

Exploratory Analysis of High Dimensional Data: Parametric Methods for Dimensionality Reduction, Visualization and Feature Extraction with Applications in Computational Biology

by

Damith A. Senanayake

[ORCID: 0000-0003-1233-4844](#)

A thesis submitted in total fulfillment for the
degree of Doctor of Philosophy

in the
Department of Mechanical Engineering
Melbourne School of Engineering
THE UNIVERSITY OF MELBOURNE

May 2020

THE UNIVERSITY OF MELBOURNE

Abstract

Department of Mechanical Engineering
Melbourne School of Engineering

Doctor of Philosophy

by [Damith A. Senanayake](#)
[ORCID: 0000-0003-1233-4844](#)

Recent advances in experimental technologies have facilitated the gathering of data with thousands of variables. Because of this, modern data analysis tasks often encounter high dimensional data, which are challenging to analyse. Such analysis is made more difficult with the lack of ground-truth. In this thesis, I have explored two aspects of high-dimensional exploratory data analysis: 1) Dimensionality Reduction and Visualization of high-dimensional data to gain insights into the structure of the data and, 2) Extraction and interpretation of feature subsets (motifs) which explain the structure of high-dimensional data. I have presented methods that are built on concepts of Neural Networks - a powerful modern branch of optimization techniques. Through my work, I have shown that using relatively infrequently used variants of neural networks and building on their concepts (e.g. vector quantization and Hebbian Learning) we can produce powerful dimensionality reduction methods that address major gap areas of research. I have further shown that using unsupervised deep neural networks with contextual regularization, we can produce a framework to extract minimal motifs that optimally reconstruct complex nonlinear structures present in high dimensional data.

Declaration of Authorship

I, DAMITH ASANKA SENANAYAKE, declare that this thesis titled, ‘Exploratory Analysis of Highly Dimensional Data: Parametric Methods for Dimensionality Reduction, Visualization and Feature Extraction with Applications in Computational Biology’ and the work presented in it are my own. I confirm that:

- The thesis comprises only my original work towards the PhD Engineering except where indicated in the preface;
- due acknowledgement has been made in the text to all other material used; and
- the thesis is fewer than the maximum word limit in length, exclusive of tables, maps, bibliographies and appendices as approved by the Research Higher Degrees Committee.

Signed:



Date: 18th of May 2020

Preface

The work done towards the completion of this thesis have been compiled as four publications, each addressing a specific research question in my research. The status of the publications currently included in this thesis is as follows.

1. Research Question 1: **Gap Encoding Growing Self Organizing maps for Dimensionality Reduction** - Unpublished material not submitted for publication. To be submitted to International Conference on Machine Learning 2020 (Vienna, Austria).
Algorithm development, programming and packaging : Damith Senanayake
Manuscript Preparation : Damith Senanayake
Manuscript Revisions and Proof Reading : Chalini Wijetunge, Malin Prematilake and Saman Halgamuge.
2. Research Question 2: **Self Organizing Nebulous Growths for Incremental Visualization of High Dimensional Data** - Accepted with minor revisions for publication in IEEE Transactions. Neural Networks and Learning Systems, currently published on arXiv.org.
Algorithm Development, Programming, Packaging : Damith Senanayake
Manuscript Preparation : Damith Senanayake, Wei Wang
Manuscript Revisions and Proof Reading : Shalin H. Naik, Saman Halgamuge.
3. Research Question 3: **SONG allows an evolving single cell atlas with landmarks and spatially persistent visualizations** - Unpublished material not submitted for publication. In progress to be submitted to Nature Biotechnology.
Experiment Design : Damith Senanayake, Luyi Tian, Shalin H. Naik
Manuscript Preparation and Proof Reading : Damith Senanayake, Shalin H. Naik, Luyi Tian, Saman Halgamuge.
4. Research Question 4: **Unsupervised Deep Neural Networks extract the ‘Genetic Penstrokes’ of heterogeneous cellular populations with minimal assumptions and without a priori granularization** - Unpublished material not submitted for publication. To be submitted to international Conference on Bioinformatics 2020.
Experiment Design : Damith Senanayake Manuscript Preparation and Proof Reading: Damith Senanayake, Saman Halgamuge.

Funding Sources:

- University of Melbourne: Melbourne International Research Scholarship
- Australian Research Council: Discovery Project 150103512

Acknowledgements

This work has been in the brewing throughout the most exciting years of my life, and it has been a journey worth every struggle and challenge I faced. The years I spent on this research and degree has been naught short of formative of my intellect, my rigor, and my outlook on not just the academic premises of my work, but rather the world around me as a whole. While it was an empowering journey, I would be remiss if I did not say it was a humbling experience. Therefore, I would like to experience my humble gratitude to all who helped me to this work.

First, I thank all my teachers, who not only instilled the value of intellectual endeavors, but always encouraged me to question the perceived realities, not only at school, but also in my cultural worldview as well. I will not have enough space to thank them all by name, but I must thank specifically Ven. Vitiyala Indasiri Thero, for his encouragement of my inquisitive nature from early childhood. I must also mention my Chemistry and Physics teachers during my final years in school who truly instilled the methodical approach to science in me. In my undergraduate studies, few has had an impact on truly rigorous learning as Dr. Chandana Gamage, whose guidance I am eternally thankful for. I would also like to specially thank Mrs. Vishakha Nanayakkara, and Dr. Shehan Perera for their encouragement to pursue a career in higher studies, but I would be forgoing a great duty if I did not thank each and every staff member of the department of Computer Science and Engineering of the University of Moratuwa.

My deepest gratitude is however, reserved for those around me. My friends, especially Kalpa, a friend as true as whom has never been, and all other friends in my closest circle who have urged me on to do my best. Those in the Optimization and Pattern Recognition Laboratory, both present and past, who have been mentors and more than just colleagues in this inquisitive journey. My brother, who is ever the stickler, has never failed to keep me on my tracks, and has never let the passing of my father be a void of a role model in my life. My brother's family, who have always had a bed to sleep and a plate at the table for me. My wife, who loves me and nurtures me more than I could ever wish or hope for, who was the light that led me ashore in the recent moments of darkness. My aunt and uncle, who facilitated my early education as though I was one of their own also deserves special thanks for nurturing and moulding me into who I am. And above all, my mother, who brought me life, more than once, and even fought for her own without me by her side so that I could do my best...

I would like to thank all my mentors, Dr. Isaam Saeed, who showed me the very first paths into my research, Dr. Chalini Wijetunge, who has been a great mentor, both in terms of technical and personal tedium in the PhD student's life, Dr. Sen-Lin Tang,

who has been a great wealth of guidance in matters of biology, and Dr. Shalin Naik, the acquaintance of whom is one of the biggest fortunes of my life. Last but not least, Prof. Saman Halgamuge, who undoubtedly had more influence in shaping my outlook on science than anyone else in my life.

I thank those who facilitated my career, the staff of the Department of Mechanical Engineering of University of Melbourne and the Naik Laboratory of the Walter Eliza Hall Institute for Medical Research.

Finally, my heartfelt gratitude to the people of Sri Lanka and Australia, who have provided for my education and health, and whose spirit of community may never wither.

Contents

Abstract	i
Declaration of Authorship	ii
Preface	iii
Acknowledgements	v
List of Figures	ix
1 Introduction and Literature Survey	1
1.1 Literature Review	1
1.1.1 High-Dimensional Data	1
1.1.2 Exploratory Analysis of High-dimensional Data	3
1.1.3 Dimensionality Reduction Techniques	4
1.1.3.1 Manifold Learning and Topological Manifolds	6
1.1.3.2 t-SNE for More Intuitively Separate Cluster Visualizations	15
1.1.3.3 UMAP for Robust Global Topology Preservation	16
1.1.4 Neural Networks	18
1.1.4.1 Feed Forward Neural Networks	20
1.1.4.2 The Backpropagation Algorithm	22
1.1.4.3 Pragmatics: Deep Networks and Regularization	24
1.1.5 Other Architectures: Self Organizing Feature Maps	25
1.1.5.1 Hebbian Learning and Hebb-like Neural Networks	27
1.2 Single Cell Data Analysis	28
1.2.1 Background and Motivation	29
1.2.1.1 Haematopoiesis and Cell Lineages	30
1.2.1.2 Differential Expression of Genes Across Heterogeneous Cell Populations	32
1.3 Motivation and Research Questions	33
1.3.1 Structure of Data vs Motifs of Features	33
2 How can the SOM algorithms be improved to represent non-uniform densities in the input space better?	36
2.1 Introduction	37

2.2	Summary and Contributions	75
2.3	Caveats	75
3	How can the SOM-based methods be used as a parametric alternative to modern robust nonlinear dimensionality reduction techniques?	76
3.1	Introduction	77
3.2	Summary and Contributions	120
4	How can the use of SONG algorithm improve the analysis of RNA-Seq Data	121
4.1	Introduction	122
4.2	Summary and Contributions	135
5	What would applying feature extraction capabilities of Neural Networks reveal of the underlying genetic dynamics in RNA-Seq Experiments?	136
5.1	Background	137
5.2	Summary and Contributions	152
6	Concluding Remarks	153
A	Appendix : ENVirT: inference of ecological characteristics of viruses from metagenomic data	157
	Bibliography	170

List of Figures

1.1	Reduction of the S-Curve data down to two dimensions. We see that while PCA captures the two linear combinations of the original data that maximizes the variance, it violates the local arrangement of data points. .	5
1.2	While PCA ‘pinches’ certain regions in the S-shaped sheet in order to retain the variance of the original data, MDS is more successful at maintaining relative distances in small neighborhoods. However, we still see that the insides of the bends are more compressed than the outsides. . . .	7
1.3	The variation of $\hat{d}$ with dimensionality. The $\hat{d}$ value decreases at an increasing rate as dimensionality increases.	8
1.4	Elbow point (green) shows the end of highly discernible neighbors. Beyond this point, the distances are not clearly separable as ‘close’ or ‘distant’	8
1.5	The LLE algorithm summarized. a) For each input, a group of closest neighbors are selected. b) The input is then reconstructed using a linear combination of its neighbors, c) the weights used for linear reconstruction is then used to embed the output coordinates [1].	10
1.6	How Isomap the k-NN Graph to reduce dimensionality. First a k-NN Graph is created on the discrete input set (a so-called S-Curve dataset). Next, the pairwise shortest path distances are calculated. These distances are preserved in the embedded graph in 2 dimensions. The distance between the red and blue input points are preserved along the edge of the S-shape rather than the straightline distance between them in the input space.	11
1.7	Different graph structures embedded in two dimensions using Laplacian Eigenmaps[2]. As we see, the visualizations have emphasized the ‘hole’ areas and ‘compact’ areas of the network.	12
1.8	The simulated data consists of five Gaussian clusters and one S curve. Note that PCA fails to capture the S-curve topology and complex distribution of clusters causes them to be mixed in the representation. IsoMap, LLE and (Laplacian) Eigen maps are sensitive to noise, hence the visible distortions.	13
1.9	Five different digits visualized from the UCI-Digits dataset using SNE. While the clusters are well organized, poor separation is apparent.	15
1.10	Five different digits from the UCI digits dataset visualized using t-SNE. While the clusters are well preserved, they are well separated, providing us more intuitive information about the structure of the data than SNE did.	16
1.11	Visualization of flow cytometry measurements obtained on a population of immune cells[3] (CC BY-NC 4.0)	17

1.12	An abstraction of a biological neuron compared to an artificial neuron based on [4]. A biological neuron gathers the synapses from neighboring connected neurons (shown as black arrows) and aggregates this signal and passes through the axon to the next neuron (blue arrow). The signals are not uniform due to physiological differences between each synapse. This is reflected in an artificial neuron as weighted inputs to the neuron (black) and summation, followed by a non-linear activation function.	19
1.13	A feedforward neural network with hidden layers (based on)[5]	21
1.14	Each square represents the input weights of a hidden neuron. Note that no proximity information has been used in the extraction of these features. The autoencoder has learned the 'edge representations' of the image set it was trained on.	25
1.15	Blue represents the input dataset. The grid represents the defined organization of coding vectors. The coding vectors are the nodes of the grid. (CC BY-SA 3.0)	27
1.16	Approximation of a complex nonlinear structure using GNG (figure based on [6]. The blue points represent the coding vectors, and the blue lines represent the edges connecting the coding vectors. We see that as training proceeds (from left to right), the number of vertices increase, approximating the density of the input space (space within the scaffold boundary).	28
1.17	Different Groups of PBMC cells visualized using t-SNE on a two-dimensional map [7] (CC BY 4.0).	31
1.18	Three different types of pseudotime trajectories are identified [8]. a) Cyclic pseudotime (where cells express genes in a repetitive manner), b) Bifurcating pseudotime, where the cells evolve in two different trajectories with a single origin, and c) Multifurcating pseudotime, which as multiple trajectories originating from a single origin (CC BY 4.0).	31
1.19	Heatmap visualizing a cluster analysis of a set of samples, and sets of genes with high differential expression [9]. Each cluster of samples (columns) have similarly expressed profiles of genes (rows). A hierarchy clusters can be inferred from the relative similarity of the gene expression profile of each sample pair (CC BY 4.0).	32
1.20	Visualizing the MNIST dataset using UMAP shows that similar digits are clustered together, and different clusters are separated in the visualization.	34

Chapter 1

Introduction and Literature Survey

Experimental validation of scientific claims and observations rely on the collection of ample data. All human knowledge is in fact, driven by the collection, analysis and interpretation of the said analysis of data gathered by empirical or non-empirical means. In my work, I build on the importance of data analysis in experimental studies, to investigate how extracting knowledge from vast amounts of data can be computationally automated. Specifically, my work focuses on what we call high-dimensional data, their gathering, their analysis, and their interpretation in a framework of topological and geometrical concepts, realized by the algorithmic implementation with the use of Neural Networks, a contemporary popular class of algorithms used in optimization and machine learning domains.

1.1 Literature Review

1.1.1 High-Dimensional Data

Bringing intuition into the domain of statistics, ‘data’ can be considered as measurements of various entities gathered at various scenarios. The sources of these measurements may vary in their rigor, from highly controlled experiments, to casual surveys or census of arbitrary characteristics. No matter the source, the gathering, curation, analysis and interpretation of data is a pivotal and vital part of our modern lives. Especially in experimental sciences, the quality of the outcomes of these experiments, rely heavily on the quality of the data involved. Be they claims about hitherto unknown phenomena as often encountered in fields like astronomy or physics, or groundbreaking discoveries of

new drugs for serious diseases or illnesses, the processes of gathering the data, processing the data and interpreting the data plays an inevitably important role.

A common reality in most instances where data analysis is involved, is that for each observation, one encounters multiple measurements. For instance, in census data gathering, for each individual or household, a number of demographic, financial or cultural measurements are accounted for. This is often the reality in most other fields. In experimental sciences this holds especially true as in a field where we grapple with the unknown, the gathering of as much information on the phenomena is necessary.

Here we encounter the concept of high-dimensional data (for which we may use the terms multi-dimensional, multivariate, or highly dimensional data interchangeably throughout this work). And with this increasing dimensionality, we encounter a new set of problems when analysing and interpreting such data. The most banal or ubiquitous of these problems arise when we conduct exploratory analysis where we cannot obtain any sense of the ‘structure’ of the data when the dimensionality exceeds 3 dimensions. While data often comes in various forms, such as numerical, categorical, binary, textual or time-series etc, we constrain our analysis in this work to only encompass numerical data. Our rationale for this restriction is fairly straightforward. We consider that most (if not all) other forms that are frequently encountered in reality can be converted into a numerical format. Categorical data can be converted into a numerical format by either employing label encoding or bitmap indexing (often called one-hot encoding in computational theory [10]), or even using a kernel operator or a kernel trick [11, 12]. Similarly textual or natural language data can also be converted into a numerical format using a kernel method. Furthermore, time-series or any time domain data can be efficiently converted into a numerical format using frequency domain transfers such as Fast Fourier Transform [13]. Acknowledging that these methods themselves may have their pitfalls and weaknesses, we consider addressing them beyond the scope of this work.

Furthermore, we define constraints on the type of analysis we consider in this work. Specifically, our main focus on this work is on the exploratory or unsupervised analysis of data. The reasoning behind this is that we consider the problem of unsupervised learning or exploratory analysis encompasses a broad range of applications, as most experimental studies often relies on unsupervised structural analysis to gain prior knowledge to help in final predictive tasks. In addition to this, we believe that not taking ground-truth for granted is a more reasonable stance as in most scientific domains, little or no prior information about the ground-truth or in common parlance ‘what we are looking for’ is available. Therefore we define a high-dimensional dataset to be a matrix of numbers $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $D \gg 2$. Additionally, a low-dimensional representation

of each of the data points in $\mathbf{X}$ is defined as $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ ($\mathbf{Y} \in \mathbb{R}^d$ and $d \ll D$, such that)there is a correspondence γ between $\mathbf{X}$ and $\mathbf{Y}$, such that,

$$\mathbf{y}_i = \gamma(\mathbf{x}_i) \quad (1.1)$$

We use this notation for high-dimensional data throughout our work.

1.1.2 Exploratory Analysis of High-dimensional Data

Exploratory Data Analysis has been in conversation for as long as data analysis has been, and the use of computational techniques for the analysis has been an immediate intuition [14]. In his well-known work (albeit introductory), Velleman [14] introduces some of the basic tenets of exploratory analysis, which I paraphrase here as:

- The importance of a *feel* (the intuitive philosophy) of a method outweighs the arithmetic tedium of a method
- The exploratory methods should be compatible with other established data analyses

Therefore, in our work, I too focus mainly on the philosophical underpinnings rather than minute algorithmic improvements one can do to improve the performance of methods, as well as build our work in context of existing analysis workflows.

In their seminal work, Friedman and Tukey [15] introduce important aspects of exploratory data analysis methods, which can be summarized as

- Intuitive explorability: Where *displays* visually exhibit important structural information of the data
- Residual explainability: Where a model captures the variance of data while accounting for deviations
- Resistance to aberrations: Where few unusual data observations do not skew the model biases unreasonably

As I focus mainly on the development of computational techniques, I try my best to adhere to these principles in developing such methods.

Although these concepts are for exploratory data analysis in general, they can be extended to high-dimensional data quite easily. We define the tasks of data analysis in our work in three main components or aspects.

- Visualization of high-dimensional data
- Feature extraction and prototyping of high-dimensional data
- Resistance to noise and tolerance through parametrization of inputs

The rest of this literature survey will be focused on these aspects of high-dimensional data analysis.

First, we consider the task of visualization of high-dimensional data. However, we note here that the task of visualizing high-dimensionality data falls under the broad category of Dimensionality Reduction of data.

1.1.3 Dimensionality Reduction Techniques

One straightforward work around of analyzing highly dimensional data is to simply reduce the dimensionality of the data. The most obvious method for this is to eliminate variables which are redundant (i.e. having patterns similar to other variables) or contain little information (i.e. low variance throughout observations). However, such elimination often requires pairwise tests of variables, and often relies on analyst intervention for selecting which features to eliminate (e.g. χ^2 tests).

Therefore analysts often employ more robust methods that convert highly dimensional data into a low dimensional representation by preserving the structure of the original (high dimensional) data. However, the definition of the preserved structure depends on the domain of application and the underpinnings of the reduction method.

Principal Component Analysis

Principal Component Analysis (PCA) [16] converts a data matrix $\mathbf{X} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^D\}$ with zero column-wise mean (i.e. each variable must be shifted to zero mean), into a low dimensional representation $\mathbf{T} = \{\mathbf{t}_i | \mathbf{t}_i \in \mathbb{R}^d\}$ where $d \ll D$ such that:

$$\mathbf{T} = \mathbf{X} \cdot \mathbf{W}^T \tag{1.2}$$

and $\mathbf{W}$ is a $d \times D$ matrix, defined such that $\mathbf{T}$ retains the maximum possible variance in $\mathbf{X}$.

Intuitively, what PCA does is that it converts a high dimensional input dataset, into a low dimensional representation that captures as much variance of the original data as possible, through a linear transform. Therefore, in the reduced representation $\mathbf{T}$, each column can be considered a linear combination of the columns in the original data $\mathbf{X}$.

PCA is a frequently used method due to its simplicity in assumptions. Finding of $\mathbf{W}$ can often be achieved by an eigen decomposition of the covariance matrix of $\mathbf{X}$, and can be calculated efficiently.

However, PCA also suffers in terms of its ability to find efficient reductions of the data because of the linear nature of its transformation. To demonstrate this, we use the “S Curve” dataset, where a planar distribution of data has been embedded in a three dimensional space with an “S” shape [17].

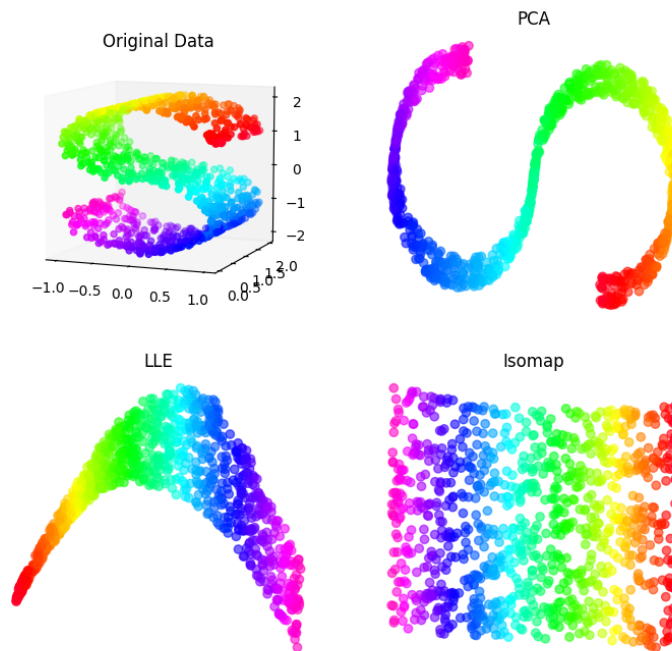


FIGURE 1.1: Reduction of the S-Curve data down to two dimensions. We see that while PCA captures the two linear combinations of the original data that maximizes the variance, it violates the local arrangement of data points.

In Figure 1.1, we show that PCA has captured two dimensions which maximizes the variance of the original data. However, due to the fact that they are linear combinations of the original data, there is dissarrangement of datapoints in the middle of the representation as the nonlinear structure of the S curve has been missed by these linear

combinations. We note that two other methods, LLE and IsoMap used here have captured both the global and the local structures well. We will discuss these methods later under nonlinear dimensionality reduction or manifold learning in Chapter 1.1.3.1.

While PCA is an efficient and useful dimensionality reduction algorithm, when it comes to visualizations, it often encounters the issues we described above. Therefore in order to visualize structures of the data, nonlinear dimensionality reduction methods are widely used. In the following sections, we delve deep into these nonlinear dimensionality reduction methods. However, we use PCA in our experiments to obtain compact representations of the data, but not for visualization.

Multidimensional Scaling

We observed that in an attempt to preserve the ‘structure’ of the data, PCA attempts to recreate the variance of the overall dataset, and how it fails to capture complex nonlinear structures in visualization. Multidimensional Scaling (MDS)[18] is one of the classical approaches where the structure of the data is defined in a more intuitive way.

The basic intuition is that in a low-dimensional representation, the pairwise distances between any two observations is to be preserved. This is obtained by minimizing a ‘stress’ function which can be defined as:

$$S(\mathbf{X}, \mathbf{Y}) = \sum_{\forall i \in \{1, \dots, |X|\}} \sum_{\forall j \in \{1, \dots, |X|\}} \left(\frac{d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{y}_i, \mathbf{y}_j)}{\sigma_{ij}} \right)^2 \quad (1.3)$$

where σ_{ij} is a local normalizing term, often taken to be equal to $d(\mathbf{x}_i, \mathbf{x}_j)$, where $d(a, b)$ is a distance metric. Note that in this case, we consider the case of metric multidimensional scaling. A more general version of classical multidimensional scaling[19] exists, and follows a similar principal.

1.1.3.1 Manifold Learning and Topological Manifolds

Although MDS escapes the linear constraints of PCA, it still suffers at the hand of highly complex structures present in the data [20]. We demonstrate this shortcoming in our Figure 1.2, which can be attributed to MDS attempting to preserve both global and local distances simultaneously. To elaborate this phenomenon, we must first examine the behavior of data and pairwise distances in higher dimensions.

The ‘curse of dimensionality’, first so dubbed by Bellman in his work [21] has a rigorous definition, and yet for convenience of our reading, we shall consider the more intuitive approach. Specifically, rather than considering a general distance function, let

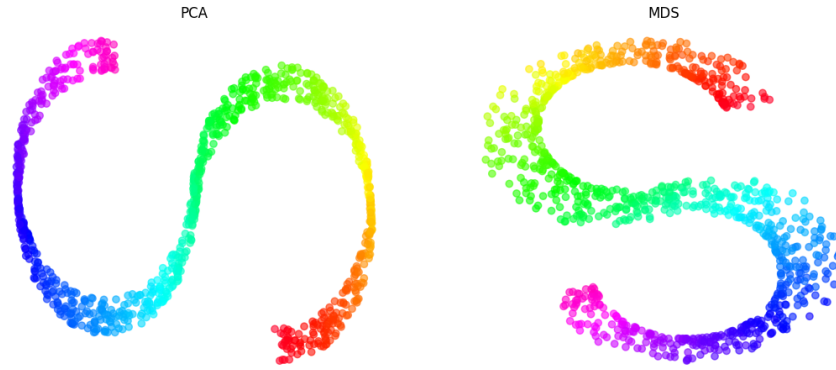


FIGURE 1.2: While PCA ‘pinches’ certain regions in the S-shaped sheet in order to retain the variance of the original data, MDS is more successful at maintaining relative distances in small neighborhoods. However, we still see that the insides of the bends are more compressed than the outsides.

us consider the Euclidean distance function, which is most widely used in calculating distances.

Consider a set of uniformly distributed data points, and select one of these points. When one considers the distance to all other points from this selected point as an indication of the ‘structure’ of the data, one may benefit more by considering the relative distance rather than the absolute distance. To this end, we normalize the distances as Beyer [22] did as follows:

$$\hat{d} = \frac{d_{max} - d_{min}}{d_{min}} \quad (1.4)$$

which can be considered as an indication of how discernible ‘close neighbors’ and ‘further neighbors’ are. They also note that as the dimensionality of the data increases, the following can be observed:

$$\lim_{D \rightarrow \infty} \hat{d} = 0 \quad (1.5)$$

i.e. the difference between close neighbors vs distant neighbors become indiscernible.

To demonstrate this, we generate datasets with 2000 random uniform data points in each, and then calculate the $\hat{d}$ for a random point in the dataset. We change the dimensionality of the dataset from 1 to 10000. Next, we normalize these $\hat{d}$ values (so that we have a reference of the scale) and plot the $\hat{d}$ values against the dimensionality.

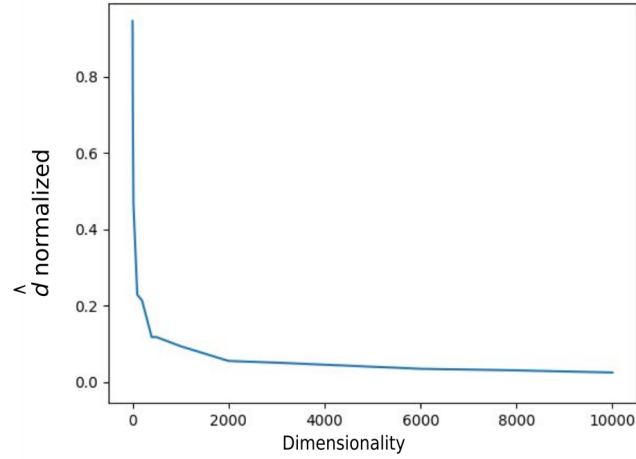


FIGURE 1.3: The variation of $\hat{d}$ with dimensionality. The $\hat{d}$ value decreases at an increasing rate as dimensionality increases.

As the results of this demonstration, we see that the discernibility between distant vs close points decreases as the dimensionality increases. However, we next consider for a given high dimensionality, how does the discernibility vary with distance itself: in short, what is the furthest point we can discern clearly as a ‘distant point’.

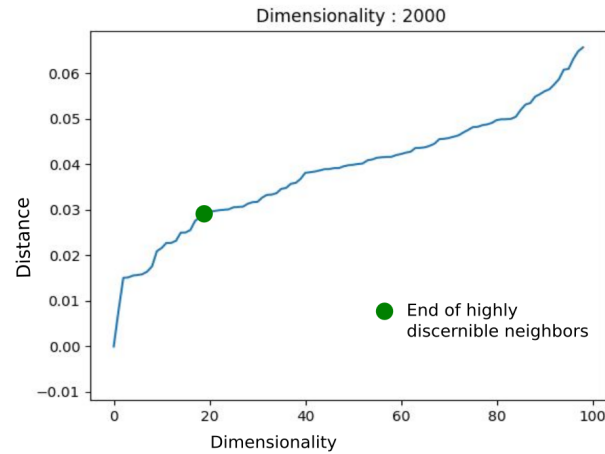


FIGURE 1.4: Elbow point (green) shows the end of highly discernible neighbors. Beyond this point, the distances are not clearly separable as ‘close’ or ‘distant’

As seen in Figure 1.4, even at a dimensionality as high as 2000 dimensions, the first few neighbors are highly discernible. This implies that meaningful structures can be made by considering the distance differences between the first few neighbors of each point. This can be considered the foundation of manifold learning. We note here that this entails the assumption that the data is distributed uniformly in the high-dimensional space.

While there are rigorous definitions for the term ‘manifold’ often used in computational geometry, and differential topology [23, 24], we follow the more intuitive understanding that a manifold is a space where small neighborhoods resembles a Euclidean space (i.e. displays Euclidean properties). A most apparent example is how a map of a suburb may represent a two dimensional Cartesian grid, on which all Euclidean distance calculations hold fast, and yet is in reality that suburb is a part of the spherical earth surface.

Relating this to our previous understanding of how the euclidean distances of high-dimensional spaces are useful in very small neighborhoods, it is a straightforward gap to bridge when we consider a complex highly dimensional dataset as a manifold which displays Euclidean properties in small neighborhoods. However, the collection of all these neighborhoods to form a singular data structure has not been discussed so far.

Topology[25] can be considered as the binding agent that holds the concepts of manifolds and high-dimensional data together. Although topology is a robust discipline in its own right, for our purposes we borrow some of the basic definitions of topology. On a set of data $\mathbf{X}$ one can define a topology $\mathcal{T}$ as a collection of subsets of $\mathbf{X}$ such that it forms a hierarchical structure on X . By defining a basis $\mathcal{B}$ of the topology $\mathcal{T}$ on the collection of small neighborhoods of $\mathbf{X}$ we can define a structure of the high dimensional dataset. And by identifying an n -dimensional Euclidean space E_n such that there exists a separable metric space M (in $\mathcal{T}$) that is locally homeomorphic with E_n , we can define this M to be a Topological n -manifold. Intuitively, a topological n -manifold locally displays properties of an n -dimensional euclidean space.

This concept of topological manifolds therefore now provides us a new approach to visualize structures in high dimensional data. A most intuitive approach would be to break down a large high dimensional dataset into a collection of small neighborhoods, such that each of these small neighborhoods are topological d -manifolds, where d is the dimensionality that we wish to reduce the original data down to. Specifically in visualization tasks, $d = 2, 3$. Motivated by identifying a manifold by reconstructing small neighborhoods, we next describe methods that utilize this philosophy of *manifold learning*.

Locally Linear Embedding

One of the earliest manifold learning technique in usage is Locally Linear Embedding (LLE) [1]. LLE first decomposes the high-dimensional data into a collection of small neighborhoods around each point. Next, we reconstruct each datapoint as a linear combination of its neighbors. The matrix of coefficients used for this reconstruction is then used to embed a set of low-dimensional points in the visualization space.

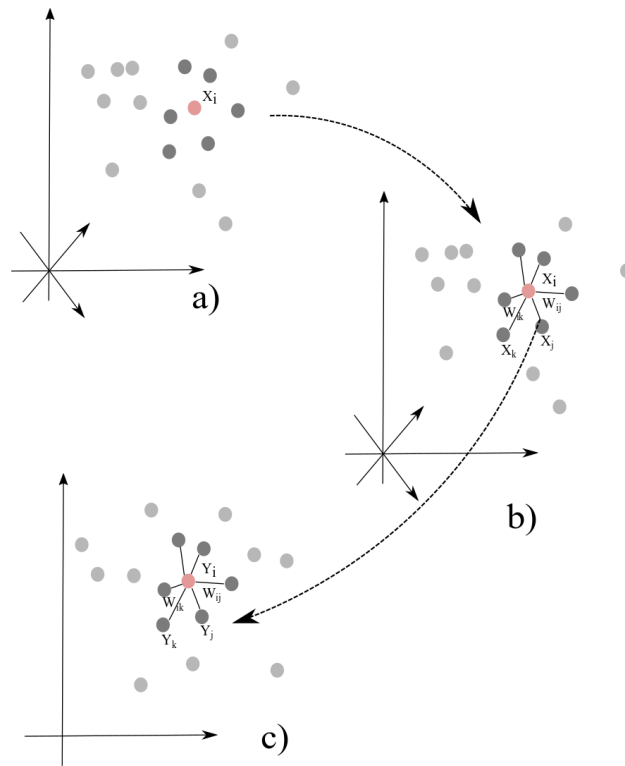


FIGURE 1.5: The LLE algorithm summarized. a) For each input, a group of closest neighbors are selected. b) The input is then reconstructed using a linear combination of its neighbors, c) the weights used for linear reconstruction is then used to embed the output coordinates [1].

Figure 1.5, which we have recreated based on [1], summarizes this algorithm. However, one important note is that LLE attempts to reconstruct all points as linear combinations of its neighbors, and this is done to highly abnormal points as well. This makes LLE highly susceptible to noisy data[26].

Manifolds with MDS

As we discussed earlier, MDS is a more robust method than PCA, and is inherently nonlinear. To overcome the limitation of MDS where attempting to preserve global distances in high dimensional spaces skews its visualizations as we discussed earlier, modified versions of MDS are also discussed in existing literature.

We discuss two main approaches for using MDS as a manifold learning method: 1) Construct the manifold prior to the calculation of the pairwise distances where we get a pairwise distance matrix which is non-euclidean as input to MDS or 2) Conduct MDS on only a small neighborhood of Euclidean distances at a time.

The first approach has one notable example, called IsoMap [27]. In IsoMap, we first

decompose the input into a set of small neighborhoods, and construct a densely connected graph in each of these neighborhoods. Since these neighborhoods constitute the complete dataset, we have a set of graphs which approximates the complete manifold of the input data. Next, we calculate all-pairs shortest path distance calculation using Dijkstra's algorithm[28], thus providing us a 'geodesic distance' between any two pairs of points along the manifold. Note that for disconnected components of graphs, the distance between any two points in two disconnected components is considered to be infinite. Multidimensional Scaling is then used for embedding the low-dimensional output points to reconstruct this geodesic pairwise distance matrix.

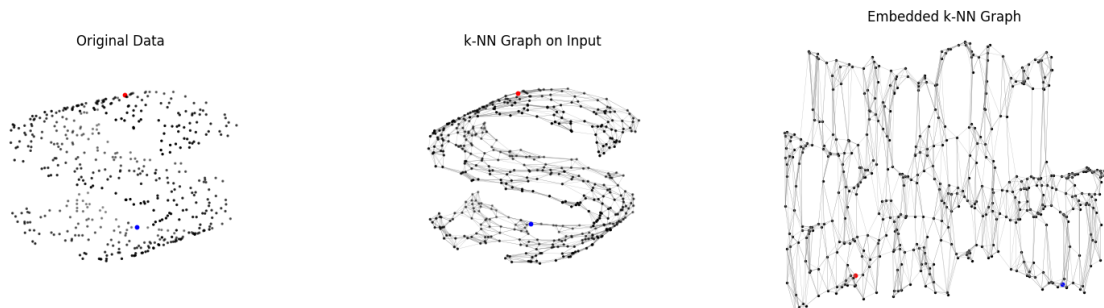


FIGURE 1.6: How Isomap the k-NN Graph to reduce dimensionality. First a k-NN Graph is created on the discrete input set (a so-called S-Curve dataset). Next, the pairwise shortest path distances are calculated. These distances are preserved in the embedded graph in 2 dimensions. The distance between the red and blue input points are preserved along the edge of the S-shape rather than the straightline distance between them in the input space.

The second approach also has one notable example, called Local Multidimensional Scaling (LMDS) [29]. Similar to IsoMap, it first decomposes the input data into small neighborhoods. However, rather than attempting to construct a geodesic distance matrix, it attempts to reconstruct these small neighborhoods in the output space.

It is noteworthy that in both these approaches, the manifold was approximated by a locally connected graph, and MDS was used as an 'Embedding' technique of the graph. This brings to our attention another angle that the dimensionality reduction problem can be looked at from: a graph embedding problem.

Having constructed a locally connected approximation of the manifold in the form of a graph, what other techniques may we use (other than MDS) to embed this graph in a low-dimensional output space? One of the most useful concepts in doing so is the concept of the graph Laplacian.

Spectral Embedding

Spectral Embedding (SE), also known as Laplacian Eigenmaps [30] uses a different technique to embed the graph approximating the manifold. SE first defines the adjacency

matrix of the graph as A and the degree diagonal of the matrix as D . The Laplacian L of the graph is defined as:

$$L = D - A. \quad (1.6)$$

However, in their work, Belkin *et al* define a modified Laplacian as:

$$L = D - W, \quad (1.7)$$

where W is a weight matrix representing the weights of the edges in place of the adjacency. Finding the embedding of the graph is then formulated as solving the Eigenvalue problem:

$$L\mathbf{f} = \lambda D\mathbf{f}. \quad (1.8)$$

Essentially, the first k solutions for $\mathbf{f}$ are then considered to be the k - dimensional reduced representation of the original data.

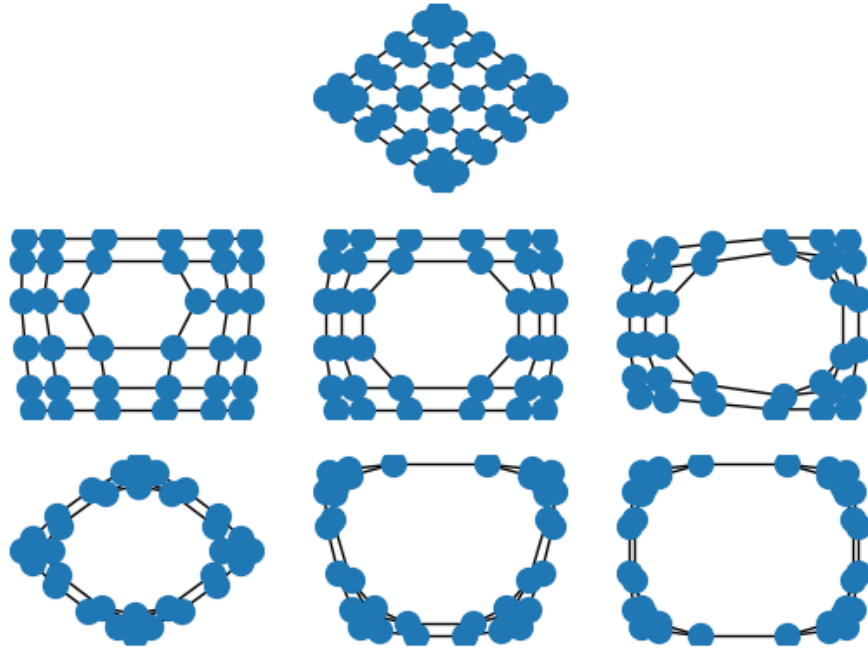


FIGURE 1.7: Different graph structures embedded in two dimensions using Laplacian Eigenmaps[2]. As we see, the visualizations have emphasized the ‘hole’ areas and ‘compact’ areas of the network.

In Figure 1.7, different structures of graphs are embedded using Laplacian Eigenmaps. The X axis of these scatter plots represent the component values of the eigenvector which has the highest eigenvalue and the Y axis represents the component values of the eigenvector with the second highest eigenvalue. Note that the graphs where the connections are sparse, this sparsity is emphasized in the embedding.

Stochastic Neighborhoods and Contemporary Visualization Methods

While all the manifold learning methods described above rely on constructing locally connected graphs, we note that in all of these methods, the membership of a data point in a graph is deterministic (i.e. binary). However, this becomes problematic when the data has either a significant number of data points that are outside the usual span of a distribution, or a small number of data points that are significantly far away from the center of a cluster. We point to some examples in Figure 1.8.

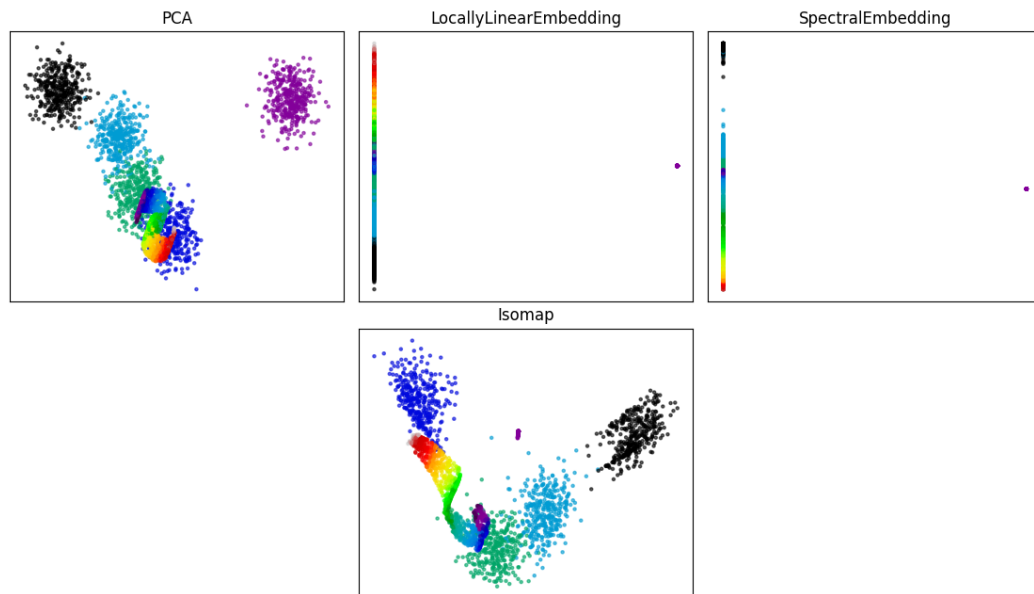


FIGURE 1.8: The simulated data consists of five Gaussian clusters and one S curve. Note that PCA fails to capture the S-curve topology and complex distribution of clusters causes them to be mixed in the representation. IsoMap, LLE and (Laplacian) Eigen maps are sensitive to noise, hence the visible distortions.

This shortcoming may be avoided by not considering the preservation of all neighborhoods equally correctly. With this in mind we now discuss the concept of a ‘stochastic neighborhood’ rather than a ‘deterministic neighborhood’.

Originally proposed by Geoffrey Hinton [31], Stochastic Neighbor Embedding (SNE) approaches the construction of the manifold graph as a collection of pseudo-probability distributions. In SNE, data in a given neighborhood is assumed to be distributed in a

Gaussian manner. Therefore, all distances between input pairs $\mathbf{x}_i$ and $\mathbf{x}_j$ are converted into probability values p_{ij} of the form:

$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / \sigma_i^2)}{\sum_{\forall k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / \sigma_i^2)} \quad (1.9)$$

where $\|\cdot\|$ stands for the euclidean norm of a vector and σ_i is a local constant that represents the perplexity of the distribution.

Using this definition, probability distribution for the whole input set is calculated as the symmetric probability set $P = P_{ij}$ for each input $\mathbf{x}_i, \mathbf{x}_j$ where:

$$P_{ij} = \frac{p_{ij} + p_{ji}}{2} \quad (1.10)$$

Having found a stochastic representation of the input, we next define a set of low-dimensional corresponding vectors $\mathbf{Y}$ and define a similar probability distribution as:

$$Q_{ij} = \frac{q_{ij} + q_{ji}}{2}, \quad (1.11)$$

where,

$$q_{ij} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)}{\sum_{\forall k \neq i} \exp(-\|\mathbf{y}_i - \mathbf{y}_k\|^2)}. \quad (1.12)$$

Note that in this case, Q also follows a gaussian distribution.

We can see that altering the positions of $\mathbf{y}_i$ s changes the distribution Q , and we would like Q to represent P as much as possible. Therefore, we need to optimize $\mathbf{Y}$ such that some error between Q and P is minimized. Hinton defines this error to be the Kullback-Liebler divergence (KLD) and defines the error to be minimized as:

$$E = \mathbf{KLD}(P||Q). \quad (1.13)$$

As noted above, both P and Q assumes a Gaussian Distribution. However, for highly dimensional data, as we noted earlier, the sizes of the neighborhoods for manifold learning changes drastically. Therefore, SNE still suffers from a fair amount of mixing of clusters. This is exemplified by Figure 1.9, where we use SNE to visualize images from 5 different digits provided by the UCI Digits dataset[32].

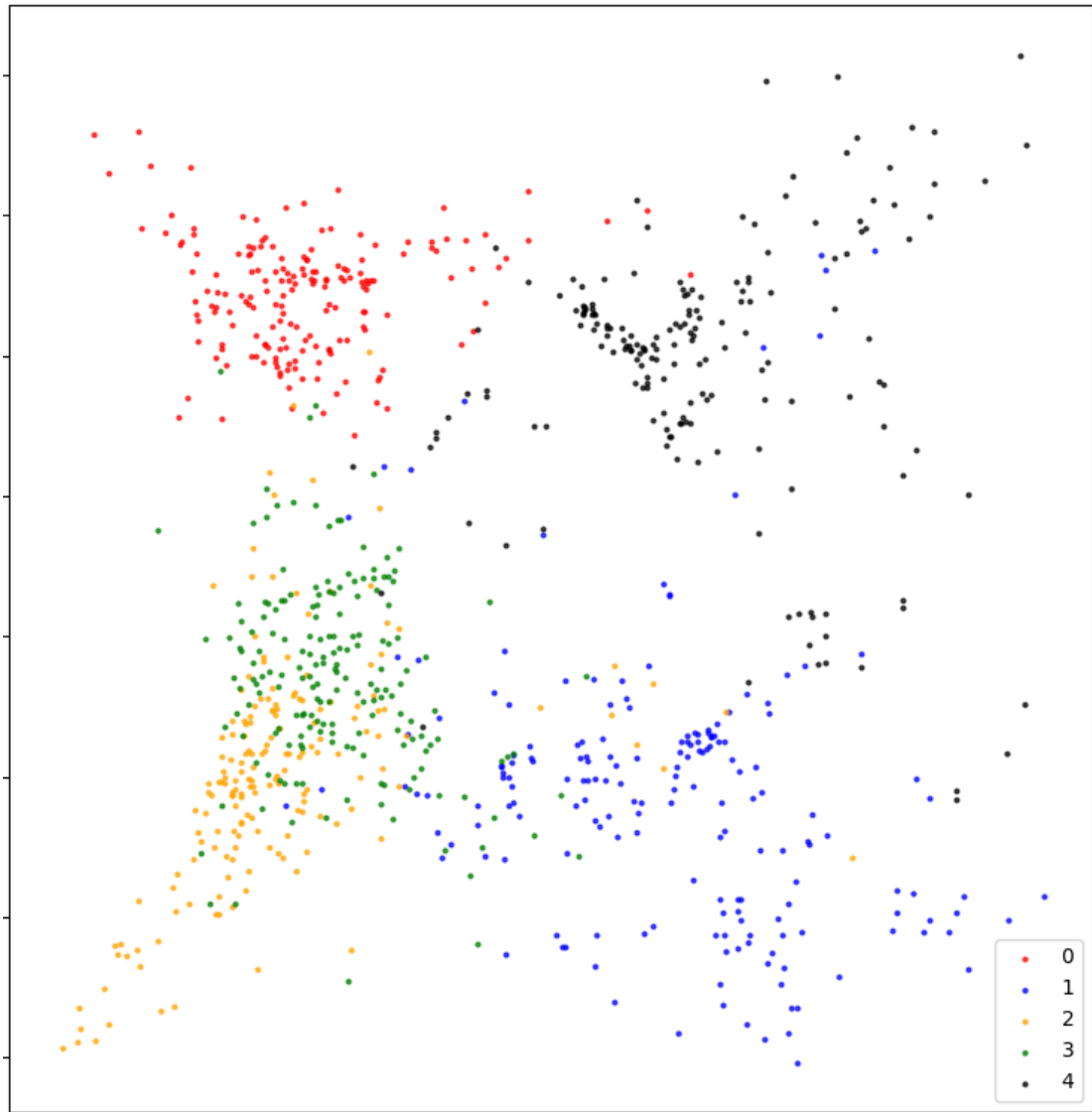


FIGURE 1.9: Five different digits visualized from the UCI-Digits dataset using SNE. While the clusters are well organized, poor separation is apparent.

1.1.3.2 t-SNE for More Intuitively Separate Cluster Visualizations

This problem was identified and called the ‘crowding problem’ by Vandermaaten [20]. As a solution for this discrepancy of dimensionalities, Vandermaaten and Hinton proposed modifications to the Q such that instead of a Gaussian distribution, it follows a students-t distribution [20]. Since the t distribution has a heavier tail than the Gaussian distribution, the moderately distant data points in P were allowed to be placed relatively closer in Q . This modified method is called t-distributed Stochastic Nonlinear Embedding (t-SNE) and is arguably the most widely used nonlinear visualization technique as of the writing of this review. We show how t-SNE improves the visualization of clustered data points in the Figure 1.10 where we used t-SNE to visualize the same data as in Figure 1.9.

