images of fashion items belonging to 10 classes, each image having 28 x 28 pixels, therefore 784 pixel intensity levels (dimensions), associated with a known ground-truth label. Similar to Fashion - MNIST dataset, MNIST dataset is a collection of 60k images of hand written digits, each with 784 pixels and an associated label from 0 to 9. Since both MNIST and Fashion MNIST datasets have known ground-truth labels, we start with two randomly selected classes and present two more classes to the algorithm at each increment. We ran a K-Means clustering on the visualizations provided for MNIST and Fashion MNIST by each method, and calculated the Adjusted Mutual Information (AMI) [27] scores against the ground-truth labels. The AMI scores were averaged over five iterations with random initializations. Additionally, both MNIST and Fashion MNIST datasets are reduced to 20 dimensions using Principal Components Analysis (PCA) as a preprocessing step in order to reduce the running time of our experiments. We assume that the first 20 principle components capture most of the variance in the datasets [28]. Each of the intermediate and incrementally growing datasets of Wong, MNIST and Fashion MNIST datasets is visualized using SONG, SONG + Reinit, Parametric t-SNE, t-SNE and UMAP.

**Results:** For all three datasets: Wong (Fig. 2), Fashion MNIST (Fig. 3) and MNIST (Fig. 4), SONG shows high level of cluster mixing in visualizations, which becomes more evident in later increments. Although UMAP shows similar relative placement of clusters at later increments for the MNIST and Fashion MNIST datasets, arbitrary rotations of the complete map are visible even for such visualizations. This may be due to UMAP using Spectral Embedding as the heuristic initialization instead of random initialization. Table II summarizes the AMI Scores for the Fashion MNIST and MNIST datasets in heterogeneous incremental visualization scenarios. Note that in Table II, we have highlighted the best scores for each increment in the model-retaining methods. Since for the model-reinitializing methods such incremental visualization is not directly comparable with the model-retaining methods, we have only highlighted the winner for the complete dataset out of the model-reinitializing methods. In Table II, SONG provides superior cluster purity than Parametric t-SNE, confirming our observations on the level of cluster mixing present in visualizations by Parametric t-SNE. Here, SONG shows an average improvement of 14.98% for Fashion MNIST and 49.73% for MNIST in AMI compared to Parametric t-SNE. We observe that out of the non-parametric algorithms, SONG + Reinit is comparable but slightly inferior to UMAP, and superior to t-SNE.

For the Wong dataset (see Fig. 2), SONG + Reinit, Parametric t-SNE, t-SNE and UMAP all have drastic movement of clusters in consecutive visualizations. In t-SNE, we see a set

### Table I

| Algorithm          | Hyperparameters                               |
|--------------------|-----------------------------------------------|
| SONG/SONG-Reinit   | \( k = 2, \, t_{max} = 100, \alpha_0 = 1.0, \alpha = 1.577, \, b = 0.895 \) |
| Parametric t-SNE   | Perplexity = 30, epochs = 400, batch_size = 128 |
| t-SNE              | Perplexity = 30                                |
| UMAP               | \( n_{neighbors} = 15, \alpha_0 = 1.0, \alpha = 1.577, \, b = 0.895 \) |

### Table II

| No. Classes | Fashion MNIST | MNIST |
|-------------|---------------|-------|
|             | SONG          | 70.9  | 86.1  | 84.9  | 71.2  | 61.5  | 88.4  | 88.0  | 79.2  | 75.0  | 81.0  |
|             | Parametric t-SNE | 50.3  | 80.6  | 76.1  | 60.2  | 57.8  | 39.2  | 58.8  | 60.8  | 56.8  | 59.3  |
|             | SONG + Reinit | 70.9  | 76.8  | 78.4  | 69.1  | 61.0  | 88.4  | 86.4  | 77.8  | 75.3  | 81.0  |
|             | t-SNE         | 14.2  | 56.0  | 59.9  | 57.3  | 56.3  | 89.3  | 67.0  | 72.1  | 71.2  | 73.8  |
|             | UMAP          | 25.2  | 77.3  | 79.7  | 67.5  | 59.1  | 92.2  | 92.0  | 81.8  | 81.8  | 84.9  |

