Department of Computer Science
Computer Graphics and Visualization Research Group

DimenFix-t-SNE: Fixed-dimension t-SNE for Visual Cluster Analysis

Master Thesis

Zixuan Han

Supervisors:
Thomas Höllt, TU Delft, Thesis Advisor
Jesse Krijthe, TU Delft
Fernando Paulovich, TU Eindhoven

FINAL version

Delft, July 2025

Abstract

Dimensionality reduction (DR) algorithms have been proven useful in various tasks when it comes to exploring high-dimensional data. Being one of the most used DR techniques, t-SNE is often valued for its ability to map nonlinear manifolds and preserving local structures. However, t-SNE produces anonymous axes which lack interpretability. To enhance the interpretability of t-SNE, this work proposes DimenFix-t-SNE, an extension of the existing meta-strategy DimenFix. DimenFix-t-SNE implements DimenFix using t-SNE, explicitly preserving the selected input feature in one of t-SNE's output dimensions. We generalize upon DimenFix by introducing individual ranges for every point and allowing more user control on the "pushing" process. For fixing nominal features in particular, we develop a new pushing mode Rescale, as well as an ordering method for the feature along the fixed axis. The extended method also supports traceable switching between features on the fixed axis, adding to the interactive component. Finally, we implement and test the algorithm as a plugin for ManiVault, providing an interactive and explainable visual analytics tool for high-dimensional data.

Preface

So long, and thanks for all the fish.

First and foremost, a huge thank you to my supervisor, Thomas Höllt, for his invaluable supervision throughout this project. Thank you for offering the opportunity to work on this project, and for the time and effort you put in to provide guidance and feedback during our weekly meetings. I have learned and experienced a great deal about research during this period, none of which would have been possible without the encouragement and support you have offered. I am also especially grateful for the opportunity to publish a research paper based on this work.

I would like to thank my external supervisor, Fernando Paulovich, whose earlier work laid the foundation for my project. His suggestions were a constant source of inspiration, and I feel fortunate to have collaborated with him on publishing the research paper. I would also like to thank Jesse Krijthe for kindly agreeing to be a member of my thesis committee and for the time he has dedicated to this role.

I would like to thank all the Ph.D. students in the Computer Graphics and Visualization research group for their help, advice, and the friendliness they extended. I am especially grateful to Soumyadeep Basu for his support with research nuances and for generously sharing datasets and other valuable information.

Finally, I want to thank my family and friends for their emotional support and belief in me throughout this process. Although some are separated by distance, their support has been omnipresent throughout this journey (and thank you, Internet, for making that possible). And to those who shared moments and laughters with me in person, your presence has meant more than words can express.

Zixuan Han
Delft, July 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visualization and interpretation of high-dimensional data is an important task in various domains, ranging from health and biology to astronomy and physics, and encompassing fields like chemistry and social science [8]. Due to limitations of human perception, the challenge of visualizing high-dimensional data is reducing the dimensionality of the input data, making it depictable and comprehensible [10]. Dimensionality reduction (DR) methods are applied to map the high-dimensional dataset to a two- or three-dimensional embedding which can be displayed in a scatterplot [25]. DR algorithms produce a low-dimensional embedding while preserving important aspects of the original data, for example, clusters. The transformation enables analysts to visually explore high-dimensional datasets.

DR algorithms can be categorized into projection-based methods and manifold learning [10]. Projective techniques reduce dimensionality by projecting high-dimensional data onto a lower-dimensional space using linear transformations. Examples include Principal Components Analysis (PCA) [17] and metric Multidimensional Scaling (MDS) [19]. On the other hand, manifold learning methods depict relationships of data points on non-linear manifolds in an unsupervised manner. The t-distributed Stochastic Neighbor Embedding (t-SNE) [25] is one of the representative algorithms in manifold learning. In the context of applying DR techniques in visual cluster analysis, t-SNE is also one of the most used methods [42].

Preserving local relations and displaying robust performance on clustering separation, t-SNE is a well suited method for visualizing data with nonlinear structures [3]. However, being a non-linear DR algorithm, embeddings generated using t-SNE have anonymous axes which are not interpretable. Various methods are proposed to explain how different input features contribute to the final layout, including local interpretations of dimensions [5], interactive correlation inspection [7] and meaningful distance metrics [12]. Being effective in enhancing the explainability of t-SNE embeddings, these methods either require domain-specific knowledge or are post-hoc interpretations which do not preserve specific values or order of values in their output embeddings.

Class labels and other meta-information are also integrated into some t-SNE processes for interpretability. Meta-information is used either to steer the embedding process by providing semantic structures and optimized class landmarks [9, 39, 26], or to discount prior knowledge and uncover complementary structures [12]. Together, these methods demonstrate the versatility of supervised DR in extracting structure using meta-information.

Taking a step further, Luo et al. [24] proposed DimenFix, a meta-DR strategy that preserves a selected feature or external labels from input data explicitly in the output embedding, while not heavily impacting neighborhood preservation and local distances. This strategy enables understanding of the data distribution with respect to the fixed feature. Initially employed to adapt with Force-Scheme [35] , the strategy can be integrated into any gradient-descent DR method.

Despite promising results produced with the Force-Scheme based implementation of DimenFix, limitations remain. One major shortcoming is that when fixing class labels on one axis, preservation of pairwise distances decreases. This is caused by assuming a random order for the nominal attribute – class labels – which decreased the quality of the final layout. The method also applies uniform ranges for all data points when moving them along the fixed axis, disregarding class density or any user preference. Moreover, in the context of interaction, when switching from one axis to another, recomputation instead of continuation happens, making it hard to keep track of the movement of individual points, clusters, or salient groups under attention.

This paper introduces DimenFix-t-SNE, an extended t-SNE version based on DimenFix. In addition to implementing the t-SNE algorithm adapted to DimenFix, we propose a method to order the nominal features in a meaningful manner. When the fixed axis represents a nominal attribute, either class labels or any nominal feature in the input, the attribute is reordered during the gradient-descent iterations while preserving neighborhood and local distances. User-steerable parameters are implemented, such as individual moving ranges for each data instance, to allow users to interact and explore the data structure more freely. Furthermore, a swift axis switching method is developed, so that when an initial embedding fixing feature A is generated, it would take less time to switch to a new embedding in which feature B is the fixed axis. In addition, the switching progress is made more traceable by adding intermediate iterations, showing how points move gradually towards their designated positions. Finally, the algorithm is implemented as a plugin for Manivault [38], a visual analytics framework for high-dimensional data. Our evaluations prove that DimenFix-t-SNE produces clearer and more helpful embeddings regarding specific visual cluster related tasks, and enhances intrinsic interpretability for t-SNE embeddings in a generalizable manner. We published a research paper based on this work in *Computers & Graphics*, Volume 130, August 2025, Article 104231. The paper is available at: https://doi.org/10.1016/j.cag.2025.104231.

The following chapters of this paper are structured as follows. Background knowledge on t-SNE and Dimenfix are provided in Chapter 2. Related works which addressed interpreting t-SNE embeddings and class-aware t-SNE variants can be found in Chapter 3. The Dimenfix-t-SNE algorithm is introduced in Chapter 4, including class ordering and swift axis switching methods. In Chapter 5, evaluation designs and results are displayed. Finally, Chapter 6 concludes the work, discusses the findings, and proposes possibilities for future research.

Chapter 2

Background

In this chapter, we provide the necessary background for understanding our approach. Being one of the most used DR methods for visualizing high-dimensional data, t-SNE is discussed in detail in Section 2.1. The meta-strategy DimenFix, on which this work is based, will be described in Section 2.2. The algorithm proposed in this work is implemented using HDILib, a library for the scalable analysis of large and high-dimensional data, and published as a plugin of an interactive visual analytics framework Manivault. Both will be introduced in Section 2.3.

2.1 t-SNE

The t-distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reduction technique extended from Stochastic Neighbor Embedding (SNE) [14]. t-SNE converts a high-dimensional dataset into a matrix of pairwise similarities and visualizes the resulting similarity data. The technique captures both local structures in high-dimensional data and global structures such as clusters.

Three steps can be identified in a t-SNE process: similarity calculation, gradient descent and updating the embedding. We denote X as the high-dimensional input data matrix, Y as the low-dimensional embedding.

After initialization, t-SNE starts by converting high-dimensional Euclidean distances between data points into similarities. For data points x_i and x_j $(i \neq j)$ in X, their similarity is the conditional probability that x_i would pick x_j as its neighbor, when the selection is based on the probability density of x_j under a Gaussian distribution centered at x_i, with a standard deviation of σ_i. When $i = j$, the probability is set to 0. Conditional probability $p_{i|j}$ of x_i picking x_j as its neighbor is

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}.$$

A hyperparameter *perplexity* is used to determine σ_i. A smaller σ_i is usually expected in dense regions than in sparse regions in the high dimensional space. σ_i induces a probability distribution P_i over all other data points, with an entropy that increases as σ_i does. A binary search is performed to determine σ_i that produces a P_i with the user defined *perplexity. Perplexity* measures the effective number of neighbors and is defined as

$$Perp(P_i) = 2^{H(P_i)},$$

where $H(P_i)$ is the Shannon entropy of P_i

$$H(P_i) = -\sum_j p_{j|i} log_2 p_{j|i}.$$

Since σ_i is different between data points, the conditional probability matrix is not symmetric. To ensure that each data point has a significant effect on the cost function, a normalized symmetric joint probability p_{ij} is defined as

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}.$$

This produces a symmetric similarity matrix P for the high-dimensional data.

Similar to the probabilities in the high dimension, the calculation for low-dimensional similarity matrix Q is

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

The only difference between p_{ij} and q_{ij} is that q_{ij} is calculated using a Student t-distribution with a single degree of freedom.

t-SNE aims to minimize the divergence between the two distributions P and Q. This is done through a gradient descent process. The Kullback-Leibler divergence [20] is used as the cost function in this process, defined as

$$KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Based on the Kullback-Leibler divergence, the gradient can be derived by

$$\frac{\partial KL(P \parallel Q)}{\partial y_i} = 4 \sum_{j \neq i} \left(p_{ij} - q_{ij}\right) q_{ij} \left(y_i - y_j\right).$$

The gradient can also be represented with a pair of attracting and repulsing forces:

$$\frac{\partial KL(P \parallel Q)}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} q_{ij} \left(y_i - y_j\right) - q_{ij}^2 \left(y_i - y_j\right)). \tag{2.1}$$

This annotation indicates that when points i and j are placed closer to each other in the low dimensional space than they are in the high dimensional space, the repulsive force (second term in Equation 2.1) becomes larger to drive them farther apart, and vise versa.

Given the gradient, the embedding can be updated as

$$\gamma^{(t)} = \gamma^{(t-1)} + \eta \frac{\partial KL(P \parallel Q)}{\partial \gamma^{(t-1)}} + \alpha^{(t)} (\gamma^{(t-1)} - \gamma^{(t-2)}).$$

Here, $\gamma^{(t)}$ stands for the embedding at iteration t and η the learning rate. A momentum $\alpha^{(t)}(\gamma^{(t-1)} - \gamma^{(t-2)})$ is added to the gradient, representing an exponentially decaying sum of previous gradients. The momentum reduces the number of total iteration required and helps avoid poor local minima.

To better explore the possible global structures of the data, two optimization methods are proposed. Both of them happen during the first few iterations of the gradient descent.

The first method is early compression. A L2-penalty is added to the cost function proportional to the sum of squared distanced of the embedded points from the origin. The modified cost function is

$$C = KL(P \parallel Q) + \lambda \sum_i \|Y_i\|^2.$$

This additional term penalizes large distances from the origin and helps to keep the embeddings centered and prevents excessive spreading.

The second optimization method is early exaggeration. All p_{ij}'s are multiplied by a hyper-parameter α and Q remains unchanged. The cost function with early exaggeration is

$$KL(P \parallel Q) = \sum_i \sum_{j \neq i} \alpha p_{ij} \log \frac{\alpha p_{ij}}{q_{ij}}.$$

While q_{ij}'s still add up to 1, most of them are too small to model the high dimensional p_{ij}'s. So during the gradient descent, the focus would be to model larger p_{ij}'s with fairly large q_{ij}'s. This results in forming tight yet widely separated clusters in the embedding, which corresponds to natural clusters in the high dimensional data. Early exaggeration also creates empty space between clusters for them to move around to reveal the global data structure.

The magnitude λ in the penalty term of early compression, the exaggeration rate α in early exaggeration and the iteration at which they are removed are set by hand, but experiments showed that the behavior is fairly robust across different parameters.

2.2 DimenFix

Dimenfix [24] is a meta-DR strategy for feature preservation. Aiming to explicitly represent the contribution of features to the embedding produced using nonlinear DR techniques, DimenFix fixes the values of a chosen feature or external labels in one of the embedded dimensions.

DimenFix can be adapted to any gradient-descent-like method. We denote X as the high-dimensional input data matrix, and Y as the low-dimensional output embedding. Choosing X^j as the feature to be preserved (fixed) in output Y^1, the low-dimension embedding is initialized randomly, except for the fixed axis, which will be initialized as the fixed feature X^j. A fixing step is introduced between gradient descent and updating the embedding in each iteration. During one iteration, after gradient descent, a new embedding $\hat{Y}$ is produced. However, instead of updating the output embedding Y by $Y = \hat{Y}$ as in a regular Force-Scheme process, Y^1 is set to be equal to the preserved (fixed) feature X^j, as indicated in line 7 of Algorithm 1.

The essence of DimenFix lies in the "fixing" step, for it explicitly preserves input X^j in the output embedding Y. Two modes are proposed for this moving step: the Strictly Fixed Mode (Section 2.2.1) and the Moving-In-Range (Section 2.2.2) Mode.

2.2.1 Strictly Fixed Mode

In the Strictly Fixed Mode, points are only allowed to move freely on axes other than Y^1, while Y^1 is fixed to resemble the chosen preserved feature X^j. In line 7 of Algorithm 1, we apply DimenFix using the Strictly Fixed Mode, using the exact value of the chosen feature to update the fixed axis.

2.2.2 Move-In-Range Mode

The Moving-In-Range Mode allows points to move along Y^1 within a uniform, user-defined range. Slightly different from what's shown in Algorithm 1, the Moving-In-Range Mode gives each point a range in which it can move along the fixed axis Y^1. Two moving strategies are further implemented for this Mode, the Uniform mode and the Gaussian mode. Let $[-\alpha, \alpha]$ be the moving range set for every data point, and $\hat{Y}^1$ be the new calculated value vector for the fixed dimension in the embedding.

Uniform Mode

In Uniform mode, substituting line 7 of Algorithm 1, the fixed dimension values y_s^1, $1 \leq s \leq N$ are updated as

Algorithm 1: Force-Scheme Implementation of DimenFix, adapted from Luo et al. [24]

```
Input: $X$: original dataset, $X^j$: original feature to be preserved, $\Delta$: learning rate, max:
       maximum number of iterations
Output: $Y$: final embedding
1 Initialize $y_i^1 = x_i^j$, $y_i^k = \beta$, $1 \le i \le N$, $2 \le k \le n$, where $\beta$ is a random number in $[0, 1]$;
2 while $it < max$ do
3     foreach $x_r \in X$ do
4         foreach $x_s \neq x_r \in X$ do
5             $\vec{v} = y_s - y_r$;
6             $y_s = \frac{(\delta(x_r,x_s)-d(y_r,y_s))\cdot|\delta(x_r,x_s)-d(y_r,y_s)|}{\Delta} \cdot \frac{\vec{v}}{\|\vec{v}\|}$;
7         $\boxed{y_s^1 = x_s^j;}$ – DimenFix step: Strictly Fixed Mode
8     $it++$;
```

$$y_s^1 = \begin{cases} x_s^j - \alpha, & \hat{y}_s^1 < x_s^j - \alpha, \\ x_s^j + \alpha, & \hat{y}_s^1 > x_s^j + \alpha, \\ \hat{y}_s^1 & otherwise. \end{cases}$$

This ensures that points moving outside the predefined ranges during gradient descent are "pushed" back.

Gaussian Mode

The Uniform mode makes the movements of points inconsistent between the fixed dimension and other dimensions. For better results, the Gaussian mode scales the movement of points using a Gaussian function, allowing points to move beyond the range limits in the fixed dimension. In this mode, movement along the fixed axis is defined as

$$y_s^1 = x_s^j + (sign(x_s^j - \hat{y}_s^1) \times d_s),$$

replacing line 7 of Algorithm 1. To calculate the moving distance d_s, the Gaussian weighing function centered at $x = 0$ is defined as

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{x^2}{2\sigma^2}). \tag{2.2}$$

With $x = x^j - y^1$, Equation 2.2 defines a fraction used to scale the distance between the original and current position of the fixed dimension. Calculating σ with the user-defined confidence interval CI and moving range α, $f(x)$ is further adapted to weigh how much movement should actually be applied to the fixed dimension, resulting in a moving distance as

$$d_s = exp(-\frac{(x_s^j - \hat{y}_s^1)^2}{2\sigma^2}).$$

Points are moved accordingly after the distance calculation, and this moving process happens in each iteration.

