

Self-Supervised Dimensionality Reduction with Neural Networks and Pseudo-labeling

Mateus Espadoto^{1,3}^a, Nina S. T. Hirata¹^b and Alexandru C. Telea²^c

¹*Institute of Mathematics and Statistics, University of São Paulo, Brazil*

²*Department of Information and Computing Sciences, University of Utrecht*

³*Johann Bernoulli Institute, University of Groningen*
{mespadot, nina}@ime.usp.br, a.c.telea@uu.nl

Keywords: Dimensionality Reduction, Machine Learning, Neural Networks, Autoencoders

Abstract: Dimensionality reduction (DR) is used to explore high-dimensional data in many applications. Deep learning techniques such as autoencoders have been used to provide fast, simple to use, and high-quality DR. However, such methods yield worse visual cluster separation than popular methods such as t-SNE and UMAP. We propose a deep learning DR method called Self-Supervised Network Projection (SSNP) which does DR based on pseudo-labels obtained from clustering. We show that SSNP produces better cluster separation than autoencoders, has out-of-sample, inverse mapping, and clustering capabilities, and is very fast and easy to use.

1 INTRODUCTION

Analyzing high-dimensional data, a central task in data science and machine learning, is challenging due to the large size of the data both in the number of observations and measurements recorded per observation (also known as dimensions, features, or variables). As such, visualization of high-dimensional data has become an important, and active, field of information visualization (Kehrer and Hauser, 2013; Liu et al., 2015; Nonato and Aupetit, 2018; Espadoto et al., 2019a).

Dimensionality reduction (DR) methods, also called projections, have gained a particular place in the set of visualization methods for high-dimensional data. Compared to other techniques such as glyphs (Yates et al., 2014), parallel coordinate plots (Inselberg and Dimsdale, 1990), table lenses (Rao and Card, 1994; Telea, 2006), and scatter plot matrices (Becker et al., 1996), DR methods scale visually far better both on the number of observations and dimensions, allowing the visual exploration of datasets having thousands of dimensions and hundreds of thousands of samples. Many DR techniques have been proposed (Nonato and Aupetit, 2018; Espadoto et al., 2019a), with

PCA (Jolliffe, 1986), t-SNE (Maaten and Hinton, 2008), and UMAP (McInnes and Healy, 2018) having become particularly popular.

Neural networks, while very popular in machine learning, are less commonly used for DR, with self-organizing maps (Kohonen, 1997) and autoencoders (Hinton and Salakhutdinov, 2006) being notable examples. While having useful properties such as ease of use, speed, and out-of-sample capability, they lag behind t-SNE and UMAP in terms of good visual cluster separation, which is essential for the interpretation of the created visualizations. More recent deep-learning DR methods include NNP (Espadoto et al., 2020), which learns how to mimic existing DR techniques, and ReNDA (Becker et al., 2020), which uses two neural networks to accomplish DR.

DR methods aim to satisfy multiple functional requirements (quality, cluster separation, out-of-sample support, stability, inverse mapping) and non-functional ones (speed, ease of use, genericity). However, to our knowledge, none of the existing methods scores well on all such criteria (Espadoto et al., 2019a). In this paper, we propose to address this by a new DR method based on a single neural network trained with a dual objective, one of reconstructing the input data, as in a typical autoencoder, and the other of data classification using *pseudo-labels* cheaply computed by a clustering algorithm to introduce neighborhood information, since intuitively,

^a <https://orcid.org/0000-0002-1922-4309>

^b <https://orcid.org/0000-0001-9722-5764>

^c <https://orcid.org/0000-0003-0750-0502>

clustering gathers and aggregates fine-grained distance information at a higher level, telling how *groups* of samples are similar to each other (at cluster level), and this aggregated information helps us next in producing projections which reflect this higher-level similarity.

Our method aims to *jointly* cover all the following characteristics, which, to our knowledge, is not yet achieved by existing DR methods:

Quality (C1): We provide better cluster separation than standard autoencoders, and close to state-of-the-art DR methods, as measured by well-known metrics in DR literature;

Scalability (C2): Our method can do inference in linear time in the number of dimensions and observations, allowing us to project datasets of up to a million observations and hundreds of dimensions in a few seconds using commodity GPU hardware;

Ease of use (C3): Our method produces good results with minimal or no parameter tuning;

Genericity (C4): We can handle any kind of high-dimensional data that can be represented as real-valued vectors;

Stability and out-of-sample support (C5): We can project new observations for a learned projection without recomputing it, in contrast to standard t-SNE and any other non-parametric methods;

Inverse mapping (C6): We provide, out-of-the-box, an inverse mapping from the low- to the high-dimensional space;

Clustering (C7): We provide, out-of-the-box, the ability to learn how to mimic clustering algorithms, by assigning labels to unseen data.

We structure our paper as follows: Section 2 presents the notation used and discusses related work on dimensionality reduction, Section 3 details our method, Section 4 presents the results that support our contributions outlined above, Section 5 discusses our proposal, and Section 6 concludes the paper.

2 BACKGROUND

We start with a few notations. Let $\mathbf{x} = (x^1, \dots, x^n)$, $x^i \in \mathbb{R}$, $1 \leq i \leq n$ be a n -dimensional (nD) real-valued sample, and let $D = \{\mathbf{x}_i\}$, $1 \leq i \leq N$ be a dataset of N samples. D can be seen as a table with N rows (samples) and n columns (dimensions). A DR, or projection, technique is a function

$$P: \mathbb{R}^n \rightarrow \mathbb{R}^q, \quad (1)$$

where $q \ll n$, and typically $q = 2$. The projection $P(\mathbf{x})$ of a sample $\mathbf{x} \in D$ is a qD point $\mathbf{p} \in \mathbb{R}^q$. Projecting a set D yields thus a qD scatter plot, which

we denote next as $P(D)$. The inverse of P , denoted $P^{-1}(\mathbf{p})$, maps a qD point to the high-dimensional space \mathbb{R}^n .

Dimensionality reduction: Tens of DR methods have been proposed in the last decades, as extensively described and reviewed in various surveys (Hoffman and Grinstein, 2002; Maaten and Postma, 2009; Engel et al., 2012; Sorzano et al., 2014; Liu et al., 2015; Cunningham and Ghahramani, 2015; Xie et al., 2017; Nonato and Aupetit, 2018; Espadoto et al., 2019a). Below we only highlight a few representative ones, referring for the others to the aforementioned surveys.

