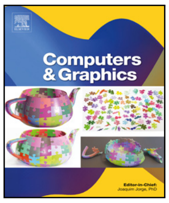
Computers & Graphics 124 (2024) 104093

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag



Special Section on EuroVA 2023

Controlling the scatterplot shapes of 2D and 3D multidimensional projections

Alister Machado [*], Alexandru Telea, Michael Behrisch

Department of Information and Computing Sciences, Utrecht University, 3584 CS Utrecht, The Netherlands

ARTICLE INFO

Keywords:
Dimensionality reduction
Data visualization
Projection

ABSTRACT

Multidimensional projections are effective techniques for depicting high-dimensional data. The point patterns created by such techniques, or a technique's *visual signature*, depend — apart from the data themselves — on the technique design and its parameter settings. Controlling such visual signatures — something that only few projections allow — can bring additional freedom for generating insightful depictions of the data. We present a novel projection technique — ShaRP — that allows explicit control on such visual signatures in terms of shapes of similar-value point clusters (settable to rectangles, triangles, ellipses, and convex polygons) and the projection space (2D or 3D Euclidean or $\mathcal{S}^2$). We show that ShaRP scales computationally well with dimensionality and dataset size, provides its signature-control by a small set of parameters, allows trading off projection quality to signature enforcement, and can be used to generate decision maps to explore the behavior of trained machine-learning classifiers.

1. Introduction

Exploring high-dimensional data is a key challenge in the fields of information visualization and visual analytics. This task is especially difficult for datasets having many samples, each in turn coming with many dimensions to be depicted. Dimensionality Reduction (DR), also called projection, techniques are one of the most popular tools for this task. Simply put, projections map a high-dimensional dataset to a low-dimensional (typically 2D or 3D) scatterplot, which is next visualized. Projections scale well both computationally and visually with both the sample and dimension counts. Many projection techniques have been proposed in the last decades, with various approaches to tackle computational scalability, quality of the produced scatterplots, robustness to data noise, and ease of use [1–7].

When constructing the low-dimensional scatterplot, projection techniques aim to preserve as much as possible of the so-called *data structure* [8]. That is, *data* elements such as compact groups of samples, outlier samples, or areas having different sample densities, are mapped to analogous *visual* elements in the low-dimensional space. This allows users to search and inspect the low-dimensional space for the presence of such visual elements. When these are found in the scatterplot, the presence of their high-dimensional data counterparts can be inferred. Prior research has shown that exploring scatterplots to locate such visual patterns can help uncovering topological aspects, such as groupings, outliers, and correlations in the data [9–11].

However, no projection technique can perfectly map all possible data patterns to corresponding visual patterns for any high-dimensional dataset. Different projection techniques encode their mapping goals in a so-called cost function which is then minimized to construct the scatterplot. As such, the exact nature of the produced visual patterns depends not only on the underlying *data*, but also on how the DR technique is designed to minimize a specific cost function. For example, for the same dataset, t-SNE tends to create organic, round, structures; Auto-Encoders create starburst-like clusters; PCA tends to create elliptic patterns; ISOMAP [12] creates line-like structures; and UMAP creates very dense, round, clusters, separated by large amounts of whitespace — to mention just a few [7,8]. We further call such aspects that a projection technique tends to generate, regardless of the dataset it visualizes, the *visual signature* of that projection technique.

We believe that users can benefit from having *direct* control over the visual signatures of a projection technique for several reasons. Firstly, the possibility of *picking* a visual signature for a projection gives users *direct* control over what is thus far an emergent property implicitly determined by a DR algorithm's design and not necessarily a portrayal of true data patterns. For example, assume we see elliptical visual patterns (clusters) in a projection. With current projection techniques, we do not know if these reflect actual (elliptical) patterns in the *data* or they are produced by the projection technique. In contrast, our method allows controlling such patterns — so, when seeing them, we know that they are due to the projection and not the data; in other words, we argue that precise visual signature control is better than no control. To refine this example: In cases where users employ pseudolabels in their pipeline, outputting shapes that match the pseudolabeling

[*] Corresponding author.
E-mail addresses: a.machadodosreis@uu.nl (A. Machado), a.telea@uu.nl (A. Telea), m.behrisch@uu.nl (M. Behrisch).

https://doi.org/10.1016/j.cag.2024.104093
Received 29 November 2023; Received in revised form 24 August 2024; Accepted 22 September 2024
Available online 24 September 2024

algorithm's assumptions would ensure a "match" between high and low dimensional space — for instance, producing round shapes when using k-means, which assumes isotropic (Gaussian clusters). Secondly, controlling the generated visual patterns can help in creating 2D or 3D scatterplots which better match a given interactive exploration task at hand. For example, when performing interactive data labeling using rectangular selections or displaying image thumbnails over data clusters (see Fig. 3), a projection whose clusters resemble rectangles would be more suitable than one creating various-shaped clusters (modulo other aspects of the two projections, *e.g.*, quality).

However, controlling such visual signatures is typically hard with current projection methods, as these are designed around cost functions which typically do not incorporate measures on, or controls of, the visual patterns they create. Recently, the **Sha**pe **R**egularized Neural **P**rojection (ShaRP) technique was proposed to fill this gap [13]. ShaRP achieves such control by extending the earlier autoencoder-based SSNP projection technique [14] by a variational autoencoder that allows constraining the shapes of created visual patterns. ShaRP scales well with both dimension and sample count, works both in a supervised way (by class labels) and self-supervised way (by pseudolabels created by data clustering), generically handles any quantitative high-dimensional dataset, and is simple to implement. It also provides, by design, an inverse projection mapping useful for interpolating new data space points and generating Decision Boundary Maps [15] (see Section 4.5), as well as a mechanism to project data into the non-Euclidean space $\mathcal{S}^2$, the surface of a 3D ball, providing an entirely novel view into the data.