2.3 HDILib and ManiVault

The High Dimensional Inspector Library (HDILib) [28, 29] provides scalable analysis for large and high-dimensional data. Implemented in C++, OpenGL and JavaScript, it contains scalable manifold-learning algorithms, visualizations as well as visual-analytics frameworks, and has been used by several visual-analytics applications. Apart from a regular CPU-based t-SNE implementation, HDILib also contains a GPGPU linear complexity t-SNE optimization [30], enhancing interactivity and performance on large datasets. The GPU-based implementation approximates the repulsive forces between data points using adaptive-resolution textures, significantly reducing the computation time of the t-SNE optimization.

Developed by Vieth et al. [38], ManiVault is an open-source framework for visual analytics of high-dimensional data. The main purpose of ManiVault is to streamline the visual analytics workflow development for analysing complex high-dimensional datasets. ManiVault has a modular, plugin-based architecture, which makes the framework easily extensible, facilitating rapid prototyping. Plugins serving various purposes, from data loading to visualization and analytics are open-source accessible. Modules can be integrated using ManiVault's messaging API, bringing more efficiency into the workflow construction process.

A t-SNE-Analysis plugin [34] is developed using HDILib's t-SNE implementation. Several customizable settings are included, such as exaggeration factors and initialization methods, to enhance flexibility. Moreover, both CPU and GPU versions are supported in the plugin, facilitating faster calculation and smoother user interaction. The DimenFix-t-SNE algorithm proposed in this work is developed extending upon this plugin and the HDILib t-SNE implementation (GPGPU version).

Chapter 3

Related Works

Being a manifold learning DR technique, t-SNE produces embeddings with anonymous axes which lack semantic meanings. Moreover, the basic t-SNE technique can not take into account meta information such as ranking or class labels. As visual analytics develop, methods have been proposed aiming to provide explanations for t-SNE embeddings, which will be covered in Section 3.1. Further on, supervised t-SNE variants which utilize class label information are developed. Section 3.2 describes works that leverage class label information during the t-SNE process. Finally, we discuss how t-SNE and its variants are often applied in the context of visual cluster analysis tasks in Section 3.3.

3.1 Explaining t-SNE embeddings

For DR techniques, "explaining a DR result/method" often means presenting textual or visual artifacts that provide qualitative understanding of the relationship between the input and the model's output [33]. In this text, we use the terms explainablity and interpretability interchangeably. Methods proposed to explain the layout produced by t-SNE can be generally categorized as providing intrinsic explainability or post-hoc explainability. Molnar [27] provided the following definition and examples for this categorization: Intrinsic explainability can be found in most sparse linear techniques, due to their relatively simple structure. In method designs, intrinsic explainability is usually achieved by restricting the model's complexity before training, or in the case of DR, before learning the embedding. Post-hoc methods treat the model as a black-box, and try to analyse the output after training without modifying the model. Summary statistics on feature importance, visualization based analytics and explanation of the selected data instance are common methods of enhancing post-hoc explainability.

Appearing more in practice, post-hoc explainability often applies to more general approaches. LIME [33] is a flexible post-hoc explanation technique, which provides explanation for the predictions of any classifier by learning a local interpretable model around the prediction. The explanations can be utilized in various scenarios that require trust. Designed to explain supervised models, LIME is not directly suitable for interpreting t-SNE embeddings. By adapting LIME, Bibal et al. [5] provide local interpretations of t-SNE embeddings. They modified the sampling and black-box-query steps of LIME, and provided an interpretable model that explains the projection of an instance x on a t-SNE embedding Y.

Visualization is another way of providing post-hoc explainability. Chatzimparmpas et al. [7] proposed t-viSNE for analysts to interactively explore and interpret t-SNE projections. The tool enables them to inspect aspects such as effects of hyper-parameters, neighborhood preservation, densities, and correlations between dimensions and visual patterns of a t-SNE embedding. The coherent and well-integrated collection of views further adds to the usability and effectiveness of the tool, making t-SNE results more understandable.

Providing intrinsic explainability for t-SNE is often implemented in more dedicated ways, since it involves altering the optimization process or metrics to produce more explainable embeddings. Practices in the more general machine learning domain include applying specialized loss functions and employing attention mechanisms, as well as introducing prior knowledge via data augmentation in DMT-EV [43]. Replacing the default Euclidean distance in t-SNE, Cell-driven t-SNE (c-TSNE) [12] adapts biologically meaningful distance metrics reflecting cell difference to enhance the explainability of t-SNE embeddings. Customized for high-dimensional sparse scRNA-seq data, c-TSNE improves both interpretability and clustering efficiency.

3.2 Class-aware t-SNE variants

When class labels are provided along with the input data, supervised DR techniques can be applied to leverage the information present in the class labels. These methods have been effective in revealing the global structure of the data, with respect to their labels. Multiple t-SNE variants have been proposed to take the class labels into account during the DR process.

Class-aware t-SNE (cat-SNE) [9] explicitly accounts for class labels in t-SNE to improve the KNN accuracy in low-dimensional embedding. Different from the individual σ_i adjustments around each x_i based on a fixed perplexity in t-SNE, cat-SNE adjusts the neighborhood radius such that neighbors belonging to the same class account for the majority of the probability distribution. This approach creates smaller high-dimensional neighborhoods near class boundaries and larger ones within the core of the class, effectively stretching class boundaries and compressing the interiors.

Hierarchical Constraint t-SNE (HCt-SNE) [39] allows users to integrate hierarchical constraints directly into t-SNE embeddings, encoding them in an explicit tree. The hierarchical information is turned into a novel regularization term. The method makes use of the semantic information provided in class labels, and achieves satisfying results in both quality metrics and visual assessment.

Class-Constrained t-SNE [26] combines data feature structure with class probability structure into a unified projection to enhance interpretability of the produced embeddings. Data feature similarity is represented through position proximity between data points, while class probabilities are conveyed by the proximity between data points and class landmarks, iconic representations of classes optimized in 2D space. The preservation of data feature structure and class probability structure is balanced based on a parameter controlled by user input, resulting in a merged projection that reveals patterns from both perspectives. This approach mitigates ambiguity in class relationships by optimizing the positions of class landmarks to reflect class confusion, enabling clear visual differentiation and enhanced cluster interpretability.

Contrary to the above, conditional t-SNE (ct-SNE) [18] aims to discount prior information in the form of labels, instead of gaining more information through it. They put less emphasis on information that aligns with expectations (prior knowledge), and uses class labels to factor out prior knowledge from the embedding such that it can reveal other more fine-grained structure. The method allows more complementary structures in the data to be revealed, and provided fresh insights in the data.

3.3 t-SNE in Visual Cluster Analysis

Visual cluster analysis employs DR techniques to project high-dimensional data into 2D scatterplots, in which analysts can visually identify cluster patterns [40]. As clustering is an inherent human-in-the-loop task [2], visual cluster analysis is usually based on human perception of clusters in 2D scatterplots.

Xia et al. [42] extracted the top four most common analytical tasks in visual cluster analysis within general application domains. All following tasks are performed in the projected space, given the scatterplot S generated by a DR techique, or a cluster C in S, or a point P in S:

1. Cluster Identification: identify dense and well-separated clusters in S;

2. Membership Identification: identify the cluster C point P belongs to;
3. Distance Comparison: given a cluster C_1, identify the nearest cluster C_2 to it;
4. Density Comparison: given multiple clusters, identify the cluster with the largest density.

Three major requirements are identified for a DR technique to support interactive visual cluster analysis [41]:

1. Correctly constructing and preserving neighborhood structures in the embedding space;
2. Achieving clear visual cluster separation in the embedding space;
3. Supporting effective interactions for human analysis.

Based on the requirements listed above, evaluations of DR techniques in the context of visual cluster analysis can be classified as quantitative metric evaluations or evaluations based on user perception [42]. In quantitative metrics evaluation, the overall quality of clusters in the low-dimensional embedding can be measured by aggregate metrics such as continuity and trustworthiness, while local metrics focus on small neighborhoods, assessing the embedding in a more detailed level. Perception based evaluations focus on human perception in cluster-related tasks, comparing the performance of different DR techniques on visual cluster analysis tasks [11, 37], while specific measurements often depends on the study design.

t-SNE utilizes non-linear functions and preserves local neighborhood in its optimization process. Being able to recover well-separated clusters, it is widely used in visual cluster analysis [22]. An empirical study conducted by Xia et al. [42] proves that although there is no universally best DR technique for visual cluster analysis, t-SNE outperforms other DR techniques in cluster identification and membership idenfitication. A linear variation of t-SNE, t-SNLE [6] is also one of the preferred techniques in density comparison.

Chapter 4

Methodology

In this chapter, we introduce the DimenFix-t-SNE algorithm. Section 4.1 explains the adaptation of t-SNE to DimenFix, including generalizations to the method. Section 4.2 introduces the individual moving range and how it is generated. In Section 4.3, the most important control parameter, the pushing mode, is introduced. Section 4.4 goes into details of the ordering method when a nominal feature is on the fixed axis. Finally, Section 4.5 describes the axis switching method.

4.1 DimenFix-t-SNE

Being a Gradient-Descent (GD) based DR algorithm, t-SNE works in similar ways as the Force Scheme algorithm. Both experience the workflow dipicted by Figure 4.1: after random itialization, the embedding is iteratively updated through an optimization process, until a certain terminating criteria is satisfied and the algorithm produces the final output. The difference between the two algorithms lies in the optimization step, where Force Scheme simulates physical forces to calculate the force vector acting on points based on their pairwise relationships, while t-SNE works to minimize the KL divergence between the high- and low-dimensional similarity distributions.

When adapting t-SNE to DimenFix, we follow the same method as shown in line 7 of Algorithm 1, adding a DimenFix step after the Optimization and Update steps. The new workflow is dipicted by Figure 4.2. The principle is that all other steps remain intact when the DimenFix step is inserted, allowing DimenFix to be adapted to any gradient-descent-like method. In this case, calculations of the gradient and the descent process are kept the same as described in Section 2.1.

During the actual implementation of DimenFix-t-SNE, we have identified several limitations in the orignial DimenFix method, as well as implementation details worth noting. In the following subsections, we present improvements upon the Force-Scheme adapted DimenFix, using the adaptation to t-SNE as an example. The general aim is to improve embedding quality as well as add more flexibility to the method. In Algorithm 2, we present the new DimenFix-t-SNE process in pseudocode. The function for gradient descent in t-SNE is not written explicitly, since the method

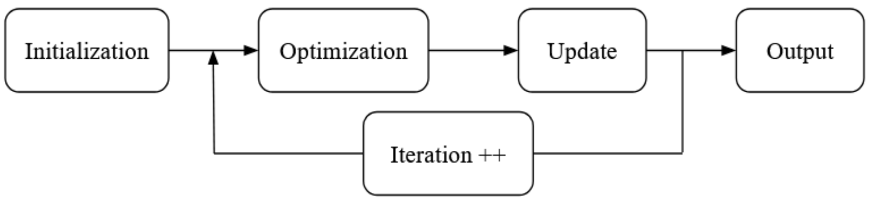


Figure 4.1: Abstract workflow of t-SNE and Force-Scheme.

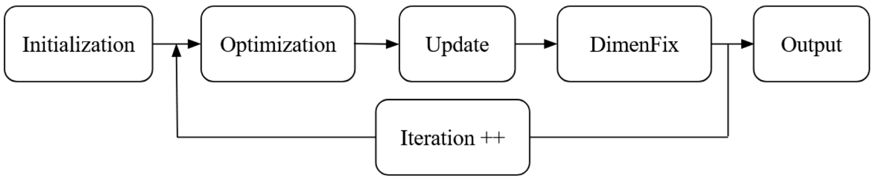


Figure 4.2: Abstract workflow of t-SNE and Force-Scheme adapted to DimenFix.

Algorithm 2: Dimenfix-t-SNE

```
Input: $X = \{x_1, ..., x_N\}$: input dataset, $V = \{v_1, ..., v_N\}$: original feature to be preserved
    or metadata, $\eta$: learning rate, $max\_it$: maximum number of iterations, $fix\_it$:
    interval in iterations at which values on the fixed axis are pushed, $\alpha$: overlap
    control
Optional Input: $R = \{[r_1^l, r_1^u], ..., [r_N^l, r_N^u]\}$: range limit defined by user
Output: $Y = \{y_1, ..., y_N\} = \{Y^1, Y^2\}$: 2d final embedding
1 Initialize each $y_i \in Y$ randomly: $y_i \sim \mathcal{U}(0, 1)$;
2 while $it < max\_it$ do
3     $\hat{Y} = gradient\_descent(X, Y, \eta)$;
4     $Y^2 = \hat{Y}^2$;
5     if $it \mod fix\_it = 0$ then
6         $Y^1 = push(R, \hat{Y}^1, mode)$;
7     else
8         $Y^1 = \hat{Y}^1$;
9     $it++$;
```

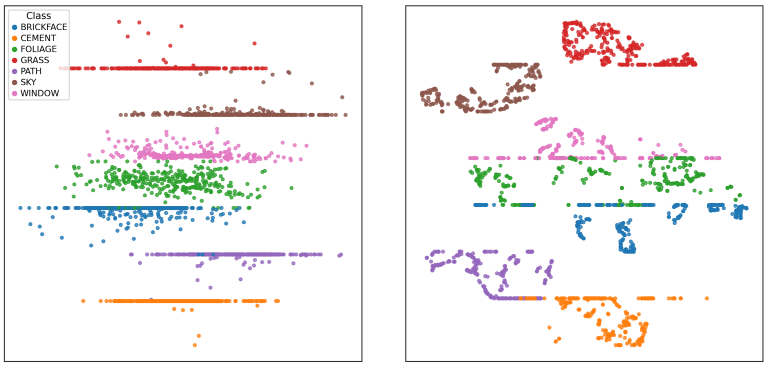
can be adapted to DR algorithms sharing similar workflows with t-SNE and Force Scheme. We discuss in following sections the generalizations to DimenFix, implemented based on t-SNE.

4.1.1 Fixed Iterations

In the original Force-Scheme adaptation of DimenFix, points are pushed to their designated positions on the fixed axis every iteration. This strategy ensures that points align strictly with their designated positions on the fixed axis at any intermediate iteration.

However, both Force-Scheme and t-SNE optimize the low-dimension embedding by gradually updating the embedding with small steps, allowing the embedding to evolve smoothly. Both can be seen as balancing between the attractive and repulsive forces between points in the embedding during optimization. The *push* step in DimenFix moves points along the fixed axis regardless of existing forces in that direction, which leads to oscillation in forces along the fixed axis. In general, pushing in every iteration causes forces along the fixed axis to never fully manifest in actual point displacement. This may hinder the movement of points along the free axis, as well as leading to bad local minima.

Based on this observation, since pushing points along the fixed axis at every iteration is not necessary, we reduce the pushing frequency by introducing the parameter *fix_it*. Instead of pushing points at every iteration, we apply pushing less frequently, based on the interval *fix_it* which can be defined by the user. As indicated by line 5 of Algorithm 2, we allow a number of regular gradient descent iterations to happen between each push operation. In this way, points are allowed more freedom to adjust on both axes without constant constraint and disturbance to the forces. t-SNE is commonly run with exaggerated forces in the first few hundered iterations (early exaggeration).



(a) $fix\_it = 1$ (b) $fix\_it = 50$

Figure 4.3: DimenFix-t-SNE on the Image Segmentation dataset, with different fixed iterations, colored with class labels.

During these iterations, exaggerated attractive forces are used to form clusters quickly and reveal the global data structure, before the forces are slowly adjusted to reflect the similarities between points faithfully. To avoid disturbing this process by forcing points to arbitrary positions, pushing can be disabled for these early iterations completely to avoid disrupting the formation of the global embedding structure and help maintain global relationships. This is in fact the case when class ordering is introduced to the method.

One other issue related to *fix_it* lies in initialization. In line 1 of Algorithm 1, the fixed axis y_i^1 is initialized with x_i^j, which indicates a *push*-like action taking place at iteration 0. This is also removed when *fix_it* is introduced, being replaced by random initialization on both axes, to both avoid disturbing the exaggerated forces and better adapt to different *push* modes which will be introduced in later sections.

