DOI: 10.1111/cgf.14291
Eurographics Conference on Visualization (EuroVis) 2021
R. Borgo, G. E. Marai, and T. von Landesberger
(Guest Editors)

Volume 40 (2021), Number 3

Guided Stable Dynamic Projections

E. F. Vernier[1,2], J. L. D. Comba[1] and A. C. Telea[3]

[1]Federal University of Rio Grande do Sul, Brazil
[2]University of Groningen, the Netherlands
[3]University of Utrecht, the Netherlands

Abstract

Projections aim to convey the relationships and similarity of high-dimensional data in a low-dimensional representation. Most such techniques are designed for static data. When used for time-dependent data, they usually fail to create a stable and suitable low dimensional representation. We propose two dynamic projection methods (PCD-tSNE and LD-tSNE) that use global guides to steer projection points. This avoids unstable movement that does not encode data dynamics while keeping t-SNE's neighborhood preservation ability. PCD-tSNE scores a good balance between stability, neighborhood preservation, and distance preservation, while LD-tSNE allows creating stable and customizable projections. We compare our methods to 11 other techniques using quality metrics and datasets provided by a recent benchmark for dynamic projections.

CCS Concepts

• ***Computing methodologies*** → ***Dimensionality reduction and manifold learning;***

1. Introduction

Many domains produce datasets with large numbers of observations (also called samples or points) and attributes (also called measurements, dimensions, or variables). Dimensionality reduction techniques, also called projections, are an established tool for visualizing such datasets in a simplified, compact, and scalable way.

The literature on static projections – that address the visualization of time-independent datasets – is quite rich, with many techniques, surveys, and benchmarks on the subject [NA19, EMK*19, SVPM14, CG15]. In contrast, far fewer techniques and comparisons thereof exist for projecting time-dependent datasets, in which the sample values change over time – which is a much harder problem.

Besides faithfully capturing the data structure – a desiderate shared with static projections – dynamic projections also face the challenge of maintaining temporal coherence. Failing this will create false motion artifacts in the projection, which can mislead the user into thinking there are data changes where none exist, or conversely. Figure 1 illustrates this: We have a 100-dimensional dataset of 2000 samples covering 10 distinct isotropic Gaussian distributions that collapse into 10 single points over 10 timesteps [RFaT16]. The images depict the results of three dynamic projection techniques (G-PCA, TF-PCA [Jol86], and TF-tSNE [vH08]) for this dataset, showing the trajectories of all data points over the ten timesteps. Knowing the dataset, we can tell that G-PCA renders quite faithfully the data dynamics and structure; TF-PCA creates an artificial amount of spiraling; and TF-tSNE creates a very large amount of apparently random and unstable motion that is not present in the data. If such variability in the projection results is seen for this simple, synthetic dataset, the choice of a good dynamic projection method for real-world datasets is clearly very hard.

Motivated by these challenges of understanding and quantifying the quality of dynamic projections, Vernier *et al.* [VGd*20] evaluated nine such techniques, and came to the conclusion that there is no perfect method, and that an inherent trade-off between stability and spatial quality (i.e., neighborhood and distance preservation) exists. The methods that scored the best on both criteria were autoencoder-based methods and Global PCA. Neighborhood-based methods, such as t-SNE and UMAP, strongly showed a lack of stability. At the same time, these are among the favorite methods for static projection, given their high capability in preserving data structure.

We aim to cover the above-identified gap by proposing ways to add stability to the neighborhood preservation ability of static projections, in particular t-SNE. We propose two approaches that use global influences to steer projected points: Our first method, LD-tSNE, offers similar flexibility in steering dynamic projections via landmarks as known for static projections, and also reaches good quality values. Our second method, PCD-tSNE, increases neighborhood influences atop an already stable Global PCA dynamic projection, scoring better than all compared counterparts in terms of spatial quality combined with stability. The global influence of both methods can be controlled via simple user parameters to find the best balance between stability and spatial quality. We compare our methods with 11 existing dynamic projections on a benchmark of 10 high-dimensional datasets using 12 metrics for both spatial quality and stability.

Section 2 overviews related work on dynamic projections of high-dimensional data. Section 3 introduces our two new methods. Section 4 presents the experimental setup we used in comparing our new methods with existing ones. Section 5 presents and discusses the evaluation results. Finally, Section 6 concludes the paper.

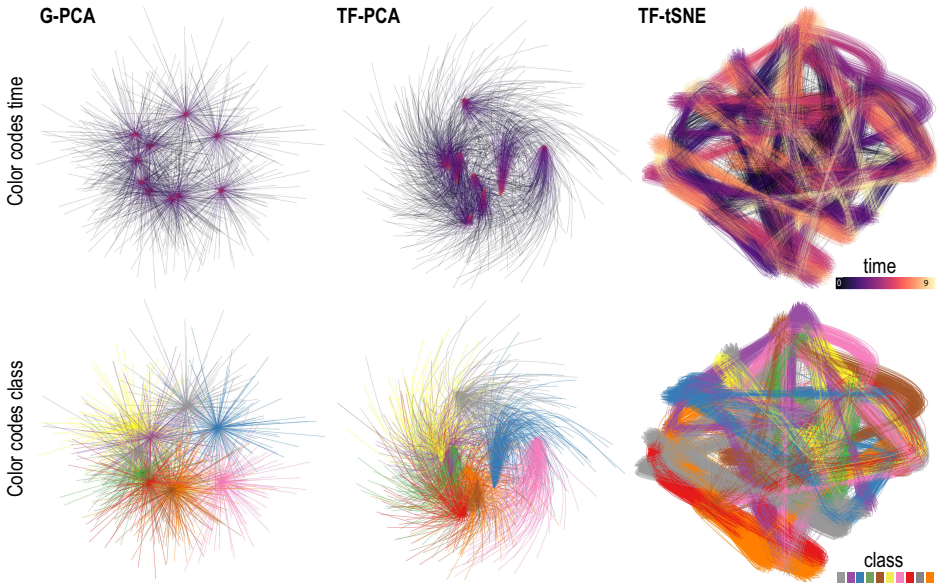


Figure 1: *A time-dependent collapsing 100-dimensional 10-Gaussian-distributions dataset (2000 points) is visualized by three projection methods. Point trails are colored by time (top) and class (bottom). The images show increasing amounts of instability artefacts.*

2. Related work

2.1. Preliminaries

We first introduce some notations. Let $\mathbf{x} \in \mathbb{R}^n$ be an n-dimensional sample (also called data point or observation). A timestep $\mathbf{D}^t = \{\mathbf{x}_i^t\}$ of our data consists of a set of N samples $\mathbf{x}_i^t$, $1 \leq i \leq N$, measured at the same time moment t. A dynamic dataset $\mathbf{D}$ is a list of T timesteps $\mathbf{D} = (\mathbf{D}^t), 1 \leq t \leq T$. For simplicity of exposition and implementation, but without loss of generality, we consider next that the sample count N is constant over time. In this case, $\mathbf{D}$ can be represented as a set of T N-by-n matrices, one per timestep t.

A projection technique is a function $P : \mathbb{R}^n \to \mathbb{R}^q$, where $q \ll n$. For visualization purposes, $q \in \{2,3\}$. Since 2D projections are by far the most commonly used, we next only consider the case $q = 2$. We denote the projection of sample $\mathbf{x}$ by $P(\mathbf{x})$. For a timestep t, let $P(\mathbf{D}^t) = \{P(\mathbf{x}^t) | \mathbf{x}^t \in \mathbf{D}^t\}$ be the 2D scatterplot of the projections of all points in $\mathbf{D}^t$. Finally, let $P(\mathbf{D})$ be the set of T scatterplots for all timesteps of dataset $\mathbf{D}$. These can be rendered as animations (see additional material [VCT20]), trail sets (as in Fig. 1), small multiples [RFaT16], or other visual encodings.

2.2. Visualization of high-dimensional data

Visualization of static high dimensional data [LMW*17] is a well studied topic populated with many techniques such as parallel coordinate plots [ID90], table lenses [RC94], scatterplot matrices [BCS96], and dimensionality reduction (DR) methods or projections [vPVdH09]. Compared to other methods, projections scale visually very well, being able to accommodate datasets of millions of samples and hundreds up to thousands of dimensions in limited screen space. Several quality metrics have been proposed to gauge how faithfully projections capture the structure of high-dimensional data, *e.g.*, trustworthiness and continuity [VK06], normalized stress and Shepard diagrams [JCC*11], neighborhood hit [PNML08], class consistency [TBB*10], and distance consistency [SNLH09]. Tens of different projection algorithms exist for static data; detailed taxonomies of such methods, benchmarks, and qualitative and quantitative evaluations are available in a range of surveys [NA19, EMK*19, Fod02, CG15, SVPM14, vPVdH09].

2.3. Strategies for dynamic projections

All current dynamic projection techniques that we are aware of are based on methods that were initially designed for static data. These base methods are adapted to achieve two goals: (a) obtaining good *spatial quality*, measured by the various static projection metrics outlined earlier in Sec. 2.2; and (b) obtaining good *stability*, defined as the ratio between changes, over time, of the projection $P(\mathbf{D})$ *vs* changes of the data $\mathbf{D}$ [VGd*20]. Besides projections, similar definitions of stability have been used to quantify dynamic treemapping algorithms [VSC*20, VCT18]. We next propose to classify these techniques as a function of how they 'adapt' the underlying base (static) projection algorithm, denoted further P_B, to handle spatial quality and stability for dynamic data.