In this paper, we extend the proposal of ShaRP in several directions, as follows.

- We introduce support for additional visual signatures, allowing the creation of 3D projection scatterplots with better properties than current state of the art methods can do;
- We extend ShaRP to create decision maps for visual exploration of classification models;
- We present additional validation of ShaRP by including more datasets to compare on.

The structure of this paper is as follows. Section 2 presents related work covering projection techniques and measuring and controlling their visual signatures. Section 3 details the ShaRP technique, including our proposed extensions. Section 4 evaluates ShaRP by studying the effect of its hyperparameters and also comparing its results with other well-known projection techniques on a variety of datasets. Section 5 discusses our proposal. Finally, Section 6 concludes with future work directions.

2. Related work

2.1. Notations

A dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1,\ldots,m}$ has m samples $\mathbf{x}_i = [x_{i1},\ldots,x_{in}]^T$, where $\mathbf{x}_i$ is a point in $\mathbb{R}^n$ with components x_{ij}, $1 \le j \le n$ and an optional label $y_i \in \{1,\ldots,K\}$. In the following, we use capitals to denote the set of all elements for the corresponding small letter, *e.g.*, $\bar{Y} = \{\bar{y}_i\}_{i=1,\ldots,m}$ is the set of (pseudo-)labels used for training. Variables marked with a hat (e.g., $\hat{\mathbf{X}}, \hat{Y}$) represent predictions as produced by a classifier or regressor. We denote the Euclidean norm by $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$ and the expected value of a function of a random variable $\mathbf{z}$ distributed according to p by $\mathbb{E}_{\mathbf{z}\sim p}[f(\mathbf{z})]$. We use θ to denote probability distribution parameters, for example $\theta = (\vec{\mu} \in \mathbb{R}^2, \vec{\sigma}^2 \in \mathbb{R}^2)$ for a 2D Diagonal Gaussian distribution.

2.2. Dimensionality reduction

Projection algorithms can be defined as functions

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

where $q \ll n$ is the dimension of the created scatterplot (typically $q \in \{2,3\}$) and η denote P's (hyper)parameters. Hence, for a sample $\mathbf{x} \in \mathbb{R}^n$, $P(\mathbf{x})$ denotes its mapping to the q-dimensional space. We denote by $P(\mathbf{X})$ the entire scatterplot created by a given technique P applied to all samples of a given dataset $\mathbf{X}$.

Many projection algorithms are available nowadays. These are described from technical perspectives (how they differ design-wise) in several surveys [1–6,16]; from the perspective of how they perform with respect to several quality metrics [7]; and based on the tasks that one aims to use them for [8]. In particular, autoencoder techniques, which share structurally some similarities to our approach, can be used to this end [17]. However, such techniques have not been deemed so far better than other non-autoencoder projection methods [7]. Examples of well-known projection techniques are Principal Component Analysis (PCA) — a simple, easy to code, but qualitatively limited method especially for complex non-planar data structures embedded in high dimensions [18]; Isomap — a technique which works well if the data resides on a (single) high-dimensional manifold [12]; t-SNE, which works well for arbitrary high-dimensional data distributions but has challenges in controlling (and predicting) the shapes of the emerging visual clusters [19–21]; and UMAP, similar to t-SNE in terms of ease and of visual cluster control [22].

2.3. Inverse projections

The ability to construct a so-called 'direct' projection mapping P (Eq. (1)) raises the question whether constructing an inverse mapping

$$P^{-1} : \mathbb{R}^q \to \mathbb{R}^n \tag{2}$$

is possible and, if so, useful. Recent work has shown that such inverse mappings can be constructed by minimizing the cost $\|P^{-1}(P(\mathbf{x})) - \mathbf{x}\|, \mathbf{x} \in \mathbf{X}$ for a given dataset $\mathbf{X}$, by using methods such as radial basis functions [23], feedforward neural networks [24], or autoencoder-based strategies [25,26]. It is important to note that inverse projections P^{-1} are not the mathematical inverse function of the projection function P, as P is often not injective and, therefore, not invertible [27]. Rather, P^{-1} are approximate, regularized, forms of such inverse functions. Inverse projections have many practical uses, such as shape and image morphing [23], data imputation [28], and constructing so-called decision maps to visualize the behavior of trained classification models [15,26,29–31]. We show further in Section 4.5 how our proposed projection method can be used to construct such decision maps.

2.4. Decision boundary maps

Projections aim to preserve data similarities in the data space by the 2D scatterplots they construct, so they can be used to assess the classification difficulty of a dataset. If points from different classes cannot be easily disentangled by a projection algorithm — leading to good visual separation in the projection plot — that might mean that a classifier will struggle with achieving good accuracy for that dataset [32].

More recently, dense 2D visualizations of classifier behavior in high-dimensional space named Decision Boundary Maps (DBM [15]) have been proposed. These map-like visualizations densely portray the partitioning effected by a classifier on the high dimensional space. Such dense maps are created using inverse projections — every pixel of a 2D image is invertex by P^{-1}, then classified by the model under study, and finally colored to show the respective class in the 2D image. Approaches to create these inverse projections vary: some DBM generation algorithms are projection-agnostic, *i.e.*, they can work atop any given projection P [15]. Others jointly learn a projection and its inverse, such as SDBM [26]. Alternatively, the invertibility of a projection algorithm can be used to draw a DBM, as done by Schulz et al. [31] who use a supervised version of UMAP.

