SN Computer Science (2023) 4:244
https://doi.org/10.1007/s42979-022-01661-5

ORIGINAL RESEARCH

Stabilizing and Simplifying Sharpened Dimensionality Reduction Using Deep Learning

Mateus Espadoto[1] · Youngjoo Kim[2] · Scott C. Trager[3] · Jos B. T. M. Roerdink[1] · Alexandru C. Telea[4]

Received: 20 June 2022 / Accepted: 30 December 2022

Abstract
Dimensionality reduction (DR) methods create 2D scatterplots of high-dimensional data for visual exploration. As such scatterplots are often used to reason about the cluster structure of the data, this requires DR methods with good cluster preservation abilities. Recently, Sharpened DR (SDR) was proposed to enhance the ability of existing DR methods to create scatterplots with good cluster structure. Following this, SDR-NNP was proposed to speed the computation of SDR by deep learning. However, both SDR and SDR-NNP require careful tuning of four parameters to control the final projection quality. In this work, we extend SDR-NNP to simplify its parameter settings. Our new method retains all the desirable properties of SDR and SDR-NNP. In addition, our method is stable *vs* setting all its parameters, making it practically a parameter-free method, and also increases the quality of the produced projections. We support our claims by extensive evaluations involving multiple datasets, parameter values, and quality metrics.

Keywords High-dimensional visualization · Dimensionality reduction · Mean shift · Neural networks

Introduction

The visual analysis of high-dimensional data is challenging due to its many observations (also known as points or samples) and values recorded per sample (also known as dimensions, features, or variables) [1–3]. Dimensionality reduction (DR), also known as projection, is particularly suited for such data, since DR methods scale visually to thousands of dimensions and hundreds of thousands of samples. DR techniques such as the well-known *t*-SNE [4] and UMAP [5] methods, can segregate *data clusters* into well-separated *visual clusters*, which enables one to reason about the former by seeing the latter, a property also known as preservation of *data structure* [6].

A recent survey [3] noted that many DR techniques score below *t*-SNE or UMAP in cluster segregation but have other important assets—simple usage and implementation, computational scalability, and out-of-sample behavior. Following this, [7] recently proposed Sharpened DR (SDR) to generically improve the cluster segregation ability of any DR technique by sharpening the input data

Mateus Espadoto, Youngjoo Kim, Scott C. Trager, Jos B. T. M. Roerdink, and Alexandru C. Telea have contributed equally to this work.

This article is part of the topical collection "Advances on Computer Vision, Imaging and Computer Graphics Theory and Applications" guest edited by Kadi Bouatouch, Augusto Sousa, Mounia Ziat, and Helen Purchase.

✉ Mateus Espadoto
mespadot@ime.usp.br

Youngjoo Kim
lyoungjookiml@gmail.com

Scott C. Trager
sctrager@astro.rug.nl

Jos B. T. M. Roerdink
j.b.t.m.roerdink@rug.nl

Alexandru C. Telea
a.c.telea@uu.nl

1 Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Nijenborgh 9, Groningen 9747, AG, The Netherlands

2 Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, São Paulo 05508-090, Brazil

3 Kapteyn Astronomical Institute, University of Groningen, Landleven 12 (Kapteynborg, 5419), Groningen 9747, AD, The Netherlands

4 Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, Utrecht 3584, CC, The Netherlands

Published online: 03 March 2023

by a variant of the Mean Shift (MS) algorithm [8]. However, SDR is impractical to use as MS is prohibitively expensive in high dimensions.

In a recent paper [9], we reduced the computational cost of SDR using deep learning. Our proposed method, called SDR-NNP, leverages an earlier DR method, called Neural Network Projection (NNP) [10], to learn the combined steps of data sharpening and projection. SDR-NNP has the following features—to our knowledge, not yet *jointly* achieved by existing DR methods:

Quality (C1): Better cluster separation than existing DR methods, as measured by well-known metrics in the DR literature;

Scalability (C2): Linear in sample and dimension counts, allowing the projection of datasets of up to a million samples and hundreds of dimensions in a few seconds on commodity GPU hardware;

Genericity (C3): Handles any real-valued (unlabeled) high-dimensional data;

Stability and out-of-sample (OOS) support (C4): Projects new samples for a learned projection without recomputing it, in contrast to standard *t*-SNE and any other non-parametric methods.

However, SDR-NNP depends on four parameters—the number of nearest neighbors k_s in the MS process, the number of MS data-sharpening iterations I, the so-called learning rate α (speed of MS), and the number of training epochs E. While SDR-NNP proposes a good default for E, it only suggests *ranges* from which users can pick I and α and does not further explore how to set k_s. Tuning each single parameter can change the projection, and also the projection quality measured by established metrics, in subtle ways. In practice, users have to examine different combinations of k_s, I, and α by trial-and-error. This is slow, since all these parameters affect the *training data* that SDR-NNP uses, i.e., one has to retrain the method after each parameter change. More importantly, if changing these parameters can lead to very different projections, then the entire goal of *stability*—that is, having a method that generates consistent results for a given input dataset—would be compromised. Simply put: Sharpening the data, as SDR-NNP does it, is useful and desired, but *only effective* in practice if it can be done in a stable, ideally parameter-free, manner.

In this paper, we address these issues by reducing SDR-NNP's four-parameter space to a *single* parameter. The new parameter K_m controls a K-means clustering process done in the input high-dimensional data. We control SDR's data-sharpening process based on the local homogeneity of neighborhoods in terms of the cluster labels they get assigned by K-means. Our new method, which we call α-SDR-NNP, keeps the quality (C1), scalability (C2), genericity (C3), and stability and OOS (C4) features of SDR-NNP listed above. Most importantly, however, the new method covers the following.

Ease of use (C5): Our new method is far stabler than SDR-NNP in terms of visual cluster separation and quality metrics of the computed projections for *both* changes in the four parameters k_s, I, α, and E it inherits from SDR-NNP *and* the new parameter K_m it adds. Practically put, our new method can be seen as parameter-free, thus stable in its application. Our method also increases the quality (C1) of the produced projections *vs* SDR-NNP for the same parameter values.

We structure this paper as follows: "Background" discusses related work on dimensionality reduction. Section "SDR-NNP and α-SDR-NNP Methods" details SDR-NNP and α-SDR-NNP. Section "Results" presents the results that support our above claims, including a detailed quantitative and qualitative comparison of SDR-NNP and α-SDR-NNP. Section "Discussion" discusses our two methods. Finally, Section "Conclusion" concludes the paper.

Background

Let $\mathbf{x} = (x^1, \dots, x^n)$, $x^i \in \mathbb{R}, 1 \leq i \leq n$ be an n-dimensional (nD) real-valued sample, and let $D = \{\mathbf{x}_j\}, 1 \leq j \leq N$ be a dataset of N samples. A DR technique is a function

$$P : \mathbb{R}^n \to \mathbb{R}^q, \tag{1}$$

where $q \ll n$, and typically $q = 2$. The projection $P(\mathbf{x})$ of a sample $\mathbf{x} \in D$ is a point $\mathbf{p} \in \mathbb{R}^q$. Projecting the whole set D yields a qD scatterplot denoted next as $P(D)$.

DR methods aim to satisfy multiple requirements. Table 1 outlines prominent ones present in several DR surveys [1–3, 11–16]. Besides these, DR techniques also require locality, steerability, and multilevel computation [2]. We do not focus on such additional requirements as these are less mainstream.

The quality (Q) and cluster separation (CS) requirements need additional explanations. Projection quality is assessed by *local* metrics that measure how a small neighborhood of points in D maps to a neighborhood in $P(D)$ and conversely. Local quality metrics include the following (see Table 2 for the formal definitions):

Trustworthiness T [17]: 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 represent actual data patterns. 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 $P(D)$;

Table 1 Summary of desirable requirements (characteristics) of DR methods

Requirement name	Description of the requirement
Quality (Q)	Captures *local* data structures well, as measured by the projection local-quality metrics in Table 2
Cluster separation (CS)	Captures data structures present at *larger* scales than local structures, *e.g.,* clusters, as visual clusters in the 2D scatterplot
Scalability (S)	Can project datasets of hundreds of dimensions and millions of samples in a few seconds on commodity hardware
Ease of use (EoU)	Has few (ideally: no) free parameters, which are intuitive and easy to tune to get the desired results
Genericity (G)	Can project any (real-valued) dataset, with or without labels
Out-of-sample (OOS)	Can fit new data in an existing projection. OOS projections are also *stable*—small input-data changes cause only small projection changes

Table 2 Local-quality metrics for projections. All metrics range in [0, 1] with 0 being lowest, and 1 being highest, quality

Metric	Definition
Trustworthiness (T)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in U_i^{(K)}} (r(i,j) - K)$
Continuity (C)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in V_i^{(K)}} (\hat{r}(i,j) - K)$
Neighborhood hit (NH)	$\frac{1}{N} \sum_{y \in P(D)} y_k^l / y_k$
Shepard diagram correlation (R)	Spearman's rank of $(\|\mathbf{x}_i - \mathbf{x}_j\|, \|P(\mathbf{x}_i) - P(\mathbf{x}_j)\|), 1 \le i \le N, i \ne j$

Continuity C [17]**:** Measures the fraction of close points in $P(D)$ that are also close in D. In the definition (Table 2), $V_i^{(K)}$ are the points in the K nearest neighbors of point i in D 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 D;

Neighborhood Hit NH [18]: Measures how well a projection $P(D)$ separates labeled data. NH is the number $\mathbf{y}_k^l$ of the k nearest neighbors of a point $\mathbf{y} \in P(D)$, denoted by $\mathbf{y}_k$, that have the same label l as $\mathbf{y}$, averaged over $P(D)$. Put simply: consider a projection, i.e., a 2D scatterplot $P(D)$, where every point has a label equal to the label the corresponding high-dimensional point projected there. If we assume a well-separated dataset in the high-dimensional space, i.e., a dataset where close points in this space have similar labels, then a good projection should keep this structure—that is, close points in the 2D scatterplot should also have similar labels. The usage and practical intuition behind the NH metric has been extensively explored in the DR literature; see for example [19]. By construction, the NH metric requires labeled datasets to be used.

Shepard diagram correlation R [20]**:** The Shepard diagram is a scatterplot of the pairwise distances between all points in $P(D)$ *vs* the corresponding distances in D. Points below, respectively above, the main diagonal show distance ranges for which false neighbors, respectively missing neighbors, occur. The closer the plot is to the main diagonal, the better the overall distance preservation is. The scatterplot's Spearman rank correlation R measures this—a value $R = 1$ indicates a perfect (positive) distance correlation.

All above metrics are *local*, i.e., capture preservation of data structure in D at the scale given by the neighborhood size K. In practice, what a 'good' K value is for a given dataset D is unknown. K can also vary locally within D as function of the point density. At a higher level, projections are used to reason about the *overall data structure* in D by creating, ideally, visual clusters that are as well separated in $P(D)$ as data clusters are in D, a property called *cluster separation* (CS). High-CS projections show, e.g., how many point clusters exist and how these correlate (or not) with labels or specific attributes [2], or predict how easy it is to train a classifier for D based on the CS in $P(D)$ [19]. In general, it is hard to design objective metrics for CS like one does for local quality, because a 'well separated data cluster' in D is not evident. Hence, CS is typically assessed on (labeled) datasets D for which the ground-truth data-separation is well known, e.g., MNIST [21].

We next discuss existing DR methods in the light of the requirements in Table 1. We group these into unsupervised and supervised methods, as follows.

Unsupervised methods: Principal Component Analysis [22] (PCA) is simple, fast, out-of-sample (OOS), and easy-to-interpret, also used as pre-processing for other DR techniques that require a moderate data dimensionality n [2]. Being linear and global, PCA has low quality and CS, especially for data of high intrinsic dimensionality.

MDS [23], Landmark MDS [24], Isomap [25], and LLE [26] with its variations [27–29] detect and project the (neighborhood of the) high-dimensional manifold on which data are embedded, and can capture nonlinear data structure. Such methods yield higher quality than PCA, but can be hard to tune, do not all support OOS, and do not work well for high-intrinsic-dimensional data.

Force-directed methods, such as LAMP [20] and LSP [18], yield good quality and good scalability, and are simple to use. Yet, not all force-directed methods have OOS ability. Clustering-based methods, such as PBC [30], share many features with force-directed methods, such as good quality, but also lack OOS.

Stochastic Neighborhood Embedding (SNE) methods, like the well-known *t*-SNE [4], have high overall quality and CS. Yet, *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 [31], and has no OOS. Tree-accelerated *t*-SNE [32], hierarchical SNE [33], approximated *t*-SNE [34], and various GPU variants of *t*-SNE [35, 36] improve scalability, but are algorithmically quite complex, and still have sensitivity, tuning, and OOS issues. Uniform Manifold Approximation and Projection (UMAP) [5] has comparable quality to *t*-SNE, is much faster, and has OOS. Still, UMAP is also sensitive to parameter tuning.

Autoencoders (AE) [37, 38] aim to generate a compressed, low-dimensional representation of the data in their bottleneck layers by training to reproduce the data input at the output. They have similar quality to PCA and are easy to set up, train, and use, are fast, and have OOS abilities. Self-organizing maps (SOM) [39] share with AE the ease of use, training, and speed. Yet, both AE and SOM lag behind *t*-SNE and UMAP in CS, which is, as explained, essential for interpreting projections.

