Improving Self-Supervised Dimensionality Reduction: Exploring Hyperparameters and Pseudo-labeling Strategies

Artur André A. M. Oliveira[1 [0000−0002−3606−1687]], Mateus Espadoto[1 [0000−0002−1922−4309]], Roberto Hirata Jr.[1 [0000−0003−3861−7260]], Nina S. T. Hirata[1 [0000−0001−9722−5764]], and Alexandru C. Telea[2 [0000−0003−0750−0502]]

[1] Institute of Mathematics and Statistics, University of São Paulo, Brazil
[2] Department of Information and Computing Sciences, Utrecht University, The Netherlands
arturao@ime.usp.br, mespadot@ime.usp.br, hirata@ime.usp.br, nina@ime.usp.br, a.c.telea@uu.nl

Abstract. Dimensionality reduction (DR) is an essential tool for the visualization of high-dimensional data. The recently proposed Self-Supervised Network Projection (SSNP) method addresses DR with a number of attractive features, such as high computational scalability, genericity, stability and out-of-sample support, computation of an inverse mapping, and the ability of data clustering. Yet, SSNP has an involved computational pipeline using self-supervision based on labels produced by clustering methods and two separate deep learning networks with multiple hyperparameters. In this paper we explore the SSNP method in detail by studying its hyperparameter space and pseudo-labeling strategies. We show how these affect SSNP's quality and how to set them to optimal values based on extensive evaluations involving multiple datasets, DR methods, and clustering algorithms.

Keywords: Dimensionality Reduction · Machine Learning · Deep Learning · Neural Networks · Autoencoders

1 INTRODUCTION

Visualization of high-dimensional data to find patterns, trends, and overall understand the data structure has become an essential ingredient of the data scientist's toolkit [24,29]. Within the palette of such visualization methods, dimensionality reduction (DR) techniques, also called projections, have gained an established position due to their high scalability both in the number of samples and number of dimensions thereof. In the last decades, tens of DR techniques have emerged [38,12], with PCA [22], t-SNE [33], and UMAP [36] having become particularly popular.

Neural-network-based techniques have been used to support DR, early examples of such approaches being self-organizing maps [26] and autoencoders [19]. More recently, the NNP technique [10] was proposed to mimic any DR technique. In parallel, the ReNDA method [3] was proposed to improve the projection quality offered by autoencoders.

Deep learning based DR methods are very fast, simple to implement, generically work for any type of quantitative high-dimensional data, are parametric, thus stable to small-scale data variations and offering out-of-sample capability, and – in the case of autoencoders – also provide the inverse mapping from the low-dimensional projection space to the high-dimensional data space. However, such methods also have some limitations. Such methods cannot typically offer the same projection quality, measured *e.g.* in terms of neighborhood preservation or cluster delineation, as classical methods like t-SNE and UMAP [10,37,9]. Inverse projection typically requires training a separate network [13]. NNP-class methods offer a higher quality than autoencoders, but require supervision in terms of using a classical DR method to project a subset of the data [10].

Recently, the Self-Supervised Neural Projection (SSNP [11]) method was proposed to alleviate the above limitations of deep learned projections. SSNP uses a single neural network trained with two objectives – *reconstructing* the projected data (as an autoencoder does) and *classifying* the same data (based on pseudo-labels created by a clustering algorithm). In more detail, SSNP aims to provide the following characteristics:

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

Scalability (C2): Linear complexity in the number of samples and dimensions, allowing the projection of datasets of a million samples and hundreds of dimensions in a few seconds on consumer-grade GPU platforms;

Ease of use (C3): Minimal or no hyperparameter tuning required;

Genericity (C4): Projects any dataset whose samples are real-valued vectors;

Stability and out-of-sample support (C5): The trained SSNP model can project new samples along existing ones in a parametric fashion;

Inverse mapping (C6): Ability to infer the high-dimensional point corresponding to a low-dimensional point in the projection space;

Clustering (C7): Ability to label (cluster) unseen data. This feature of SSNP also supports requirement C1: Intuitively, clustering aggregates low-level distance information between sample points to a higher level, telling how groups of samples relate to each other. Next, this information is used by SSNP to produce projections which preserve such data clusters well in the low dimensional space.

In our original paper [11], we show how SSNP achieves the above requirements by evaluating it on four synthetic and four real-world datasets, using two clustering algorithms to produce pseudo-labels, and compare its results with four existing DR techniques. However, this leaves the 'design space' of SSNP insufficiently explored. Similarly to [9], where the authors explored in detail the design space of NNP [10], in this paper we aim to provide more insights on how SSNP's results depend on its technical components and their hyperparameter settings. For this, we extend the evaluation in [11] by considering two additional projection techniques (MDS and Isomap) and four additional clustering algorithms (affinity propagation, DBSCAN, Gaussian mixture models, and spectral clustering). Separately, we study how SSNP's performance is influenced by the setting of the hyperparameters of both the clustering algorithms and the underlying neural network. All in all, our extended evaluation proves that SSNP

indeed complies well with requirements C1-C7, being a serious contender in the class of deep-learning-based DR techniques.

We structure this chapter as follows: Section 2 introduces notations and discusses related work. Section 3 details the SSNP method. Section 4 describes our experimental setup. Section 5 presents the results of SSNP, including the additional experiments outlined above. Section 6 discusses the obtained findings. Section 7 concludes the paper.

2 BACKGROUND

Notations: Let $\mathbf{x} = (x^1, \ldots, x^n)$, $x^i \in \mathbb{R}, 1 \leq i \leq n$ be a n-dimensional (nD) sample (also called a data point or observation). Let $D = \{\mathbf{x}_i\}$, $1 \leq i \leq N$ be a dataset of N such samples, *e.g.*, a table with N rows (samples) and n columns (dimensions). All datasets D used in this paper have class labels. Let C be the number of classes (or labels) in a dataset D. A DR, or projection, technique is a function

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

where $q \ll n$, and typically $q = 2$. The projection $\mathbf{p} = P(\mathbf{x})$ of a sample $\mathbf{x} \in D$ is a point $\mathbf{p} \in \mathbb{R}^q$. Projecting an entire dataset D yields a q-dimensional scatterplot, denoted next as $P(D)$. The inverse of P, denoted $\mathbf{x} = P^{-1}(\mathbf{p})$, maps a q-dimensional point $\mathbf{p}$ to the high-dimensional space $\mathbb{R}^n$, so that, ideally, $P(\mathbf{x}) = \mathbf{p}$, or in practice, $P(\mathbf{x})$ is close to $\mathbf{p}$.

Dimensionality reduction: Many DR methods have been proposed in the last decades [20,34,8,48,29,5,56,38,12]. We next outline how a few representative ones comply with the requirements mentioned in Sec. 1, supporting our point that no DR method fully covers all those requirements. For further evidence for this statement, we refer to the above mentioned surveys.

Principal Component Analysis [22] (PCA) is very popular due to its simplicity, speed (C2), stability and out-of-sample (OOS) support (C5), and ease of use (C3) and interpretation. PCA is also used as pre-processing step for other DR techniques that require not-too-high-dimensional data [38]. Yet, due to its linear and global nature, PCA lacks on quality (C1), especially for data of high intrinsic dimensionality.

Methods of the Manifold Learning family (MDS [51], Isomap [49], and LLE [45] and its variations [7,58,57]) aim to map to 2D the high-dimensional manifold on which data lives. Such methods generally yield higher quality (C1) than PCA. Yet, such methods can be hard to tune (C3), do not have OOS capability (C5), do not work well for data that is not restricted to a 2D manifold, and generally scale poorly (C2) with dataset size.

Force-directed methods (LAMP [21] and LSP [40]) can yield reasonably high visual quality (C1), good scalability (C2), and are simple to use (C3). However, they generally cannot do OOS (C5). For LAMP, a related inverse projection (C6) technique iLAMP [1] exists. Yet, LAMP and iLAMP are two different algorithms. Clustering-based methods, such as PBC [39], share many characteristics of force-directed methods, such as good quality (C1) and lack of OOS (C5).