Per-timeframe (TF): In this simplest strategy, P_B is applied to each timestep $\mathbf{D}^t$ to create an independent projection $P_B(\mathbf{D}^t)$. Hence, $P(\mathbf{D}) = (P_B(\mathbf{D}^t))_{1 \leq t \leq T}$. In other words, the base method P_B is not allowed to "look at the past or future" when projecting a given timestep t – it only sees the data in $\mathbf{D}^t$. Given the popularity of PCA [Jol86], t-SNE [vH08], and UMAP [MHM18], the per-timeframe strategy is often used for these base projections, leading to variants we call next TF-PCA, TF-tSNE, and TF-UMAP, respectively. Several further variations of this strategy exist. Bach *et al.* [BSH16] propose time curves which connect consecutive positions $P_B(\mathbf{x}_i^t)$ of the same point i for all moments t, using MDS for P_B. Similar curves have been used by Bernard *et al.* [BWS12] (using PCA for P_B). Brich *et al.* [BSP*20] use time curves and argue for the pro's and con's of PCA *vs* MDS for P_B. However, none of the studied base projections was found ideal concerning stability and spatial quality. At a more general level, the same strategy was used to connect different 2D scatterplots created

by other means than projections [HKF16]. Jäckle *et al.* [JFSK16] use MDS for P_B to project all n spatial dimensions of $\mathbf{D}$ to a single dimension and use the second dimension of the screen space to map time. Overall, the per-timeframe strategy favors spatial quality, which can be as high as delivered by P_B. However, stability can be (very) low since P_B is applied independently to the timeframes.

Global (G): At the other end of the spectrum, global methods apply P_B to the entire dataset, and then separate the projected points based on their timesteps, *i.e.*, $P(\mathbf{D}) = (\{\mathbf{y}^t \in P_B(\mathbf{D})\})_{1 \leq t \leq T}$, where $\mathbf{y}^t = P_B(\mathbf{x}^t)$ and $\mathbf{x}^t \in \mathbf{D}^t$. Like per-timeframe, this strategy is also simple to implement. It maximizes stability by construction. As such, many applications use this strategy, *e.g.* Hu *et al.* [HWX*10] that project 72-dimensional human body keypoints using LLE, or Fujiwara *et al.* [FLM*18] who project entire dimensions (time series) using MDS and t-SNE for computer performance analysis. The latter method was also extended to use PCA and UMAP as P_B [FSS*20].

When $\mathbf{D}$ is large, either in terms of number of samples or number of timesteps, computing a single projection $P_B(\mathbf{D})$ can be expensive. Also, the spatial quality of global techniques is typically lower than for the per-timeframe strategy since P_B now has to optimize the relative placement of points in *all* timeframes, even if such points never co-exist at the same time. Out-of-sample projection (OOS) methods can help with these issues. Simply put, an OOS technique P is constructed to optimize the projection of a subset $\mathbf{D}_s \subset \mathbf{D}$ according to one's desired quality metrics. Next, P is used to extrapolate the projection to the entire $\mathbf{D}$. Out-of-sample strategies have been proposed for many static projection methods [BPV*03]. Recently, Espadoto *et al.* [EHT20] have shown how to use deep learning to construct out-of-sample approximations of any static projection technique. Hence, OOS techniques can be used to accelerate and potentially increase the quality of global projection methods. However, the challenge is in how to select the small subset $\mathbf{D}_s$ so as to represent well the entire time-dependent dataset $\mathbf{D}$. To our knowledge, no studies of this aspect exist for dynamic projections.

Continuous (C): This strategy applies to base methods that iteratively optimize neighborhood configurations, such as t-SNE and UMAP. In the following, we call these variants C-tSNE and C-UMAP, respectively. The projection $P(\mathbf{D}^t)$ continues the gradient descent from the positions of the previous timestep $P(\mathbf{D}^{t-1})$, with the updated cost function for t. This reduces significantly the non-deterministic behavior created by removing consecutive initialization steps. Still, this can fail to produce stable projections as points are still allowed to move significantly during optimization. Dynamic t-SNE (D-tSNE) [RFaT16] aims to alleviate this by adding a penalty term to the continuous strategy using t-SNE for P_B. This limits, up to a certain extent, too large point movements between consecutive timesteps. Incremental PCA [RLL*08] projects points in a streaming fashion and is therefore amenable to project time-dependent data. Fujiwara *et al.* [FSS*20] further increase incremental PCA's stability by using Procrustes analysis to align consecutive projections, a method also proposed independently by Joia *et al.* [JCC*11]. Neves *et al.* [NMC*20] propose Xtreaming, an incremental technique that handles streaming high-dimensional data by continuously adapting UPDis [NFH*18], a projection with out-of-sample capability, thus, good stability. Overall, continuous strategies achieve a good balance between spatial quality and stability. However, this balance can be hard to tune in practice.

Vernier *et al.*'s evaluation [VGd*20] found that PCA and (Variational) Autoencoders with the global strategy – called next G-PCA, G-VAE, and G-AE respectively – were the best-suited methods for projecting temporal data. The global strategy, however, does not seem to work well with graph or neighborhood-based methods, such as t-SNE and UMAP – we denote these methods next as G-tSNE and G-UMAP, respectively.

3. Guided methods for dynamic projection

Many guided methods exist in the static projection literature [NA19, SVPM14]. Simply put, all these methods select a subset of samples $\mathbf{L} \subset \mathbf{D}$ to create P, by extrapolating $P(\mathbf{L})$ to $P(\mathbf{D})$. Conceptually speaking, the continuous strategy (Sec. 2.3) can be seen as a type of guidance, where $P(\mathbf{D}^{t+1})$ is steered by the earlier projection $P(\mathbf{D}^t)$. Similarly, the out-of-sample global strategy (Sec. 2.3) can be seen as a type of guidance where $P(\mathbf{D}_s)$ steers $P(\mathbf{D})$. However, even though this works for simple datasets, when the data present complex dynamics and large changes over time, existing continuous strategies become too restrictive. We propose two new guided methods for dynamic projection that use global influences (landmarks or suggested placements) to steer and stabilize the projection while still accounting for neighborhood preservation. The two methods use t-SNE as base projection given (a) t-SNE's high popularity for the static projection case; and (b) the difficulty of using t-SNE in a dynamic context (see Sec. 2.3), which we want to overcome. Importantly, while guided strategies mainly aim to address *scalability* for static projections, our different aim of using guidance is to address *spatial quality* and *stability*.

3.1. Landmark Dynamic t-SNE (LD-tSNE)

One idea that has been successfully used in the static case, and can be utilized to our advantage for dynamic data, is the use of *landmarks*. Landmarks or similar control point-based mechanisms are well known and have been used to aid different tasks on static data. Examples include performance improvement [PdRDK99, DT03, DT04, VCP13, PNML08, KTH17], support of out-of-sample capability [BFHL17, PSZ19], and projection customization [JCC*11, NFH*18]. Yet, we are not aware of any work that combines landmarks or control points to stabilize *dynamic* projections. We use landmarks to give the base projection P_B method a sense of global structure, in an attempt to reduce the instability inherent to neighborhood-based projection techniques such as t-SNE.

Two main aspects must be considered when using landmarks as guides: how to generate the landmarks and how to use the landmarks to steer points, as follows.

Landmark generation: Each landmark $\mathbf{l} = (\mathbf{l}^n, \mathbf{l}^q)$ consists of a high-dimensional component $\mathbf{l}^n \in \mathbb{R}^n$ and a component $\mathbf{l}^q \in \mathbb{R}^q$ in the projection space. It is important that the set $\mathbf{L} = \{l^n\}$ captures well the structure of the high-dimensional dataset $\mathbf{D}$, otherwise the "steering" may become uneven. There are different ways of achieving this goal [DT05]. For simplicity and speed, we opted to create $\mathbf{L}$ by randomly sampling k points from $\mathbf{D}$, where k is a fraction of the size of $\mathbf{D}$. For most of our tests, we set $k = N$, *i.e.*, the number of points in a timeframe (see Appendix in supplementary material). To generate the low-dimensional points $\mathbf{l}^q$, we simply project $\mathbf{L}$ using a user-chosen method. We experimented here with both PCA and t-SNE, and selected the landmark projection which yielded the best results (see Appendix in supplementary material).

Landmark steering: The first step towards steering is to select a

neighborhood-based projection technique to use. We chose here t-SNE due to its popularity and previous good results in extending it for dynamic data [RFaT16]. To describe how steering takes place, let us consider the original t-SNE cost function, given by the Kullback-Leibler (KL) divergence between the joint-probability distributions $\mathcal{P}$ and $\mathcal{Q}$ that describe point-neighborhoods in $\mathbb{R}^n$, respectively $\mathbb{R}^q$

$$C_{tsne} = D_{KL}(\mathcal{P}||\mathcal{Q}) = \sum_{i=1}^{N}\sum_{j\neq i} p_{ij}\log\frac{p_{ij}}{q_{ij}}, \tag{1}$$

where $p_{ij} = \frac{p_{i|j}+p_{j|i}}{2N}$ models the distance of two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbb{R}^n$ and

$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2/\left(2\sigma_i^2\right)\right)}{\sum_{k\neq i}^{N}\exp\left(-\left\|\mathbf{x}_i-\mathbf{x}_k\right\|^2/\left(2\sigma_i^2\right)\right)}.$$

Here, $p_{j|i}$ can be seen as a relative measure of similarity based on the local neighborhood of a point $\mathbf{x}_i$. The effective number of neighbors considered for each point is given indirectly by a user-chosen perplexity value μ: The value of σ_i is computed so that, for the user-given μ and each i, $\mu = 2^{-\sum_j^N p_{j|i}\log_2 p_{j|i}}$.