Supervised methods: ReNDA [40] uses two neural networks to implement (1) a nonlinear generalization of Fisher's Linear Discriminant Analysis [41] and (2) an autoencoder, used for regularization. ReNDA scores well on predictability and has OOS, but needs pre-training of each individual network and has low scalability. Recently, Neural Network Projections (NNP) [10] proposed to select a subset $D_s \subset D$ to project by any DR method to yield a training projection $P(D_s) \subset \mathbb{R}^2$. The pair $(D_s, P(D_s))$ is then used to train a regression neural network. NNP is very fast, simple to use, generic, and has OOS. NNP's major limitation is a lower CS than its training projection $P(D_s)$.

SDR-NNP [9], our earlier method which we extend in this paper, effectively runs NNP on a training set of high-dimensional samples which is first sharpened by mean shift (described further below). SDR-NNP keeps all desirable features of NNP except ease of use: Sharpening requires carefully setting three parameters to get good final results. We describe SDR-NNP in full detail in "SDR-NNP and α-SDR-NNP Methods" as it forms the basis of our new technique α-SDR-NNP which solves the parameter setting problem.

Semi-supervised methods: The SSNP method [42] takes a mid-path between supervised methods (e.g., NNP) and unsupervised ones (e.g., AE). Similar to NNP, SSNP has an encoder–decoder architecture. Besides the standard reconstruction loss in autoencoders (AEs), SSNP adds a classification loss. This loss uses either ground-truth labels from the dataset D or pseudolabels computed from D by a clustering algorithm. That is, SSNP aims to jointly (a) reconstruct an input dataset from a low-dimensional (more precisely, 2D) representation and (b) classify the input samples based on their (pseudo)labels. The combination of both losses creates a projection which both preserves the original dimensions of the data (a) and also the coarse-scale similarity of the data points (b). SNP produces 2D projections which look quite similar to those created by our methods described in "SDR-NNP and α-SDR-NNP Methods". However, important differences exist:

- Our methods consists of two distinct operations: high-dimensional data sharpening, followed by projection. SSNP only performs the projection step;
- SSNP is a semi-supervised method that uses *only* label information to learn how to project. Our methods, like NNP, learn from a user-selected *projection technique*;
- SSNP and our methods use fundamentally different network architectures. SSNP uses two different networks for training and inference. We use a single architecture for training and inference;
- A key goal for SDR-NNP is to enhance separation between unlabeled data clusters, so that these can next be labeled by users (see "Case Study: Astronomical Datasets"). This is out of scope for SSNP.

Sharpening data: Finding *clusters* of similar data points is a key task in data science, addressed by tens of clustering methods [43, 44]. Mean Shift (MS) [8, 45, 46] is particularly relevant to our work. MS computes the kernel density estimation of a dataset D and next shifts points in D upstream along the density gradient. This effectively clusters D, with applications in image segmentation [8] and graph drawing [47]. Recently, Sharpened DR (SDR) [7] used MS for the first time to assist DR: A dataset D is *sharpened* by a few MS iterations, not to be confused with the *clustering* goal of the original MS. The sharpened dataset is next projected by a fast, easy-to-use, but potentially low-CS DR method. Sharpening 'preconditions' the used DR method to overcome its lack of CS. Yet, as MS is very slow for high-dimensional data, this makes SDR impractical.

Table 3 summarizes the DR techniques discussed above showing how they fare with respect to the requirements discussed earlier in this section. No reviewed method satisfies all the requirements optimally. We next describe our earlier method SDR-NNP (Table 3 one but last row) and our extension to it, α-SDR-NNP (Table 3 last row).

Table 3 Summary of DR techniques in "Background" and their features from Table 1

Method	Desirable characteristics of the method				
	Q	S	EoU	G	OOS
PCA	Low	High	High	High	Yes
MDS	Mid	Low	Low	Low	No
L-MDS	Mid	Mid	Low	Low	No
Isomap	Mid	Low	Low	Low	No
LLE	Mid	Low	Low	Low	No
LAMP	Mid	Mid	Mid	High	Yes
LSP	Mid	Mid	Mid	High	No
PBC	Mid	Mid	Mid	High	No
UMAP	High	High	Low	High	Yes
t-SNE	High	Low	Low	High	No
Autoencoders	Low	High	High	Low	Yes
SOM	Low	High	High	Low	No
ReNDA	Mid	Low	Low	Mid	Yes
NNP	High	High	High	High	Yes
SDR	High	Low	Mid	High	No
SDR-NNP	High	High	Low-mid	High	Yes
α-SDR-NNP	High	High	High	High	Yes

SDR-NNP and α-SDR-NNP Methods

As "Background" explained, SDR and NNP have complementary features: SDR yields good cluster separation (CS), while NNP is fast, easy-to-use, and has OOS ability. Our combined SDR-NNP technique joins these advantages and works as follows (see also Fig. 1, parts marked in blue). We use SDR to sharpen a small data subset to create an initial 2D projection ("Sharpened Dimensionality Reduction"). Next, we train NNP on the sharpened data and its 2D projection ("SDR-NNP") and use it to project the whole dataset. Section "-α-SDR-NNP" introduces α-SDR-NNP and outlines how this methods improves upon SDR-NNP.

Sharpened Dimensionality Reduction

SDR has two main components, as follows (for full details, see [7]):

Data sharpening: Given a dataset $D \in \mathbb{R}^n$, SDR computes its density using the kernel density estimator $\rho(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^+$ defined as

$$\rho(\mathbf{x}) = \sum_{\mathbf{y}\in N(\mathbf{x})} L\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{h}\right), \tag{2}$$

where $N(\mathbf{x})$ is the set of k_s-nearest neighbors of $\mathbf{x}$ in D; L is a parabolic kernel [48]; and h is the distance of $\mathbf{x}$ to its k_s^{th} (farthest) neighbor in $N(\mathbf{x})$. In other words, the 'bandwidth' of density estimation h, which further determines how data sharpening finds clusters in the input dataset, is locally controlled by the number of nearest neighbors k_s. As we shall see next in "-α-SDR-NNP", our new method simplifies this even further by removing the need to explicitly specify k_s.

Next, SDR shifts points $\mathbf{x} \in D$ using the update rule

$$\mathbf{x}^{next} = \mathbf{x} + \alpha \frac{\nabla\rho(\mathbf{x})}{\max\left(\|\nabla\rho(\mathbf{x})\|, \epsilon\right)}, \tag{3}$$

where $\alpha \in [0, 1]$ is a 'learning rate' parameter that controls the shift speed (higher values yield higher speed) and $\epsilon = 10^{-5}$ is a regularization parameter. After every update (Eqn. 3), the density ρ is computed again (Eqn. 2). This

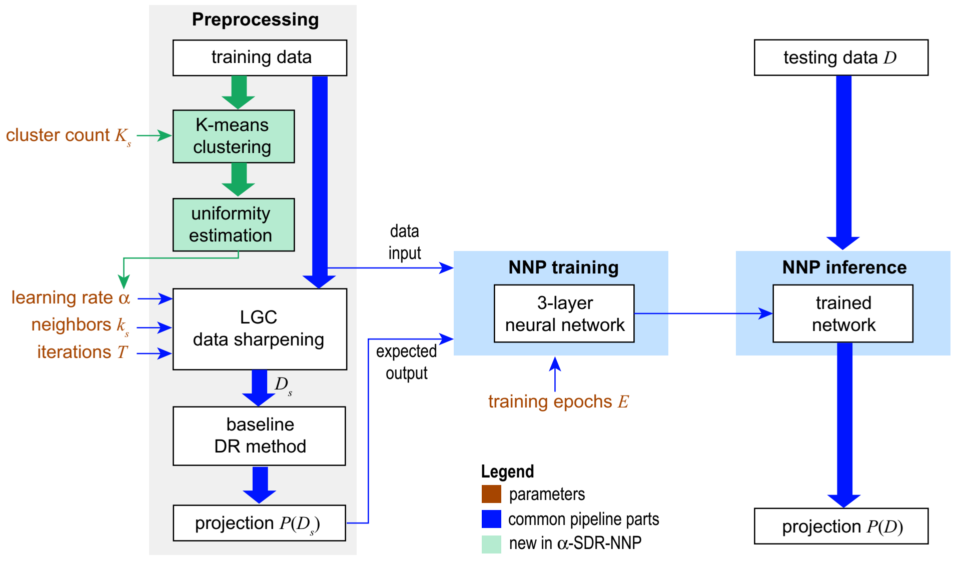


Fig. 1 Architecture of the SDR-NNP and α-SDR-NNP methods

sharpening approach is called Local Gradient Clustering (LGC) by analogy with Gradient Clustering (GC) [45].

SDR has three parameters: I (number of iterations); k_s (number of nearest neighbors); and α (learning rate), all marked in brown in Fig. 1. The SDR-NNP method uses $k_s \geq 50$ following [7], and setting α and I is discussed in "Results". The α-SDR-NNP method replaces the need to fiddle with these parameters (see "-α-SDR-NNP" and "Evaluation of α-SDR-NNP").

Projection: SDR takes the LGC-sharpened dataset D_s produced from the input dataset D and projects it by a projection method of choice P (typically fast but not necessarily OOS), called the *baseline* DR method next, to obtain a 2D projection $P(D_s)$. The data in D_s are better separated than in D due to LGC, which helps P to yield better cluster separation in $P(D_s)$ than in $P(D)$.

SDR-NNP

SDR-NNP uses SDR ("Sharpened Dimensionality Reduction") on a *small* data subset to obtain $P(D_s)$. To project the full dataset D, one next trains the NNP regressor [10] using D_s as input and $P(D_s)$ as output. The NNP network has three fully connected hidden layers with ReLU activation [49], initial weights set to He Uniform [50], and an initial bias value set to 0.0001. The output layer has two units, one per 2D coordinate, and uses sigmoid activation to constrain output values to [0, 1]. We used three different network sizes, namely, *x-small* (75, 30, 75 units per layer), *small* (150, 60, 150 units per layer), and *medium* (300, 120, 300 units per layer). We trained the network using the ADAM optimizer [51], as described in the NNP paper. After training, SDR-NNP has a regressor able to mimic the behavior of SDR for unseen data, thus adding OOS capability, and computational scalability to SDR.

α-SDR-NNP

A key challenge for the original SDR method [7] which is also shared by SDR-NNP is the *control* of the sharpening process. As outlined there, the projection results can be quite sensitive to the exact combination of α, k_s, and I parameters. To these, SDR-NNP introduces a fourth parameter, the number of training epochs E. Earlier results from both the original SDR and SDR-NNP, and as we also discuss next in "Evaluation of SDR-NNP", show that the hardest to control parameters are α and the number of nearest neighbors k_s. These two parameters influence most the visual cluster separation which is the main added value behind the SDR and SDR-NNP proposal. As Kim et al. [7] noted, these parameters are not fully independent—when changing k_s, one should also change the considered range for exploration of good α values. The interdependency of the same two parameters was observed when using LGC to bundle graph and trail drawings [47, 52].

To alleviate this problem, we must understand its causes. Consider a dataset $D \in \mathbb{R}^n$ and its density estimation ρ (Eqn. 2). For simplicity, we depict this for the 1D case in Fig. 2a. For the example in the figure, the original dataset had two quite well-separated clusters, shown by the red and cyan bars denoting high sample density ρ (Fig. 2a). LGC sharpens this density, practically separating the two high-density clusters even further (Fig. 2c). This is the desired outcome, since such well-separated data clusters will project to well-separated *visual* clusters further by SDR or SDR-NNP.

However, consider now a dataset D with the density ρ as in Fig. 2b. There is far less clear separation between data clusters here, shown by the fading-to-white bars below the density plot in image (b). From this input, LGC will create a sharpened dataset D_s looking as in Fig. 2b, i.e., very similar to the one where the density showed two very clearly separated peaks. This is due to the normalization of the gradient in the LGC update rule (Eq. 3) which means that small density variations have similar sharpening effects as large ones. This is clearly not desirable, since it separates the two density peaks in Fig. 2b too strongly. The result is *oversegmentation* of the projection $P(D_s)$, as observed in [7, 9]. Rather, we would like to obtain the dataset D_s in Fig. 2d which sharpens density *only* where it is clearly well separated from neighbor peaks, and leaves the between-peak region, where we cannot really identify two separated clusters, unchanged.

To achieve this, we need supplementary information, namely that indicated by the red and cyan bars in Fig. 2, i.e., where the high values of the dataset's density are located. This implies that we cannot use a single *global* parameter setting for LGC, but rather need to modulate LGC by such *local* information. We achieve this in our new α-NNP-SDR method as follows (see also Fig. 1, parts marked in green).