SNE (Stochastic Neighborhood Embedding) methods, of which t-SNE [33] is the most popular, have the key ability to visually segregate similar samples, thus being very good for cluster analysis. While having high visual quality (C1), t-SNE has a high

complexity of $O(N^2)$ in sample count (C2), is very sensitive to small data changes (C5), is hard to tune (C3) [54], and has no OOS capability (C5). Tree-accelerated t-SNE [32], hierarchical SNE [42], approximated t-SNE [43], and various GPU accelerations of t-SNE [44,4] improve computation time (C2). Yet, these methods require quite complex algorithms, and still largely suffer from the aforementioned sensitivity, tuning, and OOS issues. Uniform Manifold Approximation and Projection (UMAP) [36] generates projections with comparable quality to t-SNE (C1) but is faster (C2) and has OOS (C5). Yet, UMAP shares some disadvantages with t-SNE, namely the sensitivity to small data changes (C5) and parameter tuning difficulty (C3).

Deep learning: Autoencoders (AE) [19,25] create a low-dimensional data representation in their bottleneck layers by training a neural network to reproduce its high-dimensional inputs on its outputs. They produce results of comparable quality (C1) to PCA. However, they are easy to set up, train, and use (C3), are easily parallelizable (C2), and have OOS (C5) and inverse mapping (C6) abilities.

ReNDA [3] is a deep learning approach that uses two neural networks, improving on earlier work from the same authors. One network implements a nonlinear generalization of Fisher's Linear Discriminant Analysis [15]; the other network is an autoencoder used as a regularizer. ReNDA scores well on quality (C1) and has OOS (C5). However, it requires pre-training of each individual network and has low scalability (C2).

Neural Network Projections (NNP) [10] select a training subset $D_s \subset D$ to project by any user-chosen DR method to create a so-called training projection $P(D_s) \subset \mathbb{R}^2$. Next, a neural network is trained to approximate $P(D_s)$ having D_s as input. The trained network then projects unseen data by means of 2-dimensional non-linear regression. NNP is very fast (C2), simple to use (C3), and stable and with OOS ability (C5). However, the projection quality (C1) is lower than the learned projection. The NNInv technique [13], proposed by the same authors as NNP, adds inverse projection ability (C6). However, this requires setting up, training, and using a separate network.

Table 1 summarizes how the above DR techniques fare with respect to each characteristic of interest. The last row highlights SSNP which we describe separately in Sec. 3.

Table 1: Summary of DR techniques and their characteristics. Names in *italic* are techniques we compare with SSNP.

	Characteristic						
Technique	**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**

Clustering: As for DR, clustering is a field that goes back decades, with many techniques proposed over the years. Despite using different approaches, all techniques use some form of similarity measure to determine whether a sample belongs to a cluster or not. *Centroid-based* techniques, such as K-means [30], compute cluster centers and assign cluster membership based on closeness to a center. *Connectivity-based* techniques, such as Agglomerative clustering [23], group samples based their relative distances rather than distances to cluster centers. *Distribution-based* techniques, such as Gaussian Mixture Models [6], fit Gaussian distributions to the dataset and then assign samples to each distribution. *Density-based* techniques, such as DBSCAN [14], define clusters as dense areas in the data space. More recent techniques use more specialized approaches, such as Affinity Propagation [16], which uses message passing between samples, and Spectral Clustering [47], which uses the eigenvalues of the data similarity matrix to reduce the dimensionality of the data to be clustered.

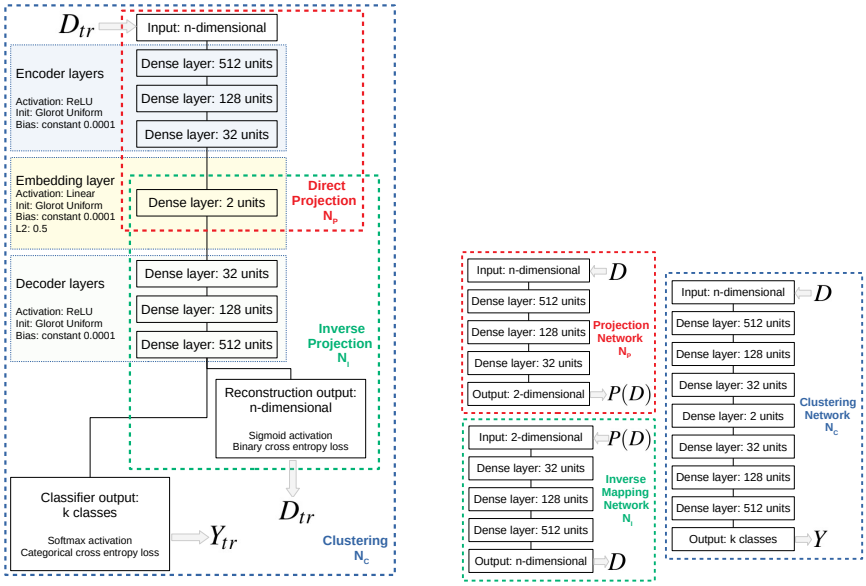
3 SSNP TECHNIQUE

As stated in Sec. 2, autoencoders have desirable DR properties (simplicity, speed, OOS, and inverse mapping abilities), but create projections of lower quality than, *e.g.*, t-SNE and UMAP. A likely cause for this is that autoencoders do not use neighborhood information during training, while t-SNE and UMAP (obviously) do that. Hence, we propose to create an autoencoder architecture with a *dual* optimization target that explicitly uses neighborhood information. First, we have a *reconstruction* target, as in standard autoencoders; next, we use a *classification* target based on labels associated with the samples. These can be "true" ground-truth labels if available for a given dataset. If not, these are pseudo-labels created by running a clustering algorithm on the input dataset. The key idea behind this is that (pseudo)labels are a compact and high-level way to encode neighborhood information, *i.e.*, same-label data are more similar than different-label data. Since classifiers learn a representation that separates input data based on labels, adding an extra classifier target to an autoencoder learns 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 $Y_{tr} \in \mathbb{N}$ by using some clustering technique. We then take samples $(\mathbf{x} \in D_{tr}, y \in Y_{tr})$ to train a neural network with a reconstruction and a classification function, added to form a joint loss. This network (Fig. 1a) 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 layers of the network to create three new networks for inference (Fig. 1b): a *projector* $N_P(\mathbf{x})$, an *inverse projector* $N_I(\mathbf{p})$, and a *classifier* $N_C(\mathbf{x})$, which mimics the clustering algorithm used to create Y_{tr}. The entire training-and-inference way of working of SSNP is summarized in Figure 2.

4 EXPERIMENTAL SETUP

In this section we detail the experimental setup we used to evaluate SSNP's performance. The obtained results are discussed next in Sec. 5.



(a) Training (b) Inference

Fig. 1: SSNP network architectures used during training (a) and inference (b).

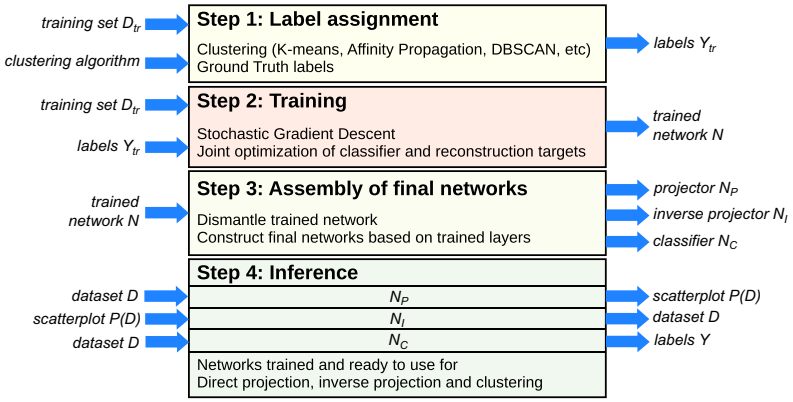


Fig. 2: SSNP training-and-inference pipeline.

4.1 Datasets

We first evaluate SSNP on synthetic datasets consisting of blobs sampled from a Gaussian distribution of different dimensionalities (100 and 700), number of clusters (5 and 10), and standard deviation σ, yielding datasets with cluster separation varying from very sharp to fuzzy clusters. All synthetic datasets have 5K samples. Next, we evaluate SSNP on four public real-world datasets that are high-dimensional, reasonably large (thousands

of samples), and have a non-trivial data structure (same datasets as used in the original SSNP paper [11]):

MNIST [28]: 70K samples of handwritten digits from 0 to 9, rendered as 28x28-pixel gray scale images, flattened to 784-element vectors;

Fashion MNIST [55]: 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) [2]: 10299 samples from 30 subjects performing activities of daily living used for human activity recognition grouped in 6 classes and described with 561 dimensions.