#### SONG/SONG-Reinit

- **Algorithm:** SONG/SONG-Reinit
- **Hyperparameters:** \( k = 2, \, t_{max} = 100, \alpha_0 = 1.0, \alpha = 1.577, \, b = 0.895 \)
- **Experiments:** Performed on Wong, MNIST and Fashion MNIST datasets to evaluate visualizations with heterogeneous increments.
Fig. 2. The Wong dataset visualized by SONG, SONG + Reinit, Parametric t-SNE and UMAP. The colors represent the CCR7 expression levels following the visualizations provided in [25], where Light Green represents high CCR7 expression and Dark Purple represents low CCR7 expressions.
of Gaussian blobs (possibly due to the Gaussian distribution assumption), with no discernible structure of cluster placement as visible in SONG and UMAP. Parametric t-SNE shows stable placement of clusters in the first two visualizations. However, when more data are presented, we see a high level of mixed clusters in the visualization.

In the Fashion MNIST visualizations (Fig. 3), we see drastic re-arrangement of placement when using SONG + Reinit, Parametric t-SNE , t-SNE and UMAP. We emphasize that SONG does not show rotations, as seen in visualizations provided by UMAP.

In Fig. 4, the hierarchy of clusters is more preserved in SONG and UMAP than t-SNE and Parametric t-SNE for the MNIST dataset. We expect in low-dimensional embedding space, the distances between clusters should vary as not all pairs of clusters are equally similar to each other, e.g., “1” should be more similar to “7” than to “3” or to “5”. In the results of t-SNE, however, the clusters are separated by similar distances, thereby the results do not provide information about the varying degrees of similarity between clusters. For both UMAP and SONG, the distances separating the clusters vary as expected. While parametric t-SNE shows similar placement of clusters with rotations or flips in the last two visualizations,
the level of cluster mixing is relatively high.

B. Visualization of Data with Homogeneous Increments

In this section, we further examine how each method performs when the incrementally added data proportionally represent all classes and clusters, using the same three datasets: Wong, Fashion-MNIST and MNIST.

Setup: For Wong dataset, we pick four different numbers of random samples, 10k, 20k, 50k and 327k. We select these numbers to investigate if the incremental inference of topology can be achieved starting from a small number of samples. For Fashion MNIST and MNIST datasets, we randomly sample four batches of data: 12k, 24k, 48k and 60k images at a time. Since Fashion-MNIST and MNIST have known ground-truths, for each visualization, we again conduct a k-means clustering to investigate the separability of clusters in the visualization, and use AMI to evaluate the clusters quality.

Furthermore, for Fashion MNIST and MNIST datasets, we develop a metric called the consecutive displacement of \( Y \) (CDY), to quantify the preservation of cluster placement in two consecutive incremental visualizations. CDY is defined as follows. Initially, we apply each algorithm to 6000 randomly sampled images, and iteratively add 6000 more to the existing
visualization, until we have presented all images in a dataset. At the \( t \)-th iteration, we record the visualizations of the existing data (without the newly added data) before and after the training with the 6000 new images, namely \( Y^{(t-1)} \) and \( Y^{(t)} \). Next we calculate the CDY of a point \( y_i \) in the existing visualization \( Y^{(t-1)} \) as:

\[
CDY(y_i) = \|y_i^{(t)} - y_i^{(t-1)}\|
\]

We record the average and standard deviations of CDY calculated for all points in the visualization. We note that the lack of a ground-truth which gives us information about an accurate placement of clusters renders a similar analysis for the Wong dataset prohibitive.

**Result:** Among compared methods, SONG shows the highest stability in cluster placement when new data are presented, as shown in Fig. 5 for the Wong dataset, and Fig. 6 for the MNIST and Fashion-MNIST datasets. Furthermore, SONG shows good quality in the clusters inferred in the output embedding as per Table III. In Table III, we have highlighted the best scores for each increment in the model-retaining methods. However, for the model-reinitializing methods, we have highlighted the winner for the complete dataset. Compared to Parametric t-SNE, SONG has an average improvement of accuracy by 8.36% on Fashion MNIST and 42.26% on MNIST. Out of three model-reinitialized methods, UMAP has similar placements of clusters in consecutive visualizations, but shows complete rotations in early to mid intermediate representations on the Wong dataset. UMAP stabilizes towards the last increments. However, for the three datasets, this stabilization happens at different stages. t-SNE shows arbitrary placement of clusters at each intermediate representation, making t-SNE not as good as UMAP or SONG for incremental visualizations.