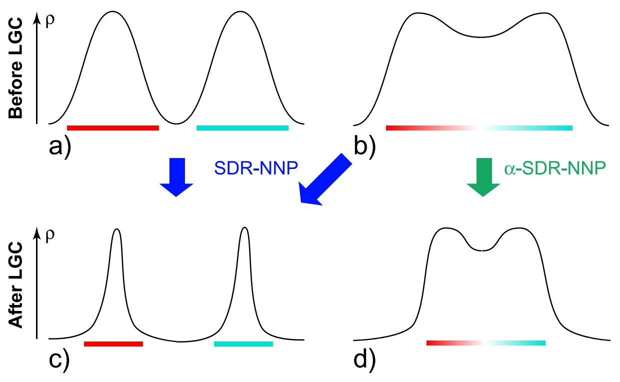


Fig. 2 Comparison of SDR-NNP (blue) with α-SDR-NNP (green). See "-α-SDR-NNP"

First, we use a simple clustering algorithm, K-means, to cluster NNP-SDR's training data D, with a user-set cluster-count parameter K_m. Next, during each LGC iteration (Eq. 3), we use the labels assigned by K-means to determine, for each point $\mathbf{x} \in D$, how *homogeneous* its neighborhood is, i.e., how many points share the same clustering label as $\mathbf{x}$. For this, we compute the NH metric (Table 2) but now using the pseudolabels assigned by K-means rather than ground-truth labels. Finally, we modify the LGC sharpening (Eq. 3) to use a *local* learning rate ($\alpha NH(\mathbf{x})$) instead of the global learning rate α. This decreases α in regions where NH is small, i.e., K-means finds that two data clusters are close to each other, such as the middle part of the density plot in Fig. 2b. Points there move far less when applying Eq. 3, yielding the desired LGC result in Fig. 2d. In areas where the density is very low (around the middle of Fig. 2a), NH is higher, so points there move fast, like in the original LGC, yielding the desired result in Fig. 2b. We show next in "Evaluation of α-SDR-NNP" that this allows us to control only the cluster count K_m of K-means to obtain more stable, and higher quality results, than when controlling α, k_s, and I in SDR-NNP.

Results

We measured the performance of SDR-NNP and α-SDR-NNP by the four metrics in Table 2 computed for $K = 7$, in line with [3, 12, 53]. Note that K, the number of nearest neighbors used to compute the metrics in Table 2, is smaller than k_s, the number of nearest neighbors used to evaluate ρ (Eqn. 2). Indeed, k_s needs to be relatively large to smooth out local noise in the computation of the gradient $\nabla\rho$; in contrast, K is typically set small to capture more local-quality aspects of a projection.

Evaluation used six publicly available real-world datasets (Table 4), all being reasonably high-dimensional and large (tens of dimensions, thousands of samples), and with a non-trivial data structure. All dimensions were rescaled to the [0, 1] range, to match NNP's sigmoid activation function in its output layer [10]. All experiments were run on a dual 16-core Intel Xeon Silver 4216 at 2.1 GHz with 256 GB RAM and an NVidia GeForce GTX 1080 Ti GPU with 11 GB VRAM. SDR was implemented in C++ using Eigen [61] for matrix computations, Nanoflann [62] for nearest-neighbor search, and the implementations of t-SNE and Landmark MDS from Tapkee [63]. NNP is implemented using the Keras framework [64]. The (α-)SDR-NNP code, datasets, and all results discussed in this paper are publicly available at [65].

Section "Evaluation of SDR-NNP" details the quality of SDR-NNP. Section "Evaluation of α-SDR-NNP" presents the evaluation of α-SDR-NNP as compared to SDR-NNP. Section "Computational Scalability" studies the computational scalability of both methods. Finally, "Case Study: Astronomical Datasets" presents an application of SDR-NNP to the analysis of astronomical data.

Evaluation of SDR-NNP

We first studied SDR-NNP's quality with respect to its parameters (number of iterations I, learning rate α, training epochs E) using Landmark MDS (LMDS), t-SNE, and PCA as baseline DR methods. A discussion on the selection of DR methods for SDR can be found in "Conclusion" from [7]. All results here and in "Evaluation of α-SDR-NNP" use a medium size for the NNP network. Results computed for other network sizes look very similar and are provided in the supplementary material.

Number of iterations I: Figure 3 shows how I affects the sharpening of clusters for LMDS and t-SNE (PCA results in supplementary material). For all datasets, 4 to 8 iterations suffice to have the clusters sharply defined in the projection. Table 5 shows quality metrics as functions of I for all three baseline projections. Increasing I can increase quality (Air Quality, Reuters with LMDS and PCA) but generally slightly decreases quality for LMDS and PCA. For t-SNE, this decrease is visible for all datasets, which is explainable by the fact that t-SNE already has a very high quality which is hard to be learned by NNP (see [10]). However,

Table 4 Datasets used in the (α)-SDR-NNP evaluations

Dataset name and provenance	Samples N	Dimensions n	Dataset description
Air Quality [54]	9358	13	Measurements from air sensors used to study and predict air quality
Concrete [55]	1030	8	Measurements of chemico-physical properties of concrete used to study concrete strength
Reuters [56]	5000	100	Attributes extracted from news report documents using TF-IDF [57], a standard method in text processing. This is a subset of the full dataset which contains data for the six most frequent classes only. Used to study how features can predict news' types (classes)
Spambase [58]	4001	57	Text dataset used to train email spam classifiers
Wisconsin [59]	569	32	Features extracted from images of breast masses used to detect malignant cells
Wine [60]	6497	11	Samples of white and red Portuguese *vinho verde* used to describe perceived wine quality

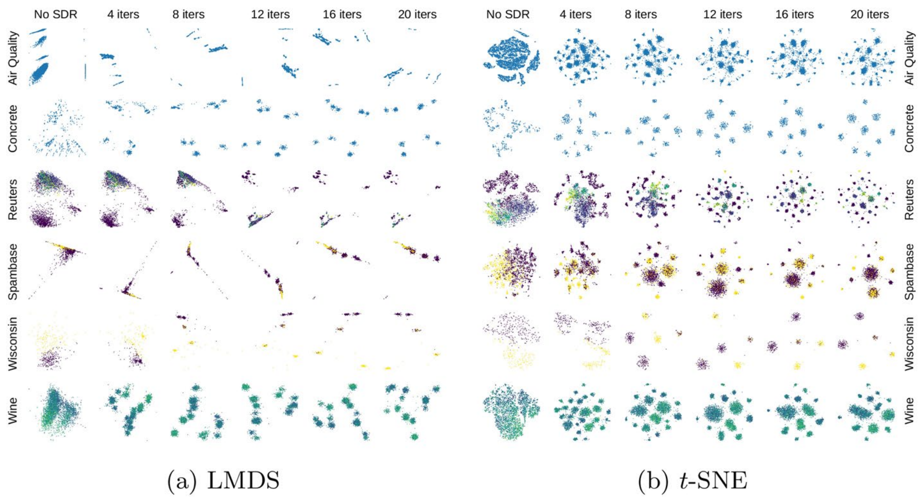


Fig. 3 Iteration parameter I effect: SDR-NNP learned from LMDS (**a**) and t-SNE (**b**) for varying I values (columns) and datasets (rows), fixed $\alpha = 0.1$, $E = 1000$ epochs

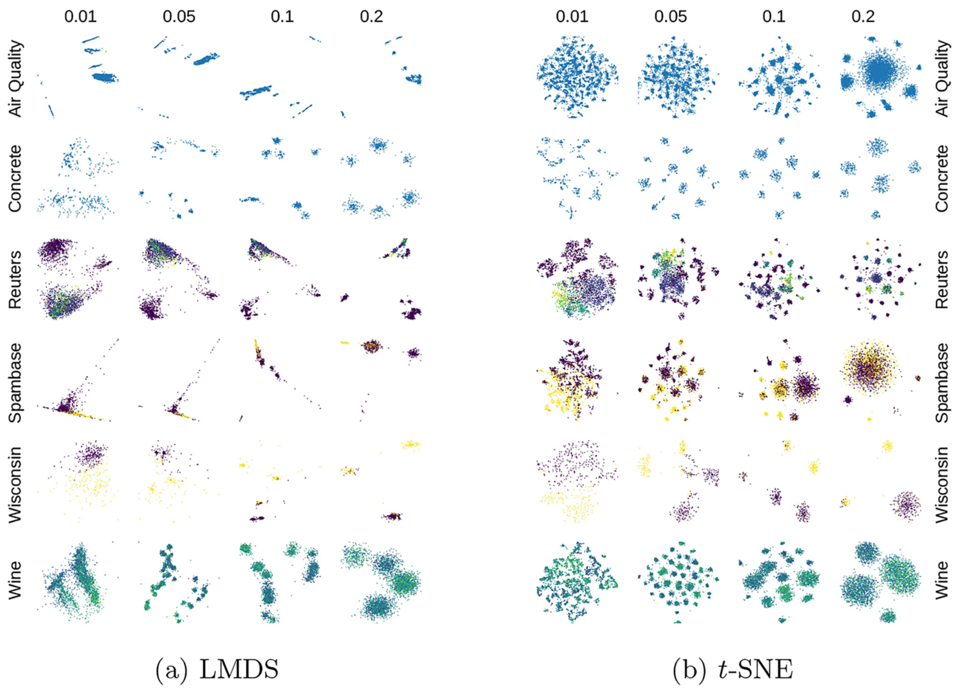


Fig. 4 Learning rate α effect: SDR-NNP learned from LMDS (**a**) and t-SNE (**b**) for varying α values (columns) and datasets (rows), fixed $I = 10$ iterations, $E = 1000$ epochs

as already argued in [7], local-quality metrics will likely decrease when using SDR to favor visual cluster separation.

Learning rate α: Figure 4 shows results for SDR-NNP when varying α for LMDS and t-SNE (PCA results in supplementary material). Too small or too large α values tend to affect the projection adversely. Values in the range $\alpha \in [0.05, 0.1]$ show the best results, i.e., a good separation of the projection into distinct clusters. Table 6 shows quality metrics as function of α for all three baseline projections. The effect of α on quality is similar with that of I with some combinations (Reuters with LMDS and PCA) showing an overall slight decrease for small α values.

Training epochs E: Figure 5 shows how E affects projection quality. The early stopping strategy used by NNP [10]—stopping training on convergence, defined as the epoch where the validation loss stops decreasing (roughly $E = 60$ in practice)—does not give good results for SDR-NNP. The resulting projections (Fig. 5a, b leftmost columns) show a fuzzy version of the training projections (Fig. 5a, b rightmost columns). This is due to the fact that SDR-NNP needs to learn *both* the LGC data sharpening and the projection P, which is more effort than learning just P, as NNP did. For more training epochs, Fig. 5 shows that SDR-NNP reproduces the training projection very faithfully. SDR-NNP

Table 5 Metrics for SDR-NNP learned from LMDS, PCA, and t-SNE, different numbers of iterations I, $\alpha = 0.1$, $E = 1000$. *NH* values miss for the Air Quality and Concrete datasets, since these are not labeled

Dataset	Iterations I	LMDS				PCA				t-SNE			
		T	C	R	NH	T	C	R	NH	T	C	R	NH
Air quality	0	0.941	0.992	0.970		0.940	0.992	0.966		0.996	0.996	0.654	
	4	0.962	0.979	0.963		0.956	0.979	0.963		0.951	0.939	0.396	
	8	0.954	0.970	0.952		0.942	0.970	0.948		0.945	0.938	0.313	
	12	0.949	0.970	0.952		0.945	0.970	0.936		0.942	0.942	0.365	
	16	0.950	0.968	0.943		0.948	0.967	0.930		0.940	0.933	0.317	
	20	0.954	0.968	0.939		0.949	0.967	0.917		0.943	0.939	0.334	
Concrete	0	0.940	0.979	0.744		0.934	0.977	0.736		0.996	0.992	0.479	
	4	0.938	0.958	0.631		0.934	0.957	0.627		0.952	0.929	0.145	
	8	0.912	0.944	0.560		0.906	0.943	0.564		0.927	0.918	0.140	
	12	0.895	0.941	0.556		0.865	0.932	0.558		0.912	0.913	0.167	
	16	0.884	0.938	0.554		0.872	0.932	0.555		0.904	0.914	0.118	
	20	0.876	0.935	0.560		0.874	0.934	0.569		0.890	0.910	0.109	
Reuters	0	0.817	0.895	0.755	0.724	0.817	0.888	0.754	0.727	0.956	0.960	0.609	0.856
	4	0.835	0.913	0.752	0.747	0.833	0.901	0.745	0.743	0.957	0.951	0.405	0.855
	8	0.858	0.915	0.713	0.775	0.854	0.906	0.706	0.765	0.950	0.910	0.258	0.845
	12	0.883	0.909	0.693	0.803	0.883	0.904	0.687	0.802	0.915	0.855	0.078	0.820
	16	0.884	0.910	0.691	0.810	0.882	0.907	0.689	0.804	0.893	0.849	0.102	0.813
	20	0.882	0.907	0.691	0.800	0.883	0.907	0.691	0.810	0.893	0.852	0.113	0.820
Spambase	0	0.740	0.909	0.529	0.852	0.747	0.912	0.513	0.849	0.954	0.958	0.408	0.914
	4	0.737	0.881	0.463	0.843	0.743	0.877	0.471	0.841	0.873	0.899	0.294	0.882
	8	0.723	0.855	0.403	0.838	0.712	0.848	0.383	0.834	0.793	0.845	0.312	0.866
	12	0.711	0.845	0.379	0.829	0.704	0.838	0.348	0.830	0.754	0.841	0.324	0.850
	16	0.701	0.837	0.370	0.828	0.701	0.837	0.322	0.830	0.744	0.834	0.332	0.845
	20	0.710	0.840	0.351	0.833	0.709	0.838	0.311	0.836	0.739	0.828	0.283	0.848
Wisconsin	0	0.895	0.959	0.926	0.941	0.896	0.959	0.928	0.943	0.950	0.939	0.679	0.976
	4	0.892	0.915	0.901	0.953	0.888	0.915	0.903	0.958	0.897	0.878	0.557	0.957
	8	0.804	0.857	0.785	0.925	0.805	0.856	0.785	0.925	0.814	0.816	0.256	0.925
	12	0.790	0.849	0.735	0.930	0.787	0.848	0.736	0.930	0.794	0.805	0.393	0.932
	16	0.780	0.847	0.721	0.916	0.780	0.844	0.718	0.922	0.779	0.820	0.432	0.913
	20	0.775	0.842	0.707	0.922	0.776	0.841	0.705	0.920	0.778	0.829	0.457	0.921
Wine	0	0.864	0.973	0.839	0.667	0.869	0.972	0.806	0.678	0.986	0.976	0.656	0.702
	4	0.867	0.932	0.709	0.669	0.864	0.930	0.686	0.665	0.911	0.894	0.341	0.673
	8	0.843	0.916	0.676	0.661	0.843	0.917	0.683	0.661	0.869	0.876	0.283	0.665
	12	0.840	0.904	0.646	0.665	0.841	0.905	0.653	0.668	0.846	0.864	0.289	0.668
	16	0.845	0.901	0.625	0.664	0.843	0.903	0.635	0.663	0.845	0.865	0.321	0.664
	20	0.842	0.899	0.579	0.659	0.846	0.898	0.593	0.664	0.845	0.868	0.292	0.666

produces good results with as little as $E = 300$ epochs, except for the Air Quality dataset, where $E = 3000$ epochs were needed for best results. On average, $E = 1000$ epochs led to good results for all datasets and other parameter settings, so we choose this as a **preset** value for E. We keep this preset also for our new method α-SDR-NNP in the latter's evaluation ("Evaluation of α-SDR-NNP").