Principal Component Analysis (Jolliffe, 1986) (PCA) has been widely used for many decades due to its simplicity, speed, and ease of use and interpretation. PCA is also used as pre-processing step for other DR techniques that require the data dimensionality to be not too high (Nonato and Aupetit, 2018). PCA is highly scalable, easy to use, predictable, and has out-of-sample capability. Yet, due to its linear and global nature, PCA lacks on quality, especially for data of high intrinsic dimensionality, and is thus worse for data visualization tasks related to cluster analysis.

The methods of the Manifold Learning family, such as MDS (Torgerson, 1958), Isomap (Tenenbaum et al., 2000) and LLE (Roweis and Saul, 2000) with its variations (Donoho and Grimes, 2003; Zhang and Zha, 2004; Zhang and Wang, 2007) try to map to 2D the high-dimensional manifold on which data is embedded, and can capture nonlinear data structure. These methods are commonly used in visualization and they generally yield higher quality results than PCA. However, these methods can be hard to tune, do not have out-of-sample capability, do not work well for data that is not restricted to a 2D manifold, and generally scale poorly with dataset size.

Force-directed methods such as LAMP (Joia et al., 2011) and LSP (Paulovich et al., 2008) are popular in visualization and have also been used for graph drawing. They can yield reasonably high visual quality, good scalability, and are simple to use. However, they generally lack out-of-sample capability. In the case of LAMP, there exists a related inverse projection technique, iLAMP (Amorim et al., 2012), however, the two techniques are actually independent and not part of a single algorithm. Clustering-based methods, such as PBC (Paulovich and Minghim, 2006), share many of the characteristics of force-directed methods, such as reasonably good quality and lack of out-of-sample capability.

The SNE (Stochastic Neighborhood Embedding) family of methods, of which t-SNE (Maaten and Hinton, 2008) is arguably the most popular, have the key

Table 1: Summary of DR techniques and their characteristics.

Technique	Characteristic						
	Quality	Scalability	Ease of use	Genericity	Out-of-sample	Inverse mapping	Clustering
PCA	low	high	high	high	yes	yes	no
MDS	mid	low	low	low	no	no	no
Isomap	mid	low	low	low	no	no	no
LLE	mid	low	low	low	no	no	no
LAMP	mid	mid	mid	high	no	no	no
LSP	mid	mid	mid	high	no	no	no
t-SNE	high	low	low	high	no	no	no
UMAP	high	high	low	high	yes	no	no
Autoencoder	low	high	high	low	yes	yes	no
ReNDA	mid	low	low	mid	yes	no	no
NNP	high	high	high	high	yes	no	no
SSNP	high	high	high	high	yes	yes	yes

ability to visually segregate similar samples, thus being very good for cluster analysis. While praised for high visual quality, t-SNE has a (high) complexity of $O(N^2)$ in sample count, is very sensitive to small data changes, can be hard to tune (Wattenberg, 2016), and has no out-of-sample capability. Several refinements of t-SNE improve speed, such as tree-accelerated t-SNE (Maaten, 2014), hierarchical SNE (Pezzotti et al., 2016), and approximated t-SNE (Pezzotti et al., 2017), and various GPU accelerations of t-SNE (Pezzotti et al., 2020; Chan et al., 2018). Yet, these methods require quite complex algorithms, and still largely suffer from the aforementioned sensitivity, tuning, and out-of-sample issues. Uniform Manifold Approximation and Projection (UMAP) (McInnes and Healy, 2018) generates projections with comparable quality to t-SNE but much faster, and with out-of-sample capability. However, UMAP shares some disadvantages with t-SNE, namely the sensitivity to small data changes and parameter tuning difficulty.

Deep learning: Autoencoders (AE) (Hinton and Salakhutdinov, 2006; Kingma and Welling, 2013) aim to generate a compressed, low-dimensional representation on their bottleneck layers by training the network to reproduce its high-dimensional inputs on its outputs. Autoencoders produce results comparable to PCA on the quality criterion. However, they are easy to set up, train, and use, are easily parallelizable, and have out-of-sample and inverse mapping capabilities.

The ReNDA method (Becker et al., 2020) is a deep learning approach that uses two networks, improving on earlier work from the same authors. One network implements a nonlinear generalization of Fisher’s Linear Discriminant Analysis (Fisher, 1936); the other network is an autoencoder used as a regularizer. ReNDA scores well on predictability and has out-of-sample capability. However, it requires pre-training of each individual network and scalability is

quite low.

Recently, Espadoto *et al.* (Espadoto et al., 2020) proposed Neural Network Projections (NNP), where a training subset $D_s \subset D$ is selected and projected by any DR method to yield a so-called training projection $P(D_s) \subset \mathbb{R}^2$. D_s is fed into a regression neural network which is trained to output a 2D scatter plot $P_{nn}(D_s) \subset \mathbb{R}^2$ by minimizing the error between $P(D_s)$ and $P_{nn}(D_s)$. The trained network then projects unseen data by means of 2-dimensional, non-linear regression. NNP is very fast, simple to use, generic, and stable. However, the results show fuzzier cluster separation (lower quality) than the training projections, and cannot do inverse projections. The NNInv technique (Espadoto et al., 2019b), proposed by the same authors, provides inverse projection capability, but this requires setting up, training, and using a separate network.

Table 1 shows a summary of the aforementioned DR techniques and how they fare with respect to each characteristic. The last row highlights SSNP, our technique, which we describe next.

3 METHOD

As stated in Section 2, autoencoders have desirable DR properties (simplicity, speed, out-of-sample and inverse mapping capabilities), but create projections of lower quality than, *e.g.*, t-SNE and UMAP. We believe that the key difference is that autoencoders do not use neighborhood information during training, while t-SNE and UMAP (obviously) do that. This raises the following questions: *Could autoencoders produce better projections if using neighborhood information?* and, if so, *How to inject neighborhood information during autoencoder training?* Our technique answers both questions by using a network architecture with a *dual* optimization target. First, we

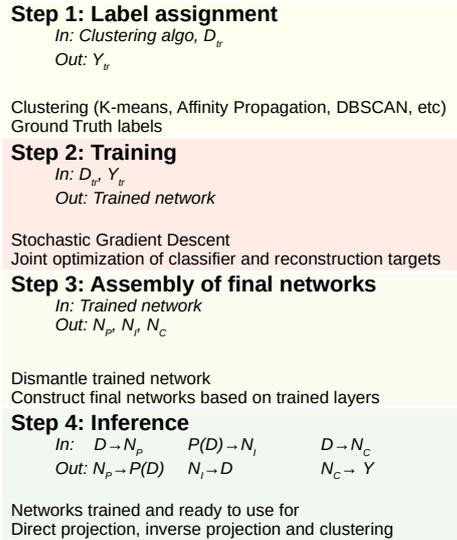


Figure 1: SSNP training-and-inference pipeline.

have a *reconstruction* target, exactly as in standard autoencoders; next, we use a *classification* target based on pseudo-labels assigned by some *clustering* algorithm *e.g.* K-means (Lloyd, 1982), Agglomerative Clustering (Kaufman and Rousseeuw, 2005), or any other way of assigning labels, including using “true” ground-truth labels if available.