Figure 4.3 illustrates results when applying DimenFix-t-SNE on the Image Segmentation dataset [1]. The two embeddings share the same random initialization and fixed axis, as well as a perplexity of 30 and a total of 600 iterations. Figure 4.3a displays the embedding with *fix_it* set to 1 (push every iteration), and in Figure 4.3b it's 50 (push every 50 iterations). By pushing every iteration, the embedding structure in Figure 4.3a is almost completely distorted. The contrast between objective qualities of the embeddings is also apparent, with a Trustworthiness [36] score of 0.84 when $fix\_it = 1$ and 0.97 when $fix\_it = 50$.

However, despite its apparent shortcomings, pushing at every iteration has the advantage of producing a smooth transition during the embedding process, especially when visualized as an animation. In contrast, when pushing is applied every *fix_it* iterations ($fix\_it > 1$), the transitions appear less consistent, with points seemingly jumping to new positions. And the larger *fix_it*, the more abrupt the "jump". This trade-off should be considered when determining a suitable *fix_it* value for DimenFix-t-SNE: use a smaller *fix_it* when a smooth visualization of the optimization process is necessary, and a reasonably bigger *fix_it* when only the final result is relevant.

4.1.2 Rotation For Minimal Push

It is clear that no matter what feature we use as the fixed axis, *push* will always introduce a certain amount of distortion to the embedding. By introducing *fix_it*, we limit this distortion by decreasing the frequency of pushes. To further minimize the disturbance introduced to the embedding through *push*, a rotation is applied to the embedding to find an angle where the modification by each *push* can be minimal.

This approach is based on t-SNE's rotational invariance and achieved through a brute-force approach. At each iteration when *push* is scheduled, a point at position $\hat{y}_i^1$ on the fixed axis after gradient descent is about to be pushed to position y_i^1. The total amount of movement along the fixed axis introduced by *push* across all points in the embedding is

$$\sum_i \left\| y_i^1 - \hat{y}_i^1 \right\| . \tag{4.1}$$

Aiming to minimize this movement, at each push iteration, a series of rotations are applied to the embedding before the actual *push.* We test the angle for the rotation at 10-degree increments. For each angle, the resulting movement is measured according to Equation 4.1. The angle that results in the least movement is selected for updating the embedding and performing the push.

4.2 Ranges

In the Moving-In-Range Mode of DimenFix described in Section 2.2, a global moving range $[-\alpha, \alpha]$ is defined for all points to move along the fixed axis. The range introduces freedom on the fixed axis to some extent, yet it is also rather limited since all points adhere to the same range. We generalize this process and make it more user-steerable by allowing ranges to be defined for every individual data point. A point in the calculated embedding after *gradient_descent* is referred to as $\hat{y}_i = \{\hat{y_i^1}, \hat{y_i^2}\}$, and its corresponding range limit $[r_i^l, r_i^u]$. One point in the updated embedding is marked as $y_i = \{y_i^1, y_i^2\}$. Users can input a vector R with a pair of min and max values per point to indicate the individual moving ranges defined for each point. Apart from user input, we implement two methods for the creation of individual moving ranges, separately for quantitative (Sectin 4.2.1) and nominal data (Section 4.2.2). We explain the normalization for both generated and inputted ranges in Section 4.2.3.

4.2.1 Quantitative Feature

With quantitative input $V = \{v_1, ..., v_N\}$, the values are first sorted, giving $V' = \{v_1', ..., v_N'\}$, where $v_i' \leq v_j', 1 \leq i < j \leq N$. We then create ranges on the fixed axis covering the value range of $[v_1, v_N]$. For an input point i with v_i' being the fixed feature value, its lower range r_i^l and upper range r_i^u are defined as

$$r_i^l = v_i' - \frac{(v_i' - v_{i-1}')\theta}{2}$$

$$r_i^u = v_i' + \frac{(v_{i+1}' - v_i')\theta}{2}.$$

The overlap control parameter θ is an user input that controls how one range overlaps with neighboring ranges. When set to 1, ranges are stacked right next to each other, covering the whole fixed axis; with $\theta > 1$, neighboring ranges would overlap with each other, to an extent based on their original size; when setting θ to 0 in this scenario, points will be pushed to strictly the feature value on the fixed axis. By introducing θ, we generalize the Strictly Fixed Mode and Move-In-Range Mode of the original DimenFix.

4.2.2 Nominal Feature

When the input V contains nominal data such as class labels, they are first mapped to consecutive integer values in a random order, and then treated in a similar way as quantitative input. In this case, we can also take class density into account. With a total of M points from S distinct classes, class labels are mapped to an array of integers $[0, 1, ..., S-1]$ and the class assigned with value i contains M_i points. The range for points within class i is defined as:

$$r_i^l = r_{i-1}^u,$$

$$r_i^u = r_i^l + \frac{M_i}{M}. \tag{4.2}$$

With $i = 0, r_0^l = 0$ and with $i = S-1, r_{S-1}^u = 1$. Density can also be left out of account when classes are highly unevenly distributed. In that case, ranges are equally distributed for each class, changing Equation 4.2 to

$$r_i^u = r_i^l + \frac{1}{S}.$$

θ still applies here in the sense that the ranges are first created with a default $\theta = 1$, and then rescaled to θ their original size with their centers unchanged.

4.2.3 Normalization

As mentioned before, user defined range limits are also allowed. So long as the input range limit array is of size $(num\_points, 2)$, users can move points towards arbitrary positions on the fixed axis.

We normalize the ranges to [0, 100] upon generation or after input. One thing worth noting here is that the fixed axis is rescaled to match the size of the free axis each iteration before *push* to ensure consistency between the forces on different axes. Combined with the normalization of ranges, values on the fixed axis only represent relative ordinal relations instead of exact values, regardless of how the ranges are generated.

4.3 Push Modes

The core idea of DimenFix is that points get "pushed" into pre-defined values or ranges on the fixed axis during the DR process, while on the other axis they are allowed to move around freely. In the Force-Scheme DimenFix adaptation, ranges are defined as a uniform ratio of the embedding size, rendering uniform moving ranges for every point. By defining per-point range limits, the *push()* function gains more flexibility. In this section, we explore another aspect of the *push()* function. *Mode* determines how points are moved (pushed) into their corresponding ranges, and it also strongly affects the final layout of points in the output embedding. We define three different modes, indicated by the mode parameter in Algorithm 2, Clipping (Section 4.3.1) and Gaussian (Section 4.3.2) adapted from the original algorithm and a new mode: Rescale (Section 4.3.3).

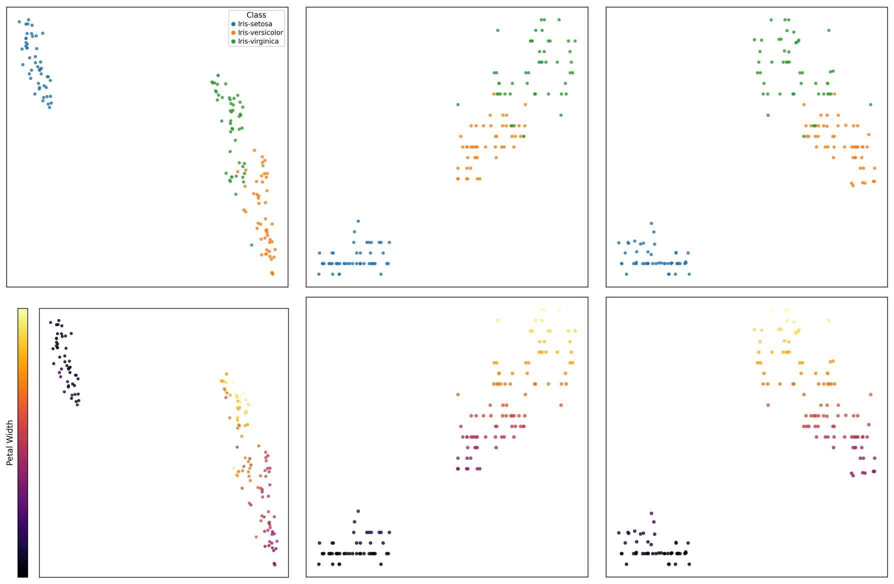
4.3.1 Clipping

By its name, the Clipping mode provides a straight-forward way of moving points same as the standard clipping function. If $\hat{y}_i^1$ exceeds the defined limits, it is set to the nearest boundary value. With the lower range r_i^l and the upper range r_i^u, the value update for point i on the fixed axis is defined as

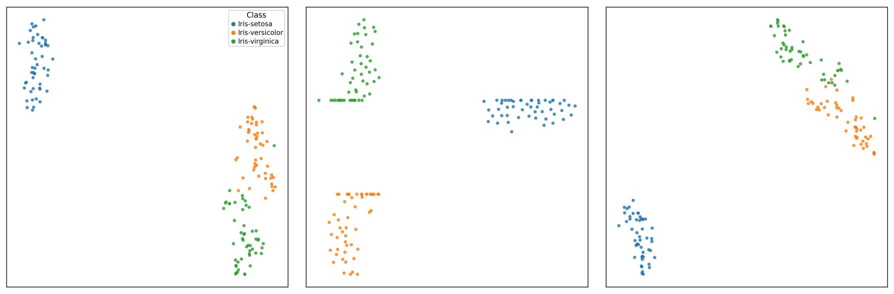
$$y_i^2 = \begin{cases} r_i^u & \hat{y}_i^1 > r_i^u \\ r_i^l & \hat{y}_i^1 < r_i^l \\ \hat{y}_i^1 & \text{otherwise.} \end{cases}$$

When fixing an ordinal feature on the fixed axis, Clipping can be used together with overlap control θ set to 0. This ensures that points are moved to the exact value (scaled) according to the selected original feature V on the fixed axis. Figure 4.4b shows the output when applying the Clipping mode on the Iris dataset, with $\theta = 0$ and *petal_width* as the fixed feature preserved on the y-axis. The top figure is colored according to class labels, and the lower one by the value of *petal_width*. Horizontal lines indicate that multiple datapoints share the same *petal_width* value. Values on the y-axis also show a clear linear trend when *petal_width* is used in the color map.

Clipping can also be applied when fixing meta-info such as class labels, which is shown in Figure 4.4e. Here the class ranges are generated regarding points' distribution in classes, in accordance with Equation 4.2, and points are colored based on their class labels. Compared with the regular t-SNE results shown in Figure 4.4a, lines appear at both edges of class ranges in



(a) Regular t-SNE (b) Clipping Petal Width (c) Gaussian Petal Width



(d) Rescale Class Labels (e) Clipping Class Labels (f) Gaussian Class Labels

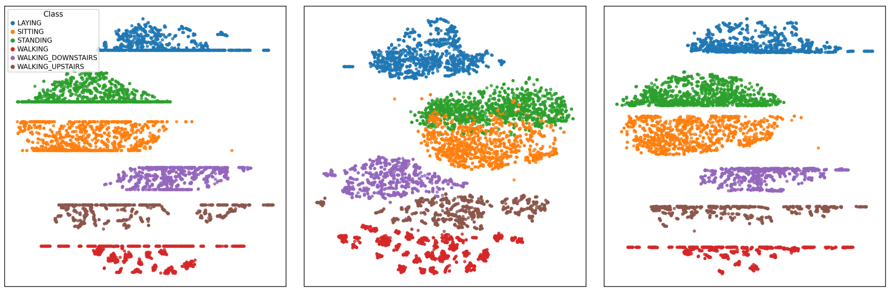
Figure 4.4: Results from different modes using DimenFix-t-SNE on the Iris dataset.

Figure 4.4e, indicating that during regular t-SNE iterations (between "pushes") points coming from separate classes moved towards each other, but were then pushed back into their predefined ranges by DimenFix.

4.3.2 Gaussian

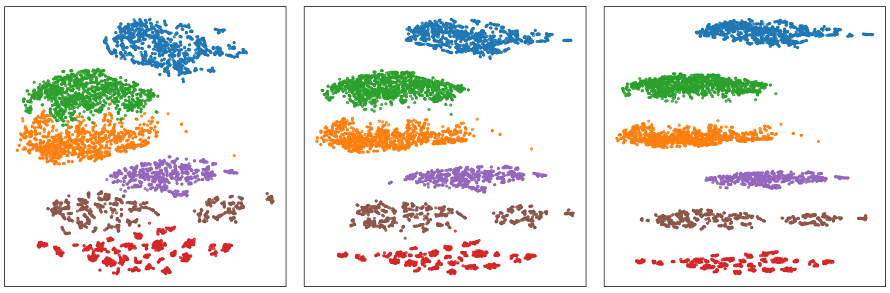
The Clipping mode ensures that after every *push*, all points are moved back to their predefined ranges on the fixed axis. However, Clipping also causes great distortion to the embedding in the sense that points moving far away from the set ranges are treated the same way as points moving just slightly outside of the ranges. The consequence is apparent: "lines" indicating points' accumulations appear on either or both sides of the set ranges, apparent in both Figure 4.4b and Figure 4.4e. This may cause the movement of the points in the fixed axis to be inconsistent with the free axes.

The Gaussian mode is designed to deal with this problem. In this mode, points are allowed to move out of their defined ranges to a limited extent. On the fixed axis, points' movements are scaled by a Gaussian function centered at the middle of their ranges. We have adapted this mode directly from the original implementation of DimenFix, matching the description in Section 2.2. Results of this mode are shown in Figures 4.4c and 4.4f. Lines created by points accumulating by the edges of ranges are far less apparent under this mode.



(a) Clipping, $\theta = 0.7$ (b) Gaussian, $\theta = 0.7$ (c) Gaussian, $\theta = 0.4$

Figure 4.5: DimenFix-t-SNE on HAR under the Clipping and Guassian modes, fixing class label.



(a) $\theta = 1.0$ (b) $\theta = 0.7$ (c) $\theta = 0.4$

Figure 4.6: DimenFix-t-SNE on HAR under the Rescale mode with different values of θ, fixing class label.

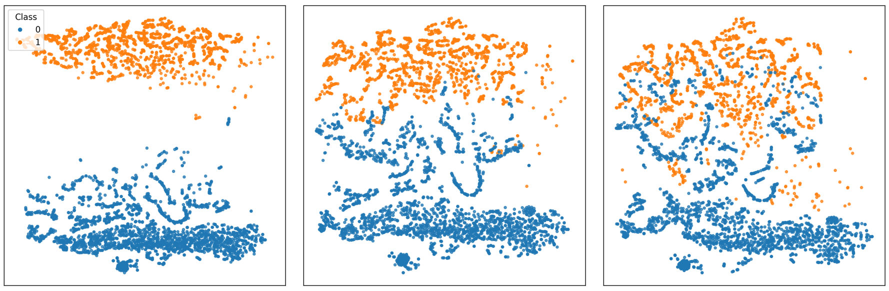
4.3.3 Rescale

The Gaussian mode improves upon the Clipping mode, making movement on the fixed axis more consistent and eliminating the "lines" of points gathering on range boundaries, especially with class labels as the fixed axis. And on simple datasets such as Iris, the mode works quite well. However, when we apply Gaussian on datasets with mixed clusters, issues emerge.

In Figure 4.5, we apply DimenFix-t-SNE to the Human Activity Recognition Using Smartphones (HAR) dataset [32], using class label as the fixed axis. The overlap control θ is set to 0.7 to keep clusters separated along the fixed axis (y-axis in this example). Clipping behaves as expected (Figure 4.5a), creating intervals between classes and lines at class borders. Gaussian, however, does not separate all classes based on θ (Figure 4.5b). Instead, the non-linear moving distance defined by the Gaussian mode makes controlling the overlap between classes harder. Trying to work around this, as shown in Figure 4.5c, when we set θ to 0.4, classes get separated properly, but at the cost of lines reappearing as in the Clipping mode. This issue arises because both Clipping and Gaussian push points to designated ranges based on the relative position of y_i^1 to the range center $\frac{r_i^l + r_i^u}{2}$, introducing inconsistency between the movements of different points within the same class (points that share the same range), thus causing distortion to the shape of clusters, making it harder to interpret the embedding.

We propose the Rescale mode to deal with the aforementioned issues. It is applied specifically when class labels or other nominal attributes are on the fixed axis. During *push*, we rescale every group of points sharing the same range as a whole to fit into the defined range. Take a group of points sharing the same range limit $[r_I^l, r_I^u]$, $N\%$ of the points' values on the fixed axis lie in the range $[y_{Nmin}, y_{Nmax}]$. The center of the group on the fixed axis is at

$$old\_center = (y_{Nmin} + y_{Nmax})/2,$$



(a) $N = 100$ (b) $N = 95$ (c) $N = 80$

Figure 4.7: DimenFix-t-SNE on Spambase under the Rescale mode with different values of N, $\theta = 1.0$, fixing class label.

when the supposed-to-be center of the group is

$$new\_center = (r_I^u + r_I^l)/2.$$

So the fixed axis value update of any point in this group can be defined as

$$y_i^1 = \frac{(\hat{y_i^1} - old\_center)(r_I^u - r_I^l)}{y_{Nmax} - y_{Nmin}} + new\_center.$$

Rescale preserves the overall shapes of groups consisting of points sharing the same range, while separating the classes nicely. Figure 4.6 shows the Rescale mode on the HAR dataset, with different values of θ. Classes are separated clearly when $\theta = 0.7$. The shape of clusters changes linearly with θ, and no "lines" appear on class borders.