2.5. Quality metrics

As outlined in Section 1, projections are computed by optimizing given cost functions that reflect which data patterns the mapping P should preserve. Hence, a technique P can be seen as a cost function (to optimize) plus an optimization algorithm. Such cost functions lead next to various quality metrics which express the ability of a computed $P(\mathbf{X})$ to preserve specific patterns of a given dataset $\mathbf{X}$. Formally put, a quality metric is a function

$$\begin{aligned} Q_P &: \mathcal{X} \times \mathcal{P}^q \to \mathbb{R}^+ \\ &(\mathbf{X}, P(\mathbf{X})) \mapsto q \end{aligned} \tag{3}$$

that maps a dataset $\mathbf{X}$ and its projection scatterplot $P(\mathbf{X})$ computed by a given technique P to a real value. Quality metrics can be computed at different *scales* to gauge different patterns present in a projection. Four such scales exist, as follows. For each scale, we discuss why its metrics cannot be used to directly capture our notion of visual signature of a projection.

Point-pairs: At the lowest level, one can quantify how well each point-pair $(\mathbf{x}_i, \mathbf{x}_j)$ from a dataset $\mathbf{X}$ is mapped to the corresponding scatterplot point-pair $(P(\mathbf{x}_i), P(\mathbf{x}_j))$. The simplest way to gauge this is to measure the so-called normalized stress

$$Q_S = \sqrt{\frac{\sum_{i,j} \left(\|\mathbf{x}_i - \mathbf{x}_j\| - \|P(\mathbf{x}_i) - P(\mathbf{x}_j)\| \right)^2}{\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|^2}} \tag{4}$$

which tells how well *distances* in $\mathbf{X}$ are preserved by $P(\mathbf{X})$. Techniques such as MDS [33] and its variants directly aim to optimize Q_S. For our goal, point-pair metrics are too fine-grained to capture visual signatures — as any such signature, obviously, involves patterns formed by more than two points.

Neighborhoods: At the next level, one can quantify how well a projection maps a given neighborhood v of points in $\mathbf{X}$. Let, next, v_i denote the k nearest neighbors of a given sample $\mathbf{x}_i \in \mathbf{X}$. Trustworthiness [34] computes how far the points in $P(v_i)$ are from the K nearest neighbors of $P(\mathbf{x}_i)$ — or, simply put, how closely-projected are the points of v_i, as

$$Q_T = 1 - \sum_{\mathbf{y} \in v_i^q} \left(r_i(\mathbf{y}) - k \right), \tag{5}$$

where v_i^q are the k-nearest neighbors of $P(\mathbf{x}_i)$ which do not map points in v_i; and $r_i(\mathbf{y})$ is the rank of the 2D point $\mathbf{y}$ in the ordered set v_i^2 of nearest neighbors of $P(\mathbf{x}_i)$. False neighbors [35] provide a similar measure. Symmetric to the above, continuity [34] measures the fraction of points close in a 2D neighborhood in $P(\mathbf{X})$ that come from data points which are also close in $\mathbf{X}$, as

$$Q_C = 1 - \sum_{\mathbf{z} \in v_i^n} \left(\hat{r}_i(\mathbf{z}) - k \right), \tag{6}$$

where v_i^n are the k-nearest neighbors of $\mathbf{x}_i$ which do not map to points in the k-nearest neighbors of $P(\mathbf{x}_i)$; and $\hat{r}_i(\mathbf{z})$ is the rank of the data point $\mathbf{z}$ in the ordered set v_i^n of nearest neighbors of $\mathbf{x}_i$. Missing neighbors [35] provide a similar measurement.

For labeled datasets, the neighborhood hit [36] measures the fraction of the k-nearest neighbors of a projection point $\mathbf{y}$ that have the same label as the point $\mathbf{y}$ itself, as

$$Q_N = \frac{|\mathbf{z} \in v_i^q \, : \, l(\mathbf{z}) = l(\mathbf{y})|}{k}, \tag{7}$$

where $l(\mathbf{y})$ denotes the label of a scatterplot point $\mathbf{y}$.

Other neighborhood-level metrics include the projection precision score [37] which measures how the distances from a data point to its k-nearest neighbors differ from the analogous distances between their scatterplot points (thus, a neighborhood-level generalization of Q_S); stretching and compression [38,39], which are variants of trustworthiness and continuity; and the average local error [35], which also generalizes Q_S to neighborhoods. The Kullback–Leibler divergence used as cost function by the by-now famous t-SNE technique [19] to compare a high-dimensional with a low-dimensional neighborhood is another neighborhood metric.

Neighborhood metrics help visually exploring projections to *e.g.* find high-quality areas (that can be reliably interpreted) and low-quality areas (which can be misleading). However, they cannot directly capture visual signatures, since they are heavily influenced by their size, given by the parameter k or, for t-SNE, its perplexity parameter. Setting k to different values will yield different quality judgments on the *same* projection [40]. In other words, neighborhood metrics only measure patterns at a given fixed scale k. More importantly, k-nearest neighborhoods can be only used to model 'circular' patterns based on the distance to a central point. Patterns such as Isomap's line-like structures, or autoencoders' starburst-like shapes (mentioned in Section 1), cannot be in general modeled by k-nearest neighbors.