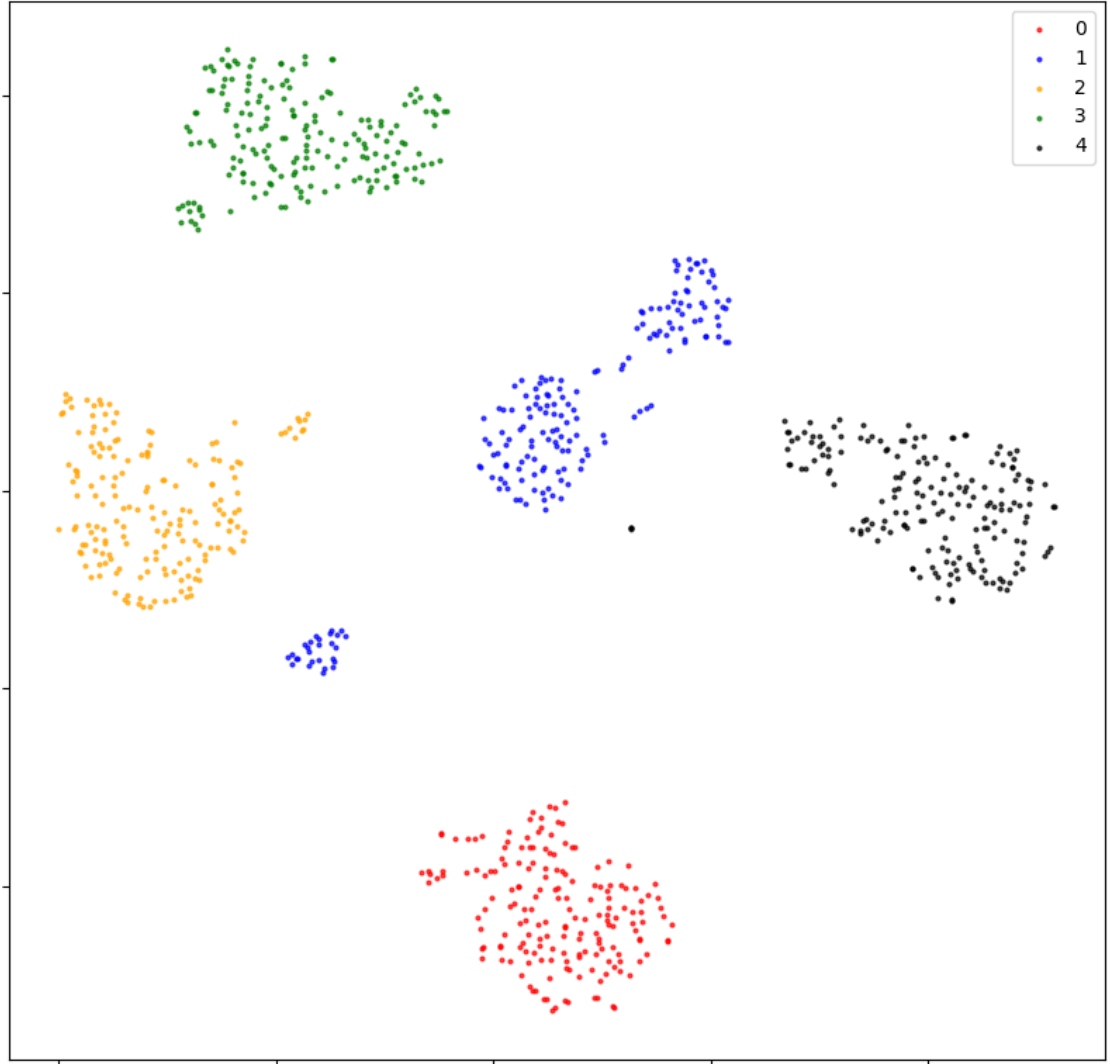


FIGURE 1.10: Five different digits from the UCI digits dataset visualized using t-SNE. While the clusters are well preserved, they are well separated, providing us more intuitive information about the structure of the data than SNE did.

While t-SNE introduces a random-walk algorithm to reduce the running time of the visualizations, it still suffers heavily from repetitive calculations of global distances (especially in calculating Q). Additionally, t-SNE only preserves the local neighborhoods, and therefore, although the clusters are differentiated, the global topology of the clusters is not consistently preserved.

1.1.3.3 UMAP for Robust Global Topology Preservation

A recent stochastic neighborhood embedding algorithm called Uniform Manifold Approximation and Projection (UMAP)[33] has addressed this by making two clear changes to the classical SNE algorithm:

1. Instead of explicit assumptions on the probability distribution on the input space, make a more relaxed assumption - specifically, assume that the manifold is Riemannian
2. Instead of using KL - Divergence, use cross entropy as error. This allows a global structure to be preserved.

Recent work by Becht *et al* [34] shows how these modifications can improve the visualizations provided by UMAP when compared to t-SNE (reproduced from [35] under CC BY-NC 4.0).

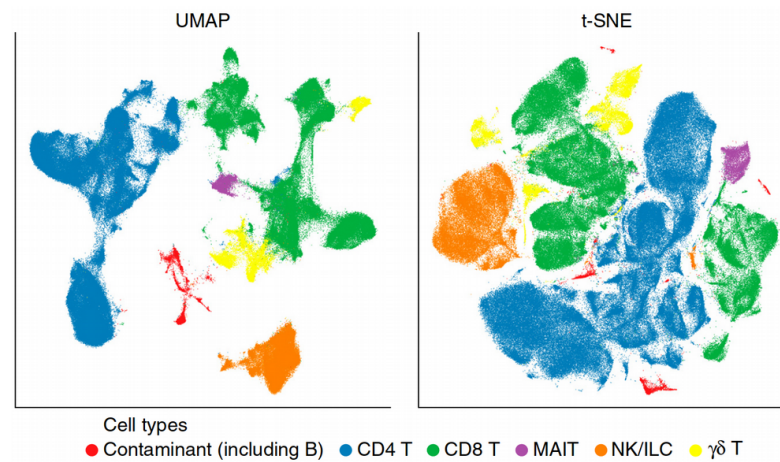


FIGURE 1.11: Visualization of flow cytometry measurements obtained on a population of immune cells[3] (CC BY-NC 4.0)

As visible in Figure 1.11, due to the lack of explicit assumptions on the probability distribution of the local neighborhoods, UMAP is able to identify more complex structures whereas t-SNE attempts to fit the data into Gaussian neighborhoods. Furthermore, the use of cross-entropy by UMAP allows the identification of global structures, which is not represented in the t-SNE visualization. This causes better cluster quality and cluster separation.

UMAP also employs several heuristics to improve its performance. Two of the most important heuristics are:

- Use of Spectral Embedding to initialize the low-dimensional embedding: Rather than using a random initialization of low-dimensional points to be optimized by minimizing the cross entropy between the two probability distributions, UMAP uses spectral embedding to initialize the low-dimensional representation. This allows for faster convergence, as well as consistent structuring of clusters when the optimization is repeated for similar data.

- Negative Sampling to optimize the ‘repulsive component’: The authors of UMAP discuss how the gradient of the cross entropy error can be decomposed into *attractive* and *repulsive* components. Since the ‘repulsion’ affects the placement of clusters, it needs to consider all inputs relative to the point for which the error is being minimized for at a given time. However, they show that taking inspiration from algorithms such as Word2Vec [36], rather than considering all non-connected nodes, randomly selecting a handful at a given time provides results without any noticeable loss of quality. This improves the running time of the UMAP algorithm significantly compared to the likes of t-SNE.

In essence, both UMAP and t-SNE first convert the distances in the input space into a pseudo-probability distribution, and then mimic this probability distribution in a low-dimensional output space. Because these distributions weigh highly noisy samples less than what can be considered to be the ‘regular’ or ‘usual’ samples, they are highly tolerant to noise. But this comes at a cost, in that it does not retain any information or a mapping between the input to output relationship. More technically, they are both non-parametric methods.

However, in reality, it is common to have cases where retention of a parametric map is useful. Such a map not only facilitates the generalization and augmentation of a model, but also provides a framework with which to learn the representations (or the rules) of the reduction mapping. While t-SNE has provided a framework with which to retain a mapping, discussion of it must be preceded with an introduction to the popular class of machine learning algorithms known as Neural Networks.

1.1.4 Neural Networks

In modern data analysis, Artificial Neural Networks, or as they are more commonly referred to, Neural Networks are ubiquitous and highly useful. Originally proposed by McCulloch and Pitts [37], neural networks were considered to be an attempt at mathematical modeling the functionality of animal brains.

In this work, we summarize the class of neural networks with one sentiment: *they are a class of bio-inspired mathematical models capable of modeling highly nonlinear relationships through repeated transformations of signals.*

As a short introduction into Neural Networks (or at least as far as our applications are concerned) we must first describe the early and abstract models of neural activity. While the functionality of a biological neuron is extremely complex, at the most abstract level,

it can be considered as a transformation applied to a set of signals to obtain an output signal.

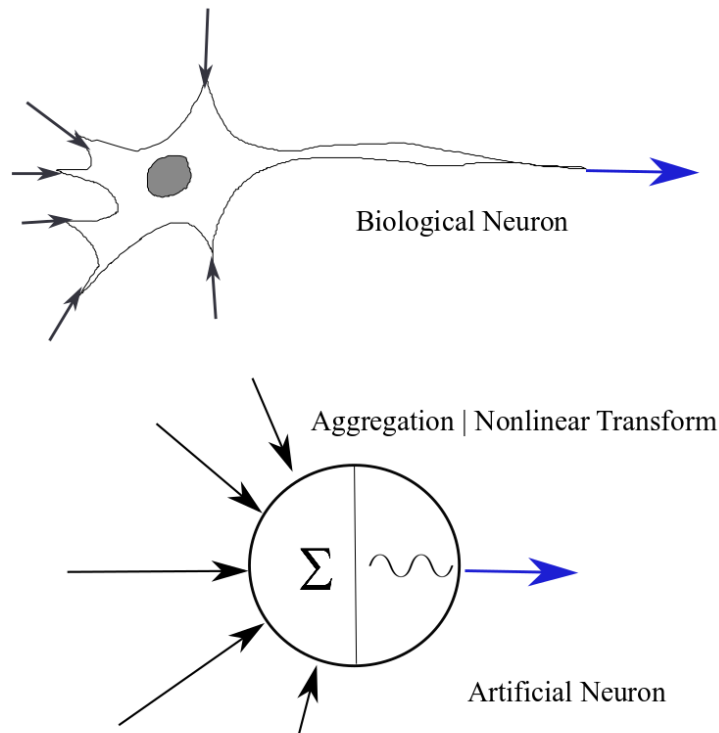


FIGURE 1.12: An abstraction of a biological neuron compared to an artificial neuron based on [4]. A biological neuron gathers the synapses from neighboring connected neurons (shown as black arrows) and aggregates this signal and passes through the axon to the next neuron (blue arrow). The signals are not uniform due to physiological differences between each synapse. This is reflected in an artificial neuron as weighted inputs to the neuron (black) and summation, followed by a non-linear activation function.

In Figure 1.12, we show how the biological neuron has inspired the artificial neural model. The aggregate output of the biological neuron can be considered as a function of the electrophysiological properties of the cell membrane [38]. For instance some assumptions and hypotheses are that, if the incoming dendrites have high conductivity (typically regulated by the surface area of the dendrite), the aggregate output is also high. And since different incoming signals take different paths, they are weighted differently, and have different influences on the final output. The nature of the cell-body may inhibit some impedance, thereby making the aggregate a nonlinear function of the incoming signals.

Following these assumptions and hypotheses, we model the artificial neuron such that different input neurons are weighted differently using a linear transformation. Next these signals are aggregated with a summation operation, and with the use of a nonlinear ‘activation function’, the nonlinearity of the output is ensured. Essentially, if we consider

the set of incoming signals as an input vector, we can summarize this activation as follows:

$$a = \alpha(\mathbf{w}^T \cdot \mathbf{x} + b) \quad (1.14)$$

where a stands for the final output, α is a nonlinear activation function (such as a step function, rectifier, sigmoid or maximum function etc.), $\mathbf{w}$ is a vector of weights (representing the dendritic properties of the neuron) and $\mathbf{x}$ represents the vector of inputs. The goal of optimizing the set of weights $\mathbf{w}$ is to obtain activations a as similar to some target y . To avoid trivial solutions and to regulate the search space, we use a scalar bias b .

In practice, a large number of neurons are combined together to form a network of highly complex transformations, which we call a neural network. Neural networks can be categorized into ‘neural architectures’ depending on how these networks are combined. In the following sections, we describe some of these strategies. However, we only delve into depths of the architectures that we use as methods in our research contributions.

1.1.4.1 Feed Forward Neural Networks

Described as one of the most simple neural networks in use [39], Feed Forward Neural Networks move information in only one direction, i.e. if the output of a neuron A is being fed to a neuron B, the output of B is never directly or indirectly given as input to neuron A. In terms of architecture, the feed forward network is one that does not form a cycle[40]. While there are many possible architectures of feed forward networks, one of the simplest architectures is the layered network model (Multilayer Feed-forward network or MLF).

Often in literature, graphs are used to represent the architecture of a neural network, where vertices represent neurons and edges represent the connections between neurons. These edges are directional, where the output of the source vertex is considered to be the input to the destination vertex, and the weight of the edge represents the weighting of this input to the destination vertex. With this in mind, a MLF network can be seen in Figure 1.13.

Having multiple nonlinear transformations from the input to the output creates complex input-to-output relation modeling possible with multilayer neural networks.

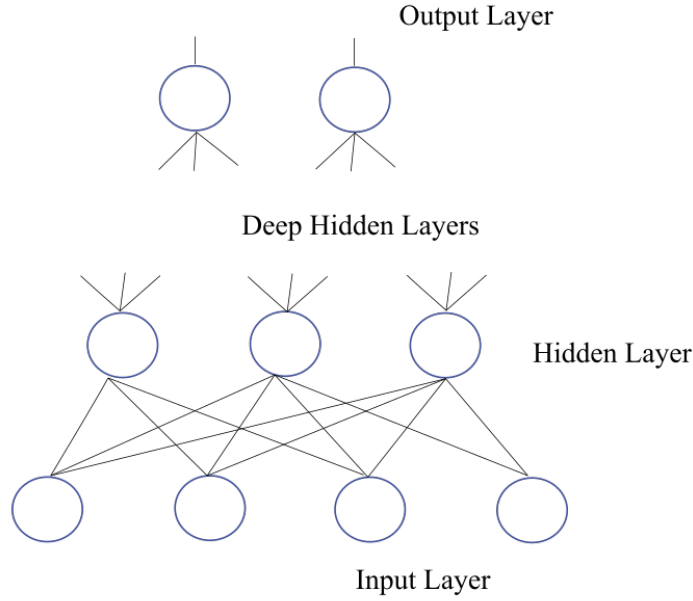


FIGURE 1.13: A feedforward neural network with hidden layers (based on)[5]

While this provides the framework for input-to-output conversion, we have yet to discuss how to optimize this conversion. Intuitively, a simple approach is to treat this as an optimization function where:

- The generated output at the last layer $\mathbf{a}_L$ must be similar to some target output $\mathbf{t}$.
- We have tunable parameters that are the weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ at each layer $l = \{0 \dots L\}$ where L is the maximum depth of the network, and $l = 0$ is the input layer.

With this in mind, we now formulate an optimization problem with the feed forward neural networks as below.

Consider the input $\mathbf{x}$, which can be converted into the activations of the first layer $\mathbf{h}_1$ as:

$$\mathbf{h}_1 = \alpha_1(\mathbf{W}_1 \cdot \mathbf{x}^T + \mathbf{b}_1) \quad (1.15)$$

where $\mathbf{W}_1$ is a matrix such that each row represents the incoming weights of a neuron in $l = 1$, and $\mathbf{b}_1$ is a vector where each component is the scalar bias of each of these neurons. α_1 is our selected activation algorithm.

Considering that the output of a layer is considered as the input to the following layer, we can generalize this to the l^{th} as follows.

$$\mathbf{h}_l = \alpha_l(\mathbf{W}_l \cdot \mathbf{h}_{l-1}^T + \mathbf{b}_l). \quad (1.16)$$

This means now we can formulate this as a more general transformation f of the form:

$$\mathbf{h}_l = f_l(\mathbf{h}_{l-1}, \Theta_l) \quad (1.17)$$

where Θ_l is the set of parameters pertaining to layer l . Therefore, we can now convert the complete transformation from input to output into the composite form:

$$\mathbf{h}_L = f_L(f_{L-1}((f_{L-2}(\dots, \Theta_{L-2}), \Theta_{L-1}), \Theta_L). \quad (1.18)$$

This provides us a convenient parametrization of the transformation of input to outputs and isolates the set of parameters Θ_l at each layer l .

While there are many approaches of optimizing neural networks such as genetic algorithms [41], and Hebbian Learning [42], gradient learning methods have been more frequently used for optimizing MLF neural networks. While we attribute the low complexity and interpretability of gradient methods to be thus preferred, we describe one of the most prominent techniques used for optimizing neural networks known as backpropagation in the following.

1.1.4.2 The Backpropagation Algorithm

Relating the neural network problem to an optimization problem, we have a set of tunable parameters $\Theta = \{\Theta_1 \dots \Theta_l\}$ and an input to output mapping $\mathbf{x} \rightarrow \mathbf{h}_L$, which we formulate into an optimization problem as:

$$\Theta_{\text{optimal}} = \underset{\Theta}{\operatorname{argmin}} \sum_{\forall \mathbf{x}, \mathbf{t}} \mathcal{L}(\mathbf{h}_L, \mathbf{t} | \mathbf{x}, \Theta), \quad (1.19)$$

where $\mathcal{L}$ is some loss / error function and $\mathbf{t}$ is the set of expected outputs. In literature, in cases where both input and output are of vector (or more generally tensor) formats, the Root Mean Squared Error (RMSE) is commonly used. Furthermore, for cases with stochastic representations (output is a set of probabilities), cross-entropy or KL divergence may also be used.

In cases where $\mathcal{L}$ is differentiable, we can employ a gradient descent strategy such that for each iteration of optimization, the following adjustment to Θ is made.

$$\Theta \leftarrow \Theta - \lambda \frac{\partial \mathcal{L}}{\partial \Theta}, \quad (1.20)$$

where λ is the learning rate or the step-size at which we conduct the gradient descent.

However, the problem arises now in that for each $\mathbf{W}_l$ and $\mathbf{b}_l$, the loss is not directly differentiable by these parameters. At this stage, we make use of the composite representation of input to output that we used, such that we can calculate the gradients *backwards* from the last layer.

Taking a more concrete example, we can calculate the adjustments to the last layer of weights as:

$$\Delta \mathbf{W}_L = \lambda \frac{\partial \mathcal{L}}{\partial \mathbf{W}_L} \quad (1.21)$$

and using the chain rule, calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_L} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \times \frac{\partial \mathbf{h}_L}{\partial \mathbf{W}_L} \quad (1.22)$$

which provides us an alternative form of the adjustments as:

$$\Delta \mathbf{W}_L = \lambda \left(\frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \times \frac{\partial \mathbf{h}_L}{\partial \mathbf{W}_L} \right) \quad (1.23)$$

similarly, the adjustments for the biases can also be calculated as:

$$\Delta \mathbf{b}_L = \lambda \left(\frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \times \frac{\partial \mathbf{h}_L}{\partial \mathbf{b}_L} \right) \quad (1.24)$$

Next, we use the composite form in Equation 1.18 to obtain the gradients and adjustments of the intermediate layers as:

$$\Delta \mathbf{W}_l = \lambda \left(\frac{\partial \mathcal{L}}{\partial \mathbf{h}_l} \times \frac{\partial \mathbf{h}_l}{\partial \mathbf{W}_l} \right) \quad (1.25)$$

and

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \times \frac{\partial \mathbf{h}_L}{\partial \mathbf{h}_{L-1}} \times \dots \times \frac{\partial \mathbf{h}_{l+1}}{\partial \mathbf{h}_l}. \quad (1.26)$$

Note that the composite form Equation 1.18 ensures that all individual components of partial derivatives in this chain rule indeed exist, provided that our selection of f_l s are all differentiable.

1.1.4.3 Pragmatics: Deep Networks and Regularization

While the collection of feedforward neural networks, along with the optimization strategy using backpropagation allows us the flexibility of a highly complex transformation from input-to-output, in practice, large networks with multiple intermediate layers, known as hidden layers are used [43]. However, the problem of learning a general map from a limited set of samples is considered to be ill posed and regularization of neural networks may be required [44].

One of the widely used examples for neural network regularization is the use of weight decay or more generally Tikhonov regularization [45], where we penalize very large weights in the neural networks. This ensures that the weights do not explode to approximate highly abnormal samples.

In unsupervised usages, where we use a neural network to reconstruct the input rather than predict a set of different variables (i.e. $\mathbf{t} = \mathbf{x}$) we employ an architecture called an ‘autoencoder’. Typically, an autoencoder consists of three layers, an input layer, and an output layer which is not different from any other network, but with the constraint on the hidden layer where we have a smaller number of neurons than the input. Because we try to get a compact representation of the input distribution with a small number of neurons in this middle layer, we call it a ‘bottleneck layer’. For regularizing these autoencoder networks, we use a strategy called sparse autoencoders [46].

Sparse autoencoders get their name due the fact that the activation of the hidden layers are sparse, i.e. we penalize over activation of the hidden neurons. The constraint on hidden neurons to be activated infrequently draws inspiration from neuroscientific concepts, where such infrequent activations influence efficient configurations of the neurons as well as preserving energy in the neural cells[47]. When enforced on an artificial neural

network, this constraint ensures that the representations learned from the original data in the hidden layers are efficiently encoded, and it has been observed that these compact features increase in interpretability [46], as we show in Figure 1.14 [46].

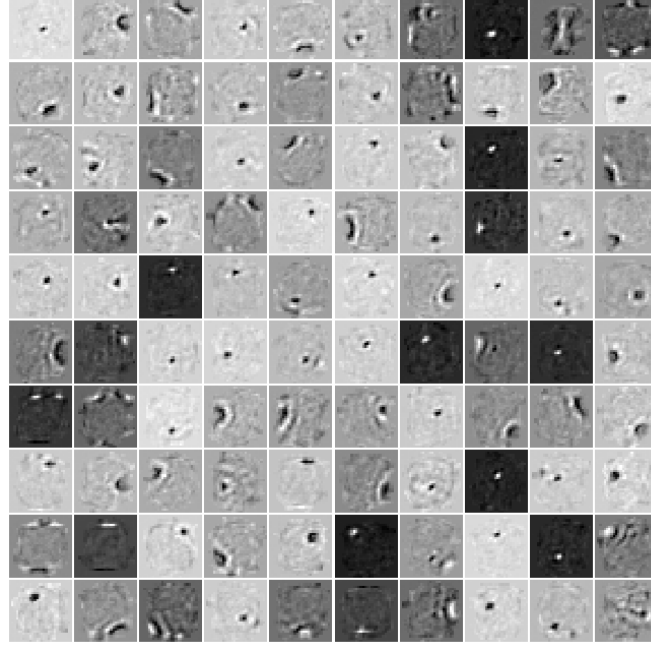


FIGURE 1.14: Each square represents the input weights of a hidden neuron. Note that no proximity information has been used in the extraction of these features. The autoencoder has learned the 'edge representations' of the image set it was trained on.

Considering the feature extraction capability of sparse autoencoders, a more straightforward approach of extracting interpretable and meaningful features from image data can be done using the proximity information of the inputs: i.e. close-by neurons have related expression patterns in the input. While specifically for image data where the proximity of pixels are highly meaningful, there exists a straightforward and popular class of neural networks called Convolutional Neural Networks [48], we attempt to generalize the notion of 'spatially related inputs' in our study with the discussions about Hebbian Neural Networks, in the following sections.

1.1.5 Other Architectures: Self Organizing Feature Maps

Having described some of the underlying principles of neural networks, we now relate the concepts used in neural networks to the original problem in our research, which is the analysis of high dimensional data. For this, we use the seminal work by Kohonen [49] on the Self Organizing Feature Maps (SOM).

While there are multiple interpretations on the SOM architecture, in our work, we use the following intuitive interpretation. A SOM places a set of weight-vectors (also known

as coding vectors) in the input space, such that they are placed in the highly dense areas of the input space. This process is known as Vector Quantization [50] in that it quantizes an input space with a closest neighbor constraint. This constraint implies that any input $\mathbf{X}$ can be encoded with a set of coding vectors $\mathbf{C}$, where each $\mathbf{x} \in \mathbf{X}$ can be represented by its closest $\mathbf{c} \in \mathbf{C}$ [50]. In summary, vector quantization is finding an optimal $\mathbf{C}_{\text{optimal}}$ such that,

$$\mathbf{C}_{\text{optimal}} = \underset{\mathbf{C}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathbf{X}} d(\mathbf{x}, \mathbf{c}_{\mathbf{x}})^2, \quad (1.27)$$

where d is some distance metric defined on the input space of $\mathbf{X}$ and $\mathbf{c}_{\mathbf{x}}$ is the closest coding vector to $\mathbf{x} \in \mathbf{X}$.

Kohonen enforces a further constraint on the ordering of $\mathbf{C}$, such that the coding vectors are organized on a low-dimensional grid. Typically, this grid is a two or three dimensional regular lattice (square, hexagonal or triangular). Specifically, this constraint ensures that coding vectors which have corresponding points on the grid that are close to each other, are located close to each other in the encoding space as well. Computationally, this is achieved by defining a set of low-dimensional vectors $\mathbf{Y}$ such that each $\mathbf{y} \in \mathbf{Y}$ has a bijectively corresponding $\mathbf{C}$. Next, we define a monotonic function σ of the distance between any two $\mathbf{y}$ s, such that as the distance increases, σ decreases. The modified task of the vector-quantization along with the new ‘self-organization’ constraint added can be written as finding $\mathbf{C}_{\text{optimal}}$ such that:

$$\mathbf{C}_{\text{optimal}} = \underset{\mathbf{C}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathbf{X}} \sum_{\mathbf{c} \in \mathbf{C}} d(\mathbf{x}, \mathbf{c})^2 \times \sigma(\mathbf{y}_{\mathbf{c}_{\mathbf{x}}}, \mathbf{y}_{\mathbf{c}}), \quad (1.28)$$

where $\mathbf{y}_{\mathbf{c}}$ is the low dimensional point mapped to $\mathbf{c}$.

A simple example of training a SOM on a two dimensional dataset can be shown in Figure 1.15 ¹ (Reproduced from original media under (CC BY-SA 3.0)).

We note here, that by representing each input $\mathbf{x}$ with the corresponding low-dimensional vector $\mathbf{y}_{\mathbf{c}_{\mathbf{x}}}$, we can obtain a low-dimensional representation of $\mathbf{X}$. Therefore, SOMs are often used as a dimensionality reduction technique. The constraints of local organization ensures that in such visualizations, the topology of the input is preserved, and thus the topological manifold is approximated.

Although the SOM algorithm has not been widely used in the recent years due to its poor separation of clusters[29] and distortion of representations[6], the principle of Self

¹https://en.wikipedia.org/wiki/Self-organizing_map#/media/File:Somtraining.svg

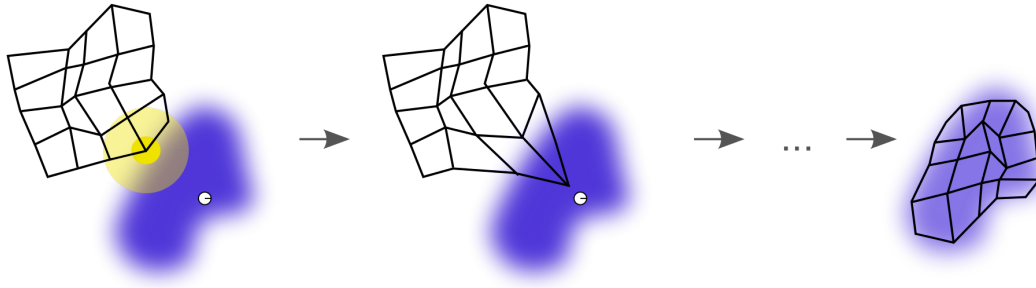


FIGURE 1.15: Blue represents the input dataset. The grid represents the defined organization of coding vectors. The coding vectors are the nodes of the grid. (CC BY-SA 3.0)

Organization has laid foundations to work relating to topological approximations using neural networks. We discuss these in conjunction with the concept of Hebbian Learning.

1.1.5.1 Hebbian Learning and Hebb-like Neural Networks

Hebbian Learning [51] is an alternative method to the back-propagated gradient descent to optimize the weights of a neural network. They have been used to optimize feed-forward networks [52], however in this case we are now attempting to relate the concepts of Hebbian learning to the concept of high-dimensional manifold learning.

The fundamental principle of Hebbian learning uses the frequency of simultaneous activation of neurons to enforce connections between them. Competitive Hebbian Learning (CHL) enforces connections only to the highest activated pairs of neurons. Martinez [53] introduces the use of CHL to approximate graphs between coding vectors in a vector quantization problem.

Unlike SOM, where the connections between neurons are fixed on a low-dimensional manifold, CHL creates edges, and strengthens them between pairs of coding vectors that are activated simultaneously. The simultaneous activation is related to the spatial proximity of coding vectors and inputs, i.e., two coding vectors are activated simultaneously if they are the closest two vectors to an input. This causes the edges to be enforced only in densely connected regions. With this, martinez introduces a new algorithm called Neural Gases [54].

Expanding on the work by Martinez, Fritzke introduced Growing Neural Gases [6], where rather than having a fixed set of coding vectors, the coding vectors grow as needed to approximate a finer and finer topology. Furthermore, Fritzke discusses and proposes an approach for general growing self organizing networks in his later work[55]. This work was later adapted to Self Organizing Maps by Alahakoon *et al* [56].

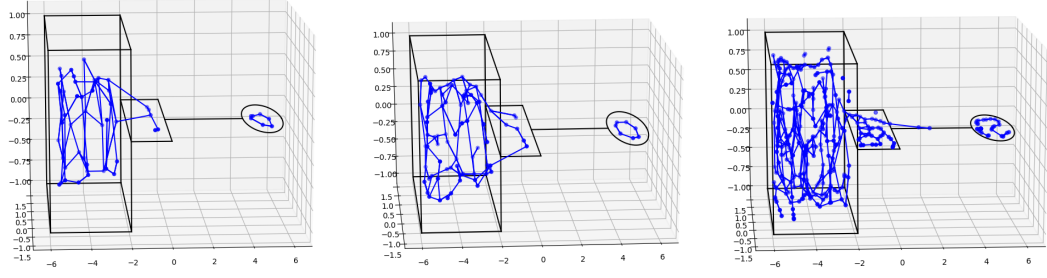


FIGURE 1.16: Approximation of a complex nonlinear structure using GNG (figure based on [6]). The blue points represent the coding vectors, and the blue lines represent the edges connecting the coding vectors. We see that as training proceeds (from left to right), the number of vertices increase, approximating the density of the input space (space within the scaffold boundary).

We note here that one of the short-comings of both Neural Gas and Growing Neural Gas is that they do not provide a visualization of the inferred topology. Although Growing Self Organizing Maps (GSOM) [56] uses a visualization strategy similar to SOM, it does not provide flexibility in the inferred topology as the graph is of a fixed topology.

While graph-embedding algorithms as we have discussed before can be used to get an embedding of the approximated topology graph by the coding vectors, this relies on the accuracy of the embedding method heavily. However, early work in visualizing the coding vectors by Estevez *et al* [57] have used concepts similar to stochastic neighbor embedding to obtain a visualization by minimizing a cross entropy error between the coding vectors and a set of low-dimensional vectors.

We note that work by Estevez bears similarity to the approach of visualization used by UMAP, and we leverage this similarity in our Question 2.

Having discussed the background work and existing literature related to our method development work (Questions 1 and 2), we now discuss the background on the domain where we apply these techniques. While these methods are general in nature, we select a highly influential field of study which is up and coming in the recent years called Single Cell Analysis.

1.2 Single Cell Data Analysis

Experimental Biology entails the empirical study of biological systems through the scientific method, and understanding of the fundamental building blocks of our complex ecologies are a necessary step in building a full picture what happens in such complex systems. While the question of ‘what counts as life’ is a philosophical problem which has intrigued humans, possibly as far as humans have been in existence, currently we

consider the cell to be the smallest functioning unit of life, and the branch of experimental biology that tackles the ‘cell’ as a unit of life is called Cellular Biology. Recent advances in molecular biology has opened up new avenues into analysing the complex dynamics of cells. However, currently, cellular biology is an ever evolving field with a high demand for data-driven empirical analysis techniques.

The biological reality of cells are often difficult to capture, mainly due to their microscopic scale. The means of capturing different measurements of cells is an evolving field of study, as we cannot rely on visual or direct observations to measure variables in a cellular system. Making matters more challenging is the lack of understanding of even minute building blocks of cells such as the genetic components.

Single Cell analysis is a relatively new area of genomic and transcriptomic studies, which were not previously feasible due to the technical and instrumental limitations[58]. With the ability to rapidly sequence RNA at a cell level, biologists have been able to get many insights into the biology of cells at an unprecedented resolution. Prominent among these insights are new cell types, discovery of new cell lineages and patterns of gene expression variations across different cell populations. Such exploration of fine-grained heterogeneity is not possible with bulk analysis.

1.2.1 Background and Motivation

For a population of relatively static and homogeneous population of cells, bulk measurements of cellular properties provide a reasonable estimate for these properties of a single cell. However, for a dynamic population of cells with varying degrees of heterogeneity, this may not be the case. Therefore, it is intuitively important to analyze these properties at an individual cell level.

Kalisky et al [59] provides an introductory approach into the single cell genomics, by giving a chronological account of the evolution of cellular biology up to the development of single-cell genomics. The authors describe about how single-cell genomics is a preferred alternative over existing methods such as Fluorescence Activated Cell Sorting (FACS) (or even a supplementary method to it) for understanding cell heterogeneity. They further note that the stochasticity observed in FACS studies is due to the stochastic expression of cells, which can be described as a log-normal distribution. Another reason why it is important to obtain a cell level genome is the pragmatic concerns about the fundamental assumptions of homogeneity of cells, i.e., it is often difficult to obtain a sample of homogeneous cells from a living sample. This implies that any bulk measurements will average over the existing cell-types, and therefore the stochastic phenomenon will not be accounted for in such a bulk measurement.

Single-cell gene expressions are an important measurement in single-cell genomic analysis. This takes into account a multitude of genes in a population of cells at the level of individual cells. Kalisky *et al*[59] describe the technical workflows in a typical single cell gene expression analysis. By using reverse transcription polymerase chain reactions (RT-PCR), tens to hundreds of microRNA(mRNA) fragments from a single cell are detected. The throughput of these processes are increased by using microfluidic arrays to simultaneously execute these procedures.

Single Cell Genome Sequencing takes a different approach than the single cell gene-expression analysis. Rather than amplifying the mRNA signals, the cells are sent through an optofluidic sorting process, and then a DNA sequencing process. This is a relatively new method owing to the maturity that single-cell technology is achieving.

However, in our study, we are focusing more on what follows the experimental gathering of measurements. Single-cell experiments often result in enormous amounts of data, the analysis of which requires rigorous methodical approaches.

Throughout literature, various approaches have been employed in this analyses. Marguerat and Bähler, in their work[60] identify the importance of the data analysis in RNA-Seq analysis, when it comes to inferring biological information from the technical experiments. They have detailed how RNA-Seq data can be analyzed to obtain various biological information, such as investigating the post-transcription gene regulation in cells, and the breadth and depth of genome transcription. They also emphasise the fact that RNA-Seq is at least as accurate as microarrays given that the sequencing depth is adequate.

While there may be various applications in biological sciences where our methods can be applied, we demonstrate our approaches with a more confined set of domains. We describe these domains in the following.

1.2.1.1 Haematopoiesis and Cell Lineages

Haematopoiesis concerns the cellular mechanisms of the formation of blood in mammalian systems. Studies show that a single progenitor (the Haematopoietic Stem Cell) gives rise to about 10 different types of cells in a mammalian circulatory system [61]. Analysing the singular cells taken at each stage of the evolution of these cells provides insight into what drives the differentiation between different groups or fates of cells[62].

There are two main kinds of heterogeneity we expect to see in an analysis:

1. Discrete Groups of Cells: These represent different kinds of cells, such as Dendritic Cells, Lymphocytes, Monocytes etc.
2. Smooth Trajectories: These represent the continuous evolution from one type of cell into another, thereby showing a smooth transition between cell types.

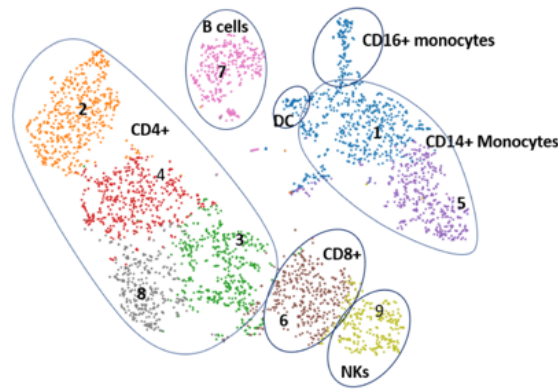


FIGURE 1.17: Different Groups of PBMC cells visualized using t-SNE on a two-dimensional map [7] (CC BY 4.0).

In Figure 1.17 (reproduced from [7] under CC BY 4.0) we see different types of cells being clustered into different groups in the visualization. However, considering the continuous evolution of one type of cell into another, we expect to see a smooth trajectory in its visualizations. Cannoodt *et al* [63] provides overview in to the single-cell trajectory inference, as the sequential steps: A) sc-RNA Seq/ sc-PCR of snapshot of a population, B) Single-Cell Expression Matrix calculation of Gene Counts (where each cell has a number of genes expressed) and Finally C) Pseudotime inference on visualizations. The final step of visualization + pseudotime inference is illustrated in Figure 1.18 (Reproduced from [8] under CC BY 4.0).

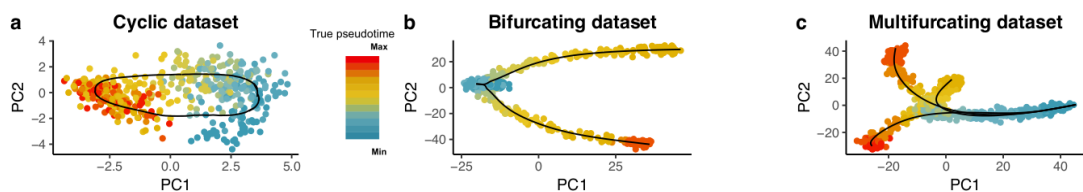


FIGURE 1.18: Three different types of pseudotime trajectories are identified [8]. a) Cyclic pseudotime (where cells express genes in a repetitive manner), b) Bifurcating pseudotime, where the cells evolve in two different trajectories with a single origin, and c) Multifurcating pseudotime, which as multiple trajectories originating from a single origin (CC BY 4.0).

We see that visualization of single cell data is an important step in analyzing trajectories [8, 64–66], and therefore we emphasize the need of robust and consistent visualization methods.

1.2.1.2 Differential Expression of Genes Across Heterogeneous Cell Populations

In cases where different clusters or smooth yet meaningful heterogeneity is observed in visualizations of Single Cell RNA-Seq data (where each cell is measured for the counts of thousands of expressed genes), it can be reasonable to assert that such heterogeneity arises from the differential expression of genes throughout the dataset. Which means different levels of expressions of specific genes in different groups of cells of genes give rise to the different clusters in the visualizations. In fact, some of the earlier studies suggest that the expression of genes in a single cell is a stochastic procedure [67]. However, since the ground-truth of the group-identities are often not known, we conversely use the differential expression of genes to identify cellular groups based on our prior knowledge on differential expression patterns.

Early statistical analysis [68] has revealed that the expression counts can be expressed as an empirical probability distribution. Robinson *et al* [69] in their widely used work edgeR assumes that this distribution is a Negative Binomial distribution. Similar statistical assumptions and analysis can be seen in other classical differential expression analysis platforms such as DESeq [70].

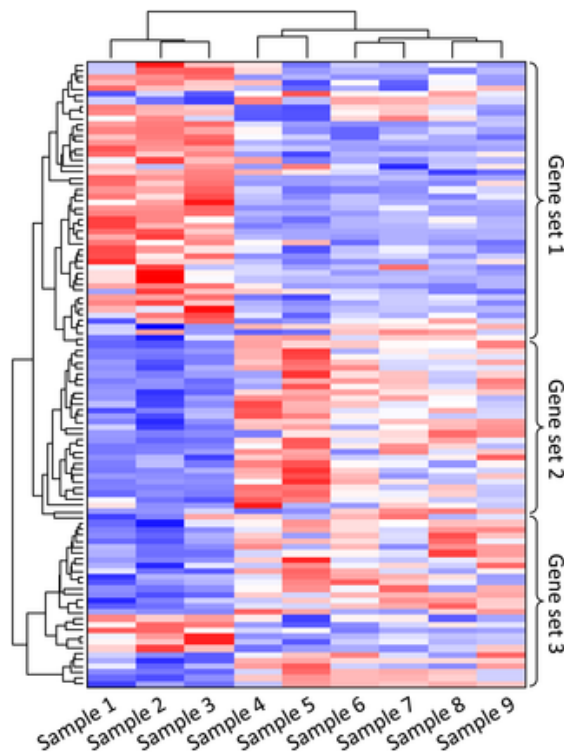


FIGURE 1.19: Heatmap visualizing a cluster analysis of a set of samples, and sets of genes with high differential expression [9]. Each cluster of samples (columns) have similarly expressed profiles of genes (rows). A hierarchy clusters can be inferred from the relative similarity of the gene expression profile of each sample pair (CC BY 4.0).

However we note that the differential expression of genes as shown in Figure 1.19 (reproduced from [9] under CC BY 4.0) is done for bulk samples. While these techniques can be used in Single Cell RNA-Seq analysis as well [71], we look further for more sensitive methods that capture the high-resolution heterogeneity [72] more accurately.

While there are many methods in existence that study the differential expression of genes through a set of statistical tests and models such as SINCERA[73], edgeR[69], DESeq2[74], DESingle[75], etc., we investigate on recent promising techniques based on Deep Learning. The rationale for this selection is that we identify Deep Learning methods, especially given recent advances in them, are more robust and can provide comparable results with less assumptions and that being parametric models, they can be used to generalize experimental observations across multiple experiments.

Since identification of the model of the Single Cell data is an important part of differentially expressed gene analysis, we note recent work by Eraslan *et al* [76], where they used deep autoencoders to denoise single cell RNA Seq data. Furthermore, the use of deep learning to model gene expression levels have been used in literature [77–80] throughout recent years.

1.3 Motivation and Research Questions

Considering the existing literature on Exploratory Data Analysis (EDA) of high dimensional data using neural network methods, I categorize my contributions throughout the PhD into 2 main areas.

1. Identification and Interpretation of the structure of highly dimensional data using neural network methods:
 - (a) Improvement of existing methods (Research Question 1) and Development of Novel methods (Research Question 2)
 - (b) Application of highdimensional structural analysis using developed methods to RNA-Seq data analysis (Research Question 3)
2. Identifying motifs in high-dimensional features themselves that underpins the structure of the high-dimensional data (Research Question 4).

1.3.1 Structure of Data vs Motifs of Features

The first three research questions in this work addresses the identification of complex nonlinear structures in the input data, and the last research question is focused on

identifying the factors that contribute to the structure of the high-dimensional data. To elaborate this difference, I take the example of a set of hand-written digits being analyzed, which is known as the MNIST[81] dataset.

The MNIST dataset has 60000 training images of handwritten digits, each having 28×28 pixels. The dataset therefore consists of 60000 entries of 784 dimensional vectors, and each element in the vector represents the pixel intensity of the corresponding pixel.

Using a nonlinear dimensionality reduction method to visualize these images without using the associated label provides information about how the digits are distributed [20].



FIGURE 1.20: Visualizing the MNIST dataset using UMAP shows that similar digits are clustered together, and different clusters are separated in the visualization.

As shown in Figure 1.20, analysing the distribution of the data provides us information about the different groups and heterogeneity between these groups. The first research questions in my research focuses on improving methods that gives us similar information. Especially in my third research quesiton, we consider extracting canonical representations of heterogeneous groups, which can be analogous to a 'printed digit', which stands in as a prototype for all hand-written digits.

However, simply by visualizing, we cannot obtain what features (or which collections of pixels) contribute to creating a different digit. In other words, we do not get any information by simply visualizing what feature sets can be considered locally meaningful. However, in literature, using unsupervised neural networks to reconstruct the MNIST dataset have shown that the learned weights of the neural networks can show patterns that resemble meaningful feature sets[82].

Recall that in Figure 1.14, we see that different neurons in the hidden layer have identified a set of minimal features that can be identified as the ‘pen-strokes’, which, when combined, forms a different hand-written digit.

Motivated by this observation, in my fourth research question, we attempt to extract the ‘genetic pen-strokes’ that provides the necessary characteristics for one group of cells to differentiate from another.

Chapter 2

How can the SOM algorithms be improved to represent non-uniform densities in the input space better?

Overview

In this chapter, we propose a new algorithm called the Gap Encoding Growing Self Organizing Maps (GEGSOM). It builds on the Growing Self Organizing Map (GSOM) algorithm and improves its cluster visualization and density representation. The proposed algorithm shows significant improvement over the GSOM in terms of visualization of high dimensional data which we show using the MNIST and Fashion MNIST datasets. The proposed algorithm also shows comparable visualizations to the UMAP and t-SNE algorithms for these datasets.

2.1 Introduction

We have discussed in our Literature Review, that the fundamental principles of the Self Organizing Maps govern the density representations provided by them. However, we identify two main disadvantages associated with the SOM and its variants such as GSOM.

1. The Regular Lattice Structure limits the representation of local topologies, and muddles local density variation
2. The quantization error that incentivises the placement of coding vectors in highly dense regions in the input-space loses the relative global placement information

overall, both these issues represent the inability to capture the *density variation* in a robust manner. While the overall ‘manifold’ may be well understood by a SOM algorithm, it does little to help an analyst understand more nuanced structural information.

There are two very broad approaches we can take to attempt to solve the overarching issue of poor capturing of density information:

1. Place the Coding Vector such that it is uniformly embedded in the ambient input space: This would ensure that any non-uniformity of the data is captured as ‘non-coding vectors’ in the output maps, which will appear as ‘empty’ regions.
2. Adjust the graph structure of the coding vectors and adjust the embedding in the output space such that the graph indeed represent the differences of densities in the input space.

The SOM and GSOM algorithms gain a benefit in terms of efficiency due to the fixed grid structure in the low-dimensional space. This is because the neighborhood distance

calculations, which requires frequent and repetitive calculations of pairwise distances is often highly efficient in low-dimensional spaces than in high-dimensional spaces. Mainly due to this reason, in this chapter, we describe and propose an approach attempting to follow the first approach, i.e., a uniform embedding of coding vectors in the input space such that highly differentiated densities are adequately represented.

Gap Encoding Growing Self-Organizing Maps for Dimensionality Reduction and Topology Preserving Visualizations

Damith Senanayake, Chalini Wijetunge,
Malin Premaratne and Saman Halgamuge

May 7, 2020

Abstract

We propose Gap Encoding Growing Self Organizing Maps (GEG-SOM); a new dimensionality reduction method built on the principle of self-organizing map (SOM), and its variants. We introduce several modifications to the original SOM algorithm. First we modify the loss function in the SOM and we remove constraints on the structure of the output visualization. The GEG-SOM attempts to emulate the density variation in the input dataset by encoding the low-density regions in the input space, as well as density proportional node placement in the output space. We use fifteen representative simulated datasets with varying numbers of clusters, and standard deviations, and three real datasets for comparison of GEG-SOM with the existing methods. Through experimental results, we demonstrate that the GEG-SOM has drastically better visualizations to those of SOM and its variants, and superior or comparable performance to the most performant methods; UMAP and t-SNE. We further demonstrate that the GEG-SOM has better performance to those in presence of noise in the input data due to the lack of assumptions in the construction of the algorithm. The experimental results show an improvement in Adjusted Rand Index Scores of up to 24% in simulated data and up to 14%

for real datasets.

1 Introduction

High-dimensional data is a commonly encountered reality in modern data collection processes, where a large number of features or variables are gathered through experiments and records. To comprehend the complex spatial topology of this high-dimensional data, human analysts would desire a two or three-dimensional visualization. Therefore, dimensionality reduction is used as a fundamental and popular exploratory analysis step in these analysis tasks [1]. The areas of applications that encounter this reality is vast, and can vary from computer vision to molecular and cellular biology.

A popular example in computer vision is projecting a set of facial images into a low-dimensional map [2]. Suppose the scenario where facial images having a low intrinsic dimensionality of 3; two positional variables and one azimuthal lighting direction. It is helpful for analysts to reduce images of 4096 pixels (64×64 pixels) into these few useful dimensions.

An example in the field of computational biology that has been gaining attention recently is the single-cell RNA-Seq analysis [3]. One useful area of study using single-cell RNA-Seq analysis is identifying the gene expression differences in tumor and normal cells. The typical workflow consists of obtaining the read counts of a large number of genes, and projecting them on to a two-dimensional map to identify the structure of the collected data [3]. This is an especially useful approach to the study of taxonomical classification of cells in organs using transcriptomics [4, 5] and developmental cell lineage studies [6]. In the subsequent subsections, we describe the dimensionality reduction problem, and examine some of the current methods that address this problem.

1.1 Existing Methods

Dimensionality reduction preserves important topological information whilst reducing the extent of the data. When formally defining the optimization objectives, one needs to consider multiple definitions:

- How are inputs related to each other (Input Similarity/Distance metric and Topology)?
- How are outputs related to each other (Output Similarity/Distance metric and Topology)?
- What is considered to be a well-preserved Topology (The mapping of input to output and loss of Topology)?

Most techniques consider a well-preserved topology to be consistent in distance metrics of input and output spaces. This means when topology is well preserved in a dimensionality reduction, the following can be observed:

- The input points that are located close to each other are mapped close together in the output space.
- The input points that are located far from each other are mapped to distant points in the output space.

In this section, we describe some of the techniques that are used for dimensionality reduction, and we attempt to identify the definitions of Similarity / Distance metrics in high and low dimensional representations of these methods, the mapping technique as well as describe what is considered as a "well preserved topology" in each of these methods. We discuss the construction, strengths and weaknesses of the most commonly used methods such as Principle Component Analysis (PCA), Multi-Dimensional Scaling (MDS) and t-SNE (t-distributed Stochastic Nonlinear Embedding), and a more recent yet promising method, Uniform Manifold Approximation and Projection (UMAP) followed by a critical discussion on SOM and its variants and a summary of our key contributions

1.2 PCA

Being one of the most fundamental dimensionality reduction methods, given a dataset $X \in \mathbb{R}^D$, (D being the dimensionality of the data), PCA [7] uses linear transformations to find a subspace that maximizes the sum of the squared distances of the projections on which is maximum. Data is reduced to a dimensionality $0 \leq d \ll D$ using a linear transformation matrix W as per equation 1.

$$Y = WX \tag{1}$$

where W is chosen to maximize the variance of Y . Intuitively, since Y is derived from X , all variance in Y must be derived from X , and therefore, the W that maximizes the variance of the set Y maximally explains the variance of X .

One major shortcoming of this method is that PCA employs a linear transformation. Therefore, a transformation that reduces the dimensionality drastically will have to use a large number of linear components to adequately explain the variance of the input set. For visualization purposes, this is problematic, as we need at most 3-dimensions to visualize the input.

Because of this, PCA quickly falls out of favor for visualization tasks, while it still remains a highly useful preprocessing technique to reduce the dimensionality with no significant loss of structural information.

To overcome this reliance on linearity of the data, nonlinear dimensionality reduction methods such as the following are used.