Our key idea is that labels –ground-truth or given by clustering – are a high-level similarity measure between data samples which can be used to infer neighborhood information, *i.e.*, same-label data are more similar than different-label data. Since classifiers seek to learn a representation that separates input data based on labels, by adding an extra classifier target to an autoencoder, we learn how to project data with better cluster separation than standard autoencoders. We call our technique Self-Supervised Neural Projection (SSNP).

SSNP first takes a training set $D_{tr} \subset D$ and assigns to it pseudo-labels $Y_{tr} \in \mathbb{Z}$ by using some clustering technique. We then take samples $(\mathbf{x} \in D_{tr}, y \in Y_{tr})$ to train a neural network with two target functions, one for reconstruction, other for classification, which are then added together to form a joint loss. The errors from this joint loss are then back-propagated to the entire network, during training. This network (Figure 2) contains a two-unit bottleneck layer, same as an autoencoder, used to generate the 2D projection when in inference mode. After training, we ‘split’ the trained layers of the network to create three new networks for inference: a *projector* $N_p(\mathbf{x})$, an *inverse projector* $N_i(\mathbf{p})$ and, as a by-product, a *classifier* $N_c(\mathbf{x})$, which mimics clustering algorithm used

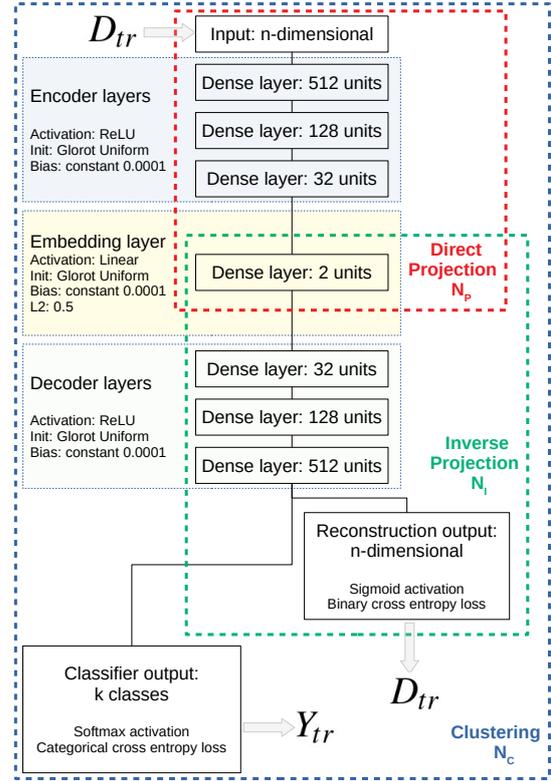


Figure 2: SSNP network architecture used for training.

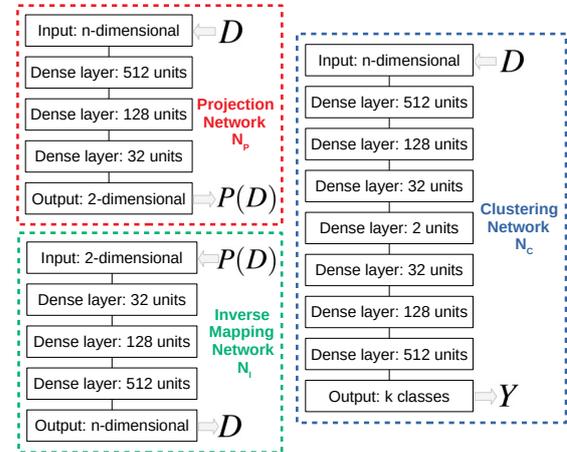


Figure 3: Three SSNP networks used during inference.

to create Y_{tr} (see Figure 3). The entire training-and-inference operation of SSNP is summarized in Figure 1.

Table 2: Projection quality metrics. Right column gives the metric ranges, with optimal values marked in bold.

Metric	Definition	Range
Trustworthiness (T)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^N \sum_{j \in U_i^{(K)}} (r(i, j) - K)$	[0, 1]
Continuity (C)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^N \sum_{j \in V_i^{(K)}} (\hat{r}(i, j) - K)$	[0, 1]
Neighborhood hit (NH)	$\frac{1}{N} \sum_{\mathbf{y} \in P(D)} \frac{y_k^j}{y_k}$	[0, 1]
Shepard diagram correlation (R)	Spearman's ρ of $(\ \mathbf{x}_i - \mathbf{x}_j\ , \ P(\mathbf{x}_i) - P(\mathbf{x}_j)\), 1 \leq i \leq N, i \neq j$	[0, 1]

4 RESULTS

To evaluate SSNP's performance we propose several experiments to compare it with other DR techniques using different datasets. We select the following evaluation metrics, which are widely used in the projection literature:

Trustworthiness (T) (Venna and Kaski, 2006): Measures the fraction of close points in D that are also close in $P(D)$. T tells how much one can trust that local patterns in a projection, *e.g.* clusters, represent actual patterns in the data. In the definition (Table 2), $U_i^{(K)}$ is the set of points that are among the K nearest neighbors of point i in the 2D space but not among the K nearest neighbors of point i in \mathbb{R}^n ; and $r(i, j)$ is the rank of the 2D point j in the ordered-set of nearest neighbors of i in 2D. We choose $K = 7$, in line with (Maaten and Postma, 2009; Martins et al., 2015);

Continuity (C) (Venna and Kaski, 2006): Measures the fraction of close points in $P(D)$ that are also close in D . In the definition (Table 2), $V_i^{(K)}$ is the set of points that are among the K nearest neighbors of point i in \mathbb{R}^n but not among the K nearest neighbors in 2D; and $\hat{r}(i, j)$ is the rank of the \mathbb{R}^n point j in the ordered set of nearest neighbors of i in \mathbb{R}^n . As with T , we choose $K = 7$;