**C. Tolerance to Noisy and Highly Mixed Clusters**

We explore how well SONG performs in the presence of noisy data, which we simulated as a series of datasets with high levels of cluster mixing and large cluster standard deviations.

**Setup:** We compare SONG against UMAP and t-SNE on a collection of 32 randomly generated Gaussian Blobs datasets using 8 different cluster standard deviations (4, 8, 10, 12, 14, 16, 18, 20) and 4 different numbers of clusters (10, 20, 50, 100) for each standard deviation. These datasets have a dimensionality of 60. In addition, for each algorithm, we calculate the Adjusted Mutual Information (AMI) score for the visualizations provided against the known labels of the Gaussian clusters.

|               | Fashion MNIST | MNIST       |
|---------------|--------------|-------------|
|               | 12k 24k 48k 60k | 12k 24k 48k 60k |
| SONG          | 59.6 60.1 58.1 59.5 | 74.8 79.4 80.7 81.9 |
| Parametric t-SNE | 51.2 55.8 57.2 54.8 | 44.9 53.3 57 61.2 |
| SONG-Init     | 59.6 58.8 61.2 61 | 74.8 79.8 80.9 84 |
| t-SNE         | 58.9 58 59.4 53.5 | 77.5 74.1 78.6 77.4 |
| UMAP          | 60.1 59.5 58.7 59 | 76.6 80.8 83.2 84.9 |

Table III shows that the AMI scores for SONG on the MNIST dataset have increased as we present more data to the SONG algorithm. For the Fashion MNIST dataset, the AMI scores for SONG remain relatively low throughout the increments. This difference of trends may be due to the higher level of mixing of classes in the Fashion MNIST dataset than the MNIST, which makes it more difficult to separate the classes in Fashion MNIST into distinct clusters despite having more data. Table III further shows that SONG provides visualizations of comparable quality to UMAP and superior to t-SNE and parametric t-SNE. We note that SONG generally produces lower AMIs than SONG + Reinit, possibly because that SONG attempts to preserve the placement of points in existing visualizations which may cause some structural changes caused by new data to be neglected. Neglecting such changes may explain the slight drop of performance in the incremental visualization vs the visualization of the complete dataset. However, we see that the incremental scores of SONG are not considerably worse than that of SONG + Reinit where SONG is trained on existing data and increment from scratch.

In Fig. 6, SONG has the lowest CDY values for both MNIST and Fashion MNIST, throughout the increments. SONG also shows small standard deviations, showing that the CDYs for all points are indeed limited. In contrast, t-SNE has the largest displacements and standard deviations. Surprisingly, the average CDYs for each increment in Parametric t-SNE is relatively higher than heuristically reinitialized UMAP. We note that parametric t-SNE has a low standard deviation of displacement compared to t-SNE. Giving a completely random re-arrangement of clusters would cause high standard deviation, this implies that parametric t-SNE produces translational or rotational displacements while keeping the cluster structure intact. Notably, UMAP does fairly well compared to other methods in terms of cluster displacement by having the second lowest average displacement and standard deviation. However, we see that in addition to having a smaller movement of points, SONG shows a strict decrease in displacement when more data are presented. SONG + Reinit, shows comparatively large average CDYs, as well as large standard deviations of CDYs, implying large movements between consecutive visualizations.

### Table III

| Method       | Fashion MNIST | MNIST       |
|--------------|--------------|-------------|
|              | 12k 24k 48k 60k | 12k 24k 48k 60k |
| SONG         | 59.6 60.1 58.1 59.5 | 74.8 79.4 80.7 81.9 |
| Parametric t-SNE | 51.2 55.8 57.2 54.8 | 44.9 53.3 57 61.2 |
| SONG-Init    | 59.6 58.8 61.2 61 | 74.8 79.8 80.9 84 |
| t-SNE        | 58.9 58 59.4 53.5 | 77.5 74.1 78.6 77.4 |
| UMAP         | 60.1 59.5 58.7 59 | 76.6 80.8 83.2 84.9 |
**Result:** Table IV shows that SONG has the highest accuracy for discerning mixing clusters in all cases. The resulting visualizations for one of these datasets for SONG, UMAP and t-SNE are provided in Fig.8. We observe that the cluster representations in the visualizations by SONG are more concentrated than that of both UMAP and t-SNE.