1.3 MDS

In MDS [8], the distances in the high dimensional space are defined as the pairwise distances in the input space for the input dataset. The algorithm then attempts to mimic these distances in the lower dimensional representation to obtain a visualization. Therefore, the objective is to minimize a *stress* between

the two pairwise distance matrices in the high and low dimensionality representations, defined in equation 2.

$$S_D = \frac{\sum_{i=1:n} \sum_{j=1:n} (d_{ij} - \hat{d}_{ij})^2}{\sum_{i=1:n} \sum_{j=1:n} d_{ij}^2} \quad (2)$$

where d_{ij} is the distance between the i th and j th inputs of the dataset, and $\hat{d}_{ij}$ is the distance between the corresponding points in the output map. In the original version of MDS, d_{ij} is the Euclidean distance.

The problems with this method is that it assumes a globally consistent topology can be obtained using a lower dimensional output. This falls apart when presented with a set of roughly equidistant points on a high dimensional space.

An intuitive explanation of this is attempting to reduce an n -simplex with $n > d + 1$ where d is the output dimensionality. For instance, one cannot maintain the equidistant relationship of more than 3 points on a 2 dimensional output space.

As a solution to this, there are variants of MDS that uses non-global and non-euclidean pairwise distances instead of the euclidean distances. Local Multidimensional Scaling [9] uses small localities of euclidean pairwise distances to approximate the global manifold, and IsoMap [2] uses a graph constructed using an all-paired shortest path algorithm which serves as a geodesic distance measure which provides a better reduction of dimensionality than the original MDS algorithm for nonlinear manifolds.

1.4 t-SNE

A widely used method that provides descriptive and consistent visualizations is t-SNE [10] which builds up on the principle of Stochastic Nonlinear Embedding(SNE) [11], where instead of defining a distance metric for the high dimensional neighbors, we define a probability distribution for the neighbors, of any given input being a neighbor of another input. These distributions are first approximated as Gaussian distributions in the following manner.

$$p_{ij} = \frac{\exp(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2})}{\sum_{l,k} \exp(-\frac{\|x_l - x_k\|^2}{2\sigma_l^2})} \quad (3)$$

where p_{ij} is the probability that the point x_j is in the neighborhood of point x_i in the input space. We then attempt to place the corresponding output vectors such that the probability distribution of the outputs matches the probability distribution approximated for the input. The output probability distribution is assumed to be a student-t distribution to accentuate the difference between clusters, thereby obtaining a better cluster separation.

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k,l} (1 + \|y_k - y_l\|^2)^{-1}} \quad (4)$$

Where q_{ij} is the probability that y_i is in the neighborhood of y_j . The disparity between the two probability distributions is calculated using the Kullback-Liebler Divergence metric. This divergence can be minimized using a gradient descent approach.

One of the underpinning assumptions in the t-SNE algorithm is the assumption that a probability distribution, specifically a Gaussian distribution, can be approximated for any given neighborhood in the input space. Therefore, t-SNE tends to be problematic in the presence of considerable amount of noisy data points.

1.5 UMAP

Uniform Manifold Approximation and Projection [12] provides comparable or better results to t-SNE in most real datasets, and uses a dimensionality reduction technique based on Riemannian geometry.

Similar to t-SNE, UMAP first constructs a collection of neighborhoods by assigning memberships to the neighboring nodes in a fuzzy simplicial set. Then the fuzzy membership values are used to construct the output graph by minimizing the cross-entropy between the fuzzy membership values of the output map.

One major assumption that is made in this technique is that there exists a Riemannian manifold on which this data lies. Since in reality, it is highly unlikely that the data is uniformly distributed in a real world data manifold, it is assumed that the Riemannian metric is not inherited from the ambient space. This assumption justifies the construction of fuzzy simplicial sets made up of k -simplices, where k is a user-defined parameter.

The use of cross-entropy to calculate the disparity between the two membership functions in UMAP, as opposed to t-SNE where the KL-Divergence metric is used, enforces the gaps between clusters to a greater degree, providing better cluster separation while maintaining the topology of the input in the local neighborhoods.

Through the analysis of the mathematical underpinnings of the two methods, one can draw parallels between UMAP and t-SNE. Where t-SNE constructs a stochastic membership function for a neighborhood of a given input vector, UMAP constructs a fuzzy membership functor based on the dissimilarity of the neighbors. They both construct the high-dimensional graph first, and optimize the low-dimensional embedding as a following step.

While t-SNE and UMAP both construct k -nearest neighbor graphs (stochastic graphs in t-SNE and simplicial sets in UMAP), only the paper on UMAP [12] provides a mathematical justification with the assumption that the data lies on a Riemannian Manifold, on which, a spherical point set with the k -nearest points is distributed with a uniform density. The key difference in the two methods is in the generality of the optimization technique used by UMAP, as it does not rely on a large difference of dimensionalities between the input and output as t-SNE does. Instead, the objective function used by UMAP considers the similarity of non-neighbor dissimilarities, in addition to the neighbor similarities when optimizing the low-dimensional embedding. This may give UMAP some advantage over t-SNE.

In practice, it is observed that t-SNE and UMAP are highly performant methods compared to other methods in use, and they provide practical insights into the structure of the data. In the next section, we describe how we leverage

strengths of the self organizing maps and its variants to provide visualizations comparable to and sometimes superior to these two methods.

1.6 Self-Organizing Map

Self Organizing Maps (SOM) [13] attempt to provide an embedding in a two-dimensional map by training a set of coding vectors, to mimic the input in the higher dimensional space. This is limited in its capacity since the size of the map is fixed, and the two-dimensional map has a fixed grid structure of a square or hexagonal lattice, and finding the appropriate size of the map is a main issue in this algorithm. The said coding vectors, given as the set W , are in the same dimensionality as that of the input set, and are augmented randomly in the input space at the beginning. As training progresses, these coding vectors are adjusted to represent a Voronoi diagram in the input space, which spans over all inputs.

The topology preservation in the SOM is achieved by defining a neighborhood function for the coding vectors. Each coding vector is associated with a coordinate in the lower dimensional manifold (typically a 2-D grid). When reducing an input $x_i \in X$ with a trained SOM, the centroid coding vector of the Voronoi region to which that input belongs, $w(x_i)$ is identified, as per Equation 1;

$$w(x_i) = \underset{\forall w \in W}{\operatorname{argmin}} \|x_i - w\|^2 \quad (5)$$

The mapped point, therefore, is taken to be the lower dimensional coordinate associated with $w(x_i)$. However, it is noteworthy that the analyst needs to know the number of coding vectors *a priori*. A simple illustration of this idea can be found in Fig. 1.

In this figure, the three colors depict the associated clusters. The scatter plot at left shows the three clusters distributed in the three dimensional input space. The black dots represent the coding vectors placed in the input space. Each connecting edge represents a neighbor relationship. The grid on the right represents the coding vectors in 2-d. The colored dots map the higher dimen-

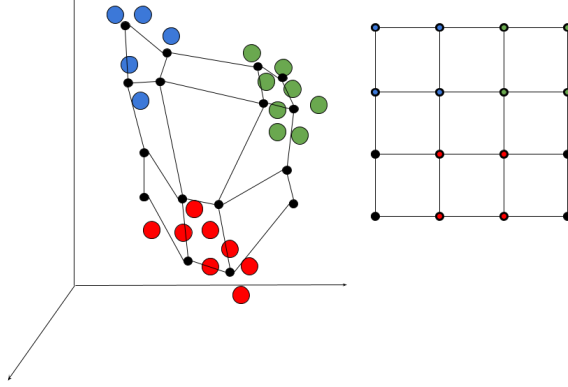


Figure 1: Illustration of a trained SOM. In the graph on the left, the colored points represent the input set. Each color indicates a separate cluster. The grid on the Right represents the output. The black connected points on the left are the neuron weights, with each connection representing the relation between the neighboring neurons in the output grid. The output grid is colored with the corresponding shade of the cluster to which each mapped input point belongs.

sional input to this 2-d space. Note that, if the colors were ignored, in the diagram on the right, an analyst would fail to clearly identify the associated mapped input clusters. Moreover, by varying the initial size of the map, one can have different mappings at different granularities. However, it is generally observed that relatively empty regions, such as the region between the blue and green clusters in this diagram will not be observed with a SOM.

1.7 The Growing Self-Organizing Map

Alahakoon *et al.* proposed a dynamically growing version of the SOM; Growing Self Organizing Maps(GSOM) [14], which attempts to overcome the issue of the unknown map size by using a dynamically growing grid instead of a fixed size grid. This growth is controlled by keeping track of the quantization error (the sum of deviations from the centroid of a Voronoi region of all the input points that fall within that region) of individual nodes in the map.

Since our work builds up on the principles of the Growing Self Organizing Maps, we provide a brief description about the operation of the algorithm in this section.

The GSOM algorithm has two phases; the Growing Phase and the Smoothing Phase. Each of these phases are described below.

1.7.1 The Growing Phase

In this phase, the GSOM algorithm approximates the size of the map required to quantize the input data. This is done by starting with a small number of inputs, and expanding as the input data is presented to the algorithm. To control the growth of the nodes, a metric of cumulative quantization error is kept. Since the goal is to minimize the overall quantization error, the rationale is to reduce the quantization error of individual neurons by increasing the number of coding neurons in the Voronoi regions with high errors. For the simplicity of managing the said growth, the assumption is made to maintain the graph structure of the low-dimensional map. The implication of this being that the weights of the neighboring neurons of a map will be connected in the input space. When each input is presented, first, the algorithm finds the best matching unit, i.e. the neuron that codes the Voronoi region which includes the input space. The quantization error for each individual neuron is then recorded against the best matching neuron. These steps are described below:

1. Present input x from input dataset X . Find the best matching unit b from the set of weight vectors W such that;

$$b = \underset{\forall i \in \{1 \dots |W|\}}{\operatorname{argmin}} \|x - w_i\|$$

2. Record the quantization error for b , (Err_b) as the Euclidean Distance from the input such that;

$$Err_b \leftarrow Err_b + \|x - w_b\|$$

where w_b is the weight of the best matching unit.

3. Move the weight vector of the coding vector towards the input ;

$$w_b \leftarrow w_b + \alpha_t(x - w_b)$$

where α_t is the learning rate at time t in the training process.

4. To enforce self-organization, the neighborhood of b is moved by smaller amounts towards the input x .

$$\forall j; \|y_j - y_b\| \leq r_t$$

do

$$w_j \leftarrow w_j + \alpha_t h_{bj}(x - w_j)$$

where y_b is the associated position vector of the best matching unit, r_t is the neighborhood radius for given time t , and α_t is the learning rate at time t . h_{bj} is the neighborhood function value assigned for neuron j in the neighborhood of b .

5. If $Err_b > GT$; conduct error distribution on node b . Here, GT is a userdefined parameter which governs the growth of the map known as the "Growth Threshold". The Growth Threshold is defined as a function of the hyper-parameter spread-factor(SF), the dimensionality of the Data(D), and the order of the minkowski norm (p) we're considering in the distance metrics [15], as per the Equation 6.

$$GT = D^{1/p} \ln(SF) \tag{6}$$

The error distribution is done to neighboring nodes, by increasing their respective errors by a predefined percentage. If b is a boundary node, i.e. one or more neighbors are missing, new neighbors adjacent to b are added.

6. Draw another input from the input set X and repeat steps 1-5 till optimization goal reached.

1.7.2 The Smoothing Phase

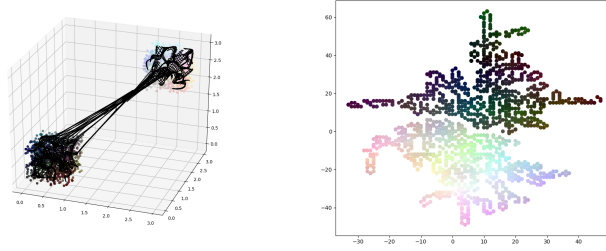
While this growing phase estimates the required size depending on the granularity desired by the analyst, it is also noteworthy that this can be equivalent to initializing a traditional self organizing map with the weights already well-organized or semi-organized, thereby leaving only minor refinements to be done. This refinement is done by using a smoothing phase, which is equivalent to training a self-organizing map, with the exception of the training neighborhoods being limited to the immediate neighbors.

The main advantage of the GSOM over other methods is that GSOM does not assume the underlying probability distribution explicitly, but rather approximates it by self-organization. This is a significant advantage as real-world data typically contains a considerable amount of distortion and noise, making them violate the assumptions of specific probability distributions. In addition to this, the GSOM algorithm can obtain compact representations of data at different levels of granularity. This is often done by using a single representative output for a number of similar inputs. This granularity is regulated by the spread factor parameter.

1.7.3 Weaknesses of the GSOM

While in principle, the GSOM has the fundamental idea of approximating a low-dimensional manifold and describing the variance along the said manifold, in practice, the visualizations obtained from it tend to be of poor quality compared to t-SNE and UMAP. The main weakness, which may render these visualizations comparatively less informative, is the lack of gaps between distinct clusters of data. We attribute this failure to visualize gaps, to the lack of an incentive to encode gap regions in the input space. We demonstrate this using a simple dataset with two clusters of data, as shown in Fig 2. The diagram on the left

(a) shows a 3-dimensional input set, while the diagram on the right (b) shows the visualization of this set using the GSOM algorithm. One can observe that the GSOM has placed its coding vectors successfully within a dense region in the input, by noting the organized nature of the 'mesh' that is embedded in the input space. Furthermore, we note the consistency of this organization by noting the smooth gradient of colors in the two clusters shown in the diagram (b).



(a) The coding vector embedding in input space (b) Output map of input space embedding

Figure 2: Dimensionality reduction using a conventional GSOM. Each vertex in the graph shown in black, is a coding vector. Graph connections are shown as black lines. The associated positions of the coding vectors are shown in (b) as a 2-D vector. The inputs are mapped to the best matching unit in the embedding space.

However, the region separating the two clusters does not contain a large number of coding vectors in (a) which is reflected by the lack of a 'gap' in the diagram Fig. 2(b). While in this trivial example, we can empirically observe the two clusters based on the two color gradients, for purely unsupervised cases, we do not have the label information that would provide us the information to separate two clusters in the input.

This problem, contributes furthermore into the so-called crowding problem; as identified by Vandermaaten *et al* [16], since multiple clusters can be mapped into a confined region in a map, and equidistant relationships between cluster centers can provide visualizations with overlapping clusters. This becomes es-

pecially apparent with real datasets with distinct clusters that become difficult to separate with the GSOM algorithm, as is evident by the Fig. 3.



Figure 3: Visualization of the first 6000 instances of the MNIST dataset. The cluster boundaries in this diagram are fuzzy and unclear. However, we observe that the local topologies of the dataset is preserved by the fact that points with similar colors are placed at close proximities to each other.

In summary, while PCA and MDS both do not make assumptions about the probability distribution of the data, they each fail due to limitations inherent to each of them; PCA due to the linear nature of the projections, and MDS due to the curse of dimensionality. Having superior performance in identifying the non-uniform density variations of the data and overcoming the curse of dimensionality and linear limitations, t-SNE and UMAP both rely on assumptions of the probability density functions in a given neighborhood. SOM variants are impervious to the curse of dimensionality better than MDS, and overcome the linear projection limitation of PCA. As an added advantage, SOM variants rely on less assumptions about the neighborhood probability densities relative to t-SNE and UMAP. However, they are less capable of identifying the low-density regions in the principle manifold identified by the algorithm. It is therefore our motivation, to attempt and overcome this difficulty of SOM variants to encode low-density regions and density variations with better sensitivity.

1.8 Key Contributions

In this work, we identify two major shortcomings of the visualizations generated by the SOMs and their variants, and attempt to overcome them in the proposed algorithm named the Gap Encoding Growing Self Organizing Map (GEG-SOM). The following subsections describe these two aspects in detail.

1.8.1 Addressing the failure to encode low probability density regions in the input manifold.

Vector Quantization methods rely on the presence of input vectors to approximate the output map arrangement in the input space. Therefore, when the probability density of a region is comparatively low, the number of required coding vectors become smaller. While this is a useful technique to limit the overall size of the map, a problem is presented when encoding comparatively low-density regions within the complete dataset. This topology distortion is a commonly encountered problem in self organizing maps. Fritzke [17] in his comparison of Growing Neural Gas to Self Organizing Maps, emphasizes this distortion as a problem inherent to SOMs.

1.8.2 Removing limitations exerted by the map structure

The Self Organizing Maps employ regular grids to map its output onto. This makes the task of regulating the growth of the map, while maintaining the high-connectivity of high-density regions of the trained map, challenging.

The key focus of GEG-SOM is to maintain the high-dimensional topologies in an output map, while accentuating the placement of different clusters in the input space. We achieve this by eliminating the above two shortcomings with the following approaches.

1. Gap Reinforcement by introducing a new loss component to the objective function which we dub the Gap Enforcement Error; thereby penalizing the non-uniform placement of nodes in the input space.

2. Introduction of non-regular grids as a viable alternative to the regular grids (triangular, hexagonal or square) for better growth regulation and visualizations; thereby incentivizing non-regular placement of nodes in the output space reflective of the non-uniformity of the input space node-placement.

Sections 2.1 and 2.2 detail our two-fold approach.

2 Method

Through our method, we attempt to obtain as uniform a node-placement in the input space as possible, while obtaining a map reflective of the residual non-uniformities. There are two main contributions that aid us in achieving this.

2.1 Self Organizing Manifold with Gap Reinforcement

Consider the problem of obtaining a topologically consistent visualization of high-dimensional data with a dimensionality of D into an output space of dimensionality d , where $d \ll D$. The success of such a projection hinges on several pivotal definitions as described in Section 1.1.

In the case of self organizing maps, these three are defined as below.

1. High Dimensional Metric: The euclidean distance between coding vectors
2. Low Dimensional Metric: The number of nodes separating two coding vectors corresponding to two separate inputs. This can also be calculated as the euclidean distance on a 2-d plane.
3. The total Quantization error which indirectly calculates the proximity of matching coding vectors to high dimensional neighbors.

With this selection of metrics, the Self Organizing maps excel at identifying local topologies consistently, provided that the distribution is relatively uniform.

The loss function incentivises the placement of coding vectors in the regions where there is a large number of input vectors to be encoded. Owing to fuzzy neighborhood selection of the algorithm, we end up having a smooth manifold stretched over the uniform data.

However, one downside of this algorithm is that there is no metric to account for the degree of separation between high density regions in a non-uniform data distribution. i.e. there is no incentive to place separating nodes between two regions that separate two clusters of inputs.

The Distortion of the topology mapping described in the work by Fritzke [17] can be resolved by including a number of idle nodes (which serve no purpose in encoding the input space other than to represent the degree of separation between two relatively far apart regions).

However, due to the self-organizing property of the GSOM algorithm, two significantly separate regions of high probability density in the input space, will be separated by only a small degree of separating neurons. The reason for this is that the relatively low probability density regions do not enforce uniformly spaced neurons in those regions, as there are only a few inputs will be forcing neurons to be held in those regions.

In the following section we propose an additional error term to the optimization procedure, to incentivize the placement of coding vectors in relatively empty regions in the input space, while not violating the approximation of a principle manifold.

2.1.1 The Gap Enforcement Error

The conventional GSOM builds upon the quantization error (E_Q) of local neighborhoods of input vectors as described in Kohonen’s work [13] given by the following equation.

$$E_Q(X, W, Y) = \sum_{i=|X|}^n \sum_{j=|W|}^p \|X_i - W_j\|^2 N(B(W, X_i), Y_j) \quad (7)$$

where;

$$B(W, X_i) = \operatorname{argmin}_{W_k \in W} (\|X_i - W_k\|) \quad (8)$$

and

$$N(p, Y_j) = \exp\left(-\frac{(\|Y_j - Y_p\|^2)}{r_t}\right) \quad (9)$$

For minimizing this error with gradient methods, one can calculate the gradient as below:

$$\Delta W_j = N(B(W, X_i), Y_j) \times (X_i - W_j) \quad (10)$$

Since function N is independent of W and B is not differentiable, the gradient is simplified.

One shortcoming of this error is that while it accounts for the local topology preservation, it doesn't incorporate elements of global topology. Therefore, the algorithm relies on the global topology being preserved as a product of a diminishing locality considered when training the algorithm. This inherently fails to proportionally represent the global topological dispersion (i.e. the gaps between high density regions).

We propose the following Gap Enforcing Error (E_G) to account for the global topology representation.

$$E_G = \sum_{i=|X|}^n \sum_{j=|W|}^p H(B(W, X_i), W) \quad (11)$$

where

$$H(i, W) = \sum_{\forall j \in |W|} \exp(-\lambda_D(1 - (D_j))^b) \quad (12)$$

and

$$D_j = \frac{\|X_i - W_j\|}{\max_{w_k \in W} (\|X_i - W_k\|)} \quad (13)$$

Here, λ_D is a positive value which we have set to be 7, as a tentative value. We have determined with simulations that, $\lambda_D \in [1, 20]$ is a reasonable range

for practical usage. Additionally, the Gap Enforcement Error we add, is a regularization term, which acts in conjunction with the Vector Quantization Error. Hence it is not expected to stand alone as a metric.

2.1.2 Estimation of the Gap Encoding Function, H

The function defined in Equation 12 requires the estimation of variable b , which is a function of the intrinsic dimensionality of the dataset. Since the estimation of the intrinsic dimensionality is a costly operation, we propose a parameter which varies with the severity of the effects of the curse of dimensionality of a dataset.

We identify that due to the curse of dimensionality, data with relatively high intrinsic dimensionality would have highly discernible distances to the neighboring points, but indiscernible large distances to distance points. Therefore, by normalizing the distances, we can obtain a curve of following category.

$$\hat{D} = 1 - \exp(-un^2) \quad (14)$$

where $\hat{D}$ is the normalized distance in the input space, and n is the position of the input we consider, when sorted in the ascending order of the distances. We estimate u for each input using a simple curve fit, and use this u to derive b in the following manner.

$$b = 1 + a \exp(-d(u/c)^k) \quad (15)$$

Through analysis, we have identified values a , d and c suitable for the data analyzed in this paper. The b value, therefore, can be approximated by the Equation 16.

$$b = 1 + 50 \exp(-4(u/40)^6) \quad (16)$$

The added error term penalizes non-uniform placement of the coding vectors in the ambient space, thereby accentuating the gap regions of the final map.

When tested on the dataset used in Fig. 2, the modifications provide a better visualization as shown in Fig. 4.

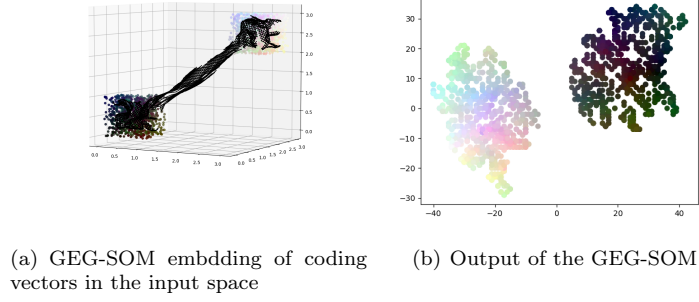


Figure 4: Dimensionality reduction using GEG-SOM. Note that the gap region accommodates a proportional amount of nodes to show the separation of clusters.

2.1.3 Gap Encoding Error vs. Distortion Error

Another strategy used to preserve gaps in a self organizing map is described in Elastic Maps [18] where an error is introduced to enforce the uniform embedding of the manifold in the ambient space that would penalize large distances as well as twisting the manifold to converge on local optima.

In Elastic Maps, the distortion error $E^{(S)}$ and the bending error $E^{(B)}$ are described as below;

$$E^{(S)} = \frac{1}{2} \sum_{\forall i,j \in |G|} \|w_i - w_j\|^2 \quad (17)$$

$$E^{(B)} = \frac{1}{2} \sum_{\forall i,j,k \in |G|} \|w_i - 2w_j + w_k\|^2 \quad (18)$$

In addition to the vector quantization error, these errors are summed to the total error to be minimized. It is noteworthy, that an expectation maximization algorithm is required to minimize this error, and a very large number of iterations is required for convergence. The gap enforcement we present here,

can be optimized using a small number of iterations (20-100) of a simple gradient descent algorithm. The adjustment values we produce in this document, is therefore in keeping with the gradient descent optimization method.

2.2 Use of Non-Regular Grids for Output Density Variation

The intuition behind the use of non-regular grids is to obtain a non-uniform output map as opposed to the regular grids. A regular map places the locations of the coding vectors at a uniform density on the output map. A non-regular map incentivises the growth of the map to have a less-dense output node placement in low-density areas of the coding vectors, as well as a high-density placement of nodes in high-density areas.

To illustrate this, consider figures 5 and 6. In figure Fig. 5(a) we have subjected a map which begins with two nodes far apart to 20 growth cycles. The two regions illustrate node placement with uniform density, while the region with higher growth concentration tends to get larger from the outside (the upper right corner).

The effect this has on the dynamic grid is that to accommodate the rapidly growing map in the upper right region, we must drift the low-density region further away from it. This becomes problematic with a gradient descent approach as an adequate drifting of low-density regions cannot be achieved due to the lack of inputs originating from those regions. Failure to control the placement of the distinct regions of the map at a fitting distance would result in the merging of two regions, losing all density information as per Fig. 5(b).

A more reasonable approach would be to accomodate the growth of high-density regions by populating them with more nodes, while incentivising the placement of nodes on the inside of the map, while growing on the boundaries of the maps as well.

This can be achieved with relative ease by removing the constraint on having a regular grid to represent the output. This does not violate the self-organization

as the proximity relations can still be achieved by considering the relative distances in the output map as per Equation 9. In Fig.6(a) we start with relatively small maps in two distinct regions, but the growth is done by creating 5 nodes at a unit distance around each point where growth needs to occur. As evident by Fig.6(b) The growth has placed nodes based on the density of the growth incidents in a region.

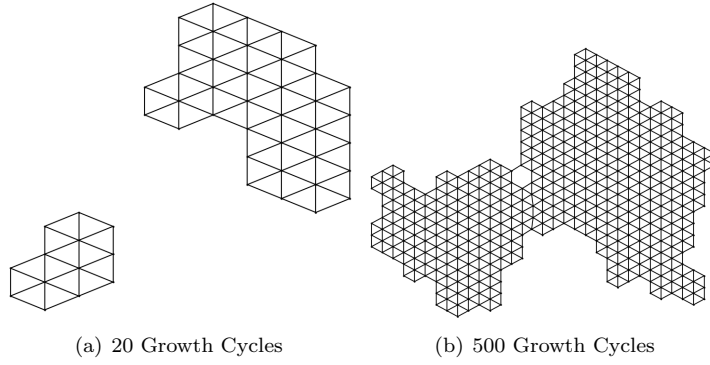


Figure 5: Two regions in a regular grid (hexagonal) subjected to (a) 20 growth cycles and (b) 500 growth cycles. The merging of the two regions preserve the node density information, while the growth happens outward

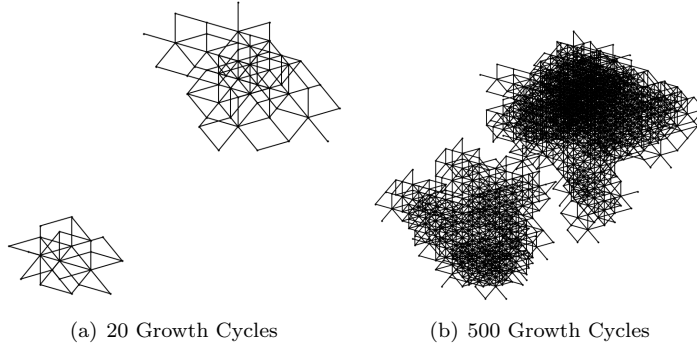


Figure 6: Two regions in a non-regular grid (pentagonal) subjected to (a) 20 growth cycles and (b) 500 growth cycles. The growth of the two regions have maintained the density information by the concentration of growth to high density regions.

This further aids the encoding of empty regions, as evident by the Fig. 6(b),

as the non-regular grid structure maintains a relatively larger distance edge distance among neighboring nodes in low-density regions (the bridging region of the two clusters) while maintaining high connectivity and smaller edge lengths in the high-density regions. *We therefore note that the use of non-regular grids provide means to reflect the density of high-dimensional node placement through the low dimensional node-placement.*

It is noteworthy, that the addition of non-regularity in the grid, does not violate the self-organizing property of the algorithm. The rationale behind this being the neighborhood function used in Equation 9, relies on the distance between two points in the output map, and not on the graph distance (number of edges needed to traverse from one point to the other). Therefore, it is important to note that in the case of non-regular grids the graph-connections, are merely a means to control the growth of the map at a given locality, and not a distance measure.

The downside of using this technique is it increases the training times due to the large number of nodes placed close to each other. Therefore, it is advisable to use non-regular grids where local topologies need to be emphatically preserved. To further aid in shaving off running time, we further propose the following pruning method.

2.3 Node Deletion for faster training times

We introduce a new variable, the age, associated with each node. The age of a node is increased when it has not been selected to be in a training neighborhood, and we delete nodes that are above a predefined threshold age.

The node deletion controls the growth of the map to eliminate idle nodes, and makes the algorithm more memory efficient. Additionally, it prevents the gap nodes from skewing the Gap Enforcement Error to erroneously consider Gap Encoding nodes as Vector Quantizing Nodes.

At this stage, we have a functioning complete algorithm, which can be summarized as per Algorithm 1.

In this algorithm, α_t is the learning rate, which we initiate at 1.0, and reduce with time. γ_t is the gap coefficient, which is typically smaller than α_t . ϕ is the factor of error distribution as per [14].

It is noteworthy, that through this algorithm, we attempt to minimize the quantization error as described in the original work by Kohonen [13]. Our modifications proposed in the Section 2.1 further provide a constraint for the separation of high-density region in the input space. We provide empirical proof for validity of our modifications in the following sections.

3 Data and Experimental Setup

3.1 Data Sets used in this study

We use both simulated and real-world datasets to demonstrate the capability of the proposed method in generating dimensionality reductions with preserved topologies. The simulated data are used to evaluate the performance of the algorithm in the presence of varying levels of noise and numbers of clusters. The real-world data are used to evaluate the performance of the algorithm in complex scenarios with varying structures of input topologies. We use commonly used image datasets (namely MNIST and Fashion-MNIST) to demonstrate the performance of the algorithm compared to other prominent methods. We also use the proposed method to reduce the dimensionality of an RNA-Seq dataset which has a considerably high dimensionality with a low-dimensional manifold that can be inferred from the high-dimensional data. The following subsections describe the datasets used in detail.

3.1.1 Simulated Data

We construct 15 datasets with 9 dimensions, and varying numbers of cluster (10, 20 and 30 respectively) to be found. For each set of clusters, we set the standard deviation of the clusters to take 5 different values (4, 5, 6, 7, and 8). The level of noise and mixing of clusters increase with the standard deviation,

making the separation of clusters more challenging.

3.1.2 Real Data

The first real dataset we use is the MNIST [19] hand written digit dataset. Each image consists of 28x28 pixels (784 dimensions), and the intensities of each pixel is given as a value between 0 and 255. Each image contains a hand-written digit (0-9).

The second real dataset is the Fashion-MNIST [20], which has a similar dimensionality to that of MNIST, but contain images of fashion items. This is a more difficult dataset to differentiate than the original MNIST dataset [21].

The third dataset we use is a single-cell RNA-Seq gene expression dataset published by Nesterowa *et al.* [22]. This contains the Gene Expression Levels of 3 types of cells; Progenitor Cells, Hematopoietic Stem Cells, and Long Term Hematopoietic Stem Cells. The gene expression profiles of 1654 single cells are recorded as normalized values in this dataset.

3.2 Experimental Setup

For datasets where we measure the clustering accuracy, the first step would be to use a dimensionality reduction method to reduce the dimensionality of the dataset down to two dimensions. The reason to reduce to two dimensions is a 2-d visualization of the structure of data is generally feasible for human analysts who study the said structure. We used the most performant methods; t-SNE and UMAP to compare with the the proposed method. Other methods, such as PCA, MDS, and GSOM were not considered in the scores, as they are considerably sub-par to t-SNE and UMAP. Then, we apply k-means clustering to the reduced representation and assign labels to the identified clusters.

In cases where the true labels are known and clear, we use the Adjusted Rand Index and the Adjusted Mutual Information scores to measure the accuracy of the clusterings provided by each method. This workflow is used to obtain the scores summarized in Table 1. This table also summarizes the dif-

ference of scores between the GEG-SOM algorithm, and the most performant of t-SNE or UMAP for each dataset. Therefore, the positive differences mean that GEG-SOM outperforms the better of the two algorithms and the negative differences are when either UMAP or t-SNE is the best performing method. We further show the percentage of the difference, compared to the most performant method out of t-SNE and UMAP, and GEG-SOM. For the real data where the clear clusters are not reasonably well separated, we provide visual proof of the reductions. What we expect from a visualization which has preserved the topology well, is that similar inputs are mapped to points closer to each other, and highly dissimilar clusters are placed far away from each other. Therefore, the tightness of similar class clusters, and the separation of dissimilar clusters is to be expected from a good visualization.

4 Results

Experiments with simulated and real world high-dimensional data demonstrated the soundness of the proposed algorithm in various scenarios. These results are discussed in detail in the following subsections.

4.1 Simulated Data

The Gaussian Blobs datasets we simulated with varying degrees of noise, were reduced to two dimensions using the proposed method GEG-SOM and the two most performant competitors; t-SNE and UMAP. The reductions of GSOM were not compared due to the considerably poor visualizations provided by the GSOM compared to UMAP and t-SNE, as is evident by Fig. 7. The performance of the three methods typically follow the trend with GEG-SOM having mostly the superior performance, UMAP providing competitive results, and t-SNE following UMAP.

We further note that the number of clusters does not contribute heavily to the variation of results between the datasets, but rather the level of noise (indi-

cated by the standard deviation) of a cluster. The quality of the dimensionality reduction using GEG-SOM is quite apparent when the level of noise is high. We attribute this increase in quality to the lack of assumptions that GEG-SOM uses about the structure of the high-dimensional data or its probability distribution, and the denoising effects taking place due to the discretization of input space by vector quantization.

Table 1: The adjusted rand index (ARI) and adjusted mutual information (AMI) scores for the simulated and real world datasets. The percentage improvements are increasingly better at high noise levels as visible from both real and simulated data.

	ARS / AMI (Rank / Rank)			Difference	% Difference
No. Clusters, stdev	t-SNE	UMAP	GEG-SOM		
10 , 4	0.85(3) / 0.87(3)	0.86(2) / 0.87(2)	0.86(1) / 0.88(1)	0.0079 / 0.0038	0.9174 / 0.4329
10 , 5	0.69(3) / 0.73(3)	0.71(2) / 0.75(2)	0.73(1) / 0.76(1)	0.0044 / 0.0092	0.6160 / 1.2273
10 , 6	0.54(3) / 0.61(3)	0.58(2) / 0.63(2)	0.59(1) / 0.64(1)	0.0124 / 0.0033	2.1457 / 0.5202
10 , 7	0.39(3) / 0.48(3)	0.43(2) / 0.51(2)	0.46(1) / 0.51(1)	0.0339 / 0.005	7.9058 / 0.9808
10 , 8	0.31(3) / 0.40(3)	0.33(2) / 0.42(1)	0.34(1) / 0.4096(2)	0.0071 / - 0.008	2.1418 / -1.9531
20 , 4	0.85(3) / 0.88(3)	0.86(2) / 0.89(2)	0.87(1) / 0.89(1)	0.0098/0.0077	1.1430 / 0.8679
20 , 5	0.65(3)/0.73(3)	0.67(2)/0.75(2)	0.71(1)/0.77(1)	0.0357/0.0244	5.2960 / 3.2664
20 , 6	0.45(3)/0.57(3)	0.46(2)/0.58(2)	0.53(1)/0.62(1)	0.0768/0.0450	16.8421/7.7667
20 , 7	0.28(3) / 0.43(3)	0.32 (2)/ 0.46(2)	0.40(1) / 0.51(1)	0.0769/0.0485	24.0613 / 10.6034
20 , 8	0.19(3) / 0.33(3)	0.22(2) / 0.36(2)	0.29(1)/ 0.40(1)	0.0658/0.0455	29.8955 / 12.7308
30 , 4	0.81(2) /0.86(2)	0.81(1) / 0.86(1)	0.80(3) / 0.86(3)	-0.0112/-0.0048	-1.3998 / 0.5593
30 , 5	0.56(3) / 0.68(3)	0.58(2) / 0.69(2)	0.62(1) /0.72(1)	0.0364/0.0228	6.2694 / 3.2825
30 , 6	0.33(3)/0.50(3)	0.34(2) /0.51(2)	0.43(1) /0.56(1)	0.0854/0.0566	25.1029 / 11.1461
30 , 7	0.20(3)/ 0.37(3)	0.22(2) /0.38(2)	0.28(1) / 0.44(1)	0.0681/ 0.0536	31.6010 / 13.9402
30 , 8	0.14(3) / 0.28(3)	0.14 (2)/0.30(2)	0.20(1)/0.35(1)	0.0568/0.0499	40.7461 / 16.8809
Real Data					
MNIST (6K)	0.66(3)/0.74(3)	0.73(1)/0.80(1)	0.72(2)/0.75(2)	-0.0148/-0.0444	-2.0825 / -5.9050
Fashion MNIST(6K)	0.43(3)/0.58(3)	0.44(2)/0.60(2)	0.50(1)/ 0.62(1)	0.0623/0.0226	14.1720 / 3.7692

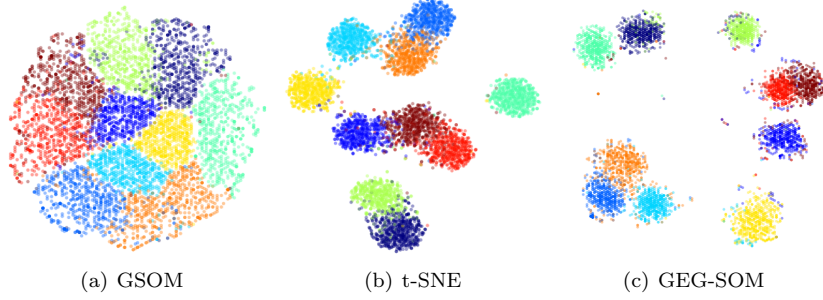


Figure 7: Ten clusters embedded in a 9-dimensional euclidean space as visualized by (a) GSOM, (b) t-SNE and (c) GEG-SOM

4.2 Real Data

We further subjected the datasets MNIST and Fashion MNIST (the first 6000 instances of each dataset) to the same reduction and obtained results consistently better than t-SNE, and comparable with UMAP in Adjusted Rand Index and Adjusted Mutual Information scores.

4.2.1 MNIST

Using the GEG-SOM algorithm on the MNIST dataset, we discover that it has superior cluster separation, with high compression and low noise than compared to both other methods (UMAP and t-SNE), as per Fig. 8. Upon observation, an analyst can see the clearly separated clusters of inputs belonging to the same class. It is further important to note that the hierarchy of the clusters, along with the shapes of the clusters, are consistent across all three methods. We note that the three clusters of green, carrot-orange and sky-blue clusters are separated into distinct clusters, a task at which t-SNE and UMAP both lag behind GEG-SOM. Additionally, we note the lack of noisy points in the GEG-SOM map, due to the vector-quantization of the input space.

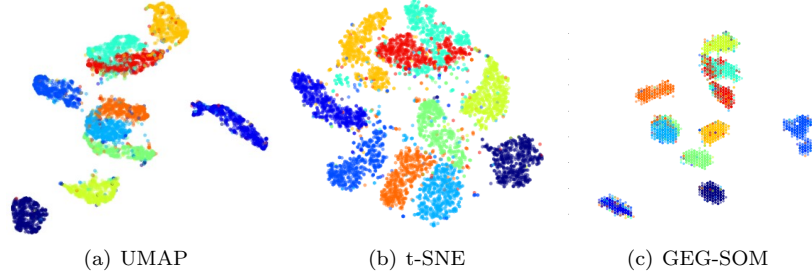


Figure 8: The first 6000 instances of the MNIST dataset reduced to two dimensions using (a)- UMAP, (b)- t-SNE, (c)- GEG-SOM

4.2.2 Fashion MNIST

Fashion MNIST presents a more difficult set of clusters to separate than MNIST. Therefore, the overall performance of all three methods is less than that observed with the MNIST dataset. However, the overall topology demonstrated from the three methods remains consistently similar while UMAP and GEG-SOM demonstrate better performance than t-SNE. A visualization of the first 6000 instances of the Fashion MNIST dataset reduced to two dimensions is illustrated in Fig. 9. We note that both UMAP and GEG-SOM have better recall in identifying the wedge-shaped cluster of points noted in dark-blue and GEG-SOM is capable of isolating the cluster from other clusters. Additionally, GEG-SOM performs better at identifying the three separate clusters in the red, green and yellow partition of the data, at which, both t-SNE and UMAP fail.

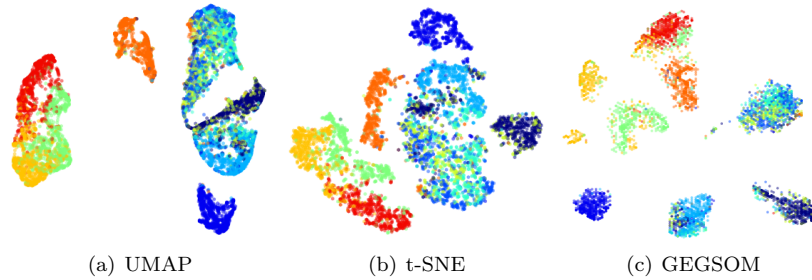


Figure 9: The first 6000 instances of the Fashion-MNIST dataset reduced to two dimensions using (a)- UMAP, (b)- t-SNE, (c)- GEG-SOM

4.2.3 Blood RNA-Seq Data

One of the most prominent areas of application of the dimensionality reduction methods of the caliber of t-SNE and UMAP is found in computational genomics. The problem of visualizing RNA-Seq datasets using nonlinear manifold learning attempts to visualize a set of individual cells, using the levels of expressed genes in each of them. Due to the high level of noise and non-linear nature of data, methods like t-SNE and UMAP are considered to be defacto standard for this task. Therefore we further used GEG-SOM on the high dimensional gene expression levels provided in Nestorowa *et al* [22]. To illustrate how the algorithm can be improved with the use of non-regular grids, we have reduced this dataset using both regular grids and non-regular grids implementation of the GEG-SOM. The visualization is provided in Fig. 10.

We do not provide a cluster quality measure in this dataset, as the expected visualization of the data is the overall global structure of the data, and not distinct clustering. The colors of the points correspond to three different cell types; orange for Long Term Hematopoietic Stem Cells, red for Hematopoietic Stem Cells and grey for Progenitor Cells. We note that the algorithm provides consistent visualizations, while maintaining global structure of the topology. Upon observation, one can see that the proposed method has consistently maintained the global topology of continuous progression of the gene expression from LT-HSC to Progenitor cells.

Overall, from the results in the Table 1, we observe that UMAP has superior performance over t-SNE at all cases. As the level of noise in the data increases, UMAP and t-SNE both struggle to separate clusters. Further observation about the superiority of the GEG-SOM in the presence of such noise can be made from the Table 1.

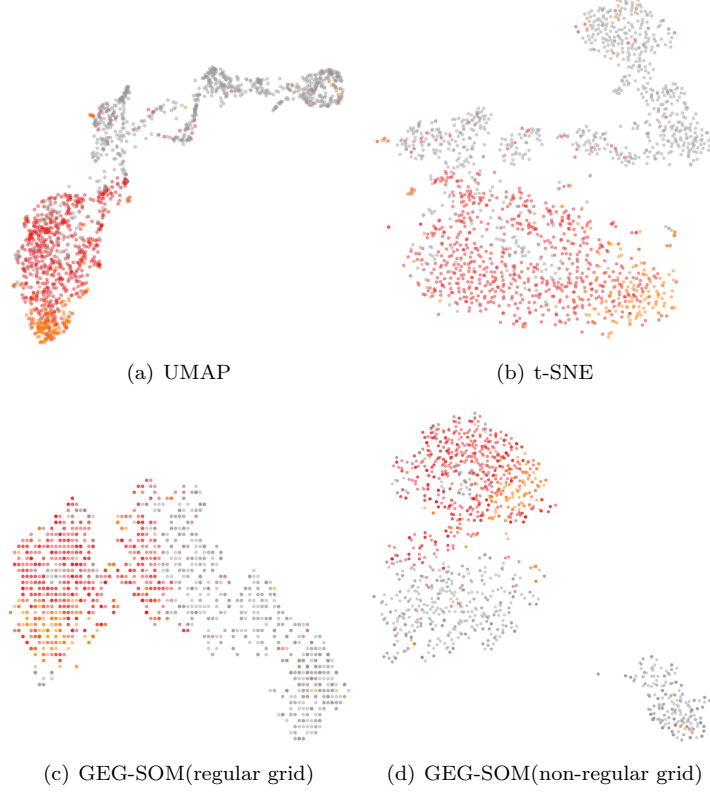


Figure 10: Visualization of the gene expression levels of 1654 cells of three different types (blood RNA-Seq data). Orange points represent LT-HSCs, Grey points represent Progenitors and Red represent HSCs.

5 Conclusions and Future Work

In this work, we propose the GEG-SOM algorithm, introducing the gap-encoding error to enforce the placement of gaps in high dimensional data visualizations generated by Growing Self Organizing Maps, along with the use of non-regular grids for better density-sensitive visualizations. We demonstrate the performance of this algorithm using multiple simulated and real datasets from different domains.

We identify that the GEG-SOM has arguably superior performance in identifying clusters and denoising input by discretizing the data. GEG-SOM is

capable of identifying clusters in the presence of heavy levels of noise, a task at which both t-SNE and UMAP perform poorly. And we further observe the capability of the algorithm to identify complex global topologies while maintaining the core principles of the self organizing algorithms.

We further examine and identify the limitations of the GEG-SOM incurred by the predefined structure of the lower-dimensional graph which affects the performance of identifying highly local topologies. Our proposed modifications alleviate this weakness to some degree, but increases the run-time of the algorithm. We suggest modifications to the grid structure, the growth criterion and the mapping of the lower-dimensional manifold to the coding vectors to extend to a more general graph to overcome these issues. In such datasets, use of t-SNE and UMAP may be more advisable. However, we demonstrate that the global topology approximated by GEG-SOM is consistently similar to that approximated by t-SNE and UMAP in the presence of low noise levels.

For visualization tasks where the separation of clusters is relatively easily achieved (due to the lack of noise in the data), but the quality of the clusters is defined by the shape of the clusters, the GEG-SOM may not perform as well as t-SNE and UMAP. The discretization of the input space by vector-quantization will further be a disadvantage where the number of samples provided to approximate such a structure is small. An example of such a dataset is the COIL-20 [23] where a small number of points per class is expected to form circular patterns of visualization.

One problem we face when extending this into more than two-dimensions is the difficulty to generalize the growth algorithm into arbitrary dimensionalities. As a future improvement, we therefore suggest to incorporate a generic growing graph (similar to that used in Growing Neural Gas algorithms).

Another possible improvement we suggest that could contribute to the enhanced performance of this algorithm is the use of statistical information of the neighborhoods for the training. For instance, the σ values used in the gradient descent algorithm are fixed values, and we do not approximate neighborhoods for individual inputs based on the distribution of datapoints around each other.

References

- [1] Luis O Jimenez and David A Landgrebe. Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(1):39–54, 1998.
- [2] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
- [3] Itay Tirosh, Benjamin Izar, Sanjay M Prakadan, Marc H Wadsworth, Daniel Treacy, John J Trombetta, Asaf Rotem, Christopher Rodman, Christine Lian, George Murphy, et al. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell rna-seq. *Science*, 352(6282):189–196, 2016.
- [4] Dominic Grün, Anna Lyubimova, Lennart Kester, Kay Wiebrands, Onur Basak, Nobuo Sasaki, Hans Clevers, and Alexander van Oudenaarden. Single-cell messenger rna sequencing reveals rare intestinal cell types. *Nature*, 525(7568):251, 2015.
- [5] Amit Zeisel, Ana B Muñoz-Manchado, Simone Codeluppi, Peter Lönnerberg, Gioele La Manno, Anna Juréus, Sueli Marques, Hermany Munguba, Liqun He, Christer Betsholtz, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142, 2015.
- [6] Sophie Petropoulos, Daniel Edsgård, Björn Reinius, Qiaolin Deng, Sarita Pauliina Panula, Simone Codeluppi, Alvaro Plaza Reyes, Sten Linnarsson, Rickard Sandberg, and Fredrik Lanner. Single-cell rna-seq reveals lineage and x chromosome dynamics in human preimplantation embryos. *Cell*, 165(4):1012–1026, 2016.

- [7] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
- [8] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.
- [9] Jarkko Venna and Samuel Kaski. Local multidimensional scaling. *Neural Networks*, 19(6):889–899, 2006.
- [10] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [11] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *NIPS*, volume 15, pages 833–840, 2002.
- [12] Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
- [13] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- [14] Dammina Alahakoon, Saman K Halgamuge, and Bala Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on neural networks*, 11(3):601–614, 2000.
- [15] Chon-Kit Kenneth Chan and Saman Halgamuge. A new generalized growth threshold for dynamic som for comparing average mutual information and oligonucleotide frequency as a species signature. *Issues*, (182), 2016.
- [16] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.
- [17] Bernd Fritzke. Kohonen feature maps and growing cell structures-a performance comparison. In *Advances in neural information processing systems*, pages 123–130, 1993.

- [18] Alexander N Gorban, Balázs Kégl, Donald C Wunsch, Andrei Y Zinovyev, et al. *Principal manifolds for data visualization and dimension reduction*, volume 58. Springer, 2008.
- [19] Yann LeCun, Corinna Cortes, and Christopher JC Burges. Mnist handwritten digit database. *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [20] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [21] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.
- [22] Sonia Nestorowa, Fiona K Hamey, Blanca Pijuan Sala, Evangelia Diamanti, Mairi Shepherd, Elisa Laurenti, Nicola K Wilson, David G Kent, and Berthold Göttgens. A single cell resolution map of mouse haematopoietic stem and progenitor cell differentiation. *Blood*, pages blood–2016, 2016.
- [23] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.