Neighborhood Hit (NH) (Paulovich et al., 2008): Measures how well-separable labeled data is in a projection $P(D)$, in a rotation-invariant fashion, from perfect separation ($NH = 1$) to no separation ($NH = 0$). NH is the number y_k^j of the k nearest neighbors of a point $\mathbf{y} \in P(D)$, denoted by \mathbf{y}_k , that have the same label as \mathbf{y} , averaged over $P(D)$. In this paper, we used $k = 3$;

Shepard diagram correlation (R) (Joia et al., 2011): The Shepard diagram is a scatter plot of the pairwise distances between all points in $P(D)$ vs the corresponding distances in D . The closer the plot is to the main diagonal, the better overall distance preservation is. Plot areas below, respectively above, the diagonal show distance *ranges* for which false neighbors, respectively missing neighbors, occur. We measure how close a Shepard diagram is to the ideal

main diagonal line by computing its Spearman rank correlation R . A value of $R = 1$ indicates a perfect (positive) correlation of distances;

We next show how SSNP performs on different datasets when compared to t-SNE, UMAP, autoencoders, and NNP. We use different algorithms to generate pseudo-labels, and also use ground-truth labels. For conciseness, we name SSNP variants using K-means, agglomerative clustering and ground-truth labels as SSNP(Km), SSNP(Agg) and SSNP(GT), respectively. We use two types of datasets: synthetic blobs and real-world data. The synthetic blobs datasets are sampled from a Gaussian distribution where we vary the number of dimensions (100 and 700), the number of cluster centers (5 and 10), and use increasing values of the standard deviation σ . This yields datasets with cluster separation varying from very sharp to fuzzy clusters. All synthetic datasets have 5K samples.

Real-world datasets are selected from publicly available sources, matching the criteria of being high-dimensional, reasonably large (thousands of samples), and having a non-trivial data structure:

MNIST (LeCun and Cortes, 2010): 70K samples of handwritten digits from 0 to 9, rendered as 28x28-pixel gray scale images, flattened to 784-element vectors;

Fashion MNIST (Xiao et al., 2017): 70K samples of 10 types of pieces of clothing, rendered as 28x28-pixel gray scale images, flattened to 784-element vectors;

Human Activity Recognition (HAR) (Anguita et al., 2012): 10299 samples from 30 subjects performing activities of daily living used for human activity recognition, described with 561 dimensions.

Reuters Newswire Dataset (Thoma, 2017): 8432 observations of news report documents, from which 5000 attributes were extracted using TF-IDF (Salton and McGill, 1986), a standard method in text processing. This is a subset of the full dataset which contains data for the six most frequent classes only.

All datasets had their attributes rescaled to the in-

Table 3: Quality measurements for the synthetic blobs experiment with 100 and 700 dimensions, 5 and 10 cluster centers.

		100 dimensions								700 dimensions								
		5 clusters				10 clusters				5 clusters				10 clusters				
Algorithm	σ	T	C	R	N	T	C	R	N	σ	T	C	R	N	T	C	R	N
AE	1.3	0.923	0.938	0.547	1.000	0.958	0.963	0.692	1.000	1.6	0.909	0.914	0.739	1.000	0.953	0.955	0.254	1.000
t-SNE		0.937	0.955	0.818	1.000	0.967	0.977	0.192	1.000		0.917	0.951	0.362	1.000	0.960	0.976	0.346	1.000
UMAP		0.921	0.949	0.868	1.000	0.957	0.970	0.721	1.000		0.906	0.933	0.878	1.000	0.954	0.965	0.471	1.000
SSNP-KM		0.910	0.919	0.687	1.000	0.956	0.959	0.602	1.000		0.904	0.908	0.568	1.000	0.953	0.955	0.399	1.000
AE	3.9	0.919	0.926	0.750	1.000	0.959	0.963	0.484	1.000	4.8	0.910	0.914	0.615	1.000	0.953	0.954	0.354	1.000
t-SNE		0.931	0.953	0.707	1.000	0.966	0.978	0.227	1.000		0.914	0.950	0.608	1.000	0.960	0.977	0.331	1.000
UMAP		0.911	0.940	0.741	1.000	0.956	0.969	0.537	1.000		0.906	0.931	0.697	1.000	0.954	0.965	0.390	1.000
SSNP-KM		0.910	0.918	0.622	1.000	0.955	0.958	0.549	1.000		0.905	0.907	0.612	1.000	0.953	0.954	0.296	1.000
AE	9.1	0.905	0.901	0.569	1.000	0.938	0.945	0.328	0.999	11.2	0.911	0.906	0.600	1.000	0.955	0.954	0.382	1.000
t-SNE		0.913	0.951	0.533	1.000	0.948	0.974	0.254	1.000		0.914	0.950	0.492	1.000	0.959	0.977	0.296	1.000
UMAP		0.888	0.939	0.535	1.000	0.929	0.966	0.342	1.000		0.905	0.931	0.557	1.000	0.953	0.965	0.336	1.000
SSNP-KM		0.888	0.917	0.595	0.998	0.927	0.952	0.437	0.995		0.904	0.906	0.557	1.000	0.950	0.945	0.314	0.998