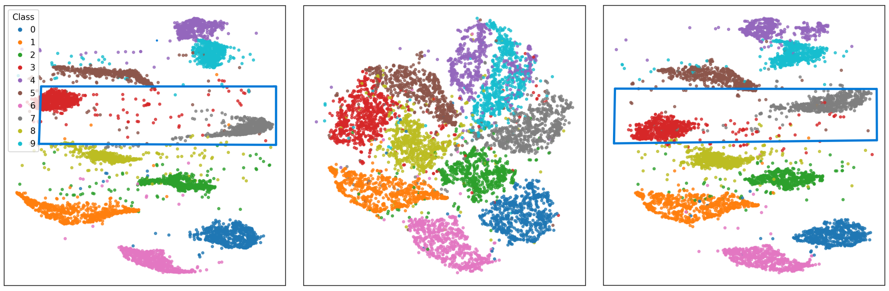
Flexibility is ensured with parameter N, indicating the extent to which rescaling is applied to the group. At $N = 100$, all points will be pushed within the designated range, while at $N = 80$, 80% of points within each group will lie inside ranges, allowing some outliers to remain astray. One thing worth noting for this mode is that the overlap control parameter θ does not directly control the overlap between neighboring ranges unless $N = 100$. When $0 < N < 100$, a certain amount of points will roam outside the set ranges, causing neighboring ranges to overlap. Figure 4.7 shows the effect of N on the Spambase dataset, with $\theta = 1.0$. At $N = 100$, the two classes are separated strictly, with a large gap between caused by outliers in the clusters. At $N = 95$, the majority of points are separated, while outliers from two classes mix with each other. At $N = 80$, more mixing happens.

Since N only controls *how many* points lie outside of the range, there is no guarantee how much overlap will exist between ranges. To control exactly how much neighboring classes overlap, we recommend using the Clipping mode. If the purpose is to preserve the shape of clusters while separating classes, Rescale works better.

4.4 Class Ordering

When using nominal meta-data such as class labels as the fixed axis, distinction between classes are made through their positions on the fixed axis. The overlap control θ and different modes allow us to layout the classes in various ways, whether it is emphasizing the separation of classes or preserving the shape of clusters. However, one issue remains.

The problem with using a nominal fixed axis lies in the fact that no intrinsic order exists in nominal data. As described in Section 4.2.2, nominal data is mapped to integer values in a random order when generating the ranges. There is thus no guarantee that such arrangement of groups is optimal. Groups that lie close to each other in the high dimensional space might be kept apart while groups that share almost no similarity might be pressed together.



(a) iteration 300 (b) iteration 349 (c) iteration 350

Figure 4.8: Intermediate iterations of DimenFix-t-SNE on MNIST under the Rescale mode, fixing class label, push every 50 iterations.

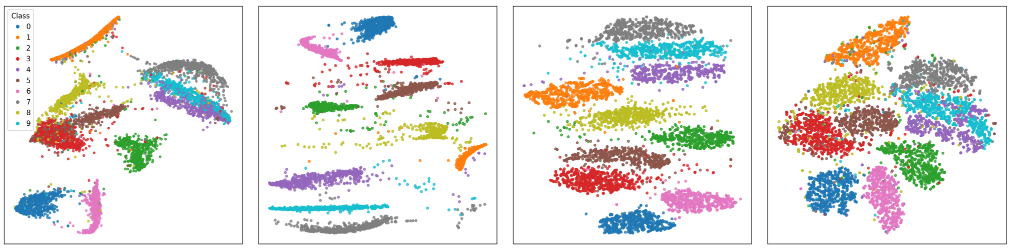
We devise a class ordering method to solve this issue. Here, we use class representatively for any nominal input. When class label is the selected feature to "fix", we generate the range limit based on the global structure of the data and adjust it during the optimization process to consistently reflect said structure.

Let's start with a DimenFix-t-SNE process using class label as the fixed axis, and assuming a random order of classes as described in Section 4.2.2. Figure 4.8a shows the embedding after the second *push* and Figure 4.8b shows the embedding before the third *push*. Since the class order is static in this case, one iteration later classes will be pushed back to their original positions. However, it is apparent that classes have tendencies to move to new positions along the fixed axis, by switching places with their neighboring classes.

We leverage this "tendency to move" information to adjust the order of classes during DimenFix. The classes' average positions $A_i = [A_{ix}, A_{iy}]$ are used to represent their inherent positions in the low-dimensional space before *push*. Considering that rotation does not affect t-SNE embeddings, PCA is applied before *push* to align the largest variance with the fixed axis. Then an order is generated based on class averages, according to A_{iy}. The aforementioned ranges are then rearranged according to this new order. After that *push* happens, moving points to their new designated ranges. Figure 4.8c shows the embedding after the third *push*, using class average positions to adjust the class order. After this *push*, the red (Class 3) and grey (Class 7) classes (highlighted) switched positions. By following the points' inherent movement, we aim to minimize classes' accumulated movement along the fixed axis.

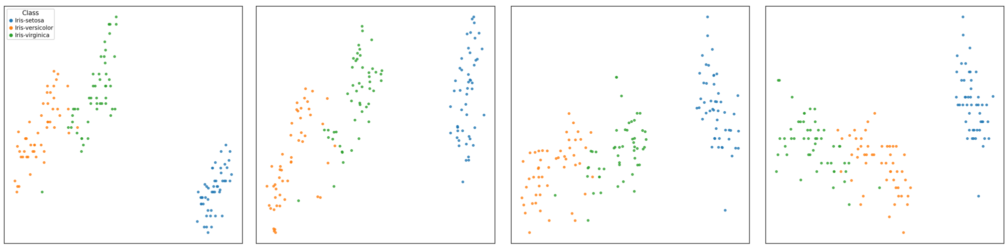
However, the adjustment we described tends to reorder the classes locally by switching between neighboring classes. This is due to t-SNE's nature of updating the positions of points with small steps as shown in Figure 4.8. *fix_it* affects the ordering result by determining the number of iterations for the classes to move. With a small *fix_it*, movement of classes between two adjacent pushes may not be enough for the local class switching to happen. This is yet another reason for introducing *fix_it* instead of pushing every iteration, and a well-chosen *fix_it* would allows us to find better orderings of classes.

Even with a suitable choice of *fix_it*, large reshuffles in the order of classes do not happen much. That being said, the final order of classes is still highly dependent of the initial order. Since nominal features usually do not have an explicit order, we try to leverage an order from the data's global structure. As described in Section 2.1, early exaggeration reveals the global structure of the embedding by forming tight yet separated clusters. We make use of this structure by calculating an initial order of classes based on it. In this case, to better preserve the global structure, *push* is disabled till the end of early exaggeration. Shown in Figure 4.9a, after early exaggeration, clusters are formed tightly. We align the direction of largest variance with the fixed axis (y-axis in this example) by applying PCA, and generate an initial order of classes based on their average positions. Figure 4.9b shows the embedding after the rotation (by PCA) and *push* is applied to the embedding for the first time. This generated order can be further adjusted based on the changes in classes' average positions. We let the adjustment of class order happen in step with



(a) Early Exaggeration (b) Rotation, First Push (c) Final Result (d) Regular t-SNE

Figure 4.9: DimenFix-t-SNE on MNIST with class ordering enabled, push every 20 iterations.



(a) Iteration 0 (b) Iteration 10 (c) Iteration 20 (d) sepal_width

Figure 4.10: DimenFix-t-SNE on the Iris dataset, the fixed axis switching from sepal_length to sepal_width (left 1-3) in 20 iterations. And DimenFix-t-SNE fixing sepal_width, generated from random initialization (rightmost).

every *push*. Every *fix_it* iterations, the embedding gets rotated by PCA, the new class average positions are calculated, a new order of classes emerge, and finally points are pushed to their new ranges.

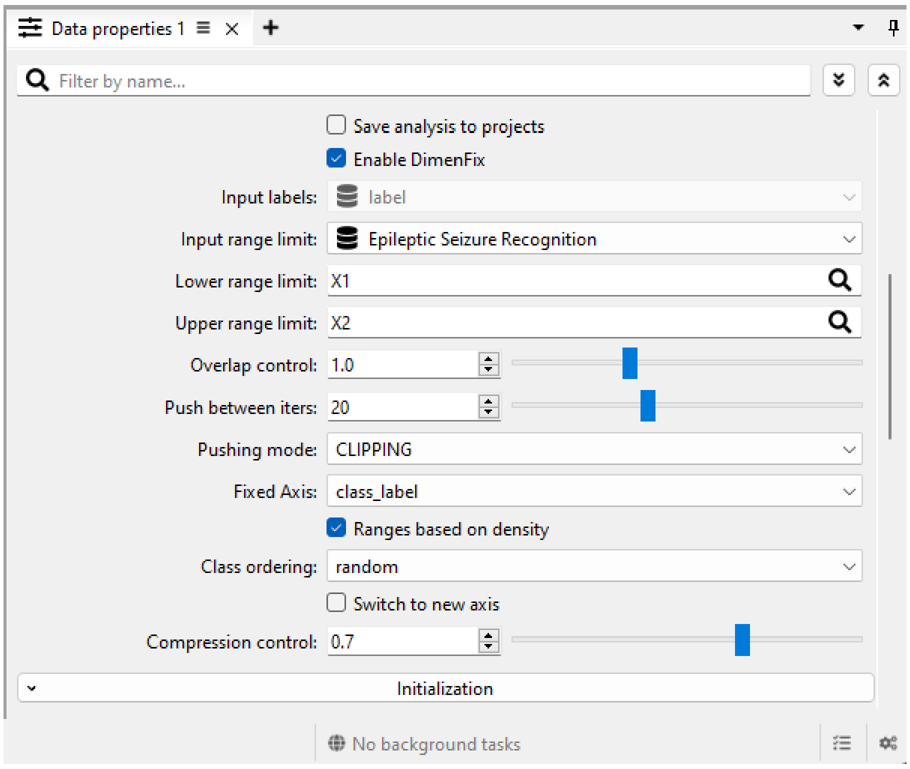
Figure 4.9c shows the final result produced with class ordering enabled, using the same random initialization and color map as the regular t-SNE producing Figure 4.9d. Group of clusters that lie close to each other in Figure 4.9d are also ordered next to each other in Figure 4.9c.

4.5 Switching

DimenFix aids the user when exploring high-dimensional data by explicitly preserving one chosen feature on the fixed axis during the DR process. In an interactive data exploration context, the user may also want to switch the feature being fixed and inspect what difference it would make to the embedding. However, no such support is present in the Force-Scheme adaption of DimenFix. In the original implementation, recomputation is needed whenever the user wants to switch between different features on the fixed axis, or between different *push* modes. This adds complication to the matter when the user wants to inspect relation between different features or compare the behavior of different pushing modes, since with recomputation all structures in the embedding are reshuffled through reinitialization. It is impossible to trace the movement of groups visually, as well as time consuming for the whole embedding to be recalculated.

We develop an easy-to-track smooth switching method to solve this issue. The aim of this feature is to shorten the time needed when switching between features or push modes, as well as providing a traceable switching procedure for the user to examine.

Aiming to make the switching traceable, we perform switching by continuing from the already calculated embedding X. When switching is enabled, the range limit and *push* mode is updated according to input, and DimenFix-t-SNE is continued from X with the new parameters. Through continuation, we avoid the reshuffling of points caused by random initializaiton, enabling the user to trace the movement of group of points visually.


Data properties 1

Filter by name...

Save analysis to projects [ ]

Enable DimenFix [✓]

Input labels: label

Input range limit: Epileptic Seizure Recognition

Lower range limit: X1

Upper range limit: X2

Overlap control: 1.0

Push between iters: 20

Pushing mode: CLIPPING

Fixed Axis: class_label

Ranges based on density [✓]

Class ordering: random

Switch to new axis [ ]

Compression control: 0.7

Initialization

No background tasks

Figure 4.11: GUI for DimenFix-t-SNE plugin on Manivault.

Problems still arise with this "continuation" approach. Since the embedding X is already a developed t-SNE embedding with a fixed axis, forces along the free axis tend to be relatively balanced. As a result, when the fixed axis is altered, the free axis often remains underdeveloped, limiting the dynamic adjustment of the embedding during switch. To address this, we apply a compression step to the embedding prior to switching, to amplify the pairwise forces and promotes further development along the free axis. Before switching happens, X is first compressed at rate $\beta (0 < \beta < 1)$.

With the switch of features, the range limit is bound to change, in many cases drastically. The movement of points from their old ranges to new ranges can be abrupt, since *push* will happen in one single iteration. This goes against our purpose in making the process traceable. So instead of pushing the points to their new ranges in one iteration, points are gradually moved towards their designated positions during N iterations. Parameter N controls how long the switching process takes. Take point p_i which is located at p_i^y on the fixed axis and needs to be pushed to $p_i^{\hat{y}}$ after switching. For the first N switching iterations, p_i's position on the fixed axis at iteration n would be

$$p_i^y + \frac{n(p_i^{\hat{y}} - p_i^y)}{N}.$$

During these N iterations, users should be able to see points moving progressively towards their new ranges. Figure 4.10 shows the switching process. At iteration 0, the embedding is a result of DimenFix-t-SNE, generated from random initialization and using sepal_length as the fixed axis. During the first 20 iterations, points are moved gradually through a linear interpolation to the positions indicated by the new fixed feature: sepal_width (Figure 4.10b and Figure 4.10c). Figure 4.10d shows the embedding generated from random initialization, using sepal_width as the fixed axis for reference. After the first N iterations, the usual DimenFix-t-SNE takes place, pushing points into their ranges by intervals determined by *fix_it*.

Apart from switching to a different axis, the user can also switch to a different mode based on existing embedding. The process remains the same, applying additional iterations after a compression step, but without changing the fixed axis.

The switching process can be achieved by other means as well. For example, recomputation can happen in the background when the user wants to switch to a different feature on the fixed axis. After the background recomputation is completed, resulting in a new embedding Y, the gradual transition described above can then be applied between embeddings X and Y, achieving traceable switching similar to the method described above.

4.6 DimenFix-t-SNE Plugin

Extending from the t-SNE-Analysis plugin developed on Manivault Studio, DimenFix-t-SNE is implemented to support the DimenFix adapted version.

The algorithm described in earlier sections can be implemented regardless of platform and t-SNE implementation. To enhance performance and further facilitate user interaction, we developed the plugin using a GPGPU t-SNE implementation. Key features include: 1. Range limit selection from the input dataset, or arbitrary user input, and generation based on parameters such as density-based and overlap control; 2. Three different *push* Modes as described in Section 4.3; 3. Class ordering for nominal fixed features; 4. Switching to a new axis by continuing from the calculated embedding. The control panel containing the user-steerable parameters is shown in Figure 4.11. This is also the control panel used by participants in the user experiment in Section 5.3.

Chapter 5

Results

In this chapter, we evaluate the DimenFix-t-SNE method based on both quantitative and qualitative methods, along with an interactive user experiment. Section 5.1 gives an overview of datasets used in the experiments. We first conduct a metrics-based evaluation, measuring the embeddings' overall quality based on a number of metrics. The run time setup and results will be presented and discussed in Section 5.2. To complement the first experiment, we conduct a user-study to further evaluate the quality of embeddings in a task-based environment, as well as to measure the interpretability of the method. Experiment setup and results will be described in Section 5.3.

5.1 Datasets

Seven datasets in total are used for evaluation. In table 5.1, an overview of the datasets is presented. The first five datasets are used in the metric-based evaluation, while the last two only appear in the user-based evaluation. In order to ensure the coverage of the evaluation, we have chosen datasets of various sizes, dimensionalities and properties. Datasets also vary in the correlation between clusters and data points under the same labels.