Algorithm 1. GEG-SOM Algorithm

```

1: procedure GEG-SOM
2:   let  $X$  be the input set of dimensionality  $D$ 
3:   initiate  $W$  to have  $n$  vectors of dimensionality  $D$ .  $n$  can be  $m + 1$  where  $m$  is
   the number of unit-distance neighbors desired by the user
4:   initiate  $Y$  to have  $n$  vectors of dimensionality 2. A usual initialization would
   have one vector at  $(0, 0)$  and  $m$  vectors placed at regular angles with a unit distance
   to the first vector.
5:   initiate  $\alpha_t \leftarrow \alpha_s$  where  $0 < \alpha_s < 1$ . is starting learning rate
6:   initiate  $\gamma_t \leftarrow \gamma_s$  where  $0 < \gamma_s < .5$  is the starting gap coefficient
7:   while  $E_t > E_{\min}$  do
8:     set all errors  $(E_k)$  to 0
9:     for  $x_i \in X$  do
10:      set  $\alpha_t \leftarrow \alpha_s \exp(-t/t_{max})$ 
11:      set  $\gamma_t \leftarrow \gamma_s \exp(-t/t_{max})$ 
12:      set  $age_{max}$  to appropriate value (usually set to  $.1|X|$ )
13:      Find  $c$  such that
          
$$c = \underset{j \in |W|}{\operatorname{argmin}} \|x_i - w_j\|$$

14:      increase the Error of the node  $c$  as
          
$$Err_c \leftarrow Err_c + \|x_i - w_c\|$$

15:      set ages of all neighbors of  $c$  to 0
16:       $\forall k \in 1...|W|; w$  adjust the weights of the map as per the rule
          
$$w_k \leftarrow (x_i - w_k)((\alpha_t N(c, y_j) + \gamma_t H(c, w_k))$$

17:      if  $Err_c > GT$  then
18:        distribute errors to neighbor nodes as per the rule
          
$$Err_l \leftarrow (1 + \phi)Err_l$$

19:      end if
20:      if  $c$  is a boundary node then
21:        create new nodes and add to the map
22:      end if
23:      delete all nodes with ages  $> age_{max}$ 
24:      set the total quantization error  $E_t$  as
          
$$E_t \leftarrow \sum_{\forall l \in \{1...|W|\}} Err_l$$

25:     end for
26:   end while
27: end procedure

```

2.2 Summary and Contributions

We demonstrate the the proposed method, GEGSOM is capable of improving the visualization capabilities of Self Organizing Maps and its variants. We also show that in the experiments we have considered, GEGSOM has superior tolerance to noise and high variance as opposed to t-SNE and UMAP. Out of all the methods considered, the GEGSOM has visualization performance possibly second only to UMAP.

The main contributions in this work are 1) A modified quantization error that accounts for global density variation in the input space and 2) A modified graph structure that better represents the local density variation in the input space. We further show that this method improves the performance of the GSOM algorithm significantly.

2.3 Caveats

We note that when it comes to the scalability of this method, GEGSOM falls far behind t-SNE and UMAP. We speculate that this may be due to the fact that the fixed structure of the map (whether it be with regular or irregular grids), a large number of ‘self-organization’ iterations are required to obtain an adequate approximation of the input manifold.

Furthermore, one of the main issues we perceive with GEGSOM is that it relies on multiple assumptions regarding the behavior of the ‘uniformity’ of a manifold with increasing dimensionality.

While this work significantly improves on traditional SOM algorithms, we consider that a more robust approach can be obtained by manipulating 1) the graph structure without any reliance on a fixed structure and 2) the output embedding accentuating different densities in the input space. With this in mind, we address our second research question in our next chapter.

Chapter 3

How can the SOM-based methods be used as a parametric alternative to modern robust nonlinear dimensionality reduction techniques?

Overview

In this chapter, a new algorithm for dimensionality reduction and visualization is proposed, called Self Organizing Nebulous Growths (SONG). We show the superiority of the SONG algorithm in incremental visualization of data against UMAP and t-SNE using three real-world datasets. We further show that in terms of static data visualization, SONG outperforms t-SNE and is comparable to UMAP. Furthermore, we show that in the presence of moderate noise, SONG is superior to both t-SNE and UMAP.

3.1 Introduction

Keeping in mind that our main goal is to produce a robust dimensionality reduction technique that retains an input-to-output mapping for the visualization, which is one of the most sought-after capabilities of the nonlinear dimensionality reduction techniques and a major gap in research, we propose our next research question. In this work, discuss the second approach of improving the visualizations provided by SOM-based algorithms, i.e., manipulating the output visualizations such that the density information in the input space are accentuated in the visualizations without attempting to uniformize the distribution of coding vectors in the input space.

For this, we use the concepts used by GNG to approximate the input density distribution directly, without relying on assumptions about the high-dimensional probability distributions. Next, we use the concepts used by NG-CE to obtain an intermediate, and explicitly non-final representation of the input probability distribution, embedded in a low-dimensional output space.

However, we do extensive reviews on the more recent improvements regarding the domain of these 'stochastic neighbor embeddings', and many representatives of this class of algorithms, namely, t-SNE and UMAP. Using these recent improvements, we develop a highly efficient, scalable and robust method, which we call 'Self Organizing Nebulous Growths'. As of the date of submission, this manuscript is submitted to the IEEE Transactions on Neural Networks and Learning Systems, and a preprint is available on arXiv.org. We make the code available at ¹.

¹<https://github.com/damithsenanayake/SONG.git>

Self Organizing Nebulous Growths for Robust and Incremental Data Visualization

Damith A. Senanayake, Wei Wang, Shalin H. Naik*, Saman Halgamuge [†]

May 7, 2020

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 the data 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 of the above datasets, 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.

*S. Naik is with the Walter and Eliza Hall Institute for Medical Research, Melbourne, Victoria, Australia

[†]D. Senanayake, W. Wang and S. Halgamuge are all with the department of Mechanical Engineering of the University of Melbourne, Victoria, Australia

Keywords: t-SNE, UMAP, SONG, Vector Quantization, Nonlinear Dimensionality Reduction, Heterogeneous Incremental Learning

1 Introduction

In data analysis today, we often encounter high-dimensional datasets with each dimension representing a variable or feature. 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 so large that it has to be divided and processed sequentially [1]. Second, there are scenarios where data is incrementally acquired through a series of experiments, such as the continuous acquisition of Geographical Data [2] or data gathered by mining social media [3]. 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 does not change drastically after the new data is added for consistency in data interpretation.

Recently, t-distributed Stochastic Neighbor Embedding (t-SNE) [4] and Uniform Manifold Approximation and Projection (UMAP) [5] have shown great success in nonlinear dimensionality reduction of static datasets. However, t-SNE and UMAP are non-parametric by design and therefore have to be *reinitialized* and *retrained*, thus not ideally suited for incremental data visualization. The parametric variant of t-SNE, called parametric t-SNE [6], may suit better for such scenarios, as it can be further trained with either the new data or a combination of old and new data without reinitializing. However, as we will show later, the visualization quality of parametric t-SNE is not as good as t-SNE for the datasets considered in our experiments.

In this work, we propose a new parametric dimensionality reduction method called Self-Organizing

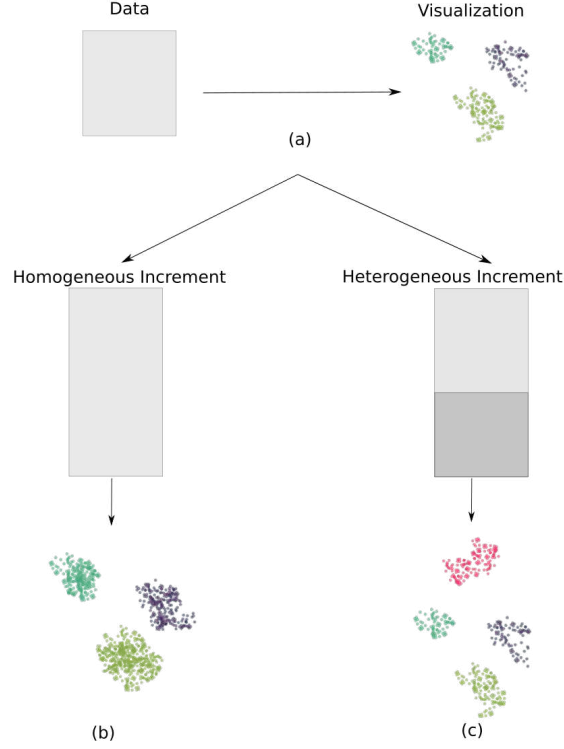


Figure 1: Illustration of incremental data visualization: a) the visualization of the initially available data contain three clusters; b) the initial data are augmented with homogeneous (similar) data, where clusters become denser in the visualization; c) the initial data are augmented with heterogeneous (dissimilar) data; new clusters are added to the visualization.

Nebulous Growths (SONG) for incremental data visualization. SONG combines the strength of t-SNE and UMAP where it provides noise-tolerant, interpretable visualizations and variants of the parametric method called Self-Organizing Map (SOM) [7] where they can learn a parametric model for visualizing high-dimensional data. We show that SONG provides visualization quality comparable to or better than that of UMAP and t-SNE in static data scenarios while being significantly more efficient than the parametric t-SNE for incremental data visualization.

2 Related Work

In static data visualization scenarios, t-SNE and UMAP are two state-of-the-art methods frequently used for dimensionality reduction. Both t-SNE and UMAP are inspired by Stochastic Neighbor Embedding (SNE) [8] which consists of two main stages. First, a graph is created 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. Second, these pseudo-probabilities are preserved on an output graph of low-dimensionality (typically two or three). Due to the probabilistic nature, the edges in SNE are stochastic, thus different from the binary edges used in conventional methods such as Isomap [9] and LLE [10] to represent the similarity between two connected points. As a result, SNE is more tolerant to outliers than LLE or Isomap. Additionally, the base distributions for converting the pair-wise distances to pseudo-probabilities in SNE can vary, allowing us to control the cluster tightness and separation in the visualizations. Similarly, varying the loss function of SNE which calculates the discrepancy between the two sets of pseudo-probability distributions in the input and output spaces, we can control the granularity of the preserved topology.

UMAP and t-SNE differ in the selection of base distributions and loss functions. t-SNE calculates the edge-weights by assuming Gaussian distributions for the pairwise distances in the input space but Student’s t-distributions in the output space for a clear cluster separation. In contrast, UMAP assumes that the local neighborhoods lie on a Riemannian manifold and 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. For the loss function, t-SNE uses KL-Divergence while UMAP uses cross-entropy. As a result, UMAP provides higher separation of clusters and more consistent global topology preservation in visualization than t-SNE.

Due to its success in static data visualization, t-SNE has been the inspiration for several general-purpose visualization methods including Trimap [11] with better global topology preservation and LargeVis [12] which is more efficient in large datasets than t-SNE. Also, t-SNE has been made more efficient in Barnes-Hut-SNE [13], extended to suit specific applications [14, 15], and inspired other application-specific visualization methods such as viSNE [16] used on single-cell transcriptomic data.

However, being non-parametric, t-SNE and its successors [11, 12, 13] as well as UMAP need to

be *reinitialized* and *retrained* at each increment of data. In UMAP, the heuristic initialization using Spectral Embedding (Laplacian Eigenmaps) [17] 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 investigated how well such heuristic initializations perform when the new data have a heterogeneous structure to the existing data.

One previously explored strategy for retaining a parametric mapping in t-SNE is to train a parametric regression model using the training data and the visualizations obtained by t-SNE. For example, Kernel t-SNE [18] builds a linear model between the currently embedded vectors and the normalized Gram Matrix of the new inputs. However, the Gram matrix and linear weights do not adjust to new data, thus Kernel t-SNE may not be suitable for heterogeneous increments. On the other hand, parametric t-SNE [6] uses a feed-forward neural network as the regression model which may be continually trained to adapt to the heterogeneous increments. Another strategy, used in the single-cell RNA-Seq data visualization tool scvis [19], is to train a Variational Autoencoder (VAE) on the input data with the coding space limited to 2 or 3-dimensions for visualization. The VAE is regularized by a t-SNE objective for better cluster separation in the visualization. However, scvis has been shown to perform poorly at separating distinct clusters compared to t-SNE [20]. Additionally, parametric t-SNE and scvis suffer from issues commonly associated with deep neural networks such as requiring a large amount of training data [21], high computational complexity [22] and lack of model interpretability [23].

Self-Organizing Map (SOM) [7] and its variants are arguably the only dimensionality reduction methods that retain an adjustable parametric graph on the input space to approximate the input data distribution locally. SOMs obtain the graph by vector encoding or vector quantization, 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. The coding vectors, now representing the centroids of the Voronoi regions, are then mapped onto a low-dimensional uniform output grid. In practice, SOM implementations often use a 2 or 3-dimensional uniform grid, that has either a square, a triangular or a hexagonal topology for the locally connected output vectors. The topology preservation of SOMs is achieved by moving the 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 and fixed output grid [24]. SNE-based methods are not affected by this problem as their outputs are adjusted to reflect the topology of the high-dimensional space. Second, the size of the map (the number of coding vectors) needs to be known *a priori*. Growing Cell Structures (GCS) [24] and its successor Growing Neural Gas (GNG) [25] use a non-uniform triangulation (where the coding vectors represent the vertices of the triangles) of the input space to tackle these two problems. GCS uses a force-directed graph drawing [26] which can only visualize 2 or 3-dimensional graphs. Force-directed drawing algorithms such as ‘Spring’ [26] for graphs of arbitrary dimensionality have been successfully used in specific applications such as single-cell transcriptomic trajectory visualization [27], where the graph is sparse, i.e., number of edges is small compared to the number of possible pairs of vertices. However, these algorithms are not suited for visualizing the coding vectors of GNG with less-sparse graphs of arbitrary dimensionality.

By using cross-entropy minimization similar to UMAP for visualizing the coding vectors, Neural Gas Cross-Entropy (NG-CE) [28] overcomes the inability to embed less-sparse graphs shown by spring-like algorithms. However, NG-CE has not been extended to support the dynamically growing nature of GNG. Growing SOM (GSOM) [29] provides another approach to overcome SOM’s deficiency of unknown map size by using a uniform but progressively growing output grid. However, similar to SOM, GSOM also uses a uniform output grid.

In the following section, we describe our proposed algorithm, Self-Organizing Nebulous Growths (SONG) which draws inspiration from SNE and NG-CE in using the discrepancy between input and output probability distributions to obtain a topology-preserving visualization, and GNG and GSOM in robust parametrization of the input topology with a growing network of coding vectors.

3 Method

In the proposed SONG, we use a set of coding vectors $\mathbf{C} = \{\mathbf{c} \in \mathbb{R}^D\}$ to partition and represent the input dataset $\mathbf{X} = \{\mathbf{x} \in \mathbb{R}^D\}$. For an input $\mathbf{x}_i \in \mathbf{X}$, we define an index set $\mathbf{I}^{(k)} = \{i_l | l = 1, \dots, k\}$ for a user-defined k , where $\mathbf{c}_{i_l}$ is the l -th closest coding vector to $\mathbf{x}_i$. Moreover, we define a set of directional edges between coding vectors $\mathbf{C}$ and an adjacency matrix $\mathbf{E}$, such that if a coding vector $\mathbf{c}_m$ is one of the closest neighbors to another coding vector $\mathbf{c}_l$, they are connected by an edge with edge strength $\mathbf{E}(l, m) > 0$. We organize the graph $\{\mathbf{C}, \mathbf{E}\}$ to approximate the input topology.

We also define a set of low-dimensional vectors $\mathbf{Y} = \{\mathbf{y} \in \mathbb{R}^d\}$, $d \ll D$ which has a bijective correspondence with the set of coding vectors $\mathbf{C}$. When $d = 2$ or 3 , $\mathbf{Y}$ represents the visualization of the input space, i.e., the input $\mathbf{x}_i \in \mathbf{X}$ is visualized as $\mathbf{y}_{i_1} \in \mathbf{Y}$. We preserve the topology of $\mathbf{C}$ given by $\mathbf{E}$ in $\mathbf{Y}$ by positioning $\mathbf{Y}$ such that, if $\mathbf{E}(l, m) > 0$ or $\mathbf{E}(m, l) > 0$, $\mathbf{y}_l$ and $\mathbf{y}_m$ will be close to each other in the visualization. Typically, the number of $\mathbf{c} \in \mathbf{C}$ and corresponding low dimensional vectors $\mathbf{y} \in \mathbf{Y}$ is far less than the number of input data points $\mathbf{X}$. By retaining the parameters $\mathbf{C}$, $\mathbf{E}$ and $\mathbf{Y}$, SONG obtains a parametric mapping from input data to visualization.

We initialize a SONG model by randomly placing $d+1$ coding vectors $\mathbf{C}$ in the input space, since $d+1$ is the minimum number of coding vectors needed 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. $\mathbf{E} = \mathbf{0}$). The corresponding $\mathbf{Y}$ are also randomly placed in the d -dimensional output space. Next, we approximate local topology of any given $\mathbf{x} \in \mathbf{X}$ using $\mathbf{C}$, and project this approximated topology to $\mathbf{Y}$ in the visualization space. To be specific, SONG randomly samples an input point $\mathbf{x}_i \in \mathbf{X}$ and performs the following steps at each iteration until terminated:

1. *Updating the Directional Edges in $\mathbf{E}$ between Coding Vectors $\mathbf{C}$ based on $\mathbf{x}_i$* : This step modifies the adjacency matrix $\mathbf{E}$ to add or remove the edges between coding vectors based on local density information at $\mathbf{x}_i$. Eventually, if no edge is added or removed for repeated sampling of all input $\mathbf{x}_i$ 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 3.1.
2. *Self-Organization of Coding Vectors $\mathbf{C}$* : This step moves $\mathbf{x}_i$'s closest coding vector $\mathbf{c}_{i_1}$ closer to $\mathbf{x}_i$, along with any coding vectors $\mathbf{c}_j$ if $\mathbf{c}_{i_1}$ and $\mathbf{c}_j$ are connected by an edge as indicated by $\mathbf{E}(i_1, j) > 0$. This movement enforces the closeness of coding vectors connected by edges. We describe this in detail in Section 3.2.
3. *Topology Preservation of the Low-dimensional Points $\mathbf{Y}$* : Given that $\mathbf{c}_{i_1}$ encodes $\mathbf{x}_i$ and corresponds to $\mathbf{y}_{i_1}$ in the output space, we organize low-dimensional points $\mathbf{y}_j \in \mathbf{Y}$ in the locality of $\mathbf{y}_{i_1}$ such that the coding vector topology at $\mathbf{c}_{i_1}$ is preserved in the output space. This step is described in Section 3.3.
4. *Growing $\mathbf{C}$ and $\mathbf{Y}$ to Refine the Inferred Topology*: There may be cases where $\mathbf{c}_{i_1}$ and its

neighboring coding vectors are insufficient to capture the local fine topology at $\mathbf{x}_i$, e.g., inputs from multiple subclusters may have the same $\mathbf{c}_{i_1}$. In such cases, we place new coding vectors close to $\mathbf{c}_{i_1}$, and new corresponding low-dimensional vectors close to $\mathbf{y}_{i_1}$, without reinitializing the parametric model $\{\mathbf{X}, \mathbf{C}, \mathbf{E}\}$. Details of this are in Section 3.4.

For a given epoch, we randomly sample (without replacement) a new input $\mathbf{x}_i \in \mathbf{X}$ and repeat the four steps, until all $\mathbf{x}_i \in \mathbf{X}$ 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 $\mathbf{X}'$ are presented, we simply allow $\mathbf{x}_i$ to be sampled from $\mathbf{X}'$ at the next iteration, and continue training without reinitializing the parameters $\mathbf{C}$, $\mathbf{Y}$ and $\mathbf{E}$.

3.1 Updating the Directional Edges in $\mathbf{E}$ between Coding Vectors based on $\mathbf{x}_i$

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

- *Renewal*: we reset $e_{i_1j}^t$ to 1 if $j \in \mathbf{I}^{(k)}$.
- *Decay*: if $e_{i_1j}^{t-1} > 0$, we decay it by a constant multiplier ϵ , i.e., $e_{i_1j}^t = e_{i_1j}^{t-1} \cdot \epsilon$ and $\epsilon \in (0, 1)$. Note that for edges with strengths $e_{i_1j}^{t-1} > 0$ for $j \notin \mathbf{I}^{(k)}$ at current iteration t , this decay operation would weaken such edges repeatedly. Therefore, edges created at an earlier stage of training that no longer connects two close coding vectors will be weakened.
- *Pruning*: we set $e_{i_1j}^t$ to 0 if $e_{i_1j}^{t-1} < e_{\min}$. This helps to obtain a sparse graph by removing edges that connect distant coding vectors as such edges would be weakened due to lack of frequent renewal and frequent decay. Note that $e_{\min}$ 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 $\mathbf{x}_i$. Since the edge strengths $e_{i_1j} \in [0, 1]$, larger edge strengths can be interpreted as $\mathbf{c}_{i_1}$ and neighboring coding vectors representing finer topologies (shorter distances between $\mathbf{C}$). 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 [30].

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

$$\mathbf{E}_s = \frac{\mathbf{E} + \mathbf{E}^T}{2} \quad (1)$$

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

3.2 Self-Organization of Coding Vectors $\mathbf{C}$

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

$$QE(x) = \frac{1}{2} \|\mathbf{x}_i - \mathbf{c}_{i_1}\|^2 \quad (2)$$

However, moving $\mathbf{c}_{i_1}$ independent of other coding vectors may cause the coding vectors sharing an edge with $\mathbf{c}_{i_1}$ to be no longer close to $\mathbf{c}_{i_1}$, which disorganizes the graph. To avoid this, we also move $\mathbf{c}_{i_1}$'s neighboring coding vectors $\mathbf{c}_j$ (as indicated by $\mathbf{E}_s(i_1, j) > 0$) towards $\mathbf{x}_i$ by a smaller amount than that of $\mathbf{c}_{i_1}$. Moreover, the more distant $\mathbf{c}_j$ is from $\mathbf{c}_{i_1}$, the smaller the movement of $\mathbf{c}_j$ 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 $\mathbf{x}_i$ increases. In addition, to penalize large neighborhoods (and thereby large edge lengths), we scale the loss function by a constant specific to the neighborhood $\mathbf{I}^{(k)}$, and in this work, we select the square of the largest distance from $\mathbf{x}_i$, i.e., $\|\mathbf{x}_i - \mathbf{c}_{i_k}\|^2$ as this constant. Note that we treat the distance to the k^{th} coding vector from x as a constant for the considered neighborhood, and the gradient of $\|\mathbf{x}_i - \mathbf{c}_{i_k}\|$ w.r.t. $\mathbf{c}_{i_k}$ is not calculated. The final loss function is:

$$\mathcal{L}(\mathbf{x}_i) = -\frac{\|\mathbf{x}_i - \mathbf{c}_{i_k}\|^2}{2} \sum_{\mathbf{c}_j \in \mathcal{N}_{i_1}} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{c}_j\|^2}{\|\mathbf{x}_i - \mathbf{c}_{i_k}\|^2}\right) \quad (3)$$

where $\mathcal{N}_{i_1} = \{\mathbf{c}_j \mid \mathbf{E}_s(j, i_1) > 0\}$ is the set of $\mathbf{c}_{i_1}$'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 $\mathbf{c}$ for the sampled $\mathbf{x}_i$ as:

$$\frac{\partial \mathcal{L}(\mathbf{x}_i)}{\partial \mathbf{c}_j} = (\mathbf{x}_i - \mathbf{c}_j) \times \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{c}_j\|^2}{\|\mathbf{x}_i - \mathbf{c}_{i_k}\|^2}\right) \quad (4)$$

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

3.3 Topology Preservation of the Low-dimensional Points $\mathbf{Y}$

Similar to UMAP and inspired by NG-CE, SONG optimizes the embedding $\mathbf{Y}$ 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 $\mathbf{x}_i \in \mathbf{X}$ as:

$$CE(\mathbf{x}_i) = \sum_{\forall j} -p_{i_1j} \log(q_{i_1j}) - (1 - p_{i_1j}) \log(1 - q_{i_1j}) \quad (5)$$

where q_{i_1j} is the probability that output points $\mathbf{y}_{i_1}$ and $\mathbf{y}_j$ are located close together, and is calculated using the following rational quadratic function:

$$q_{i_1j} = \frac{1}{1 + a\|\mathbf{y}_{i_1} - \mathbf{y}_j\|^{2b}} \quad (6)$$

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 $CE_{\text{attr}} = \sum -p_{i_1j} \log(q_{i_1j})$ that attracts $\mathbf{y}_{i_1}$ towards $\mathbf{y}_j$, and an repulsion component $CE_{\text{rep}} = \sum -(1 - p_{i_1j}) \log(1 - q_{i_1j})$ that repulses $\mathbf{y}_{i_1}$ from $\mathbf{y}_j$. The attraction component heavily influences local arrangement at $\mathbf{y}_{i_1}$, since distant $\mathbf{y}_j$ results in $p_{i_1j} = 0$. The repulsion component, on the other hand, influences the global arrangement of neighborhoods, as $1 - p_{i_1j} = 1$ for such distant $\mathbf{y}_j$. Due to the difference of influences, we derive the gradients of these two components

separately. The attraction component has a gradient as defined by:

$$\frac{\partial CE_{\text{attr}}}{\partial \mathbf{y}} = (\mathbf{y}_{i_1} - \mathbf{y}_j) \cdot \frac{2ab \cdot p_{ij} \cdot \|\mathbf{y}_j - \mathbf{y}_{i_1}\|^{2b-2}}{1 + \|\mathbf{y}_j - \mathbf{y}_{i_1}\|^{2b}} \quad (7)$$

The gradient for the repulsion component is:

$$\frac{\partial CE_{\text{rep}}}{\partial \mathbf{y}} = (\mathbf{y}_{i_1} - \mathbf{y}_j) \cdot \frac{2b \cdot (1 - p_{ij})}{\|\mathbf{y}_j - \mathbf{y}_{i_1}\|^2 (1 + \|\mathbf{y}_j - \mathbf{y}_{i_1}\|^{2b})} \quad (8)$$

We use stochastic gradient descent to minimize CE_{attr} and CE_{rep} w.r.t $\mathbf{y}_{i_1}, \mathbf{y}_j \in \mathbf{Y}$. We select stochastic gradient descent over batch gradient descent to avoid convergence on sub-optimal organizations [31]. Moreover, since the edge renewal rate is proportional to the p_{i_1j} , we propose to use the symmetric edge strengths $\hat{e}_{i_1j} \in E_s$ as an approximation of p_{i_1j} in CE_{attr} . This avoids the explicit assumptions on p_{i_1j} as made by t-SNE and UMAP. In a similar fashion, we use the negative sampling of edges (i.e. sampling of j where $\mathbf{E}_s(i_1, j) = 0$) to approximate $(1 - p_{i_1j})$ in CE_{rep} . In this negative sampling, for a very large dataset, we randomly sample a set of non-edges, such that for each $\mathbf{x}_i$, the number of sampled non-edges n_{ns} equals the number of edges connected to $\mathbf{c}_{i_1}$ ($n(e_i)$) multiplied by a constant rate. Similar ideas have been used in Word2Vec [32] and UMAP [5]. The algorithmic summary of this step is shown in Algorithm 3.

3.4 Growing $\mathbf{C}$ and $\mathbf{Y}$ to Refine the Inferred Topology

By iterating the above three steps from Sections 3.2 - 3.3, the topology of $\mathbf{x}$ can be approximated using $\{\mathbf{C}, \mathbf{E}\}$ and preserved onto $\mathbf{Y}$ in the visualization space. However, since the optimal number of coding vectors $\mathbf{C}$ is unknown *a priori*, SONG starts with a small number of $\mathbf{C}$ which may be insufficient to capture all the structures in $\mathbf{X}$ such as clusters and sub-clusters. Therefore, we grow the sizes of $\mathbf{C}$ and $\mathbf{Y}$ 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 $\mathbf{c}_{i_1}$ as:

$$G_{i_1}(t) \leftarrow G_{i_1}(t-1) + \|\mathbf{x}_i - \mathbf{c}_{i_1}\| \quad (9)$$

where t is the index of current iteration. When any $G_{i_1}(t)$ exceeds a predefined threshold θ_g , we place a new coding vector $\mathbf{c}$ at the centroid between $\mathbf{x}_i$ and its k nearest coding vectors, so that the regions that have high Growth Error get more populated with coding vectors. In Supplement Section 1.2, we describe how we have calculated the θ_g from a hyperparameter called the Spread Factor (SF) as defined in [29, 33]. Due to the stochastic sampling, the current $\mathbf{x}_i$ 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 $\mathbf{c}_{i_1}$. The placement of new $\mathbf{y}$ is conducted similarly, and by placing the $\mathbf{y}$ close to the neighborhood of $\mathbf{y}_{i_1}$, the convergence of the new $\mathbf{y}$ to a suitable position is made faster, than by placing the new $\mathbf{y}$ randomly on the output map. We summarize this step in Algorithm 4. These four steps form a complete iteration of the SONG algorithm, which we summarize in Algorithm 1. In the next section, we evaluate the performance of the SONG algorithm.

4 Experiments and Results

In this section, we compare SONG against Parametric t-SNE and non-parametric methods: t-SNE [4] and UMAP [5] on a series of data visualization tasks. First, we consider incremental data visualization with heterogeneous increments of data in Section 4.1 and homogeneous increments of data in Section 4.2. It is noteworthy that the former is more likely the case to be assumed in real problems with incremental data streams due to lack of the ground-truth. Then we evaluate the visualization quality of SONG in static data visualization scenarios: we assess the SONG’s robustness to noisy and highly mixed clusters in Section 4.3, and SONG’s capability in preservation of topologies in Section 4.4.

In our analysis, we define model-retaining methods as methods that reuse a pretrained model and refine it when presented with new data, while model-reinitializing methods reinitialize and retrain a model from scratch. Therefore, SONG and Parametric t-SNE are model-retaining methods, and t-SNE and UMAP are 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 conducted a study on SONG’s sensitivity to hyper-parameters, which we have included in our Supplement Section 3.3. Four key hyper-parameters that affect the graph inference (number of coding vectors in a neighborhood: k , edge-strength decay rate: ϵ , initial learning rate: α_0 and Spread Factor: SF as defined in [29] and [33]), and two parameters that affect the projection (a and b) were identified in this study. Through this experiment, we found that higher spread-factors and lower k values preserve a finer topology. Therefore for a more faithful representation of high dimensional data, we recommend a high Spread Factor ($SF > 0.9$) and a small number of coding vectors in a neighborhood ($k < 5$).

We use the hyper-parameters in Table 1 for each method. For t-SNE [4] and UMAP [5], the recommended hyper-parameters in the 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 ¹. The tuned hyper-parameters for SONG are noted in Table 1.

4.1 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 [34], MNIST hand-written digit dataset [35] and the Fashion MNIST dataset [36]. 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 Follides, 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 [20]. Since we have no ground-truth labels and UMAP provides superior qualitative cluster separation on this dataset [20], 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’ [37] 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,

¹https://github.com/jsilter/parametric_tsne

each image having 28×28 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) [38] 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 [39]. 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 the most stable placement of clusters when new data are presented, compared to parametric t-SNE and model-reinitialized methods, UMAP, t-SNE and SONG + Reinit. For the Wong dataset in Fig. 2, both SONG and Parametric t-SNE show similar cluster placements in the first two visualizations. However, Parametric t-SNE visualizations become distorted as more data are presented. In contrast, SONG provides consistently stable visualizations. On MNIST and Fashion MNIST datasets, the cluster placements provided by SONG have a noticeable change for the first two increments, but become more stable in their later increments. Parametric t-SNE shows a 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 2 summarizes the AMI Scores for the Fashion MNIST and MNIST datasets in heterogeneous incremental visualization scenarios. Note that in Table 2, 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 2, 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. We also consider incremental visualizations with kernel t-SNE in Supplement Section 3.4. The results show that kernel t-SNE performs poorly when the new data is heterogeneous.

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 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.

4.2 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 $\mathbf{Y}$* (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 $\mathbf{Y}^{(t-1)}$ and $\mathbf{Y}^{(t)}$. Next we calculate the CDY of a point $\mathbf{y}_i$ in the existing visualization $\mathbf{Y}^{(t-1)}$ as:

$$\text{CDY}(\mathbf{y}_i) = \|\mathbf{y}_i^{(t)} - \mathbf{y}_i^{(t-1)}\|$$

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 3. In Table 3, 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.

On Wong dataset, UMAP and SONG have more similar cluster placement in the homogeneous increment cases (see Fig. 5) than that of the heterogeneous increment cases (see Fig. 2). This

increased similarity may possibly be due to the initial intermediate data samples in homogeneous cases more accurately representative of the global structure of the data than that of heterogeneous cases. However, SONG shows no rotations in the visualization when data is augmented; in contrast, UMAP shows different orientations of similar cluster placements.

Table 3 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 3 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 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 from scratch at each increment on existing data and the newly presented data.

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. Given 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.

4.3 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.

Result: Table 4 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.7. 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 4, 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.

4.4 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 [40], which is frequently used to assess the topology preservation of visualization methods [4][5].

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 preserve the circular topologies better than t-SNE, where t-SNE shows an arch-like shape instead of circular structures.

4.5 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 ($\mathbf{X}$ and $\mathbf{C}$) multiple times for self-organization, SONG has a performance bottleneck which renders it slower than UMAP.

5 Conclusions and Future Work

In this work, we have presented a parametric nonlinear dimensionality reduction method called Self-Organizing Nebulous Growths (SONG) 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 4.1) and homogeneous (Section 4.2) data increments, and observed that in both cases, SONG is superior to parametric t-SNE in preserving cluster placements when incorporating new data. Additionally, SONG’s cluster visualization quality is on par with UMAP and superior to non-parametric t-SNE. We also showed that SONG is robust to noisy and highly mixed clusters (Section 4.3), and that SONG is capable of preserving specific topologies (Section 4.4) inferred from the input.

The main merit of SONG is its usefulness in visualizing large datasets where considerable heterogeneity is present. This heterogeneity may be due to undesired batch effects [41] or genuine variation in the populations of 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 [42].

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 needs to be recalculated several times, whereas in UMAP, the high-dimensional

KNN graph is constructed only once. We have empirically determined that 8-10 recalculations of the graph are 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-organizing algorithms are prone to sub-optimal solutions [31]. This may be viable at later stages of training 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. 5), 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 use SONG in semi-supervised learning 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. This work is partially funded by ARC DP150103512.

References

- [1] Danyel 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.
- [2] Lorena Montoya. Geo-data acquisition through mobile GIS and digital video: an urban disaster management perspective. *Environmental Modelling & Software*, 18(10):869–876, 2003.
 - [3] 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.
 - [4] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
 - [5] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
 - [6] Laurens Van Der Maaten. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pages 384–391, 2009.
 - [7] Teuvo Kohonen. Self-Organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.
 - [8] Geoffrey E Hinton and Sam T Roweis. Stochastic Neighbor Embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.
 - [9] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
 - [10] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *science*, 290(5500):2323–2326, 2000.
 - [11] Ehsan Amid and Manfred K Warmuth. TriMap: Large-scale dimensionality reduction using triplets. *arXiv preprint arXiv:1910.00204*, 2019.
 - [12] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th international conference on world wide web*, pages 287–297, 2016.
 - [13] Laurens Van Der Maaten. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

- [14] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature methods*, 16(3):243–245, 2019.
- [15] Weijing Song, Lizhe Wang, Peng Liu, and Kim-Kwang Raymond Choo. Improved t-SNE based manifold dimensional reduction for remote sensing data processing. *Multimedia Tools and Applications*, 78(4):4311–4326, 2019.
- [16] El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe’er. viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature biotechnology*, 31(6):545, 2013.
- [17] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
- [18] Andrej Gisbrecht, Alexander Schulz, and Barbara Hammer. Parametric nonlinear dimensionality reduction using Kernel t-SNE. *Neurocomputing*, 147:71–82, 2015.
- [19] Jiarui Ding, Anne Condon, and Sohrab P Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature communications*, 9(1):2002, 2018.
- [20] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Gehrmann, and Evan W Newell. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology*, 37(1):38, 2019.
- [21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.
- [22] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.
- [23] 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.

- [24] Bernd Fritzke. Growing Cell structures—a self-organizing network for unsupervised and supervised learning. *Neural networks*, 7(9):1441–1460, 1994.
- [25] Bernd Fritzke. A Growing Neural Gas network learns topologies. In *Advances in neural information processing systems*, pages 625–632, 1995.
- [26] Tomihisa Kamada, Satoru Kawai, et al. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.
- [27] Caleb Weinreb, Samuel Wolock, and Allon M Klein. SPRING: a kinetic interface for visualizing high dimensional single-cell expression data. *Bioinformatics*, 34(7):1246–1248, 2018.
- [28] Pablo A Estévez, Cristián J Figueroa, and Kazimo 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.
- [29] Daminda Alahakoon, Saman K Halgamuge, and Bala Srinivasan. Dynamic Self-Organizing Maps with controlled growth for knowledge discovery. *IEEE Transactions on neural networks*, 11(3):601–614, 2000.
- [30] James Munkres. *Topology*. Pearson Education, 2014.
- [31] Jean-Claude Fort, Patrick Letremy, and Marie Cottrell. Advantages and drawbacks of the Batch Kohonen algorithm. In *ESANN*, volume 2, pages 223–230, 2002.
- [32] Tomas 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.
- [33] Chon-Kit Kenneth Chan and Saman Halgamuge. Investigation of average mutual information for species separation using GSOM. In *International Conference on Future Generation Information Technology*, pages 42–49. Springer, 2009.
- [34] Michael Thomas Wong, David Eng Hui Ong, Frances Sheau Huei Lim, Karen Wei Weng Teng, Naomi McGovern, Sriram Narayanan, Wen Qi Ho, Daniela Cerny, Henry Kun Kiaang Tan, 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.

- [35] 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.
- [36] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [37] 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.
- [38] Nguyen Xuan Vinh, Julien Epps, 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.
- [39] 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.
- [40] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia Object Image Library (coil-20). 1996.
- [41] Luke C Gandolfo and Terence P Speed. Rle plots: Visualizing unwanted variation in high dimensional data. *PLoS one*, 13(2):e0191629, 2018.
- [42] 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 is a PhD candidate at the University of Melbourne, who completed his BSc in Computer Science and Engineering at the University of Moratuwa in 2015. His research focuses on developing analysis methods for highly dimensional data using neural networks, computational and algebraic geometry and topology, with applications in bioinformatics.

Wei Wang, PhD 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 focuses on deep generative models, Bayesian statistics and image processing, with a current interest in exploring compact and highly performant deep neural network structures.

Shalin Naik, PhD 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 (FIEEE) is a Professor in the Department of Mechanical Engineering, and former Director of Research School of Engineering at the Australian National University, who completed his PhD at the Technical University Darmstadt. He was a member of Australian Research Council (ARC) College of Experts for Engineering, Information and Computing Sciences. His fundamental research contributions are in Unsupervised and Near Unsupervised type learning and Bioinspired Optimization.

Algorithm 1: SONG Algorithm with Decaying Learning Rate

```

1  $t \leftarrow$  Iteration index, initialized as 0;
2  $t_{max} \leftarrow$  Maximum number of iterations ;
3  $d \leftarrow$  Output dimensionality; usually 2 or 3;
4  $k \leftarrow$  Number of neighbors to consider at a given locality,  $k \geq d + 1$ ;
5  $\alpha \leftarrow$  Learning rate starting at  $\alpha_0$ ;
6  $\mathbf{C} \leftarrow$  Random matrix of size  $(d + 1) \times D$ ;
7  $E \leftarrow$  Edges on  $\mathbf{C}$ , from each  $\mathbf{c}$  to other  $\mathbf{c}$ s, all initialized as non-edges (0);
8  $\mathbf{Y} \leftarrow$  Random matrix of size  $(d + 1) \times d$ ;
9  $r \leftarrow$  Number of negative edges to select, per positive edge for negative sampling;
10  $a, b \leftarrow$  Appropriate parameters to get desired spread and tightness as per Eq. 6;
11  $\theta_g \leftarrow$  User defined growth threshold;
12 while  $t < t_{max}$  do
13   for  $x_i \in X$  do
14     Update  $E_s$  as per Section 3.1;
15     Update  $\mathbf{I}^{(k)}$ ;
16      $ns = r \cdot n(\hat{e}_{i_1})$ , here  $n(\hat{e}_{i_1})$  is the number of edges from or to  $\mathbf{c}_{i_1}$ ;
17     Record the neighbors of  $\mathbf{c}_{i_1}$  as  $\mathcal{N}_{i_1}^{t-1} \leftarrow \{j \mid E_s(i_1, j) > 0\}$  ;
18     Perform Edge Curation as per Algorithm 2 ;
19     Record the new set of neighbors  $\mathcal{N}_{i_1}^t$ ;
20     if  $\mathcal{N}_{i_1}^{t-1} == \mathcal{N}_{i_1}^t$  then
21       | End the execution of the algorithm and return;
22     end
23     Perform Self-Organization of Coding Vectors  $\mathbf{C}$ :
24     for  $j \in \mathcal{N}_{i_1}^t$  do
25       |  $\mathbf{c}_j \leftarrow \mathbf{c}_j + \alpha \cdot \frac{\partial \mathcal{L}(\mathbf{x}_i)}{\partial \mathbf{c}_j}$ 
26     end
27     Update  $\mathbf{Y}$  as per Algorithm 3;
28      $G_{i_1} \leftarrow G_{i_1} + \|\mathbf{x}_i - \mathbf{c}_{i_1}\|$ ;
29     if  $G_{i_1} > \theta_g$  then
30       | Grow  $\mathbf{C}$  and  $\mathbf{Y}$  as per Algorithm 4;
31     end
32   end
33    $t \leftarrow t + 1$ ;
34    $\alpha \leftarrow \alpha_0 \times (1 - \frac{t}{t_{max}})$ ;
35 end

```

Algorithm 2: Updating the Directional Edges in $\mathbf{E}$ between Coding Vectors based on $\mathbf{x}_i$

```

1 for  $j \in \mathbf{I}^{(k)}$  do
2   if  $\|\mathbf{x}_i - \mathbf{c}_j\| \leq \|\mathbf{x}_i - \mathbf{c}_k\|$  then
3     | Renew edges as  $E(i_1, j) = 1$ ;
4   else
5     | Decay edges as  $E(i_1, j) \leftarrow \epsilon \cdot E(i_1, j)$ ;
6   end
7   if  $E(i_1, j) < e_{min}$  then
8     | Prune edges as  $E(i_1, j) \leftarrow 0$ ;
9   end
10 end

```

Algorithm 3: Topology Preservation of the Low-dimensional Points $\mathbf{Y}$

```

1 /* Organization of Local Neighborhood */;
2 for  $j \in \mathcal{N}_{i_1}^t$  do
3   |  $\mathbf{y}_j \leftarrow \mathbf{y}_j + \alpha \cdot (\mathbf{y}_{i_1} - \mathbf{y}_j) \cdot \frac{2ab \cdot \hat{e}_{i_1 j} \cdot \|\mathbf{y}_j - \mathbf{y}_{i_1}\|^{2b-2}}{1 + \|\mathbf{y}_j - \mathbf{y}_{i_1}\|^{2b}}$ 
4 end
5 /* Negative Sampling for Repulsion */;
6 Select  $ns$  random samples  $J = \{j_1, \dots, j_{ns}\}$  with  $E_s(i_1, j) = 0$ ;
7 for  $j \in J$  do
8   |  $\mathbf{y}_j \leftarrow \mathbf{y}_j - \alpha \cdot (\mathbf{y}_{i_1} - \mathbf{y}_j) \cdot \frac{2b}{\|\mathbf{y}_j - \mathbf{y}_{i_1}\|^2 (1 + \|\mathbf{y}_j - \mathbf{y}_{i_1}\|^{2b})}$ 
9 end

```

Algorithm 4: Growing $\mathbf{C}$ and $\mathbf{Y}$ to Refine the Inferred Topology

```

1 create new coding vector such that  $\mathbf{w}_n \leftarrow \frac{1}{k} \sum_{l=\{1 \dots k\}} \mathbf{w}_{i_l}$ ;
2 create new low-dimensional vector such that  $\mathbf{y}_n \leftarrow \frac{1}{k} \sum_{l=\{1 \dots k\}} \mathbf{y}_{i_l}$ ;
3 for  $j \in \{i_1, \dots, i_k\}$  do
4   |  $E(j, n) = 1$ ;
5 end

```

Table 1: Hyperparameters used for SONG, Parametric t-SNE, t-SNE and UMAP throughout our experiments

Algorithm	Hyperparameters
SONG/SONG-Reinit	$k = 2, \epsilon = 0.99, t_{\max} = 100, \alpha_0 = 1.0, a = 1.577, b = 0.895$
Parametric t-SNE	Perplexity = 30, epochs = 400, batch_size = 128
t-SNE	Perplexity = 30
UMAP	n_neighbors = 15, , $\alpha_0 = 1.0, a = 1.577, b = 0.895$

Table 2: AMI Scores on Heterogeneous Increments of Fashion MNIST and MNIST datasets. For fair comparison model-retaining methods (SONG and Parametric t-SNE) and model-reinitializing methods (SONG + Reinit, t-SNE and UMAP) are separated. The best AMI scores are highlighted.

No. Classes	Fashion MNIST					MNIST				
	2	4	6	8	10	2	4	6	8	10
SONG	70.9	86.1	84	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

Table 3: AMI scores of the visualized homogeneous increments of the Fashion MNIST and MNIST datasets. In SONG and Parametric t-SNE, a trained model from one intermediate dataset is updated and used to visualize the next dataset. SONG + REINIT, t-SNE, and UMAP are reinitialized and retrained at each increment.

	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-Reinit	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

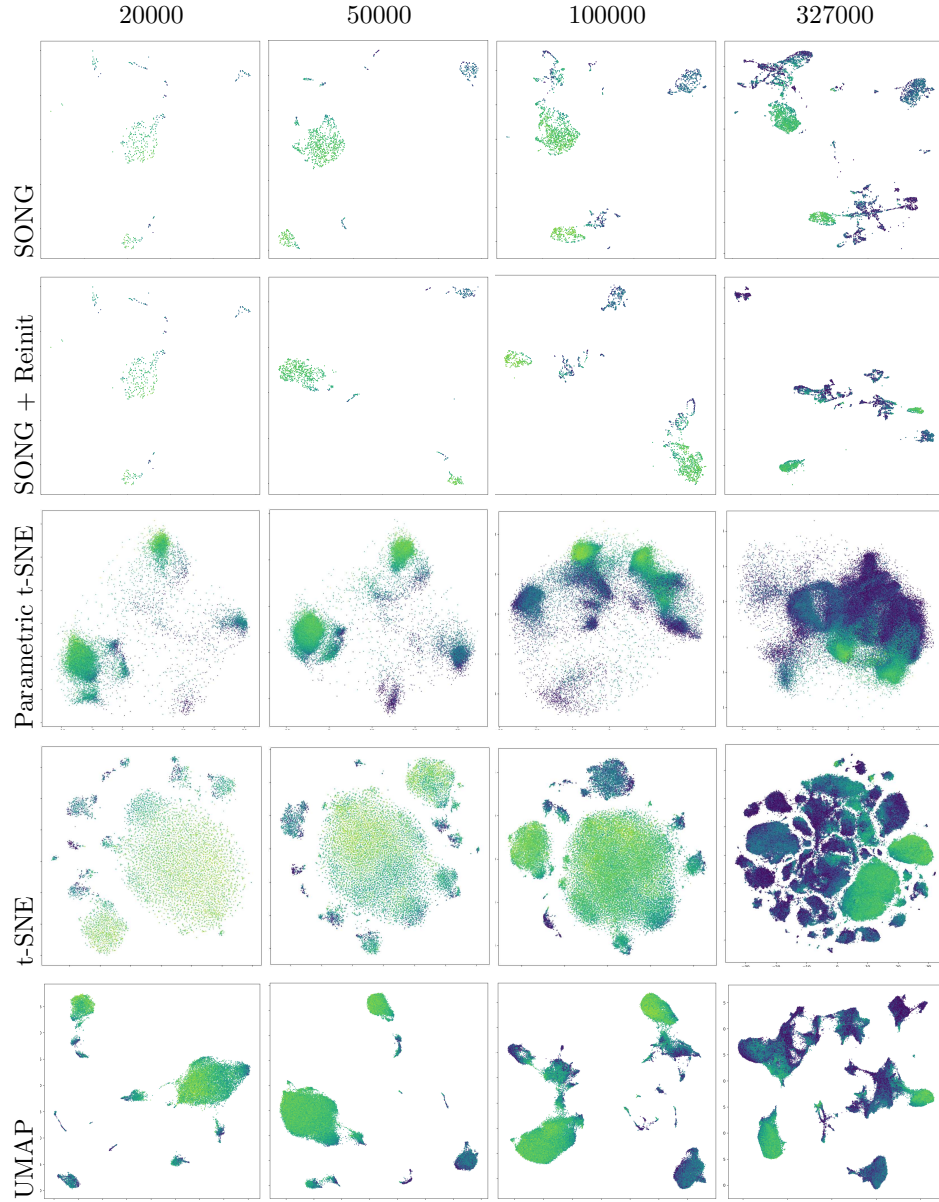


Figure 2: The Wong dataset visualized by SONG, SONG + Reinit, Parametric t-SNE and UMAP. The colors represent the CCR7 expression levels following the visualizations provided in [20], where Light Green represents high CCR7 expression and Dark Purple represents low CCR7 expressions.