Cluster separation: The projections in Figs. 3, 4, 5 deserve some comments. As visible there, varying the I and α parameters can create artificial *oversegmentation*—the appearance of many small clusters in the projection, which is an artificial cluster separation (CS), see, e.g., Fig. 4b, Reuters, $\alpha \geq 0.1$. This effect is strongest, and undesirable, for baseline projections which already do have a good CS, such as t-SNE. In contrast, for projections with a low CS, such as LMDS, artificial oversegmentation is far less present. Like SDR, SDR-NNP is best used when combined with baseline DR methods with a *low* CS capability. We show next in "Evaluation of α-SDR-NNP" how our new method, α-SDR-NNP, largely removes all these parameter setting issues.

Evaluation of α-SDR-NNP

Section "Evaluation of SDR-NNP" shows how SDR-NNP improves cluster separation as compared to the baseline DR projection it builds atop of. However, this evaluation also showed that, to get good results, we still have to fine tune the SDR parameters I and α. The effect of the parameter k_s was not explored. The only parameter which showed to have a good preset is $E = 1000$ training epochs.

Our new method, α-SDR-NNP, has all these parameters of SDR-NNP plus an additional one, the number of K-means clusters K_m ("-α-SDR-NNP"). Still, we argue that α-SDR-NNP is easier to use than SDR-NNP, and support this point by three evaluations on two labeled datasets (Reuters and Wine), as follows.

Visual comparison: We first run α-SDR-NNP and SDR-NNP for various combinations of their free parameters I, α, and k_s, setting K_m to the true number of clusters in the evaluated datasets (Reuters: $K_m = 6$; Wine: $K_m = 7$). For this, we take the parameter ranges found to deliver good results from SDR-NNP's evaluation ("Evaluation of SDR-NNP") and sample each range with five values, leading to the values $I \in \{1, 5, 8, 12, 20\}$ iterations, $k_s \in \{15, 30, 50, 80, 100\}$ neighbors, and $\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.4\}$ learning rates. For each combination of parameter values, we compute the projections of SDR-NNP and α-SDR-NNP for the Wine and Reuters datasets.

Visually comparing all these $5 \times 5 \times 5 \times 2 = 250$ projection-pairs is not practical. Hence, we next set each of the three parameters to its median value in its sample set and visually compare the results for the 5×5 combinations of the other two free parameters; see Figs. 6, 7, 8, 9, 10, 11. Since all these figures are structured similarly, we only explain how to interpret the first one (the others can be interpreted similarly): Figure 6 show the SDR-NNP and α-SDR-NNP projections for varying k_s and α for a fixed value of $I = 8$ iterations for the Reuters dataset. We directly see that the SDR-NNP projections (top) change considerably more than the α-SDR-NNP projections (bottom). In particular, SDR-NNP's results become increasingly fuzzy with dropping visual cluster separation for higher α values. A very similar effect is visible when varying I and α (Fig. 7) and I and k_s, respectively (Fig. 8). In contrast, α-SDR-NNP shows the same visual cluster separation for all parameter combinations. For the Wine dataset (Figs. 9, 10, 11), this fuzzy effect is less visible. Yet, the visual cluster separation of the SDR-NNP projections varies quite a lot, while α-SDR-NNP generates more stable results for the different parameter values. We conclude that, in practice, users can ignore fine-tuning k_s and α for α-SDR-NNP, and simply use the median values in their respective sample sets as default settings.

Quality metrics comparison: Table 7 compares the four projection quality metrics (Table 2) of SDR-NNP *vs* α-SDR-NNP for all the experiments shown earlier in Figs. 6, 7, 8, 9, 10, 11. Each table shows the effect of varying one of the three parameters I, k_s, and α. For such a parameter value, the shown metrics are aggregates of the 25 combinations of values of the other two parameters—for example, the first row of Table 7, column T, shows the average value of trustworthiness computed for all 25 value combinations of k_s and α and for $I = 1$. From this table, we see that α-SDR-NNP achieves similar but often higher values of quality metrics, the increase being as large as 9%. We also see that, in general, the standard deviation values are smaller for α-SDR-NNP than for SDR-NNP. This means that our new method achieves its quality metrics more consistently—or, in other words, that these values are less susceptible to change when one varies the three parameters, which is desirable. We note that an increase of several percentage points in projection quality is significant. Recent surveys [3] showed that top-quality projection methods in the entire DR literature of the last decades differ by as few as 2 to 5 percentage points. NNP, the method that we use to drive our own α-SDR-NNP technique, had a quality of a few percentage points *lower* than the state-of-the-art projection methods it tries to imitate, most notably t-SNE [10]. Moreover, as also mentioned in "Introduction", our key goal with this work was not to increase absolute *quality* of the obtained projection, but increase the *stability* and ease of computing a good-quality projection without having to fiddle with the four parameters of its predecessor method, SDR-NNP. The fact we also obtained higher quality, along with the desired ease-of-use, is an extra bonus point for our method.

Effect of K_m: The results so far show that α-SDR-NNP is stable with respect to the original three parameters I, α, and k_s of SDR-NNP. However, α-SDR-NNP introduces one new parameter, the number of clusters K_m for K-means. To study how stable our method is for this new parameter, we run it for varying values of K_m. Specifically, for the Reuters and Wine datasets, we set K_m to be half, equal to, and double the true number of clusters known to exist in these datasets. Figure 12 shows that the resulting projections are very similar in terms of visual clusters being produced. This means that our method is not sensitive to setting K_m, unlike SDR-NNP's sensitivity to setting I, α, and k_s.

Putting it all together: Figure 13 shows a so-called 'projection of projections' [3] for the Reuters and Wine dataset. Every point in such a scatterplot is a given projection technique. Green points are SDR-NNP and purple points are α-SDR-NNP. The different same-color points represent instances of the respective technique for the different values of I, k_s, and α discussed in the above evaluations. Points are projected to 2D using MDS based on the values of their four quality metrics. Points in the projection which are close indicate methods which perform similarly quality-wise. For both datasets, we see a high concentration of purple points in a tail-like structure, while the green points are far more spread around. This indicates that α-SDR-NNP generates more consistent (similar) quality values than SDR-NNP, thus, is less sensitive in this respect to parameter changes. This strengthens our claim that α-SDR-NNP allows users to generate good projections with less parameter tweaking than SDR-NNP.

Computational Scalability

We measured scalability by comparing the execution time of the original SDR method with SDR-NNP using samples from the GALAH dataset (described next in "Case Study: Astronomical Datasets") with increasing sizes, namely, 1K, 2K, 5K, 10K, 20K, 30K, and 40K samples. Using more samples was not needed, since SDR already took over 3 h at 40K samples. Figure 14 and Table 8 show these results. For $|D_s| = 10K$ training samples and $E = 1000$ epochs, SDR-NNP takes about 373 s to train (Fig. 14, orange line). Still, this is already faster than SDR for 15K samples. In inference mode (after training), SDR-NNP is orders of magnitude faster than SDR, taking less than *1 s* to project 40K samples (Fig. 14, green curve). SDR takes *over 3 h* for the same data size (Fig. 14, blue curve). For α-SDR-NNP, training time is slightly higher than for SDR-NNP due to the cost of K-means clustering and *NH* computation (Fig. 1, green steps), but inference time is identical to SDR-NNP.

Case Study: Astronomical Datasets

We used SDR-NNP for a use-case using real-world astronomical data—the same subset of 10K samples from the GALactic Archaeology with HERMES survey (GALAH DR2) [66] used by Kim et al. to show that SDR-NNP can create similar projections to SDR. The GALAH DR2 dataset consists of various stellar abundance attributes of 342,682 stars. Data cleaning followed [7]: (1) cross-match the star ID of GALAH DR2 with *Gaia* data release 2 (Gaia DR2) to gain extra information on stellar kinematics (i.e., 6D phase-space coordinates—x, y, z, u, v, and w) [66–68]; (2) exclude stars with implausible values (exceeding 25K parsec in x, y, and z attributes), having unreliable stellar abundances, or with missing values in any dimension. Pre-processing delivered 76,270 stars (samples) from which we took the same randomly selected subset D of 10K stars as in [7] to run SDR with the same $\alpha = 0.18$ ("Sharpened Dimensionality Reduction"). We trained SDR-NNP on these 10K stars and used the trained SDR-NNP network to project the remaining 66,270 stars.

Figure 15 shows SDR-NNP applied to the 66K test data with LMDS and t-SNE as baseline DR methods. Points are colored based on the value of the attribute [Fe/H], which is of interest to domain experts to explain possible data clusters. The first four columns show SDR-NNP for varying training epoch counts E. The red column shows the training projection $P(D_s)$ of 10K samples. We see that the structure of the training projection (four clusters) is well reflected by SDR-NNP from $E = 300$ epochs onwards. The test projections are more fuzzy. This is expected, as these contain 66K *unseen* samples which, albeit drawn from the same dataset, cannot perfectly match the four clusters determined by the 10K training samples. The rightmost column in Fig. 15 shows the result of the 'raw' NNP method, i.e., trained to imitate LMDS, and t-SNE, *without* the sharpening step of SDR, respectively. These results show clearly far less cluster separation (CS) than either the SDR-NNP training projection (red column) or the inferred SDR-NNP projections (leftmost four columns). This shows the added value of the *sharpening* step: Without it, NNP, albeit fast and OOS-capable, cannot produce useful projections. Table 9 shows quality metrics corresponding to the images in Fig. 15 which support the above observations.

SDR-NNP's good cluster separation allows astronomers to easily label clusters for further analysis to infer the physical meaning of stars. To show this, we manually labeled clusters from SDR-NNP learned from LMDS to reproduce the same analyses made by Kim et al. (Fig. 10 in [7]) to understand the origin and location of stars in each cluster. Figure 16a shows the manually labeled clusters by one of the authors (astronomy expert). Stars from class 5 are separately labeled as outliers. Figure 16b,c shows the Tinsley diagram [69] and the copper abundance of the stars—a tracer of supernovae type 1a—as a function of their iron abundance, respectively. From these plots, astronomers can identify class-1 stars as thin-disk stars, class-2 stars as metal-rich thick disk stars, class-3 and class-5 (outlier) stars as the normal thick disk stars, and class-4 stars as Gaia Enceladus (GES)—a group of stars that originated from a galaxy that merged with the Milky Way several billions years ago. The original SDR method could not do this analysis and find class-4 stars, since it could run on the entire dataset due to its prohibitively low speed.

Discussion

We discuss how SDR-NNP and our new extension, α-SDR-NNP, perform with respect to the criteria laid out in "Introduction".

Quality (C1): SDR-NNP can create projections which are very similar visually, but also in terms of quality metrics, to those created by SDR. Importantly, the strong separation of similar-valued samples, the key property that SDR promoted, is retained by SDR-NNP. Combined with properties C2–C4 (which SDR does not have), this makes SDR-NNP superior to SDR. Compared to NNP used on the unsharpened data (Fig. 15), SDR-NNP shows significantly better cluster separation, which makes it superior to NNP. Atop all these, α-SDR-NNP shows, for most parameter values, higher quality metrics—thus better projections—than SDR-NNP.

Scalability (C2): (α-)SDR-NNP is faster than SDR alone from roughly 15K samples onwards, even when considering training time. In inference mode (after training), (α-)SDR-NNP is several orders of magnitude faster than SDR, being able to project tens of thousands of observations in under a second on a high-end PC. Importantly, (α-)SDR-NNP's speed is *linear* in the number of dimensions and samples (a property inherited from the NNP architecture), and can handle samples in a *streaming* fashion, one at a time, *i.e.*, does not need to hold the entire high-dimensional dataset in memory. This makes (α-)SDR-NNP scalable to large datasets of millions of samples. Compared to SDR-NNP, our new method α-SDR-NNP is slightly slower for training but has identical inference speed.