Classes: One level higher than k-nearest neighborhoods, one can quantify projections of labeled data at the *class* level. Visual separation metrics [41–44] gauge how well points from each class, in a projection, are visually separated from points of other classes. The underlying idea is that ground-truth information on the class separation in $\mathbf{X}$ can be used to compute how well $P(\mathbf{X})$ reflects that separation. Sedlmair and Aupetit [45] surveyed of 15 such metrics. Their conclusion is that the distance consistency metric (DSC, [46]), also presented as class consistency measure (CCM, [47]), best encodes the perception of class separation in labeled scatterplots. DSC is computed assuming that, in a well-separated labeled dataset D, a point with label l is closest to the centroid (barycenter) of all points with the same label l (that is, the centroid of class l), than to centroids of other classes. A good projection should keep this property on $P(\mathbf{X})$. DSC is computed as

$$Q_{DSC} = \frac{\left| \{ \mathbf{y} \in P(\mathbf{X}) \, : \, l(\mathbf{y}) = \arg\min_{l' \in Y} \|\mathbf{y} - c(l')\| \} \right|}{m}, \tag{8}$$

where $c(y) = |P_y(\mathbf{X})|^{-1} \sum_{\mathbf{y} \in P_y(\mathbf{X})} \mathbf{y}$ denotes the centroid of all projection points having class y.

Visual (class) separation metrics are very useful in assessing the usability of a projection scatterplot for machine-learning-related tasks such as finding how well are classes separated in a dataset. In turn, this can predict the ease of classification of that dataset via its projection [32]. However, such metrics are not useful for our task of assessing (and next, controlling) the visual patterns created by a projection technique. Moreover, such metrics are only applicable to labeled data.

Scatterplot: At the highest level, one can compute quality metrics for an entire projection scatterplot. This is the most frequently used deployment of quality metrics since it allows easy comparison of multiple scatterplots by a single, aggregated, value. All point-pair, neighborhood, and class-level metrics can be aggregated at scatterplot level. Apart from this, separated metrics exist that gauge an entire scatterplot. For example, the Shepard diagram [48] computes a scatterplot of data-distances *vs* scatterplot-distances over all point pairs in $\mathbf{X}$, respectively $P(\mathbf{X})$. The diagram can be reduced to a single quality value, called the Shepard goodness, by computing its Spearman rank correlation, thus capturing how well a projection preserves distances (generalizing Q_S) for an entire scatterplot (see Q_R, Table 1).

However, as all earlier discussed metrics, scatterplot-level metrics cannot capture the *patterns* that create a projection's visual signature. Still, such metrics are useful to quickly compare the quality of different projections, so we will use them to this end in our evaluation (Section 4).

Table 1
Projection quality metrics used in this paper. All metrics range over $[0 = \text{worst}, 1 = \text{best}]$ or are linearly mapped thereto.

Metric	Definition
Trustworthiness (Q_T)	$1 - \frac{2}{mk(2m-3k-1)} \sum_{i=1}^{m} \sum_{\mathbf{y}\in\nu_i^2} (r_i(\mathbf{y}) - k)$
Continuity (Q_C)	$1 - \frac{2}{mk(2m-3k-1)} \sum_{i=1}^{m} \sum_{\mathbf{z}\in\nu_i^n} (\hat{r}_i(\mathbf{z}) - k)$
Neighborhood hit (Q_N)	$\sum_{i=1}^{m} \frac{\lvert \mathbf{z}\in\nu_i^2 \ : \ l(\mathbf{z})=l(\mathbf{y})\rvert}{mk}$
Shepard goodness (Q_R)	Spearman rank of scatterplot $\{(\lVert\mathbf{x}_i - \mathbf{x}_j\rVert, \lVert P(\mathbf{x}_i) - P(\mathbf{x}_j)\rVert)\}, 1 \le i \le m, i \ne j$
Distance consistency (Q_{DSC})	$\frac{\left\lvert \{\mathbf{y}\in P(\mathbf{X}) : l(\mathbf{y})=\arg\min_{l'\in Y} \lVert\mathbf{y}-c(l')\rVert\}\right\rvert}{m}$

2.6. Visual signatures

Summarizing the above, we see that projection-quality metrics, while useful in gauging various desirable aspects of a projection at different scales, cannot really capture the patterns that constitute a projection's visual signature. As such, they cannot be used to design cost functions to actually *drive* a projection to generate specific visual signatures. However, several works have addressed the measurement — and only very partially, the control — of visual signatures of projections, as follows.

Initial work on perception-driven DR [11,49] aims to factor the humans' perceptual capabilities in the DR computation and quality assessment. Yet, developing quality metrics that target a variety of *visual patterns* is nontrivial, since these heuristics must simulate the humans' complex pattern recognition process. Moreover, even if we possessed such a visual quality metric, it is not evident what would be a suitable *ground truth* to compare it to, since that ground truth has to somehow measure the patterns present in the data. More importantly, our goal here is different: while perception-driven DR aims to measure how users *see* patterns in a projection scatterplot, we aim to allow users to *control* how that scatterplot will look like in terms of its patterns. As such, perception-driven DR research can come after ours — to assess how users perceive our pattern-controlled projections — but not precede or replace it.

Cutura et al. [50] use space-filling curves to adapt the position of data points in image thumbnail scatterplots such that they are non-overlapping. This can be seen as a local way to control the generation of visual patterns in a projection. Their idea is effective but limited to image datasets, while our proposed technique is designed to be generic.

Abbas et al. [51,52] proposed a visual quality metric that mimics the users' ability to distinguish one or more clusters in a scatterplot. The main differences of our work to this is that we (a) do not aim to propose a technique to separate clusters, but to enforce the *shapes* of such clusters; and (b) our ground truth is not user perception, but the data space.

The perplexity parameter in t-SNE controls, among other things, a projection's the visual appearance. Its effect is intricately enmeshed within t-SNE, leading to cluster shapes, sizes, and distances that do not necessarily convey meaning [53]. Our technique minimizes the variability of this visual appearance. Related to this, since fully doing away with hyperparameters might be infeasible, techniques such as HyperNP [54] learn to simulate their effects on the resulting projection. This helps users quickly scan different perplexity values to choose the one generating a projection with the desired visual signature. In contrast to HyperNP, our technique uses a *single* set of hyperparameters to generate user-chosen visual patterns, reducing the need for such costly simulations.