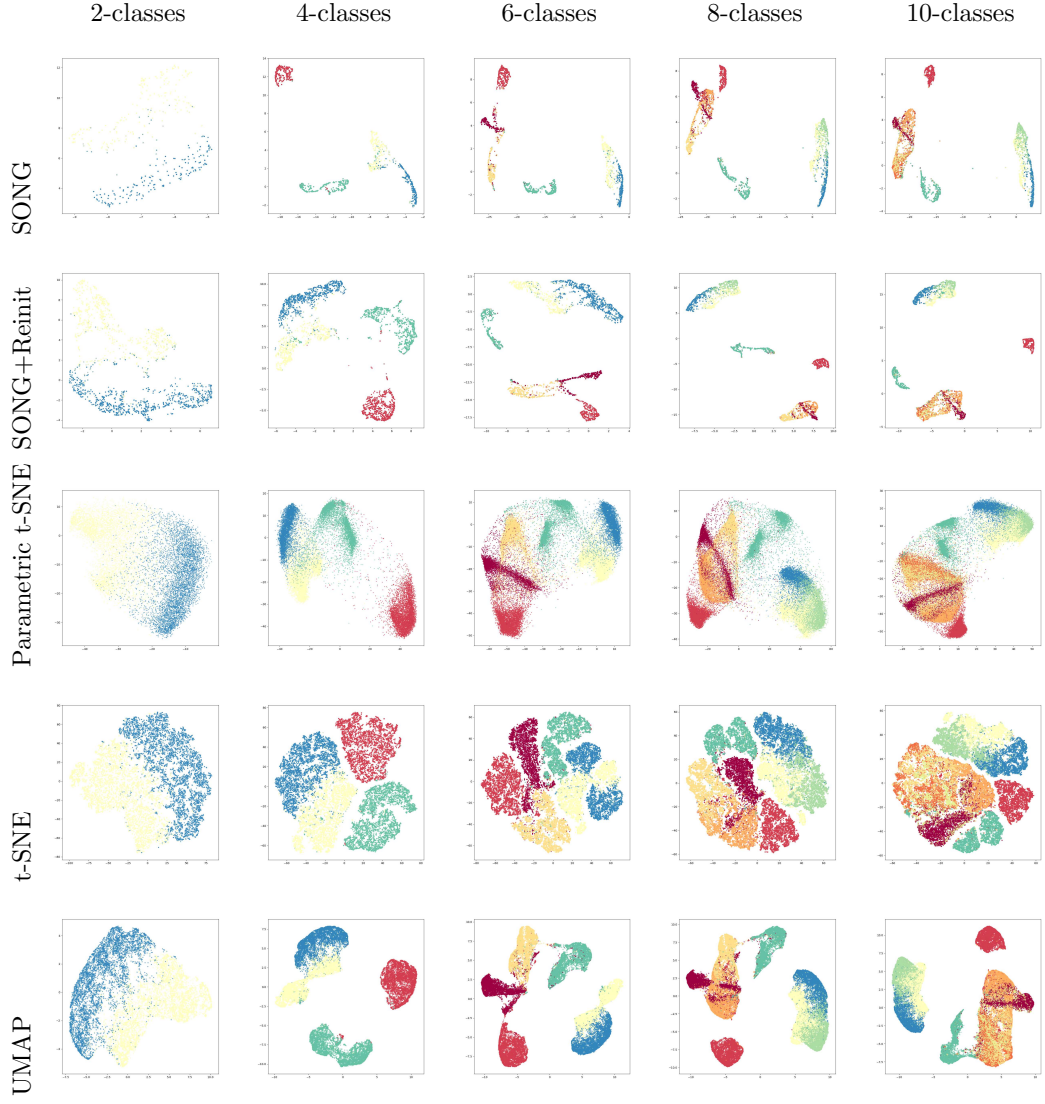


Figure 3: Incremental visualization of the Fashion MNIST dataset using SONG, SONG + Reinit, Parametric t-SNE, t-SNE and UMAP, where two classes are added at a time.

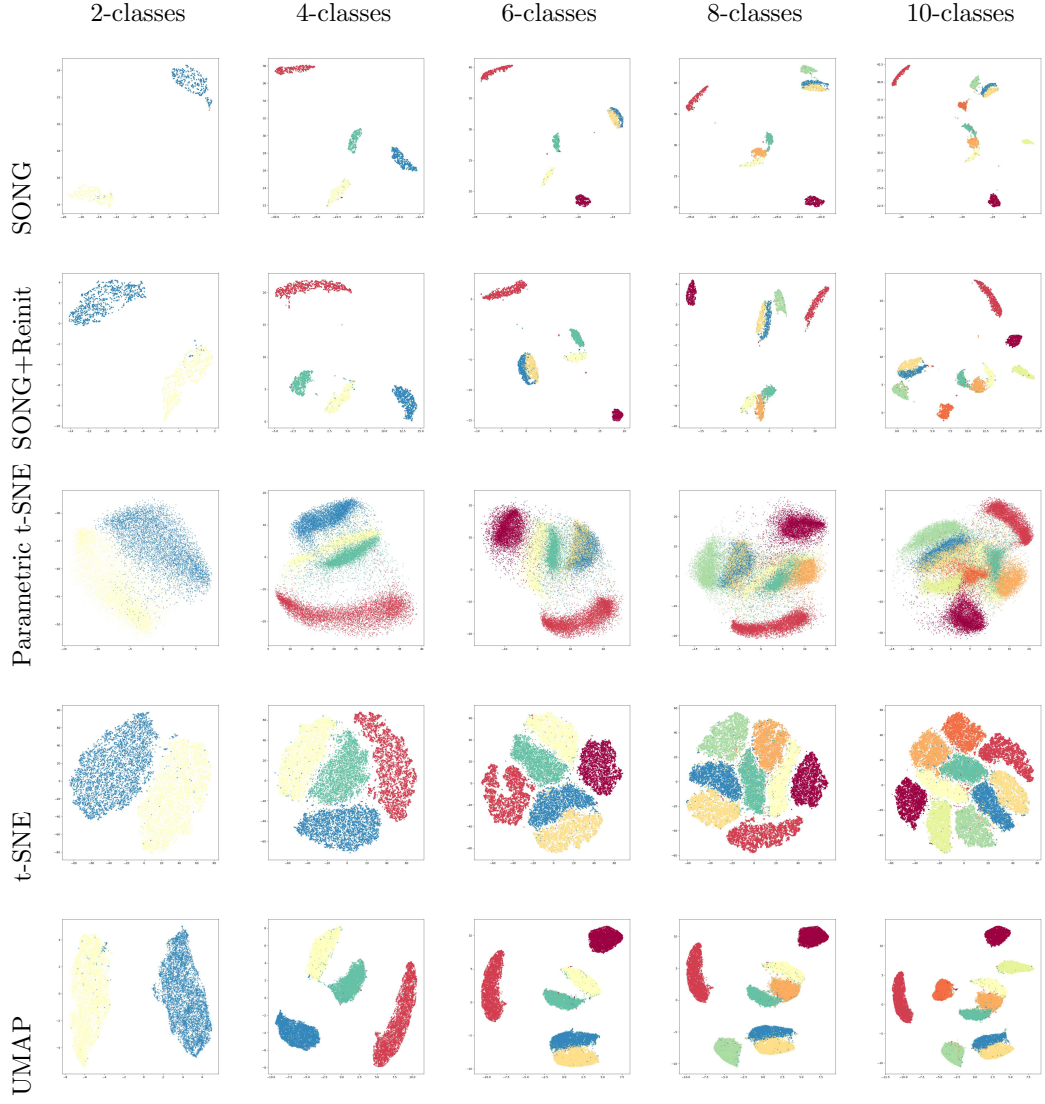


Figure 4: Incremental visualization of the MNIST dataset using SONG, SONG + Reinit, Parametric t-SNE, t-SNE and UMAP, where two classes are added at a time.

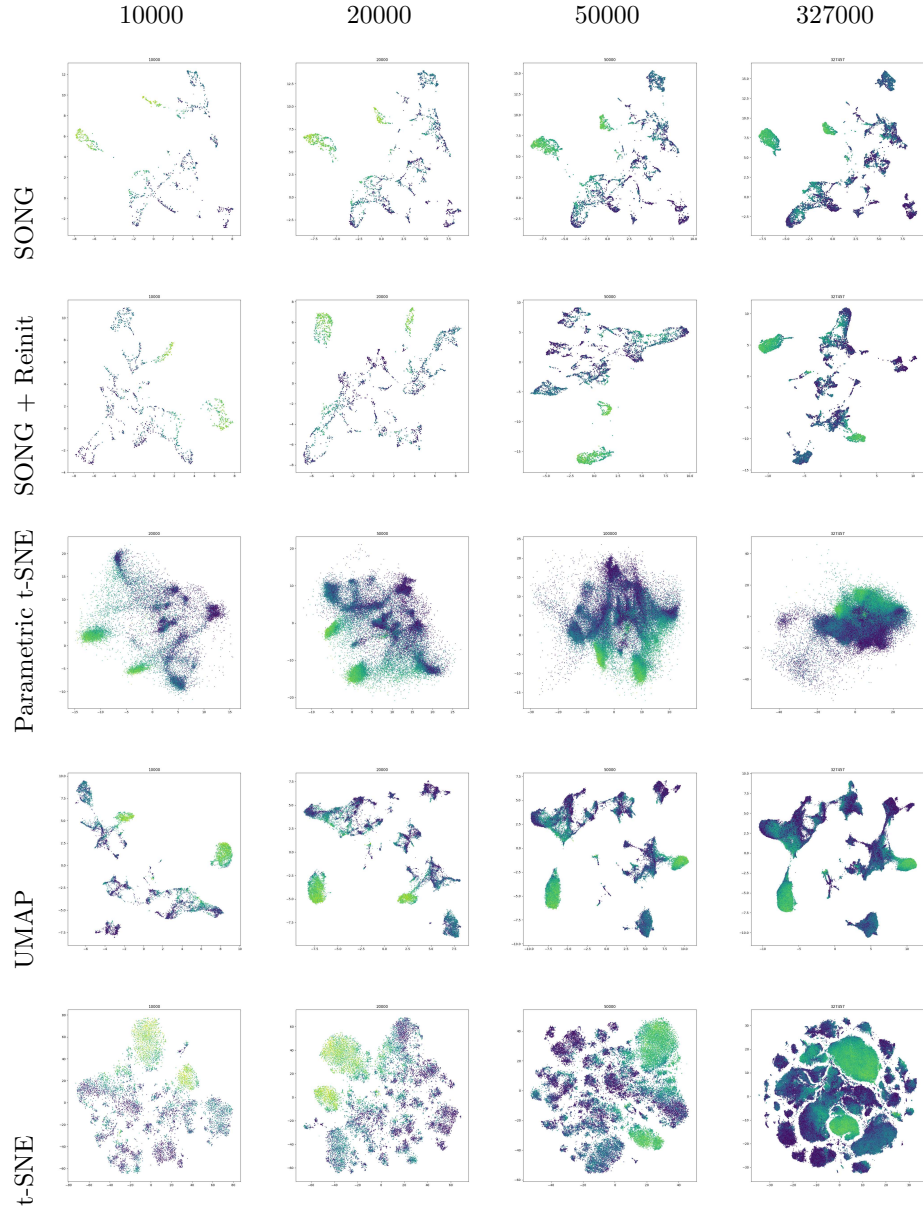


Figure 5: Random samples of varying sizes from the Wong dataset presented to SONG, Parametric t-SNE, SONG + Reinit , UMAP and t-SNE incrementally.

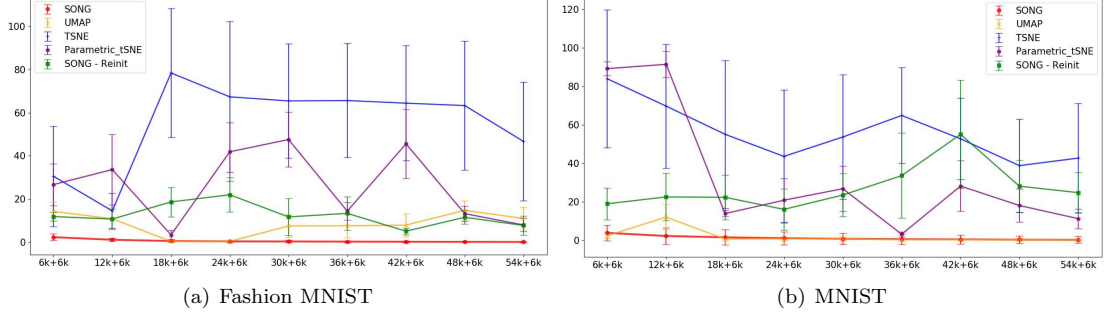


Figure 6: The average Consecutive Displacement of Y for each established point after subsequent presentation of 6000 images to each algorithm.

Table 4: The AMI scores for different Gaussian Blobs configurations in 60 Dimensions. Each configuration of Gaussian Blobs have different numbers of clusters and different cluster standard deviations. Having a large number of clusters and a large standard deviation increases the probability of mixing of clusters.

		SONG				UMAP				t-SNE			
		<i>No. Clusters</i>											
		10	20	50	100	10	20	50	100	10	20	50	100
Cluster Std. Deviation	4	100	100	100	100	100	100	100	100	100	100	100	100
	8	99.9	99.6	99.2	99.1	99.9	99.2	98.4	97.1	99.7	99.2	98.0	96.0
	10	97.6	95.4		89.2	96.8	92.8	84.4	67.9	95.7	90.4	80.3	68.4
				91.4									
	12	90.0	84.1	75.2	65.8	86.5	75.4	51.0	33.2	82.7	68.5	48.5	34.5
	14	77.6	66.6	52.2	39.7	70.9	51.3	24.8	19.6	65.8	43.5	23.0	21.3
	16	62.8	49.2	31.2	21.2	54.2	30.3	13.4	15.8	46.7	23.4	12.8	17.5
	18	50.0	35.2	16.7	16.0	36.8	17.2	8.7	14.5	32.9	13.7	8.7	15.6
	20	38.1	22.6	10.2	14.6	25.7	10.3	6.84	13.6	22.7	9.1	7.0	14.5

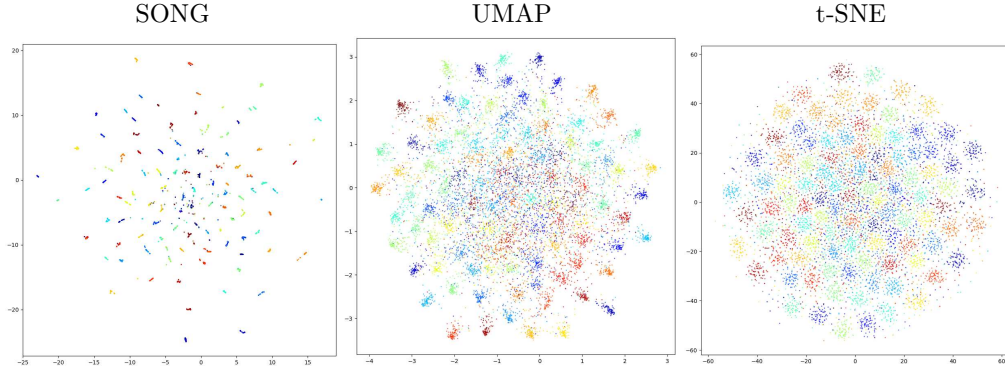


Figure 7: The visualizations of a dataset having 100 clusters, each cluster having a standard deviation of 10, and 60 dimensions using the three methods SONG, UMAP and t-SNE

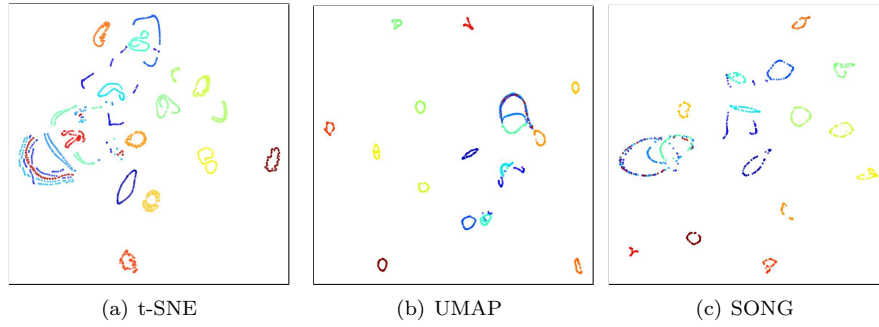


Figure 8: COIL-20 dataset when reduced using the three algorithms a)t-SNE, b)UMAP and c)SONG. Both UMAP and SONG preserve the circular topologies, even in clusters where the classes are not well separated, to a greater degree than t-SNE.

Supplementary Material for Self Organizing Nebulous Growths for Robust and Incremental Data Visualization

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

1 METHOD DETAILS

In this section, we first describe in detail the initialization of Coding Vectors, and the corresponding embedding. Next we describe how we control the growth of the coding vectors. At last, we describe how we approximate the hyperparameters required for output embedding optimization.

1.1 Initialization of Coding Vectors for Topology Preserving Visualization

The number of initial coding vectors $\mathbf{C} = \{\mathbf{c} \in \mathbb{R}^D\}$ can be arbitrarily small if we only wanted to infer the topology of the input $\mathbf{X}$. However, in addition to inferring the topology, in our proposed SONG algorithm, we attempt to obtain a visualization of the coding vectors as an embedding $\mathbf{Y} = \{\mathbf{y} \in \mathbb{R}^d\}$ in addition to the inference of the topology. Furthermore, we start with a very small number of coding vectors, and grow the sets $\mathbf{C}$ and $\mathbf{Y}$ as needed to infer the finer topology of the dataset $\mathbf{X} = \{\mathbf{x} \in \mathbb{R}^D\}$, compared to the topology inferred by the initial set of coding vectors.

We cannot pick an arbitrarily small number of coding vectors initially because this presents a problem in growing the set of output vectors, which we will explain below. Our growth of $\mathbf{Y}$ relies on calculating the centroid of the $\mathbf{y}_j \in \mathbf{Y}$ such that $j \in I_i^{(k)}$, where $I_i^{(k)}$ is the index set of the k -nearest coding vectors to an input $\mathbf{x}_i \in \mathbf{X}$. If $k \leq d$, where d is the dimensionality of the embedding, we can always find a hyperplane in dimensionality d such that all of the k points in the $\mathbf{y}$ neighborhood lies on. For example, if $d = 2$, any 2 points can be connected by a line and if $d = 3$, any 3 points can be considered to be coplanar. Therefore, any new created $\mathbf{y}$ that lies on the centroids of these neighborhoods are going to be coplanar to the neighborhood. This provides visualizations as a straight line in 2-dimensions, and as a plane in 3 dimensions (thereby collapsing one dimension of the output). Therefore, we need at least $d + 1$ coding vectors at the beginning, so that the visualizations will not be colinear (in 2d) or coplanar (in 3d).

1.2 Growth Threshold θ_g Approximation

To regulate the growth of coding vectors, we use the growth threshold as defined by Alahakoon et al[1] and Chan et al [2]. This allows us to fix the hyperparameter that we expose to the user to be fixed in the range $(0, 1)$.

We define a spread factor SF such that:

$$\theta_g = -D^{\frac{1}{p}} \log(SF) \quad \text{Equation 1.}$$

where $SF \in (0, 1)$.

2 CHOOSING THE PARAMETERS FOR THE OUTPUT EMBEDDING

We draw inspiration from UMAP and follow their rationale for choosing a and b as we describe in the following. For the low-dimensional pseudo-probability q_{ij} in our algorithm, we use the function:

$$q_{ij} = \frac{1}{1 + a\|\mathbf{y}_i - \mathbf{y}_j\|^{2b}} \quad \text{Equation 2.}$$

By regulating the a and b values, we can regulate how tight the neighborhoods are in the output visualizations. Having larger a increases the size of the clusters and having a larger b increases the distance between two clusters. This is because of having a light tail in the function improves the separation of clusters (by having a low probability that two neighboring output points are placed far from each other). Therefore, a and b are found by fitting the q_{ij} function on a curve f of the form :

$$f(i, j) = \begin{cases} 1; & \text{if } \|\mathbf{y}_i - \mathbf{y}_j\| < \text{min_dist} \\ \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\| - \text{min_dist}}{\text{spread}}\right); & \text{otherwise} \end{cases} \quad \text{Equation 3.}$$

Here, min_dist is the minimum distance we expect to be between two output points, and spread regulates the spread of a given cluster, as described in [3]

3 SUPPLEMENTARY RESULTS

3.1 Visualizations for the Extended Collection of Simulated Datasets

We direct our readers to the following link: http://bit.ly/song_gauss_blobs, where we have compiled the visualizations provided by SONG, UMAP and t-SNE. The link contains 3 folders

Gaussian_blobs_SONG, Gaussian_blobs_UMAP and Gaussian_blobs_TSNE, each folder containing the visualizations provided by the method corresponding to the folder name. The files are named by the cluster standard deviation, number of clusters and the number of dimensions as ‘cs_{cluster standard deviation}_nc_{number of clusters}_nd_{number of dimensions}.png’. Each folder contains 125 scatter plots, 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, we observe that SONG has better separation of clusters where UMAP and t-SNE show fuzzy cluster boundaries when the level of cluster mixing increases. This contributes to higher cluster purity as reflected by the Adjusted Mutual Information scores associated with the visualizations.

3.2 Running Time Comparison of SONG with UMAP and t-SNE

We compare the running time of three algorithms SONG, UMAP and t-SNE on the Wong, Fashion MNIST and MNIST datasets. The execution times were recorded on a computer with an Intel Xeon Gold 6154 CPU (3.00 GHz), and 48 GB of total memory, using Python 2.7.15+ as the common interpreter for the three algorithms. Our results are summarized in the Table 1. In this case, both MNIST and Fashion MNIST have been reduced to 20 Principal Components. Wong dataset has 39 features representing different surface marker expressions. We see that UMAP is the fastest of the three algorithms for all three datasets and t-SNE is the slowest. SONG has performance close to UMAP.

Dataset	No. Datapoints	SONG	UMAP	t-SNE
MNIST	60000	180s	67s	679s
Fashion MNIST	60000	95s	69s	616s
WONG	327000	1203s	670s	20309s

Table S 1. Running times of the three algorithms SONG, UMAP and t-SNE for three datasets. We see that SONG has better performance than t-SNE and slower performance than UMAP.

3.3 Sensitivity to Hyper-parameters

We have divided the hyper-parameters employed in SONG into two categories: A) hyper-parameters used for graph inference: initial learning rate, number of neighboring coding vectors in the immediate neighborhoods, edge-strength decay rate (ϵ) and the spread factor, and B) hyper-parameters governing the projection into visualization space: a and b , which can be regulated with spread and

min_dist parameters.

We constructed a dataset with 4000 data points uniformly and randomly distributed on a 3D sphere, and used SONG with 3 different configurations to obtain visualizations. We summarize these configurations in Table 2.

Configuration	Fixed Hyper-parameters	Variable Hyper-parameters
C1	$\alpha_0 = 1.$, min_dist = 0.1, spread = 1.0, $\epsilon = 0.99$	$SF \in \{0.01, 0.1, 0.5, 0.9, 0.99\}$, $k \in \{1, 2, 5, 10, 15\}$
C2	$SF = .99$, min_dist = 0.1, spread = 1.0, $k = 2$	$\epsilon \in \{0.0001, 0.1, 0.5, 0.9, 0.999\}$, $\alpha_0 \in \{0.01, 0.1, 0.2, 0.5, 1.0\}$
C3	$SF = .99$, $\alpha_0 = 1.$, $\epsilon = 0.99$, $k = 2$	min_dist $\in \{0.01, 0.05, 0.1, 0.2, 0.5\}$, spread $\in \{0.1, 0.2, 0.5, 1.0, 2.0\}$

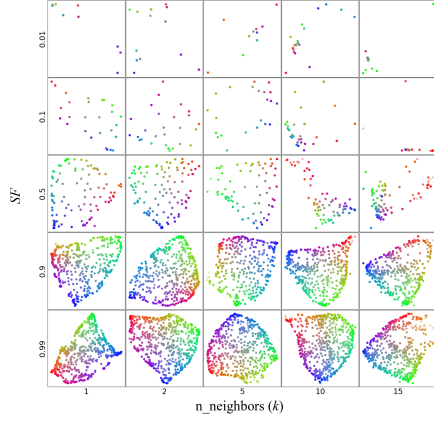
Table S 2. Three different configurations of SONG used for this experiment.

In each of these configurations, for each variable hyperparameter, we obtained a visualization for our input dataset, and we summarize this qualitative result in Figure 1.

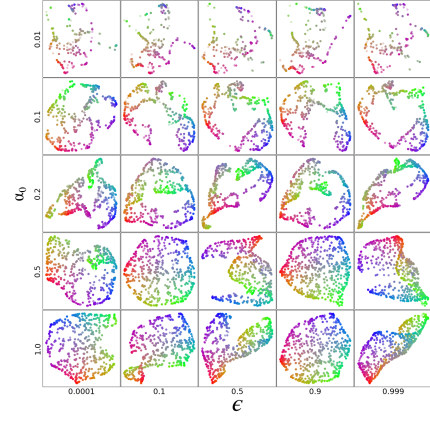
3.4 Kernel t-SNE performance on heterogeneous increments

In this section we use kernel t-SNE to visualize the MNIST and Fashion MNIST datasets given as homogenous and heterogeneous increments, to investigate the ability of a trained kernel t-SNE model to generalize to new data.

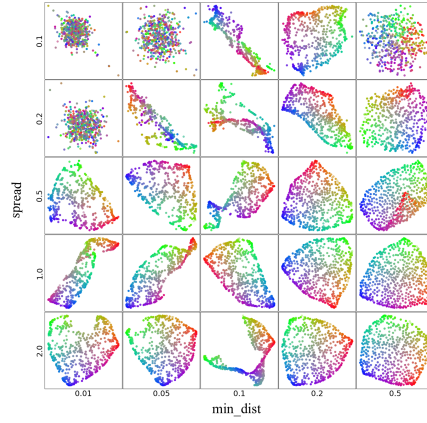
In this experiment, we presented one training set, and two test sets, sampled from two separate datasets: MNIST and Fashion MNIST. Each training set contains 10000 data points belonging to 5 classes from the input data, and the two test sets are presented as: a) 10000 homogeneous inputs (data points belonging to the same 5 classes of the training set) and b) 10000 heterogeneous inputs (data points belonging to 5 new classes not shown in the training set). Results for kernel t-SNE on the two datasets are shown in Figure 2 and SONG in Figure 3. For both datasets considered, Kernel t-SNE is capable of mapping the new data into existing clusters, and performs well provided that the new data is sampled from these classes. However, when new classes are presented, Kernel t-SNE fails at identifying new clusters or placing them separately. In contrast, Figure 3 shows that



C1



C2



C3

Figure S 1. Visualization results for configurations C1, C2 and C3

SONG refines the inferred clusters when presented with data sampled from known classes, and creates new clusters to represent the data from new classes.

In Section IV Experiments and Results, parametric t-SNE was considered as the closest alternative to the proposed SONG among all compared methods (parametric t-SNE, t-SNE and UMAP). This is due to the ability of parametric t-SNE to learn the distributions of heterogenous increments.

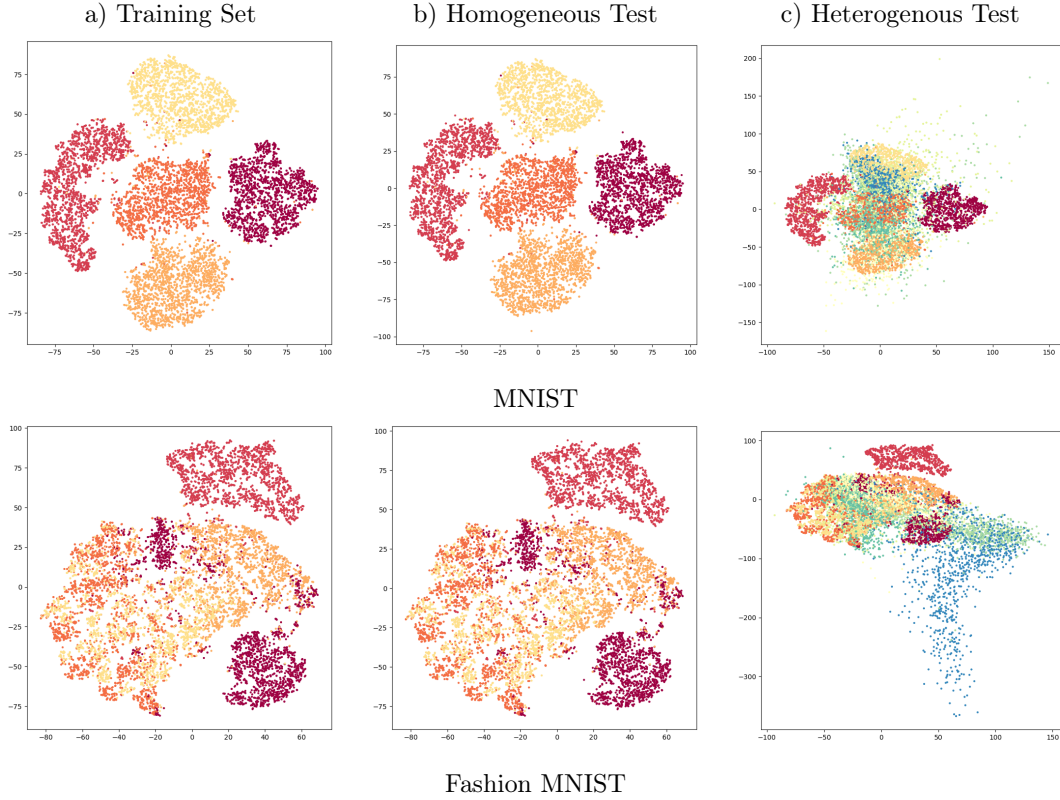


Figure S 2. a) Kernel t-SNE embedding of 10000 images belonging to 5 classes is used to calculate the gram matrix. b) the calculated gram matrix is used to embed 10000 new images from the same 5 classes. c) 10000 random images from 5 new classes) embedded using the gram matrix.

Although compared to parametric t-SNE, kernel t-SNE tends to provide relatively less noisy and more separated visualization for the homogeneous increments [4], here we have shown that kernel t-SNE is not designed to incorporate heterogeneous data increments.

In conclusion, while kernel t-SNE does retain a parametric mapping from the input space to the output space, it may not generalize well to data increments with heterogeneous distributions. This can be attributed to the fact that the once the parameters are calculated on a training set, the parameters are not adjusted to suit the distribution of the new data.

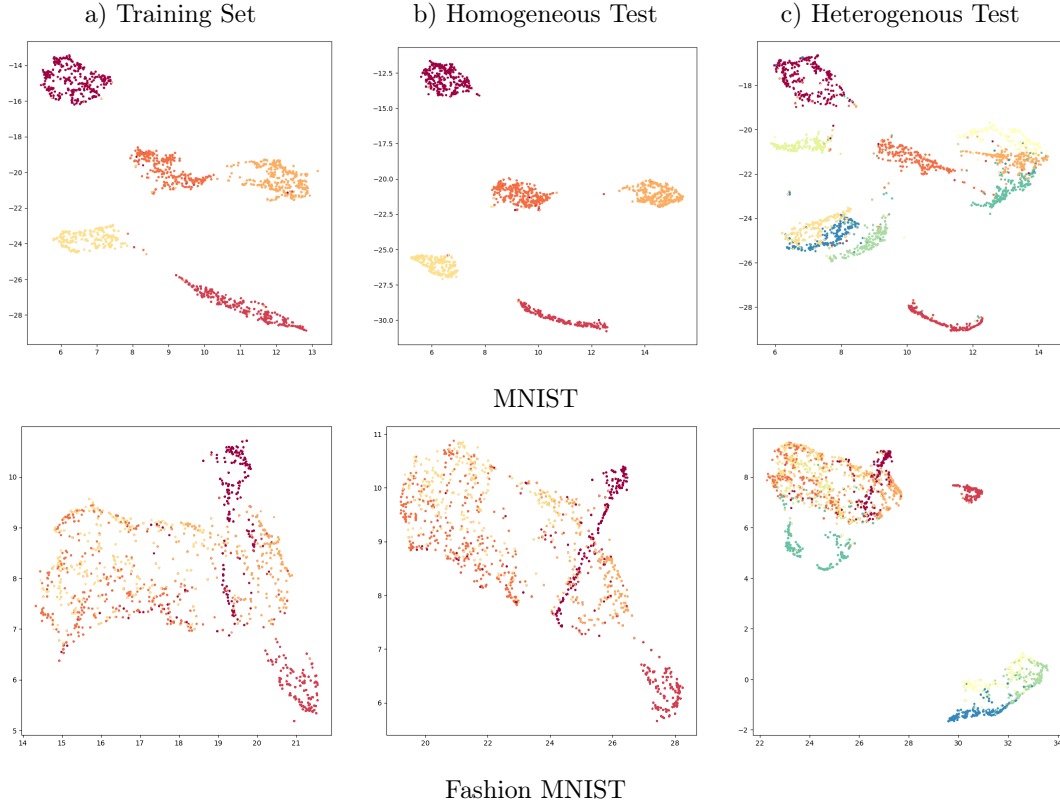


Figure S 3. a) 10000 images from 5 classes used to train the initial SONG model, b) 10000 new images from the same 5 classes further differentiate the clusters and reinforce the visualization c) 10000 random images with 5 new classes alter the structure to represent the new clusters.

REFERENCES

- [1] Daminda Alahakoon, Saman K Halgamuge, and Bala Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on neural networks*, 11(3):601–614, 2000.
- [2] Chon-Kit Kenneth Chan and Saman Halgamuge. Investigation of average mutual information for species separation using gsom. In *International Conference on Future Generation Information Technology*, pages 42–49. Springer, 2009.
- [3] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

- [4] Andrej Gisbrecht, Alexander Schulz, and Barbara Hammer. Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, 147:71–82, 2015.

3.2 Summary and Contributions

In this work, we have proposed a new parametric dimensionality reduction algorithm called Self-Organizing Nebulous Growths. In addition to providing visualizations on par with state-of-the-art non-parametric dimensionality reduction algorithms such as t-SNE and UMAP, SONG can retain a parametric mapping from the input space to the output space. This parametric mapping is obtained by retaining the sets of coding vectors ($\mathbf{C}$), edges between coding vectors ($\mathbf{E}$) and the corresponding low-dimensional points ($\mathbf{Y}$), as we have described in Section II of the above manuscript.

With this work, we have conclusively shown that the SOM-based algorithm we proposed, SONG, far out-performs the deep learning based alternatives, the most prominent of which is the Parametric t-SNE algorithm. We have further shown that SONG is superior in terms of reducing dimensionality and visualizing data in the presence of highly noisy datasets.

A thorough analysis needs to be conducted on the philosophical question - “How much do we prefer the *old* visualization to retain its structure in the presence of new data, especially if we do not know the level of heterogeneity present in the increments”. We propose the introduction of a parameter to regulate this preference called the ‘agility’ of a parametric mapping. In our follow-up work which we describe in the next chapter, we investigate two main aspects of the SONG algorithm which can be used in RNA-Seq data visualization: 1) The incremental visualization of multi-source and heterogeneous RNA-Seq data in a meaningful manner and 2) The use of SONG to provide canonical representations and/or abstractions of heterogeneous cell populations.

Chapter 4

How can the use of SONG algorithm improve the analysis of RNA-Seq Data

Overview

In this chapter we present a framework for incremental visualization and abstraction of Single Cell data. We show that SONG can visualize data incrementally, whether this data is homogeneous or heterogeneous to the currently encountered data. We also apply the SONG algorithm in integrating multiple datasets on a single visual map, and use three datasets obtained from the Human Cell Atlas project to demonstrate this. We further propose a framework for obtaining abstractions which can act as ‘landmark cells’ representing prototype cells for a population or a sub-population of cells.

4.1 Introduction

Single Cell experiments are a current and highly influential branch in biology. Facilitated by the recent advances of high throughput sequencing and microfluidics, single cell studies have provided insights into cellular mechanics at a higher resolution than bulk or population studies. The main advantage of having such a fine-grained analysis is that it allows us to explore inter-cellular heterogeneity with much greater accuracy.

Single cell experiments often gather a large number of measurements per each cell. RNA-Seq experiments gather readings of thousands of genes that may be expressed at any given time and surface marker expressions are gathered by tens. This essentially provides the analysts with highly dimensional datasets. As with any form of science in its infancy, Single Cell experiments suffer from the lack of ground-truth. The vast majority of the underlying mechanics of individual cells are unknown, hence any analysis would have to be preceded by exploratory analysis.

Exploratory analysis (or unsupervised learning) in single-cell experiments therefore heavily relies on visualization of highly dimensional data. In fact, it is a pivotal and one of the most important steps in the analysis of single-cell data analysis. Methods for efficiently and robustly visualizing complex structures in an intuitive manner are therefore in high demand.

Recent methods such as t-SNE and UMAP have been used widely in the domain of single-cell data visualization. However, one frequently encountered shortcoming of these methods is its lack of global topology preservation when repeating visualizations.

As we have shown in our previous chapter, SONG provides a robust alternative to t-SNE and UMAP in terms of providing topologically consistent visualizations. Furthermore, in this chapter, we hope to examine how SONG can help identify ‘cellular prototypes’ in a more robust way than existing methods.

SONG allows an evolving single cell atlas with landmarks and spatially persistent visualizations

Damith Senanayake, Luyi Tian, Saman Halgamuge, Shalin H. Naik

ABSTRACT

Two-dimensional representation of multi-dimensional data derived from single cell analyses has had great utility for the understanding of cellular heterogeneity. Non-linear algorithms such as tSNE and UMAP are gold-standard methods that achieve visualisation of cluster separation. However, when new data is added, clusters may change their two dimensional location, i.e., such methods do not allow spatial preservation or ‘landmarking’ when new data is added – a desirable feature for Atlas projects that rely on continuous addition of new data that be incorporated with prior to data. We present Self Organizing Nebulous Growth (SONG) as a non-linear, parametric dimensionality reduction method that builds on neural gas and allows multi-omic, multi-donor incremental visualisation of single cell data. In addition, its features provide a novel approach for data harmonisation such as biological and technical variability. SONG can therefore provide a universal visually explorable and parametric reference for the Human Cell Atlas that can evolve.

1 Introduction

Single-cell studies have gained significant momentum over the recent years, facilitated by the advancements in microfluidics and sequencing technologies. The ability to investigate the cellular functions at a single-cell resolution has greatly advanced our knowledge about the functionality of numerous variables in complex cellular systems. For instance, vast heterogeneity of distinct groups of cells present in a given tissue is clearly visible when considering the properties of cells at the level of individual cells, which may be missed when analyzing the data in bulk.

In the recent years, with the advent of robust dimensionality reduction and visualization techniques such as t-SNE and UMAP, the analysis of multivariate single-cell data has become highly intuitive and convenient. A typical workflow involves the visualization of such highly multivariate or multi-dimensional data on a two or three dimensional map in order to gain information about the heterogeneity and structure of the data. Both t-SNE and UMAP are capable of providing clear cluster separation in visualizations, where these clusters represent different groups of cells, either in their functionality or in their lineage. Especially when it comes to UMAP, the visualization of highly dimensional data is laid out in an intuitively interpretable format where the separation between clusters is accentuated and the local neighborhoods are well-preserved.

Although t-SNE is a robust visualization technique which has been proven to provide robust visualizations in many fields, one of the major disadvantages observed in the t-SNE usage is its lack of global topology preservation when replicating experiments. This means that although cluster purities are maintained, the global placement of clusters is highly randomized when replicating experiments. The root of this problem can be traced to a pragmatic trade-off, where we compromise the global arrangement for a faster and accurate local cluster placement. UMAP on the other hand is at an advantage, where it uses a stable heuristic initialization of the visualization prior to optimization. However, we have shown in our previous work[?] that this too is dependent on the complexity of the low-dimensional manifold it approximates.

Not completely robbing both these ubiquitous methods of their merits, we identify that for tasks where the preservation of global structure is important, and where canonical representations of data is more important than providing visualizations that are faithful for a given dataset, both UMAP and t-SNE are ill-equipped. This is exemplified by recent work such as PAGA, where UMAP representations are used to obtain hierarchical canonical representations, (which in our work we dub ‘Landmarks’) are obtained by employing several assumptions, and is dependent on other algorithms.

In our previous works, we have shown that SONG is capable of incrementally learning visualizations of growing datasets, whether these increments are homogeneous or heterogeneous to the data that has already been observed. Building on the philosophical underpinnings that allow SONG to achieve this task, we now provide a framework wherein one can achieve a set of canonical representations without the need to average over arbitrarily defined population boundaries or partitioning of the data. Instead, we reuse the concept of vector quantization to provide us with the robust, hierarchical and canonical representations of the cells.

1.1 Cellular Landmarks Using SONG

The SONG algorithm employs a technique called ‘vector quantization’, i.e., the partitioning of an input vector-space such that each partition can be represented using a ‘coding vector’. The SONG algorithm’s foundations ensures that these coding

vectors are inferred in a hierarchical manner, where at the highest level of the hierarchy, there exists a small number of coding vectors, each encoding relatively large regions in the input space, and at the lowest levels each coding vector encodes a highly specific region. We propose that this hierarchical representations of the input space can be used for hierarchical prototyping of cells, and we call these cellular prototypes 'cellular landmarks'.

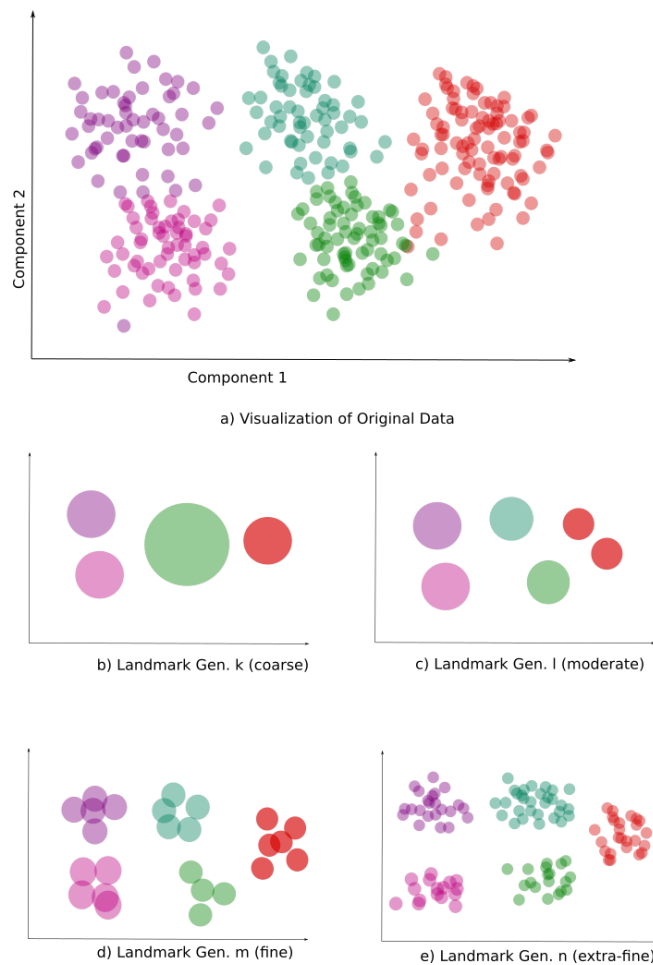


Figure 1. a) Two dimensional visualization of original data. Each cell is shown as a singular point in the scatter plot, and each color represents a different group of cells. We observe some amount of cluster mixing. In b), c), d) and e), each circle represents a landmark. The size of the landmark represents the size of the region each landmark represents. The color of the landmark represents the largest 'group' of cells in the region that landmark encodes. Landmarks are created as 'generations' where generations $k < l < m < n$. b) shows a highly coarse granularity of landmarks and this granularity gets more refined as more landmarks are created.

In Figure 1, we demonstrate how our proposed cellular prototyping behaves. We use SONG to first create a highly coarse encoding providing the landmarks at generation 1, where the generations that follow create landmarks with finer encodings. We here note that ideally, landmarks at the finest level we obtain, should account for the truly different groups of cells, and avoid encoding cells which can be considered high levels of noise. Because of this, we show that in Figure 1 e), we see less cluster mixing than in Figure 1 a).

In this study, we show that this landmarking can be obtained with incrementally presenting data to SONG.

2 Results

In this section we compare the performance of SONG against t-SNE and UMAP. We compare with these two methods as they are arguably the most widely-used methods for visualizations in RNA-Seq analysis.

In our previous work² we have shown that SONG is indeed capable of incorporating new data into an existing map. A further validation of the clustering quality of SONG (for a non-incremental dataset) compared to t-SNE and UMAP, as well as

its superior performance in homogeneous and heterogeneous increments of data is included in the Appendix. In this work, we investigate the capability of the SONG algorithm to integrate multi-source data in Atlas-like projects and the concept of genetic land-marking.

First, we investigate how SONG performs in integrating datasets from multiple sources.

2.1 Multi-Source Data Integration

setup: The main usage of using SONG as an incremental and contextual visualization of ever-growing datasets in large-scale projects such as the human cell atlas. We next use three datasets from the human cell atlas project to demonstrate this. The first dataset we use is the Human Retinal Cells dataset², which has 45k cells. The second one we use the complete atlas of activated human T Cells² having 200k cells, and the third, the Immune Cell Census² which has 770k cells. All in all, the three datasets combine to over 1 M cells.

As preprocessing, we used the incremental PCA provided by scikit-learn python packages to reduce all three datasets from a dimensionality of 63k down to 80. This aids us in obtaining a fast visualization.

Next, we use the SONG algorithm to visualize these three datasets in the order of a) Retinal Cells, b) T Cells, and c) Immune Cell Census. Note that since T Cells themselves are human immune cells, we expect an overlap between the datasets b) and c). However, the overlap between a) and the others is not expected to be high.

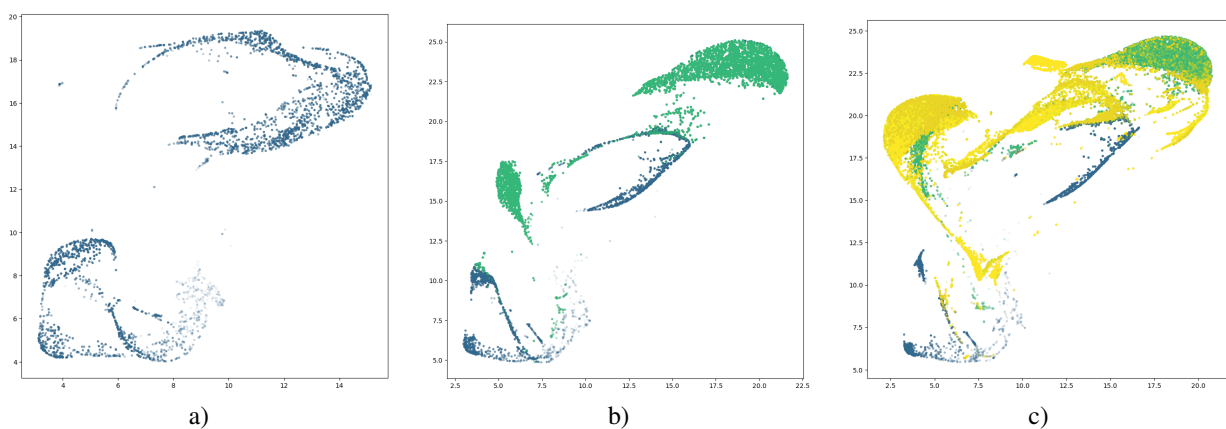


Figure 2. Visualizing 3 datasets from the Human Cell Atlas project. the first figure shows the initial data, and the subsequent figures show how the data is augmented subsequently.

result : In Figure 2 we see that the initial cluster structure of the the human retinal cell atlas (subfigure 1) is maintained when we augment the data with human T cell atlas (subfigure 2). The overlapping areas possibly represent the T-Cells detected in the retinal cells. This overlap is more evident in the subfigure 3, where we have augmented the data with the complete immune cell census. While we see very little overlap between immune cells and retinal cells, a considerable amount of overlap happens as expected in the T-Cells and the Immune Cells.

2.2 Single Cell Landmarking

In RNA-Seq data, we encounter data with multiple groups of cells. However, the variation between cells in a given group have less variance than the variance across multiple groups. Therefore, the variance can be considered hierarchical in nature. With this in mind, we raise the question, ‘From which point does any given cell vary in a group or a sub-group?’.

An approximation to a ‘canonical representation of a cell’ that represents a given subset of cells in a population can be obtained by averaging the features in the said population. However, such a ‘pseudo-cell’ may be sensitive to noise, and may not be generalizable across multiple repetitive experiments. However, the vector quantization principle we use in the SONG algorithm is a more robust alternative to simple averaging, as we have demonstrated in our previous work, since it is highly tolerant to noise. An intuitive follow-up question on this is ‘how can one use the coding vectors as pseudo-cells?’.

With this in mind we introduce the concept of cellular landmarks. We define a cellular landmark to be a pseudo-cell, that represents the ‘ideal’ cell for a given sub-group. Since SONG grows its set of coding vectors incrementally, we can utilize this growth to get a hierarchy of landmarks. First, we demonstrate how the hierarchical growth of landmarks capture the ever-refining variance of cells when we keep augmenting the data.

setup: For a basic demonstration of landmarks on a 2d visualization we use the Immune Cell Census dataset provided by the HCA project to demonstrate that at first, SONG infers a coarse topology of the cell populations. In Fig 3, the left-most

scatter plot represents such a coarse inference from 70000 cells. Note that the map has low-opacity, showing a low-density of points generally. However, as we include new data, in the second subfigure, we see that the opacity increases in dense regions, and as we proceed, these dense regions show more minute differences, and break off into new clusters.

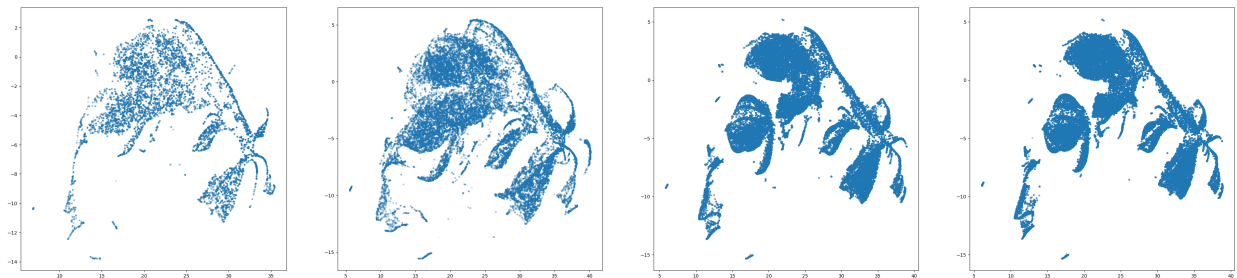


Figure 3. As new cells are added to the mix, the existing clusters are further enforced and made more dense. In the plots, the clusters become more and more opaque as more cells are added. Some clusters become better separated due to the enforcement of differences.

result: We observe in Figure 3 that the initial cluster structure is preserved throughout. However, we do not see the complete separation of minute clusters earlier on. When we present more and more data, we see that the existing clusters get augmented, as evident by the increasing opacity of the clusters, as well as minute clusters getting separated by getting refined.

In summary, this experiment shows that at early stages, when only a small amount of data is presented, SONG infers a relatively small amount of landmarks. And these landmarks are abstractions of the whole populations of cells, where the small number of landmarks show fewer clusters than the visualizations where a large number of landmarks are present. Furthermore, as we increase the amount of presented data, the landmarks spawn new landmarks to capture the variance of increasingly smaller sub-populations. This is apparent in the Growing number of clusters as well as the increased separation between them.

2.3 Landmarking Cytot Data

To demonstrate the concept of cellular landmarks, we use the Wong et al dataset, as we did in our previous work. Our goal in this study is to obtain abstractions of the data in the input-space, and visualize these abstractions (i.e. landmarking), rather than visualizing the whole dataset.

As our first study on landmarking, we use the complete dataset to train a SONG model. However, instead of visualizing the set of inputs using the SONG model, we use the coding vectors of the trained model landmark the dataset. We have considered the SONG generation in 10 different generations, where at early generations we use a smaller spread-factor value and at later generations we use a higher spread factor value. At each generation, we plot the respective locations of the coding vectors on a 2D scatter plot. We consider the size of the landmarks to be proportional to the number of data points encoded by a landmark. We select four generations, 1, 4, 7 and 10, and show them in Figure 4. While these landmarks have different levels of 39 expressed surface markers, to demonstrate how the heterogeneity of the landmarks, we choose one of them (CCR7) and use this surface marker to overlay colors on the scatter plot.