Reuters Newswire Dataset [50]: 8432 samples of news report documents, from which 5000 attributes are extracted using TF-IDF [46], a standard method in text processing.

All datasets had their attributes rescaled to the range $[0,1]$, to conform with the sigmoid activation function used by the reconstruction layer (see Fig. 1a).

4.2 Projection Quality Metrics

We measure projection quality by four metrics widely used in the projection literature (see Tab. 2 for their definitions). All metrics range in $[0,1]$ with 0 indicating poorest, and 1 indicating best, values:

Table 2: Projection quality metrics used in evaluating SSNP

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)	$\frac{1}{N}\sum_{\mathbf{y}\in P(D)}\frac{\mathbf{y}_k^l}{\mathbf{y}_k}$
Shepard diagram correlation (R)	Spearman's ρ of $(\|\mathbf{x}_i-\mathbf{x}_j\|, \|P(\mathbf{x}_i)-P(\mathbf{x}_j)\|), 1\le i\le N, i\neq j$

Trustworthiness T [53] is 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 data patterns. In the definition (Tab. 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 $K=7$ following [34,35];

Continuity C [53] is the fraction of close points in $P(D)$ that are also close in D. In the definition (Tab. 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 for T, we use $K=7$;

Neighborhood hit NH [40] 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 $\mathbf{y}_K^l$ of the K nearest neighbors of a point $\mathbf{y}\in P(D)$, denoted by $\mathbf{y}_K$, that have the same label as $\mathbf{y}$, averaged over $P(D)$. In this paper, we use $K=3$;

Shepard diagram correlation R [21]: 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 diagonal by computing its Spearman rank correlation R. A value of $R = 1$ indicates a perfect (positive) correlation of distances.

4.3 Dimensionality Reduction Techniques Compared Against

We compared SSNP against six DR techniques, namely t-SNE, UMAP, MDS, Isomap, autoencoders (AE), and NNP (see also Tab. 1). We selected these techniques based on popularity (t-SNE, UMAP, MDS, Isomap) or on similar operation (AE and NNP are also deep learning based, like SSNP) and also on having desirable properties to compare against. For instance, t-SNE and UMAP are known to produce strong visual cluster separation by evaluating local neighborhoods. MDS, on the other hand, tries to preserve global distances between samples. Isomap can be seen as an extension of MDS that uses local neighborhood information to infer geodesic distances. AE produce results similar to PCA, which preserves global distances. Finally, NNP does not have specific built-in heuristics but rather aims to mimic and accelerate other DR techniques. For all these DR techniques, we used default values for their hyperparameters.

4.4 Clustering Techniques for Pseudo-labeling

In addition to using ground-truth labels in SSNP, we also used six clustering algorithms to generate the pseudo-labels for using during SSNP training (Sec. 3). Table 3 lists all clustering algorithms used, as well as the hyperparameters used in all experiments, except when noted otherwise. Hyperparameters not listed in Tab. 3 used default values. We used these algorithms since they employ quite different approaches to clustering, which could produce different results for SSNP.

We selected two of these clustering algorithms alongside two datasets — K-means and DBSCAN, HAR and MNIST — to further explore the effect of their main hyperparameters on the quality of the SSNP projection. For K-means, we studied the $n\_clusters$ parameter by choosing values well below and above the known number of clusters C in the data — $n\_clusters = \{5, 10, 15, 20, 30\}$ for MNIST ($C = 10$), $n\_clusters = \{3, 6, 9, 12, 18\}$ for HAR ($C = 6$). For DBSCAN, we explored the eps parameter, which determines the maximum distance between samples for them to be considered as neighbors. We used $eps = \{6.1, 6.3, ..., 6.9\}$ for MNIST and $eps = \{1.9, 2.1, ..., 2.7\}$ for HAR.

4.5 Neural Network Hyperparameter Settings

We further evaluated SSNP by using several hyperparameter settings for its neural network. To avoid a huge hyperparameter space, for each parameter explored, we kept the other parameters set to their defaults, similarly to the strategy used to explore NNP [9]. The explored hyperparameters are described next (see also Tab. 4).

Table 3: Clustering algorithms used as for pseudo-label creation and their hyperparameters used during testing. Ground-truth is listed here as another labeling strategy.

Algorithm	**Acronym**	**Hyperparameters**
Ground Truth Labels	SSNP(GT)	none
Affinity Propagation	SSNP(AP)	none
Agglomerative Clustering	SSNP(Agg)	$n\_clusters = 2 \times C$
DBSCAN	SSNP(DB)	$eps = 5$
Gaussian Mixture Model	SSNP(GMM)	$n\_components = 2 \times C$
K-means	SSNP(Km)	$n\_clusters = 2 \times C$
Spectral Clustering	SSNP(SC)	$n\_clusters = 2 \times C$

L2 regularization [27] decreases layer weights to small but non-null values, leading to every weight only slightly contributing to the model. It works by adding a penalization term $\lambda\|\mathbf{w}\|^2$ to the cost function, where $\mathbf{w}$ are the weights of a selected network layer. The parameter $\lambda \in [0,1]$ controls the amount of regularization;

Embedding layer activation: The embedding (bottleneck) layer creates the 2D projection after training (Fig. 1). Changing the activation function of this layer affects the projection's overall shape. We used four activation functions for this layer (see Tab. 4);

Weight initialization: A neural network has thousands of parameters whose initialization can affect the training outcome. We used three common initialization types: random uniformly distributed in the range $[-0.05, 0.05]$, Glorot uniform [17] with the range $[-b,b]$ for $b = \sqrt{6/(l_{in}+l_{out})}$, where l_{in} and l_{out} are the number of input and output units in the layer, and He uniform [18], which uses the range $[-b,b]$ with $b = \sqrt{6/l_{in}}$;

Training epochs: We explored SSNP's performance for different numbers of epochs η ranging from 1 to 20.

Table 4: SSNP neural network parameters explored with default values in bold.

Dimension	**Values**
L2 regularization	$\lambda = \{\mathbf{0}, 0.1, 0.5, 1.0\}$
Embedding layer activation	$\alpha = \{$ **ReLU**, sigmoid, tanh, Leaky RELU $\}$
Weight initialization	$\phi = \{$**Glorot uniform**, He uniform, Random uniform $\}$
Training Epochs	$\eta = \{1, 2, 3, 5, \mathbf{10}, 20\}$

5 RESULTS

We next present the results for all experiments conducted to demonstrate SSNP's quality and robustness to hyperparameter selection.

5.1 Quality On Synthetic Datasets

Figure 3 shows the SSNP projection of the synthetic blob datasets with SSNP(Km) with K-means set to use the correct (ground-truth) number of clusters alongside AE, t-SNE, and UMAP. In most cases SSNP(Km) shows better visual cluster separation than

autoencoders. The t-SNE and UMAP projections look almost the same regardless of the standard deviation σ of the blobs, while SSNP(Km) shows more spread clusters for larger σ, which is the desired effect. We omit the plots and measurements for NNP for space reasons and since these are very close to the ones created by the learned technique [10].

Table 5 shows the quality metrics for this experiment for datasets using 5 and 10 clusters. For all configurations, SSNP performs very similarly quality-wise to AE, t-SNE, and UMAP. Section 5.2, which studies more challenging, real-world, datasets will bring more insight in this comparison.

Table 5: Quality metrics, synthetic blobs experiment with 100 and 700 dimensions, 5 and 10 clusters, and $\sigma \in [1.3, 11.2]$.

		100 dimensions									**700 dimensions**							
		5 clusters				**10 clusters**					**5 clusters**				**10 clusters**			
Projection	σ	*T*	*C*	*R*	*NH*	*T*	*C*	*R*	*NH*	σ	*T*	*C*	*R*	*NH*	*T*	*C*	*R*	*NH*
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

5.2 Quality On Real-World Datasets

Figure 4 shows the projections of real-world datasets by SSNP with ground-truth labels (SSNP(GT)), SSNP with pseudo-labels created by the six clustering algorithms in Tab. 3, and projections created by AE, t-SNE, UMAP, MDS, and Isomap. We omit again the results for NNP since they are very close to the ones created by t-SNE and UMAP. SSNP and AE 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.