Finally, Makhzani et al. [55] propose an approach similar to ours. They adapt an auto-encoder into an adversarial setting to increase the quality of the projection (*e.g.*, better cluster separability) that the vanilla auto-encoder would create. Our method instead aims mainly to put cluster shapes under the user's control. In other words, they optimize a general-purpose quality metric, but do not control the visual patterns explicitly; we control these patterns explicitly, and measure a posteriori what the quality price incurred by the control is.

3. ShaRP: Shape-regularized neural projection

We now present ShaRP, our novel DR technique, which is based on deep neural networks. Such networks can, in general, approximate complex non-linear functions and have several desirable features that ShaRP inherits:

Scalable: ShaRP scales linearly in the sample count because it avoids precomputing pairwise distances or covariance matrices, like in PCA or t-SNE, and lends itself to hardware-acceleration through GPUs or TPUs tailored for fast deep learning.

Parametric: ShaRP operates in a "learn once, project as needed" fashion. It learns to parameterize a projection function instead of only outputting the projected points, such as t-SNE or UMAP. This allows ShaRP to project data it was not trained on along with existing data (out-of-sample ability).

Generic: ShaRP handles any dataset comprised of numeric features and can be applied to a wide range of datasets using the same or only slightly adapted hyperparameter settings.

Sound: ShaRP scores comparably to state of the art techniques in relevant projection quality metrics.

To these, ShaRP adds two flavors of **Shape Regularization**:

- Intra-projection: ShaRP creates point clusters having shapes coming from the same (user-controlled) *family*: ellipses, rectangles, triangles.
- Inter-projection: Running ShaRP over different datasets produces a consistent *visual signature* where differences in the projections are driven mainly by data patterns.

Invertible: Since ShaRP uses deep learning, it can be directly used to invert the projection mapping P to compute inverse projections P^{-1}. We show in Section 4.5 how we can use this ability to compute decision maps to explore the behavior of trained classification models.

ShaRP is implemented in Python using Keras (Tensorflow backend) [56], Tensorflow Probability [57] for sampling and calculating log-probabilities under different distributions. It is publicly available at https://github.com/amreis/sharp.

3.1. Method description

ShaRP belongs to the family of Representation Learning [58] techniques, *i.e.*, it learns a *latent encoding* for input data. A latent encoding is a vector $\mathbf{z} \in \mathbb{R}^q$, where $\mathbf{z} = f(\mathbf{x})$ is a low-dimensional representation of the input $\mathbf{x} \in \mathbb{R}^n$ that enables a reconstruction of $\mathbf{x}$ with minimal errors. As we aim to create 2D and 3D projections, $q \in \{2, 3\}$ in our case.

ShaRP builds atop of the recent DR method SSNP [14] (see next Fig. 1). SSNP extends a vanilla auto-encoder with loss $\mathcal{L}_{\text{AE}}$ with a classifier head (with an accompanying loss $\mathcal{L}_{\text{class}}$), yielding the total loss to be optimized as

$$\mathcal{L}_{\text{SSNP}}(\mathbf{X}, \hat{\mathbf{X}}, \bar{Y}, \hat{Y}) = \mathcal{L}_{\text{AE}}(\mathbf{X}, \hat{\mathbf{X}}) + \rho\mathcal{L}_{\text{class}}(\bar{Y}, \hat{Y}). \tag{9}$$

The projection $\mathbf{z}_i \in \mathbb{R}^q$ of each input $\mathbf{x}_i$ is generated by the bottleneck layer of the network. The classification loss $\mathcal{L}_{\text{class}}$, together with target

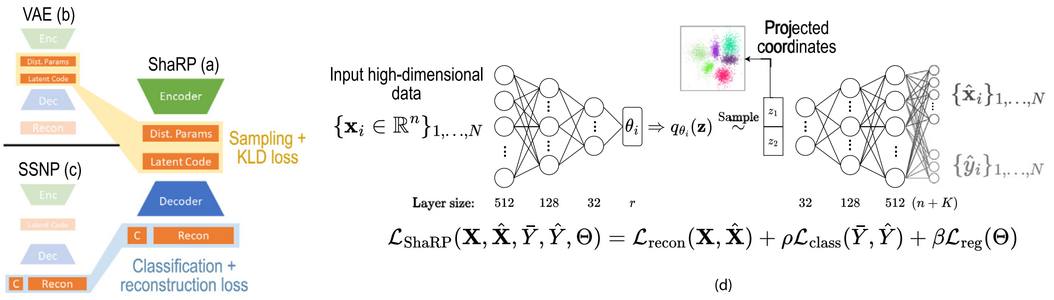

VAE (b)
Enc
Dist. Params
Latent Code
Dec
Recon
SSNP (c)
Enc
Latent Code
Dec
C Recon
ShaRP (a)
Encoder
Dist. Params
Latent Code
Sampling + KLD loss
Decoder
C Recon
Classification + reconstruction loss
Input high-dimensional data
$\{\mathbf{x}_i \in \mathbb{R}^n\}_{1,...,N}$
Projected coordinates
$\theta_i \Rightarrow q_{\theta_i}(\mathbf{z})$
Sample
z_1
z_2
$\{\hat{\mathbf{x}}_i\}_{1,...,N}$
$\{\hat{y}_i\}_{1,...,N}$
Layer size: 512 128 32 r 32 128 512 (n+K)
$\mathcal{L}_{\mathrm{ShaRP}}(\mathbf{X},\hat{\mathbf{X}},\bar{Y},\hat{Y},\Theta) = \mathcal{L}_{\mathrm{recon}}(\mathbf{X},\hat{\mathbf{X}}) + \rho\mathcal{L}_{\mathrm{class}}(\bar{Y},\hat{Y}) + \beta\mathcal{L}_{\mathrm{reg}}(\Theta)$
(d)

Fig. 1. Relation of ShaRP (a) with VAEs (b) and SSNP (c). Detailed architecture of ShaRP (d). See Section 3.1.