We note that unlike equivalent methods such as PAGA, SONG does not rely on reinitializations to provide this granular landmarking, and does not rely on other clustering algorithms.

However, we note that the strength of SONG in this landmarking lies in the ability to include new data points into an existing map. As we have discussed in our previous work, we note two possible scenarios of increments: 1) Homogeneous Increments where the incoming new data follows the same data distribution as the existing map and 2) Heterogeneous Increments where the incoming data changes the cluster structure of the existing data. We have shown in our previous work that SONG has the ability to handle both these cases in terms of visualizing these increments in context to the existing visualizations. Now, we consider how we landmark such incremental scenarios.

To demonstrate the homogeneous incremental scenario, we randomly sample increasing numbers of cells at 10 generations (generation 1 having the smallest number of cells and generation 10 has all the cells in the dataset). Here, our assumption is that such a random uniform sampling will proportionally sample the complete dataset and will have a homogeneous manifold across multiple increments. Next, we visualize the landmarks inferred in a similar fashion to our previous experiment.

While we note that in Figure 5 we see that the number of landmarks are less than the number we saw in Figure 4. However we see that the structure of landmarks is similar in both cases.

Next, we use a non-random sampling based on the cell ordering, and thus ensure that the increments are not homogeneous. Using this nonrandom sampling, we visualize landmarks when we add new groups of cells into the existing visualizations.

In Figure 6, we show that the landmarking strategy can be employed even in the highly heterogeneous increments.

Overall, we note that in incremental cases, the landmarks are relatively smaller, compared to the case where we present all data at once.

3 Conclusions

In this work, we have used the SONG algorithm for two new usages, 1) Building workflows to integrate multi-source data, and 2) Introducing the concept of Cellular Landmarks. Through our experiments, we have shown that SONG is capable of facilitating contextually incremental visualizations in workflows where data from multiple sources are integrated into the same visualization.

We introduced the concept of cellular landmarks. However, further analysis needs to be conducted to analyse the inferred landmarks.

In our extended work in the Appendix, we have further validated that the purity of the clusters provided in the visualizations by SONG are on par with those of UMAP and t-SNE. We have further demonstrated that SONG far outperforms UMAP and t-SNE in terms of the preservation of cluster topologies when presented with new data.

4 Methods

4.1 Incremental PCA

We have used the incremental PCA available in the Scikit Learn package. In this we have not performed any preprocessing on the human cell atlas data to eliminate genes from different datasets. The reason for this is that all three datasets have the same genes and for different cell-groups, elimination of zero genes would create misalignment of datasets (mismatch of features). Therefore, the only preprocessing we have done is reduced all data from the human cell atlas down to 80 dimensions using the incremental PCA method.

4.2 SONG, UMAP and t-SNE

We used the Python package for SONG available at ¹ and have used the UMAP distribution made available by the authors of the UMAP method. We used the scikit-learn implementation of t-SNE.

Appendix

Quality of visualizations compared to t-SNE and UMAP

We compare SONG's capability to visualize RNA-Seq data while preserving the accurate cluster placements as follows.

setup: In this experiment, we simulated 35 RNA-Seq datasets using splatter which have varying numbers of clusters (2, 5, 10, 20, 40) and varying probabilities of each gene being differentially expressed in a cluster of cells (1, 0.5, 0.1, 0.08, 0.05, 0.02, 0.01). All 35 datasets have 50000 cells and 1000 genes, and we have used PCA to get the first 50 principle components to emulate a typical analysis pipeline.

We conducted two analyses for each dataset, where we calculated 1) The classification accuracy of the visualization by considering the 10 nearest neighbors, and 2) The adjusted mutual information of clusters identified using a k-means clustering on the visualization.

result: We plot the scores in Fig 7. We see that in terms of classification accuracy, SONG has comparable performance overall to both UMAP and t-SNE. However, we see that in terms of cluster separation quality, SONG outperforms both UMAP and t-SNE.

Stability of visualizations under homogeneous increments

We consider the case where a dataset is augmented multiple times with data having a similar distribution to the currently encountered data. In such a case, we wish the new data to be incorporated into the existing clusters, without changing the position of the clusters significantly.

setup: To simulate this scenario, we simulated a dataset of 75000 cells, having 5000 genes using splatter, and having 5 different clusters. We extract the first 50 principle components, and visualize them using t-SNE, UMAP and SONG. As per the observations in², we do not consider the parametric t-SNE due to the poor cluster separation and quality it provides.

In this experiment, rather than presenting data all at once, we present these algorithms with the first n datapoints of the dataset where n is iteratively updated to take the values 15000, 30000, 45000, 60000 and 75000. We hope to simulate the acquisition and availability of data in an incremental manner with this approach. Next, we calculate the average movement of a point from one visualization to the next. This gives us the average point movement and the standard deviation of point

¹<https://github.com/damithsenanayake/SONG.git>

movements *for each increment*. This gives us two metrics to measure the quality of incremental visualizations 1) A qualitative metric on the visualizations where we 'observe' that the clusters are stable and of good quality, and 2) a quantitative metric of the cluster positions, and the trend of stabilization if any is visible.

result: Throughout the increments, quantitatively, we see in Fig ?? that t-SNE shows highly disruptive movement of clusters from one increment to the next. UMAP shows significantly better cluster separation however, we see a similar movement of clusters. SONG on the otherhand, shows a trend of 'growing' clusters in their initial place, and this is quantitatively verified by the low standard deviations and averages of the movements of points in each increment. Further more, we see a stabilization trend in SONG where we see a lower and lower movement as increments progress.

Stability of visualizations with heterogeneous visualizations

Although SONG shows superior preservation of existing clusters when they're proportionally augmented with new data, in reality, data may represent new clusters or classes, which we would like to see separated into a new cluster. However, we would also like to preserve the existing cluster structure to some extent, so that the new classes or clusters can be contextually interpreted.

setup: With this in mind, we simulate a dataset with 5 different groups of cells using splatter, and present each of our algorithms 1, 2, ... up to all 5 groups of cells incrementally. Similar to our previous setup, we further calculate the movement of data points w.r.t. consecutive increments.

result: We show that in Fig 9, SONG still shows the lowest amount of movement except at the third increment where UMAP has a lower movement. However, we further note that that qualitatively, the positions and the shapes of the clusters as new clusters are presented is drastically altering from one increment to the next in t-SNE, and in UMAP, with the exception of the third increment, we see that new clusters at times completely reorient the clusters. However, SONG includes new clusters accordingly without complete rearrangement of the remainder of the existing clusters.

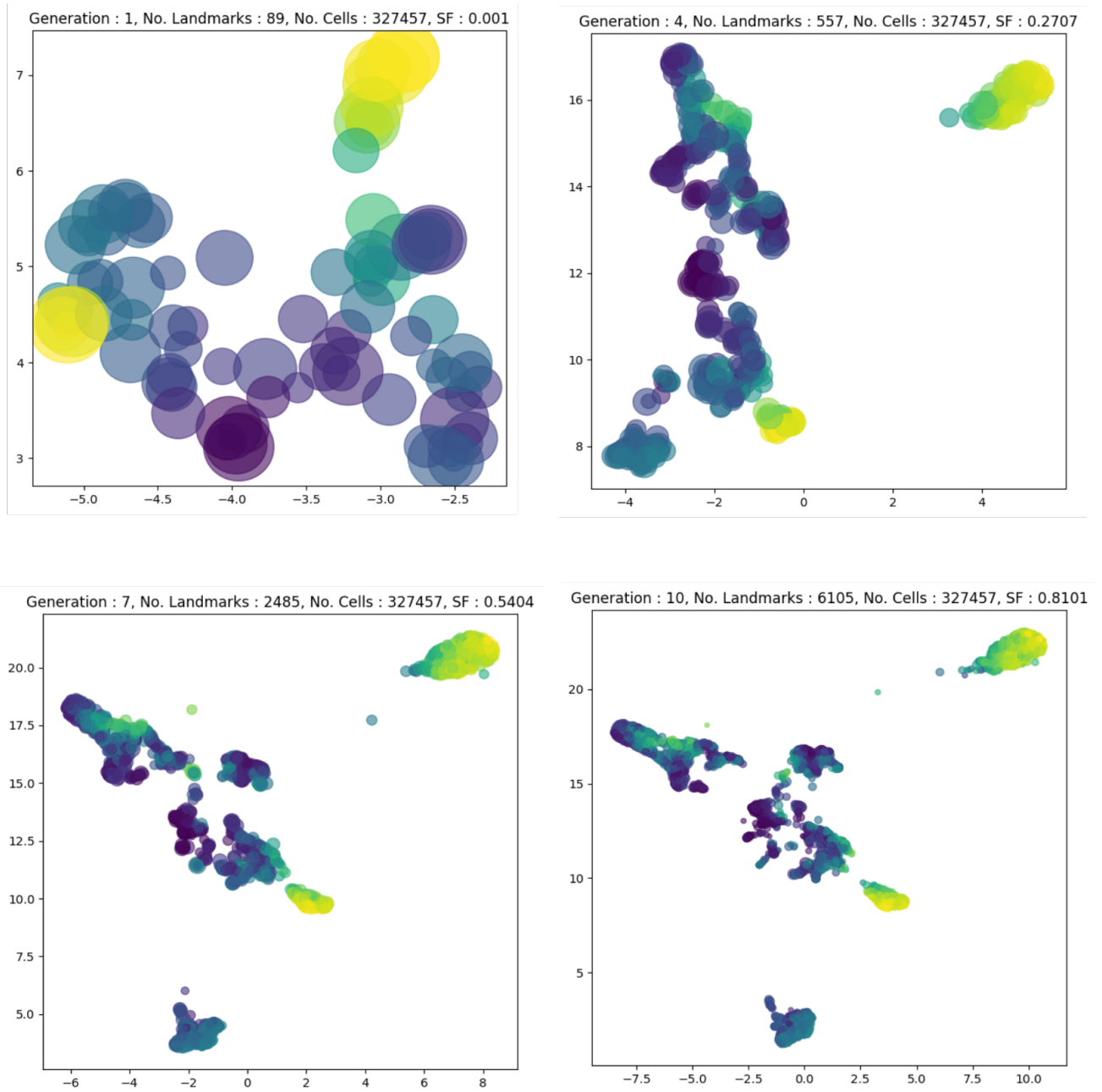


Figure 4. At earlier generations, we see a small number of large landmarks. As the generations proceed, we see that the number of landmarks increases, while each landmarks encodes a smaller and smaller region in the input space.

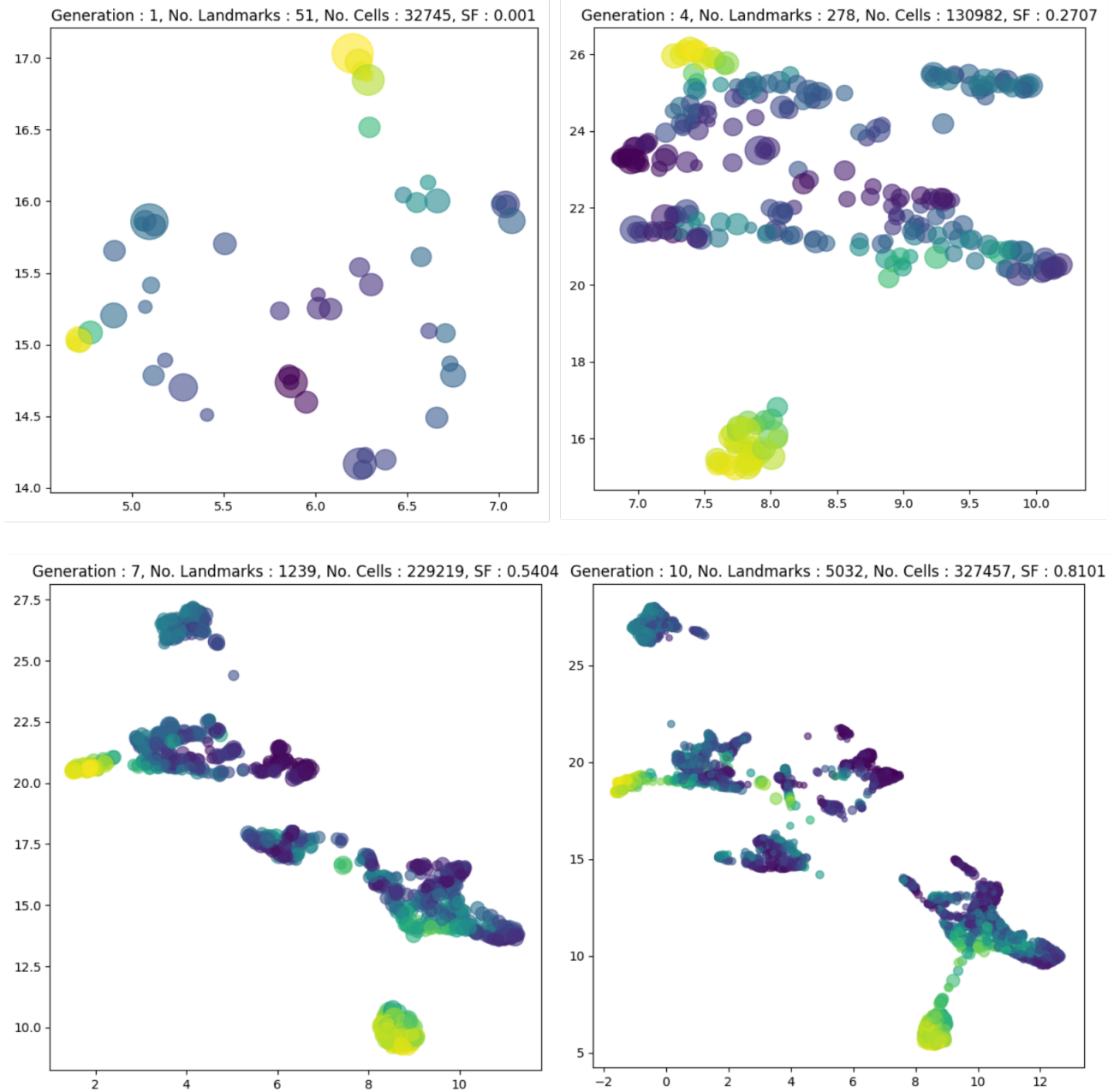


Figure 5. While similar to using the whole dataset, we see a small number of landmarks at early generations, we note that the size of the circles are smaller. The rationale behind this is the small number each landmark encodes.

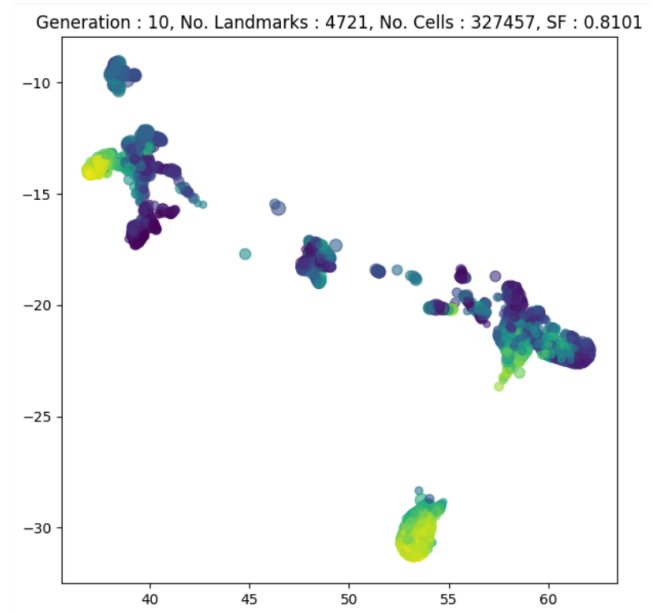
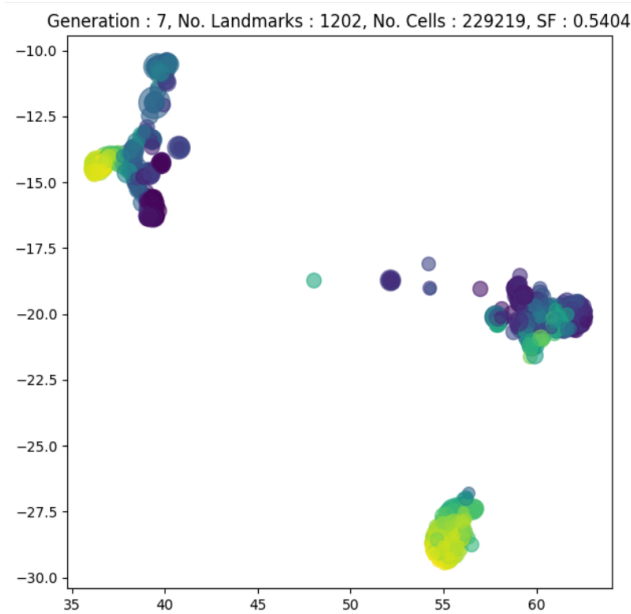
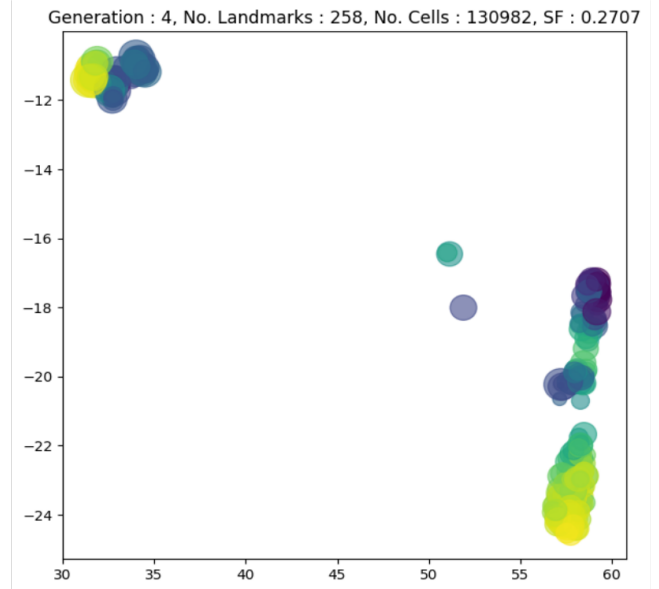
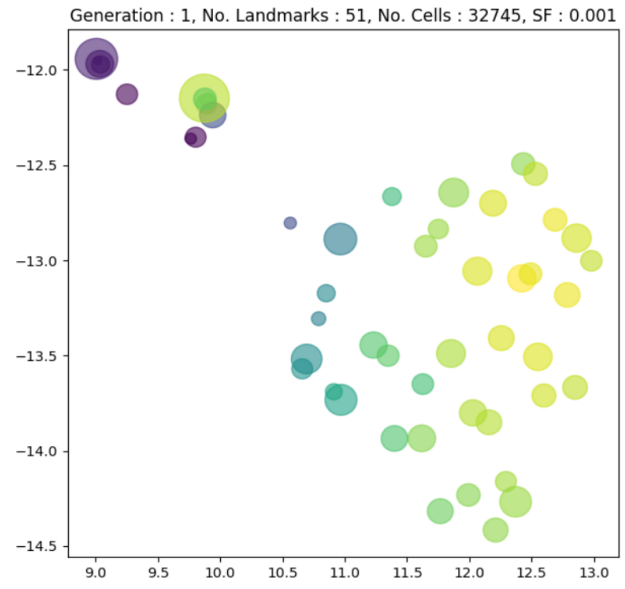


Figure 6. We add new groups into the existing visualizations. New landmarks are created in existing clusters as well as the newly identified groups. Similar to previous cases when increasing generations the landmark size decreases while the number of landmarks increases.

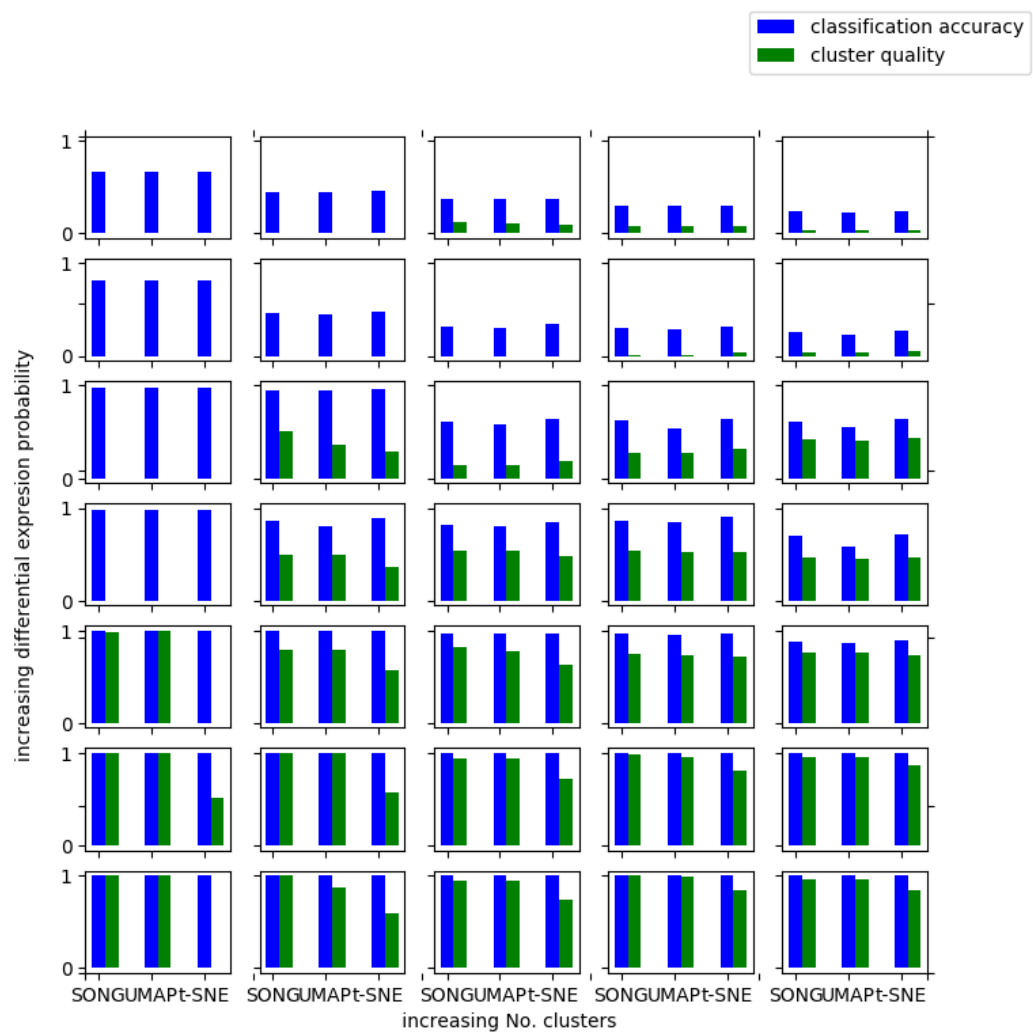


Figure 7. Caption

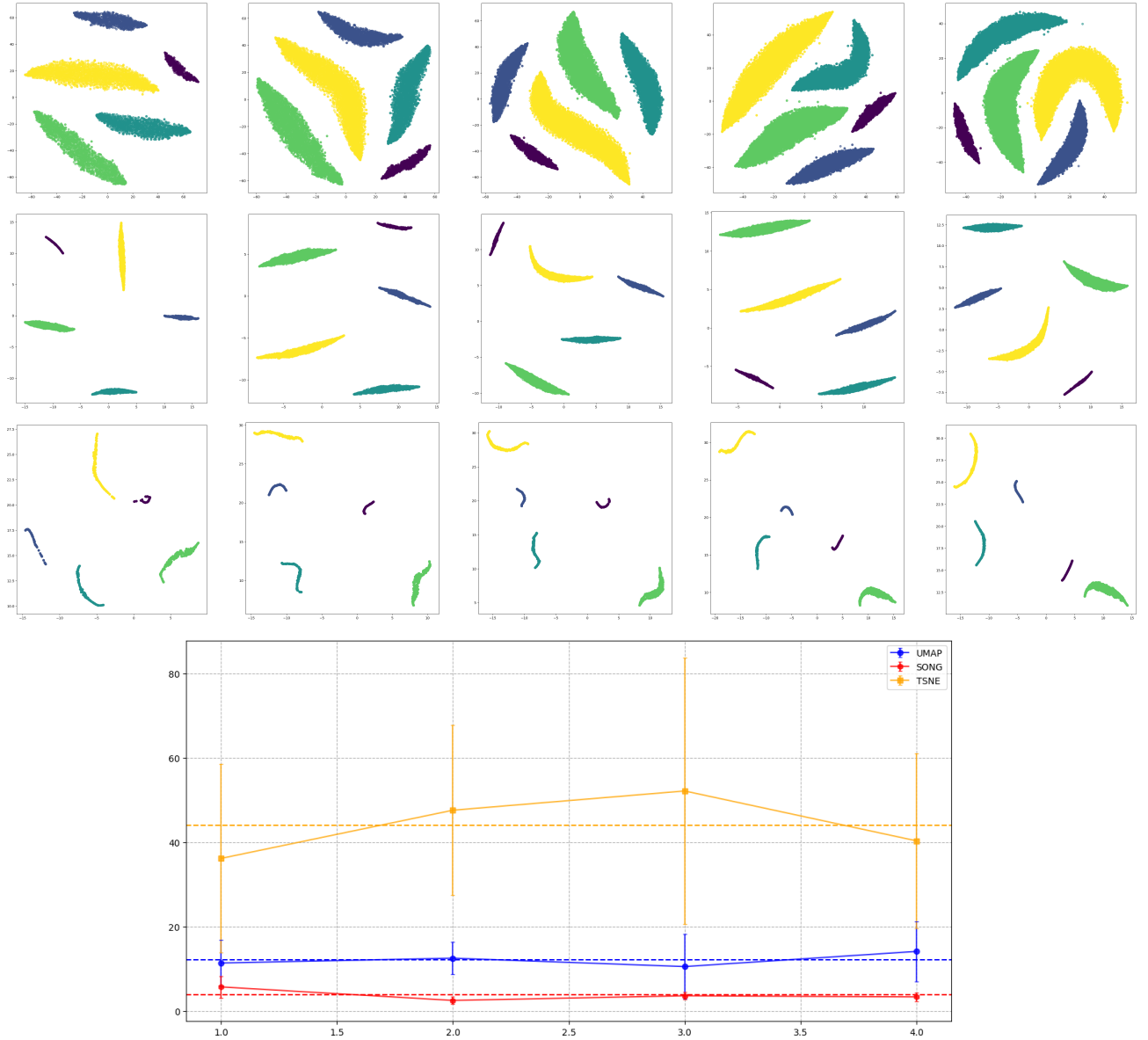


Figure 8. Homogeneous increments with t-SNE, UMAP and SONG. Starting with only 7500 cells, we see that t-SNE and UMAP drastically change their cluster positions from one increment to the next. SONG retains the relative cluster structure throughout. In the plot, the points show the average euclidean displacement of existing points in an increment. The error bars represent the standard deviation. The dashed lines represent the overall average of point movement. We see that SONG has the lowest average movement and lowest standard deviation of the three methods.

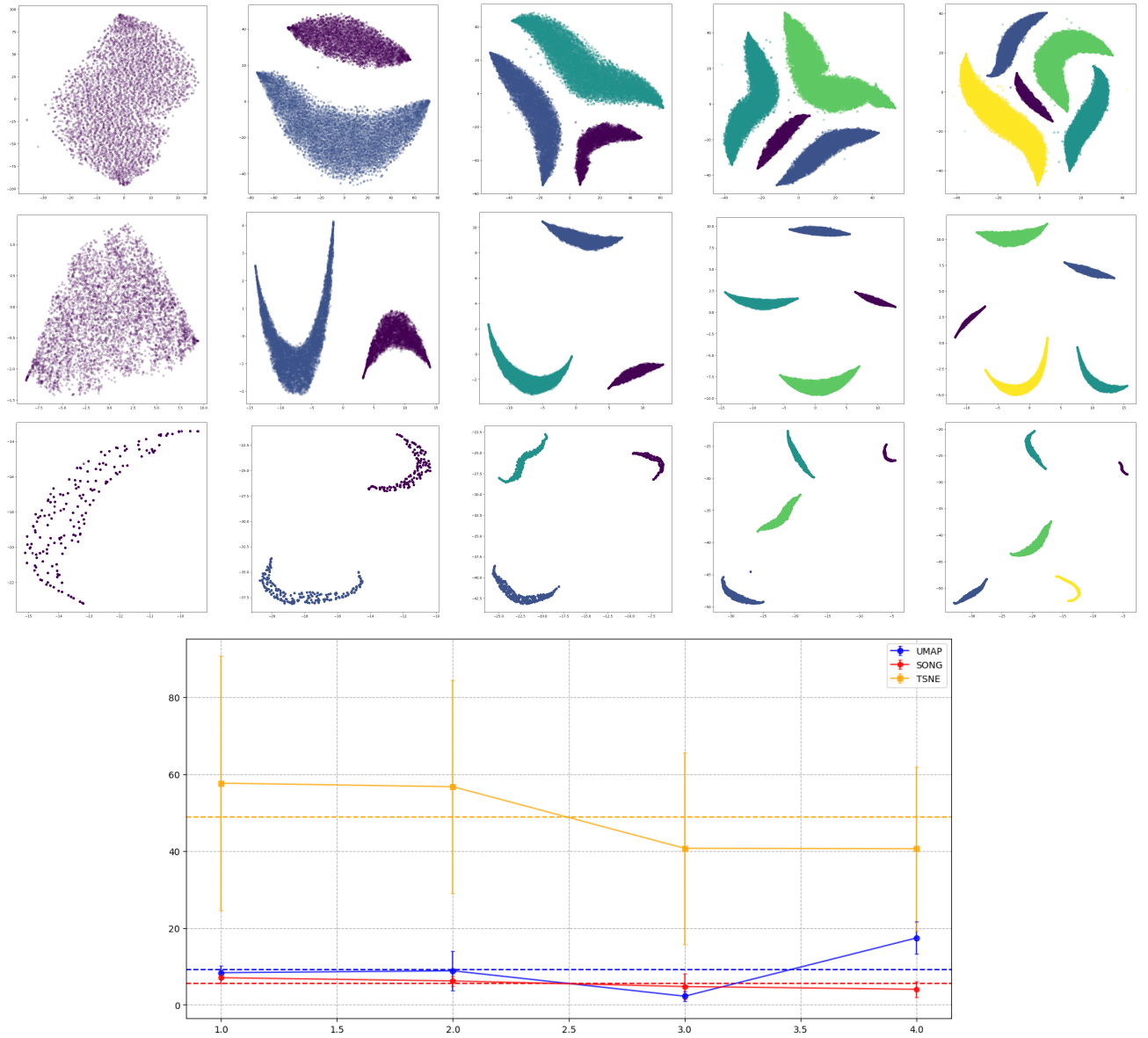


Figure 9. Highly heterogeneous increments of data provided to the three algorithms, t-SNE, UMAP and SONG respectively. The curves show the average displacement of existing points in each increment. The dashed lines show the overall average throughout the increments.

4.2 Summary and Contributions

In this work, we have shown with simulated data that SONG is superior at the preservation of global structures in the visualizations to UMAP and t-SNE. Furthermore, we have introduced the concept of cellular landmarks.

We believe that the most important contribution of this work is in cellular landmarking. When regarding the philosophical underpinnings of the vector quantization, we hope to provide these cellular landmarks as approximations of ‘ground-truth abstractions’. These ground-truth abstractions can be considered as the archetype of a population of cells.

Taking a more concrete example, if we consider a group of cells such as the Dendritic cells in the human blood, the coarsest abstraction of all of these will be considered the Dendritic Landmark, which will be represented as a large cluster in a visualization. However, if we consider the subtypes of dendritic cells, we encounter the conventional dendritic cells (cDCs) and the plasmacytoid dendritic cells (pDCs). Another level deeper, we encounter that the cDCs have two types (Type 1 and 2). At these sub-cluster levels, if we represent a whole sub-cluster of cDCs to be represented by one prototype cell, then we can consider it to be the cDC landmark. As a byproduct of this, we get a hierarchy of land-marks. However, such a hierarchy must be observed at multi-omic levels (i.e. in addition to the Cytof landmarking we have done here, we must also conduct RNA-Seq landmarking etc.). We propose this expansion of studies as immediate future directions of research.

Chapter 5

What would applying feature extraction capabilities of Neural Networks reveal of the underlying genetic dynamics in RNA-Seq Experiments?

Overview

In this chapter, we propose a framework for extracting motifs of genes that regulate the heterogeneity of cells using RNA-Seq data. We use sparse autoencoders to obtain a reduced representation that captured highly non-linear, complex structures of the data. Furthermore, with the use of sparse autoencoders, we show that we can obtain feature subsets that have a minimal set of meaningful genes that contribute to the overall heterogeneity of the original data. We demonstrate the power of this approach using RNA-Seq datasets of Haematopoietic Stem Cells, and PBMCs. We further show that we can obtain a minimal set of useful or meaningful genes that contribute to different levels of heterogeneity in a hierarchically structured dataset.

5.1 Background

As previously discussed, the dimensionality of RNA-Seq experiments present a challenge in structural analysis of data gathered. We have discussed how to gain insights into the structure of the data through topological analysis in the high-dimensional input space. However, few formal techniques exist with techniques capable of explaining the underlying dynamics of the structure.

Existing methods, often rely on the known *truths* of the cellular dynamics. They often rely on the so-called marker-genes which are differentially expressed in certain populations with statistical significance. On the other-hand, few methods handle the task as a feature extraction problem without relying on many assumptions to get a reduced input space.

In literature, high-dimensional feature extraction is often treated as a problem where we consider the statistical correlation between pairs of variables. However, for a large number of variables, typically in the 10s of thousands, this analysis is problematic. Furthermore, such methods are non-parametric, whereas any analysis on a dataset is only valid on that particular dataset, and no model is retained that can be generalized into other datasets.

Eraslan *et al* in their recent work [83] describe the advantages of using deep neural networks for modeling RNA Seq data. Among them, we focus on the *ability to model complex dependencies* and the *interpretability of feature components* as founding motivation of our next chapter.

Although generative Neural Networks have been used in as early as 2017 [84] in an attempt to model and denoise RNA-Seq data, the feature extraction aspect of using

such neural networks has not progressed far. In this chapter, we demonstrate the power of using simple feature extraction techniques to identify underlying genetic dynamics of complex RNA-Seq datasets.

RESEARCH

Unsupervised Deep Neural Networks extract the ‘Genetic Penstrokes’ of heterogeneous cellular populations with minimal assumptions and without a priori granularization

Damith Senanayake^{*†} and Saman K Halgamuge

^{*}Correspondence:

senanayaked@student.unimelb.edu.au

Department of Mechanical Engineering, University of Melbourne, Parkville, Australia, AU

Full list of author information is available at the end of the article

[†]Equal contributor

Abstract

The analysis of high-dimensional data obtained in single-cell RNA-Seq experiments show that the differential expression of genes (DEG) in each cell contributes to the overall functional and phenotypical heterogeneity between individual cells. However, extracting the frequent motifs of genes that give rise to this heterogeneity by extracting sets of local features is a prominent problem in single-cell data analysis. Previous work relies on a global feature approximation, i.e. clustering the data into discrete partitions to extract such motifs. However, in practice such clustering may be unreliable or even incorrect, as we often encounter high levels of experimental noise muddling the cluster boundaries, or encounter smooth trajectories without discrete clusters in cell lineages. In this work, we show that using unsupervised neural networks, we can extract such motifs, which we name ‘genetic penstrokes’ directly from the input space, without prior clustering. We represent each of these motifs as an ordering of genes by their contribution to each motif. Furthermore, these motifs identify both positive and negative contributions from genes. We show through our experiments that these penstrokes can be used to reconstruct approximations of the original data, capturing the heterogeneity of the original data. Furthermore, we also show that these penstrokes provide us a framework with which we can simulate manipulations of the heterogeneity.

Keywords: Deep Learning; Genetic Penstrokes

1 Introduction

RNA-Seq experiments provide highly dimensional data which are often affected by various undesired variability such as batch-effects[1], transcriptomic and technical noise[2]. Furthermore, the identification of different cell-types in an RNA-Seq dataset often involves the manual annotation of cluster specific marker-genes [3]. Even in the more general cases such as edgeR[4] or DESeq[5] where the differential expression of individual genes are considered across multiple conditions (e.g. treated vs untreated), they provide little information as per the correlation or competition between the expression levels of different genes across the different populations. Furthermore, in cases where population boundaries are not clear-cut (such as a smooth trajectory of cell lineages starting from a pluripotent progenitor to a functional cell e.g. [6]), the reliance on clear separation of clusters or populations may be an inevitable pitfall.

Especially in the recent studies regarding the differential expression of genes across multiple populations in domains such as haematopoiesis, we encounter populations of which the true functional or cellular identities are not experimentally known (i.e. no ground-truth)[7]. In such cases the contemporary methods often rely on the empirical approximations of ground-truth. For example, platforms such as Seurat[8] or ScanPY [9] first rely on explicit graph clustering methods to differentiate between multiple cell 'types'. This clustering is followed by a statistical test that determines which of the genes are highly differentially expressed across populations. However, it is note-worthy that these methods are non-parametric, i.e. they do not retain any generalizable set of approximations across multiple experiments which can be then verified by augmenting with more data.

In this work, we present an approach where we attempt to extract feature-subsets, which we dub the 'pen-strokes' of genes. These 'pen-strokes' are combined to make different 'characters', and two significantly different characters can be expected to be analogous to two different types of cells. In each pen-stroke, we have an ordering of the genes in the RNA-Seq dataset considered, which represents the importance of each gene in the pen-stroke. If a combination of such pen-strokes correspond to one or more populations of cells in the original data, then we get both an ontology of genes, as well as a hierarchy of genes that contribute to the differentiation observed across the dataset as we describe in Section 2.

2 Results

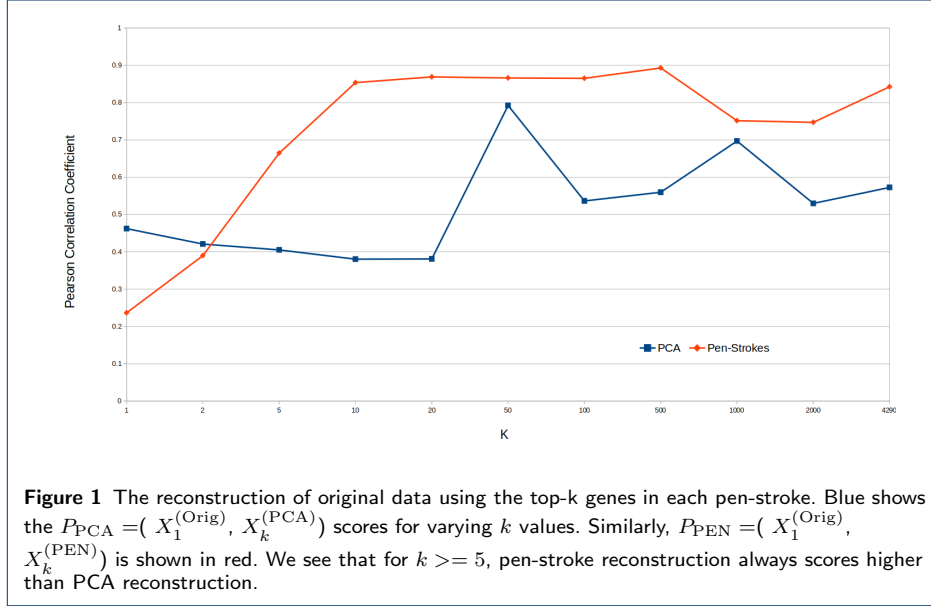
2.1 Reconstruction of Original Variance Using pen-strokes

In this section, we investigate the possibility of reconstructing a dataset using.

Setup: We compare the quality of reconstructing a dataset using a reduced number of genes as we describe in Section 4.3. As a baseline or a benchmark method, we use the principal component analysis of the dataset. We extract the first 15 principal components each of which is a linear combination of the dimensions (genes), where we have a total of 4290 genes. To obtain a minimal set of genes which capture the overall variance of the data, for each component we consider the genes corresponding to the k largest loading coefficients, where $k = \{10, 20, 50, 4290\}$. We then disregard all genes that are not in any of the aforementioned top- k gene sets, and recalculate the reduced representation considering only the remaining genes. This is followed by the nonlinear dimensionality reduction and comparison as described in Section 4.3. Let us call the one-dimensional UMAP representation we get for the reduced PCA reconstruction as $X_k^{(PCA)}$.

We conduct a similar reduction using the Pen-Stroke Method with the use of Sparse Autoencoders. We extract 15 latent variables (pen-strokes) that best reconstructs the high-dimensional input, and then we extract the largest k weights for each of these 15 latent variables (which are neurons in the sparse autoencoder). Using these pen-strokes, we perform the sort-and-curtail filtering which we describe in Section 4.2.1, which provides us a reduced set of genes to be used in the reconstruction. Using this reduced set of weights, we then recalculate the hidden activations corresponding to the input data, which provides us a new 15 dimensional representation. Similar to what we did with the PCA reductions, next we conduct the one-sense analysis, by obtaining a one-dimensional nonlinear reduction using UMAP. Let us call this one-dimensional representation $X_k^{(PEN)}$.

In summary, in our one-sense analysis we calculate the Pearson's Correlation Coefficient between the pairs $P_{PCA} = (X_1^{(Orig)}, X_k^{(PCA)})$ and $P_{PEN} = (X_1^{(Orig)}, X_k^{(PEN)})$.



Result: We observe in Figure 1 that the reconstruction quality stabilizes around 85% for pen-strokes and for PCA it is around 55%. However, we note that this increase seen in pen-strokes does not show a stable trend, as the relationship between k and the reconstruction scores are nonlinear.

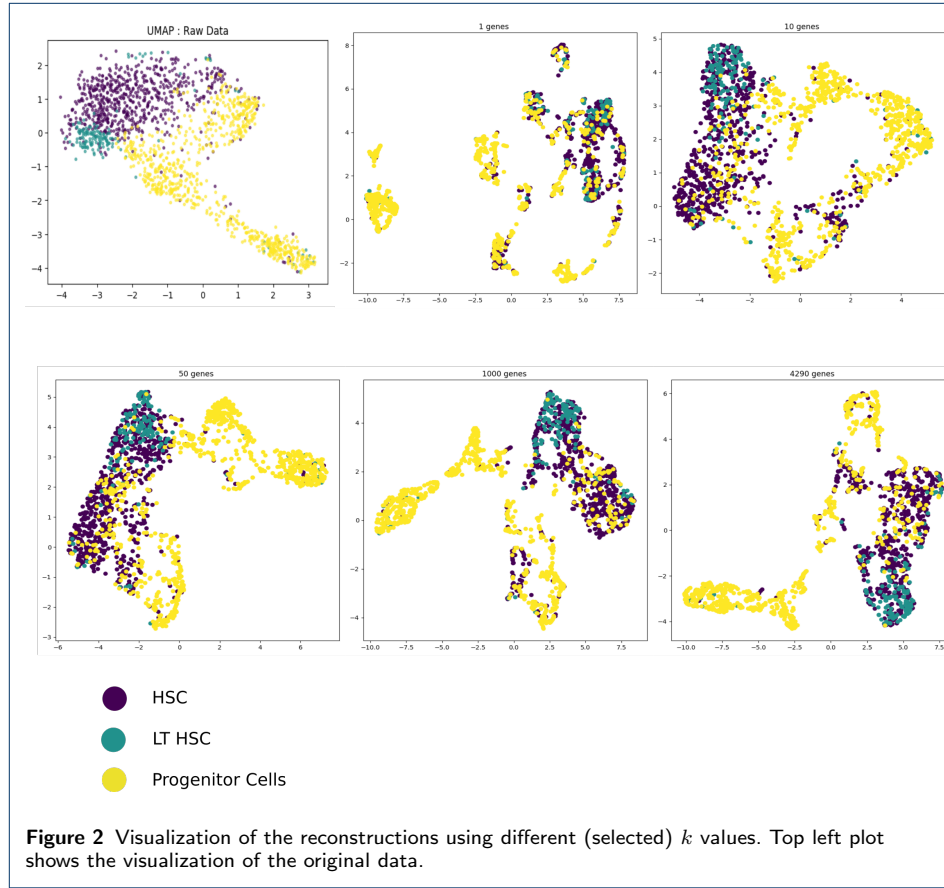
This reconstruction quality can be further observed to be increasing in terms of visualizing the reconstruction. We use UMAP to visualize the reduced representations obtained by the pen-stroke reconstructions. Note that for different k values, we obtain a 15-dimensional representation using the pen-stroke reconstruction. Furthermore, we attribute the increased performance of pen-strokes to its ability to capture nonlinear relationships between genes.

Figure 2 shows the reconstruction using different k values. We note that as k increases, the quality of clusters improve, while the general topology of the original data is maintained. One important observation we make in Figure 2 is that as k increases, the discrete clusters become well separated.

Having displayed the ability of the genetic pen-strokes to capture nonlinear and optimal relationships between genes, we next investigate how the pen-strokes capture vast heterogeneity, unlike the smooth trajectory observed in the Nesterowa et al dataset.

2.2 Pen-strokes capturing the heterogeneity in PBMC data

For this task, we use two datasets; the 3k and 33k PBMC datasets available at 10x genomics. We used the PBMC-33k dataset to investigate how the pen-strokes capture the variance and heterogeneity across highly heterogeneous populations. Next, we use the PBMC-3K dataset to attempt and identify the optimal feature sub-sets that accentuate the heterogeneity in different cell populations.

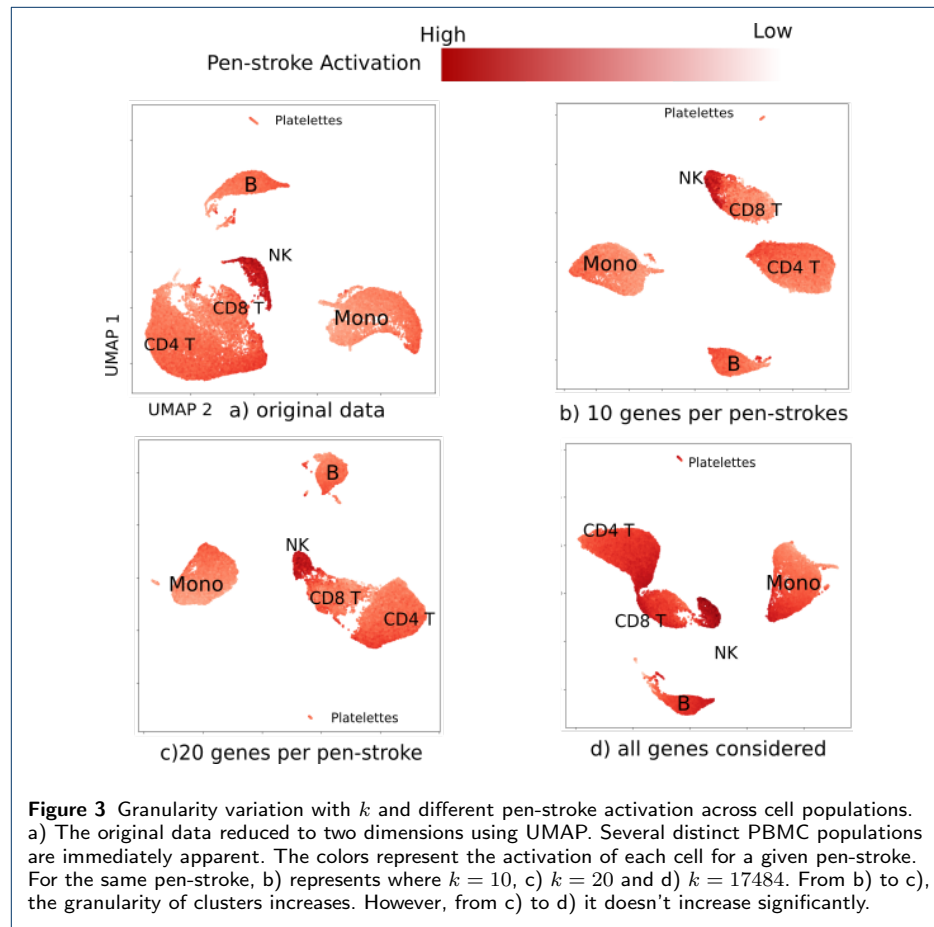


2.2.1 Granularity of Heterogeneity with Pen-strokes

In our previous experiment, we saw that when the number of top genes (k) used in the reconstruction of the original data varies, the variation of the original data captured by the reconstruction varies. Since we used a dataset with a smooth trajectory, and saw distinct clusters in the reconstruction, it is possible that the variation captured in the pen-strokes may explain discrete populations. Therefore, we next investigate the behavior of the captured heterogeneity, i.e. the capturing of distinct populations using a reduced set of genes, and the gene weightings obtained by extracting 20 pen-strokes.

Setup: Using the 33k PBMC dataset [10], we extract 20 pen-strokes. Similar to our analysis in Section 2.1, we extract three different k s of genes, namely, 10, 20 and 17484. Note that 17484 is the number of total genes remaining after elimination of genes in preprocessing. Next, we use UMAP plots to investigate how each distinct population of cells are reconstructed for each of the aforementioned k s.

Results: In Figure 3 we have used a single random pen-stroke to color the visualization in all four subplots. The visualization is obtained by using 15 pen-strokes as input to UMAP, with different k values. We see that the granularity of clusters increases with k (from Figure 3 b) to c)) and yet does not show much improvement after $k = 20$ (see d)). We see that by increasing k gradually, we can obtain an optimally granular representation. However, further analysis needs to be conducted as to at what k the granularity becomes stable. We note that for



this particular dataset, $k = 20$ provides a reconstruction that captures the cluster structure of the original data.

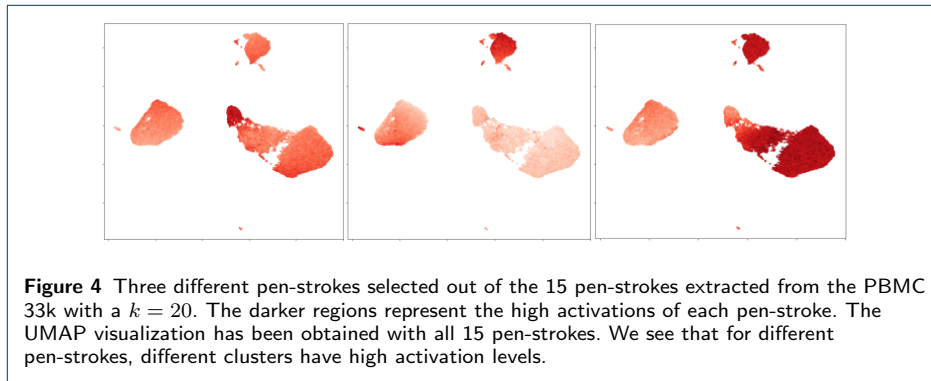
Furthermore, in Figure 4, we use three different pen-strokes (all having $k = 20$) to color the UMAP plot obtained using all 15 pen-strokes. We observe that different pen-strokes are activated highly in different groups of cells.