We evaluated our method on relatively small datasets—up to 10K samples (see Table 4). Of course, larger high-dimensional datasets exist. However, we preferred to use these datasets, since they are well known in the visualization and machine learning communities, as part of multiple benchmarks and many papers. As such, readers can directly compare our projection results for these datasets with other results in the literature. Moreover, we note that using larger datasets will only *help* our method. Computation speed of our projections is linear in the sample count, as stated above, which is the optimal result one can get. Using larger datasets implies having higher sample densities, so estimating this density only increases in accuracy (Eqn. 2). As such, using smaller datasets is actually a bigger challenge for our method.

Genericity (C3): (α-)SDR-NNP can project any dataset having quantitative variables and any dimension count. Tables 5, 6, 9, and 7 show that (α-)SDR-NNP achieves high quality on datasets of different nature and coming from a wide range of application domains (air sensors, civil engineering, text mining, imaging, and chemistry).

Stability and out-of-sample support (C4): (α-)SDR-NNP inherits the stability and OOS support of NNP, making it possible to train on a small subset of a given dataset and then stably project additional data drawn from the same distribution. Moreover, our new method, α-SDR-NNP, is stable with respect to its *single* free parameter K_m, which makes it by construction a good method to sharpen-and-project high-dimensional data. Simply put, α-SDR-NNP delivers *similar* visual results for a given dataset and any settings of K_m, which means in practice that users can benefit from a sharpened projection without worrying about how they set the parameter controlling the computation of this projection.

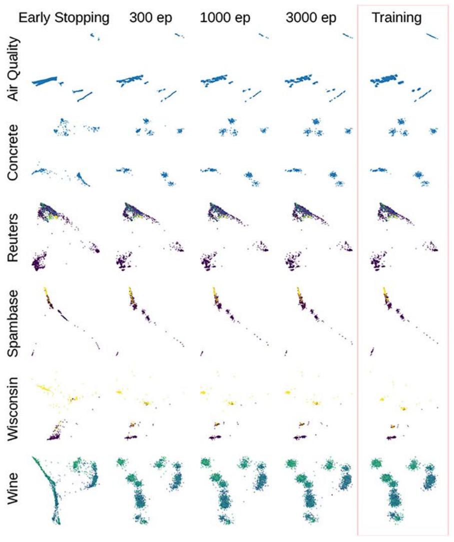
Ease of use (C5): Once trained, (α-)SDR-NNP is parameter-free. SDR-NNP has three relevant parameters affecting its pre-processing LGC step—number of sharpening iterations I, learning rate α, neighbors k_s used in estimating the data density; and one affecting training—number of training epochs E. We showed that $E = 1000$ is a good preset for SDR-NNP, thus also for α-SDR-NNP. The other three parameters affect the resulting learned projection in several ways, such as changing the visual cluster separation (undersegmentation, oversegmentation). Hence, SDR-NNP is not easy-to-use, as it requires some amount of parameter tuning experimentation. Moreover, every parameter change implies *retraining* which takes minutes (see "Computational Scalability").

α-SDR-NNP largely solves this problem by adapting the learning rate to the local cluster structure of the data, which is estimated by K-means. This requires introducing a new parameter, the number K_m of K-means clusters used during training ("-α-SDR-NNP"). We showed that α-SDR-NNP is far less sensitive to changes of I, α, and k_s, so these parameters can be simply set to default values; and is also insensitive to setting K_m, as shown in "Evaluation of α-SDR-NNP". Intuitively put, our new method allows the user to control the desired final projection outcome at a higher and more *global* level (that is, in terms of expected clusters K_m) than the *local* controls that SDR-NNP required in terms of number of iterations I, learning rate α, and density-estimation bandwidth k_s. This is especially important, since such local parameters can vary a lot over a given dataset, so there is no way to determine good global defaults for them for the entire dataset. In contrast, the expected number of clusters K_m is a much higher level parameter which does not depend strongly on the local data structure, so, for a given dataset, is a much easier-to-control setting. Given the above, our new method can be seen as virtually parameter-free, thus easy and effective to use.

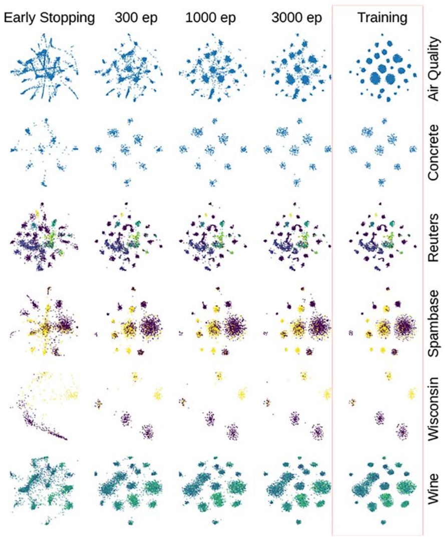
Limitations: While inheriting the above-mentioned desirable properties from NNP, (α-)SDR-NNP also inherits some of its limitations. Its OOS support cannot extend to datasets of a completely different nature than those it was trained on—arguably, a limitation that most machine learning methods have. Also, (α-)SDR-NNP is only as good as the baseline projection P that was used during training. Using a

Table 6 Metrics for SDR-NNP learned from LMDS, PCA, and t-SNE, learning rates α, $I = 10$ iterations, $E = 1000$ epochs. NH values miss for the Air Quality and Concrete datasets, since these are not labeled

Dataset	Learning rate α	LMDS				PCA				t-SNE			
		T	C	R	NH	T	C	R	NH	T	C	R	NH
Air Quality	0.01	0.971	0.990	0.969		0.968	0.990	0.964		0.958	0.926	0.052	
	0.05	0.976	0.983	0.963		0.973	0.984	0.964		0.938	0.911	0.175	
	0.1	0.951	0.969	0.948		0.940	0.969	0.941		0.943	0.939	0.378	
	0.2	0.866	0.941	0.911		0.862	0.928	0.905		0.824	0.876	0.635	
Concrete	0.01	0.959	0.983	0.731		0.950	0.979	0.721		0.994	0.988	0.486	
	0.05	0.932	0.957	0.601		0.929	0.953	0.601		0.947	0.929	0.219	
	0.1	0.870	0.933	0.578		0.889	0.935	0.583		0.913	0.918	0.057	
	0.2	0.858	0.920	0.540		0.859	0.921	0.542		0.857	0.900	0.223	
Reuters	0.01	0.822	0.900	0.758	0.729	0.821	0.892	0.755	0.730	0.956	0.960	0.608	0.853
	0.05	0.839	0.913	0.737	0.745	0.838	0.903	0.735	0.745	0.955	0.949	0.386	0.849
	0.1	0.870	0.910	0.698	0.783	0.871	0.905	0.693	0.790	0.936	0.885	0.159	0.829
	0.2	0.866	0.902	0.700	0.784	0.867	0.903	0.693	0.785	0.890	0.848	0.096	0.815
Spambase	0.01	0.755	0.911	0.527	0.860	0.756	0.915	0.523	0.852	0.958	0.942	0.383	0.905
	0.05	0.775	0.893	0.415	0.840	0.787	0.894	0.426	0.858	0.851	0.874	0.261	0.874
	0.1	0.712	0.843	0.380	0.832	0.704	0.839	0.367	0.828	0.769	0.848	0.341	0.863
	0.2	0.604	0.667	0.265	0.760	0.606	0.671	0.263	0.764	0.635	0.676	0.317	0.802
Wisconsin	0.01	0.900	0.960	0.932	0.947	0.898	0.960	0.932	0.949	0.955	0.941	0.635	0.966
	0.05	0.868	0.885	0.869	0.946	0.874	0.890	0.870	0.941	0.876	0.861	0.597	0.948
	0.1	0.803	0.856	0.757	0.928	0.800	0.851	0.753	0.929	0.802	0.841	0.494	0.927
	0.2	0.717	0.764	0.693	0.905	0.718	0.758	0.684	0.918	0.725	0.749	0.596	0.909
Wine	0.01	0.895	0.972	0.811	0.674	0.898	0.971	0.783	0.681	0.983	0.951	0.448	0.696
	0.05	0.914	0.944	0.734	0.671	0.920	0.945	0.714	0.670	0.927	0.862	0.135	0.670
	0.1	0.837	0.913	0.672	0.658	0.848	0.913	0.672	0.664	0.864	0.882	0.250	0.661
	0.2	0.739	0.821	0.479	0.653	0.742	0.825	0.484	0.644	0.744	0.808	0.404	0.646



(a) LMDS



(b) t-SNE

Fig. 5 Training epochs E effect: SDR-NNP learned from LMDS (**a**) and t-SNE (**b**) for varying E values (columns) and datasets (rows), fixed $I = 10$, $\alpha = 0.1$. Red column shows the training projections $P(D_s)$

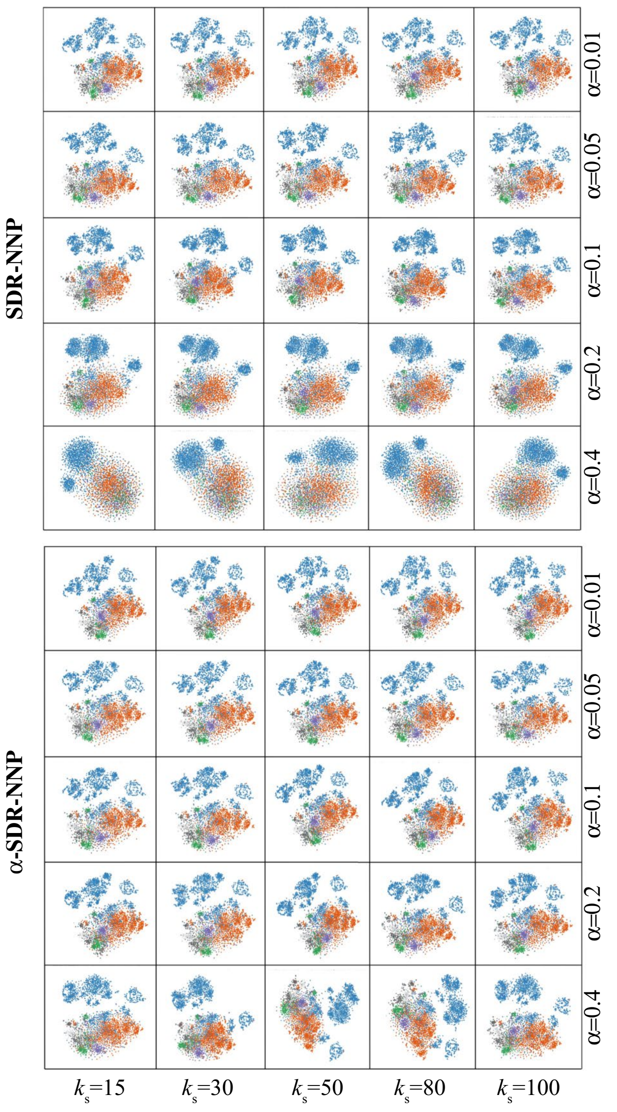


Fig. 6 SDR-NNP (top) *vs* α-SDR-NNP (bottom), Reuters dataset. Fixed $I = 8$ iterations, varying number of neighbors k_s and learning rate α

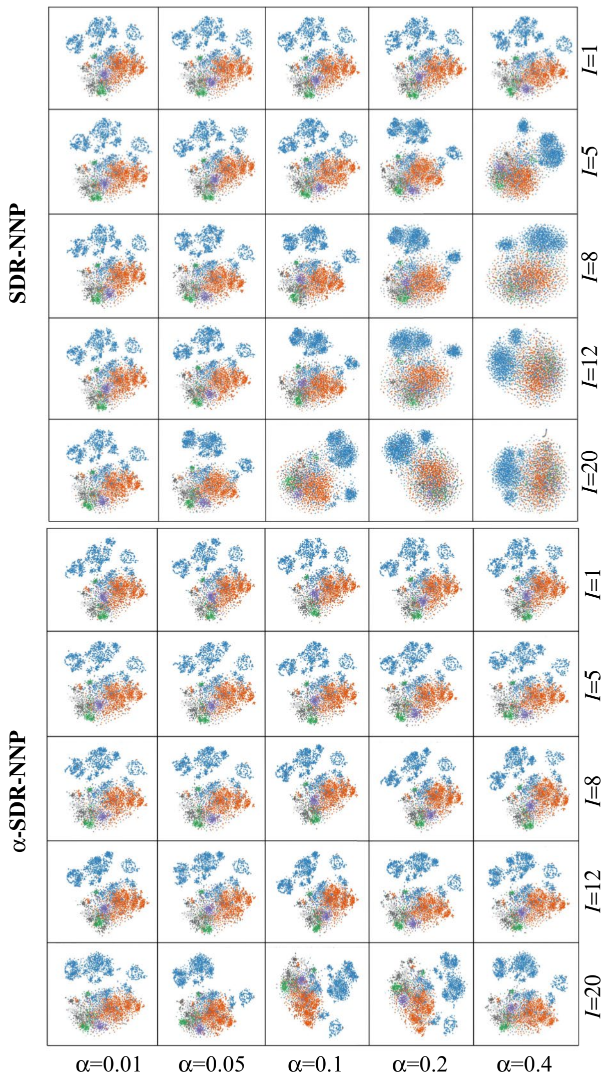


Fig. 7 SDR-NNP (top) *vs* α-SDR-NNP (bottom), Reuters dataset. Fixed $k_s = 50$ neighbors, varying number of iterations I and learning rate α

low-quality projection leads to (α-)SDR-NNP learning, and reproducing, that behavior.