We refer to the Supplement Section 3.1 for an extended set of visualizations, where we additionally changed the dimensionality to observe how it affects these observations. In this extended study we test the visualization performance of the three algorithms on an additional 125 datasets. These datasets are generated by simulating datasets corresponding to 5 cluster standard deviations (1, 2, 3, 4, 10), 5 numbers of clusters (3, 4, 20, 50, 100) and 5 numbers of dimensions (3, 15, 45, 60, 120). In these visualizations, consistent to our observations in Table IV we observe that SONG has better separation of clusters while UMAP and t-SNE show fuzzy cluster boundaries when the level of cluster mixing increases.

### D. Qualitative Topology Preservation of SONG

To qualitatively examine the capability of SONG to preserve specific topologies in the input data, we used the COIL-20 dataset [29], which is frequently used to test the topology preservation of visualization methods [6][7].

**Setup:** We compare SONG with UMAP and t-SNE on visualizing the COIL-20 dataset, which has 20 different objects, each photographed at pose intervals of 5-degrees, resulting in 1440 images in total. Each image has 4096 pixels. As preprocessing, we reduced the COIL-20 dataset down to its first 300 principle components. Because of the rotating pose angles, we expect to see 20 circular clusters in our visualization, where each cluster represents a different object.

**Result:** The separation of circular clusters in SONG is similar to UMAP as shown in Fig. 8. For highly inseparable clusters, SONG and UMAP preserves the circular topologies better than t-SNE, where t-SNE shows an arch-like shape instead of circular structures.

### E. Running Time Comparison with t-SNE and UMAP

In our Supplement Section 3.2, we show that SONG is faster for the same dataset configurations than t-SNE. However, as SONG needs to recalculate the pairwise distances between two sets of high-dimensional vectors (X and C) multiple times for self organization, SONG has a performance bottleneck which renders it slower than UMAP.

### IV. Conclusions and Future Work

In this work, we have presented a parametric nonlinear dimensionality reduction method that can provide topology-preserving visualizations of high-dimensional data, while allowing new data to be mapped into existing visualizations without complete reinitialization. In our experiments, we presented SONG with both heterogeneous (Section III-A) and homogeneous (Section III-B) data increments, and observed that in both cases, SONG is superior than parametric t-SNE in preserving cluster placements when incorporating new data. We further observed in the aforementioned Sections that the visualizations provided by SONG has cluster quality on par with UMAP, and superior to non-parametric t-SNE. We also conclude that SONG is robust to noisy and highly mixed clusters (Section III-C), and that SONG is capable of preserving specific topologies (Section III-D) inferred from the input.

In addition, SONG has a few merits compared to existing methods. First of all, although originated from SOM, SONG is a general dimensionality reduction method. SOM relies on a uniform grid of 2 or 3 dimensions to obtain the self organization of coding vectors, which renders SOM with limited utility in providing representations of more than 3 dimensions. Since SONG uses a generic graph independent of output dimensionality, SONG can be a viable preprocessing technique to reduce the dimensionality down to more than three dimensions.

Second, SONG is especially useful for visualization of large datasets where considerable heterogeneity is present. This heterogeneity may be the product of undesired batch effects [30], or genuine variation of populations of the data. Consequently, SONG may be a promising tool for large-scale
benchmarking projects that require coordination and curation of highly heterogeneous data, such as the Human Cell Atlas [31].

However, SONG has a few limitations to be addressed in future work. First, SONG has a higher computational complexity than UMAP because in SONG, the high-dimensional parametric graph in the input space need to be recalculated several times, whereas in UMAP, the high-dimensional KNN graph is constructed only once. The number of recalculations required to obtain a stable graph influences how fast SONG works. We have empirically determined that 8-10 recalculations of the graph is sufficient to provide a comparable approximation with UMAP. One possible direction of minimizing this graph reconstruction bottle-neck is to use batch gradient descent instead of stochastic gradient descent at later stages of learning. In our implementation, we chose stochastic gradient descent in an attempt to obtain an optimal visualization quality as the batch versions of Self Organization algorithms are prone to sub-optimal solutions [20]. This may be viable at later stages when the graph is relatively stable and unchanging compared to the earlier stages. It should be noted that SONG still is less complex than t-SNE because SONG uses a negative sampling trick where we do not compute pairwise embedding distances globally (see Supplement Section 3.2).