We have shown that these pen-strokes have different regions of activations. Next, we demonstrate how the reconstruction of original data using extracted pen-strokes affect the phenotypical properties of cells.

2.3 Validation of pen-strokes which contribute to phenotypical heterogeneity

In this experiment, we use two measurements of a single-cell population to determine the correlation between penstroke activations and phenotypical expressions. For this, we use the 8k CBMC dataset provided by Cite-Seq, where we have a dataset of 8k CBMC cells and 17000 genes after preprocessing. Additional to the RNA-Seq data, this experiment has 12 different surface marker expression levels recorded for each cell as well. Therefore, we can use the surface marker expressions as empirical ground-truths providing phenotypical heterogeneity.

Setup: Similar to our previous experiments, extracted 15 pen-strokes from the RNA-Seq data, and used the top - k genes ($k = 20$) to provide reconstructions of the



dataset. Next, we use UMAP to obtain a 2 dimensional representation of the data thus reconstructed using pen-strokes. We next overlay the expression of different markers onto the UMAP plots thus obtained, and compare it with the pen-stroke expression maps to determine correlations.

Results:

We see in Figure 5, the surface markers are expressed heavily in tight regions in the visualization of the reconstruction. This implies that combinations of the pen-strokes may capture the underlying genetic factors that drive the surface-marker expressions.

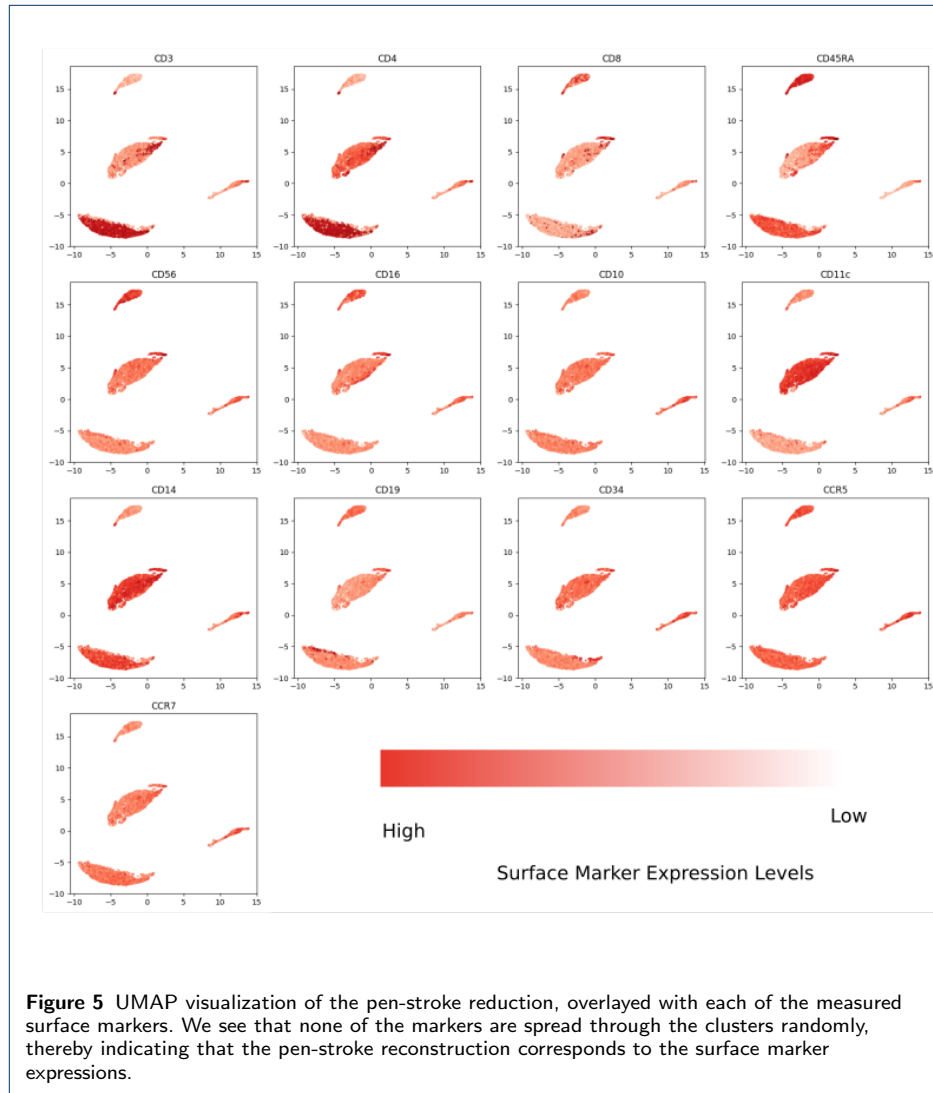
The important analysis is the identification of which combinations of pen-strokes contribute to a given marker. As per Figure 6, we consider two pen-strokes correspond to the CD 11C and CD 56 markers. We see that for CD 11C, we see only a weak correlation as high activations are present in clusters where CD 11C is not high as well. On the other hand, the CD 56 expression is directly correlated to the pen-stroke we have plotted underneath it. Since in reality, nonlinear combinations of pen-strokes give rise to the cluster heterogeneity, further studies need to be conducted to find which combinations of pen-strokes give rise to which of the surface markers.

2.3.1 Accentuating the Heterogeneity with Pen-strokes

The selective curtailing of pen-strokes provides us with increased control over the granularity of the representations of the input data, and it indicates that pen-strokes are capable of capturing some form of canonical representation of the hierarchy of clusters and groups. We hypothesize that it is the manipulation of these canonical representations that provide us with the flexibility to manipulate the granularity of the representation. In such a case, it should also allow us to emphasize and accentuate the subtle differences of different cell groups. This is especially useful in cases where clusters of highly different cells are erroneously mixed due to experimental limitations.

To validate this hypothesis, and to see how we can accentuate such subtle differences, we conduct the following experiment.

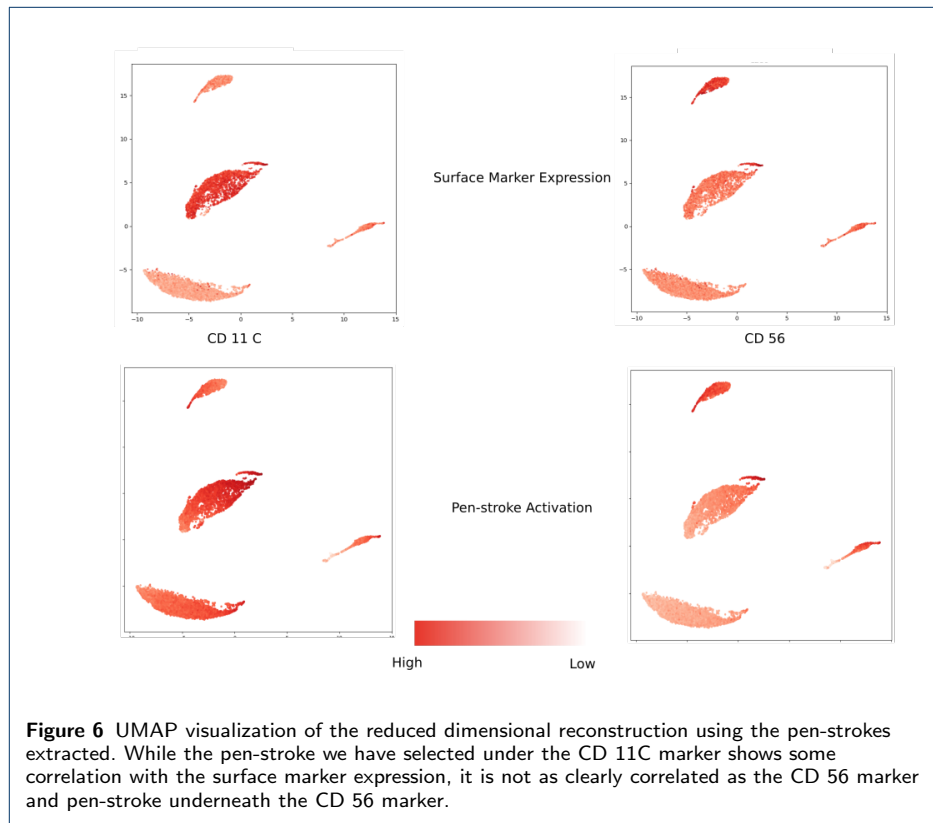
Setup: We use the PBMC 3k dataset[11], where we see only a slight separation between the CD4 T Memory / Naive cells. Since our goal is to identify what governs such slight heterogeneity, we wish to accentuate this difference by identifying the



optimal set of features to pull apart the two clusters. Therefore, we first eliminate all other types of cells and isolate only the CD4 T cells. Next, we preprocess to remove noisy and/or low-expression genes to remove technical noise. With this, we have a dataset that has 14766 genes. We extract 15 pen-strokes, and consider five different k values where $k = \{10, 20, 100, 150, 14766\}$.

Results: In Figure 7 we see that as we increase the k , we see better separation between the two populations for the selected pen-stroke. Upon looking up the genes that correspond to the penstroke, we see the following set of genes regulated respectively : S100A4 $\uparrow$, JUN $\uparrow$, MT-CO2 $\downarrow$, CD8B $\downarrow$, ANXA1 $\uparrow$, IER2 $\uparrow$, ZFP36 $\uparrow$, KLF6 $\uparrow$, DUSP1 $\uparrow$, FOS $\uparrow$, MT-ATP6 $\downarrow$, CD7 $\downarrow$, S100A11 $\uparrow$, IL32 $\uparrow$,..., where $\uparrow$ stands for up-regulated genes and $\downarrow$ stands for down-regulated genes. These genes, are ordered in descending order of their contribution to this particular pen-stroke. We note here that the gene S100A4 is a known marker gene for CD4 Memory T Cells.

While this identification of pen-strokes that accentuate the heterogeneity among populations indicates that the pen-strokes can provide insights into the functional



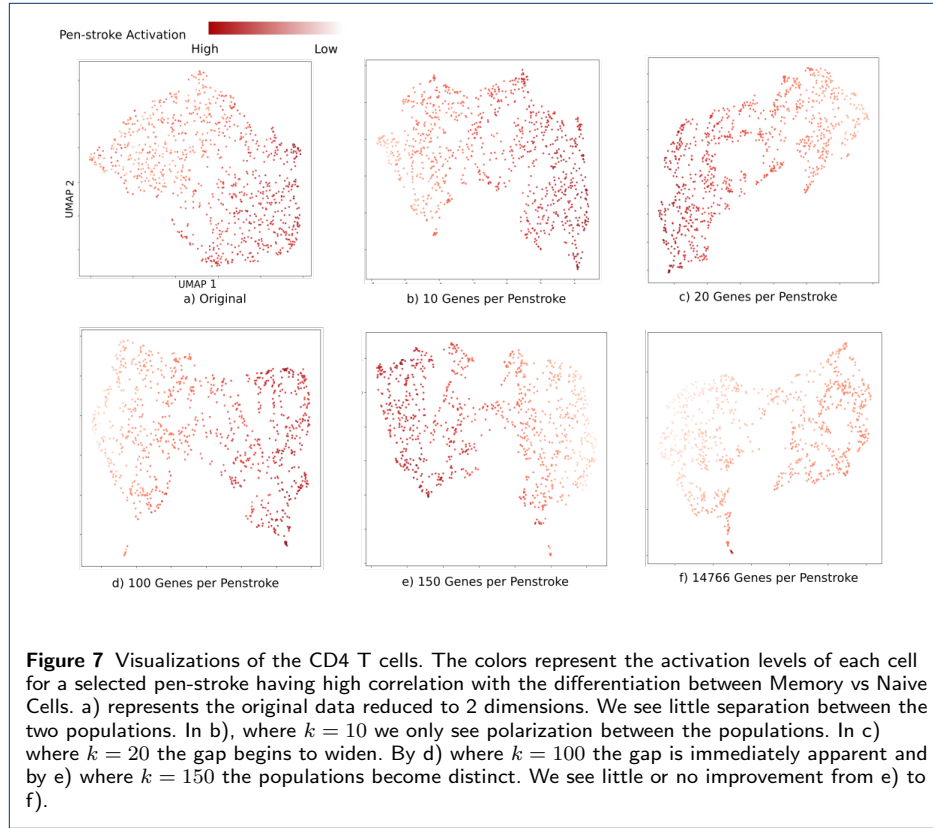
heterogeneity between groups of cells. However, so far, we have no empirical ground-truth to validate this hypothesis.

3 Conclusions

Through our manuscript, we have introduced a new concept for identifying genetic features contributing to the variation of single-cell datasets called “Genetic Pen-strokes”. We have experimentally shown that pen-strokes are better at extracting feature sets that contribute to cellular heterogeneity better than linear feature extraction methods such as PCA. Furthermore, we have qualitatively shown that pen-strokes are capable of capturing varying degrees of heterogeneity in cellular datasets, and by doing so, are capable of identifying and accentuating subtle heterogeneities that may not be clear due to noise in the data.

We have further shown that genetic pen-strokes are capable of capturing functional or phenotypical heterogeneity, as exemplified by capturing of patterns of surface marker expressions independently. We observed that while individual pen-strokes may not be directly interpretable, hierarchically and nonlinearly combining the pen-strokes capture the patterns of functional heterogeneity with considerable high quality.

One of the main-strengths of the pen-stroke method is that in order to identify gene-ontologies or marker genes that contribute to the cellular heterogeneity, it does not rely on manually clustering the cells. Instead, it is reasonable to claim that pen-strokes infer fuzzy clusters in the input space. This allows the pen-stroke method



to be more robust to noise, as we have seen by its ability to identify and emphasize the difference between Memory and Naive CD4 T cells.

4 Methods

4.1 Penstroke Definition and Implementation

4.1.1 Defining a Penstroke

We take inspiration from the work of Ng [12] in extracting meaningful feature subsets (motifs) using neural networks without relying on spatial proximity of features as done by convolutional neural networks[13]. First, let us define a motif of features which we call a penstroke as follows.

Let $\mathcal{P} = \{P_1, P_m\}$ be a set of m penstrokes. Each penstroke P_i where $i = \{1, \dots, m\}$ can be defined as:

$$P_i = \eta(\lambda_{i0} + \sum_{k=1}^{n_g} \lambda_{ik} g_k) \quad (1)$$

where λ_{ij} is a coefficient corresponding to the i^{th} penstroke and j^{th} gene, with the exception of λ_{i0} which is a bias term. g_k is the expression vector for the k^{th} gene, i.e. the expression count for the gene g_k for all cells. η is a nonlinear transformation conducted on the linear combination of genes we obtain by multiplying the

expression vectors by the λ coefficients. In our implementation, we have used the sigmoid transformation such that:

$$\eta(x) = \frac{1}{(1 + \exp(-x))}. \quad (2)$$

To obtain an ordering of genes that have the highest influence on a penstroke, we order the genes in the descending order of the absolute value of the associated λ s. This is to say that the top- k genes in a penstroke P_i can be written as:

$$G_k = \{g_{j_1}, \dots, g_{j_k}\} \quad (3)$$

where $|\lambda_{j_1}| > |\lambda_{j_2}| > \dots > |\lambda_{j_k}|$.

4.1.2 Sparse Autoencoder for Penstroke Extraction

In our implementation, we use a sparse autoencoder with one hidden layer to extract the λ values. We define the sparse autoencoder as below.

Let X be a gene barcode matrix (where the rows represent individual cells and the columns represent the genes, and each element has the count of the gene expressed for the given cell). We define a weight matrix $\mathbf{F}_{g \times p}$ where g is the number of genes and p is the number of penstrokes. We further define a bias vector $B_{1 \times p}$. This allows us to implement a neural network that has g input neurons and p output neurons. The activation of the output layer H is defined as:

$$H = \eta(X\mathbf{F} + B) \quad (4)$$

We add another layer for the reconstruction of the original data after the layer with p neurons. This provides us an autoencoder with g, p , and g neurons in each of the input, hidden and output layers respectively.

The reconstruction of the original data is defined as:

$$X' = \eta(H\mathbf{F}^T + C) \quad (5)$$

where C is a bias vector for the output layer.

We define the loss to be minimized as:

$$\mathcal{L} = \frac{\|X - X'\|^2}{2} + \alpha KLD(H||\rho) + \beta L_2(\mathbf{F}) \quad (6)$$

where $KLD(H||\rho)$ is the sparsity loss for the hidden activations and $L_2(\mathbf{F})$ is a weight decay regularization term. α and β are hyperparameters governing the influence of each of these regularization terms.

Finally, we now relate the implementation of our sparse autoencoder to our definition of penstrokes. Here, we consider each of the hidden neurons to correspond to a penstroke. The incoming weights and the bias of the hidden neurons are therefore approximations for λ values.

The bias vector B represents $\lambda_{10}, \dots, \lambda_{p0}$, and each of the row i of $\mathbf{F}$ represents the λ_{ik} values of a penstroke.

4.2 Reconstruction Using Generative Models

While there exists recent work that attempted to reconstruct RNA-Seq data using a small number of nonlinear latent variables with the use of generative neural networks [14, 15], we go a step further by attempting to extract meaningful feature subsets, each with a minimal set of genes. In our work, we use sparse auto-encoders as the generative model. However, to extract the most influential set of genes, we use a 'sort-and-curtail' operation to the weights.

We define the generative sparse auto-encoder with one hidden layer. However, it is note-worthy that for more complex hierarchical decomposition one may use more hidden layers. In order to make the latent variables represent interpretable features, we enforce a sparsity constraint on the autoencoders [12].

4.2.1 Feature Sorting and Curtailing

We extract the weight matrix of the first layer of the sparse encoder $\mathbf{F}$, and for each weight vector corresponding to a hidden neuron, we sort the weights in descending order of their absolute values. Since each weight corresponds to a gene in the input, this provides us an ordering of the importance of each gene's contribution to a hidden neuron.

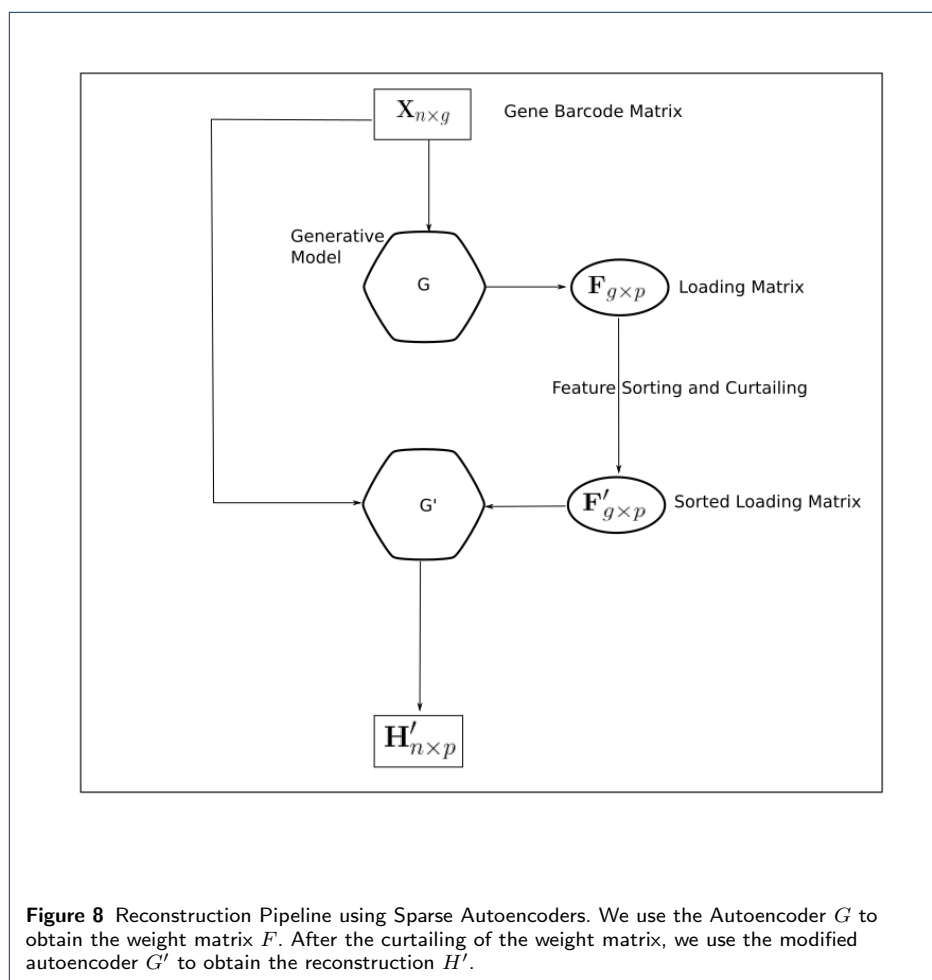
To reduce the number of genes contributing to the reconstruction of the original data, we disregard the genes that have low absolute values for each hidden neuron. We do this curtailing of weights by setting these weights to zero, i.e., after sorting each weight vector, we set all weights after the k -th position (where k is chosen by the analyst) to zero.

Next, we use the curtailed weight matrix to obtain the hidden activations of the auto-encoder to reduce the original data. The overall flow of this reconstruction is summarized in Figure 8.

4.3 Evaluating the Reconstruction by a Reduced Feature Set

Once we obtain a reduced representation using the pen-strokes or the PCA benchmark, we next compare the overall variance of this reconstruction against the original data. In order to do this, we employ an analysis extended from the one-sense analysis, which we describe below.

Let the original dataset be X_o and let a reduced features-set reconstruction by using a method m of X_o be X_m . Typically, the dimensionality of X_m is far smaller than the dimensionality of X_o . Next, we use a non-linear dimensionality reduction technique (in this case we use UMAP) to obtain a one-dimensional representation of each of X_o and X_m , which we call X_1^o and X_1^m . Next, we conduct two analyses, 1) we calculate the pearson correlation coefficient between X_1^o and X_1^m and 2) we construct the scatter plots between X_1^o vs X_1^m . While a perfect reconstruction would have a perfect $X_1^o = X_1^m$ structure (providing a correlation of 1.00), we note that since X_m has a lower dimensionality than X_o , it is acceptable to have a slight deviation from it. Instead, in the scatter plots, what we would expect to see is tight patches of similar cells (data points) and a linear trend with some dispersion. However, we would like a method m to have as high a correlation score as possible.



References

1. Uri Shaham, Kelly P Stanton, Jun Zhao, Huamin Li, Khadir Raddassi, Ruth Montgomery, and Yuval Kluger. Removal of batch effects using distribution-matching residual networks. *Bioinformatics*, 33(16):2539–2546, 2017.
2. Catalina A Vallejos, Davide Risso, Antonio Scialdone, Sandrine Dudoit, and John C Marioni. Normalizing single-cell rna sequencing data: challenges and opportunities. *Nature methods*, 14(6):565, 2017.
3. Tamim Abdelaal, Lieke Michielsen, Davy Cats, Dylan Hoogduin, Hailiang Mei, Marcel JT Reinders, and Ahmed Mahfouz. A comparison of automatic cell identification methods for single-cell rna-sequencing data. *bioRxiv*, page 644435, 2019.
4. Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010.
5. Simon Anders and Wolfgang Huber. Differential expression of rna-seq data at the gene level—the deseq package. *Heidelberg, Germany: European Molecular Biology Laboratory (EMBL)*, 2012.
6. Sonia Nestorowa, Fiona K Hamey, Blanca Pijuan Sala, Evangelia Diamanti, Mairi Shepherd, Elisa Laurenti, Nicola K Wilson, David G Kent, and Berthold Göttgens. A single-cell resolution map of mouse hematopoietic stem and progenitor cell differentiation. *Blood*, 128(8):e20–e31, 2016.
7. Bo Li, Nathanael Fillmore, Yongsheng Bai, Mike Collins, James A Thomson, Ron Stewart, and Colin N Dewey. Evaluation of de novo transcriptome assemblies from rna-seq data. *Genome biology*, 15(12):553, 2014.
8. Differential expression testing.
9. F Alexander Wolf, Philipp Angerer, and Fabian J Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19(1):15, 2018.
10. Dataset downloads.
11. Dataset downloads.
12. Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.
13. Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.
14. Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell rna-seq

- denoising using a deep count autoencoder. *Nature communications*, 10(1):390, 2019.
15. Yue Deng, Feng Bao, Qionghai Dai, Lani F Wu, and Steven J Altschuler. Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning. *Nature methods*, 16(4):311, 2019.

5.2 Summary and Contributions

In this chapter we have shown that the pen-stroke method is capable of identifying regulatory gene combinations that correspond to cellular heterogeneity present in RNA-Seq datasets. Through our analysis we have shown that the nonlinear nature of this method provides an advantage over linear methods such as PCA. Furthermore, we have also shown that pen-strokes provide us with a flexible framework to interpret cellular heterogeneity at multiple granularities. We have also shown that these pen-strokes, either when taken alone or as nonlinear combinations of them represent meaningful functional features, which are verified by separate phenotypical measurements.

As future work, an interesting aspect that remains to be investigated is the interpretation of pen-strokes that are not immediately correlated to a functional feature or a group of cells. We hypothesize that these may indeed represent subtle functional features of genes that contribute to the overall heterogeneity, such as cell-cycle genes. We propose two further studies to validate this hypothesis.

First, we need to conduct an ablation study on the pen-strokes, where we eliminate a single pen-stroke and investigate how it affects to the observed heterogeneity. This would provide us insight into the importance of each pen-stroke for the heterogeneity of the overall data.

Second, for pen-strokes that do have a significant contribution to the overall heterogeneity of the data, but is as of yet unverifiable by existing literature as we have done in the case of identifying the pen-stroke that separates the CD4 Memory vs Naive T cells, experimental validation of the genetic components may be required.

Currently, the challenge we are facing is introducing quantitative measures to investigate the contributions of pen-strokes, as we resort to visual inspection and interpretation.

The ambition of this method is that in keeping with our motivations in Chapter 4, we believe that this method can provide a canonical representation of genetic dynamics where new data can be meaningfully integrated (as the sparse autoencoders are inherently parametric).

Chapter 6

Concluding Remarks

In this thesis, I have discussed the fundamentals of high-dimensional data analysis, some of the prominent problems and gaps in such analysis and variants of unsupervised neural networks may be used to approach solving these problems and bridging these gaps.

In Chapter 1, I have discussed the background of high-dimensional data analysis. I first elaborate on the practical scenarios where high-dimensional data arises, and I describe how exploratory data analysis is an important part in analysing such data. Next, I describe some of the fundamental methods used in the domain of exploratory high-dimensional data analysis, and some of the prominent weaknesses and short-comings of these methods. I have emphasized the compatibility of Hebbian learning as a robust alternative to latent variable model for visualization and dimensionality reduction, and how regularization can be used to improve the interpretability of unsupervised neural networks.

In Chapter 2, I describe my first approach to solving an important issue with the Self Organizing Map (SOM) algorithm and its variant methods, namely the poor representation of gaps and distinct groups or clusters in the visualizations of the SOM algorithm. I do so by presenting a modified version of the Growing SOM (GSOM) algorithm, which we call the Gap Encoding GSOM (GEGSOM). However, we observe that this approach is not robust in terms of its generalizability to very high dimensions. Nevertheless, we show that it is still highly useful in terms of its identification of nonlinear structures with minimal assumptions.

Expanding on ideas underpinning the GEGSOM algorithm, and drawing inspiration from recent advances in machine learning, I next present a new dimensionality reduction algorithm called Self Organizing Nebulous Growths (SONG) capable of incrementally visualizing highly nonlinear structures inferred from high-dimensional data, which I have

compiled in Chapter 3. I show that the quality visualization of this method is superior or on par with the existing state-of-the-art non-parametric visualization methods. Furthermore, we also show being a parametric method, SONG has the added ability of incremental visualization with low distortion in the presence of heterogeneous data.

It is my humble belief that the introduction of SONG opens up new avenues in exploratory data analysis. First, SONG facilitates the visualization of data in an incremental manner. This allows analysts to cope with highly changing data sizes. Second, we can obtain abstractions of the data, which are independent of nuisance factors such as technical noise, and can be hierarchically manipulated to represent different granularities of the topological structure of the data.

In Chapter 4, I discuss the applicability of this new algorithm in Single Cell data analysis, which is a domain where high-dimensional data visualization is a frequent problem. We leverage vector quantization used in SONG to obtain abstractions of cells which we call landmarks from high-dimensional data. I further illustrate the utility of SONG in up and coming research problems such as producing atlases of multi-source high-dimensional data.

Finally, in Chapter 5, I describe how we use deep learning to analyse and explain the heterogeneity of single-cell high-dimensional data. I show in this work that deep learning can be used to extract complex non-linear relationships between variables that contribute to the overall variation of cells.

Putting this research into context, I have shown in this work, that Hebbian learning and Vector Quantizing Artificial Neural Networks can be leveraged to produce robust dimensionality reduction algorithms. Furthermore, I have shown that using unsupervised neural networks and appropriate regularization, we can extract motifs from high dimensional data.

The most intriguing aspect of the methods I have presented throughout this thesis is that being parametric and unsupervised, they lay the foundation for a broad, comprehensive paradigm of incremental learning. Having observed the results using the methods I have discussed, let us lay out the fundamental tenets of our incremental learning using unsupervised learning as follows:

- **Incompleteness of Data:** The most fundamental assumption is that the data is incomplete, and this assumption was utilized heavily in our Chapter 3. This means to say that the current data may not adequately describe the true objective structure. The currently available data may be missing groups or subgroups of data points, which may be discovered at consequent experiments.

- **Parametric Spatial Approximations:** The approximations obtained for the structure of high-dimensional data is parametric in nature, and it parametrizes the input space spatially. The methods I have presented in both Chapters 2 and 3 follow this tenet. The parameters may adjust for subsequent datasets. However, we note that these spatial approximations may be explicit (as in the case of SONG, where each coding vector represents a spatial region) or implicit (as in the case of pen-strokes, where each pen-stroke represents a different local feature that defines a spatial locality).
- **Control Over ‘Memory’ of Old Structures:** While data obtained in subsequent experiments may change the observed structure, the methods should provide a framework to control how much of the old structure is retained when adapting to the new structure. In Chapter 4, we have introduced a new parameter called ‘agility’ to the SONG algorithm in an attempt to regulate this.
- **Robustness of Parametrization:** The parametric approximations should be highly tolerant to undesired variability. This undesired variability may be encountered in the form of technical noise (as we observed in our methods proposed in Chapters 2 and 3), or batch effects. However, having control over what counts as ‘noise’ may allow analysts to manipulate the granularity of the approximations, which we used in our landmarking concept in Chapter 4.
- **Interpretability of Parametrizations:** In the case of SONG and its possible extensions, the interpretability of the parametrization is straightforward as we have discussed in Chapter 3. However, in implicit approximations such as our pen-stroke extraction in Chapter 5, interpretability is a salient property to strive for.

As future work, I propose studies to be conducted expanding on each of these tenets. For any parametric unsupervised neural network method, we may consider each of these tenets as a different aspect of the ‘strength’ of the said method.

As an example of these studies, we propose that a more robust experimental study on how to manipulate the agility of the SONG algorithm may shed light into the dynamic nature of the structure of incremental data. Although we have shown that varying the agility parameter affects how much of the old structure is retained in Chapter 4, this study needs thorough experimental expansion.

Furthermore, the use of each of the output dimensions of SONG as a latent variable, which can be then used to *reconstruct* high-dimensional data remains to be investigated. For example, such an approach would enable many algorithms in the same class as SONG (including SOM and GSOM) to be a more robust ‘encoder’ in an autoencoder.

If successful, with the use of such an autoencoder, one may produce a dynamic, spatially interpretable heuristic for determining the network topology of an unsupervised generative neural network model.

In terms of the application area, the use of SONG and pen-strokes may have broad applications in standardizing RNA-Seq experimental results. Furthermore, expansive experiments of distinct pen-strokes, using RNA-Seq data obtained from various species and/or tissues may provide us with an ontology of motifs, which governs the cellular genetic mechanisms. We could possibly identify using the data, common motifs of genes that regulate specific phenotypical functions (for instance, cell-cycle). We deem this a feasible goal due to the promise shown by the identification of CD4 T memory/naive regulator motifs identified in our work in Chapter 5. The most intriguing aspect of this would be the discovery of new genes, which till now, have not been associated with known functions through experiments, but are commonly associated with specific functions by pen-strokes. In such cases, the pen-stroke method can provide directions for experiment design as well. Expanding on the idea of standardizing the representation of the biological states of cells, incremental algorithms such as SONG, being parametric, provides a way to ‘prototype’ cellular entities as we have shown with the landmarking of single cells in Chapter 4. These entities may be specific states in a cell’s life cycle, groups or sub-groups of cells.

Appendix A

Appendix : ENVirT: inference of ecological characteristics of viruses from metagenomic data

Being a co-author in this work[\[85\]](#), I have contributed to this work in multiple ways. First, I contributed to programming the algorithm, optimization of the running time of the algorithm as well as distribution and packaging. I also contributed in manuscript preparation, revising the manuscript and finally, presenting the paper on behalf of the authors at the International Conference on Bioinformatics (InCoB) 2018, which was held at the Jawaharlal Nehru University, New Delhi, India.

RESEARCH

Open Access

ENVirT: inference of ecological characteristics of viruses from metagenomic data



Duleepa Jayasundara^{1*} , Damayanthi Herath^{2,3}, Damith Senanayake², Isaam Saeed², Cheng-Yu Yang⁴, Yuan Sun², Bill C. Chang⁵, Sen-Lin Tang^{4*} and Saman K. Halgamuge^{2,6}

From 17th International Conference on Bioinformatics (InCoB 2018)
New Delhi, India. 26-28 September 2018

Abstract

Background: Estimating the parameters that describe the ecology of viruses, particularly those that are novel, can be made possible using metagenomic approaches. However, the best-performing existing methods require databases to first estimate an average genome length of a viral community before being able to estimate other parameters, such as viral richness. Although this approach has been widely used, it can adversely skew results since the majority of viruses are yet to be catalogued in databases.

Results: In this paper, we present ENVirT, a method for estimating the richness of novel viral mixtures, and for the first time we also show that it is possible to simultaneously estimate the average genome length without a priori information. This is shown to be a significant improvement over database-dependent methods, since we can now robustly analyze samples that may include novel viral types under-represented in current databases. We demonstrate that the viral richness estimates produced by ENVirT are several orders of magnitude higher in accuracy than the estimates produced by existing methods named PHACCS and CatchAll when benchmarked against simulated data. We repeated the analysis of 20 metavirome samples using ENVirT, which produced results in close agreement with complementary *in vitro* analyses.

Conclusions: These insights were previously not captured by existing computational methods. As such, ENVirT is shown to be an essential tool for enhancing our understanding of novel viral populations.

Keywords: Richness estimation, Viral metagenomics, Average genome length

Background

Viruses account for the significant majority of Earth's biota and are vital in shaping our biosphere, but just as critically are causative agents of a plethora of plant, animal and human diseases. Despite their abundance, we are still only beginning to understand their overarching ecological roles, with the vast majority of viruses yet to be discovered. Due to the absence of conserved marker genes

such as the 16S rRNA gene found in bacteria, which has been used to identify bacterial species as well as their phylogeny [1], early *in vitro* approaches have been limited to analyzing individual viruses in isolation [2, 3]. However, viral populations often co-occur depending on their host or environment and holistic approaches are required to understand their overall functionality. Recent attempts to study such viral mixtures using metagenomics have provided significant insights into the dynamics between viral communities, their hosts and their environment. With the rapid development of metagenomics protocols tailored toward viral mixtures, modern computational approaches can now infer various ecological parameters, such as:

*Correspondence: duleepal@gmail.com; sltang@gate.sinica.edu.tw

¹School of Public Health and Community Medicine, University of New South Wales, Randwick, NSW 2052, Australia

⁴Biodiversity Research Center, Academia Sinica, Nan-Kang, Taipei 11529, Taiwan

Full list of author information is available at the end of the article



© The Author(s). 2019 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated.

species abundance, richness, the Shannon-Weiner Index [4], and population evenness [5–7], directly from next-generation sequencing data (i.e. the “metavirome”). These inferred parameters provide essential information that can be used to probe deeper into the population dynamics of viral communities. Developing computational models that can produce robust and unbiased estimates of these parameters, however, is non-trivial.

For instance, PHACCS [8] uses an extended Lander-Waterman model [9, 10] to predict theoretical distributions of virome data that are compared to a distribution of observed virome data. The ecological parameters of the underlying viral populations are thus inferred when the difference between a theoretical distribution generated by PHACCS and the observed distribution is minimal. Other methods, such as CatchAll, take a similar approach but use different representations of virome data and operate under a different set of assumptions about the relationship between the underlying viral populations and the observed metagenomic data [11]. Other methods include those which rely on information in existing genomic databases, and are best applied when samples are known to contain viral types that are represented in these databases ([12–14]). In general, PHACCS and its extensions thereof, are the most widely used and are the best performing [15, 16]. However, the limitation of these methods is the assumption that the average genome length of a viral mixture in an uncharacterized sample is known, which in reality is not the case. This can lead to potentially erroneous or misleading results if an incorrect average genome length of a virome is assumed [8]. Consequently, these methods are paired with complementary methods to infer an average genome length of a virome [5, 6, 13, 17].

These complementary methods infer an average genome length using three broad approaches. The first approach makes an assumption that similar viral genomes are in similar environments, and uses the average genome length of known viruses in those environments as input to PHACCS (i.e. 50 kbp for marine viruses) [10, 16, 18]. This approach does not hold for the vast majority of viromes, since the variation in genome length can be quite large between viruses of similar environments (predominantly distributed from 1.2 kbp - 2.5 Mbp, based on 4991 viral genomes catalogued by NCBI). The second approach uses database-driven computational methods such as GAAS ([5, 6, 13, 17]) to infer an average genome length based on sequence similarity to existing viral genomes. These methods are heavily biased due to the under-representation of novel viral types in current databases. The third approach is to use *in vitro* methods, including: Transmission Electron Microscopy (TEM), traditional culture-based approaches, or techniques such as Pulsed Field Gel Electrophoresis (PFGE) [18–20].

PFGE is considered to be the gold standard in experimentally determining the length of DNA molecules, but requires a relatively large volume of DNA [19, 21] and is biased toward the more abundant viruses in a sample (i.e. dependent on the relative concentration of DNA per viral type). As such, these methods are not ideal for estimating the average genome length of a virome, leaving PHACCS and its derivatives poorly equipped to analyze the virome of environmental samples.

In this paper, we present ENVirT, a database-independent algorithm which estimates ecological parameters, including the viral richness, and for the first time also provides a simultaneous estimate of the average genome length. The formulation of ENVirT extends the original PHACCS model, and introduces a novel 4-dimensional heuristic optimization algorithm based on the Genetic Algorithm in combination with a unique niching strategy to arrive at estimates of both viral richness and average genome length. ENVirT requires only virome data as input, and does not rely on any other information or external databases during parameter estimation, which makes it better suited to analyzing experimental samples that typically contain novel viruses.

We also show that re-analysis of 20 virome samples from a diverse set of environments and sampling experiments produces novel insights into the respective viral mixtures that were previously not captured when analyzed using PHACCS.

Methods

ENVirT is based on a novel 4-dimensional heuristic optimization algorithm to simultaneously estimate viral richness, evenness and for the first time the average genome length of a virome. It is formulated as an extension to the original PHACCS algorithm. The proposed extensions allow ENVirT to perform faster and independently of other databases required by PHACCS. The subsequent derivation of the ENVirT algorithm uses the following notation:

M denotes the number of genotypes (richness); L denotes the average genome length of each genotype (bp); f_i represents the relative abundance of the i^{th} genotype ($i \in 1, \dots, M$), where i is the abundance rank of a genotype after they have been sorted based on their relative abundance; R denotes the total number of reads in a metavirome, and r is the corresponding average read length (bp); o denotes the minimum overlap for assembling reads (bp); $(C_1, C_2, C_3, \dots, C_R)$ denotes the observed contig spectrum, where C_q ($q \in 1, 2, 3, \dots, R$) is the observed number of contigs that comprise q reads (e.g. C_1 is the number of singletons, C_2 is the the number of contigs each having 2 reads, etc.); and $O_q = q.C_q$ is the number of reads that form contigs that comprise q reads

($q \in 1, 2, 3, \dots, R$). An important assumption made in this formulation is that the f_i s follow one of the four theoretical distributions: power-law, exponential, logarithmic or log-normal, as defined in Eqs. 1, 2, 3 and 4 respectively where d denotes the distribution specific real valued parameter. This assumption is justified in [22–24].

$$f_i = \frac{i^{-d}}{\sum_{j=1}^M j^{-d}} \quad (1)$$

$$f_i = \frac{\exp(-i.d)}{\sum_{j=1}^M \exp(-j.d)} \quad (2)$$

$$f_i = \frac{(\log(i+1))^{-d}}{\sum_{j=1}^M (\log(j+1))^{-d}} \quad (3)$$

$$f_i = \frac{\exp(m_i.d)}{\sum_{j=1}^M \exp(m_j.d)} \quad (4)$$

$$m_i = \frac{M}{\sqrt{2\pi}} \cdot \left(\exp\left(\frac{-t_i^2}{2}\right) - \exp\left(\frac{-t_{i+1}^2}{2}\right) \right)$$

$$t_1 = -\infty, t_{M+1} = +\infty,$$

$t_{i+1} = \sqrt{2} \cdot \text{erf}^{-1}\left(\frac{2}{M} + \text{erf}\left(\frac{t_i}{\sqrt{2}}\right)\right)$ where $d \geq 0$ and $j \in \{1, 2, \dots, M\}$, erf denotes the error function and erf^{-1} denotes the inverse error function.

All four functional forms of f_i (i.e. Eqs. 1, 2, 3 and 4) depend on M and a distribution specific parameter d . Let us denote the function defining the relative abundance of the i^{th} genotype as $F_i(M, T, d)$ where T denotes the distribution function given by Eqs. 1, 2, 3 or 4. Once M , T and d are known, the relative abundance of each genotype contained in the virome can be calculated.

Following the derivation in [10], if the expected number of reads contributing to contigs having exactly q number of reads is E_q ($q \in \{1, 2, 3, \dots, R\}$):

$$E_q = \sum_{i=1}^M F_i(M, T, d) \cdot R \cdot q \cdot p_i^{(q-1)} \cdot (1-p_i)^2 \quad (5)$$

where,

$$p_i = 1 - \exp\left(-\left(r - o\right) \cdot F_i(M, T, d) \cdot \frac{R}{L}\right) \quad (6)$$

Accordingly, the expected contig spectrum of a metagenome having population parameters M, L, T, d and, sequenced and assembled with parameters R, r, o is:

$\left(\frac{E_1}{1}, \frac{E_2}{2}, \frac{E_3}{3}, \dots, \frac{E_R}{R}\right)$. Given the values of R, r, o and $(O_1, O_2, O_3, \dots, O_R)$, our aim is to find M, L, T and d such that the difference between $(O_1, O_2, O_3, \dots, O_R)$ and $(E_1, E_2, E_3, \dots, E_R)$ is minimum. Similar to [8, 10] we use the variance weighted squared difference between $(O_1, O_2, O_3, \dots, O_R)$ and $(E_1, E_2, E_3, \dots, E_R)$ denoted by $S(M, L, T, d)$ as the similarity measure between the observed and expected contig spectra.

The ENVirT formulation is thus the minimization of the error between an expected contig spectrum E and

the experimentally observed contig spectrum O , which is represented by an error function S :

$$S(M, L, T, d) = \sum_{q=1}^R \frac{(O_q - E_q)^2}{V_q^2} \quad (7)$$

where,

$$V_q^2 = \sum_{i=1}^M F_i(M, T, d) \cdot R \cdot q \cdot p_i^{(q-1)} \cdot (1-p_i)^2 \cdot \left(1 - q \cdot p_i^{(q-1)} \cdot (1-p_i)^2\right) \quad (8)$$

This error function S has multiple local minima but one global minimum (see Additional file 1). PHACCS now assumes that L is known, thereby greatly simplifying the optimization problem.

However, there are undesirable consequences to this assumption as an incorrect value of L has been reported to cause wild fluctuations in the estimation of any ecological parameter [8]. We propose that since L is unknown for any given real-world data set, L should be treated as such and instead estimated during the minimization of S .

The landscape of S is such that an optimal solution can be found using brute force but is subject to multiple local minima. We propose an optimization scheme based on the standard Genetic Algorithm (GA), which uses a heuristic approach to explore the parameter landscape of S . GA has been widely used in the scientific community to solve combinatorial optimization problems and since M and L are integers and S is non-linear in our problem formulation, GA is well suited to minimize S . However, since GA is also susceptible to local optima, we adopt a niching strategy [25] as follows: we first applied niching along the dimension of T for each of the four candidate distributions (see Additional file 1); we then applied niching along the dimension of L for each subspace of T separating the search space further into N_L sub-spaces. Table 1 shows that when ENVirT is applied with this niching strategy the algorithm is better able to find an optimal solution.

Optimization procedure

Inputs to ENVirT are the observed contig spectrum (C), number of reads in the virome (R), average read length in base-pairs (r), minimum overlap considered in assembling reads (o) and the boundaries of the domain within which values for M, L, T and d should be searched for. These boundaries are denoted the subscripts LB and UB , which correspond to the lower and upper bound of a variable, respectively. ENVirT outputs estimates for M, L, T and d , along with a residual model error denoted by S_{min} (i.e. the minimum value of Eq. 7). These estimates are obtained by iteratively performing the following steps:

- S1: For each of the candidate distributions along the niched dimension T , perform steps S1-S3:

Table 1 Performance of ENVirT in comparison to standard GA algorithm on simulated contig spectra

Input parameters (expected result)						Estimated values by ENVirT					Estimated values by GA without niching				
L_0	M_0	T_0	d_0	Evenness	f_{max}	L	M	T	d	S_{min}	L	M	T	d	S_{min}
12500	300	exp	0.030	0.790	2.956%	12500	300	exp	0.030	0.00×10^0	39500	12400	exp	0.095	3.49×10^{-2}
12500	1000	log	0.900	0.995	0.661%	14972	838	log	0.893	6.56×10^{-3}	310000	100	lgn	1.063	2.59×10^{-1}
12500	5000	lgn	2.500	0.655	11.849%	12500	5000	lgn	2.500	0.00×10^0	12500	5000	lgn	2.500	0.00×10^0
12500	10000	pl	0.700	0.913	1.997%	12500	10000	pl	0.700	0.00×10^0	29500	1400	log	1.911	6.38×10^0
50000	300	exp	0.030	0.790	2.956%	50000	300	exp	0.030	0.00×10^0	41000	100	pl	0.378	1.53×10^{-1}
50000	1000	log	0.900	0.995	0.661%	50000	1000	log	0.900	0.00×10^0	100500	600	lgn	0.531	3.48×10^{-2}
50000	5000	lgn	2.500	0.655	11.849%	50000	5000	lgn	2.500	0.00×10^0	50000	5100	lgn	2.506	1.92×10^{-2}
50000	10000	pl	0.700	0.913	1.997%	52787	10175	pl	0.707	1.72×10^{-3}	41000	9800	pl	0.677	2.22×10^{-2}
125000	300	exp	0.030	0.790	2.956%	125000	300	exp	0.030	0.00×10^0	58500	11000	exp	0.014	2.70×10^{-2}
125000	1000	log	0.900	0.995	0.661%	125000	1000	log	0.900	0.00×10^0	69000	1800	log	0.943	3.94×10^{-4}
125000	5000	lgn	2.500	0.655	11.849%	125000	5000	lgn	2.500	0.00×10^0	125000	5000	lgn	2.500	0.00×10^0
125000	10000	pl	0.700	0.913	1.997%	116341	9824	pl	0.691	1.96×10^{-4}	203000	15000	lgn	1.922	9.34×10^{-1}
300000	300	exp	0.030	0.790	2.956%	300000	300	exp	0.030	0.00×10^0	67000	400	lgn	0.543	5.36×10^{-2}
300000	1000	log	0.900	0.995	0.661%	217303	1373	log	0.899	1.26×10^{-7}	156000	1900	log	0.931	1.93×10^{-5}
300000	5000	lgn	2.500	0.655	11.849%	300000	5000	lgn	2.500	0.00×10^0	310000	7400	lgn	2.635	1.09×10^{-1}
300000	10000	pl	0.700	0.913	1.997%	277000	9800	pl	0.690	3.00×10^{-5}	77000	5600	log	1.658	2.97×10^{-2}

Contig spectra were generated with parameters: $R = 10000$, $r = 100bp$ and $\alpha = 35bp$. pl = power-law distribution, exp = exponential distribution, log = logarithmic distribution and lgn = lognormal distribution. f_{max} = relative abundance of the dominant genotype. S_{min} = the value of the cost function corresponding to the estimated values of M , L , T and d . GA = Genetic Algorithm. We chose $M_{LB} = 1$, $M_{UB} = 15000$, $L_{LB} = 10000$, $L_{UB} = 310000$, $d_{LB} = 0.01$ and $d_{UB} = 5$ for both ENVirT and GA without niching. In order to apply the second niching strategy of ENVirT, we chose $N_L = 29$

- S2: Choose a value for N_L to apply the second niching strategy. In this step, the L axis is divided into N_L number of overlapping windows having a constant window width. The window width (W_L) along the L dimension is calculated as follows.

$$W_L = \frac{2(L_{UB} - L_{LB})}{(N_L + 1)}.$$

Let $W_{sp}(j)$ and $W_{ep}(j)$ be the starting and ending positions respectively of the j^{th} window ($j \in 1, \dots, N_L$) along the L dimension. Then,

$$W_{sp}(1) = L_{LB}$$

$$W_{sp}(j+1) = W_{sp}(j) + \frac{1}{2} \cdot W_L \text{ for } j = 1, \dots, (N_L - 1)$$

$$W_{ep}(j) = W_{sp}(j) + W_L \text{ for } j = 1, \dots, N_L$$

This definition ensures that an overlap of $\frac{1}{2} \cdot W_L$ exists between each consecutive pair of windows such that the L values occurring along a boundary in one window occur in the middle of the next window. This property is important to avoid the possible negligence of boundary values by GA.