SSNP with pseudo-labels shows better cluster separation than AE but slightly worse than SSNP(GT). For the more challenging HAR and Reuters datasets, SSNP(GT) looks better than t-SNE and UMAP. In almost all cases, SSNP yields a better visual cluster separation than MDS and Isomap. We see also that, for almost all clustering algorithm-dataset combinations, SSNP creates elongated clusters in a star-like pattern. We believe this is so since one of the network's targets is a *classifier* (Sec. 3) 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; samples that are far away from a decision boundary are placed near the tips of the star, according to its class.

Table 6 shows the four quality metrics (Sec. 4.2) for this experiment. SSNP with pseudo-labels consistently shows better cluster separation (higher *NH*) than AE as well as better distance preservation (higher *R*). For the harder HAR and Reuters datasets, SSNP(GT) shows *NH* results that are similar to and even higher than those for t-SNE and UMAP. Also, SSNP(GT) scores consistently higher than MDS and Isomap on all quality metrics, which correlates with these two projection techniques having been found as of moderate quality in earlier studies [12]. For the *T* and *C* metrics, SSNP(GT) outperforms again AE in most cases; for FashionMNIST and HAR, SSNP yields *T* and *C* values close

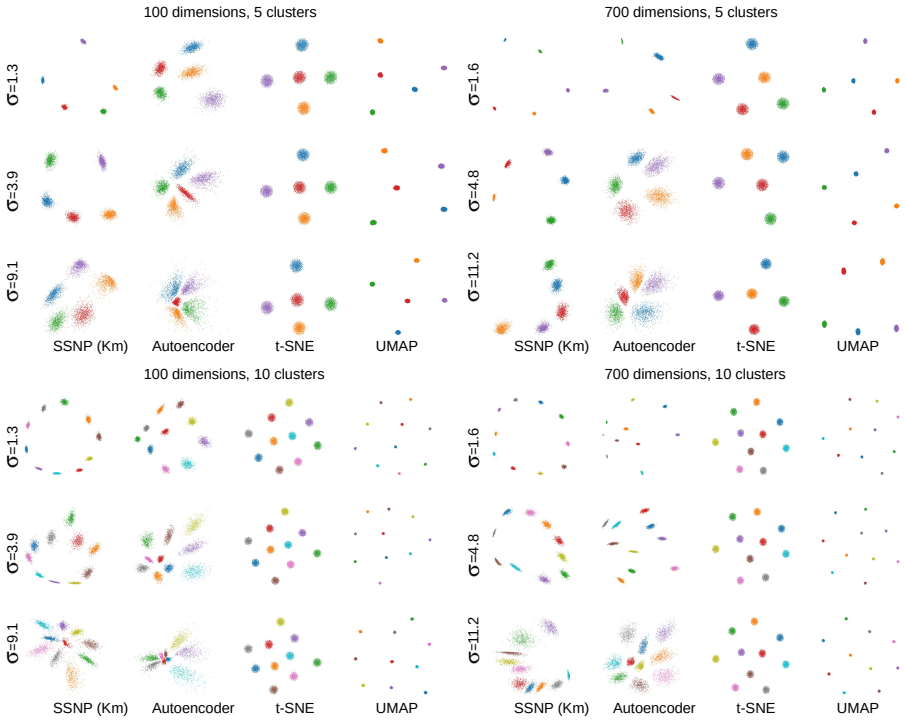


Fig. 3: Projection of synthetic blobs datasets with SSNP(Km) and other techniques, with different number of dimensions and clusters. In each quadrant, rows show datasets having increasing standard deviation σ.

to the ones for NNP, t-SNE, and UMAP. Separately, we see that the clustering algorithm choice influences the four quality metrics in several ways. DBSCAN (DB) yields in nearly all cases the lowest quality values while K-means (Km) and Agglomerative (AG) yield overall the best quality values. Spectral clustering (SC) is also a quite good option if one is mainly interested in cluster separation (high *NH* values). Finally, Affinity Propagation (AP) and Gaussian Mixture Models (GMM) score in between Km and AG (best overall) and DB (worst overall). From the above, we conclude that Km and AG are good default clustering methods that SSNP can use in practice.

5.3 Quality *vs* Clustering Hyperparameters

Figure 5 shows projections of the HAR and MNIST datasets created by SSNP with pseudo-labels assigned by DBSCAN and K-means and using the various clustering hyperparameter settings described in Sec. 4.4.

For DBSCAN, we see that as the value of *eps* increases, the SSNP projection seems to vary between global- and local-distance preservation. This effect is more pronounced for the HAR dataset, where we see the number of clusters in the data varying from two

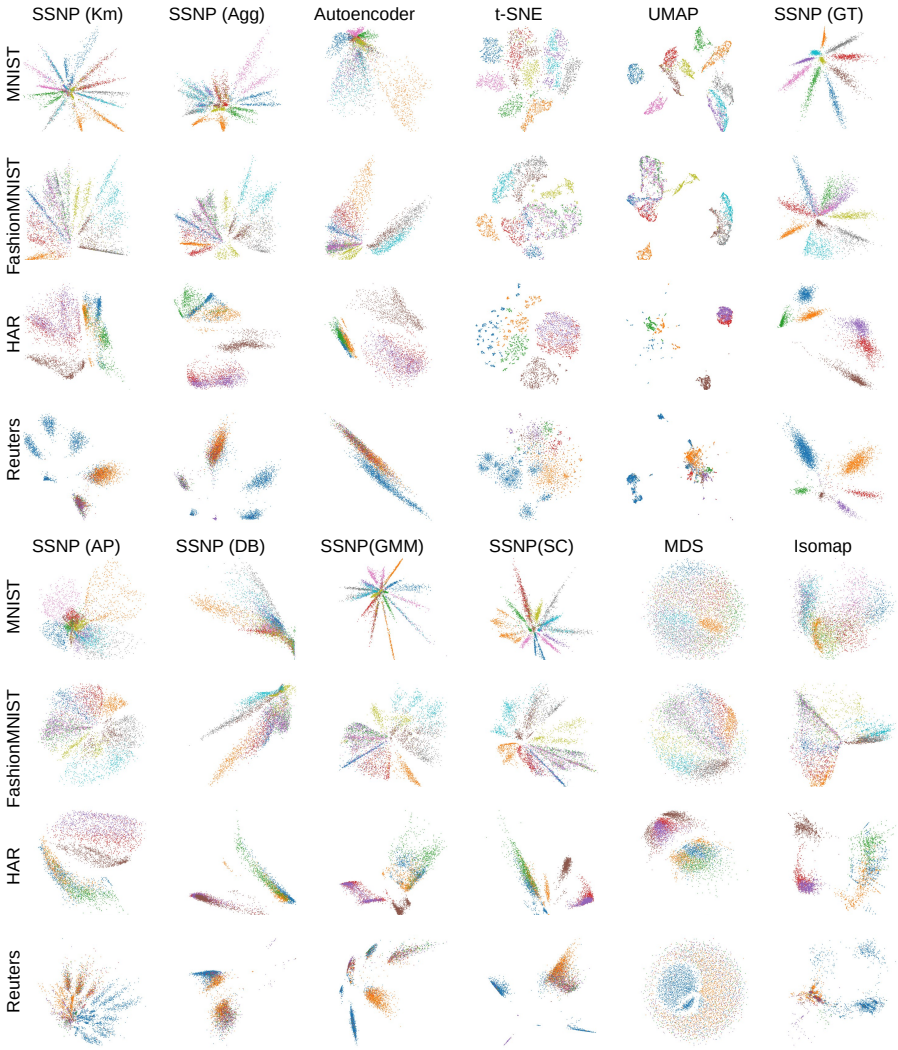


Fig. 4: Projection of real-world datasets with SSNP (ground-truth labels and pseudo-labels computed by six clustering methods) compared to Autoencoders, t-SNE, UMAP, MDS, and Isomap.

($eps = 1.9$) and three ($eps = 2.7$). For the MNIST dataset, the increase in eps only makes the entire projection take a sharper shape, with no improvement in cluster separation. Overall, SSNP with DBSCAN having low eps values produces results similar to an autoencoder, which defeats the purpose of using SSNP. This correlates to the earlier findings in Sec. 5.2 that showed that DBSCAN is not a good clustering companion for

Table 6: Quality measurements for the real-world datasets (Sec. 5.2).