Separately, SDR-NNP is prone to instability in the generated projection, which manifests itself as over- or under-segmentation of the data into too many, respectively too few, visual clusters, as a function of its parameter values. As discussed above, our new method α-SDR-NNP largely removes this problem. Still, α-SDR-NNP has its own limitations. Its only parameter K_m, the number of K-means clusters ("-α-SDR-NNP"), can be set quite freely to values differing as much as twice from the true number of clusters in the data (see Fig. 12 and related text). Still, there can be datasets for which the user has no idea, even within this error margin, to what a good K_m setting is. Exploring how α-SDR-NNP behaves in those cases and, if necessary, refining it to be even less sensitive on the K_m setting, is for future work.

Applications: As our method is generic (C3), it can handle high-dimensional datasets coming from any application domain. A particular application domain where such

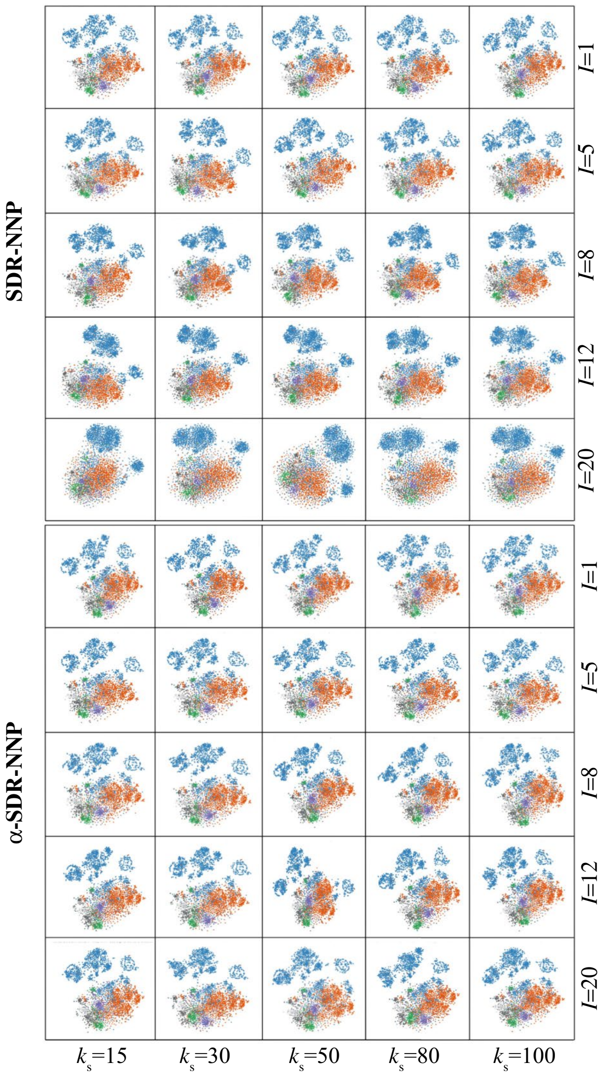


Fig. 8 SDR-NNP (top) *vs* α-SDR-NNP (bottom), Reuters dataset. Fixed $\alpha = 0.1$ learning rate, varying number of iterations I and neighbors k_s

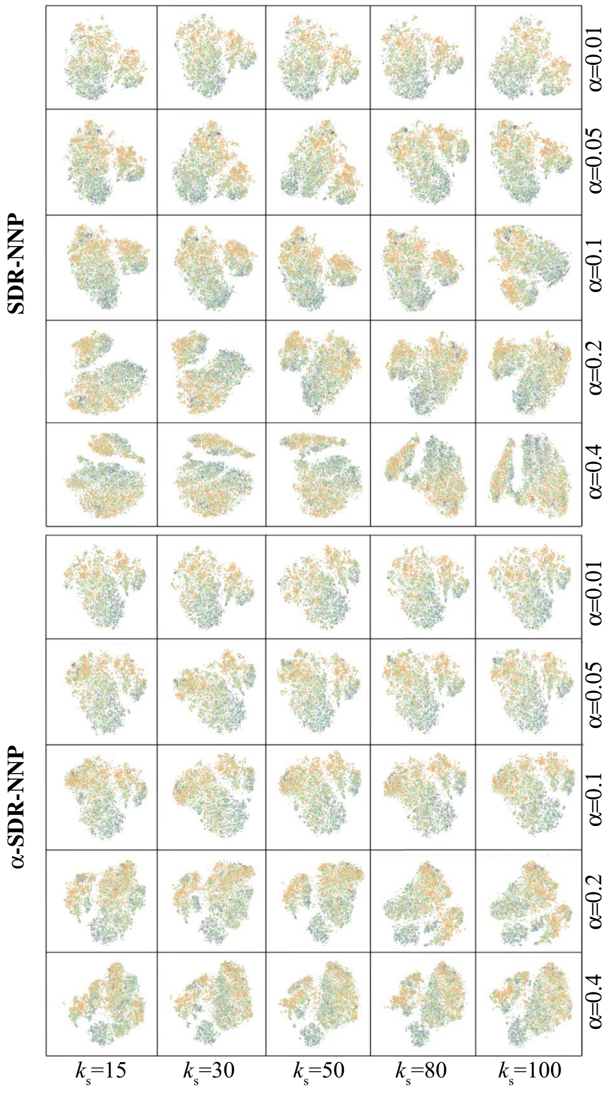


Fig. 9 SDR-NNP (top) *vs* α-SDR-NNP (bottom), Wine dataset. Fixed $I = 8$ iterations, varying number of neighbors k_s and learning rate α

projections are very useful is in engineering classification models. More particularly, engineering classifiers for *image* data is an attractive application area of α-SDR-NNP, since the method can be easily enhanced to display the actual images corresponding to the projected points. Thereby, users can examine a projection, e.g., labeled by ground-truth or inferred information, and get insights in why and where misclassifications occur. Such scenarios involving using projections have been presented in recent research on medical image classification [70–72] and cell imaging [72, 73]. α-SDR-NNP is especially attractive for such use-cases, since these provide a known number of clusters K_m in the data to be detected, equal to the number of classes to be inferred. As such, setting the single free parameter K_m of our method is simple. We are considering exploring how our method can address such use-cases in future work.

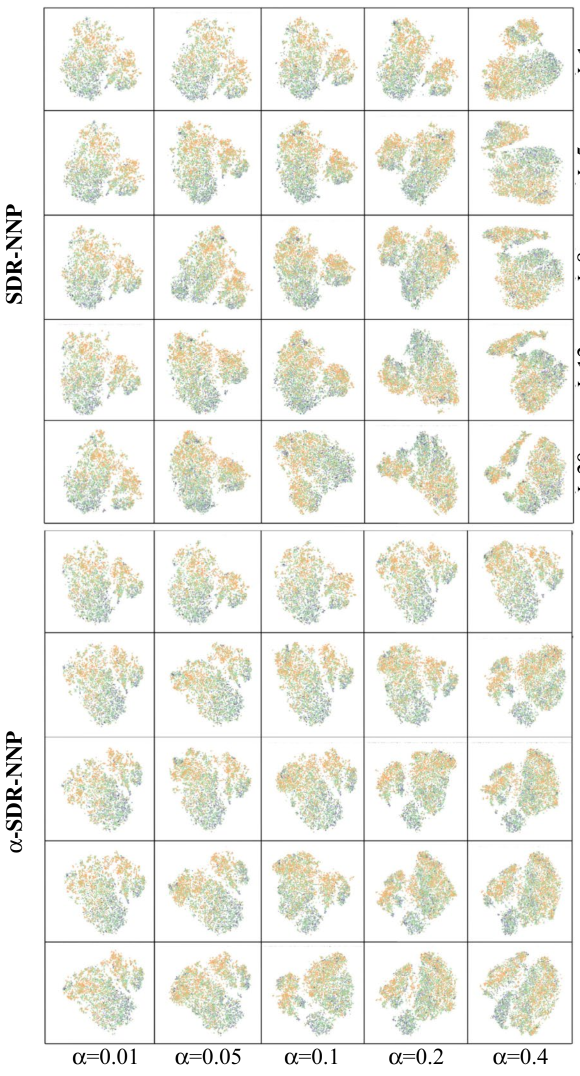


Fig. 10 SDR-NNP (top) *vs* α-SDR-NNP (bottom), Wine dataset. Fixed $k_s = 50$ neighbors, varying number of iterations I and learning rate α

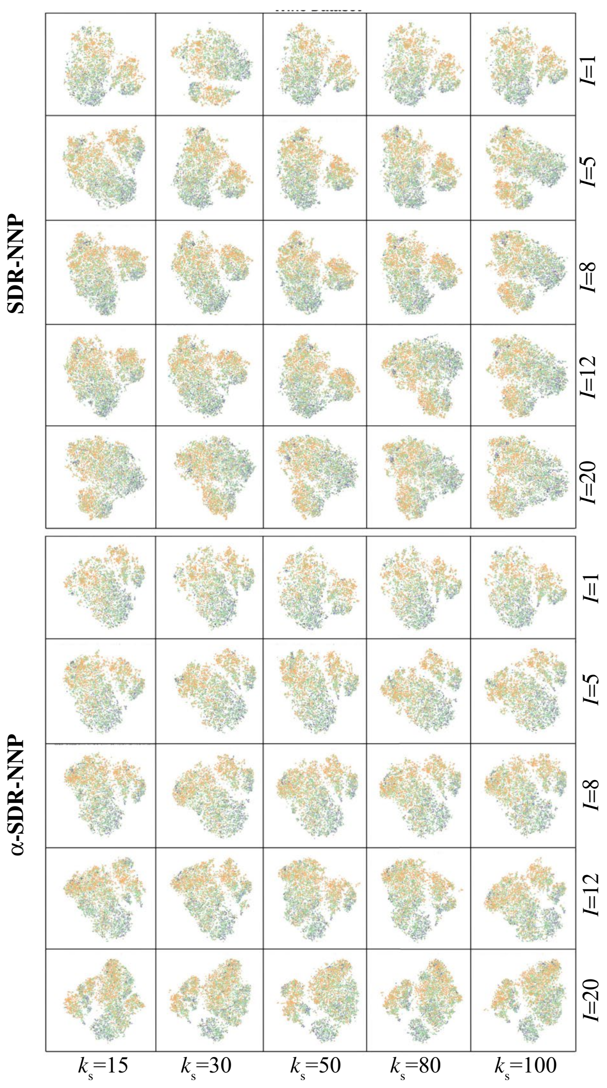


Fig. 11 SDR-NNP (top) *vs* α-SDR-NNP (bottom), Wine dataset. Fixed $\alpha = 0.1$ learning rate, varying number of iterations I and neighbors k_s

Conclusion

We have presented SDR-NNP and α-SDR-NNP, two new methods for computing projections of high-dimensional datasets for visual exploration. Our methods have several desirable and complementary characteristics of two earlier projection methods, namely NNP (speed, out-of-sample support, ability to accurately imitate a wide range of existing projection techniques) and SDR (projecting complex datasets into visually well-separated clusters of similar samples). SDR-NNP removes the main obstacle for practical usage of SDR—its high computational time. α-SDR-NNP further enhances SDR-NNP's ease of use by removing the latter method's sensitivity to parameter setting—α-SDR-NNP is essentially a parameter-free method. α-SDR-NNP also increases the quality metrics of its resulting projections as compared to SDR-NNP. We have demonstrated both methods on a range of datasets coming from different application domains. In particular, we showed how SDR-NNP can bring