A Student's t-distribution with one degree of freedom is used to compute the joint-probability distribution in $\mathbb{R}^q$ as

$$q_{ij} = \frac{\left(1+\left\|\mathbf{y}_i-\mathbf{y}_j\right\|^2\right)^{-1}}{\sum_{k,i\neq k}\left(1+\left\|\mathbf{y}_i-\mathbf{y}_k\right\|^2\right)^{-1}}.$$

The gradient of the cost function, given by

$$\frac{\partial C_{tsne}}{\partial \mathbf{y}_i} = 4\sum_j\left(p_{ij}-q_{ij}\right)\left(\mathbf{y}_i-\mathbf{y}_j\right)\left(1+\left\|\mathbf{y}_i-\mathbf{y}_j\right\|^2\right)^{-1} \tag{2}$$

is used to incrementally move the points $\mathbf{y}_i$ to reduce the cost C_{tsne}.

To add landmark influence to t-SNE we will, similarly to Rauber *et al.* [RFaT16], add a second term to the cost function. In their work, the extra term was used to penalize *any* kind of 2D movement. In our case, we want to *guide* the placement of points $\mathbf{y}_i$ based on the similarity of $\mathbf{x}_i$ with the landmarks $\mathbf{l}^n$. Figure 2 illustrates these global and local influences. In Fig. 2a, the landmarks in $\mathbf{L}$ (light blue) produce attraction and repulsion forces to guide the placement of the red point $\mathbf{y}_i$. In Fig. 2b, the remaining points $\mathbf{y}_j$, $j\neq i$ (gray in the figure), exert similar forces, influencing and being influenced by $\mathbf{y}_i$, just like in a regular t-SNE projection.

We weigh the global and local influences by a factor $\lambda\in[0,1]$ giving the total cost function

$$C = (1-\lambda)C_{tsne} + \lambda C_{landmarks}. \tag{3}$$

In the above, $C_{landmarks}$ is similar to the original t-SNE cost function C_{tsne}. However, instead of considering p_{ij} for all pairs of points in $\mathbf{D}$ or $\mathbf{D}^t$, we let only the landmarks $\mathbf{l}\in L$ act upon each $\mathbf{y}_i$, *i.e.*

$$C_{landmarks} = \sum_i\sum_{l\in L} p_{i|l}\log\frac{p_{i|l}}{q_{il}}. \tag{4}$$

For these influences to work consistently through all time steps t, several aspects differ from the original t-SNE. In Eqn. 4, we use the *asymmetric* $p_{i|l}$ instead of the symmetric p_{il} used in Eqn. 1. Indeed, we want the landmarks to influence the points, not the other way round. Secondly, for the computation of σ_l for each landmark, we only take into consideration the landmark points $\mathbf{L}$. These two modifications ensure that the forces are consistent and do not fluctuate depending on the local density of points in $\mathbf{D}$ or $\mathbf{D}^t$.

From Eqns. 2, 3, and 4, we find the gradient of C as

$$\frac{\partial C}{\partial \mathbf{y}_i} = (1-\lambda)\left(4\sum_j\left(p_{ij}-q_{ij}\right)\left(\mathbf{y}_i-\mathbf{y}_j\right)\left(1+\left\|\mathbf{y}_i-\mathbf{y}_j\right\|^2\right)^{-1}\right) + \lambda\left(4\sum_{l\in L}\left(p_{i|l}-q_{ij}\right)\left(\mathbf{y}_i-\mathbf{y}_j\right)\left(1+\left\|\mathbf{y}_i-\mathbf{y}_j\right\|^2\right)^{-1}\right).$$

To accelerate convergence, improve initialization, and create tighter clusters, exaggeration terms are used [vH08, van15, LRH*19, LS17]. These are scalars that multiply p_{ij}, suggesting greater similarity between points than $\mathcal{P}$ captures. We do the same by adding two factors α and β to grant additional influence on how much points in $\mathbf{D}^t$ affect each other (α = local), respectively how much the landmarks "pull" the projected points (β = global), leading to the final cost gradient

$$\frac{\partial C}{\partial \mathbf{y}_i} = (1-\lambda)\left(4\sum_j\left(\alpha p_{ij}-q_{ij}\right)\left(\mathbf{y}_i-\mathbf{y}_j\right)\left(1+\left\|\mathbf{y}_i-\mathbf{y}_j\right\|^2\right)^{-1}\right) + \lambda\left(4\sum_{l\in L}\left(\beta p_{i|l}-q_{ij}\right)\left(\mathbf{y}_i-\mathbf{y}_j\right)\left(1+\left\|\mathbf{y}_i-\mathbf{y}_j\right\|^2\right)^{-1}\right).$$

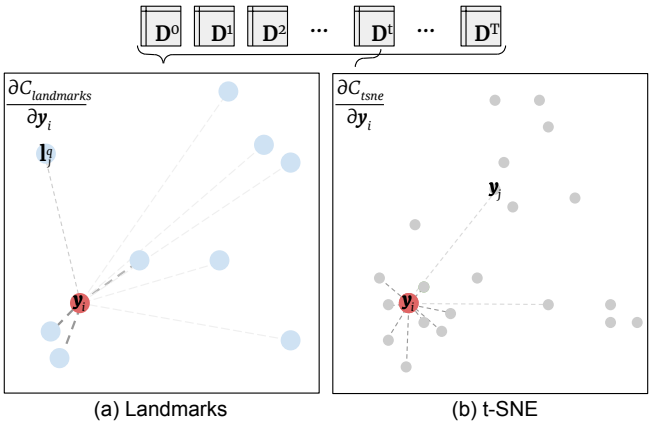


Figure 2: *Effect of landmarks (a) and regular projection points (b) upon a point $\mathbf{y}_i$ in LD-tSNE. See Sec. 3.1.*

Regarding algorithmic complexity, the original unoptimized implementation of the t-SNE method is $\mathcal{O}(n^2)$ for both computation and memory [vH08]. Our LD-tSNE algorithm has an additional cost $\mathcal{O}(ln)$ given by the interaction of the landmarks with the points in the projection, where l is the number of landmarks and n the number of points in the projection. Therefore, the final time and memory complexity are given as $\mathcal{O}(n^2+ln)$, or, since n^2 dominates the cost, LD-tSNE can be considered $\mathcal{O}(n^2)$.

3.2. Principal Component Dynamic t-SNE (PCD-tSNE)

Our second dynamic projection, PCD-tSNE, is a guided method that allies the stability of G-PCA with the neighborhood preservation capabilities of t-SNE. Just like D-tSNE and LD-tSNE, it includes an

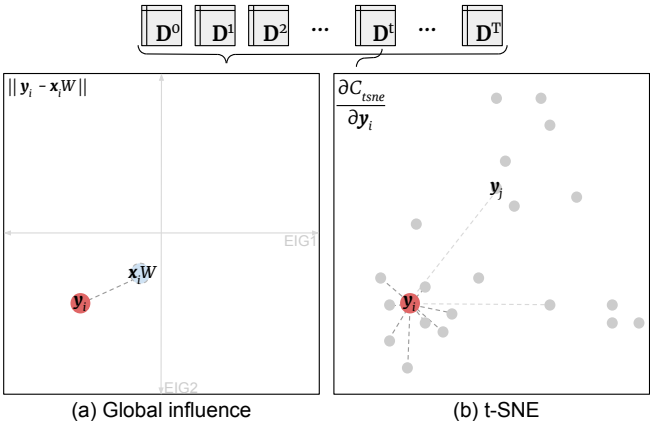
additional term to the t-SNE cost function that adds stabilization to the otherwise unstable C-tSNE.

The first step in PCD-tSNE is to compute a projection matrix W constructed from the top-q eigenvectors of the covariance matrix of $\mathbf{D}$. Simply put, W describes the (two, in our case) orthogonal axes of largest data variation over the whole dataset. For each point $\mathbf{x}_i \in \mathbf{D}$, we apply a transformation $\mathbf{x}_i W$ to map $\mathbf{x}_i$ to $\mathbb{R}^q$. More specifically, this places $\mathbf{x}_i$ exactly as G-PCA would, which was proven earlier [VGd*20] to create *stable* projections.

The placement of each projection point $\mathbf{y}_i$ is next given by two factors (see Fig. 3): an attraction to the position $\mathbf{x}_i W$, marked in light blue in Fig. 3a; and the influence of all other points in $\mathbf{D}^t$ upon $\mathbf{y}_i$, as given by tSNE, these points being shown in gray in Fig. 3b. With these elements, the gradient of our cost function is given by:

$$\frac{\partial C}{\partial \mathbf{y}_i} = (1-\lambda)\frac{\partial C_{tsne}}{\partial \mathbf{y}_i} + \lambda \|\mathbf{y}_i - \mathbf{x}_i W\|. \tag{5}$$

Here, $\lambda \in [0,1]$, similarly to LD-tSNE, weighs the balance of local and global influences. More specifically, by adjusting λ, we can achieve an exact C-tSNE projection ($\lambda = 0$), an exact G-PCA projection ($\lambda = 1$), or a projection in between these variants.



(a) Global influence (b) t-SNE

Figure 3: *Effect of global influence (a) and regular projection points (b) upon a point $\mathbf{y}_i$ in PCD-tSNE. See Sec. 3.2.*

Regarding complexity, for PCD-tSNE, the first step is to compute the top-2 eigenvectors of the original data. There are many numerical methods designed to efficiently perform this computation with cost as low as $\mathcal{O}(kn)$ for k singular values [Kre05, CD06]. Once the PCD-tSNE optimization starts, the additional term introduced on Equation 5 represents a simple convex function, which means that convergence is reached efficiently and PCD-tSNE performs similarly to C-tSNE, that is, in $\mathcal{O}(n^2)$ time.