Dataset	Method	*T*	*C*	*R*	*NH*	Method	*T*	*C*	*R*	*NH*
MNIST	SSNP(Km)	0.882	0.903	0.264	0.767	SSNP(AP)	0.827	0.940	0.094	0.729
	SSNP(AG)	0.859	0.925	0.262	0.800	SSNP(DB)	0.689	0.802	0.032	0.588
	AE	0.887	0.920	0.009	0.726	SSNP(GMM)	0.880	0.895	0.257	0.755
	SSNP(GT)	0.774	0.920	0.398	0.986	SSNP(SC)	0.849	0.925	0.164	0.831
	NNP	0.948	0.969	0.397	0.891	MDS	0.754	0.862	0.618	0.580
	TSNE	0.985	0.972	0.412	0.944	Isomap	0.759	0.958	0.528	0.618
	UMAP	0.958	0.974	0.389	0.913					
FashionMNIST	SSNP(Km)	0.958	0.982	0.757	0.739	SSNP(AP)	0.947	0.986	0.750	0.728
	SSNP(AG)	0.950	0.978	0.707	0.753	SSNP(DB)	0.890	0.921	0.431	0.665
	AE	0.961	0.977	0.538	0.725	SSNP(GMM)	0.952	0.982	0.689	0.737
	SSNP(GT)	0.863	0.944	0.466	0.884	SSNP(SC)	0.957	0.981	0.706	0.756
	NNP	0.963	0.986	0.679	0.765	MDS	0.923	0.957	0.903	0.652
	TSNE	0.990	0.987	0.664	0.843	Isomap	0.920	0.976	0.749	0.685
	UMAP	0.982	0.988	0.633	0.805					
HAR	SSNP(Km)	0.932	0.969	0.761	0.811	SSNP(AP)	0.929	0.972	0.736	0.787
	SSNP(AG)	0.926	0.964	0.724	0.846	SSNP(DB)	0.852	0.909	0.759	0.690
	AE	0.937	0.970	0.805	0.786	SSNP(GMM)	0.924	0.966	0.768	0.796
	SSNP(GT)	0.876	0.946	0.746	0.985	SSNP(SC)	0.893	0.952	0.811	0.805
	NNP	0.961	0.984	0.592	0.903	MDS	0.911	0.890	0.941	0.765
	TSNE	0.992	0.985	0.578	0.969	Isomap	0.925	0.971	0.896	0.861
	UMAP	0.980	0.989	0.737	0.933					
Reuters	SSNP(Km)	0.794	0.859	0.605	0.738	SSNP(AP)	0.631	0.768	0.039	0.742
	SSNP(AG)	0.771	0.824	0.507	0.736	SSNP(DB)	0.574	0.650	0.360	0.705
	AE	0.747	0.731	0.420	0.685	SSNP(GMM)	0.622	0.788	0.460	0.793
	SSNP(GT)	0.720	0.810	0.426	0.977	SSNP(SC)	0.607	0.758	0.027	0.730
	NNP	0.904	0.957	0.594	0.860	MDS	0.575	0.757	0.551	0.699
	TSNE	0.955	0.959	0.588	0.887	Isomap	0.634	0.785	0.150	0.765
	UMAP	0.930	0.963	0.674	0.884					

Table 7: Quality measurements for the cluster hyperparameter experiment (Sec . 5.3).

Dataset	Technique	Parameter	*T*	*C*	*R*	*NH*
MNIST	DBSCAN	eps=6.1	0.685	0.821	0.097	0.555
		eps=6.3	0.679	0.798	0.012	0.570
		eps=6.5	0.722	0.812	0.044	0.614
		eps=6.7	0.698	0.801	0.022	0.576
		eps=6.9	0.729	0.825	0.011	0.605
	K-means	n_clusters=5	0.782	0.905	0.408	0.641
		n_clusters=10	0.834	0.916	0.379	0.697
		n_clusters=15	0.867	0.927	0.410	0.760
		n_clusters=20	0.880	0.909	0.047	0.755
		n_clusters=30	0.899	0.932	0.358	0.790
HAR	DBSCAN	eps=1.9	0.854	0.928	0.917	0.696
		eps=2.1	0.848	0.920	0.841	0.650
		eps=2.3	0.875	0.914	0.717	0.685
		eps=2.5	0.896	0.924	0.844	0.725
		eps=2.7	0.898	0.933	0.887	0.749
	K-means	n_clusters=3	0.887	0.939	0.932	0.693
		n_clusters=6	0.921	0.959	0.749	0.767
		n_clusters=9	0.920	0.965	0.877	0.812
		n_clusters=12	0.930	0.968	0.854	0.815
		n_clusters=18	0.937	0.972	0.840	0.812

SSNP. The quality metrics in Tab. 7 strengthen this hypothesis – we do not see any clear trend of these metrics being improved by varying *eps* in a specific direction.

For K-means, we see that the value of *n_clusters* has a great effect on the overall shape of the SSNP projection. Particularly, when *n_clusters* is higher than the true number of classes in the data (10 for MNIST, 6 for HAR), we see that the cluster separation gets sharper. This suggests that, when the true number of clusters is not known, starting with a reasonably high number of clusters will produce better results

for SSNP with K-means. This is confirmed by the quality metrics in Tab. 7 which show higher values for higher *n_clusters* settings.

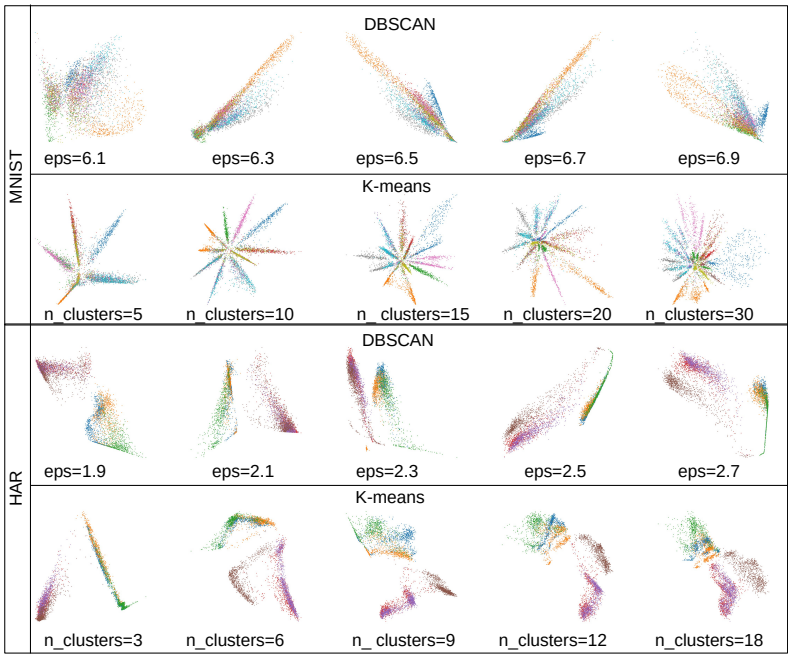


Fig. 5: Projections of MNIST and HAR datasets using different hyperparameters for the DBSCAN and K-means clustering methods (see Sec. 5.3 and Tab. 7).

5.4 Quality *vs* Neural Network Settings

We next show how the different neural network hyperparameter settings affect the SSNP results following the sampling of these parameters discussed in Sec. 4.5. We also use this analysis to derive good default values for these parameters.

L2 regularization: Figure 6 shows projections created with different amounts of L2 regularization during SSNP's training. We see that regularization has a detrimental effect to the visual quality of the projection. For values of $\lambda \geq 0.5$, the projection points collapse to a single point, marked by the red circles in the figure. Table 8 shows the metric values for this experiment confirming that all quality values decrease with λ. We conclude that SSNP obtains optimal results without regularization.

Activation functions: Figure 7 shows the effect of using different activation functions α in the embedding layer. We see that the ReLU and LeakyReLU activations produce similarly good results. Both produce visual cluster separation comparable to t-SNE and UMAP (see Fig. 4), albeit with a distinct star or radial shape. The sigmoid activation collapses all data points into a single diagonal, making it a poor choice for the embedding

Table 8: Quality measurements for SSNP for different training hyperparameters. NA indicates that the measurement failed for the respective experiment (Sec. 5.4).