- S3: Perform GA to find the minimum of $S(M, L, T, d)$ (i.e. the cost function for GA) within each of the N_L number of sub-spaces where the j^{th} subspace is defined by:

$$M_{LB} \leq M \leq M_{UB}$$

$$W_{sp}(j) \leq L \leq W_{ep}(j) \text{ where } (j \in 1, \dots, N_L)$$

$$d_{LB} \leq d \leq d_{UB}$$

- S4: Out of the $4N_L$ solutions obtained by performing GA on $4N_L$ number of sub-spaces, identify M , L , T and d

corresponding to the solution with the minimum cost function (i.e. $S(M, L, T, d)$) value.

Practical considerations

For the practical application of the ENVirT algorithm, we treat d as a discrete variable with a step size of 0.01. Moreover, we do not observe contig spectra with nonzero values for C_R . In fact, the maximum q with a non-zero value for C_q is much less than R in real-world metagenomes. Therefore, depending on the length of the observed contig spectrum, we can safely consider a value much less than R for q_{max} in the actual calculation, and apply a cutoff to the maximum length of the spectrum. We recommend discretizing M with a minimum step size of $10^{(\text{ceiling}(\log(M_{UB})) - 2)}$ when $\log(M_{UB}) > 2$. We also recommend discretizing L with a minimum step size of $0.025 W_L$. Once a solution is found in the discretized search space, we iteratively reduce the step size and repeat the optimization procedure. In its current formulation, ENVirT produces relatively accurate parameter estimates when the variation of genome lengths L satisfies $-\log(v) > 2$ in simulated data. Filtering non-viral DNA in vitro prior to sequencing using DNase to remove free DNAs prior to viral DNA isolation, or using a computational method post-sequencing, could improve the integrity of downstream processing by ENVirT. An example of the latter is to map the sequences against an existing database

such as GenBank to identify non-viral DNA sequences [26]. However, it could be challenging for environments lacking host genomic information.

For the convenience of users, the software bundle containing the algorithm implementation is available for download at <https://github.com/senanayaked/ENVirT.git>. The scripts are available in Matlab (MathWorks, Massachusetts, USA) which can be executed via a user-friendly graphical user interface. Data and instructions for a sample execution of the algorithm are contained in the README.md file available with this download.

Experimental metavirome data

Table 2 summarizes the 20 publicly available experimental virome that were analyzed in this study. The objective of selecting these datasets for analysis was to capture a variety of sampling environments, protocols and viral populations with which to validate the utility of ENVirT.

Simulations

To objectively evaluate the performance of ENVirT in comparison to PHACCS and Catchall, we artificially constructed viromes under two simulation scenarios. These simulation scenarios were designed to mimic the variability that has been observed in real-world data sets.

When estimating viral richness using virome data, the evenness of the underlying populations plays a critical role in arriving at robust estimates. Evenness in the context of a viral population is given by the following equations:

$$\text{evenness} = \frac{-\sum_{i=1}^M f_i \cdot \ln(f_i)}{\ln(M)} \quad (9)$$

where f_i is the relative abundance of the i^{th} genotype ($i \in 1, 2, 3, \dots, M$). Evenness measures whether the population is skewed toward a particular set of dominant viral types (i.e. when evenness approaches 0) or whether all viral types are equally abundant (i.e. when evenness approaches 1); the range of possible evenness values for any given population is (0, 1].

Simulation scenario 1:

We simulated mixtures of viral populations that had varying degrees of viral richness and evenness, subject to the constraint that each viral type has the same average genome length. For all simulated mixtures, the contig spectra were generated based on a total read count of 10,000, a read length of 100bp and a minimum overlap of 35bp. These parameters were chosen in accordance with the default parameters used by Circonspect.

Simulation scenario 2:

As an extension to Simulation Scenario 1, we simulated mixtures of viral populations that not only had varying degrees of richness and evenness, but also variable

genome lengths. This simulation scenario is more in-line with the expected characteristics of real-world samples. Here we assumed that genome lengths are normally distributed with $\mathcal{N}(L, (L.v)^2)$ where L denotes the average genome length and v denotes the coefficient of variation of the considered genome length distribution.

Results

Simulating viral mixtures

Quantitatively comparing the accuracy of viral richness estimates requires data that represents ground truth. We generated such data sets by simulating viromes based on a known number of viral genotypes, average genome length and relative abundance distribution. This simulation study follows the same methodology as previous studies of PHACCS and CatchAll [3, 8, 11]. We provided PHACCS with the true average genome length (L), whereas ENVirT was required to estimate L based only on the simulated data itself. We note that this benchmark study is highly advantageous to PHACCS in that PHACCS is given critical information that ENVirT will be required to estimate.

Simulation scenario 1 (fixed average genome length):

We divided these simulations into two groups. First, we limited the range of simulations to a set of 16 conservative benchmark data sets that represent low to moderately complex viral mixtures. As an extension to these results, we then show how the algorithms perform on a wider simulation range of 77 benchmark data sets that were designed to identify the parameter limits at which reliable estimates are attainable for each method.

We observed that ENVirT produced an average estimation error of 0.91%, whereas PHACCS produced an average estimation error of 585.87% (Table 3). We also see that ENVirT is better able to optimize the model parameters, and select the most appropriate relative abundance model in all 16 simulations. In several cases, PHACCS and CatchAll were not able to produce any reasonable estimates. In accordance with previous reports, CatchAll produced significant overestimates when a discounted parametric model was not selected by its internal model selection procedure [15, 16].

The results for the extended 77 data sets indicate a similar trend as the initial set of 16 simulations, with ENVirT outperforming both PHACCS and CatchAll (Figure S4 of Additional file 2). We found that at certain extremes, neither PHACCS nor CatchAll was able to produce an estimate of viral richness. Interestingly, these instances tend to underestimate richness (up to -83.48%) as the number of viral types increases. In contrast, PHACCS tends to largely overestimate richness when the true value of richness is in the range of 300-10,000. ENVirT is stable at both extremes (low and high values of richness), and has

Table 2 Comparison between PHACCS+GAAS/BLAST and ENViT estimates of viral richness and average genome length on viral metagenomes derived from different environments

Source	Sample name	ENViT			PHACCS		
		<i>L</i>	<i>M</i>	Evenness	<i>L</i> [§]	<i>M</i>	Evenness
French Lakes	Lake Bourget	62279	42999	0.84862	13089*	33311	0.92228
[6]	Lake Pavin	81110	792	0.82202	12274*	2628	0.89747
Feitsui	V1	24112	587	0.84216	44297*	3059	0.72402
Reservoir	V2	16613	1288	0.88611	43926*	513	0.93042
[17]	V3	31019	617	0.93707	95269*	174	0.94079
	V4	16535	1092	0.89225	62395*	399	0.91161
	V5	15177	1121	0.89919	41377*	419	0.93946
	V6	46677	1929	0.79735	125321*	221	0.90320
Fermented	Shrimp	27337	4931	0.92204	39839 [†]	4606	0.90349
food	Kimchi	53837	1395	0.88842	48220 [†]	1415	0.89653
[32]	Sauerkraut	277163	719	0.80599	36494 [†]	2692	0.86619
Perennial ponds	Ilij	75242	1703	0.88137	71477*	1687	0.88550
of the	Molomhar	394921	223	0.87082	60959*	1318	0.89228
Mauritanian Sahara	Hamdoun	176346	515	0.66600	60479*	217	0.88719
[5]	El Berbera	81118	6199	0.69961	76501*	5696	0.71009
Human	X-1	175863	559	0.83496	50000 [‡]	815	0.92174
gut	H1-1	497223	609	0.62918	50000 [‡]	397	0.92259
[28]	H1-2	387877	212	0.73163	50000 [‡]	353	0.92904
	H1-7	282786	151	0.78132	50000 [‡]	315	0.92531
	H1-8	570706	121	0.68525	50000 [‡]	239	0.94400

M = estimated richness, *L* = estimated average genome length (bp). § = Average genome length used in the original publication. * = An estimate based on GAAS software ([33]). † = An estimate based on a BLAST search. ‡ = Assumed value

a maximum estimation error of only − 16.58% at a richness value of 45,000. We also note that when estimating an average genome length using only information contained in the aggregate statistics of contig spectra, ENViT performs with an average error of 9.13% (Fig. 1). ENViT's estimation accuracy of *L* tends to fall at *L* values greater than 100 kbp compared to lower *L* values. Other methods such as GAAS use sequence similarity to compare the DNA sequence data of a virome to known viral sequences in databases. Since ENViT and these database-dependent methods use fundamentally different approaches and entirely different types of data to estimate an average genome length, we do not directly compare these methods here.

Simulation scenario 2 (variable genome lengths):

In general, it is reasonable to assume that constituent viruses in a virome do not have the same genome length. To evaluate how our method performs in comparison to PHACCS in this regard, we generated an additional 140 contig spectra representing populations with predefined degrees of genome length variation. The resulting spectra represent populations with viral genome lengths distributed according to: $N(L, (Lv)^2)$, where *L*

denotes the average genome length and *v* is the coefficient of variation. For comparison between PHACCS and ENViT, we considered only a power-law distribution as the model for relative viral abundance to ensure that viral types of lower abundance are captured by each respective model. The results indicate that for both PHACCS and ENViT there is an estimation error that increases exponentially with the increase in genome length variation *v*. Notably, we observed that ENViT is more performant relative to PHACCS at larger values of variation. ENViT is up to 55.62% more stable than PHACCS in the presence of genome length variation at lower viral richness (*M* = 300) and 9.80% more stable at higher viral richness (*M* = 10,000).

With respect to ENViT's estimation of *L* in the presence of a wide variation of genome length (i.e. when $-\log(v) < 2$), we note that the error of the estimates produced increases. Conversely, if the expected variation between genome lengths is sufficiently small (i.e. $-\log(v) > 2$), ENViT is observed to produce more robust estimates (Fig. 2).

Normal distributions with different variances were selected as the simplest and appropriate forms to model real world scenarios. Two alternative approaches that may be taken to model genome length distributions are the

Table 3 Performance comparison between ENVirT, PHACCS and CatchAll on simulated contig spectra

Input parameters (expected result)						ENVirT					PHACCS					CatchAll
L_0	M_0	T_0	d_0	Evenness	f_{max}	M	T	d	S_{min}		M	T	d	S_{min}		M
12500	300	exp	0.030	0.790	2.956%	300	exp	0.030	0.00×10^0		4096	exp	0.030	1.37×10^{-3}		2829.6^P
12500	1000	log	0.900	0.995	0.661%	1000	log	0.900	0.00×10^0		1000	log	0.900	0.00×10^0		92628.3^C
12500	5000	lgn	2.500	0.655	11.849%	5000	lgn	2.500	0.00×10^0		23563	pl	1.313	1.01×10^4		3246.1^P
12500	10000	pl	0.700	0.913	1.997%	10000	pl	0.700	0.00×10^0		10000	pl	0.700	0.00×10^0		696.3^P
50000	300	exp	0.030	0.790	2.956%	300	exp	0.030	0.00×10^0		10000	exp	0.030	4.31×10^{-4}		15712.6^P
50000	1000	log	0.900	0.995	0.661%	1000	log	0.900	0.00×10^0		1000	log	0.900	0.00×10^0		n/a
50000	5000	lgn	2.500	0.655	11.849%	5000	lgn	2.500	0.00×10^0		4996	lgn	2.500	1.78×10^{-3}		799.8^P
50000	10000	pl	0.700	0.913	1.997%	10000	pl	0.700	0.00×10^0		10000	pl	0.700	0.00×10^0		413688.9^C
125000	300	exp	0.030	0.790	2.956%	300	exp	0.030	0.00×10^0		10000	exp	0.060	1.87×10^{-4}		70340.9^C
125000	1000	log	0.900	0.995	0.661%	1000	log	0.900	0.00×10^0		1000	log	0.900	0.00×10^0		n/a
125000	5000	lgn	2.500	0.655	11.849%	5000	lgn	2.500	0.00×10^0		5000	lgn	2.500	0.00×10^0		2303.2^P
125000	10000	pl	0.700	0.913	1.997%	10000	pl	0.700	0.00×10^0		10000	pl	0.700	0.00×10^0		n/a
300000	300	exp	0.030	0.790	2.956%	300	exp	0.030	0.00×10^0		4096	exp	0.030	7.92×10^{-5}		160243.9^C
300000	1000	log	0.900	0.995	0.661%	1000	log	0.900	0.00×10^0		1000	log	0.900	0.00×10^0		n/a
300000	5000	lgn	2.500	0.655	11.849%	5000	lgn	2.500	0.00×10^0		5000	lgn	2.500	0.00×10^0		146552.7^C
300000	10000	pl	0.700	0.913	1.997%	8547	pl	0.689	3.00×10^{-3}		10000	pl	0.700	0.00×10^0		n/a

Contig spectra were generated with parameters: $R = 10000$, $r = 100$ bp and $o = 35$ bp. Both ENVirT and PHACCS were provided with the true average genome length (L_0) value. pl = power-law distribution, exp = exponential distribution, log = logarithmic distribution and lgn = lognormal distribution. S_{min} = the value of the cost function corresponding to the estimated values of M , T and d for each method. For each spectrum, the CatchAll estimate having the minimum error compared to M_0 is reported. P = best discounted parametric model produced by CatchAll. C = Chao1 non-parametric estimate. n/a denotes samples for which CatchAll failed to produce an output

use of existing data and the use of in vitro method, flow cytometry. However, mentioned alternative approaches pose limitations. The existing data on genome lengths of viruses are limited and their use to derive the viral genome length distributions may be inaccurate. Flow cytometry may be used to visualize the distribution of particle sizes of a sample of viruses [27] and may be used to infer the viral genome length distribution in vitro. However, conducting multiple experiments with flow cytometry is quite expensive, constrained by the limited availability of machines and no experiment has been conducted for this purpose as yet to the best of our knowledge.

Running Time Comparison

The running time of ENVirT is directly proportional to the number of genotypes in the population for given values of L_{LB} , L_{UB} , d_{LB} , d_{UB} and N_L . To compare the computational efficiency of the methods, we analyzed simulated contig spectra of populations with low-high richness having parameters: $M = [5000, 45000, 100000]$, $L = 50$ kbp, power law distribution and $d = 0.7$. The experiment specific parameters were: $R = 10000$, $r = 100$ bp and $o = 35$ bp. Input parameters to ENVirT were given as follows: $M_{LB} = 1$, $M_{UB} = 120000$, $L_{LB} = 15$ kbp, $L_{UB} = 75$ kbp,

$d_{LB} = 0.01$, $d_{UB} = 5$ and $N_L = 5$. Under these parameters, ENVirT consumed 35.57, 62.17 and 85.77 mins of wall clock time to analyze contig spectra of populations having richness of 5000, 45000 and 100000 respectively, over the four types of relative abundance distributions from 15 kbp to 75 kbp. The running time of PHACCS is directly proportional to the number of genotypes in the population. To analyze the same three contig spectra, PHACCS consumed 44.52, 150.8 and 167.6s of wall clock time respectively, over the four types of relative abundance distributions when it is provided with the correct average genome lengths (L) of the population. Therefore, to analyze contig spectra of populations with richness of 5000, 45000 and 100000 over the average genome lengths (L) from 15 kbp to 75 kbp to determine the best estimate for L , PHACCS would take 742, 2513.3 and 2793.3 h respectively. Hence, ENVirT would take only 0.04 - 0.08% of the time taken by an approach based on PHACCS to analyze a contig spectrum over a range of L to determine the best estimate for the average genome length of a contig spectrum without making unfair or biased predictions. These measurements were taken on a desktop computer running Windows 7 (64-bit) operating system on an Intel Core i7-4790 CPU @3.60 GHz with 16 GB RAM. The contig spectra were trimmed to a length of 50 before

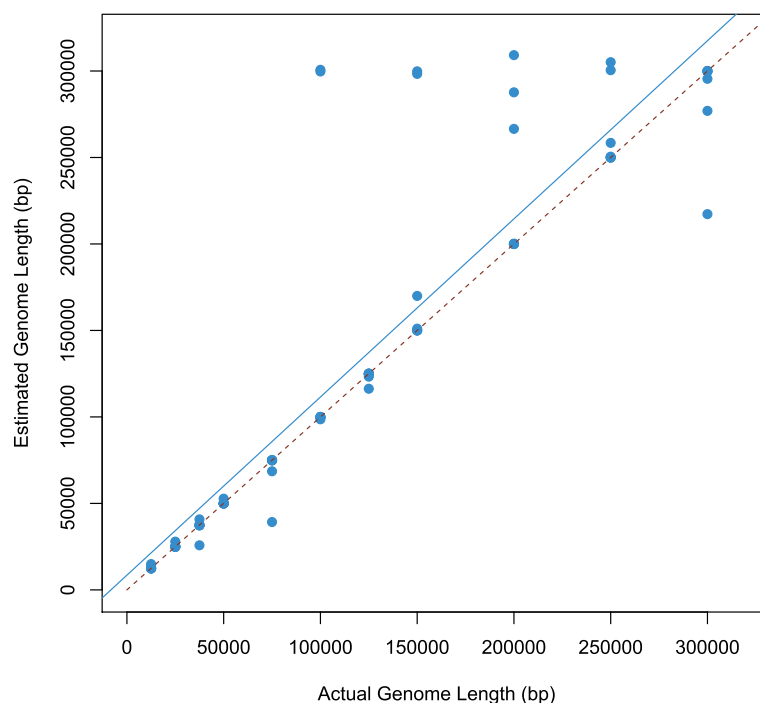


Fig. 1 Estimated average genome length versus true average genome length for ENVirT. This analysis uses only the information contained in the contig spectra for a given virome. As such, it is shown that ENVirT does not require underlying sequence data or other databases to estimate an average genome length. Here, ENVirT is able to estimate the true average genome length with an average error of 9.13%

analyzing using PHACCS, because PHACCS could not find the optimal results with the original contig spectra.

Analysis of 20 experimental viromes

To validate ENVirT and its applicability to experimental viromes, we repeated the analysis of 20 samples using ENVirT and compared the results with those obtained using PHACCS, as well as previously conducted in vitro analysis of each respective sample. As required by PHACCS, we provided the algorithm with either GAAS or BLAST-based estimates of genome length to estimate viral richness. We then compared this combination of PHACCS+GAAS/BLAST with ENVirT. We note that we have excluded CatchAll from all subsequent comparisons, since it has been shown to perform poorly on all our simulations. For data sets where contig spectra needs to be re-calculated, we used the contig spectrum generation software Circonspect (version 0.2.6, <https://sourceforge.net/projects/circonspect/>) with parameter settings as described in Online Methods. Figure S5 of Additional file 2 shows that the number of iterations used by Circonspect to generate the contig spectra is sufficient.

A summary of the parameter estimates produced by ENVirT and PHACCS is presented in Table 2 and depicted in Figure S6 of Additional file 2. As a measure of performance, we use the value of S_{min} (the minimum value of the cost function described in Eq. 7), which represents

the residual error of a generated model. The ideal value of S_{min} is 0, which corresponds to a perfect model of the observed virome data. We found that in all cases, ENVirT clearly outperforms PHACCS+GAAS/BLAST in terms of this performance metric (Figure S7 of Additional file 2).

Lake Bouget and Lake Pavin:

Average genome length values of 13,089bp and 12,274bp as reported by GAAS, were used in the analysis for Lake Bourget and Lake Pavin respectively. Using these genome length estimates, PHACCS estimated viral richness to be 33,331 and 2628, respectively. In contrast, ENVirT produced estimates of viral richness, with a much smaller model error, of 42,999 and 792 respectively. We used Circonspect to compute the contig spectra over 1000 iterations, where each iteration considered 10,000 reads (Figure S5 of Additional file 2). Moreover, we see a 5-fold difference in the genome length estimates produced by ENVirT (62.2 kbp and 81.1 kbp) relative to GAAS, which in turn, explains the difference in diversity estimates produced by PHACCS.

Fetsui Reservoir, North Taiwan:

The six reservoir samples, V1-V6 were sampled before and after the occurrence of typhoons over a 2-year period. As such, it is expected that the diversity estimates will vary in accordance with these seasonal disturbances,

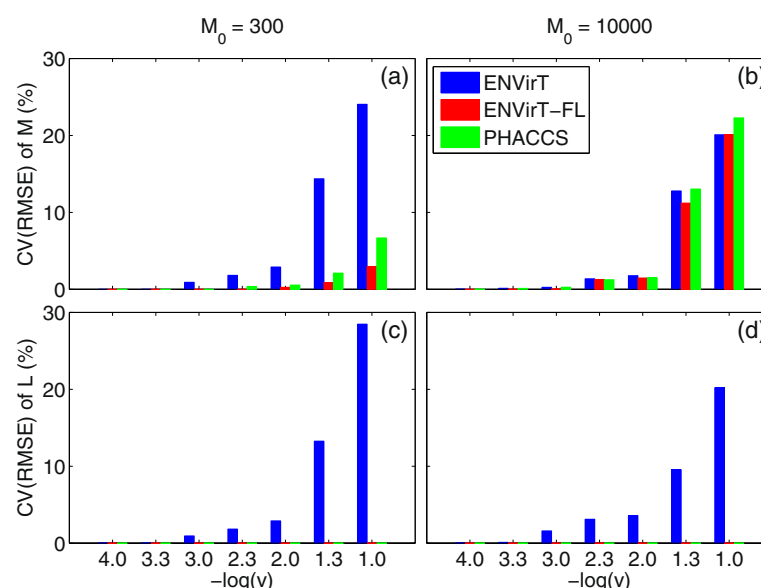


Fig. 2 **a** CV(RMSE) of estimated M when $M_0 = 300$, **b** CV(RMSE) of estimated M when $M_0 = 10000$, **c** CV(RMSE) of estimated L when $M_0 = 300$ and **d** CV(RMSE) of estimated L when $M_0 = 10000$: of spectra in Simulation Scenario 2 categorized under different values of v ($v \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$). CV(RMSE) = Coefficient of Variation of the Root Mean Squared Error, M_0 = simulated true richness, M = estimated richness and L = estimated average genome length. All 140 spectra used here were derived from populations simulated using a genome length distribution of $\mathcal{N}(L_0, (L_0 \cdot v)^2)$ and $L_0 = 50$ kbp. ENViT-FL = ENViT algorithm given a fixed value for L . Only the power-law distribution was considered in all three methods. Values summarized in the figure are given in Table S1 of Additional file 1)

as observed in the richness estimates of ENViT and PHACCS. We observed that GAAS produced much larger estimates of average genome length based on hits to similar viruses. ENViT produced a much tighter range of richness estimates (587–1929) over all samples, whereas PHACCS estimated a much broader range of richness (174–3059).

Fermented food:

A BLAST search was used to identify closely related viral types for each of the Shrimp, Kimchi and Sauerkraut samples to estimate the respective average genome lengths. This produced estimates of 39.8 kbp, 48.2 kbp and 36.4 kbp based on similar viruses. The relative differences between richness estimates between PHACCS and ENViT were 6.59% and 1.43% for the Shrimp and Kimchi samples, respectively. However, ENViT estimated the average genome length of the Sauerkraut sample to be 277.1 kbp, in contrast to the BLAST-based estimate of 36.5 kbp. This is reflected in ENViT's lower richness estimates. This critical difference is explained in the subsequent discussion.

Mauritanian Sahara:

Four perennial pond samples were analyzed. The Ilij, Hamdoun and El Berbera samples were in close agreement with previously reported richness estimates. However, of particular interest is the Molomhar sample, which

had a predicted richness of 223 by ENViT and 1318 by PHACCS. We also note that ENViT estimated a much larger average genome length of 394.92 kbp in comparison to the re-calculated GAAS estimate of 60.96 kbp. The much lower model error produced by ENViT suggests that it was better able to estimate a more representative viral richness.

Human gut:

An assumed average genome length value of 50 kbp was used for all human gut samples as per the recommendations of the original study [28]. The overall richness of all 5 samples was relatively lower than the other samples that were analyzed. However, there was a close agreement between the order of magnitude of the richness estimates produced by ENViT and PHACCS. However, the observed genome length estimates produced by ENViT clearly indicate the presence of much larger genomes (175.6 - 570.7 kbp).

Discussion

We have clearly demonstrated in the benchmark analysis using simulated data that ENViT can estimate viral richness with a higher accuracy and computational efficiency than PHACCS, despite providing additional information that advantaged the latter. To the best of our knowledge, we have also demonstrated for the first time that additional databases are not required to infer the average

genome length for an experimental sample. This is in contrast to the widely used PHACCS algorithm which relies on other methods, such as GAAS and BLAST, as well as other databases to analyze experimental data.

The formulation of ENVirT can be considered as an extension to the original model used by PHACCS but is still fundamentally different. ENVirT estimates thus are uncorrelated with the estimates generated by PHACCS, hence there is no systematic bias in our extended model compared to the original model. When benchmarked against simulated data, the proposed extensions allow ENVirT to accurately estimate an average genome length with an average error of 9.13%, while at the same time being 9.80-55.62% more accurate than PHACCS in the presence of genome length variation and up to 66.90% more accurate than PHACCS and CatchAll at varying levels of viral richness.

However, we did observe that there is a significant reduction in estimation accuracy for all methods when the evenness of a viral population approaches one (i.e. all viruses are equally abundant; $d = 0$; refer Eqs. 1 to 4). This translates to an optimization landscape for Eq. 7 that has multiple global minima, meaning that there are multiple equally valid solutions. To some degree, the proposed niching strategy of ENVirT is able to find a global minima that is close to the desired solution, but Additional file 1 shows that when evenness is equal to one, a single solution to the minimization of S (Eq. 7) does not exist.

Our analysis of 20 experimental viromes revealed unique insights into each of the underlying viral populations. In all cases, we found that the results produced by ENVirT were more consistent with the findings of each respective study than the results produced by PHACCS + GAAS/BLAST. For instance, a common observation among these analyses is that larger viral genomes were not considered when estimating richness using PHACCS + GAAS/BLAST. Our analyses show that this behaviour can skew richness estimates to the point where very different conclusions can be drawn from that data.

This is most notable in the Sauerkraut sample which contains viruses that have much larger genomes, including T4-like viruses, SOP1-like viruses and Mimiviruses, as identified using MEGAN [29]. This sample also contains many unclassified viruses, which we expect to be larger viruses. While ENVirT is able to correctly account for these viruses, PHACCS + GAAS/BLAST is unable to do so. Instead, it predicts a viral mixture containing much smaller viral genomes. This is then reflected in very different richness estimates.

Similar results were obtained for the Human Gut samples when analyzed using ENVirT. In this instance, larger estimates of average genome length could be indicative of host DNA contamination or gene transfer agents that had likely affected the samples [28, 30].

Bacterial species, *Mycoplasma* with larger genome lengths ($> 0.5 - 1$ Mb) may not be removed using a $0.2\mu\text{m}$ filter. In fact, unknown viruses had previously been excluded from downstream computational analyses, which could include much larger viruses [28] that could pass through the filter. Again these findings could be not observed using PHACCS+GAAS/BLAST. This also suggests the importance of a methodology to learn the variation of genome lengths of a virome which has not been addressed by ENVirT or PHACCS. Although the exact reason for the observation of average viral genome lengths larger than 500 kbp is unknown, the results suggest that large viruses might be more common in human gut vial assemblages than our current understanding.

Mesotrophic lakes, such as Lake Bourget, are expected to be much more nutrient rich than oligotrophic lakes like Lake Pavin and hence contain higher microbial and viral prevalence [6]. This hypothesis was confirmed by both ENVirT and PHACCS. However, ENVirT was better able to optimize a population model than PHACCS+GAAS/BLAST, suggesting that previous estimates did not capture the full extent of the relative diversity of both lakes.

The Feitsui Reservoir samples were collected based on the hypothesis that viral diversity increases after a typhoon [17]. The original study confirmed that terrestrial viruses infiltrate these marine communities, contributing to larger average genome lengths. This phenomena was corroborated by ENVirT but could not be fully explained using PHACCS + GAAS/BLAST. For example, a high proportion of *Mimiviridae* and *Phycodnaviridae* were detected in sample V6. These correspond to relatively large viral taxa, which should skew the average genome length to much larger values. This is true for ENVirT but not for PHACCS+GAAS/BLAST. Moreover, smaller viral taxa (*Circoviruses*, *Nanoviruses* or *Microviruses*) were detected in samples V2, V4 and V5, which is again in agreement with ENVirT estimates. As a result, we see that ENVirT is better able to optimize the corresponding population models at much lower residual error than PHACCS+GAAS/BLAST.

Previous results based on Transmission Electron Microscopy for the Molomhar samples had identified relatively large *Mimivirus*-like particles (300nm viral particles) [5]. This agrees with the average genome length estimates produced by ENVirT. Additional in vitro analysis of the Sahara and Namib samples revealed higher molecular weight DNA (270 - 350 kbp) than other samples, again confirming ENVirT's estimates of viruses with distinctively larger genome lengths in both samples. These larger viral genomes were not represented in the results produced by PHACCS+GAAS/BLAST.

The technique of Multiple Displacement Amplification (MDA) used in sample preparation prior to sequencing

may have introduced biases towards certain viral types which could affect the estimated values of species richness obtained by ENVirT and PHACCS [31]. In this study, metaviromic data sets from the French lakes, Feitsui reservoirs, Sahara desert and human gut were all amplified using MDA. In their current versions both PHACCS and ENVirT cannot rule out this bias. Owing to the improvement of sequencing such that the most recent platforms require much less DNA concentration, MDA would not be necessary in the preparation of viral DNA, and the bias will no longer be problematic in species richness estimation.

We note that ENVirT is not capable of inferring the variations in genome lengths of a given virome. There are several possible extensions to ENVirT that could alleviate this limitation and enhance its performance on experimental data. For example, a phage community could have several different discrete and dictating genome lengths such as 5 kbp, 50 kbp, 100 kbp and 200 kbp. Reformulating ENVirT's objective function to account for variations in genome length rather than assuming a point estimate could improve richness estimation accuracy over ENVirT in such scenarios. Section Simulation Scenario 2 shows how ENVirT and PHACCS deviate from expected estimates when such variability is present in the virome. Additional heuristics could also be implemented to reduce the computational cost of the algorithm. For instance, GA could be applied to a number of subspaces considerably less than $4N_L$ based on the features of the local optima found at the search space boundaries of $S(M, L, T, d)$. The suggested extensions shall also include a strategy to improve the estimation accuracy of L when $L > 100 \text{ kbp}$. It is also worth exploring how ENVirT can be improved to analyze contig spectra generated from recently introduced assemblers such as de-bruijn graph assemblers, to increase the applicability of ENVirT.

Conclusions

Estimating the parameters that describe a viral community underpins our ability to deeply understand viral ecology. In this regard, ENVirT is shown to be faster and more accurate than the most performant algorithm that has previously been developed on simulated benchmark datasets. Moreover, ENVirT does not rely on reference databases to estimate viral richness or an average genome length for novel experimental data. We have evaluated the performance of ENVirT on simulated data, highlighting its improvement and utility over existing methods. We have also demonstrated its validity in analyzing 20 experimental samples from a wide range of environments, revealing unique insights that were previously not observed. ENVirT is thus set to be an essential tool for studying viral ecology.

Additional files

Additional file 1: Supplementary Methods. Detailed description of ENVirT methodology. (PDF 822 kb)

Additional file 2: Supplementary Results. Supplementary figures and tables. (PDF 234 kb)

Funding

This work was supported partially by Australia Research Council [grant numbers LP140100670 and DP150103512] and the Biodiversity Research Center, Academia Sinica, Taiwan. DJ, DH, DS and YS were funded by the MFRS and MRS scholarships of The University of Melbourne. Publication costs were funded by The Australian National University.

Availability of data and materials

<https://github.com/senanayaked/ENVirT.git>. All simulated and experimental data can be provided on request. Correspondence regarding accessing, processing and analysis of experimental data used in this study may be directed to sltang@gate.sinica.edu.tw

About this supplement

This article has been published as part of *BMC Bioinformatics Volume 19 Supplement 13, 2018: 17th International Conference on Bioinformatics (InCoB 2018): bioinformatics*. The full contents of the supplement are available online at <https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-19-supplement-13>.

Authors' contributions

IS and SLT conceptualized the research project. DJ formulated and implemented the algorithm, generated initial simulated data and performed the analysis of simulated and real data. DH designed the extended simulation study. DH and DS performed the extended simulation study on ENVirT and other existing methods. DS developed the graphical user interface of ENVirT and manages the software distribution. YS and SKH contributed in formulating the optimization algorithm. CYY, IS, BC and SLT processed and contributed in the analysis of real data. IS and DJ prepared the initial draft of the manuscript. DH, DS, SLT and SKH contributed in preparing the final version of the manuscript. All authors read and approved the final manuscript.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

¹School of Public Health and Community Medicine, University of New South Wales, Randwick, NSW 2052, Australia. ²Optimisation and Pattern Recognition Research Group, Department of Mechanical Engineering, Melbourne School of Engineering, The University of Melbourne, Parkville, VIC 3010, Australia. ³Department of Computer Engineering, University of Peradeniya, Peradeniya, Sri Lanka. ⁴Biodiversity Research Center, Academia Sinica, Nan-Kang, Taipei 11529, Taiwan. ⁵Yourgene Bioscience, No. 376-5, Fuxing Rd., Shu-Lin District, New Taipei City, Taiwan. ⁶Research School of Engineering, College of Engineering and Computer Science, The Australian National University, Canberra, ACT 2601, Australia.

Received: 29 May 2018 Accepted: 25 September 2018

Published: 4 February 2019

References

- Edwards RA, Rohwer F. Viral metagenomics. *Nat Rev Microbiol*. 2005;3:504–10.
- Bunge J, Fitzpatrick M. Estimating the number of species - a review. *J Am Stat Assoc*. 1993;88(421):364–273.
- Bunge J. Estimating the number of species with CatchAll. In: *Biocomputing*. World Scientific; 2011. p. 121–30.
- Shannon C, Weaver W. *The Mathematical Theory of Communication*: Univ of Illinois Press, Urbana; 1963.
- Fancello L, Trape S, Robert C, Boyer M, Popgeorgiev N, Raoult D, Desnues C. Viruses in the desert: a metagenomic survey of viral communities in four perennial ponds of the mauritanian sahara. *ISME J*. 2013;7:359–69. <https://doi.org/10.1038/ismej.2012.101>.
- Roux S, Enault F, Robin A, Ravet V, Personnic S, Theil S, Colombet J, Sime-Ngando T, Debroas D. Assessing the diversity and specificity of two freshwater viral communities through metagenomics. *PLoS ONE*. 2012;7(3):33641. <https://doi.org/10.1371/journal.pone.0033641>.
- Hill M. Diversity and evenness: a unifying notation and its consequences. *Ecology*. 1973;54:427–31.
- Angly F, Rodriguez-Brito B, Bangor D, McNairnie P, Breitbart M, Salamon P, Felts B, Nulton J, Mahaffy J, Rohwer F. PhacCS, an online tool for estimating the structure and diversity of uncultured viral communities using metagenomic information. *BMC Bioinformatics*. 2005;6(1):41. <https://doi.org/10.1186/1471-2105-6-41>.
- Lander E, Waterman M. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*. 1988;2:231–9.
- Breitbart M, Salamon P, Andresen B, Mahaffy JM, Segall AM, Mead D, Azam F, Rohwer F. Genomic analysis of uncultured marine viral communities. *Proc Natl Acad Sci*. 2002;99(22):14250–5. <https://doi.org/10.1073/pnas.202488399>.
- Bunge J, Woodard L, Böhning D, Foster JA, Connolly S, Allen HK. Estimating population diversity with catchall. *Bioinformatics*. 2012;28(7):1045–7. <https://doi.org/10.1093/bioinformatics/bts075>.
- Edgar RC. Search and clustering orders of magnitude faster than blast. *Bioinformatics*. 2010;26(19):2460–1. <https://doi.org/10.1093/bioinformatics/btq461>.
- Reyes A, Haynes M, Hanson N, Angly F, Heath A, Rohwer F, Gordon J. Viruses in the faecal microbiota of monozygotic twins and their mothers. *Nature*. 2010;466:334–8. <https://doi.org/10.1038/nature09199>.
- Roux S, Faubladier M, Mahul A, Paulhe N, Bernard A, Debroas D, Enault F. Metavir: a web server dedicated to virome analysis. *Bioinformatics*. 2011;27(21):3074–5. <https://doi.org/10.1093/bioinformatics/btr519>.
- Allen H, Bunge J, Foster J, Bayles D, Stanton T. Estimation of viral richness from shotgun metagenomes using a frequency count approach. *Microbiome*. 2013;1(1):5. <https://doi.org/10.1186/2049-2618-1-5>.
- Aguirre de Carcer D, Angly F, Alcamí A. Evaluation of viral genome assembly and diversity estimation in deep metagenomes. *BMC Genomics*. 2014;15(1):989. <https://doi.org/10.1186/1471-2164-15-989>.
- Tseng C-H, Chiang P-W, Shiah F-K, Chen Y-L, Liou J-R, Hsu T-C, Maheswararajah S, Saeed I, Halgamuge S, Tang S-L. Microbial and viral metagenomes of a subtropical freshwater reservoir subject to climatic disturbances. *ISME J*. 2013;7(12):2374–86.
- Steward GF, Montiel JL, Azam F. Genome size distributions indicate variability and similarities among marine viral assemblages from diverse environments. *Limnol Oceanogr*. 2000;45(8):1697–706.
- Ferris MM, Yan X, Habberset RC, Shou Y, Lemanski CL, Jett JH, Yoshida TM, Marrone BL. Performance assessment of dna fragment sizing by high-sensitivity flow cytometry and pulsed-field gel electrophoresis. *J Clin Microbiol*. 2004;42(5):1965–76.
- Díez B, Antón J, Guixa-Boixereu N, Pedrós-Alió C, Rodríguez-Valera F. Pulsed-field gel electrophoresis analysis of virus assemblages present in a hypersaline environment. *Int Microbiol*. 2010;3(3):159–64.
- Prigent M, Leroy M, Confalonieri F, Dutertre M, DuBow MS. A diversity of bacteriophage forms and genomes can be isolated from the surface sands of the sahara desert. *Extremophiles*. 2005;9(4):289–96.
- Ulrich W. Ecological characteristic of rare species : the case of parasitic hymenoptera. *Pol J Ecol*. 2001;49(4):379–89.
- Sugihara G. Minimal community structure: an explanation of species abundance patterns. *Am Nat*. 1980;116(6):770–87.
- McGill B. A test of unified neutral theory of biodiversity. *Nature*. 2003;422:881–5.
- Mahfoud SW. Niching methods for genetic algorithms. Urbana. 1995;51(95001):62–94.
- Wagner J, Maksimovic J, Farries G, Sim WH, Bishop RF, Cameron DJ, Catto-Smith AG, Kirkwood CD. Bacteriophages in gut samples from pediatric crohn's disease patients: metagenomic analysis using 454 pyrosequencing. *Inflamm Bowel Dis*. 2013;19(8):1598–608.
- Ma L, Zhu S, Tian Y, Zhang W, Wang S, Chen C, Wu L, Yan X. Label-free analysis of single viruses with a resolution comparable to that of electron microscopy and the throughput of flow cytometry. *Angew Chem*. 2016;128(35):10395–9.
- Minot S, Sinha R, Chen J, Li H, Keilbaugh SA, Wu GD, Lewis JD, Bushman FD. The human gut virome: inter-individual variation and dynamic response to diet. *Genome Res*. 2011;21(10):1616–25.
- Huson DH, Auch AF, Qi J, Schuster SC. Megan analysis of metagenomic data. *Genome Res*. 2007;17(3):377–86.
- Lang AS, Zhaxybayeva O, Beatty JT. Gene transfer agents: phage-like elements of genetic exchange. *Nat Rev Microbiol*. 2012;10(7):472–82.
- Kim K-H, Bae J-W. Amplification methods bias metagenomic libraries of uncultured single-stranded and double-stranded dna viruses. *Appl Environ Microbiol*. 2011;77(21):7663–8.
- Park E-J, Kim K-H, Abell GC, Kim M-S, Roh SW, Bae J-W. Metagenomic analysis of the viral communities in fermented foods. *Appl Environ Microbiol*. 2011;77(4):1284–91.
- Angly FE, Willner D, Prieto-Davó A, Edwards RA, Schmieder R, Vega-Thurber R, Antonopoulos DA, Barott K, Cottrell MT, Desnues C, Dinsdale EA, Furlan M, Haynes M, Henn MR, Hu Y, Kirchman DL, McDole T, McPherson JD, Meyer F, Miller RM, Mundt E, Naviaux RK, Rodriguez-Mueller B, Stevens R, Wegley L, Zhang L, Zhu B, Rohwer F. The gaas metagenomic tool and its estimations of viral and microbial average genome size in four major biomes. *PLoS Comput Biol*. 2009;5(12):1000593. <https://doi.org/10.1371/journal.pcbi.1000593>.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions



Bibliography

- [1] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
- [2] Spectral embedding, Oct 2019. URL https://networkx.github.io/documentation/stable/auto_examples/drawing/plot_spectral_grid.html.
- [3] Michael Thomas Wong, David Eng Hui Ong, Frances Sheau Huei Lim, Karen Wei Weng Teng, Naomi McGovern, Sriram Narayanan, Wen Qi Ho, Daniela Cerny, Henry Kun Kiaang Tan, 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.
- [4] D Swain, MM Ali, and Robert A Weller. Estimation of mixed-layer depth from surface parameters. *Journal of Marine Research*, 64(5):745–758, 2006.
- [5] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.
- [6] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in neural information processing systems*, pages 625–632, 1995.
- [7] Jinguo Chen, Foo Cheung, Rongye Shi, Huizhi Zhou, Wenrui Lu, CHI Consortium, et al. Pbm fixation and processing for chromium single-cell rna sequencing. *Journal of translational medicine*, 16(1):198, 2018.
- [8] Koen Van den Berge, Hector Roux De Bezieux, Kelly Street, Wouter Saelens, Robrecht Cannoodt, Yvan Saeys, Sandrine Dudoit, and Lieven Clement. Trajectory-based differential expression analysis for single-cell sequencing data. *Nature communications*, 11(1):1–13, 2020.
- [9] Rohan Lowe, Neil Shirley, Mark Bleackley, Stephen Dolan, and Thomas Shafee. Transcriptomics technologies. *PLoS computational biology*, 13(5), 2017.

- [10] Maico Cassel and F Lima. Evaluating one-hot encoding finite state machines for seu reliability in sram-based fpgas. In *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, pages 6–pp. IEEE, 2006.
- [11] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106, 2003.
- [12] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.
- [13] Charles Van Loan. *Computational frameworks for the fast Fourier transform*, volume 10. Siam, 1992.
- [14] Paul F Velleman and David C Hoaglin. *Applications, basics, and computing of exploratory data analysis*. Duxbury Press, 1981.
- [15] Jerome H Friedman and John W Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on computers*, 100(9):881–890, 1974.
- [16] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
- [17] Cem Ornek and Elif Vural. Nonlinear supervised dimensionality reduction via smooth regular embeddings. *Pattern Recognition*, 87:55–66, 2017.
- [18] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.
- [19] Joseph B Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.
- [20] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [21] Richard E Bellman. *Adaptive control processes: a guided tour*, volume 2045. Princeton university press, 2015.
- [22] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.
- [23] Victor Guillemin and Alan Pollack. *Differential topology*, volume 370. American Mathematical Soc., 2010.
- [24] Morris W Hirsch. *Differential topology*, volume 33. Springer Science & Business Media, 2012.

- [25] James Munkres. *Topology*. Pearson Education, 2014.
- [26] Ye Qingwei, Sun Yang, Wang Xiaodong, and Zhou Yu. An improved lle algorithm with sparse constraint. *Journal of Computational and Theoretical Nanoscience*, 10(12):2872–2876, 2013.
- [27] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
- [28] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
- [29] Jarkko Venna and Samuel Kaski. Local multidimensional scaling. *Neural Networks*, 19(6):889–899, 2006.
- [30] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
- [31] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.
- [32] Fevzi Alimoglu, Ethem Alpaydin, and Yagmur Denizhan. Combining multiple classifiers for pen-based handwritten digit recognition. 1996.
- [33] Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
- [34] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel 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.
- [35] Etienne Becht, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Evaluation of umap as an alternative to t-sne for single-cell data. *BioRxiv*, page 298430, 2018.
- [36] Tomas 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.
- [37] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

- [38] Peter Fromherz, Carsten O Müller, and Rolf Weis. Neuron transistor: electrical transfer function measured by the patch-clamp technique. *Physical review letters*, 71(24):4079, 1993.
- [39] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [40] Andreas Zell. *Simulation neuronaler netze*, volume 1. Addison-Wesley Bonn, 1994.
- [41] J David Schaffer, Darrell Whitley, and Larry J Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37. IEEE, 1992.
- [42] Bart Kosko. Differential hebbian learning. In *AIP Conference proceedings*, volume 151, pages 277–282. AIP, 1986.
- [43] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [44] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.
- [45] Giorgio Gnecco and Marcello Sanguineti. The weight-decay technique in learning from data: An optimization point of view. *Computational Management Science*, 6(1):53–79, 2009.
- [46] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.
- [47] Damien Drix, Verena Vanessa Hafner, and Michael Schmuker. Sparse coding with a somato-dendritic rule. *BioRxiv*, page 451955, 2018.
- [48] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.
- [49] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- [50] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.
- [51] Donald Olding Hebb and DO Hebb. *The organization of behavior*, volume 65. Wiley New York, 1949.
- [52] Erkki Oja. Principal components, minor components, and linear neural networks. *Neural networks*, 5(6):927–935, 1992.

- [53] Thomas Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In *International conference on artificial neural networks*, pages 427–434. Springer, 1993.
- [54] Thomas Martinetz, Klaus Schulten, et al. A” neural-gas” network learns topologies. 1991.
- [55] Bernd Fritzke. Growing self-organizing networks-why? In *ESANN*, volume 96, pages 61–72. Citeseer, 1996.
- [56] Daminda Alahakoon, Saman K Halgamuge, and Bala Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on neural networks*, 11(3):601–614, 2000.
- [57] Pablo A Estévez, Cristián J Figueroa, and Kazumi Saito. Cross-entropy embedding of high-dimensional data using the neural gas model. *Neural Networks*, 18(5-6):727–737, 2005.
- [58] Ehud Shapiro, Tamir Biezuner, and Sten Linnarsson. Single-cell sequencing-based technologies will revolutionize whole-organism science. *Nature Reviews Genetics*, 14(9):618–630, 2013.
- [59] Tomer Kalisky and Stephen R Quake. Single-cell genomics. *Nature methods*, 8(4):311–314, 2011.
- [60] Samuel Marguerat and Jürg Bähler. Rna-seq: from technology to biology. *Cellular and molecular life sciences*, 67(4):569–579, 2010.
- [61] Stuart H Orkin and Leonard I Zon. Hematopoiesis: an evolving paradigm for stem cell biology. *Cell*, 132(4):631–644, 2008.
- [62] Amit Grover, Alejandra Sanjuan-Pla, Supat Thongjuea, Joana Carrelha, Alice Giustacchini, Adriana Gambardella, Iain Macaulay, Elena Mancini, Tiago C Luis, Adam Mead, et al. Single-cell rna sequencing reveals molecular and functional platelet bias of aged haematopoietic stem cells. *Nature communications*, 7:11075, 2016.
- [63] Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. Computational methods for trajectory inference from single-cell transcriptomics. *European journal of immunology*, 46(11):2496–2506, 2016.
- [64] Caleb Weinreb, Samuel Wolock, Betsabeh K Tusi, Merav Socolovsky, and Allon M Klein. Fundamental limits on dynamic inference from single-cell snapshots. *Proceedings of the National Academy of Sciences*, 115(10):E2467–E2476, 2018.

- [65] F Alexander Wolf, Philipp Angerer, and Fabian J Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19(1):15, 2018.
- [66] Kevin R Moon, David van Dijk, Zheng Wang, William Chen, Matthew J Hirn, Ronald R Coifman, Natalia B Ivanova, Guy Wolf, and Smita Krishnaswamy. Phate: a dimensionality reduction method for visualizing trajectory structures in high-dimensional biological data. *bioRxiv*, page 120378, 2017.
- [67] Michael B Elowitz, Arnold J Levine, Eric D Siggia, and Peter S Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, 2002.
- [68] GK Smyth. Stat appl genet mol biol. linear models and empirical bayes methods for assessing differential expression in microarray experiments, 2004.
- [69] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010.
- [70] Simon Anders and Wolfgang Huber. Differential expression of rna-seq data at the gene level—the deseq package. *Heidelberg, Germany: European Molecular Biology Laboratory (EMBL)*, 2012.
- [71] Tianyu Wang, Boyang Li, Craig E Nelson, and Sheida Nabavi. Comparative analysis of differential gene expression analysis tools for single-cell rna sequencing data. *BMC bioinformatics*, 20(1):40, 2019.
- [72] Spyros Darmanis, Steven A Sloan, Ye Zhang, Martin Enge, Christine Caneda, Lawrence M Shuer, Melanie G Hayden Gephart, Ben A Barres, and Stephen R Quake. A survey of human brain transcriptome diversity at the single cell level. *Proceedings of the National Academy of Sciences*, 112(23):7285–7290, 2015.
- [73] Minzhe Guo, Hui Wang, S Steven Potter, Jeffrey A Whitsett, and Yan Xu. Sincera: a pipeline for single-cell rna-seq profiling analysis. *PLoS computational biology*, 11(11):e1004575, 2015.
- [74] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550, 2014.
- [75] Zhun Miao, Ke Deng, Xiaowo Wang, and Xuegong Zhang. Desingle for detecting three types of differential expression in single-cell rna-seq data. *Bioinformatics*, 34(18):3223–3224, 2018.

- [76] Gökçen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell rna-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):390, 2019.
- [77] Yifei Chen, Yi Li, Rajiv Narayan, Aravind Subramanian, and Xiaohui Xie. Gene expression inference with deep learning. *Bioinformatics*, 32(12):1832–1839, 2016.
- [78] Michael KK Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.
- [79] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.
- [80] Felix Buggenthin, Florian Buettner, Philipp S Hoppe, Max Endeke, Manuel Kroiss, Michael Strasser, Michael Schwarzfischer, Dirk Loeffler, Konstantinos D Kokkaliaris, Oliver Hilsenbeck, et al. Prospective identification of hematopoietic lineage choice by deep learning. *Nature methods*, 14(4):403, 2017.
- [81] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [82] Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [83] Gökçen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.
- [84] Padideh Danaee, Reza Ghaeini, and David A Hendrix. A deep learning approach for cancer detection and relevant gene identification. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 219–229. World Scientific, 2017.
- [85] Duleepa Jayasundara, Damayanthi Herath, Damith Senanayake, Isaam Saeed, Cheng-Yu Yang, Yuan Sun, Bill C Chang, Sen-Lin Tang, and Saman K Halgamuge. Envirt: inference of ecological characteristics of viruses from metagenomic data. *BMC bioinformatics*, 19(13):377, 2019.