4. Evaluation procedure

We next present our evaluation of the two proposed dynamic projection methods, LD-tSNE and PCD-tSNE. For the evaluation, we started with the benchmark in Vernier *et al.*' [VGd*20], which is to our knowledge the only benchmark dedicated specifically to dynamic projections. The implementation of per-timeframe and global methods were provided by [VGd*20]; D-tSNE was provided by [RFaT16]; the remaining techniques considered in our evaluation (Sec. 3) were implemented by ourselves. All source code, datasets, and obtained results can be found in our online repository [VCT20].

4.1. Methods

Vernier *et al.*'s [VGd*20] benchmark contained 9 methods – five global ones (G-AE, G-VAE, G-tSNE, G-UMAP, G-PCA), three per-timeframe ones (TF-tSNE, TF-UMAP, TF-PCA), and one continuous (D-tSNE). Atop of those we added C-tSNE and C-UMAP, and the two newly proposed methods, LD-tSNE and PCD-tSNE. The parameters used in the benchmark are available in the supplemental material.

4.2. Quality Metrics

Following Vernier *et al.* [VGd*20] and [EMK*19], we used 8 *spatial* and 4 *temporal* quality metrics, as follows. Temporal metrics measure the correspondence of movement of projection points in $\mathbb{R}^q$ with regard to their change in the data space $\mathbb{R}^n$ space, *i.e.*, stability.

4.2.1. Spatial metrics

Spatial metrics measure how well a projection maps the underlying high-dimensional data, and can be divided into neighborhood preservation metrics ($S_{NP}, S_{NH}, S_{Trust}, S_{Cont}$) and distance preservation metrics ($S_{Stress}, S_{Pearson}, S_{Spearman}, S_{Kendall}$). Note that these do not necessarily relate to how humans perceive the projection [WFC*18].

Neighborhood preservation (S_{NP}) is the fraction of the k-nearest neighbors of $\mathbf{x} \in \mathbf{D}$ whose projections are in the k-nearest neighbors of $P(\mathbf{x})$.

Trustworthiness (S_{Trust}) measures how well the k nearest neighbors $\nu^k(P(\mathbf{x}))$ of a projected point $P(\mathbf{x})$ match the k nearest neighbors $\nu^k(\mathbf{x})$ of a data point $\mathbf{x}$, specifically, how *few* missing neighbors [MCMT14] a projected point has. If $U^k(\mathbf{x})$ is the set of points in $\mathbf{D}$ that project in $\nu^k(P(\mathbf{x}))$ but are not in $\nu^k(\mathbf{x})$, and $r^P(\mathbf{x},\mathbf{y})$ is the rank of $\mathbf{y}$ in the ordered set of nearest neighbors $\nu^k(P(\mathbf{x}))$, trustworthiness is defined as $1 - \frac{2}{Nk(2N-3k-1)} \sum_{x=1}^{N} \sum_{y \in U^k(\mathbf{x})} (r^P(\mathbf{x},\mathbf{y}) - k)$.

Continuity (S_{Cont}) measures how *many* missing neighbors a projected point has. Let $V^k(\mathbf{x})$ be the points that are in $\nu^k(\mathbf{x})$ but do not project in $\nu^k(P(\mathbf{x}))$. Let $r(\mathbf{x},\mathbf{y})$ be the rank of $\mathbf{y}$ in the ordered set of neighbors $\nu^k(\mathbf{x})$. Continuity is then defined as $1 - \frac{2}{Nk(2N-3k-1)} \sum_{x=1}^{N} \sum_{y \in V^k(\mathbf{x})} (r(\mathbf{x},\mathbf{y}) - k)$.

Neighborhood hit (S_{NH}) is the fraction of the k-nearest neighbors of a projected point $P(\mathbf{x})$ that have the same class label as $P(\mathbf{x})$. Since we use labeled datasets with reasonably well-separated classes in $\mathbb{R}^n$ (see next Sec. 4.3), a projection P that is good for class-separation tasks should have a high S_{NH} value.

All the above metrics range in $[0,1]$, with 1 indicating optimal value. We compute S_{NP}, S_{Trust}, and S_{Cont} for multiple (20) neighborhood sizes equally spread between $k = 1\%$ and $k = 20\%$ of the point count N. For S_{NH}, we use 20 values for k, ranging from 0.25% to 5% of N. We next average the results for different neighborhood sizes k, following [VGd*20, MMT15].

Normalized stress (S_{Stress}) measures the pairwise difference of distances of points in $\mathbf{D}$ and $P(\mathbf{D})$. We define S_{Stress} as $\sum_t \sum_{ij} (d^t_{ij} -$

$\delta_{ij}^t)^2 / T \sum_{ij} (\delta_{ij}^t)^2$, where d_{ij}^t and δ_{ij}^t are the Euclidean distances between data points $\mathbf{x}_i^t$ and $\mathbf{x}_j^t$, and between their projections $P(\mathbf{x}_i^t)$ and $P(\mathbf{x}_j^t)$, respectively for every point pair (i, j) and timeframe $1 \leq t \leq T$. To ease analysis, we scale distances using standardization.

Shepard diagram metrics. The Shepard diagram is a scatterplot of d_{ij} by δ_{ij}, for every point pair (i, j) [JCC*11]. A diagram close to a diagonal line indicates good distance preservation. Scatterplots spreading above or below the diagonal indicate distance compression (potential false neighbors), respectively stretching (potential missing neighbors) from $\mathbf{D}$ to $P(\mathbf{D})$. We use Pearson correlation, Spearman rank, and Kendall tau to measure the linearity and monotonicity of the relationship of d_{ij} with δ_{ij} in Shepard diagrams. The three resulting metrics $S_{Pearson}, S_{Spearman}, S_{Kendall}$ range in $[-1, 1]$, with 1 being the ideal distance-preservation case.

4.2.2. Temporal stability metrics

We estimate how stable a projection is by studying the relationship of the *data* change of a point from $\mathbf{x}_i^t$ to $\mathbf{x}_i^{t+1}$, measured by $c_i^t = \|\mathbf{x}_i^t - \mathbf{x}_i^{t+1}\|$, and the movement of the corresponding *projections* from $P(\mathbf{x}_i^t)$ to $P(\mathbf{x}_i^{t+1})$, measured by $\kappa^t = \|P(\mathbf{x}_i^t) - P(\mathbf{x}_i^{t+1})\|$. For stable P, we ideally would want κ_i^t to be proportional, or at least correlated with, c_i^t. We use the following metrics [VGd*20] to capture this notion of stability.

Normalized temporal stress (T_{Stress}) is defined as $\sum_{it} (c_i^t - \kappa_i^t)^2 / (c_i^t)^2$. As for S_{Stress}, we normalize distances using standardization. Low T_{Stress} values tell that the $\mathbb{R}^q$ changes κ_i^t reflect closely their $\mathbb{R}^n$ counterparts c_i^t, which is what we want.

Temporal Shepard diagram metrics: We measure the Pearson and Spearman correlation and Kendall's tau ($T_{Pearson}, T_{Spearman}, T_{Kendall}$) between c_i^t and κ_i^t for every sample i and timestep t. High values indicate that the $\mathbb{R}^q$ changes κ_i^t are strongly correlated with their $\mathbb{R}^n$ counterparts c_i^t, which is desirable.

4.3. Datasets

We used 10 public datasets extracted from different sources and portraying a wide range of temporal phenomena, such as videos, sound recordings, sports statistics, algorithm behavior, and a few synthetic datasets with easily recognizable dynamics [VCT20]. The collection also exhibits significant variations in measurable traits such as the number of samples N, the number of timesteps T, dimensionality n, intrinsic dimensionality ρ_n (percentage of dimensions that describe 95% of the data variance), and sparsity ratio σ_n (percentage of zeros in the data), as shown by Table 1. These traits have been used earlier [EMK*19] to indicate that a benchmark captures an as wide as possible (within the benchmark's size bounds) spread of phenomena of different natures.

cartolastd: This dataset has player statistics for the second turn of the 2017 Brazilian soccer championship. Data was extracted from an open-source project [GG19] that scrapes the Cartola FC [Glo19] platform. Each of the 19 timesteps is a tournament round. Samples are players, with dimensions being per-match performance (number of goals, assistances, fouls, defenses) and player position (goalkeeper, right or left-back, defender, midfield, forward).

cifar10cnn: Samples are images classified by a convolutional network trained for the CIFAR10 [Kri09] dataset. Dimensions are activations of neurons of the network's last hidden layers. Timesteps represent training epochs. This dataset is similar to the one produced for MNIST [LC10] by [RFFT17], but consider the significantly harder-to-classify CIFAR10 dataset.