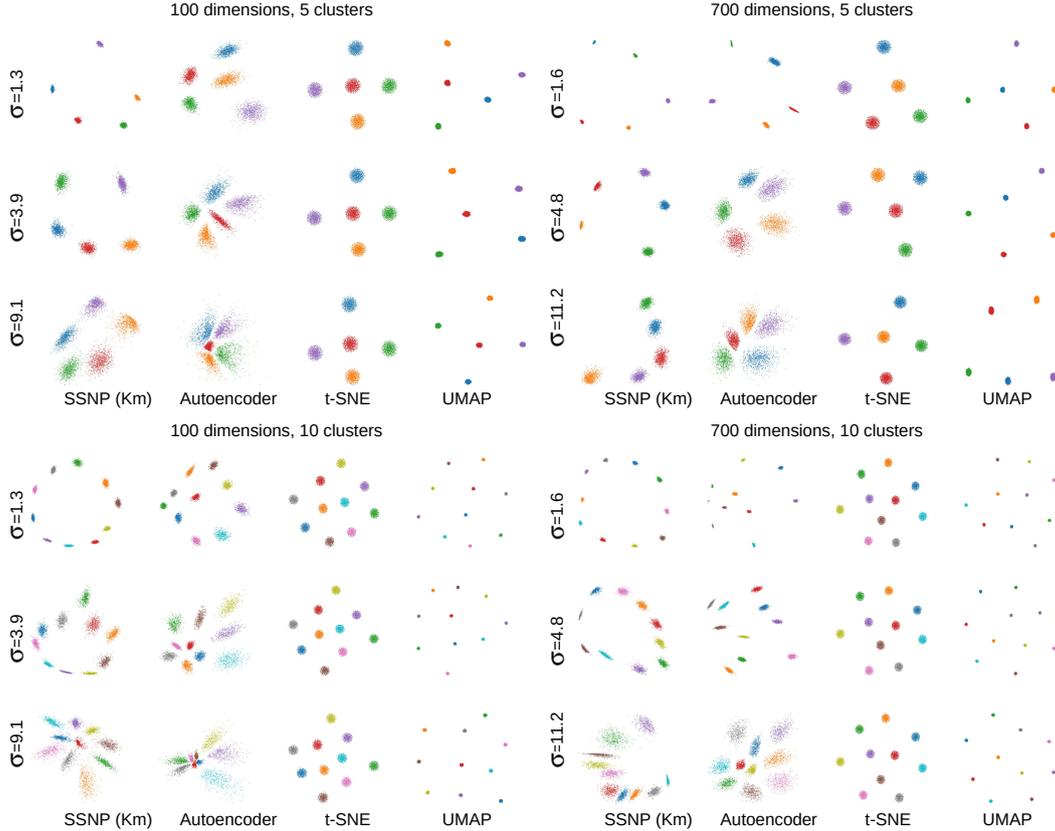


Figure 4: Projection of synthetic blobs datasets with SSNP(Km) and other techniques, with different number of dimensions and clusters. In each quadrant, rows show datasets having increasing standard deviation σ .

terval $[0, 1]$, to conform with the sigmoid activation function used at the reconstruction layer (see Figure 2).

4.1 Quality: Synthetic Datasets

Figure 4 shows the projection of the synthetic blob datasets with SSNP(Km) using the correct number of clusters, alongside Autoencoders, t-SNE and UMAP. We see that in most cases SSNP-Km shows better visual cluster separation than Autoencoders. Also, SSNP-Km preserves the spread of the data better than

t-SNE and UMAP, which can be seen by the fact that the projections using these look almost the same regardless of the standard deviation in the data. We omit the plots and measurements for NNP, since these are very close to the ones created by the technique it is trying to mimic, typically t-SNE— see *e.g.* (Espadoto et al., 2020).

Table 3 shows the quality measurements for this experiment for the datasets using 5 and 10 cluster centers. We see that SSNP performs very similarly quality-wise to AE, t-SNE, and UMAP. We will bring more insight to this comparison in Section 4.2, which

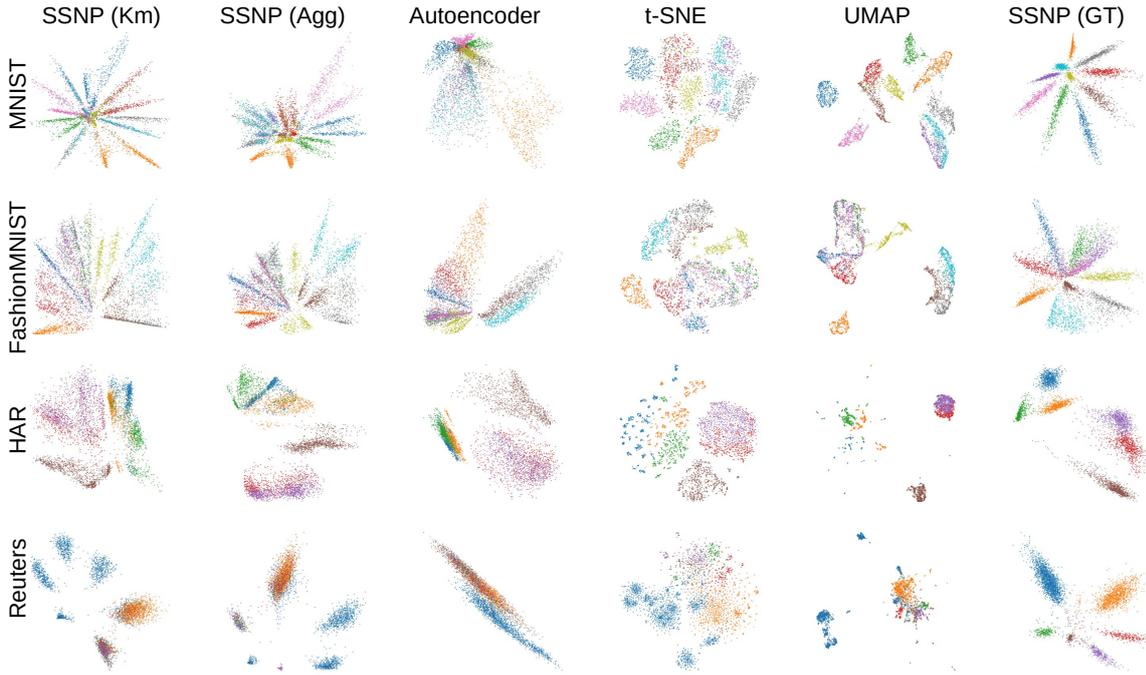


Figure 5: Projection of real-world datasets with SSNP and other techniques. Left to right: SSNP using K-means, SSNP using Agglomerative clustering, Autoencoder, t-SNE, UMAP, and SSNP using ground truth labels.

Table 4: Quality measurements for the real-world datasets.

Dataset	Method	T	C	R	NH
MNIST	SSNP(Km)	0.882	0.903	0.264	0.767
	SSNP(Agg)	0.859	0.925	0.262	0.800
	AE	0.887	0.920	0.009	0.726
	SSNP(GT)	0.774	0.920	0.398	0.986
	NNP	0.948	0.969	0.397	0.891
	t-SNE	0.985	0.972	0.412	0.944
	UMAP	0.958	0.974	0.389	0.913
FashionMNIST	SSNP(Km)	0.958	0.982	0.757	0.739
	SSNP(Agg)	0.950	0.978	0.707	0.753
	AE	0.961	0.977	0.538	0.725
	SSNP(GT)	0.863	0.944	0.466	0.884
	NNP	0.963	0.986	0.679	0.765
	t-SNE	0.990	0.987	0.664	0.843
	UMAP	0.982	0.988	0.633	0.805
HAR	SSNP(Km)	0.932	0.969	0.761	0.811
	SSNP(Agg)	0.926	0.964	0.724	0.846
	AE	0.937	0.970	0.805	0.786
	SSNP(GT)	0.876	0.946	0.746	0.985
	NNP	0.961	0.984	0.592	0.903
	t-SNE	0.992	0.985	0.578	0.969
	UMAP	0.980	0.989	0.737	0.933
Reuters	SSNP(Km)	0.794	0.859	0.605	0.738
	SSNP(Agg)	0.771	0.824	0.507	0.736
	AE	0.747	0.731	0.420	0.685
	SSNP(GT)	0.720	0.810	0.426	0.977
	NNP	0.904	0.957	0.594	0.860
	t-SNE	0.955	0.959	0.588	0.887
	UMAP	0.930	0.963	0.674	0.884

studies more challenging, real-world, datasets.