Method	Parameter	Value	*T*	*C*	*R*	*NH*
SSNP(GT)	α	LeakyReLU	0.780	0.930	0.429	0.971
		ReLU	0.789	0.921	0.402	0.983
		sigmoid	0.703	0.891	0.088	0.746
		tanh	0.784	0.929	0.190	0.983
	η	2	0.781	0.924	0.428	0.903
		3	0.787	0.926	0.428	0.940
		5	0.786	0.925	0.419	0.966
		10	0.789	0.921	0.402	0.983
		20	0.797	0.920	0.391	0.989
	ϕ	Glorot	0.789	0.921	0.402	0.983
		He	0.789	0.928	0.328	0.982
		Random	0.758	0.905	0.071	0.927
	λ	0	0.789	0.921	0.402	0.983
		0.1	0.757	0.909	0.360	0.870
		0.5	0.538	0.502	NA	0.101
		1	0.538	0.502	NA	0.101
SSNP(Km)	α	LeakyReLU	0.863	0.919	0.177	0.748
		ReLU	0.888	0.916	0.119	0.768
		sigmoid	0.678	0.872	0.196	0.568
		tanh	0.884	0.928	0.265	0.774
	η	2	0.847	0.927	0.267	0.726
		3	0.827	0.926	0.244	0.714
		5	0.854	0.915	0.323	0.775
		10	0.881	0.908	0.188	0.770
		20	0.886	0.911	0.128	0.766
	ϕ	Glorot	0.884	0.915	0.333	0.784
		He	0.874	0.903	0.267	0.753
		Random	0.741	0.869	0.115	0.640
	λ	0	0.888	0.924	0.351	0.763
		0.1	0.872	0.910	0.352	0.753
		0.5	0.538	0.502	NA	0.101
		1	0.538	0.502	NA	0.101

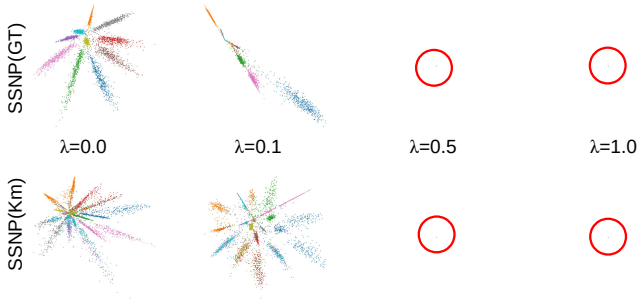


Fig. 6: Projections created with SSNP(GT) and SSNP(Km) for the MNIST dataset varying the amount of L2 regularization λ (Sec. 5.4).

layer. Finally, the tanh activation produced the best cluster separation of all, with results that look very close to the ones by t-SNE and UMAP for this dataset (see again Fig. 4). We conclude that the tanh activation function is the best option for SSNP.

Initialization: Figure 8 shows how weight initialization affects projection quality. We see that both Glorot and He uniform initializations produce good and comparable results,

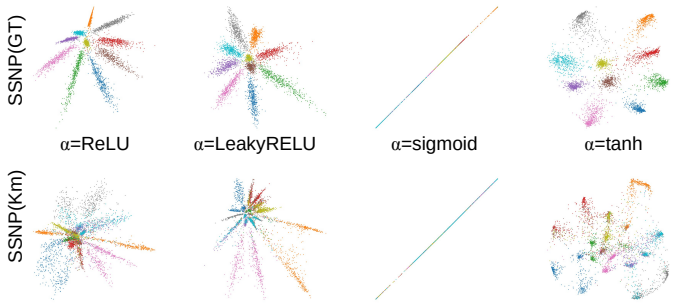


Fig. 7: Projections of the MNIST dataset using SSNP(GT) and SSNP(Km) varying the activation function α (Sec. 5.4).

whereas random initialization yields very poor results. We opt for using He uniform as the default initialization, which correlates with the same choice (obtained by an independent investigation) for NNP [9].

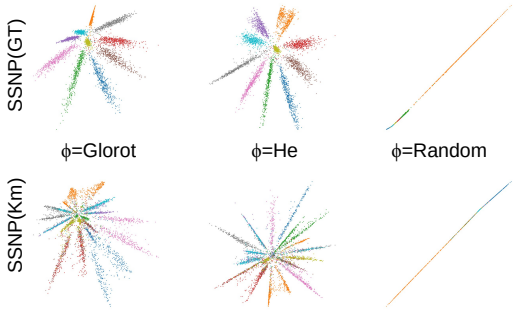


Fig. 8: Projections of the MNIST dataset using SSNP(GT) and SSNP(Km) varying the weight initialization strategy ϕ (Sec. 5.4).

Training epochs: Finally, Figure 9 shows projections created with SSNP trained for different numbers η of epochs. With as little as $\eta = 3$ training epochs, SSNP already produces good cluster separation. As η increases, the created visual clusters become sharper. However, there seems to be little improvement when going from $\eta = 10$ to $\eta = 20$. As such, we conclude that a good default is $\eta = 10$ training epochs. Interestingly, this is significantly less than the 50 epochs needed by NNP to achieve good projection quality [9], especially if we consider that SSNP has to train a more complex, dual-objective, network.

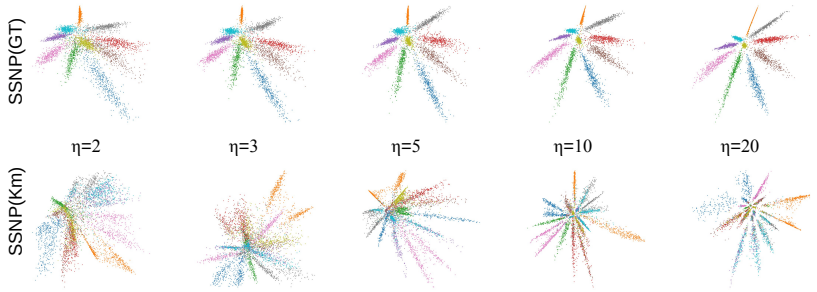


Fig. 9: Projections of MNIST dataset using SSNP(GT) and SSNP(Km) varying the number of training epochs η (Sec. 5.4).

5.5 Computational Scalability

Using SSNP means (a) training the network and next (b) using the trained network in inference mode (see also Fig. 1). We analyze these two times next.

Setup time: Table 9 shows the time needed to set up SSNP and three other projection techniques. For SSNP, NP, and AE, this is the training time of the respective neural networks using 10 training epochs. Note that we used 10K training samples, which is largely sufficient to train SSNP to obtain good results. In practice, SSNP obtains good results (quality-wise) with as few as 1K samples. For UMAP and t-SNE, this is the time needed to actually project the data since these techniques do not have a training phase. We see that the SSNP variants using clustering take about the same time as t-SNE and UMAP and less than NNP. SSNP(GT), which does not need clustering, is far faster than these competitors, with the exception of AE which is about twice faster. This is explainable since SSNP uses a dual-objective network (Sec. 3), one of these being essentially the same as AE.

Table 9: Setup time for different projection methods for 10K training samples, MNIST dataset.

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

Inference time: Figure 10 shows the time needed to project up to 1M samples using SSNP and the other compared projection techniques. For SSNP, AE, and NNP, this is the inference time using the respective trained networks. For t-SNE and UMAP, this is the actual projection time, as described earlier in this section. Being GPU-accelerated neural networks, SSNP, AE, and NNP perform very fast, all being able to project up

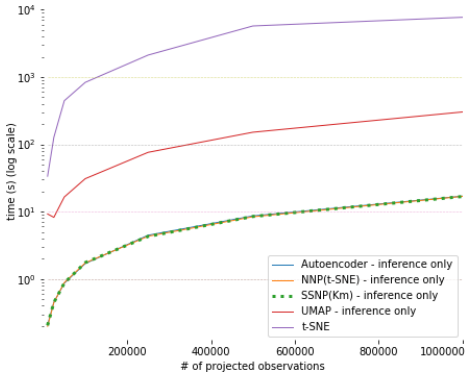


Fig. 10: Inference time for SSNP and other techniques (log scale). Techniques using training use 10K samples from the MNIST dataset. Inference is done on MNIST upsampled up to 1M samples.

to 1M samples in a few seconds – an order of magnitude faster than UMAP, and over three orders of magnitude faster than t-SNE. We also see that SSNP, AE, and NNP have practically the same speed. This is expected since they have comparably large and similar-architecture neural networks which, after training, take the same time to execute their inference.

5.6 Inverse Projection

Recalling from Sec. 2, an *inverse* projection $P^{-1}(\mathbf{p})$ aims to create a data point $\mathbf{x}$ so that its projection $P(\mathbf{x})$ is as close as possible to $\mathbf{p}$. Hence, we can test how well a method computes P^{-1} for a given direct projection function P by evaluating how close $P^{-1}(P(\mathbf{x}))$ is to the data point $\mathbf{x}$ itself. To test this, we consider points $\mathbf{x}$ being images in the MNIST dataset and P and P^{-1} being computed by SSNP as described in Sec. 3).