Table 7 Quality metrics comparing SDR-NNP with α-SDR-NNP

Dataset	Iterations I	SDR-NNP				α-SDR-NNP			
		T	C	R	NH	T	C	R	NH
(a) Number of iterations I (average)									
Wine	1	0.981	0.959	0.676	0.686	0.992	0.975	0.678	0.711
Wine	5	0.959	0.925	0.638	0.656	0.978	0.936	0.648	0.680
Wine	8	0.952	0.911	0.624	0.648	0.967	0.911	0.614	0.667
Wine	12	0.944	0.904	0.623	0.643	0.957	0.886	0.589	0.658
Wine	20	0.936	0.888	0.597	0.639	0.940	0.857	0.536	0.648
Reuters	1	0.967	0.963	0.614	0.855	0.967	0.963	0.619	0.856
Reuters	5	0.946	0.941	0.637	0.839	0.966	0.962	0.622	0.853
Reuters	8	0.927	0.921	0.637	0.823	0.964	0.960	0.625	0.851
Reuters	12	0.908	0.898	0.631	0.808	0.960	0.955	0.626	0.848
Reuters	20	0.881	0.864	0.617	0.788	0.952	0.943	0.635	0.841
(b) Number of iterations I (standard deviation)									
Wine	1	0.021	0.030	0.018	0.040	0.003	0.007	0.015	0.014
Wine	5	0.035	0.051	0.071	0.039	0.019	0.052	0.057	0.031
Wine	8	0.037	0.055	0.086	0.037	0.027	0.069	0.079	0.034
Wine	12	0.038	0.054	0.072	0.035	0.035	0.079	0.103	0.036
Wine	20	0.039	0.060	0.098	0.029	0.043	0.083	0.118	0.036
Reuters	1	0.001	0.001	0.008	0.002	0.000	0.000	0.007	0.002
Reuters	5	0.034	0.036	0.027	0.026	0.002	0.002	0.015	0.003
Reuters	8	0.055	0.060	0.028	0.044	0.005	0.005	0.013	0.006
Reuters	12	0.066	0.075	0.027	0.054	0.010	0.012	0.017	0.009
Reuters	20	0.076	0.096	0.054	0.059	0.020	0.029	0.020	0.017
(c) Number of neighbors k_s (average)									
Wine	15	0.948	0.912	0.623	0.647	0.962	0.901	0.593	0.672
Wine	30	0.954	0.917	0.629	0.653	0.965	0.909	0.607	0.673
Wine	50	0.956	0.919	0.635	0.656	0.968	0.915	0.618	0.673
Wine	80	0.957	0.920	0.638	0.658	0.969	0.919	0.624	0.673
Wine	100	0.957	0.919	0.634	0.658	0.970	0.922	0.622	0.673
Reuters	15	0.923	0.913	0.623	0.820	0.962	0.956	0.626	0.850
Reuters	30	0.925	0.917	0.629	0.822	0.962	0.957	0.626	0.849
Reuters	50	0.926	0.917	0.625	0.823	0.962	0.956	0.623	0.850
Reuters	80	0.927	0.919	0.629	0.823	0.962	0.957	0.624	0.850
Reuters	100	0.928	0.922	0.630	0.824	0.962	0.957	0.628	0.850
(d) Number of neighbors k_s (standard deviation)									
Wine	15	0.044	0.060	0.091	0.043	0.040	0.086	0.122	0.038
Wine	30	0.038	0.057	0.079	0.040	0.036	0.080	0.098	0.038
Wine	50	0.036	0.056	0.077	0.040	0.032	0.074	0.087	0.039
Wine	80	0.035	0.053	0.071	0.038	0.030	0.071	0.086	0.039
Wine	100	0.035	0.055	0.073	0.038	0.030	0.069	0.080	0.039
Reuters	15	0.065	0.079	0.041	0.052	0.012	0.017	0.015	0.010
Reuters	30	0.062	0.072	0.035	0.049	0.011	0.016	0.016	0.010
Reuters	50	0.061	0.074	0.037	0.048	0.012	0.018	0.017	0.011
Reuters	80	0.060	0.069	0.025	0.048	0.011	0.015	0.015	0.011
Reuters	100	0.058	0.064	0.029	0.047	0.011	0.014	0.017	0.011

Table 7 (continued)

Dataset	LR	SDR-NNP				α-SDR-NNP			
		T	C	R	NH	T	C	R	NH
(e) Learning rate α (average)									
Wine	1	0.981	0.959	0.676	0.686	0.992	0.975	0.678	0.711
Wine	5	0.959	0.925	0.638	0.656	0.978	0.936	0.648	0.680
Wine	8	0.952	0.911	0.624	0.648	0.967	0.911	0.614	0.667
Wine	12	0.944	0.904	0.623	0.643	0.957	0.886	0.589	0.658
Wine	20	0.936	0.888	0.597	0.639	0.940	0.857	0.536	0.648
Reuters	1	0.967	0.963	0.614	0.855	0.967	0.963	0.619	0.856
Reuters	5	0.946	0.941	0.637	0.839	0.966	0.962	0.622	0.853
Reuters	8	0.927	0.921	0.637	0.823	0.964	0.960	0.625	0.851
Reuters	12	0.908	0.898	0.631	0.808	0.960	0.955	0.626	0.848
Reuters	20	0.881	0.864	0.617	0.788	0.952	0.943	0.635	0.841
(f) Learning rate α (standard deviation)									
Wine	0.01	0.001	0.004	0.012	0.010	0.000	0.001	0.013	0.004
Wine	0.05	0.011	0.019	0.009	0.031	0.008	0.024	0.021	0.023
Wine	0.1	0.020	0.029	0.018	0.031	0.017	0.051	0.064	0.028
Wine	0.2	0.026	0.038	0.058	0.018	0.029	0.070	0.092	0.031
Wine	0.4	0.023	0.036	0.078	0.008	0.039	0.075	0.115	0.028
Reuters	0.01	0.001	0.000	0.009	0.002	0.001	0.000	0.007	0.001
Reuters	0.05	0.006	0.005	0.017	0.004	0.001	0.001	0.008	0.002
Reuters	0.1	0.027	0.027	0.024	0.021	0.002	0.002	0.018	0.003
Reuters	0.2	0.054	0.062	0.026	0.043	0.007	0.008	0.018	0.007
Reuters	0.4	0.070	0.087	0.055	0.056	0.019	0.027	0.018	0.015

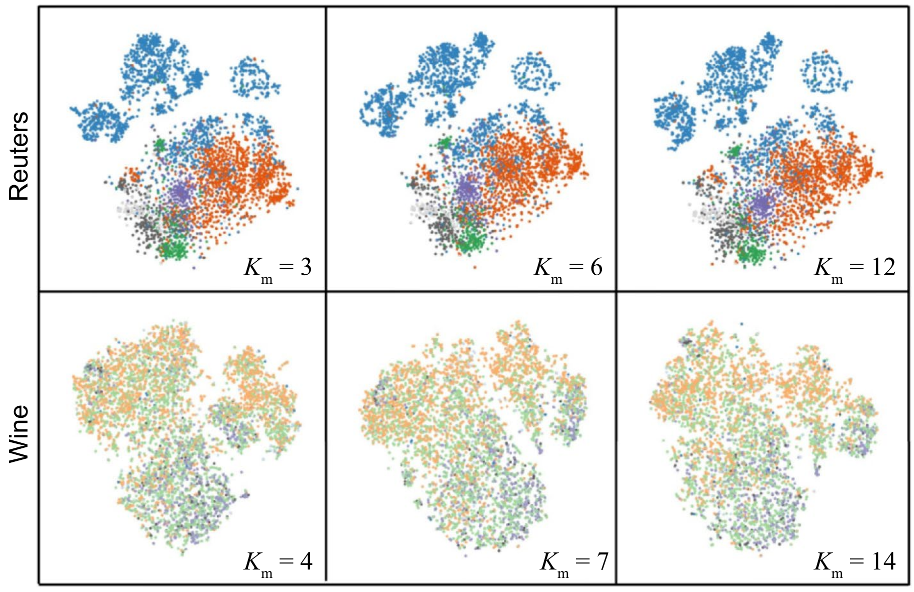


Fig. 12 α-SDR-NNP for different numbers of K-means clusters K_m. Columns show results when setting K_m to half, equal to, and double the true number of clusters (6 for Reuters and 7 for Wine)

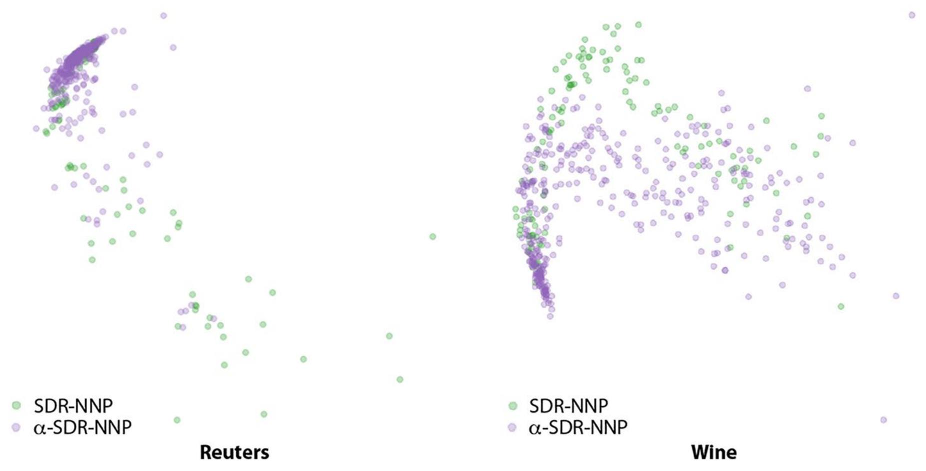


Fig. 13 MDS projection of projections computed for all experiments colored by method (SDR-NNP: green; α-SDR-NNP: purple), Reuters and Wine datasets

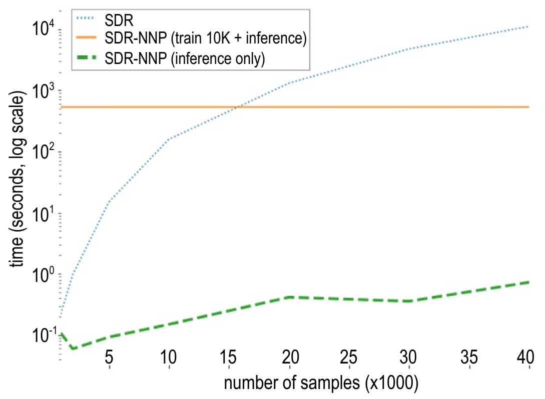


Fig. 14 Performance of SDR *vs* SDR-NNP on the GALAH dataset (time in log scale), 1K to 40K samples. SDR-NNP trained with 10K samples for $E = 1000$ epochs. See also Table 14

Table 8 Time measurements for SDR and (α-)SDR-NNP in seconds, GALAH dataset. See also Fig. 14

Samples	SDR	(α-)SDR-NNP inference
1000	0.220	0.108
2000	0.957	0.059
5000	14.799	0.092
10000	157.420	0.149
20000	1302.268	0.414
30000	4736.995	0.355
40000	11058.267	0.727

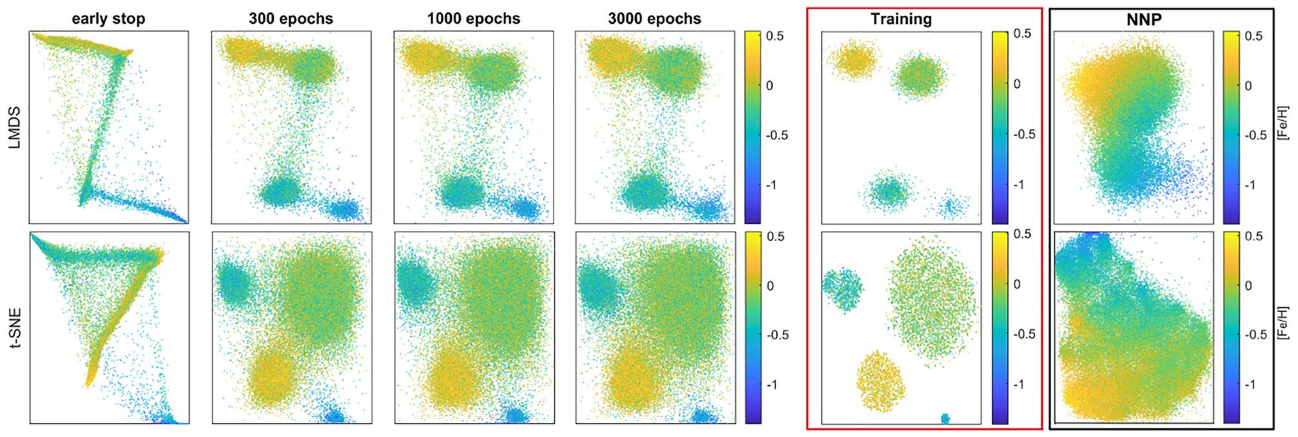


Fig. 15 SDR-NNP of 66K samples learned from LMDS (top) and t-SNE (bottom) for different numbers of training epochs E (four leftmost columns). SDR-NNP parameters are $I = 10$ iterations, and $\alpha = 0.18$. Red column: training projection (10K samples). Rightmost column: NNP trained with LMDS and t-SNE instead of SDR applied to the same test data

Table 9 Metrics for SDR-NNP learned from LMDS, PCA, and t-SNE on the GALAH dataset for train and test samples, varying number of training epochs *E* ('early' indicates the early stopping heuristic)

Mode	E	LMDS			PCA			t-SNE		
		T	C	R	T	C	R	T	C	R
Train	Early	0.802	0.877	0.676	0.787	0.853	0.668	0.789	0.870	0.638
	300	0.703	0.790	0.629	0.697	0.778	0.617	0.694	0.770	0.496
	1000	0.695	0.772	0.615	0.693	0.762	0.607	0.692	0.748	0.481
	3000	0.692	0.756	0.600	0.693	0.756	0.601	0.691	0.734	0.447
Test	Early	0.775	0.862	0.601	0.754	0.828	0.592	0.756	0.853	0.574
	300	0.667	0.768	0.558	0.660	0.759	0.544	0.652	0.755	0.470
	1000	0.661	0.749	0.547	0.658	0.746	0.539	0.642	0.734	0.449
	3000	0.657	0.737	0.534	0.655	0.739	0.539	0.641	0.720	0.416

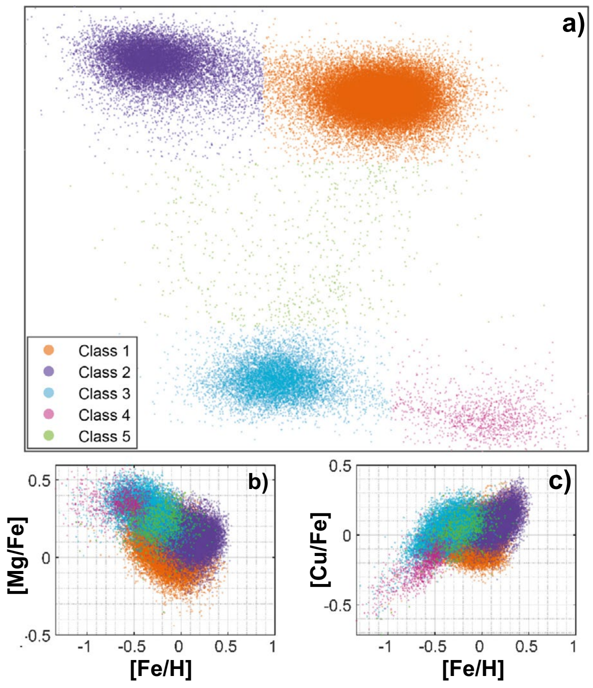


Fig. 16 Analysis of GALAH DR2 with SDR-NNP learned from LMDS. (**a**) Labeling of clusters (classes 1–4) and outliers (class 5). (**b**) Tinsley diagram and (**c**) copper abundance of stars *vs* their iron abundance. Astronomers can infer from (**b**, **c**) that class 1 is mostly thin-disk stars, class 2 is mostly metal-rich thick disk stars, classes 5 and 3 are normal thick disk stars, and class 4 is the Gaia Enceladus (GES) in the Milky Way

added value in the exploration of a large and recent astronomical dataset leading to findings which were not achievable by SDR or NNP alone.