4.2 Quality: Real-World Datasets

Figure 5 shows in the first three columns the projection of real-world datasets by SSNP using pseudo-labels assigned by K-means and Agglomerative Clustering, alongside the projection created by an autoencoder. The next three columns show the same datasets projected by t-SNE, UMAP, and SSNP using ground-truth labels. Again, we omit the plots and measurements for NNP, since they are very close to the ones created by t-SNE and UMAP.

Figure 5 shows that SSNP with pseudo-labels shows better cluster separation than Autoencoders, and, for more challenging datasets, such as HAR and Reuters, SSNP with ground-truth labels looks better than t-SNE and UMAP. SSNP and Autoencoder were trained for 10 epochs in all cases. SSNP used twice the number of classes as the target number of clusters for the clustering algorithms used for pseudo-labeling. We see also that SSNP creates elongated clusters, in a star-like pattern. We believe this is due to the fact that one of the targets of the network is a *classifier*, which is trained to partition the space based on the data. This results in placing samples that are near a decision boundary between classes closer to the center of the star pattern; samples that are far away from a

decision boundary are placed near the tips of the star, according to its class.

Table 4 shows the quality measurement for this experiment. We see that SSNP with pseudo-labels consistently shows better cluster separation than Autoencoders, as measured by the NH , as well as better distance correlation, as measured by R . For the harder HAR and Reuters datasets, SSNP with ground-truth labels shows NH results that are competitive and even better than the ones for t-SNE and UMAP. For the T and C metrics, SSNP outperforms again Autoencoders in most cases; for FashionMNIST and HAR, SSNP yields T and C values close to the ones for NNP, t-SNE, and UMAP.

4.3 Computational Scalability

Table 5 shows the time required to set up SSNP and other projection techniques. For SSNP, this means training the network. For t-SNE, this means actual projection of the data, since this technique is non-parametric. All SSNP variants take far under a minute to set up, with SSNP(GT) being the fastest, as it does not need the clustering step. Of the pseudo-labeling varieties, SSNP(Km) is the fastest. We used 10K training samples, which is on the conservative side. In practice, we get good results (quality-wise) with as few as 1K samples.

Table 5: Set-up time for different methods, using 10K training samples, MNIST dataset. All SSNP variants and Autoencoders were trained for 10 epochs. t-SNE is here for reference only, since it is a non-parametric technique.

Method	Training time (s)
SSNP(GT)	6.029
SSNP(Km)	20.478
SSNP(Agg)	31.954
Autoencoder	3.734
UMAP	25.143
t-SNE	33.620
NNP(t-SNE)	51.181

Figure 6 shows the time needed to project up to 1M samples using SSNP and the other techniques. Being GPU-accelerated neural networks, SSNP, Autoencoders and NNP perform very fast, all being able to project up to 1M samples in a few seconds – an order of magnitude faster than UMAP, and over three orders of magnitude faster than t-SNE.

4.4 Inverse Projection

Figure 7 shows a set of digits from the MNIST dataset – both the actual images x and the ones obtained by inverse projection $P^{-1}(P(\mathbf{x}))$. We see that

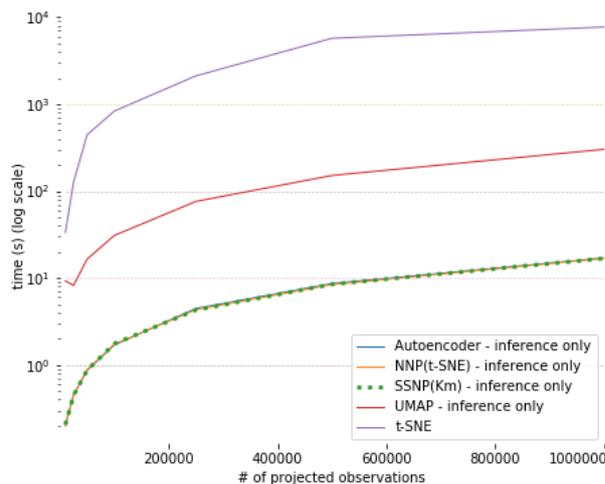


Figure 6: Inference time for SSNP and other techniques. All techniques trained with 10K samples from the MNIST dataset. Inference done on MNIST upsampled up to 1M samples.

SSNP(Km) yields results very similar to Autoencoders, so SSNP’s dual-optimization target succeeds in learning a good inverse mapping based on the direct mapping given by the pseudo-labels (Section 3). Table 6 strengthens this insight by showing the values for Mean Squared Error between original and inversely-projected data for SSNP(Km) and Autoencoder, which, again, are very similar. Furthermore, the SSNP MSE errors are of the same order of magnitude – that is, small – as those obtained by the recent NNInv technique and the older iLAMP (Amorim et al., 2012) one – compare Table 6 with Figure 2 in (Espadoto et al., 2019b), not included here for space reasons.

Table 6: Inverse projection Mean Square Error (MSE) for SSNP(Km) and Autoencoder, trained with 5K samples, tested with 1K samples.

Dataset	SSNP(Km)		Autoencoder	
	Train	Test	Train	Test
MNIST	0.0474	0.0480	0.0424	0.0440
FashionMNIST	0.0309	0.0326	0.0291	0.0305
HAR	0.0072	0.0074	0.0066	0.0067
Reuters	0.0002	0.0002	0.0002	0.0002

4.5 Data clustering

Table 7 shows how SSNP performs when doing classification or clustering, which corresponds respectively to its usage of pseudo-labels or ground-truth labels. We see that SSNP generates good results in both cases when compared to the ground-truth labels and, respectively, the underlying clustering algorithm. We stress that this is a side result of SSNP. While one

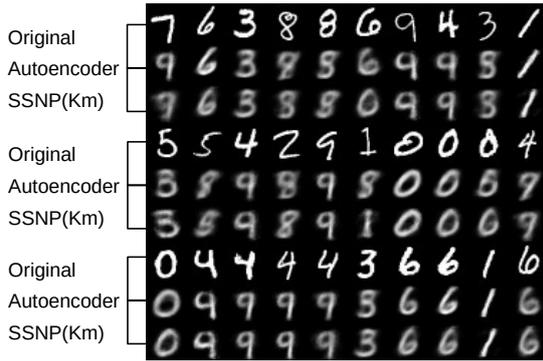


Figure 7: Sample images from MNIST inversely projected by SSNP and Autoencoder, both trained with 10 epochs and 5K samples. Bright images show the original data.

gets this for free, SSNP only *mimics* the underlying clustering algorithm that it learns, rather than doing clustering from scratch.