Figure 11 shows a set of digits from the MNIST dataset – both the actual images $\mathbf{x}$ and the ones obtained by $P^{-1}(P(\mathbf{x}))$. We see that SSNP(Km) yields results very similar to AE, both of these being visually quite close images to the actual images $\mathbf{x}$, modulo a limited amount of fuzziness. Hence, SSNP's dual-optimization target succeeds in learning a good inverse mapping based on the direct mapping given by the pseudo-labels (Sec. 3). Table 10 strengthens this insight by showing the values of the Mean Squared Error (MSE) between the original and inversely-projected images $\frac{1}{|D|}\sum_{\mathbf{x}\in D}\|\mathbf{x}-P^{-1}(P(\mathbf{x}))\|^2$ for SSNP(Km) and AE for both the training and test sets. These errors, again, are very similar. Furthermore, the SSNP MSE errors are of the same order of magnitude – that is, very small – as those obtained by the recent NNInv technique [13] and the older iLAMP [1] technique that also compute inverse projections – compare Tab. 10 with Fig. 2 in [13] (not included here for space reasons). Summarizing the above, we conclude that SSNP achieves a quality of inverse projections on par with existing state-of-the-art techniques.

Table 10: Inverse projection Mean Square Error (MSE) for SSNP(Km) and AE, trained with 5K samples and tested with 1K samples, different datasets.

	SSNP(Km)		Autoencoder	
Dataset	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

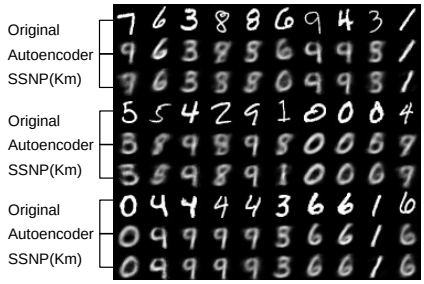


Fig. 11: Sample images from MNIST inversely projected by SSNP and AE, both trained with 10 epochs and 5K samples, MNIST dataset. Bright images show the original images that the inverse projection should be able to reproduce.

5.7 Data clustering

Table 11 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 (GT) labels and, respectively, the underlying clustering algorithm K-means (Km), which emerged as one of the best clustering companions for SSNP (Sec. 5.2). However, we should stress that classification or clustering is only a *side* result of SSNP, needed for computing the dual-objective cost that the network uses (Sec. 3). While one gets this by-product for free, SSNP only *mimics* the underlying clustering algorithm that it learns, rather than doing data clustering from scratch. As such, we do not advocate using SSNP as a potential replacement for clustering algorithms.

Table 11: Classification/clustering accuracy of SSNP when compared to ground truth (GT) and clustering labels (Km), trained with 5K samples, tested with 1K samples.

	SSNP(GT)		SSNP(Km)	
Dataset	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

5.8 Implementation details

All experiments discussed in this section 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 12 lists all open-source software libraries used to build SSNP and the other tested techniques. Our neural network implementations leverages the GPU power by using the Tensorflow Keras framework. The t-SNE implementation used is a parallel version of Barnes-Hut t-SNE [52,31], 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. The implementation of MDS, Isomap, and all clustering techniques comes from Scikit-Learn [41]. Our implementation, plus all code used in this experiment, are publicly available at `https://github.com/mespadoto/ssnp`.

Table 12: Software used for the SSNP implementation and evaluation.

Technique	Software used publicly available at
SSNP (our technique) Autoencoders	keras.io (TensorFlow backend)
t-SNE	github.com/DmitryUlyanov/Multicore-t-SNE
UMAP	github.com/lmcinnes/umap
Affinity Propagation Agglomerative Clustering DBSCAN Gaussian Mixture Model K-means Spectral Clustering	scikit-learn.org

6 DISCUSSION

We discuss next how the available hyperparameter settings influence the performance of SSNP with respect to the seven criteria laid out in Sec. 1.

Quality (C1): As shown in Figures 3 and 4, SSNP provides better cluster separation than Autoencoders, MDS, and Isomap, and comparable quality to t-SNE and UMAP, as measured by the selected metrics (Tables 5 and 6). Interestingly, using ground-truth labels (SSNP(GT)) does not always yield the highest quality metrics as compared to using pseudo-labels produced by clustering. Related to the latter, K-means (Km) and Agglomerative clustering (AG) yield, overall, higher quality metrics for most tested datasets as compared to DBSCAN, Gaussian mixture models, Spectral clustering, and Affinity propagation. When we consider the neighborhood hit (NH) metric, which models the closest from all studied metrics the ability of a projection to segregate similar samples into visually distinct clusters, SSNP(GT) performs better than all tested methods, t-SNE and UMAP included. Importantly, note that SSNP uses labels only during training and *not* during inference, so it can be fairly compared with such other projection methods.

Scalability (C2): SSNP(GT) is roughly half the speed of Autoencoders during training which is expected given its dual-optimization target. Training SSNP with pseudo-labels

is slower, roughly the speed of t-SNE or UMAP, which is explained by the time taken by the underlying clustering algorithm which dominates the actual training time. 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 practically identical to Autoencoders and NNP, and one order of magnitude faster than UMAP and three orders faster than t-SNE, being also linear in the sample and dimension counts. This shows SSNP's suitability to situations 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. Furthermore, we examined several hyperparameters of SSNP and found good default values (in terms of obtaining high quality metrics) as follows: no L2 regularization, tanh activation function for the embedding layer, and He uniform weight initialization. The clustering algorithm default is K-means or Agglomerative, with K-means slightly preferred for speed reasons. As such, SSNP can be used with no parameter tweaking efforts needed.

Genericity (C4): We show results for SSNP with different types of high-dimensional data, namely tabular (HAR), images (MNIST, FashionMNIST), and text (Reuters). As these datasets come from quite different sources and as the SSNP method itself makes no assumption on the nature or structure of the data, we believe that SSNP is generically applicable to any high-dimensional real-valued dataset.

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, in contrast to t-SNE and other non-parametric methods.

Inverse mapping (C6): SSNP shows inverse mapping results which are, quality-wise, very close to results from Autoencoders, NNInv and iLAMP, these being state-of-the-art methods for computing inverse projections. Additionally, SSNP computes the inverse projection at no extra cost or need for a separate implementation, in contrast to NNInv and iLAMP.

Clustering (C7): SSNP is able to mimic the behavior of the clustering algorithm used as its input, as a byproduct of its training with labeled data. 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. However, we stress that SSNP should not be considered as a replacement for state-of-the-art clustering algorithms, since it only learns to *mimic* the actual clustering. This is similar to the distinction between a classifier and an actual clustering technique.

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

Limitations: While scoring high on several criteria, SSNP also has several limitations. Quality-wise, its operation in pseudo-labeling mode cannot reach the high quality values for all metrics that are delivered by t-SNE or UMAP for challenging datasets (Tab. 6). We believe that this is affected by the number of clusters used during training, which is related to the neighborhood size that t-SNE and UMAP use. More involved strategies in setting this number of clusters can be explored to further increase SSNP's quality. Visually, while we argue for the reason of the star-shaped cluster structures produced by SSNP (Sec. 5.2), such patterns can be less suitable for visual exploration than the blob-like patterns produced typically by t-SNE. Using a tanh activation function partially alleviates this issue (Sec. 5.4). However, more studies are needed to explore other activation functions that allow even better control of the visual cluster shapes. Most importantly however, SSNP is a *learning* method. As with any such method, its quality will decrease when inferring on (that is, projecting) datasets which are too far away from the ones used during training, an issue also present for NNP and autoencoders. In contrast, methods that do not use training can obtain similar quality for any input dataset. Yet, the price to pay for such methods is that they cannot guarantee stability and out-of-sample behavior, which come with SSNP by default.

7 CONCLUSION

We presented an in-depth analysis of a dimensionality reduction (DR) method called Self-Supervised Neural Projection (SSNP) recently proposed by us. SSNP uses a neural network with a dual objective – reconstruction of the high-dimensional input data and classification of the data – to achieve several desirable characteristics of a general-purpose DR method. 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.