Dataset	**Size**	**Dimensions**	**Type**	**Properties**
Iris	150	4	Real	well separated clusters
Image Segmentation [1]	2310	19	Real	mixed labels and clusters
(Sampled) MNIST	7000	784	Real	well separated clusters
Epileptic Seizure Recognition [31]	11500	178	Real	time series data
Spambase [15]	4601	57	Real	mixed labels and clusters
single-molecule FISH (smFISH) [23]	2360	627	Real	hierarchical labels
Comparative single-nucleus profiling of motor cortex [4]	38099	541	Real	hierarchical labels

Table 5.1: Evaluation Datasets Overview

5.2 Metrics-Based Evaluation

In the metric-based experiment, we focus on measuring the quality of DimenFix-t-SNE created embeddings. We first compare general results produced by DimenFix-t-SNE when fixing a quantitative feature on one axis to those produced by regular t-SNE, and then go on to inspecting the different effects of *push* modes. Furthermore, we measure the effectiveness of class ordering by comparing the ordering enabled embedding to random ordered ones. Finally, switching is evaluated by comparing embeddings produced after switching to embeddings with the same fixed axis produced directly from random initialization.

Five metrics are used for this measurement, using n to denote the number of points in the dataset, and k as the number of nearest neighbors.

Trustworthiness measures local structure preservation by penalizing false neighbors introduced in the embedding, defined as

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in U_k^{(i)}} (r_{ij} - k),$$

where r_{ij} is the rank of point j in the high-dimensional space with respect to point i, and $U_k^{(i)}$ is the set of points that are in the k-nearest neighbors of i in the embedding space but not in the high-dimensional space.

Continuity penalizes lost neighbors that were in the high-dimensional space but not in the embedding, defined as

$$C(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in V_k^{(i)}} (\hat{r}_{ij} - k),$$

where $\hat{r}_{ij}$ is the rank of point j in the low-dimensional space with respect to point i, $V_k^{(i)}$ is the set of points that are in the k-nearest neighbors of i in the high-dimensional space but not in the low-dimensional space.

InterContinuity discounts neighbors with the same class label when calculating continuity, and shows how well the neighborhood structure between datapoints from different labels is preserved. InterContinuity is calculated by

$$\mathrm{IC}(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{\substack{j \in V_k^{(i)} \\ y_i \neq y_j}} (\hat{r}_{ij} - k)\,.$$

We use this metric to measure inter-class neighborhood preservation.

The first three metrics mainly measure neighborhood preservation, while the last two metrics focus on inter-cluster structures of the embedding and are aggregated iteratively from partial distortions. We use M to denote the number of sampled clusters from the high-dimensional space, and w_i is the weight assigned to cluster i. $m_i^{compress}$ and $m_i^{stretch}$ are a pair of distortion measures, standing separately for missing-groups and false-groups.

Steadiness [16] measures how well clusters in the projected space form clusters in the high-dimensional space and can be calculated by

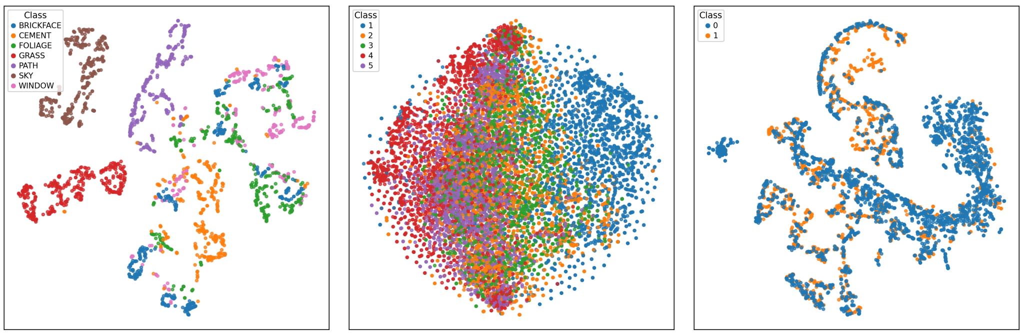
$$\mathrm{S} = 1 - \frac{\sum_{i=1}^{N} w_i \, m_i^{\text{stretch}}}{\sum_{i=1}^{N} w_i}.$$

Cohesiveness [16] measures how well the clusters in the high-dimensional space stay as clusters in the projected space, defined as

$$\mathrm{C} = 1 - \frac{\sum_{i=1}^{M} w_i \, m_i^{\text{compress}}}{\sum_{i=1}^{M} w_i}.$$

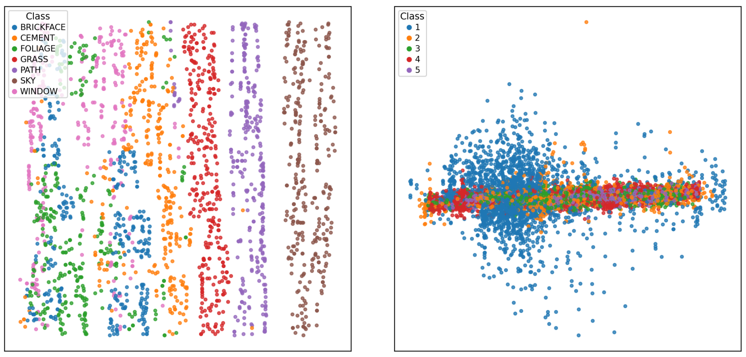
5.2.1 Quantitative Axis

We measure the general quality of embedding created by DimenFix-t-SNE, when using a quantitative input feature as the fixed axis. We use the Iris, Image Segmentation and Epileptic Seizure Recognition datasets, covering small ($\sim 10^2$ samples), medium ($\sim 10^3$ samples), and large ($\sim 10^4$ samples) datasets, with low ($\sim 10^0$), moderate ($\sim 10^1$), and high ($\sim 10^2$) input dimensionalities, respectively. For each dataset, we randomly choose between 1 and 4 input features (corresponding to their input dimensionalities) as the fixed axis and run DimenFix-t-SNE. Regular t-SNE



(a) Image Segmentation (b) ESR (c) Spambase

Figure 5.1: Regular t-SNE Embeddings on three datasets used for evaluation (Image Segmentation, Epileptic Seizure Recognition and Spambase), with perplexity 30 and 1000 iterations.



(a) Image Segmentation (b) ESR

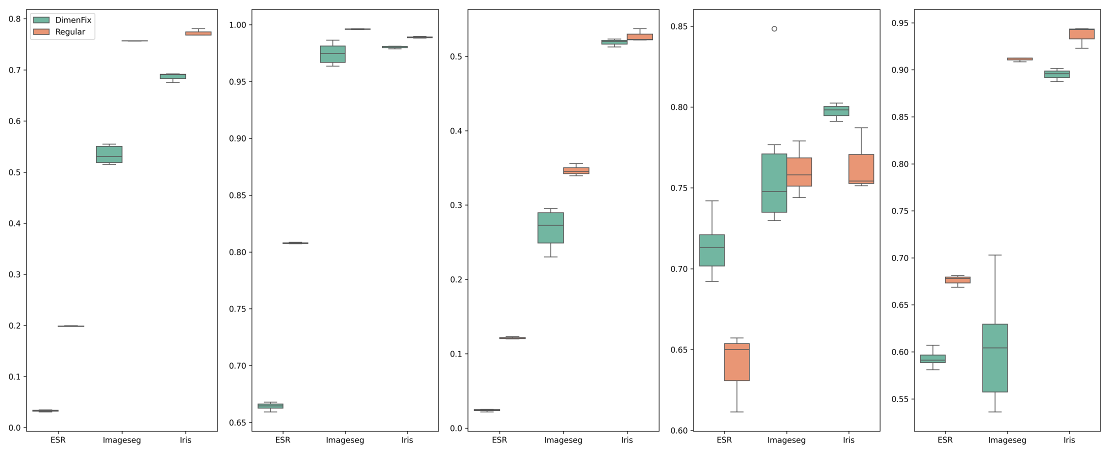
Figure 5.2: DimenFix-t-SNE embeddings on two datasets (Image Segmentation and Epileptic Seizure Recognition), fixing a random-picked quantitative input feature, with perplexity 30 and 1000 iterations.

is applied on the same datasets to create the baseline for comparison. For all datasets we use a perplexity of 30 and 1000 iterations. On DimenFix-t-SNE, we set the fix_iter to 20 and apply the Clipping mode, moving points to exact positions indicated by the input feature. To capture the variability of the data, we calculate each embedding 3 times with 3 different random initializations.

Figure 5.3 shows the results by dataset and metric. Each boxplot aggregates the results of the randomly selected axes across three random runs.

For all metrics except Cohesiveness, the baseline regular t-SNE outperforms DimenFix-t-SNE. This performance is expected, since by forcing one output axis to resemble the chosen input feature, the embedding is bound to create distortions. The difference in performance is less apparent for the Iris dataset, since it only has four input dimensions and a limited amount of instances. Clusters defined by class labels are still semi-identifiable when fixing a quantitative axis, as shown in Figure 4.4c and Figure 4.4b. On larger and more complex datasets, however, the distortion is more apparent, not only negatively affecting the preservation of neighbors (indicated by Continuity, Trustworthiness and InterContinuity), but also harming the inter-cluster structure (indicated by Steadiness).

However, for Cohesiveness, DimenFix-t-SNE either outperforms regular t-SNE by a large margin, or achieves comparable results as regular t-SNE. By looking at the embeddings created by DimenFix-t-SNE, we notice that the embeddings collapse along the fixed dimension (Figure 5.2b), which might be the reason behind this. When a randomly-picked feature is chosen as the fixed axis in the output, other variables get compressed into lines corresponding to the fixed feature



(a) Continuity (b) Trustworthiness (c) InterContinuity (d) Cohesiveness (e) Steadiness

Figure 5.3: Comparison of regular t-SNE to DimenFix-t-SNE, fixing randomly picked quantitative axes. For each pair of results on the same dataset, the boxplot on the left displays DimenFix results while the right displays regular t-SNE.

values. When the chosen fixed axis is not evenly distributed in value, or has a poor alignment with the inherent cluster structure of the dataset (which is true most of the time), the layout loses intra-class variation across all directions. In that case, points, regardless of which cluster do they originally come from, will be projected tightly together into several dense lines, resulting in higher Cohesiveness even when the structure is in fact highly distorted. When the chosen feature is more evenly distributed in value (Figure 5.2a), the effect is less apparent, some classes can be separated along the free axis, resulting in comparable scores of Cohesiveness with regular t-SNE.

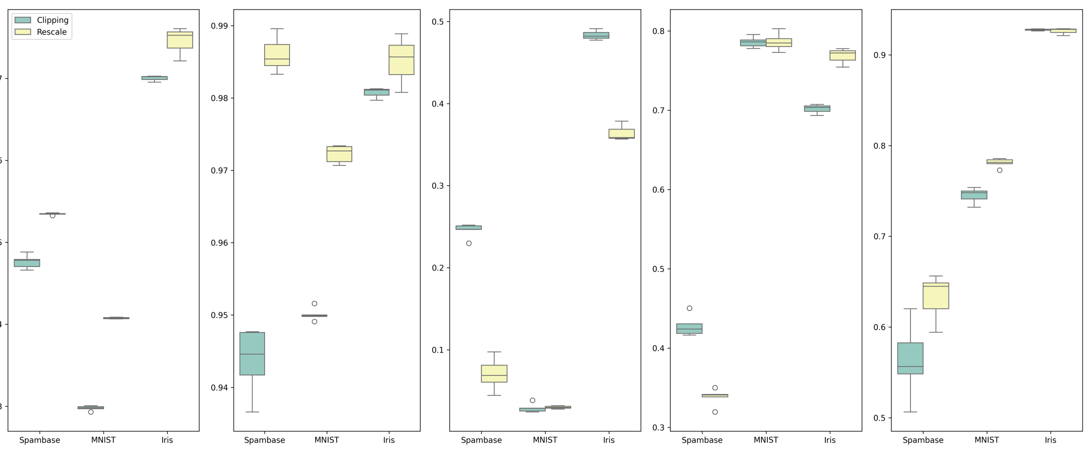
5.2.2 Modes

We measure the quality of embedding created by DimenFix-t-SNE under the Rescale mode. Since the Rescale mode is specially designed for nominal fixed axis, we use class labels as the fixed axis. We use the Iris, Sampled MNIST and Spambase datasets. Iris and MNIST have well-seperated cluster structures, matching with their labels, while Spambase contains only 2 classes (spam or not spam), and Figure 5.1c shows that the two classes are mixed within different clusters. For each dataset, DimenFix-t-SNE with the Rescale and Clipping modes are applied. For all datasets we use a perplexity of 30 and 1000 iterations. On DimenFix-t-SNE, we set the fix_iter to 20, and class ranges are generated based on density. Classes are mapped to a default order based on labels. To capture the variability of the data, we calculate each embedding 3 times with 3 different random initializations.

Figure 5.4 shows the results by dataset and metric. Each boxplot aggregates the results across all three random runs.

When measuring neighborhood preservation (Continuity and Trustworthiness) and the inter-cluster reliability (Steadiness), the Rescale mode consistently out-performs the Clipping mode on all datasets. This is in line with our expectation, since in allowing the clusters to keep their relative shape during *push*, the method tries to preserve local neighborhood relations. The same can be said for Steadiness, since it extracts clusters from the embedding space and verifies if they form clusters in the high-dimensional space. Clipping may push points under neighboring class labels into a same line, forming false clusters in the embedding space, while Rescale seperates classes by rescaling the whole cluster as a whole, minimizing the formation of false clusters.

For Cohesiveness, however, performance differs according to datasets. Clipping outperforms Rescale on the Spambase dataset, while the two show similar performance on the MNIST dataset, and Rescale outperforms Clipping on the Iris dataset. Cohesiveness measures how well clusters



(a) Continuity (b) Trustworthiness (c) InterContinuity (d) Cohesiveness (e) Steadiness

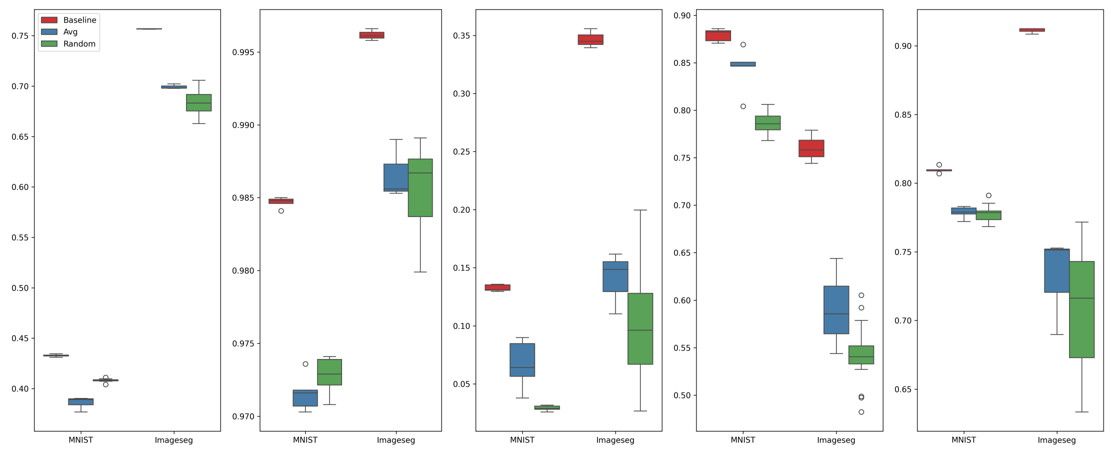
Figure 5.4: Comparison of Clipping and Rescale on DimenFix-t-SNE, using class label as fixed axis. For each pair of results on the same dataset, the boxplot on the left displays the Clipping mode results while the right displays the Rescale mode.

in the high-dimensional space are preserved in the embedding space, in other words, it accounts for missing groups. For the two highly-mixed classes in Spambase, large portions of points need to be moved in order to seperate them. Under the Clipping mode, points from different classes are simply pushed on to one line, collapsing it into a 1d-t-SNE-like distribution, while Rescale works to seperate them by scaling and shifting all points in the two classes. Consequently, Rescale introduces more missing groups since it forces points of different labels in one cluster to seperate, while Clipping might just push part of the points onto the seperating line and leave the rest in place. For the Iris dataset, where the label matches visual clusters nicely, Cohesiveness reaches a higher score because of the overlap between classes allowed by the Rescale mode, which better preserves the inter-cluster structure. On the MNIST dataset, where visual clusters match their labels in general, with marginal mixed portions, effects of the two modes even out each other, resulting in similar scores.

When it comes to measuring InterContinuity, we notice that Rescale performs worse than Clipping except on the MNIST dataset. This can be explained by how InterContinuity is computed: by disregarding points with the same class label as the current instance when calculating Continuity. When DimenFix-t-SNE works to push points into their designated ranges based on class labels, distortions occur because classes are not inherently seperable along the fixed axis. Under the Clipping mode, after one *push*, only points that go outside of their designated ranges are pushed back, while points inside the ranges remain in place. When the Rescale mode is applied, the whole class is rescaled to fit into the range, causing points from neighboring classes to move further away from each other compared with Clipping, thus the decrease in InterContinuity.