Table 1: *Datasets used and their traits (from [VGd*20]).*

dataset	samples N	timesteps T	dimensions n	classes	intrinsic dim. ρ_n	sparsity ratio σ_n
cartolastd	696	19	17	5	0.6470	0.0000
cifar10cnn	1000	30	10	10	0.6599	0.0000
esc50	320	108	128	8	0.0345	0.0000
fashion	1000	10	784	10	0.4762	0.2971
gaussians	2000	10	100	10	0.3680	0.0000
nnset	80	30	8070	8	0.0057	0.0001
qtables	180	40	1200	9	0.0077	0.0007
quickdraw	600	89	784	6	0.4309	0.9013
sorts	80	100	100	8	0.3505	0.0100
walk	300	50	100	3	0.4783	0.0001

esc50: Sound samples of 8 classes (brushing teeth, chainsaw, crying baby, engine, laughing, rain, siren, wind) compressed to 128 frequencies and smoothed over time. Extracted from Piczak's ESC50 dataset [Pic15].

fashion: This is a subsample of 100 images from each of the 10 classes (T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot) of the FashionMNIST [XRV17] dataset. Each image is recorded over 10 timesteps, with decreasing amounts of noise over time.

gaussians: Isotropic gaussian blobs in $\mathbb{R}^{100}$ with diminishing spread over time. Used originally to evaluate D-tSNE [RFaT16].

nnset: Internal states (weights and biases) of several neural networks during 30 training epochs to learn classifying the MNIST dataset. The networks have the same architecture but use different optimizers, batch sizes, and training-set sizes.

qtables: Internal state of agents learning to move a car up a hill using the reinforcement learning algorithm Q-learning [Wat89]. The nine classes represent variations of learning rates and discounts.

quickdraw: Drawing sequences for 600 objects of 6 different classes drawn by random people. Samples represent the pixels of individual drawings. Timesteps (89) represent the drawing stage. Data is extracted from the "Quick, Draw!" Google AI experiment [JRK*16].

sorts: This dataset was designed to compare eight sorting algorithms. They sort each different arrays of 100 random values in $[0, 1]$. We take snapshots of the algorithms' intermediate states until sorting completion. A sample is an (algorithm, array) run, its dimensions being the partially-sorted array values at a given sorting step.

walk: Synthetic dataset similar to *gaussians* [RFaT16], but with more complex dynamics. It contains 3 high-dimensional clusters that oscillate (approach, mingle, cross, and then drift apart) in $\mathbb{R}^{100}$ over 50 timesteps. This dataset tests how well the studied projections can capture the approaching, mingling, and drifting-away dynamics mentioned above.

5. Evaluation results

We used each of the selected 13 projection techniques (Sec. 4.1) to project the 10 datasets in the benchmark (Sec. 4.3). For every (dataset, method) pair, we compute 12 quality metrics (4 related to distance presentation, 4 related to neighborhood preservation, and 4 stability

metrics, see Sec. 4.2), and analyze the results at different levels of aggregation. For a direct impression, the animations of each (dataset, method) pair can be found in our online repository [VCT20].

5.1. Visual comparison of dynamic projections

We start with a simple, visual comparison of dynamic projection results. Figure 4 shows the trail-sets – curves linking $P(\mathbf{x}_i^t)$ for all t – for the *cifar10cnn* dataset, created by the 13 tested dynamic projection methods, organized following the taxonomy in Sec. 2.3. Points represent the last layer of neural network activations trained to classify the dataset, over 30 training epochs. Given the problem, we *expect* that activations 'segregate' into 10 distinct sets, corresponding to the 10 classes of images in the dataset. The trails should start from a roughly common area (middle of the projection), indicating lack of differentiation at training start, and evolve *smoothy*, that is, without major twists and bends, over epochs, to increasingly differentiated clusters, a phenomenon shown earlier in for the (far) easier-to-classify MNIST dataset [RFFT17] by C-tSNE. We see that only a few dynamic projections exhibit this pattern: G-VAE, G-UMAP, and our proposals, PCD-tSNE and LD-tSNE. All other dynamic projections do not show the convergence of trails to (ten) distinct clusters (red circles in the figure). Saliently, all TF variants show far too high dynamics - long trails turning and twisting, suggesting chaotic dynamics, which we know it is not the case from [RFFT17]. Other projections (G-AE, G-PCA, G-tSNE, C-UMAP, D-tSNE) do not show a clear convergence of trails into 10 clusters, which again, we know should be expected. Overall, we argue that PCD-tSNE and LD-tSNE capture the (known) ground-truth of the training dynamics better than most tested counterparts.

5.2. Overview of quality metrics

Figure 5 shows the results for each method separated by metric class. The three swarm plots [Ekl12] in the figure address each of the three metric categories outlined above (distance preservation, neighborhood preservation, temporal stability). Columns in a plot indicate methods. Each point in a column corresponds to the averaged result over the four normalized metrics in the respective class for a (method, dataset) pair. Methods in each plot are ordered by how high they score for a given metric class, with methods to the left scoring higher. Methods are categorically color-coded to ease comparison between the plots.

A method to be considered suitable for dynamic projections must be stable and achieve good distance and neighborhood preservation. Vernier *et al.*'s benchmark [VGd*20] concluded that, from all their 9 tested methods, G-PCA and Autoencoder-based techniques (G-AE, G-VAE) struck the best balance between these desiderates. We argue that, in this light, our new PCD-tSNE method is even more effective at projecting dynamic data. Indeed, as Fig. 5 (bottom plot) shows, PCD-tSNE's stability is comparable to G-PCA, G-AE, and G-VAE, being the third-most stable of all tested methods. At the same time, PCD-tSNE achieves better results in distance preservation, being the best of all tested methods. Regarding neighborhood preservation (Fig. 5, middle plot), PCD-tSNE is only surpassed by the TF (timeframe) and C (continuous) methods. This is not surprising since these methods do not have temporal constraints. This implies, as Fig. 5 (bottom plot) confirms, that the TF and C methods score very poorly for stability.

Additionally, PCD-tSNE overcomes two limitations of AE-based methods and G-PCA: Autoencoders are based on neural networks, which can be challenging to set up, optimize for the architecture, and train; PCA based methods are sensitive to outliers and do not explicitly try to preserve local features.

Concerning LD-tSNE, we see that this method did not achieve metric results as good as other state-of-the-art methods. Yet, it scores in the top half of all methods for all three considered metric classes. Also, its strength lies in its customizability (see next Sec. 5.5): If we want the projection to adhere to a certain shape, or we have some prior knowledge over the high-dimensional space and we want areas of the projections to carry a certain data-related semantic, we can easily place landmarks to drive the projection to that behavior. This extends the same flexibility, known earlier for static projections (see *e.g.* [JCC*11, PdRDK99]), to dynamic projections.

5.3. Stability and spatial quality trade-off

While the swarm plots in Fig. 5 help us see which methods score best for a given metric class, they do not let us easily compare methods from the perspective of *multiple* metrics. To achieve this, we use two star plots (Fig. 6), as follows. Each image is a scatterplot having temporal stability as the x axis and distance and neighborhood preservation, respectively, as the y axis. Each colored point shows the average metric values for a given technique over all datasets. Spokes emerging from a point show the average metric values for each of the 10 datasets run by the respective technique. For more insight into the behavior of the methods, we highlighted the spokes for the two best methods in each plot, *i.e.* the points placed closest to the top-right corner of the plot.

Figure 6a shows that PCD-tSNE and G-VAE are the best methods for distance preservation *and* stability, closely followed by G-PCA. Yet, the spokes of PCD-tSNE (pink) are shorter than those of G-VAE (blue). That is, PCD-tSNE achieves a consistently higher distance preservation and stability over all 10 tested datasets than G-VAE, which has a higher variability. Similarly, Figure 6b shows that PCD-tSNE scores highest in terms of neighborhood preservation *and* stability, closely followed by G-VAE and G-AE. Again, the spokes of PCD-tSNE are shorter than those of G-VAE, telling that PCD-tSNE achieves its high scores more consistently than G-VAE. We see that G-VAE performs worst for the *walk*, *nnset*, and *fashion* datasets, from both the perspective of distance preservation and neighborhood preservation (longest blue spokes in Figs. 6a,b for G-VAE, indicated by a cross, triangle, and check icons); for these datasets, PCD-tSNE performs quite well (short pink spokes). Also, there seems to be an inverse correlation between neighborhood and distance preservation for TF-tSNE, TF-UMAP, and G-UMAP, indicating that these methods are very good at neighborhood, but not distance, preservation. Separately, we see that our two methods, PCD-tSNE and LD-tSNE, are the best methods, stability-wise, from the t-SNE class, and perform far better, on all three metrics, than D-tSNE. In other words, if one wants to leverage t-SNE's ability for dynamic datasets, our methods are the best from the considered variants. Finally, we see that temporal stability and distance preservation appear to be well correlated over all tested methods (points in Fig. 6a close to the diagonal), which is to our knowledge a new finding in the projection literature. In contrast, no similar correlation appears between stability and neighborhood preservation (Fig. 6b).

5.4. Global *vs* local influence control

As outlined in Sec. 3.2, the PCD-tSNE method has a parameter λ that modulates the amount of global influence applied to the points being projected. When projecting a sample $\mathbf{x}_i \in \mathbf{D}$, this global influence