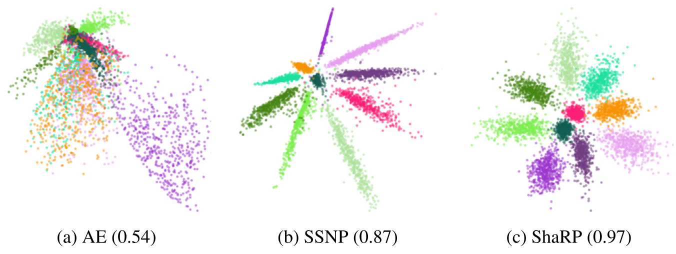


Fig. 2. Comparison of projections of the MNIST dataset learned using (a) Auto-encoders, (b) SSNP [14], and (c) ShaRP. SSNP and ShaRP were trained using the *ground truth* labels as class information — encoded, here and next, by colors. Values in brackets are Distance Consistency scores (higher is better, see Table 1).

Table 2
Overview of desirable properties in three strongly related techniques (VAE, SSNP, ShaRP), together with whether each technique fulfills the property completely (•), partially (∘), or does not fulfill it (×).

Property	VAE	SSNP	ShaRP (ours)
Projection-oriented	×	•	•
Can use (pseudo-)labels	×	•	•
DBM generation	×	•	•
Latent space regularization	•	×	•
Shape regularization	∘	×	•
Non-Euclidean projection	∘	×	•

labels or *pseudolabels* generated by a clustering algorithm, enables SSNP to separate data clusters better than plain auto-encoders. Fig. 2 illustrates this for the well-known MNIST dataset. Yet, as the figure also shows, SSNP collapses some clusters into elongated shapes, which we argue is (a) unnatural, as it suggests some anisotropy in the sample distribution; (b) space-inefficient, as much white space is not used to depict data; and (c) suboptimal for visualization as we cannot *e.g.* easily select a cluster by rubberband tools or efficiently annotate it with a rectangular icon.

ShaRP overcomes these shortcomings of SSNP by an explicit user-controlled shape regularization mechanism, described next (see also Section 3.2 for examples). ShaRP replaces SSNP's Auto-Encoder (AE) with a Variational AE (VAE) [59]. The key AE-VAE difference is the latter's use of a *sampling* process in the network's bottleneck layer. This, coupled with a necessary KL divergence (KLD) regularization term

$$\mathcal{L}_{\mathrm{reg}}(\theta) = D_{\mathrm{KL}}(q_\theta \parallel p) \doteq \mathbb{E}_{\mathbf{z}\sim q_\theta}[\log(q_\theta(\mathbf{z})/p(\mathbf{z}))] \tag{10}$$

has as an immediate effect on the regularization of the learned latent space: Using $\mathcal{L}_{\mathrm{reg}}$ pushes the learned probability distributions q_θ towards a standard form p defined a priori (e.g., a standard Gaussian distribution) which prevents learning degenerate distributions. Also, crucially for our goals, this loss can be *exploited* to model different shape regularization constraints (see next Section 3.2). Our complete loss function then reads as

$$\begin{aligned}\mathcal{L}_{\mathrm{ShaRP}}(\mathbf{X},\hat{\mathbf{X}},\bar{Y},\hat{Y},\Theta) =& \mathcal{L}_{\mathrm{recon}}(\mathbf{X},\hat{\mathbf{X}}) + \rho\mathcal{L}_{\mathrm{class}}(\bar{Y},\hat{Y}) + \beta\mathcal{L}_{\mathrm{reg}}(\Theta) \\ =& \mathcal{L}_{\mathrm{SSNP}}(\mathbf{X},\hat{\mathbf{X}},\bar{Y},\hat{Y}) + \beta\mathcal{L}_{\mathrm{reg}}(\Theta),\end{aligned} \tag{11}$$

where we make the connection to the SSNP loss explicit.

By using a suitable *sampling* process, the clusters emerging in the projection will be shape-regularized. For example, a 2D Gaussian sampling distribution yields *elliptical* shapes (see Fig. 2(c)) because the equidensity contours of a 2D Gaussian are ellipses. This is dependent on $\mathcal{L}_{\mathrm{reg}}$ preventing the degenerate learning of low (respectively, high) variances, which would give rise to point-like (resp., line-like) shapes in the projection.

Table 2 summarizes the differences between using a pure VAE, SSNP, and ShaRP. A VAE is good at producing a latent representation that enables downstream tasks, so its latent space usually has more than 2 or 3 dimensions and visualizing (projecting) it is not a main concern. This also means that, even though VAEs are *in theory* capable of shape regularization and non-Euclidean projection, they meet such goals only partial due to the aforementioned reason. SSNP is designed to generate projections, and has support for (self-)supervision through (pseudo-)labels. It performs no regularization on the learned latent space due to the use of a vanilla auto-encoder, which also means it is unable of shape regularization or non-Euclidean projections. ShaRP fulfills all of the listed properties. It generates projections and DBMs on a regularized latent space, allows for (self-)supervision, and has the extra capabilities of shape regularization and non-Euclidean projection.