Table 7: Classification/clustering accuracy of SSNP when compared to ground truth (GT) and clustering labels (Km), trained with 5K observations, test with 1K observations.

Dataset	SSNP(GT)		SSNP(Km)	
	Train	Test	Train	Test
MNIST	0.984	0.942	0.947	0.817
FashionMNIST	0.866	0.815	0.902	0.831
HAR	0.974	0.974	0.931	0.919
Reuters	0.974	0.837	0.998	0.948

4.6 Implementation details

All experiments discussed above were run on a 4-core Intel Xeon E3-1240 v6 at 3.7 GHz with 64 GB RAM and an NVidia GeForce GTX 1070 GPU with 8 GB VRAM. Table 8 lists all open-source software libraries used to build SSNP and the other tested techniques. Our neural network implementation leverages the GPU power by using the Keras framework. The t-SNE implementation used is a parallel version of Barnes-Hut t-SNE (Ulyanov, 2016; Maaten, 2013), run on all four available CPU cores for all tests. The UMAP reference implementation is not parallel, but is quite fast (compared to t-SNE) and well-optimized. Our implementation, plus all code used in this experiment, are publicly available at <https://github.com/mespadoto/ssnp>.

Table 8: Software used for the evaluation.

Technique	Software used publicly available at
SSNP (our technique)	keras.io (TensorFlow backend)
Autoencoder	github.com/DmitryUlyanov/Multicore-t-SNE
t-SNE	github.com/lmcinnes/umap
UMAP	github.com/lmcinnes/umap
K-means	scikit-learn.org
Agglomerative Clustering	scikit-learn.org

5 DISCUSSION

We discuss next how SSNP performs with respect to the seven criteria laid out in Section 1.

Quality (C1): As shown in Figures 4 and 5, SSNP provides better cluster separation than Autoencoders, as measured by the selected metrics (Tables 3 and 4). The choice of clustering algorithm does not seem to be a key factor, with K-means and Agglomerative clustering yielding similar results for all metrics;

Scalability (C2): SSNP, while not as fast to train as standard Autoencoders, is still very fast, with most of the training time being used by clustering – visible by the fact that SSNP(GT)’s training time is close to Autoencoder’s one. In our experiments, K-means seems to be faster than Agglomerative clustering, being thus more suitable when training SSNP with very large datasets. Inference time for SSNP is very close to Autoencoders and NNP, and much faster than UMAP (let alone classical t-SNE), which shows SSNP’s suitability to cases where one needs to project large amounts of data, such as streaming applications;

Ease of use (C3): SSNP yielded good projection results with little training (10 epochs), little training data (5K samples) and a simple heuristic of setting the number of clusters for the clustering step to twice the number of expected clusters in the data. Other clustering techniques which do not require setting the number of clusters can be used, such as DBSCAN (Ester et al., 1996) and Affinity Propagation (Frey and Dueck, 2007), making SSNP usage even simpler. We see this experiments as part of future work;

Genericity (C4): We show results for SSNP with different types of high-dimensional data, namely tabular (HAR), images (MNIST, FashionMNIST), and text (Reuters). We believe this shows the versatility of the method;

Stability and out-of-sample support (C5): All measurements we show for SSNP are based on inference, *i.e.*, we pass the data through the trained network to compute them. This is evidence of the out-of-sample capability, which allows one to project new data without recomputing the projection, as is the case for t-SNE and other non-parametric methods;

Inverse mapping (C6): SSNP shows inverse mapping results which are, quality-wise, very similar to results from Autoencoders, NNInv and iLAMP, which is evidence of SSNP’s inverse mapping abilities;

Clustering (C7): SSNP is able to mimic the behavior of the clustering algorithm used as its input, as a byproduct of the training with labels. We show that SSNP produces competitive results when compared to pseudo- or ground truth labels. Although SSNP is not a clustering algorithm, it provides this for free (with no additional execution cost), which can be useful in cases where one wants to do both clustering and DR.

In addition to the good performance shown for the aforementioned criteria, a key strength of SSNP is its ability of performing all the operations described in Section 4 after a *single training phase*, which saves effort and time in cases where all or a subset of those results (*e.g.*, direct projection, inverse projection, clustering) are needed.

6 CONCLUSION

We presented a new dimensionality reduction technique called SSNP. Through several experiments, we showed that SSNP creates projections of high-dimensional data that obtain a better visual cluster separation than autoencoders, the technique that SSNP is closest to, and have similar (albeit slightly lower) quality to those obtained by state-of-the-art methods. SSNP is, to our knowledge, the only technique that jointly addresses *all* characteristics listed in Section 1 of this paper, namely producing projections that exhibit a good visual separation of similar samples, handling datasets of millions of elements in seconds, being easy to use (no complex parameters to set), handling generically any type of high-dimensional data, providing out-of-sample support, and providing an inverse projection function.

One low-hanging fruit is to study SSNP in conjunction with more advanced clustering algorithms than the K-means and agglomerative clustering used in this paper, yielding potentially even better visual cluster separation. A more ambitious, but realizable, goal is to have SSNP learn its pseudo-labeling during training and therefore remove the need for using a separate clustering algorithm. We plan to explore this directions in future work.