Second, the current version of SONG cannot adjust the trade-off between two aspects of cluster placement: 1) preserving the already inferred topological representations and 2) adapting to represent new data which potentially alter the existing topology in high-dimensional space. This trade-off is indicated by the discrepancy between the AMI scores of SONG and SONG + Reinit calculated on the incremental data visualizations. Besides, in the Wong dataset (Fig. 2 and Fig. 3), the topology of the visualizations obtained using SONG shows differences from that using SONG + Reinit, which may be explained by SONG’s preference to preserve the topology in existing visualizations. Future work would explore the introduction of an ‘agility’ parameter that can regulate the aforementioned trade-off.

At last, throughout our manuscript, we discuss SONG as an unsupervised learning method. Another research direction is to incorporate known or partially known labels of data to enhance cluster quality and separation of clusters.

ACKNOWLEDGEMENTS

The authors thank Mr. Rajith Vidanaarachchi and Ms. Tamasha Malepathirana for proof-reading the manuscript. This work is partially funded by ARC DP150103512.

REFERENCES

[1] Emma Pierson and Christopher Yau. Zifa: Dimensionality reduction for zero-inflated single-cell gene expression analysis. Genome biology, 16(1):241, 2015.
[2] Stephen B Montgomery, Micha Sammeth, Maria Gutierrez-Arcelus, Radoslaw P Lach, Catherine Ingle, James Nisbett, Roderic Guigo, and Emmanouil T Dermitzakis. Transcriptome genetics using second generation sequencing in a caucasian population. Nature, 464(7289):773, 2010.
[3] Daniel Fisher, Igor Popov, Steven Drucker, et al. Trust me, i’m partially right: incremental visualization lets analysts explore large datasets faster. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 1673–1682. ACM, 2012.
[4] Lorena Montoya. Geo-data acquisition through mobile gis and digital video: an urban disaster management perspective. Environmental Modelling & Software, 18(10):869–876, 2003.
[5] Heng Zhang, Benyu Zhang, Teresa Mah, Dong Zhuang, Jeremy Tantrum, and Ying Li. Visualization application for mining of social networks, May 1 2008. US Patent App. 11/555,279.
[6] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(Nov):2579–2605, 2008.
[7] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.
[8] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation, 15(6):1373–1396, 2003.
[9] Jiawei Ding, Anne Condon, and Sobrah P Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. Nature communications, 9(1):2002, 2018.
[10] Laurens Van Der Maaten. Learning a parametric embedding by preserving local structure. In Artificial Intelligence and Statistics, pages 384–391, 2009.
[11] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530, 2016.
[12] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In Advances in neural information processing systems, pages 2377–2385, 2015.
[13] Irene Sturm, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Interpretable deep neural networks for single-trial eeg classification. Journal of neuroscience methods, 274:141–145, 2016.
[14] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. Biological cybernetics, 43(1):59–69, 1982.
[15] Bernd Fritzke. Growing cell structuresa self-organizing network for unsupervised and supervised learning. Neural networks, 7(9):1441–1460, 1994.
[16] Bernd Fritzke. A growing neural gas network learns topologies. In Advances in neural information processing systems, pages 625–632, 1995.
[17] Pablo A Estévez, Cristián J Figueroa, and Kazimio Saito. Cross-entropy approach to data visualization based on the neural gas network. In Proceedings, 2005 IEEE International Joint Conference on Neural Networks, 2005., volume 5, pages 2724–2729. IEEE, 2005.
[18] Damdina Alahakoon, Saman K Halgamuge, and Bala Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Transactions on neural networks, 13(3):601–614, 2000.
[19] James Munkres. Topology. Pearson Education, 2014.
[20] Jean-Claude Fort, Patrick Letremy, and Marie Cottrell. Advantages and drawbacks of the batch kohonen algorithm. In ESANN, volume 2, pages 223–230, 2002.
[21] Thomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.
[22] Michael Thomas Wong, David Eng Hui Ong, Frances Sheau Huei Lim, Karen Wei Weng Teng, Naomi McGovern, Sriman Narayanan, Wen Qi Ho, Daniela Cerny, Henry Kun Kiaan Tang, Rosslyn Anicete, et al. A high-dimensional atlas of human t cell diversity reveals tissue-specific trafficking and cytokine signatures. Immunity, 45(2):442–456, 2016.
[23] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2:18, 2010.
[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.
[25] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immmanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Dimensionality reduction for visualizing single-cell data using umap. Nature biotechnology, 37(1):38, 2019.
[26] David R Parks, Mario Roederer, and Wayne A Moore. A new logicle display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. Cytometry Part A: The Journal of the International Society for Analytical Cytology, 69(6):541–551, 2006.
[27] Yuanjun Xuan Vinh, Julien Eppe, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In Proceedings of the 26th annual international conference on machine learning, pages 1073–1080. ACM, 2009.
[28] LJ Cao, Kok Seng Chua, WK Chong, HP Lee, and QM Gu. A comparison of pca, kpca and ica for dimensionality reduction in support vector machine. Neurocomputing, 55(1-2):321–336, 2003.