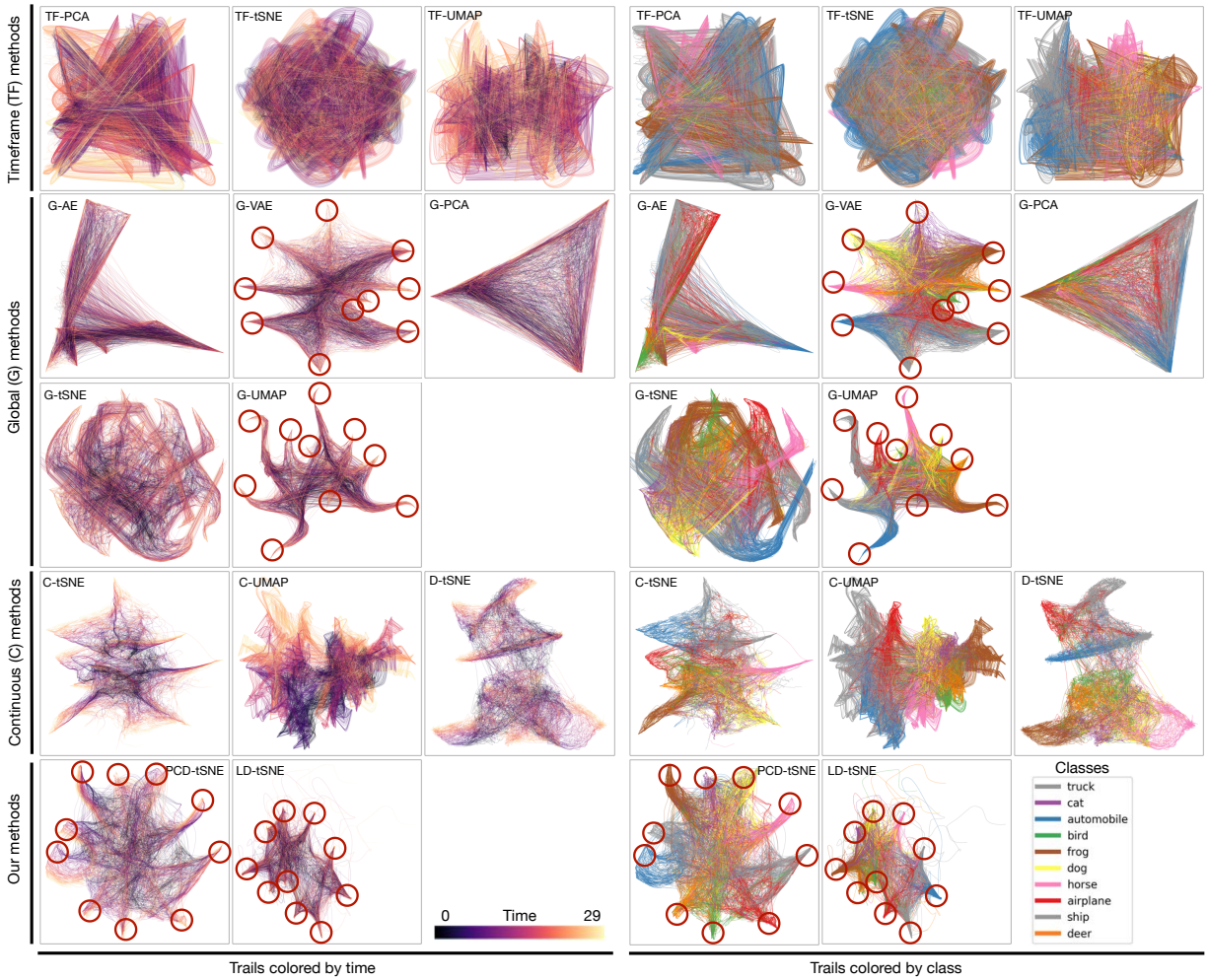


Figure 4: *Trails showing the "hidden activity" [RFFT17] of a convolutional neural network trained on the* CIFAR10 *[Kri09] dataset, computed by all 13 tested dynamic projections. Red circles show clusters of trail endpoints which indicate training convergence. Images without red circles show (suboptimal) projection methods where it is not possible to see this training convergence.*

refers to minimizing the distance of $P(\mathbf{x}_i)$ to the position given by the transformation matrix W composed of the top-q eigenvectors of $\mathbf{D}$. Another way to interpret this global influence is to think of $\mathbf{x}_i W$ as the position that G-PCA would generate; and to think about λ as how much we want PCD-tSNE to approximate G-PCA.

If we use high λ values (close to 1), PCD-tSNE gets very close to G-PCA, a method that has shown to be very stable, produce good distance preservation, but has low neighborhood preservation (Fig. 5). Conversely, with low λ values (close to 0), no global influences act upon PCD-tSNE, which turns into C-tSNE, a method that has high neighborhood preservation, but low stability and distance preservation (Fig. 5). Figure 7 supports and refines this insight. For each dataset (rows), we compute the mean distance preservation (MDP), mean neighborhood preservation (MNP), and mean temporal stability (MTS) over the respective metrics in each class (see Sec. 4.2). For each table row, we normalize values between 0 and 1, to better see the spread of values of the respective metric for each dataset. The leftmost column shows the metric results of G-PCA; the rightmost one shows the results of C-tSNE. The six middle columns show the results of PCD-tSNE with λ in $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$. Cells are colored using an ordinal colormap (dark green=low, bright yellow=high metric values) on the normalized values. The color gradients show that PCD-tSNE indeed yields metric values that are very similar, for high, respectively low, λ to those of G-PCA, respectively C-tSNE. Also, we see that PCD-tSNE can integrate characteristics of both C-tSNE and G-PCA and achieves the best balance between all quality metrics, as shown by the overall brightest columns in the middle of the table. Interestingly, PCD-tSNE is also often able to achieve the best result for certain quality metrics (maximum averages, marked bold in Fig. 7). This shows that PCD-tSNE doesn't simply *interpolate* projections (like, for example, in [KHS*17]), but uses the characteristics of both C-tSNE and G-PCA to create a better projection. In Fig. 7, note that different rows show different trends, which is expected since we consider different datasets and metrics.

Finally, Figure 7 shows that G-PCA and C-tSNE are not always optimal – the best projection lies sometimes in between, which is what PCD-tSNE obtains. Separately, it shows that optimal parameters depend on the dataset. The considered MDP, MNP, and MTS quality metrics could be used for automatic finding of such optimal

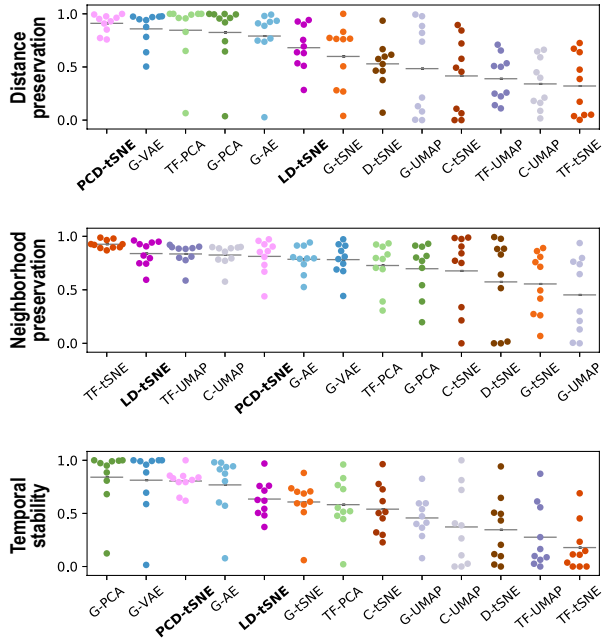


Figure 5: *Swarm plot ordering methods from best to worse for each metric class. Each point corresponds to the average metric result over the 4 metrics in a given class normalized to* $[0,1]$ *for each (method, dataset) pair. Horizontal lines show average metric values over all datasets for each (method, metric class) pair.*

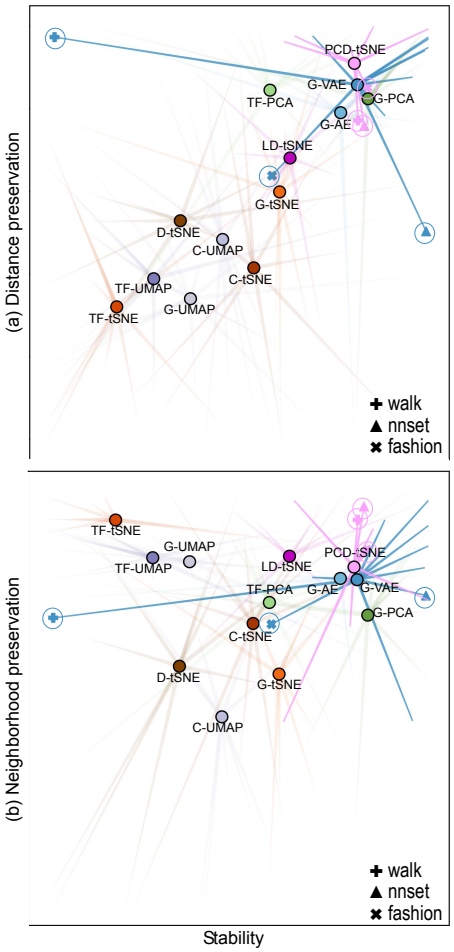


Figure 6: *Star plots compare dynamic projection methods from the perspective of stability* vs *distance (a) and stability* vs *neighborhood (b) preservation. A point shows the average values of these metrics for a given technique over all datasets. Spokes show the average metric values for each dataset for a given technique. Spokes of PCD-tSNE and VAE, the two techniques that score best, are in bold. Methods with big spoke fans show high variation on quality metrics. Short spokes show consistent results of a method over all 10 datasets.*

parameters – or good preset values for all datasets – by grid search, following the approach in [EMK*19] for static projections.

5.5. Using landmarks to steer dynamic projections

The key trait of LD-tSNE is that it allows steering a *dynamic* projection by changing the landmark point positions $\mathbf{l}^q$. If we monitor a high-dimensional process and we know what final and failed states look like, we can place landmarks indicating these states. As the process evolves, we will have a clear picture of which samples failed or succeeded; which samples are on the "right track"; and how similar samples are in in-between states among themselves and to these known states. Guiding landmarks are also valuable if we want a system that is consistent over slightly different datasets.

Landmark-based methods present many challenges in practice: How to choose a small set of points in $\mathbb{R}^n$ that is representative of $\mathbf{D}$? How many points do we need, and how do we select these representatives? De Silva *et al.* [DT05] propose regression models for picking the best landmarks, while Pezzotti *et al.* [PHL*16] use a hierarchical approach. Another option is to synthesize landmarks using models that approximate the manifold (e.g. Autoencoders). For simplicity, we select our landmarks by random sampling $\mathbf{D}$, as in earlier work considering static projections, *e.g.* [JCC*11, PdRDK99].