3.2. Controlling cluster shapes

We use as regularization targets the following shapes.

Ellipses. Consider a diagonal Multivariate Normal distribution, *i.e.*, $\mathbf{z}_i \sim \mathcal{N}(\vec{\mu}, \mathrm{diag}(\vec{\sigma}^2))$. The natural prior to use here is the standard Multivariate Normal distribution $\mathcal{N}(\vec{\mathbf{0}}, \mathbf{I})$ which simplifies sampling, propagating gradients, and calculating the KLD loss [59]. By using this prior, we encourage learned probability distributions to be as close as possible to a standard Gaussian. Hence, the learned projection will output data clusters that resemble circles or ellipses (see next Fig. 8).

Using a Gaussian sampling distribution is standard for VAEs. For our projection goals, tweaking the sampling distribution and using suitable priors allows favoring different cluster shapes. We can use *any* distribution as long as we have (i) access to log-probabilities of samples under the learned distribution and the prior; (ii) a way to propagate gradients through the sampling process (using a reparametrization trick or otherwise).

Access to the log-probabilities of samples under learned distributions and the prior removes the (constraining) need to analytically calculate the KLD since we can re-express it as a sample-based computation as

$$D_{\mathrm{KL}}(q_\theta \parallel p) \approx \frac{1}{m}\sum_{i=1}^{m}\left(\log q_\theta(\mathbf{z}_i) - \log p(\mathbf{z}_i)\right), \tag{12}$$

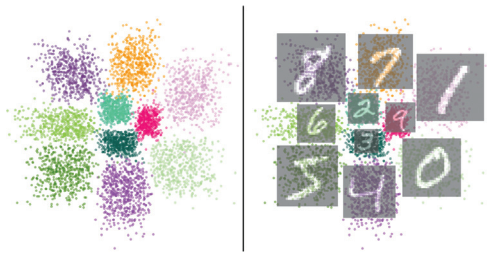


Fig. 3. Rectangular shaping can be convenient for data labeling tasks, as illustrated by the right image where class representatives are overlaid atop their clusters. We achieve this using a Generalized Normal distribution, here shown on the MNIST dataset for $\omega = 10$ (left).

where the approximation holds if $\mathbf{z}_i \sim q_\theta(\cdot)$. Our next examples of regularization shape targets use Eq. (12) for computing the (approximate) KLD.

Rectangles. To create rectangular clusters, we use a Generalized Normal ($\mathcal{GN}$) probability distribution. It introduces an additional shape parameter to the Gaussian (here denoted ω) and has a density function of the form

$$p(x|\mu,\alpha,\omega) \propto \exp\left(-(|x-\mu|/\alpha)^{\omega}\right).$$

Tuning ω makes the distribution tails heavier or lighter. This is akin to the Minkowski p-norm where $p \to \infty$ values (analogous to ω) make sets of equidistant points approach axis-aligned squares instead of circles ($p = 2$). Using this distribution for sampling, with a high ω, yields cluster shapes that resemble squares/rectangles instead of ellipses (see Figs. 3, 4).

Triangles. We first define an equilateral triangle in $\mathbb{R}^2$ by arranging its vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ in a matrix as

$$\mathbf{T} = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3] = \begin{bmatrix} 0 & 1/2 & 1 \\ 0 & \sqrt{3}/2 & 0 \end{bmatrix}.$$

To remove bias in the base shape, we use a base equilateral, unit-edge, triangle. The initial bias can be overcome through the training process via the following necessary extension that we add to the sampling scheme. Any convex combination of the barycentric coordinates $\mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{v}_3$ gives a point inside the triangle. Formally: for a vector $\mathbf{w} = [w_1\, w_2\, w_3]^T$ with $w_i \in [0,1]$, $1 \le i \le 3$ and $\sum_{i=1}^{3} w_i = 1$, $\mathbf{p} = \mathbf{Tw}$ is inside the triangle.

We now need to use a sampling distribution in ShaRP that generates vectors with the same properties as $\mathbf{w}$ above. The Dirichlet probability distribution

$$\mathbf{w} \sim \mathrm{Dir}(\alpha_1,\alpha_2,\alpha_3) \Rightarrow \mathbf{w} \in [0,1]^3, \quad \sum_{i=1}^{3} w_i = 1 \quad (\alpha_i > 0,\ \forall i)$$

does exactly that. We choose as prior the "uniform" distribution on the triangle, which corresponds to $\mathrm{Dir}(1,1,1)$.

If we stopped here, our algorithm would fail to learn a *useful* embedding since every data point will draw samples from the same base triangle, *i.e.*, the encoding layer will map all points to the same region in 2D space. Hence, we augment our sampling scheme to allow triangles to be rotated, scaled, and translated.

The set of learned parameters used to force shapes into triangles is $\theta = (\phi \in [-\pi,\pi], s_x \in \mathbb{R}_+, s_y \in \mathbb{R}_+, t_x \in \mathbb{R}, t_y \in \mathbb{R}, \alpha_1, \alpha_2, \alpha_3)$. Here, ϕ is a rotation angle; s_x and s_y are scaling factors of the x and y axes; t_x and t_y are translation amounts in the x and y directions; and α_i are the sampling distribution parameters. A forward pass through this layer is given by

$$\mathbf{w} \sim \mathrm{Dir}(\alpha_1,\alpha_2,\alpha_3)$$

$$\mathbf{p} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \mathbf{Tw} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

As a result, we get triangle-shaped clusters (see Fig. 5). This parameterization can generate every possible triangle in $\mathbb{R}^2$ and is more convenient than learning triangle vertices directly. We can even add individual regularization losses depending on the parameter's semantics. For example, we choose *not* to add regularization to ϕ, to allow it to freely range over its domain; we regularize s_x, s_y towards 1 and t_x, t_y towards 0. Fig. 6 shows the effect this has on different datasets, and also what happens when we remove a degree of freedom from the sampling scheme, namely setting the t_x, t_y translation amounts to 0.