ACKNOWLEDGMENTS

This study was financed in part by FAPESP (2015/22308-2 and 2017/25835-9) and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

- Amorim, E., Brazil, E. V., Daniels, J., Joia, P., Nonato, L. G., and Sousa, M. C. (2012). iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In *Proc. IEEE VAST*, pages 53–62.
- Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Proc. Intl. Workshop on Ambient Assisted Living*, pages 216–223. Springer.
- Becker, M., Lippel, J., Stuhlsatz, A., and Zielke, T. (2020). Robust dimensionality reduction for data visualization with deep neural networks. *Graphical Models*, 108:101060.
- Becker, R., Cleveland, W., and Shyu, M. (1996). The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2):123–155.
- Chan, D., Rao, R., Huang, F., and Canny, J. (2018). T-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. In *Proc. SBAC-PAD*, pages 330–338.
- Cunningham, J. and Ghahramani, Z. (2015). Linear dimensionality reduction: Survey, insights, and generalizations. *JMLR*, 16:2859–2900.
- Donoho, D. L. and Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc Natl Acad Sci*, 100(10):5591–5596.
- Engel, D., Hattenberger, L., and Hamann, B. (2012). A survey of dimension reduction methods for high-dimensional data analysis and visualization. In *Proc. IRTG Workshop*, volume 27, pages 135–149. Schloss Dagstuhl.
- Espadoto, M., Hirata, N., and Telea, A. (2020). Deep learning multidimensional projections. *J. Information Visualization*. doi.org/10.1177/1473871620909485.
- Espadoto, M., Martins, R. M., Kerren, A., Hirata, N. S., and Telea, A. C. (2019a). Towards a quantitative survey of dimension reduction techniques. *IEEE TVCG*. Publisher: IEEE.
- Espadoto, M., Rodrigues, F. C. M., Hirata, N. S. T., Hirata Jr., R., and Telea, A. C. (2019b). Deep learning inverse multidimensional projections. In *Proc. EuroVA*. Eurographics.
- Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, pages 226–231.
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.
- Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315(5814):972–976. Publisher: AAAS.
- Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507. Publisher: AAAS.
- Hoffman, P. and Grinstein, G. (2002). A survey of visualizations for high-dimensional data mining. *Information*

- Visualization in Data Mining and Knowledge Discovery*, 104:47–82. Publisher: Morgan Kaufmann.
- Inselberg, A. and Dimsdale, B. (1990). Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proc. IEEE Visualization*, pages 361–378.
- Joia, P., Coimbra, D., Cuminato, J. A., Paulovich, F. V., and Nonato, L. G. (2011). Local affine multidimensional projection. *IEEE TVCG*, 17(12):2563–2571.
- Jolliffe, I. T. (1986). Principal component analysis and factor analysis. In *Principal Component Analysis*, pages 115–128. Springer.
- Kaufman, L. and Rousseeuw, P. (2005). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.
- Kehrer, J. and Hauser, H. (2013). Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE TVCG*, 19(3):495–513.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114. eprint: 1312.6114.
- Kohonen, T. (1997). *Self-organizing Maps*. Springer.
- LeCun, Y. and Cortes, C. (2010). MNIST handwritten digits dataset. <http://yann.lecun.com/exdb/mnist>.
- Liu, S., Maljovec, D., Wang, B., Bremer, P.-T., and Pascucci, V. (2015). Visualizing high-dimensional data: Advances in the past decade. *IEEE TVCG*, 23(3):1249–1268.
- Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Trans Inf Theor*, 28(2):129–137.
- Maaten, L. v. d. (2013). Barnes-hut-SNE. *arXiv preprint arXiv:1301.3342*.
- Maaten, L. v. d. (2014). Accelerating t-SNE using tree-based algorithms. *JMLR*, 15:3221–3245.
- Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *JMLR*, 9:2579–2605.
- Maaten, L. v. d. and Postma, E. (2009). Dimensionality reduction: A comparative review. Technical report, Tilburg University, Netherlands.
- Martins, R. M., Minghim, R., Telea, A. C., and others (2015). Explaining neighborhood preservation for multidimensional projections. In *CGVC*, pages 7–14.
- McInnes, L. and Healy, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426v1 [stat.ML]*.
- Nonato, L. and Apupetit, M. (2018). Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG*.
- Paulovich, F. V. and Minghim, R. (2006). Text map explorer: a tool to create and explore document maps. In *Proc. Intl. Conference on Information Visualisation (IV)*, pages 245–251. IEEE.
- Paulovich, F. V., Nonato, L. G., Minghim, R., and Levkowitz, H. (2008). Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE TVCG*, 14(3):564–575.
- Pezzotti, N., Höllt, T., Lelieveldt, B., Eisemann, E., and Vilanova, A. (2016). Hierarchical stochastic neighbor embedding. *Computer Graphics Forum*, 35(3):21–30.
- Pezzotti, N., Lelieveldt, B., Maaten, L. v. d., Höllt, T., Eisemann, E., and Vilanova, A. (2017). Approximated and user steerable t-SNE for progressive visual analytics. *IEEE TVCG*, 23:1739–1752.
- Pezzotti, N., Thijssen, J., Mordvintsev, A., Holtt, T., Lew, B. v., Lelieveldt, B., Eisemann, E., and Vilanova, A. (2020). GPGPU linear complexity t-SNE optimization. *IEEE TVCG*, 26(1):1172–1181.
- Rao, R. and Card, S. K. (1994). The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proc. ACM SIGCHI*, pages 318–322.
- Roweis, S. T. and Saul, L. L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. Publisher: American Association for the Advancement of Science.
- Salton, G. and McGill, M. J. (1986). *Introduction to modern information retrieval*. McGraw-Hill.
- Sorzano, C., Vargas, J., and Pascual-Montano, A. (2014). A survey of dimensionality reduction techniques. *arXiv:1403.2877 [stat.ML]*.
- Telea, A. C. (2006). Combining extended table lens and treemap techniques for visualizing tabular data. In *Proc. EuroVis*, pages 120–127.
- Tenenbaum, J. B., Silva, V. D., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.
- Thoma, M. (2017). The Reuters dataset.
- Torgerson, W. S. (1958). *Theory and Methods of Scaling*. Wiley.
- Ulyanov, D. (2016). Multicore-TSNE.
- Venna, J. and Kaski, S. (2006). Visualizing gene interaction graphs with local multidimensional scaling. In *Proc. ESANN*, pages 557–562.
- Wattenberg, M. (2016). How to use t-SNE effectively. <https://distill.pub/2016/misread-tsne>.
- Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*.
- Xie, H., Li, J., and Xue, H. (2017). A survey of dimensionality reduction techniques based on random projection. *arXiv:1706.04371 [cs.LG]*.
- Yates, A., Webb, A., Sharpnack, M., Chamberlin, H., Huang, K., and Machiraju, R. (2014). Visualizing multidimensional data with glyph SPLoMs. *Computer Graphics Forum*, 33(3):301–310.
- Zhang, Z. and Wang, J. (2007). MLLE: Modified locally linear embedding using multiple weights. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1593–1600.
- Zhang, Z. and Zha, H. (2004). Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing*, 26(1):313–338.