Figure 8 shows how landmark placement can steer a dynamic projection. We use the *gaussians* dataset, as we know its dynamics, so we can assess how well landmark steering works on the resulting projections. Points are colored per cluster; landmarks are drawn white. Figure 8a shows several timesteps of LD-tSNE with landmarks placed by G-PCA. We see how clusters 'implode' over time. While clusters stay roughly in the same place over time in the projection (a good indication of stability), their *spatial* organization is not ideal for monitoring the phenomenon. Figure 8b shows LD-tSNE for the same dataset, with the same landmarks $\mathbf{l}^n$ selected from $\mathbf{D}$, but with the 2D landmarks $\mathbf{l}^q$ placed manually into 10 horizontally-aligned, similar-size, clusters. The images show the same 'implosion' effect over time as in Fig. 8a. We argue that the *dynamics* of the data is now much easier to see due to the separation of clusters given by our 2D landmarks' placement. The point made is that the freedom of landmark placement of LD-tSNE allows one to separate the issues of spatial disentanglement of samples in the projection (done by the landmark placement) from monitoring the dynamics of the data (taken care of by LD-tSNE). The Appendix (supplementary material) shows additional information on this and related experiments.

6. Conclusion

We have presented two projection methods that leverage the good neighborhood-preservation ability of t-SNE for dynamic (time-dependent) data. For this, we use guidance in the form of landmarks

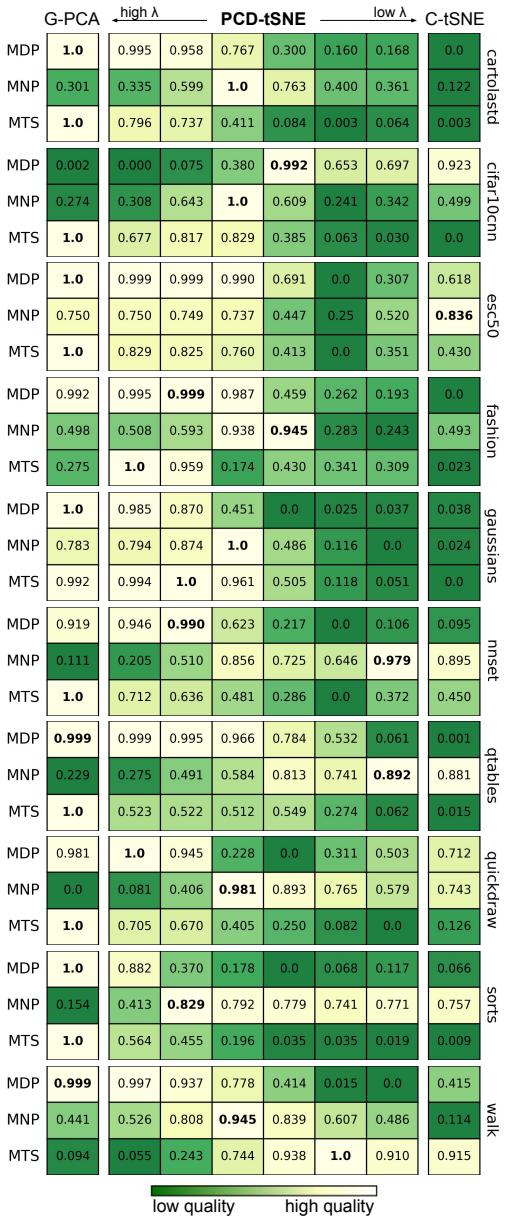


Figure 7: *Mean distance preservation (MDP), mean neighborhood preservation (MNP), and mean temporal stability (MTS) per dataset, as in Figs. 5 and 6, but normalized over the 8 runs in each subplot. The leftmost column is for G-PCA. The next 6 columns are for PCD-tSNE with* $\lambda \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$. *The rightmost column is for C-tSNE. By changing* λ*, PCD-tSNE generates a smooth gradient, simulating G-PCA and C-tSNE at the extremes and producing hybrids in-between (Sec. 3.2). The best balance between all metric classes is often found in this compromise.*

(for our first method, LD-tSNE), respectively attractors to principal vectors (for our second method, PCD-tSNE). We compared our methods against 11 dynamic projection techniques on 10 datasets using 8 spatial quality and 4 stability metrics. The comparison showed that PCD-tSNE scores better than all compared methods on the combined spatial quality and stability criteria. LD-tSNE obtained second-best scores on neighborhood preservation, allowing flexible placement

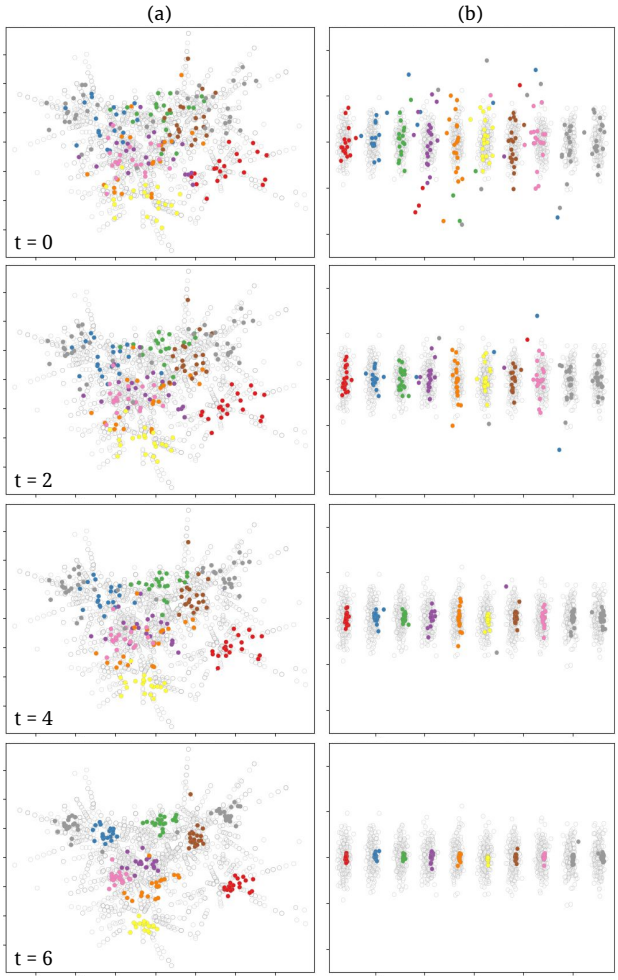


Figure 8: *Projection of the* gaussians *dataset with landmarks (gray points) placed (a) by G-PCA and (b) manually according to cluster label. Points are colored by cluster label. The implosion dynamics known to be present in the data is visible in both cases. Yet, the manual landmark placement creates a less cluttered view.*

of landmarks to drive the shape of the resulting dynamic projection. While our work – for sure – does not solve the problem of dynamic projection of high-dimensional data, we argue that our methods bring added value to users interested in this goal.

We next aim to extend our methods to handle streaming data. Adapting our work to use deep learning, similar to [EHT20], would lead to high-quality and computationally scalable dynamic projections. Additional validation of our methods on more datasets, and with concrete use-cases and user tasks, is also important. Finally, developing new metrics to measure the quality of dynamic projections for specific tasks, thereby extending the insights in [NA19] for the dynamic case, is a long-term goal we aim to pursue.

7. Acknowledgements

This study was financed in part by CAPES (Finance Code 001) and CNPq (Process 304336/2019-0).

References

[BCS96] BECKER R. A., CLEVELAND W. S., SHYU M.-J.: The visual design and control of trellis display. *JCGS 5*, 2 (1996), 123–155. 2

[BFHL17] BOYTSOV A., FOUQUET F., HARTMANN T., LETRAON Y.: Visualizing and exploring dynamic high-dimensional datasets with LION-tSNE, 2017. arXiv:1708.04983. 3

[BPV*03] BENGIO Y., PAIEMENT J.-F., VINCENT P., DELALLEAU O., ROUX N. L., OUIMET M.: Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In *Proc. NIPS* (2003), pp. 177–184. 3

[BSH16] BACH B., SHI C., HEULOT N.: Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE TVCG 22*, 1 (2016), 559–568. 2

[BSP*20] BRICH N., SCHULZ C., PETER J., KLINGERT W., SCHENK M., WEISKOPF D., KRONE M.: Visual analysis of multivariate intensive care surveillance data. In *Proc. Eurographics Workshop on Visual Computing for Biology and Medicine* (2020). 2

[BWS12] BERNARD J., WILHELM N., SCHERER M.: TimeSeriesPaths: Projection-based explorative analysis of multivariate time series data. *Journal of WSCG* (2012), 97–106. 2

[CD06] CLINE A. K., DHILLON I. S.: *Computation of the Singular Value Decomposition*. CRC Press, jan 2006. 5

[CG15] CUNNINGHAM J., GHAHRAMANI Z.: Linear dimensionality reduction: Survey, insights, and generalizations. *JMLR 16* (2015), 2859–2900. 1, 2

[DT03] DE SILVA V., TENENBAUM J. B.: Global versus local methods in nonlinear dimensionality reduction. *Advances in Neural Information Processing Systems* (2003). 3

[DT04] DE SILVA V., TENENBAUM J. B.: *Sparse multidimensional scaling using landmark points*. Tech. Rep. 6, Stanford University, 2004. 3

[DT05] DE SILVA V., TENENBAUM J. B.: Selecting landmark points for sparse manifold learning. *Advances in Neural Information Processing Systems* (2005), 1241–1248. 3, 9

[EHT20] ESPADOTO M., HIRATA N. S. T., TELEA A. C.: Deep learning multidimensional projections. *Information Visualization 19*, 3 (2020), 247–269. 3, 10

[Ekl12] EKLUND A.: Beeswarm: The bee swarm plot, an alternative to stripchart, 2012. R package version 0.1 5. 7

[EMK*19] ESPADOTO M., MARTINS R. M., KERREN A., HIRATA N. S. T., TELEA A. C.: Towards a quantitative survey of dimension reduction techniques. *IEEE TVCG* (2019). 1, 2, 5, 6, 9