Our evaluation added two additional dimensionality reduction methods, four clustering algorithms, and also explored the hyperparameter space of both the clustering algorithms and neural network training to gauge SSNP's behavior. The evaluation results led to establishing default values for all these hyperparameters which obtain high quality values and also turn SSNP into a parameter-free method. Additionally, the obtained results show that SSNP with ground-truth labels yields higher quality in terms of visual cluster separation than all tested projections including the state-of-the-art t-SNE and UMAP methods. When pseudo-labels are used due to the lack of true labels, SSNP achieves lower but still competitive results with t-SNE and UMAP, slightly to significantly higher quality than autoencoders, and significantly higher quality than MDS and Isomap.

As future work, we consider studying better heuristics for controlling the clustering process which we believe are a low hanging fruit towards improving SSNP's quality. Another interesting direction is to explore other activation function designs that can offer control to the end users on the shape of the visual clusters that the projection

creates, which would be, to our knowledge, an unique feature in the family of projection techniques. 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.

ACKNOWLEDGMENTS

This study was financed in part by FAPESP grants 2015/22308-2, 2017/25835-9 and 2020/13275-1, and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Amorim, E., Brazil, E.V., Daniels, J., Joia, P., Nonato, L.G., Sousa, M.C.: iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In: Proc. IEEE VAST. pp. 53–62 (2012)
2. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: Proc. Intl. Workshop on Ambient Assisted Living. pp. 216–223. Springer (2012)
3. Becker, M., Lippel, J., Stuhlsatz, A., Zielke, T.: Robust dimensionality reduction for data visualization with deep neural networks. Graphical Models **108**, 101060 (2020)
4. 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. pp. 330–338 (2018)
5. Cunningham, J., Ghahramani, Z.: Linear dimensionality reduction: Survey, insights, and generalizations. JMLR **16**, 2859–2900 (2015)
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Methodological) **39**(1), 1–22 (1977)
7. Donoho, D.L., Grimes, C.: Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proc Natl Acad Sci **100**(10), 5591–5596 (2003)
8. Engel, D., Hattenberger, L., Hamann, B.: A survey of dimension reduction methods for high-dimensional data analysis and visualization. In: Proc. IRTG Workshop. vol. 27, pp. 135–149. Schloss Dagstuhl (2012)
9. Espadoto, M., Falcao, A., Hirata, N., Telea, A.: Improving neural network-based multidimensional projections. In: Proc. IVAPP (2020)
10. Espadoto, M., Hirata, N., Telea, A.: Deep learning multidimensional projections. J. Information Visualization (2020), doi.org/10.1177/1473871620909485
11. Espadoto, M., Hirata, N.S., Telea, A.C.: Self-supervised dimensionality reduction with neural networks and pseudo-labeling. In: Proc. IVAPP. pp. 27–37. SCITEPRESS (2021)
12. Espadoto, M., Martins, R.M., Kerren, A., Hirata, N.S., Telea, A.C.: Towards a quantitative survey of dimension reduction techniques. IEEE TVCG (2019), publisher: IEEE
13. Espadoto, M., Rodrigues, F.C.M., Hirata, N.S.T., Hirata Jr., R., Telea, A.C.: Deep learning inverse multidimensional projections. In: Proc. EuroVA. Eurographics (2019)
14. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDD. vol. 96, pp. 226–231 (1996)
15. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of eugenics **7**(2), 179–188 (1936)

16. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**(5814), 972–976 (2007)
17. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proc. AISTATS. pp. 249–256 (2010)
18. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proc. IEEE ICCV. pp. 1026–1034 (2015)
19. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006), publisher: AAAS
20. Hoffman, P., Grinstein, G.: A survey of visualizations for high-dimensional data mining. Information Visualization in Data Mining and Knowledge Discovery **104**, 47–82 (2002), publisher: Morgan Kaufmann
21. Joia, P., Coimbra, D., Cuminato, J.A., Paulovich, F.V., Nonato, L.G.: Local affine multidimensional projection. IEEE TVCG **17**(12), 2563–2571 (2011)
22. Jolliffe, I.T.: Principal component analysis and factor analysis. In: Principal Component Analysis, pp. 115–128. Springer (1986)
23. Kaufman, L., Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley (2005)
24. Kehrer, J., Hauser, H.: Visualization and visual analysis of multifaceted scientific data: A survey. IEEE TVCG **19**(3), 495–513 (2013)
25. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. CoRR **abs/1312.6114** (2013), eprint: 1312.6114
26. Kohonen, T.: Self-organizing Maps. Springer (1997)
27. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Proc. NIPS. pp. 950–957 (1992)
28. LeCun, Y., Cortes, C.: MNIST handwritten digits dataset (2010), `http://yann.lecun.com/exdb/mnist`
29. Liu, S., Maljovec, D., Wang, B., Bremer, P.T., Pascucci, V.: Visualizing high-dimensional data: Advances in the past decade. IEEE TVCG **23**(3), 1249–1268 (2015)
30. Lloyd, S.: Least squares quantization in PCM. IEEE Trans Inf Theor **28**(2), 129–137 (1982)
31. Maaten, L.v.d.: Barnes-hut-SNE. arXiv preprint arXiv:1301.3342 (2013)
32. Maaten, L.v.d.: Accelerating t-SNE using tree-based algorithms. JMLR **15**, 3221–3245 (2014)
33. Maaten, L.v.d., Hinton, G.: Visualizing data using t-SNE. JMLR **9**, 2579–2605 (2008)
34. Maaten, L.v.d., Postma, E.: Dimensionality reduction: A comparative review. Tech. rep., Tilburg University, Netherlands (2009)
35. Martins, R.M., Minghim, R., Telea, A.C., others: Explaining neighborhood preservation for multidimensional projections. In: CGVC. pp. 7–14 (2015)
36. McInnes, L., Healy, J.: UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426v1 [stat.ML] (2018)
37. Modrakowski, T.S., Espadoto, M., Falcão, A.X., Hirata, N.S.T., Telea, A.: Improving deep learning projections by neighborhood analysis. In: Communication in Computer and Information Science. Springer (2020)
38. 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
39. Paulovich, F.V., Minghim, R.: Text map explorer: a tool to create and explore document maps. In: Proc. Intl. Conference on Information Visualisation (IV). pp. 245–251. IEEE (2006)
40. Paulovich, F.V., Nonato, L.G., Minghim, R., Levkowitz, H.: Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. IEEE TVCG **14**(3), 564–575 (2008)

41. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research (JMLR) **12**, 2825–2830 (2011)
42. Pezzotti, N., Höllt, T., Lelieveldt, B., Eisemann, E., Vilanova, A.: Hierarchical stochastic neighbor embedding. Computer Graphics Forum **35**(3), 21–30 (2016)
43. 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 **23**, 1739–1752 (2017)
44. 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 **26**(1), 1172–1181 (2020)
45. Roweis, S.T., Saul, L.L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000), publisher: American Association for the Advancement of Science
46. Salton, G., McGill, M.J.: Introduction to modern information retrieval. McGraw-Hill (1986)
47. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE TPAMI **22**(8), 888–905 (2000)
48. Sorzano, C., Vargas, J., Pascual-Montano, A.: A survey of dimensionality reduction techniques (2014), arXiv:1403.2877 [stat.ML]
49. Tenenbaum, J.B., Silva, V.D., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
50. Thoma, M.: The Reuters dataset (Jul 2017), `https://martin-thoma.com/nlp-reuters`
51. Torgerson, W.S.: Theory and Methods of Scaling. Wiley (1958)
52. Ulyanov, D.: Multicore-TSNE (2016), `https://github.com/DmitryUlyanov/Multicore-TSNE`
53. Venna, J., Kaski, S.: Visualizing gene interaction graphs with local multidimensional scaling. In: Proc. ESANN. pp. 557–562 (2006)
54. Wattenberg, M.: How to use t-SNE effectively (2016), `https://distill.pub/2016/misread-tsne`
55. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms (2017), arXiv:1708.07747
56. Xie, H., Li, J., Xue, H.: A survey of dimensionality reduction techniques based on random projection (2017), arXiv:1706.04371 [cs.LG]
57. Zhang, Z., Wang, J.: MLLE: Modified locally linear embedding using multiple weights. In: Advances in Neural Information Processing Systems (NIPS). pp. 1593–1600 (2007)
58. Zhang, Z., Zha, H.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM Journal on Scientific Computing **26**(1), 313–338 (2004)