Future work can target several directions. Our work showed that it is possible to learn sharpening methods for high-dimensional data, so it is interesting to apply our techniques to other domains data sharpening is used, e.g., image segmentation, graph bundling, and data clustering and simplification. For the projection use-case, refining α-SDR-NNP's network architecture to speed its training is of high practical interest. Finally, deploying α-SDR-NNP as a main tool for astronomers to analyze their million-sample datasets is a goal we aim to pursue in the short term.

Author Contributions Not applicable.

Funding This work is supported by the DSSC Doctoral Training Programme co-funded by the Marie Sklodowska-Curie COFUND project (DSSC 754315), and FAPESP under Grant 2020/13275-1, Brazil. The GALAH survey is based on observations made at the Australian Astronomical Observatory, under programmes A/2013B/13, A/2014A/25, A/2015A/19, A/2017A/18. We acknowledge the traditional owners of the land on which the AAT stands, the Gamilaraay people, and pay our respects to elders past and present. This work has made use of data from the European Space Agency (ESA) mission *Gaia* (https://www.cosmos.esa.int/gaia), processed by the *Gaia* Data Processing and Analysis Consortium (DPAC, https://www.cosmos.esa.int/web/gaia/dpac/consortium). Funding for the DPAC has been provided by national institutions, in particular the institutions participating in the *Gaia* Multilateral Agreement.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Availability of data and materials Not applicable.

Code availability Our implementation, and all codes used in our experiments, are publicly available at https://github.com/youngjookim/sdr.

References

1. Liu S, Maljovec D, Wang B, Bremer P-T, Pascucci V. Visualizing high-dimensional data: advances in the past decade. IEEE TVCG. 2015;23(3):1249–68.
2. Nonato L, Aupetit M. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout

enrichment. IEEE TVCG. 2018. https://doi.org/10.1109/TVCG.2018.2846735.
3. Espadoto M, Martins R, Kerren A, Hirata N, Telea A. Toward a quantitative survey of dimension reduction techniques. IEEE TVCG. 2019;27(3):2153–73.
4. Maaten L, Hinton G. Visualizing data using t-SNE. JMLR. 2008;9:2579–605.
5. McInnes L, Healy J. UMAP: uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426v1 [stat.ML] 2018.
6. Behrisch M, Blumenschein M, Kim NW, Shao L, El-Assady M, Fuchs J, Seebacher D, Diehl A, Brandes U, Pfister H, Schreck T, Weiskopf D, Keim DA. Quality metrics for information visualization. Comp Graph Forum. 2018;37(3):625–62.
7. Kim Y, Telea A, Trager S, Roerdink JBTM. Visual cluster separation using high-dimensional sharpened dimensionality reduction. Inf Vis. 2022;21(3):197–219.
8. Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis. IEEE TPAMI. 2002;24(5):603–19.
9. Kim Y, Espadoto M, Trager S, Roerdink J, Telea A. SDR-NNP: Sharpened dimensionality reduction with neural networks. In: Proc. IVAPP 2022. SciTePress
10. Espadoto M, Hirata N, Telea A. Deep learning multidimensional projections. Inform Visual. 2020;9(3):247–69.
11. Hoffman P, Grinstein G. A survey of visualizations for high-dimensional data mining. Inform Vis Data Mining Knowl Discov. 2002;2:47–82.
12. Maaten L, Postma E. Dimensionality reduction: a comparative review. Technical report, Tilburg Univ. 2009
13. Engel D, Hattenberger L, Hamann B. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In: Proc. IRTG Workshop, 2012;vol. 27, pp. 135–149. Schloss Dagstuhl
14. Sorzano C, Vargas J, Pascual-Montano A. A survey of dimensionality reduction techniques. arXiv:1403.2877 [stat.ML] 2014.
15. Cunningham J, Ghahramani Z. Linear dimensionality reduction: survey, insights, and generalizations. JMLR. 2015;16:2859–900.
16. Xie H, Li J, Xue H. A survey of dimensionality reduction techniques based on random projection. arXiv:1706.04371 [cs.LG] 2017
17. Venna J, Kaski S. Visualizing gene interaction graphs with local multidimensional scaling. In: Proc. ESANN, 2006;pp. 557–562.
18. Paulovich FV, Nonato LG, Minghim R, Levkowitz H. Least square projection: a fast high-precision multidimensional projection technique and its application to document mapping. IEEE TVCG. 2008;14(3):564–75.
19. Rauber PE, Falcão AX, Telea AC. Projections as visual aids for classification system design. Inform Visual. 2017;17(4):282–305.
20. Joia P, Coimbra D, Cuminato JA, Paulovich FV, Nonato LG. Local affine multidimensional projection. IEEE TVCG. 2011;17(12):2563–71.
21. LeCun Y, Cortes C. MNIST Handwritten Digits Dataset. http://yann.lecun.com/exdb/mnist 2010.
22. Jolliffe IT. Principal component analysis and factor analysis. In: Principal component analysis, 1986;pp. 115–128. Springer, Berlin
23. Torgerson W. Theory and methods of scaling. Boca Raton: Wiley; 1958.
24. De Silva V, Tenenbaum JB. Sparse multidimensional scaling using landmark points. Technical report, Stanford University 2004.
25. Tenenbaum JB, Silva VD, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000;290(5500):2319–23.
26. Roweis ST, Saul LLK. Nonlinear dimensionality reduction by locally linear embedding. Science. 2000;290(5500):2323–6.
27. Donoho DL, Grimes C. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. PNAS. 2003;100(10):5591–6.
28. Zhang Z, Zha H. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM J Sci Comput. 2004;26(1):313–38.
29. Zhang Z, Wang J. MLLE: Modified locally linear embedding using multiple weights. In: Proc. NIPS, 2007;pp. 1593–1600.
30. Paulovich FV, Minghim R. Text map explorer: a tool to create and explore document maps. In: Proc. IEEE Information Visualisation, 2006;pp. 245–251.
31. Wattenberg M. How to use t-SNE effectively. https://distill.pub/2016/misread-tsne 2016.
32. Maaten L. Accelerating t-SNE using tree-based algorithms. JMLR. 2014;15:3221–45.
33. Pezzotti N, Höllt T, Lelieveldt B, Eisemann E, Vilanova A. Hierarchical stochastic neighbor embedding. Comp Graph Forum. 2016;35(3):21–30.
34. Pezzotti N, Lelieveldt B, Maaten L.v.d, Höllt T, Eisemann E, Vilanova A. Approximated and user steerable t-SNE for progressive visual analytics. IEEE TVCG 2017:23, 1739–1752.
35. Pezzotti N, Thijssen J, Mordvintsev A, Hollt T, Lew B.v, Lelieveldt B, Eisemann E, Vilanova A. GPGPU linear complexity t-SNE optimization. IEEE TVCG 2020;26(1):1172–1181.
36. Chan D, Rao R, Huang F, Canny J. T-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. In: Proc. SBAC-PAD, 2018;pp. 330–338.
37. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504–7.
38. Kingma DP, Welling M. Auto-encoding variational bayes. CoRR **abs/1312.6114** 2013. eprint: arXiv:1312.6114
39. Kohonen T. Self-organizing maps. Berlin: Springer; 1997.
40. Becker M, Lippel J, Stuhlsatz A, Zielke T. Robust dimensionality reduction for data visualization with deep neural networks. Graph Models. 2020;108: 101060.
41. Fisher RA. The use of multiple measurements in taxonomic problems. Ann Eugen. 1936;7(2):179–88.
42. Espadoto M, Hirata N, Telea A. Self-supervised dimensionality reduction with neural networks and pseudo-labeling. In: Proc. IVAPP 2021.
43. Xu R, Wunsch D. Survey of clustering algorithms. IEEE Trans Neural Networks. 2005;16(3):645–78.
44. Berkhin P. A survey of clustering data mining techniques. In: Grouping multidimensional data. Berlin: Springer; 2006. p. 25–71.
45. Fukunaga K, Hostetler L. The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Trans Inf Theor. 1975;21(1):32–40.
46. Cheng Y. Mean shift, mode seeking, and clustering. IEEE TPAMI. 1995;17(8):790–9.
47. Hurter C, Ersoy O, Telea A. Graph bundling by kernel density estimation. Comp Graph Forum 2012;31(3):865–874. Wiley Online Library.
48. Epanechnikov V. Non-parametric estimation of a multivariate probability density. Theor Probab Appl+ **14** 1969
49. Agarap A.F. Deep Learning using Rectified Linear Units (ReLU). arXiv:1803.08375 [cs.NE] 2018
50. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: Proc. ICCV, 2015;pp. 1026–1034.
51. Kingma D.P, Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980 2014.
52. van der Zwan M, Codreanu V, Telea A. CUBu: Universal real-time bundling for large graphs. IEEE TVCG. 2016;22(12):2550–63.

53. Martins RM, Minghim R, Telea AC, *et al.* Explaining neighborhood preservation for multidimensional projections. In: Proc. CGVC, 2015;pp. 7–14.
54. Vito S.D, Massera E, Piga M, Martinotto L, Francia GD. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sensors and Actuators B: Chemical 2008;129(2), 750–757. https://archive.ics.uci.edu/ml/datasets/Air+Quality.
55. Yeh I-C. Modeling of strength of high-performance concrete using artificial neural networks. Cem Concr Res. 1998;28(12):1797–808.
56. Thoma M. The Reuters Dataset. https://martin-thoma.com/nlp-reuters 2017.
57. Salton G, McGill MJ. Introduction to Modern Information Retrieval, 1986. McGraw-Hill.
58. Hopkins M, Reeber E, Forman G, Suermondt J. Spambase dataset. Hewlett-Packard Labs 1999.
59. Street N, Wolberg W, Mangasarian O. Nuclear feature extraction for breast tumor diagnosis. In: Biomedical Image Processing and Biomedical Visualization, vol. 1905, 2014;pp. 861–870 1993.
60. Cortez P, Cerdeira A, Almeida F, Matos T, Reis J. Modeling wine preferences by data mining from physicochemical properties. Decis Support Sys. 2009;47(4):547–53.
61. Guennebaud G, Jacob B, et al. Eigen v3. http://eigen.tuxfamily.org 2010.
62. Blanco J.L, Rai P.K. nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees. https://github.com/jlblancoc/nanoflann 2014.
63. Lisitsyn S, Widmer C, Garcia FJI. Tapkee: An efficient dimension reduction library. JMLR. 2013;14:2355–9.
64. Chollet F, et al.: Keras. https://keras.io 2015.
65. The Authors: α-SDR-NNP implementation and results. https://github.com/youngjookim/sdr (2021).
66. Buder *et al.* S. The GALAH Survey: Second data release. Mon R R Astron Soc 2018**478**.
67. Collaboration Gaia. The Gaia mission. Astron Astrophys. 2016;595:A1.
68. Collaboration Gaia. Gaia Data Release 2-Summary of the contents and survey properties. Astron Astrophys. 2018;616:A1.
69. Tinsley B. Evolution of the stars and gas in galaxies. Fundam Cosm Phys. 1980;5:287–388.
70. Rahaman M, Li C, Yao Y, Kulwa F, Rahman MA, Wang Q, Qi S, Kong F, Zhu X, Zhao X. Identification of COVID-19 samples from chest X-ray images using deep learning: A comparison of transfer learning approaches. J Xray Sci Technol. 2020;28(5):821–39.
71. Chen H, Li C, Wang G, Li X, Rahaman M, Sun H, Hu W, Li Y, Liu W, Sun C, Ai S, Grzegorzek M. GasHis-transformer: A multiscale visual transformer approach for gastric histopathological image detection. Pattern Recogn. 2022;130: 108827.
72. Liu W, Li C, Xu N, Jiang T, Rahaman M, Sun H, Wu X, Hu W, Chen H, Sun C, Yao Y, Grzegorzek M. CVM-Cervix: A hybrid cervical Pap-smear image classification framework using CNN, visual transformer and multilayer perceptron. Pattern Recogn. 2022;130: 108829.
73. Zhang J, Li C, Kosov S, Grzegorzek M, Shirahamad K, Jiang T, Sun C, Li Z, Li H. LCU-Net: a novel low-cost U-Net for environmental microorganism image segmentation. Pattern Recogn. 2021;115: 107885.