5.2.3 Class Ordering

We measure the performance of the class ordering method in DimenFix-t-SNE, when using class label as the fixed axis. We use the Image Segmentation and (Sampled) MNIST datasets since both contain more than five classes. The MNIST dataset has highly separable cluster structures matching the class labels, while in Image Segmentation (shown in Figure 5.1a), visual clusters might contain mixed labels, and data points with the same label can belong to different visual clusters. For each dataset, both class ordering based on average positions and random ordering are applied during DimenFix. Regular t-SNE is also applied to both datasets as the baseline. On both datasets we use a perplexity of 30 and 1000 iterations. We set fix_iter to 20 when applying DimenFix-t-SNE, and use the Rescale mode to better preserve the original shapes of clusters. To



(a) Continuity (b) Trustworthiness (c) InterContinuity (d) Cohesiveness (e) Steadiness

Figure 5.5: Comparison of Average class ordering method and Random order on DimenFix-t-SNE, using class label as the fixed axis, under the Rescale mode. As well as the regular t-SNE results. For each group of results on the same dataset, the leftmost one stands for baseline t-SNE, the middle one with Average class ordering and the right one using Random order.

capture the variability of the data, we calculate each embedding 3 times with 3 different random initializations. DimenFix-t-SNE with random ordering is run 5 times under each initialization to account for the variability introduced by the random class orderings.

Figure 5.5 shows the results by dataset and metric. Each boxplot aggregates the results of the selected ordering method across all random runs.

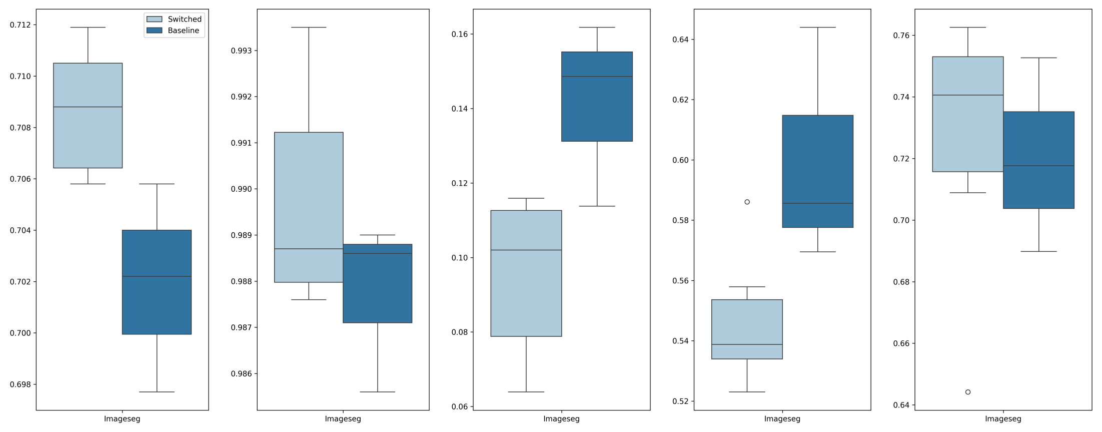
Results show that the Average class ordering method outperforms Random ordering consistently on metrics measuring inter-cluster structures (InterContinuity, Cohesiveness and Steadiness). This proves the proposed ordering method is able to preserve the global structure of data, enhancing inter-cluster reliability. The order produced by this method, though not necessarily optimal, is on average better than using a random order.

When measuring neighborhood preservation (Continuity and Trustworthiness), however, results differ between datasets. On the Image Segmentation dataset, Average class ordering still outperforms most results generated with Random order. While with the MNIST dataset, Random ordering produces better results in general. One possible explanation for this is that class centroids after early exaggeration reflect global cluster layout, but not local overlap. And the PCA applied to align the largest variance with class orders may violate neighborhood preservation, especially across class boundaries.

5.2.4 Switching

We evaluate the switching method by comparing embeddings generated directly by DimenFix-t-SNE from random initialization, using feature A as the fixed axis, with embeddings switched from fixing axis B to axis A. The Image Segmentation dataset is used, since it contains semantically meaningful input features as well as multiple class labels.

Two quantitative input features (A, B) are chosen as the initial fixed axis. We first run a DimenFix-t-SNE fixing the quantitative features for 700 iterations using the Clipping mode, then switch to fixing class label with Average ordering under the Rescale mode, and run 300 iterations (containing both the switching process and normal DimenFix). For the baseline we run DimenFix-t-SNE fixing class label for 1000 iterations, and apply Average class ordering with the Rescale mode. We use a perplexity of 30 and set fix_iter to 20. To capture the variability of the data, we calculate each embedding 3 times with 3 different random initializations.



(a) Continuity (b) Trustworthiness (c) InterContinuity (d) Cohesiveness (e) Steadiness

Figure 5.6: Comparison of DimenFix-t-SNE generated from random initialization fixing class label (right) and switched from a quantitative input axis to fixing class label (left).

Figure 5.6 shows the results by dataset and metric. Each boxplot aggregates the results across all random runs.

Difference between the switched-from-another-axis embedding and the baseline is in fact marginal for Continuity, Trustworthiness and Steadiness. For InterContinuity, the switched embedding scores slightly less than the regular one. This is because the final class order in the switched embedding depends highly on the class layout of the "fixing quantitative axis" embedding. This order may not be as good as the order produced by the Average class ordering method.

For Cohesiveness, the regular DimenFix embedding outperforms the switched one. The switched embedding's structure is dependent of the embedding X which it is switched from. Some cluster may be separated into smaller clusters during switching, thus the decrease in Cohesiveness. By choosing a suitable compression rate β, we can allow the embedding to develop more freely during switching, and decrease the impact of switching on the embedding quality.

With the above observations, we can still conclude that by applying switching we are able to create embeddings with similar quality as the directly generated ones, as well as cutting down the number of iterations needed to achieve that result.

5.3 User-Based Evaluation

Metrics indicate the objective quality of embeddings; we supplement that by conducting a user-based evaluation involving participants. The purpose is to measure how the embeddings produced using DimenFix-t-SNE would influence user's performance on visual cluster analysis tasks, their confidence in their answers as well as their cognitive loads during the tasks. Subjective comments on the embeddings are also collected, which provides additional insight into user experience.

5.3.1 Experiment Design

Participants

18 participants are recruited for the user experiment. Experiments conducted by Lewis et al. [21] indicate that expert users are reasonably consistent judges of embedding quality, while novice users tend to disagree with one another. We thus expect participants to have basic knowledge of the t-SNE algorithm, and ask them to identify their level of familiarity with both general DR techniques and t-SNE. Figure 5.7 shows the overall distribution. Most participants have at least a basic understanding of DR techniques and have at least heard of the t-SNE algorithm.

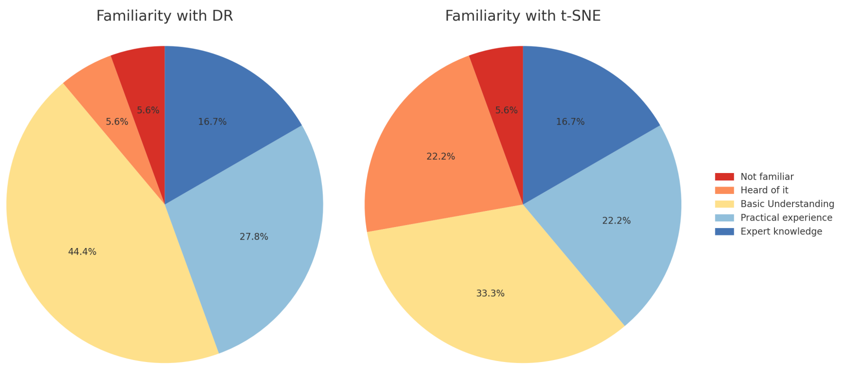
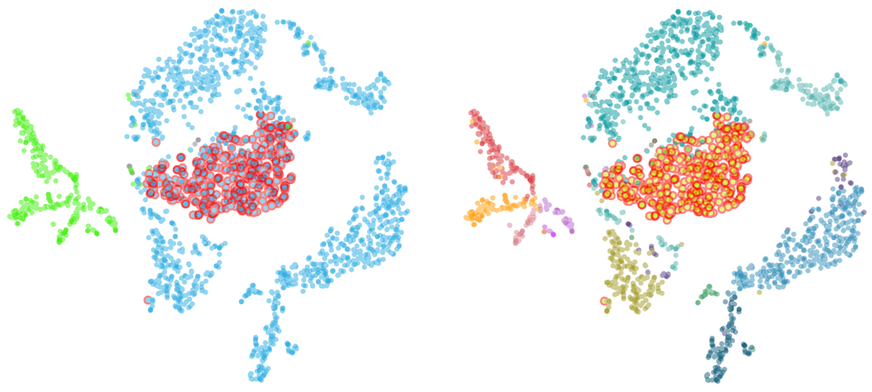


Figure 5.7: Participants' level of familiarity with DR and t-SNE.



(a) Class label NWCS. (b) Subclass label NWCS.

Figure 5.8: Task 1 using regular t-SNE, selection of subclass X highlighted in red circles.

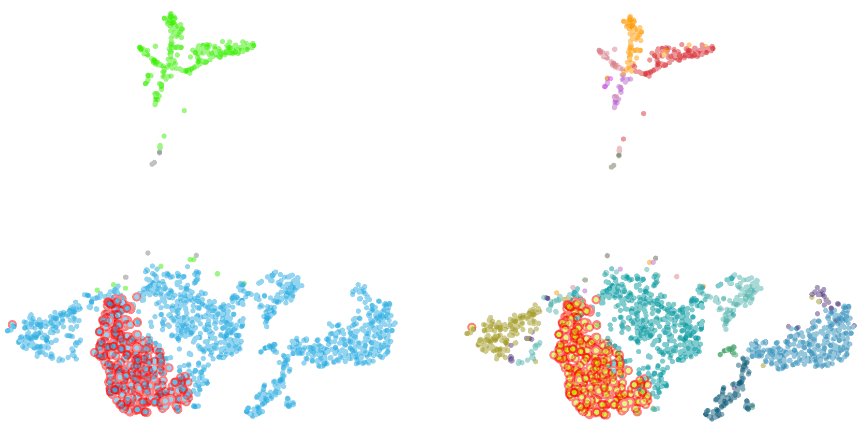
Tasks

Participants are asked to complete three types of visual cluster analysis related tasks using 2D embeddings produced by either regular t-SNE (RT) or DimenFix-t-SNE (DT). Tasks are designed to evaluate how well the visualizations support visual cluster interpretation goals.

Task 1: Identifying Hierarchical Relationships (T1). We intend to assess how well DT embeddings conveys multi-level grouping when fixing class labels on one axis. In this task, we ask participants to identify hierarchical relationships represented by cluster labels in the data. The two datasets used for this task are single-molecule FISH (smFISH) and Comparative single-nucleus profiling of motor cortex, both containing hierarchical cluster labels: class and subclass. Participants are instructed to create 2d embeddings using either RT or DT (fixing class label), and asked the question: "Which class does the subclass X belong to?" They are restricted to using a single view for the embedding, so no side-by-side views of the same embedding under different label coloring are allowed.

Figure 5.8 shows an example on the smFISH dataset applying regular t-SNE, colored separately with class label NWCS (Figure 5.8a) and subclass label NWCS (Figure 5.8b), with the subclass-X highlighted with red circles. Figure 5.9 shows the smFISH dataset after applying DimenFix-t-SNE, when using class label NWCS as the fixed axis. Here instead of defining the range size for each class based on its size, a uniform size for every class is applied. The Rescale mode is applied to preserve the shape of clusters. It is apparent that subclass-X belongs to the blue class in Figure 5.8a and Figure 5.9a.

Task 2: Matching Clusters Across Embeddings (T2). This task evaluates how well



(a) Class label NWCS. (b) Subclass label NWCS.

Figure 5.9: Task 1 using DimenFix-t-SNE, using class label NWCS as fixed axis, uniform bin size for each class, using the Rescale mode, selection of subclass X highlighted in red circles.

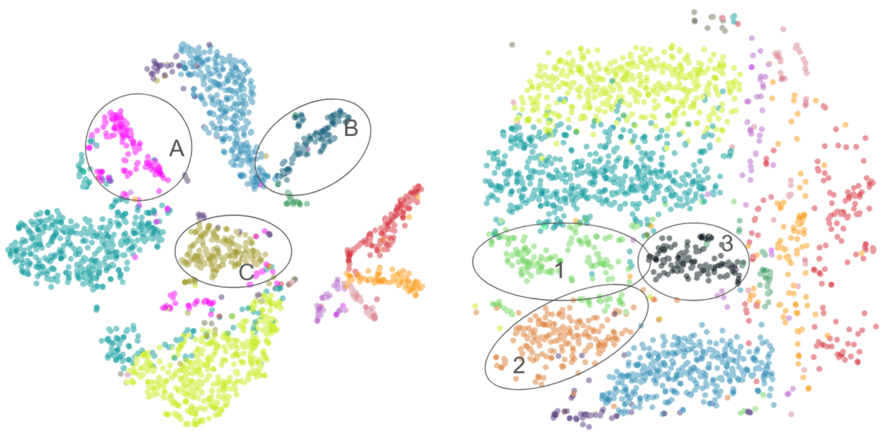


Figure 5.10: Regular t-SNE and DimenFix-t-SNE embeddings used for matching in Task 2, colors of the circled clusters shuffled.

DT embeddings support semantic consistency and preserve inter-cluster relationships, and if participants are able to identify corresponding clusters between different visualizations (DT and RT) of the same dataset.

The smFISH dataset is employed in this task. smFISH comprises of spatially resolved single-molecule FISH measurements captured across layered tissue sections. The actual spatial layout of the tissue is represented by the numerical metadata fields *spatial_X* and *spatial_Y*. In this task, the participants are presented with two embeddings side-by-side: one regular t-SNE embedding and one created by DimenFix-t-SNE, using *spatial_Y* as the fixed axis. Known corresponding clusters are colored identically, while we picked three clusters (X, Y and Z) and shuffled their colors. Figure 5.10 shows the two embeddings used in this task. We then ask the participants: "Which cluster in embedding B corresponds to cluster X in embedding A?"

Task 3: Identifying Cluster Features (T3). We aim to test if DT can add to the interpretability of embeddings by explicitly preserving input features. Participants are asked to identify specific traits of clusters in the embeddings corresponding to the selected input feature.

Two datasets are used in this task: smFISH and Image Segmentation. smFISH contains a layered spatial layout information, indicated by the metadata field *spatial_Y*. The Image Segmentation dataset contains input features which reflect specific traits of datapoints, such as average hue

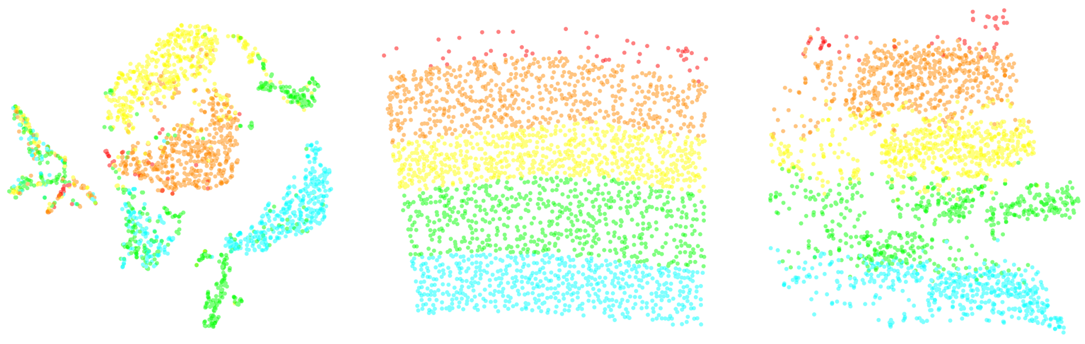

(a) Regular t-SNE (b) Spatial layout (c) DimenFix-t-SNE

Figure 5.11: Embeddings generated with the smFISH dataset, using regular t-SNE (left), displaying spatial layout according to *spatial_X* and *spatial_Y* (middle), and using DimenFix-t-SNE, fixing *spatial_Y*. All 3 colored by layer labels.

and average saturation. In this task, participants are instructed to create 2d embeddings using either RT or DT (fixing *spatial_Y* or *saturation_mean*), and asked the questions: "In which layers does cluster X appear?" and "Select the average saturation level of cluster Y."