[29] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.

[30] Luke C Gandolfo and Terence P Speed. Rle plots: Visualizing unwanted variation in high dimensional data. PloS one, 13(2):e0191629, 2018.

[31] Aviv Regev, Sarah A Teichmann, Eric S Lander, Ido Amit, Christophe Benoist, Ewan Birney, Bernd Bodenmiller, Peter Campbell, Piero Carninci, Menna Clatworthy, et al. Science forum: the human cell atlas. Elife, 6:e27041, 2017.

Damith Senanayake Damith Senanayake is a PhD candidate at the University of Melbourne. He completed his undergraduate degree in Computer Science and Engineering from the University of Moratuwa, Sri Lanka in 2015. Currently his research research focuses on developing analysis methods for highly dimensional data using neural networks and computational and algebraic geometry and topology. He further focuses on applications in bioinformatics.

Wang Wei, PhD Wei Wang received the BE degree in Mechatronic Engineering from Zhejiang University in 2013 and PhD degree at the Department of Mechanical Engineering, University of Melbourne in 2019. His research has been focused on deep generative models, Bayesian statistics and image processing, with a current interest in exploring compact and highly performant deep neural network structures.

Shalin H. Naik, PhD Shalin Naik received his BSc (Hons) from the University of QLD, Department of Microbiology and Immunology, his PhD from the University of Melbourne, Department of Medical Biology and The Walter & Eliza Hall Institute of Medical Research (WEHI), and did his postdoc at the Netherlands Cancer Institute. He is a group leader at WEHI with a research interest in clonal biology in the areas of haematopoiesis, dendritic cells, stem cells and cancer evolution.

Prof. Saman Halgamuge Prof. Saman Halgamuge (IEEE Fellow) is a Professor in the Department of Mechanical Engineering. He was previously the Director of Research School of Engineering at the Australian National University. He was a member of Australian Research Council (ARC) College of Experts for Engineering, Information and Computing Sciences. His research interests are in Machine Learning including Deep Learning, Big Data Analytics and Optimization. These applications vary from Sensor Networks in Irrigation, Smart Grids, and Sustainable Energy generation to Bioinformatics and Neuro-Engineering. His fundamental research contributions are in Big Data Analytics with Unsupervised and Near Unsupervised type learning and Bioinspired Optimization.
Fig. 5. Random samples of varying sizes from the Wong dataset presented to SONG, Parametric t-SNE, SONG + Reinit, UMAP and t-SNE incrementally.
Fig. 6. The Euclidean Displacement of each established point after subsequent presentation of 6000 images to each algorithm.

(a) Fashion MNIST

(b) MNIST
Fig. 7. The visualizations of a dataset having 100 clusters, each cluster having a standard deviation of 10, and 60 dimensions using the three methods SONG, UMAP and t-SNE.

Fig. 8. COIL-20 dataset when reduced using the three algorithms a) t-SNE, b) UMAP and c) SONG. Both UMAP and SONG preserve the circular topologies, even in clusters where the classes are not well separated, to a greater degree than t-SNE.