[FLM*18] FUJIWARA T., LI J. K., MUBARAK M., ROSS C., CAROTHERS C. D., ROSS R. B., MA K. L.: A visual analytics system for optimizing the performance of large-scale networks in supercomputing systems. *Visual Informatics 2*, 1 (2018), 98–110. 3

[Fod02] FODOR I. K.: *A Survey of Dimension Reduction Techniques*. Tech. Rep. UCRL-ID-148494, Lawrence Livermore National Labs, 2002. 2

[FSS*20] FUJIWARA T., SHILPIKA, SAKAMOTO N., NONAKA J., YAMAMOTO K., MA K. L.: A visual analytics framework for reviewing multivariate time-series data with dimensionality reduction. *IEEE TVCG* (2020). 3

[GG19] GOMIDE H., GUALBERTO A.: caRtola, 2019. https://github.com/henriquepgomide/caRtola. 6

[Glo19] GLOBO.COM: Cartola football portal, 2019. https://globoesporte.globo.com/cartola-fc. 6

[HKF16] HAROZ S., KOSARA R., FRANCONERI S.: The connected scatterplot for presenting paired time series. *IEEE TVCG* (2016). 3

[HWX*10] HU Y., WU S., XIA S., FU J., CHEN W.: Motion track: Visualizing variations of human motion data. In *Proc. IEEE Pacific VIS* (2010), pp. 153–160. 3

[ID90] INSELBERG A., DIMSDALE B.: Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proc. IEEE VIS* (1990), pp. 361–378. 2

[JCC*11] JOIA P., COIMBRA D., CUMINATO J. A., PAULOVICH F. V., NONATO L. G.: Local affine multidimensional projection. *IEEE TVCG 17*, 12 (2011), 2563–2571. 2, 3, 6, 7, 9

[JFSK16] JÄCKLE D., FISCHER F., SCHRECK T., KEIM D. A.: Temporal MDS plots for analysis of multivariate data. *IEEE TVCG 22*, 1 (2016), 141–150. 3

[Jol86] JOLLIFFE I.: *Principal Component Analysis*. Springer Verlag, 1986. 1, 2

[JRK*16] JONGEJAN J., ROWLEY H., KAWASHIMA T., KIM J., FOX-GIEG N.: The Quick, Draw! - A.I. Experiment. https://quickdraw.withgoogle.com/, 2016. 6

[KHS*17] KRUIGER J., HASSOUMI A., SCHULZ H.-J., TELEA A., HURTER C.: Multidimensional data exploration by explicitly controlled animation. *Informatics 4*, 26 (2017). 8

[Kre05] KRESSNER D.: Numerical methods for general and structured eigenvalue problems. *Lecture Notes in Computational Science and Engineering 46* (01 2005). 5

[Kri09] KRIZHEVSKY A.: *Learning multiple layers of features from tiny images*. Tech. Rep. TR-2009, University of Toronto, 2009. 6, 8

[KTH17] KRUIGER J. F., TELEA A. C., HURTER C.: Projection navigation in extremely large datasets (PNIELD). In *Proc. EuroVis: Posters* (2017), Eurographics Association, pp. 109–111. 3

[LC10] LECUN Y., CORTES C.: MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. 6

[LMW*17] LIU S., MALJOVEC D., WANG B., BREMER P. T., PASCUCCI V.: Visualizing high-dimensional data: Advances in the past decade. *IEEE TVCG 23*, 3 (2017), 1249–1268. 2

[LRH*19] LINDERMAN G. C., RACHH M., HOSKINS J. G., STEINERBERGER S., KLUGER Y.: Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods 16*, 3 (2019), 243–245. arXiv:1712.09005. 4

[LS17] LINDERMAN G. C., STEINERBERGER S.: Clustering with T-SNE, provably. *arXiv 1* (2017), 1–15. arXiv:1706.02582. 4

[MCMT14] MARTINS R. M., COIMBRA D. B., MINGHIM R., TELEA A. C.: Visual analysis of dimensionality reduction quality for parameterized projections. *CAG 41*, 1 (2014), 26–42. 5

[MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426. 2

[MMT15] MARTINS R. M., MINGHIM R., TELEA A. C.: Explaining neighborhood preservation for multidimensional projections. *CGVC* (2015). 5

[NA19] NONATO L. G., AUPETIT M.: Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG 25*, 8 (2019), 2650–2673. 1, 2, 3, 10

[NFH*18] NEVES T. T., FADEL S. G., HILASACA G. M., FATORE F. M., PAULOVICH F. V.: UPDis: A user-assisted projection technique for distance information. *Information Visualization 17*, 4 (2018), 269–281. 3

[NMC*20] NEVES T. T., MARTINS R. M., COIMBRA D. B., KUCHER K., KERREN A., PAULOVICH F.: Xtreaming: An incremental multidimensional projection technique and its application to streaming data. arXiv:2003.09017. 3

[PdRDK99] PEKALSKA E., DE RIDDER D., DUIN R. P., KRAAIJVELD M. A.: A new method of generalizing Sammon mapping with application to algorithm speed-up. In *Proc. ASCI* (1999), pp. 221–228. 3, 7, 9

[PHL*16] PEZZOTTI N., HÖLLT T., LELIEVELDT B., EISEMANN E., VILANOVA A.: Hierarchical stochastic neighbor embedding. *Computer Graphics Forum 35*, 3 (2016), 21–30. 9

[Pic15] PICZAK K. J.: ESC: Dataset for Environmental Sound Classification. In *Proc. ACM MM* (2015), pp. 1015–1018. 6

[PNML08] PAULOVICH F. V., NONATO L. G., MINGHIM R., LEVKOWITZ H.: Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE TVCG 14*, 3 (2008), 564–575. 2, 3

[PSZ19] POLIČAR P. G., STRAŽAR M., ZUPAN B.: Embedding to reference t-sne space addresses batch effects in single-cell classification. In *Discovery Science* (2019), Springer, pp. 246–260. 3

[RC94] RAO R., CARD S. K.: The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proc. SIGCHI* (1994), pp. 318–322. 2

[RFaT16] RAUBER P. E., FALCÃO A. X., TELEA A. C.: Visualizing time-dependent data using dynamic t-SNE. In *Proc. EuroVis: Short Papers* (2016), Eurographics Association, pp. 73–77. 1, 2, 3, 4, 5, 6

[RFFT17] RAUBER P. E., FADEL S. G., FALCÃO A. X., TELEA A. C.: Visualizing the hidden activity of artificial neural networks. *IEEE TVCG 23*, 1 (2017), 101–110. 6, 7, 8

[RLL*08] ROSS D. A., LIM J., LIN R.-S., , YANG M.-H.: Incremental learning for robust visual tracking. *International Journal of Computer Vision 77*, 1-3 (2008), 125–141. 3

[SNLH09] SIPS M., NEUBERT B., LEWIS J., HANRAHAN P.: Selecting good views of high-dimensional data using class consistency. *Comp Graph Forum 28*, 3 (2009), 831–838. 2

[SVPM14] SORZANO C., VARGAS J., PASCUAL-MONTANO A.: A survey of dimensionality reduction techniques, 2014. arXiv:1403.2877. 1, 2, 3

[TBB*10] TATU A., BAK P., BERTINI E., KEIM D., SCHNEIDEWIND J.: Visual quality metrics and human perception: An initial study on 2D projections of large multidimensional data. In *Proc. AVI* (2010), pp. 49–56. 2

[van15] VAN DER MAATEN L.: Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research 15* (2015), 3221–3245. 4

[VCP13] VLADYMYROV M., CARREIRA-PERPIÑÁN M. Á.: Locally linear landmarks for large-scale manifold learning. In *Machine Learning and Knowledge Discovery in Databases* (2013), pp. 256–271. 3

[VCT18] VERNIER E., COMBA J., TELEA A.: Quantitative comparison of dynamic treemaps for software evolution visualization. In *Proc. IEEE VISSOFT* (2018). 2

[VCT20] VERNIER E., COMBA J., TELEA A.: Additional resources repository. https://eduardovernier.github.io/guided-dynamic-projections-resources/, 2020. 2, 5, 6, 7

[VGd*20] VERNIER E., GARCIA R., DA SILVA I., COMBA J., TELEA A.: Quantitative evaluation of time-dependent multidimensional projection techniques. *Computer Graphics Forum 39*, 3 (2020), 241–252. 1, 2, 3, 5, 6, 7

[vH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-sne. *Journal of Machine Learning Research 9* (2008), 2579–2605. 1, 2, 4

[VK06] VENNA J., KASKI S.: Visualizing gene interaction graphs with local multidimensional scaling. In *Proc. ESANN* (2006), pp. 557–562. 2

[vPVdH09] VAN DER MAATEN L., POSTMA E., VAN DEN HERIK J.: Dimensionality reduction: A comparative review. *JMLR 10* (2009), 66–71. 2

[VSC*20] VERNIER E., SONDAG M., COMBA J., SPECKMANN B., TELEA A., VERBEEK K.: Quantitative comparison of time-dependent treemaps. *Computer Graphics Forum 39*, 3 (2020), 393–404. 2

[Wat89] WATKINS C.: Learning from delayed rewards, 1989. Ph.D. thesis, Cambridge University, UK. 6

[WFC*18] WANG Y., FENG K., CHU X., ZHANG J., FU C. W., SEDLMAIR M., YU X., CHEN B.: A Perception-Driven Approach to Supervised Dimensionality Reduction for Visualization. *IEEE Transactions on Visualization and Computer Graphics 24*, 5 (2018), 1828–1840. 5

[XRV17] XIAO H., RASUL K., VOLLGRAF R.: Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747. 6