Figure 5.11 shows examples of visualizations used for Task 3. All three embeddings are colored with the layer information input as labels. Figure 5.11b shows the physical layout of the points, according to their *spatial_X* and *spatial_Y*. Layers are divided based on the field *spatial_Y*. Figure 5.11a shows the regular t-SNE embedding colored with the layer label, and Figure 5.11c shows the dataset after applying DimenFix-t-SNE, using *spatial_Y* as the fixed quantitative axis. The Clipping mode is used in this example to make the layout of points correspond better with the layers in Figure 5.11b.

Study Design

The user study contains three sections: an introduction section, task section 1 and task section 2. In the introduction section, we introduce the user to the basic principle of DimenFix and guide the user through the usage of the DimenFix-t-SNE plugin in ManiVault. Both task sections 1 and 2 contain visual cluster analysis related tasks. More specifically, they both contain three T1 questions and three T3 questions based on different datasets, while task section 1 has an extra T2 question by the end.

Participants are divided equally into two groups for the task sections: Group A would be asked to use RT in task section 1 and use DT in task section 2, and Group B the opposite. Note that in T2, both RT and DT embeddings are presented to the participant. Consequently, we do not explicitly define which technique the participant is using in T2.

In each task section, participants are asked to:

1. Generate embeddings using RT or DT and answer the visual cluster related question;

2. Rate their confidence in the answer on a 1-5 Likert scale;

3. Report their perceived cognitive load during the section, based on a simplified version of the NASA-TLX questionnaire [13].

We collect screen recordings throughout the task sessions to capture participants' interactions and task-solving behavior. After the task sections, participants were also invited to provide open-ended feedback on the embedding quality and interpretability. The full questionnaire used for the study can be found in Appendix A.

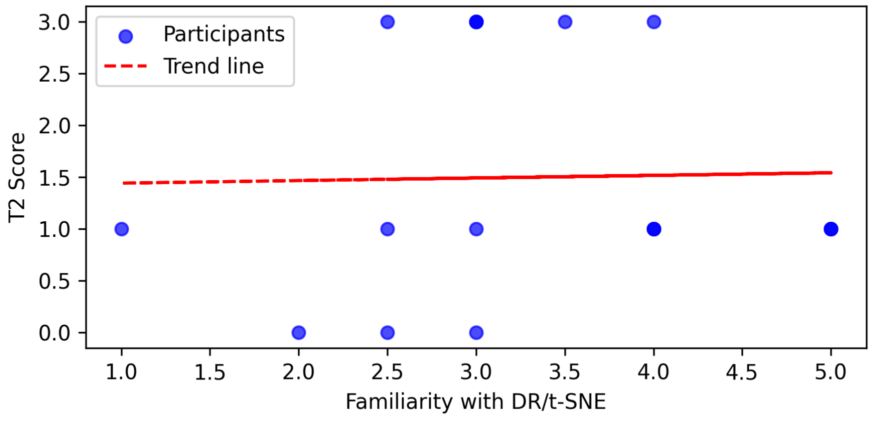


Figure 5.12: Participants' level of familiarity with DR and t-SNE against T2 Scores.

5.3.2 Results

In this section, we report the results of participants' task performance, self-rated confidence, perceived cognitive load, and subjective feedback. Where applicable, comparisons are made between t-SNE and DimenFix-t-SNE conditions, with attention to differences across task types and task sections.

Task Performance

DT's influence on participants' task performance is measured based on the accuracy of their answers. We report the results categorized by the tasks defined in Section 5.3.1.

In both task section 1 (S1) and section 2 (S2), 100% accuracy was observed for T1 (identifying hierarchical relationships) and T3 (identifying cluster features) across both groups, with one exception: in S2 T3, Group B had a 96% accuracy rate, with only one incorrect answer out of 27 responses. This minor deviation is considered marginal and does not affect the overall interpretation.

Given this ceiling effect on T1 and T3, we focus our analysis on T2 (matching clusters across embeddings). The overall accuracy for this matching task is 50%, which is significantly lower than the near-perfect accuracy observed in T1 and T3. Also, the average confidence score in the answer is 2.06 (out of 5). This result suggests that matching clusters across different embeddings is substantially more challenging, even for participants with prior knowledge of t-SNE and dimensionality reduction techniques. The 50% accuracy reflects the inherent difficulty in aligning structures when the layouts differ visually, suggesting a limitation of DimenFix-t-SNE in preserving cluster relationships.

We further examine whether participants' familiarity with DR and t-SNE influences their performance on the cluster matching task (T2). Figure 5.12 shows each participant's familiarity level with DR and t-SNE (2 scores averaged) against their scores in T2 (discrete, 0-3), revealing no apparent trend. This is confirmed by a Spearman correlation test ($\rho = 0.12, p = 0.62$), indicating that knowledge and experience with DR or t-SNE do not significantly affect task performance in T2.

Confidence Level

We aim to examine how DT influence participants' confidence in their answers while completing different tasks. A mixed ANOVA is conducted to examine the effects of tool (DT or RT) and task (T1 and T3) on participants' self-rated confidence levels. Task order and tool order are also taken into account. Results from T2 are excluded since the task setting, with participants looking at both RT and DT embeddings side by side, does not align with T1 and T3.

The analysis reveals a significant main effect of condition ($F(3, 48) = 14.32, p < .001, \eta^2 = 0.47$), indicating that confidence levels vary significantly across the four (tool, task) combinations. No significant main effect of group is found ($F(1, 16) = 0.99, p = .335, \eta^2 = 0.06$), indicating that participants in Group A and Group B do not differ in their overall confidence ratings, regardless of the tool or task condition. This suggests that whether participants use DT or RT first does not have an influence on their overall confidence levels. The interaction between group and condition is not significant either ($F(3, 48) = 0.41, p = .747, \eta^2 = 0.03$), suggesting that the pattern of confidence levels across conditions is consistent across both groups. The results suggest that the variation of confidence level between different (tool, task) combinations is not caused by the order in which the tools are used.

We run a post-hoc pairwise comparison with Bonferroni correction between confidence levels in each pairs of (tool, task) combinations. Results show that confidence level in (RT, T3) is significantly higher than that in (RT, T1), with $t(17) = 5.27, p < .001, g = 1.15$. Additionally, confidence level in (DT, T3) is significantly higher than that in (RT, T3), with ($t(17) = 3.37, p = .022, g = 0.98$). No significant difference is found between other pairs of combinations. This indicates that when completing T3, participants are more confident when using DT than RT, but for T1, no significant effect of different tools can be found.

Cognitive Load

To measure the influence of DT on participants' perceived cognitive load, we conduct a series of statistical analyses on NASA-TLX scores collected from participants during the experiment. The cognitive load is measured for each section of tasks, and the raw average score for each measurement is used for the analysis.

Overall, when using DT ($M = 8.28, SD = 2.65$) during the tasks, participants report slightly lower workload ratings than using RT ($M = 9.06, SD = 2.26$). However, the difference was not statistically significant, as shown by both a paired t-test ($t(17) = 0.96$, $p = 0.351$) and a Wilcoxon signed-rank test ($W = 63.5, p = 0.369$). We can conclude that there's no strong evidence that using DT in tasks would increase or reduce cognitive load compared to using RT.

However, within-subject comparisons reveal a significant difference in Group A, where participants report significantly higher workload when using RT (Section 1) compared to DT (Section 2), with $t(8) = 7.35$, $p < 0.001$. No such effect is observed in Group B, where $t(8) = 1.39$, $p = 0.202$.

To determine if the difference in cognitive load is induced by order (using DT in section 1, and then RT in section 2, or the opposite) or not, a between-groups analysis is conducted. Results show that there is no significant difference in overall workload between Group A ($M = 8.44$, $SD = 2.77$), who used RT first, and Group B ($M = 8.89$, $SD = 2.17$), who used DT first, $t(32.13) = -0.54$, $p = 0.596$.

Furthermore, a comparison between tasks indicates that tasks in Section 1 ($M = 9.83$, $SD = 2.41$) induce significantly higher workload than tasks in Section 2 ($M = 7.50$, $SD = 1.95$). This is proven by both a paired t-test, $t(17) = 3.82$, $p = 0.001$, and Wilcoxon test, $W = 18.0$, $p = 0.002$.

Above findings suggest that while the overall effect of DT on workload was not statistically significant, the content of the tasks themselves appears to contribute substantially to differences in participants' reported cognitive load.

We can identify two major issues in the experiment design reflected by this finding: 1. Measurement time, participants' cognitive loads are measured twice, after each section, however, the cognitive load for task 3 is included in task section 1, causing false loads to be recognized in task section 1; 2. Unbalanced tasks, different datasets are used for the same tasks in different sections. Although we tried to pick datasets with similar features, difference in their sizes and structures still causes the tasks to be substantially different.

Subjective Feedback

We ask the participants to provide feedback on the DT embeddings they've seen and used during the tasks, and used the following prompt questions for inspiration:

- Are similar items placed close together, in your opinion?
- Is the overall layout easy to understand? Why or why not?
- Can you make sense of the position of the points or classes?
- Does this embedding help you understand the data or categories better?
- Did anything about the layout surprise you or seem incorrect?

Several terms appear repeatedly in the feedback, which can be categorized into three themes: 1. clarity and cluster separation; 2. understanding and trusting the embedding; 3. interface layout and coloring.

Clarity and cluster separation. The term "clear" is mentioned 10 times, and usually with the term "difference" or "separability". When completing T1, participants tend to find DT embeddings clearer than RT embeddings, with classes divided by sharp boundaries (Clipping mode) or separated by their vertical positions (Rescale mode). The same can be said for T3, when asked to identify the clusters' in layers: "It was very clear to me which classes and layers belong together. I definitely felt more secure and could understand the data a lot easier." However, in T2, when the fixed axis in the DT embedding is *spatial_Y*, the cluster structure is less preserved compared with the RT embedding. Participant mentioned: "Most clusters remained tightly grouped, but the ones on the right in the original image (red, orange, etc.) become less clear." and "left view (RT) seems more a little more clear since classes have more separability." This is also apparent in Figure 5.11c, which can be seen as a trade-off for explicitly preserving the spatial information.

Understanding and trusting the embedding. The term "understand" appeared 11 times in the comments, accompanying the terms "helpful" and "trust", as well as "confusion". Participants in general agree that the embedding created by DT (fixing class label) is helpful and understandable: "binning the plot with some input dim always helps more to identify the label classes." Some find the DT embedding more useful when presented alongside a RT embedding, "because you can 'map out' some specific axis/dimension". This suggests that the DT embedding can provide complementary information to the corresponding RT embedding, and in that way enhancing user's understanding of the data.

However, DT also causes confusion by creating artifact in the embedding. Horizontal lines between classes which are generated by the Clipping mode can "look wierd". On the other hand, some find the lines "make it easy to identify clusters". While one participant prefers Rescale to Clipping because "the distribution remains somewhat the same", some classes can be "squashed" and "one tends to ignore data point density in each individual bin". False information can be induced because of insufficient understanding of the t-SNE algorithm: "If you do not understand exactly what method was used to place items close together, that can lead to false information being provided."

One thing also worth noting is that familiarity with the application also influence participants' understanding and confidence in the embeddings. Some participants are also confused half way through the tasks because they are not very familiar with DimenFix.

Coloring. The term "color" is mentioned 9 times. Most participants agree that coloring is helpful for identifying different clusters: "Similar items were mostly clustered together, with color coding helping to distinguish distinct clusters from each other." However, one participant also mentions that with color deficiency, identifying clusters becomes harder. Also when completing T2, where participants have to match clusters based on inter-cluster relationships provided by identical coloring of corresponding clusters, this problem becomes more apparent. This suggests an improvement in study design and more detailed considerations when recruiting participants.

Chapter 6

Conclusion and discussion

In this work, we propose DimenFix-t-SNE, a more interpretable t-SNE extending from the meta-stategy DimenFix. By explicitly preserving input information in one of the output axes, the anonymous nature of t-SNE embedding's axes is altered to introduce interpretability. Furthermore, we generalize the method by adding user-controllable parameters and flexible range limits. The performance of DimenFix on fixing nominal inputs such as class labels is improved by utilizing the global structure revealed by early exaggeration in t-SNE, rendering a reasonable ordering of nominal features and improving the inter-cluster quality of embeddings. Finally, a gradual axis switching method that allows users to track the movement of points or clusters when switching between different fixed axes is implemented to aid them in the interactive data exploration process.

We evaluate the performance of DimenFix-t-SNE based on both quantitative metrics and a user study. Results indicate that though DimenFix-t-SNE reduces the objective quality of embeddings compared to regular t-SNE, DimenFix-t-SNE embeddings are clearer and more helpful to users when faced with specific tasks such as identifying hierarchical relationships in clusters. The proposed ordering method also outperforms random ordering with regard to inter-cluster reliability and inter-class continuity. The gradual axis switching method, enabling trackable switching between axes, is also proven to result in embeddings of equal quality compared with the directly-generated ones.

DimenFix-t-SNE provides intrinsic explainability for t-SNE embeddings in a generalizable manner, without altering the gradient descent process or metrics. The application of DimenFix-t-SNE is not limited to a certain type of data, since no prior knowledge is needed. Compared with existing class-aware t-SNE variants, DimenFix-t-SNE does not take into account class label information during the gradient descent process, but use labels as extra input to separate classes explicitly. Clear visual differentiation as well as preservation of the clusters' original shapes (using the Rescale mode) are achieved.

DimenFix-t-SNE also proves that the meta-strategy DimenFix can be easily adapted to gradient descent DR techniques. Generalizations proposed in this work are also adaptable regardless of the DR technique used, since DimenFix is minimaly intrusive, only requiring access to the embedding coordinates after gradient descent. The proposed class ordering method, however, is dependent of t-SNE's early exaggeration process. Yet generalization to other techniques is still possible if prior knowledge exists to provide a crude global structure or class order of the input dataset.

This work has certain limitations. First, the proposed class ordering method is dependent of the embedding structure after early exaggeration, which further depends on the random initialization of the projected space. Our experiments show that the final order of classes varies across different random initializations, indicating that the method is not stable and the optimal class order for a certain dataset can only be approximated instead of determined. Class orders are shifted during later iterations based on the positions of class centroids, which are sensitive to outliers, introducing more instability. Further, a more delicate and efficient *push* strategy could be designed to minimize the distortion to the embedding introduced by every push. The coarse angular search to align the update direction with the global structure introduces an extra $O(N)$ complexity to every push, and does not account for other transformations that may also reduce push energy, limiting its ability to fully adapt to complex data structures. Evaluations could be more comprehensive for the proposed switching method. Although existing results suggest good performance, further testing of the method on more datasets and more complex settings would strengthen the conclusions. The method can also benefit from a well-designed user study. By utilizing switching in data-exploration related tasks we can measure if the method actually enhances users' understanding of datasets and aids in their exploration process.

Finally, while the user study provides useful insights, certain components can be improved. It would benefit from recruiting more participants with expert experience in t-SNE and dimensionality reduction. Also, the task design did not fully cover the range of activities typically involved in visual cluster analysis, potentially limiting the generalizability of the findings.

Nonetheless, DimenFix-t-SNE is a valuable and practical enhancement to the traditional t-SNE technique by introducing interpretability without compromising flexibility or generality. Despite areas for improvement in evaluation and study design, the results demonstrate that DimenFix-t-SNE enables clearer visual structures, supports meaningful cluster interpretation, and provides user-controllable guidance during exploratory analysis. Its minimal intrusiveness and compatibility with gradient descent-based DR techniques make it a promising candidate for broader applications.

Bibliography

[1] Image Segmentation. UCI Machine Learning Repository, 1990. [Online]. Available: https://doi.org/10.24432/C5GP4N. 15, 25

[2] Michaël Aupetit, Michael Sedlmair, Mostafa M. Abbas, Abdelkader Baggag, and Halima Bensmail. Toward perception-based evaluation of clustering techniques for visual analytics. In *2019 IEEE Visualization Conference (VIS)*, pages 141–145, 2019. 10

[3] Shaeela Ayesha, Muhammad Kashif Hanif, and Ramzan Talib. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59:44–58, 2020. 1

[4] Trygve E. Bakken, Nikolas L. Jorstad, Qiwen Hu, Blue B. Lake, Wei Tian, Brian E. Kalmbach, Megan Crow, and et al. Comparative cellular analysis of motor cortex in human, marmoset and mouse. *Nature*, 598(7879):111–119, October 2021. 25

[5] Adrien Bibal, Viet Minh Vu, Géraldin Nanfack, and Benoît Frénay. Explaining t-sne embeddings locally by adapting lime. In *The European Symposium on Artificial Neural Networks*, 2020. 1, 9

[6] Kerstin Bunte, Michael Biehl, and Barbara Hammer. A general framework for dimensionality-reducing data visualization mapping. *Neural Computation*, 24(3):771–804, 2012. 11

[7] Angelos Chatzimparmpas, Rafael M. Martins, and Andreas Kerren. t-visne: Interactive assessment and interpretation of t-sne projections. *IEEE Transactions on Visualization and Computer Graphics*, 26(8):2696–2714, 2020. 1, 9

[8] Wenqiang Cui. Visual analytics: A comprehensive overview. *IEEE Access*, 7:81555–81573, 2019. 1

[9] Cyril de Bodt, Dounia Mulders, Daniel López Sánchez, Michel Verleysen, and John A Lee. Class-aware t-sne: cat-sne. In *ESANN*, pages 409–414, 2019. 1, 10

[10] Daniel Engel, Lars Hüttenberger, and Bernd Hamann. A Survey of Dimension Reduction Methods for High-dimensional Data Analysis and Visualization. In *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*, pages 135–149, 2012. 1

[11] Ronak Etemadpour, Robson Motta, Jose Gustavo de Souza Paiva, Rosane Minghim, Maria Cristina Ferreira de Oliveira, and Lars Linsen. Perception-based evaluation of projection methods for multidimensional data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(1):81–94, 2015. 11

[12] Henry Han, Tianyu Zhang, Chun Li, Mary Lauren Benton, Juan Wang, and Junyi Li. Explainable t-sne for single-cell rna-seq data analysis. *bioRxiv*, 2022. 1, 10

[13] Sandra G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in Psychology*, volume 52, pages 139–183. 1988. 34

[14] Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In *Neural Information Processing Systems*, 2002. 3

[15] Reeber Erik Forman George Hopkins, Mark and Jaap Suermondt. Spambase. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C53G6X. 25

[16] Hyeon Jeon, Hyung-Kwon Ko, Jaemin Jo, Youngtaek Kim, and Jinwook Seo. Measuring and explaining the inter-cluster reliability of multidimensional projections. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):551–561, 2022. 26

[17] I. T. Jolliffe. *Principal Component Analysis.* Springer Series in Statistics. Springer New York, NY, 2 edition, 2002. 1

[18] Bo Kang, Dario Garcia Garcia, Jefrey Lijffijt, Raúl Santos-Rodríguez, and Tijl De Bie. Conditional t-sne: more informative t-sne embeddings. *Machine Learning*, 110:2905–2940, 2021. 10

[19] Joseph B. Kruskal and Myron Wish. *Multidimensional Scaling*, volume 11 of *Quantitative Applications in the Social Sciences.* SAGE Publications, 1978. 1

[20] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. 4

[21] JM Lewis, LJP van der Maaten, and VR de Sa. A behavioral investigation of dimensionality reduction. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, pages 671–676. Cognitive Science Society, 2012. 31

[22] George C. Linderman and Stefan Steinerberger. Clustering with t-sne, provably. *CoRR*, abs/1706.02582, 2017. 11

[23] Brian Long, Jeremy Miller, and The SpaceTx Consortium. Spacetx: A roadmap for benchmarking spatial transcriptomics exploration of the brain, 2023. 25

[24] Qiaodan Luo, Leonardo Christino, Fernando V. Paulovich, and Evangelos E. Milios. Dimenfix: A novel meta-dimensionality reduction method for feature preservation. *arXiv*, 2022. 1, 5, 6

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 1

[26] Linhao Meng, Stef van den Elzen, Nicola Pezzotti, and Anna Vilanova. Class-constrained t-sne: Combining data features and class probabilities. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):164–174, 2024. 1, 10

[27] Christoph Molnar. *Interpretable Machine Learning.* 2018. 9

[28] Nicola Pezzotti, Thomas Höllt, Boudewijn P. F. Lelieveldt, Elmar Eisemann, and Anna Vilanova. Hierarchical stochastic neighbor embedding. *Computer Graphics Forum*, 35:21–30, 2016. 6

[29] Nicola Pezzotti, Boudewijn PF Lelieveldt, Laurens van der Maaten, Thomas Höllt, Elmar Eisemann, and Anna Vilanova. Approximated and user steerable tsne for progressive visual analytics. *IEEE transactions on visualization and computer graphics*, 23(7):1739–1752, 2017. 6

[30] Nicola Pezzotti, Alexander Mordvintsev, Thomas Höllt, Boudewijn P. F. Lelieveldt, Elmar Eisemann, and Anna Vilanova. Linear tsne optimization for the web. *CoRR*, abs/1805.10817, 2018. 7

[31] Harun Ur Rashid. Epileptic seizure recognition. `https://www.kaggle.com/datasets/harunshimanto/epileptic-seizure-recognition`, 2018. Accessed: 2025-06-05. 25

[32] Anguita Davide Ghio Alessandro Oneto Luca Reyes-Ortiz, Jorge and Xavier Parra. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C54S4K. 19

[33] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016. 9

[34] ManiVault Studio. t-sne and hsne analysis plugin for manivault. `https://github.com/ManiVaultStudio/t-SNE-Analysis`, 2025. Accessed: 2025-06-01. 7

[35] Eduardo Tejada, Rosane Minghim, and Luis Gustavo Nonato. On improved projection techniques to support visual exploration of multi-dimensional data sets. *Information Visualization*, 2(4):218–231, 2003. 1

[36] Jarkko Venna and Samuel Kaski. Neighborhood preservation in nonlinear projection methods: An experimental study. In *International Conference on Artificial Neural Networks (ICANN)*, pages 485–491. Springer, 2001. 15

[37] Elio Ventocilla and Maria Riveiro. A comparative user study of visualization techniques for cluster analysis of multidimensional data sets. *Information Visualization*, 19(4):318–338, 2020. 11

[38] Alexander Vieth, Thomas Kroes, Julian Thijssen, Baldur van Lew, Jeroen Eggermont, Soumyadeep Basu, Elmar Eisemann, Anna Vilanova, Thomas Höllt, and Boudewijn Lelieveldt. Manivault: A flexible and extensible visual analytics framework for high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):175–185, 2024. 2, 7

[39] Viet Minh Vu, Adrien Bibal, and Benoît Frénay. Hct-sne: Hierarchical constraints with t-sne. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. 1, 10

[40] John Wenskovitch, Ian Crandell, Naren Ramakrishnan, Leanna House, Scotland Leman, and Chris North. Towards a systematic combination of dimension reduction and clustering in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):131–141, 2018. 10

[41] Jiazhi Xia, Linquan Huang, Weixing Lin, Xin Zhao, Jing Wu, Yang Chen, Ying Zhao, and Wei Chen. Interactive visual cluster analysis by contrastive dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):734–744, 2023. 11

[42] Jiazhi Xia, Yuchen Zhang, Jie Song, Yang Chen, Yunhai Wang, and Shixia Liu. Revisiting dimensionality reduction techniques for visual cluster analysis: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):529–539, 2022. 1, 10, 11

[43] Zelin Zang, Shenghui Cheng, Hanchen Xia, Liangyu Li, Yaoting Sun, Yongjie Xu, Lei Shang, Baigui Sun, and Stan Z. Li. Dmt-ev: An explainable deep network for dimension reduction. *IEEE Transactions on Visualization and Computer Graphics*, 30(3):1710–1727, 2024. 10

Appendix A

User Experiment Questionnaire

DimenFix user test - Group B (d + reg)

* Indicates required question

1. Participant Number

____

Background

This section measures your familiarity with DR techniques

2. **How familiar are you with dimensionality reduction techniques (e.g., PCA, t-SNE, UMAP)?** *

   (Select the option that best matches your experience.)

   Mark only one oval.

   - ◯ Not familiar: I've never heard of dimensionality reduction or don't know what it means.
   - ◯ Heard of it: I've heard the term(s) (e.g., PCA, t-SNE), but I'm not sure how they work or when to use them.
   - ◯ Basic understanding:I understand what dimensionality reduction is and know common methods like PCA or t-SNE, but I don't use them often.
   - ◯ Practical experience:I have used dimensionality reduction techniques in my work or studies, and I understand how they affect data representation.
   - ◯ Expert knowledge: I have in-depth knowledge of multiple DR methods, including their mathematical foundations, strengths/weaknesses, and tuning practices.

3. **How familiar are you with t-SNE?** *

(Select the option that best matches your experience.)

Mark only one oval.

- ◯ Not familiar: I've never heard of dimensionality reduction or don't know what it means.
- ◯ Heard of it: I've heard the term(s) (e.g., PCA, t-SNE), but I'm not sure how they work or when to use them.
- ◯ Basic understanding:I understand what dimensionality reduction is and know common methods like PCA or t-SNE, but I don't use them often.
- ◯ Practical experience:I have used dimensionality reduction techniques in my work or studies, and I understand how they affect data representation.
- ◯ Expert knowledge: I have in-depth knowledge of multiple DR methods, including their mathematical foundations, strengths/weaknesses, and tuning practices.

Task 0: Get familiar with DimenFix, scatterplot, coloring and cluster selection

1. Disable Dimenfix, run a regular tsne on iris
2. Color the points with cluster labels, color the points according to sepal width
3. Select a cluster
4. Create a new view and enable Dimenfix, set: push_iter, mode, ordering, input labels, fixed axis
    1. Fix sepal width on the y-axis
    2. Fix class label on the y-axis (no need to test disable or avg ordering, set to disable always)
        1. Enable density range adjustment
        2. Disable
        3. Clipping mode
        4. Gaussian or rescale mode
5. See that selection works across views

Dataset 1

Task 1: Hierarchical relationships

1. Run DimenFix tsne, fixing class NWCS
2. Color with class NWCS
3. Color with subclass NWCS
4. Allowed to switch coloring, but single view only

4. Subclass X belongs to...? *

🟩 Task1-X

Mark only one oval.

◯ class 1 in NWCS

◯ class 2 in NWCS

◯ class 3 in NWCS

5. How confident are you in your answer?

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

6. Subclass Y belongs to...? *

🟦 Task1-Y

Mark only one oval.

◯ class 1 in NWCS

◯ class 2 in NWCS

◯ class 3 in NWCS

7. How confident are you in your answer?

Mark only one oval.

1	2	3	4	5
○	○	○	○	○

8. Subclass Z belongs to...? *

🟥 Task1-Z

Mark only one oval.

○ class 1 in NWCS

○ class 2 in NWCS

○ class 3 in NWCS

9. How confident are you in your answer?

Mark only one oval.

1	2	3	4	5
○	○	○	○	○

Task 2: layers using 2 parallel views

1. Show Numerical metadata, _spatial_X/_spatial_Y
2. Run Dimenfix tsne, fixing _spatial_Y
3. Color with layer
4. Color with subclass NWCS

10. Identify the layers in which subclasses appear. (Tick all that apply) *

Check all that apply.

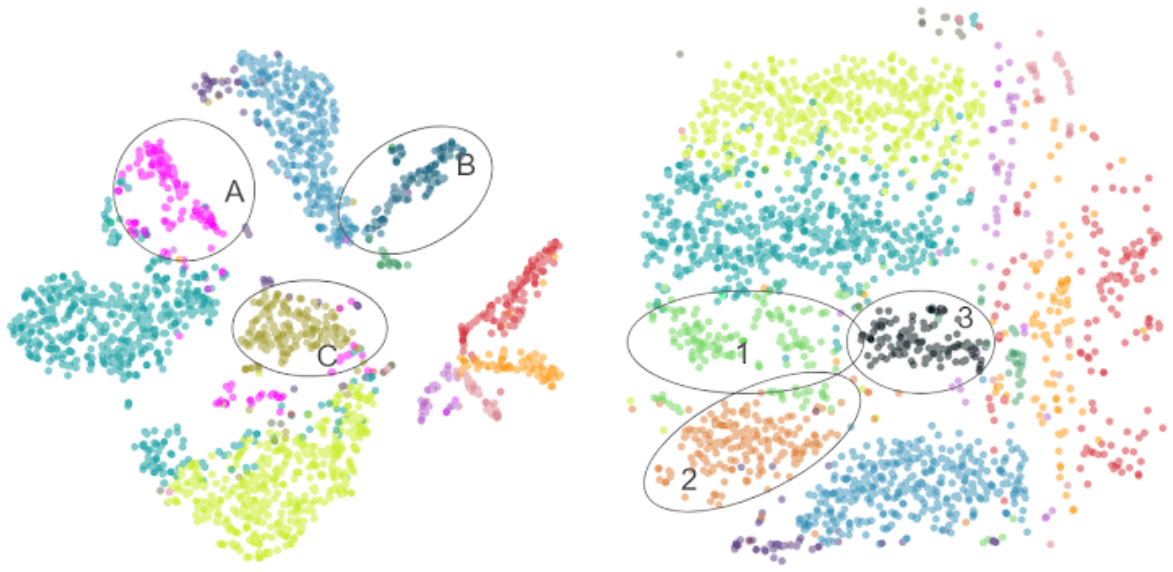
	L1	L2/3	L4	L5	L6
Class A	☐	☐	☐	☐	☐
Class B	☐	☐	☐	☐	☐
Class C	☐	☐	☐	☐	☐

11. How confident are you in your answer? *

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

12. Match the clusters back to their original colors *



Mark only one oval per row.

	class A	class B	class C
Cluster 1	⬭	⬭	⬭
Cluster 2	⬭	⬭	⬭
Cluster 3	⬭	⬭	⬭

13. How confident are you in your answer? *

Mark only one oval.

1	2	3	4	5
⬭	⬭	⬭	⬭	⬭

14. What do you think of the embedding? Feel free to comment on anything you notice or find interesting.
"Are similar items placed close together, in your opinion?"
"Is the overall layout easy to understand? Why or why not?"
"Can you make sense of the position of the points or classes?"
"Does this embedding help you understand the data or categories better?"
"Did anything about the layout surprise you or seem incorrect?"

Cognitive Load

15. How mentally demanding was the task?

Mark only one oval.

	1	2	3	4	5	
	◯	◯	◯	◯	◯	

16. How hurried or rushed was the pace of the task?

Mark only one oval.

	1	2	3	4	5	
	◯	◯	◯	◯	◯	

17. How successful were you in accomplishing what you were asked to do?

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

18. How hard did you have to work to accomplish your level of performance?

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

19. How insecure, discouraged, irritated, stressed, and annoyed were you?

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

Dataset 2

This is the Image Segmentation dataset from UCI. The instances were drawn randomly from a database of 7 outdoor images.
The images were hand-segmented to create a classification for every pixel.

Each instance is a 3x3 region.

Task3: mean saturation of classes

1. Run regular t-sne
2. Color with class
3. Color with SATURATION-MEAN

20. Identify saturation levels

Mark only one oval per row.

	Low	Medium	High	Mixed
Class A	⬭	⬭	⬭	⬭
Class B	⬭	⬭	⬭	⬭
Class C	⬭	⬭	⬭	⬭

21. How confident are you in your answer? *

Mark only one oval.

1	2	3	4	5
⬭	⬭	⬭	⬭	⬭

Dataset 3

1. Run t-SNE, single view
2. color with subclass

22. Subclass X belongs to...? *

🟥 Task4-X

Mark only one oval.

- ◯ Vascular
- ◯ Astro-Epen
- ◯ GABA

23. How confident are you in your answer? *

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

24. Subclass Y belongs to...? *

🟧 Task4-Y

Mark only one oval.

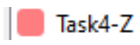
- ◯ Vascular
- ◯ Astro-Epen
- ◯ GABA

25. How confident are you in your answer? *

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

26. Subclass Z belongs to...? *



Mark only one oval.

- ⬭ Vascular
- ⬭ Astro-Epen
- ⬭ GABA

27. How confident are you in your answer? *

Mark only one oval.

1	2	3	4	5
⬭	⬭	⬭	⬭	⬭

28. What do you think of the embedding? Feel free to comment on anything you notice or find interesting.
"Are similar items placed close together, in your opinion?"
"Is the overall layout easy to understand? Why or why not?"
"Can you make sense of the position of the points or classes?"
"Does this embedding help you understand the data or categories better?"
"Did anything about the layout surprise you or seem incorrect?"

Cognitive Load

29. How mentally demanding was the task?

Mark only one oval.

	1	2	3	4	5
	⬭	⬭	⬭	⬭	⬭

30. How hurried or rushed was the pace of the task?

Mark only one oval.

	1	2	3	4	5
	⬭	⬭	⬭	⬭	⬭

31. How successful were you in accomplishing what you were asked to do?

Mark only one oval.

	1	2	3	4	5
	⬭	⬭	⬭	⬭	⬭

32. How hard did you have to work to accomplish your level of performance?

Mark only one oval.

	1	2	3	4	5
	⬭	⬭	⬭	⬭	⬭

33. How insecure, discouraged, irritated, stressed, and annoyed were you?

Mark only one oval.

	1	2	3	4	5
	◯	◯	◯	◯	◯