Convex polygons. The triangle sampling scheme described above lends itself easily to generalization to *any* convex polygon in $\mathbb{R}^2$. This can be achieved by observing that a υ-dimensional Dirichlet distribution can be used to generate barycentric coordinates $\mathbf{w} \sim \mathrm{Dir}(\alpha_1,\ldots,\alpha_\upsilon)$ of a υ-sided polygon. In practice, sampling in triangular shapes can be viewed as a special case where the convex polygon of interest has $\upsilon = 3$.

We build a matrix $\mathbf{V} \in \mathbb{R}^{2\times\upsilon}$ of the base convex polygon's υ vertices in $\mathbb{R}^2$ with equally spaced points around a circle of radius 1 for the same reasons as in the triangular case. The bias caused by this choice of initial coordinates is, as earlier, overcome by augmentations that allow for rotation, scaling, and translation. We also see that any point $\mathbf{p} = \mathbf{Vw}$ is inside the specified polygon. To achieve shape regularization, the prior distribution should encourage sampling to occur in all regions of the polygon, which is achieved when all the α_i are set to 1.

3.3. Projection on non-Euclidean spaces

Projection techniques most often concern themselves with finding "compressed" representations of high-dimensional data in a lower dimensional space $\mathbb{R}^q$ that has a standard *Euclidean* structure. This seemingly natural design choice leads to projections where one corner of the projected space is at the largest distance possible from its diagonally opposing corner — assuming, without loss of generalization, that projected points range in $[0,1]^q$. This space cannot represent well certain similarity relationships between categories. For example, consider a dataset having K classes, where each class i, $1 \le i \le K$, is similar to class $i+1 \mod K$. A projection of this dataset in Euclidean space would have to distort *at least one* of these similarity relationships. Well-known instances hereof are projections of the 3D earth surface (with geodesic distances between countries being the similarity relationships) by classical 2D cartographic methods [60].

We address this problem as follows. The sampling mechanism of ShaRP *not only* enables shape regularization, but also allows for the possibility of projecting onto spaces *other than* $\mathbb{R}^2$. For example, using a 3D von Mises-Fisher distribution (vMF) results in projecting points in $\mathcal{S}^2$, the 2-dimensional sphere — i.e., the boundary of a 3D ball. Previous work has explored the use of such distributions in VAEs [61], and our work is the first, to the best of our knowledge, to use this in a specific architecture (see Fig. 1) with the primary goal of producing high quality data projections in non-Euclidean space.

Fig. 7 (top) shows this for a subset of the 'Quick, draw!' dataset [62]. From all the 345 classes of the dataset, we selected the following ten to explore, due to similarities in their appearances: Ambulance, Bicycle, Bulldozer, Car, Firetruck, Motorbike, Pickup truck, Police car, Truck, and Van. We expect good projections to capture true data similarities, *e.g.*, show ambulances next to vans, bicycles next to motorbikes, and so on.

Images (a–c) show three viewpoints of the spherical projection created by ShaRP, with points slightly shaded to represent the spherical surface. In (a) we see the classes Car (bottom-right of sphere) and Police Car (top), which we expect to be similar and therefore projected in neighboring regions; in (b) we see to the right the classes Bicycle and Motorbike — which are again similar and projected next to each other; in (c) we see the Pickup truck (top left), Truck (center), and Firetruck (bottom) are projected in adjacent regions as well.

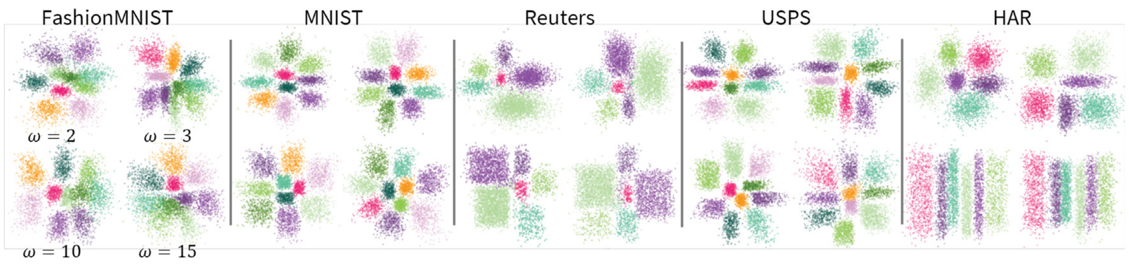


Fig. 4. Demonstration of shape regularization towards rectangles for different values of ω across five datasets. The strength of the "squarification" effect consistently increases with increasing ω, as expected.

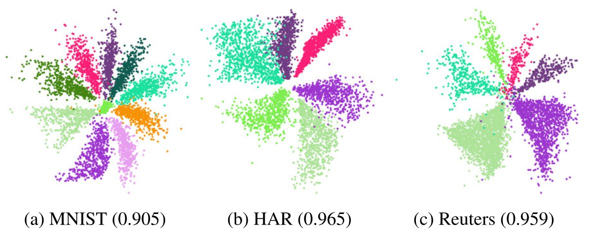


Fig. 5. The results of our Triangular shaping sampling scheme over 3 different datasets. DSC values (in brackets) are close to the best value possible, indicating good class separability.

Fig. 7 (top) d–f show three viewpoints of the same dataset, projected in 3D *Euclidean* space using t-SNE. These views show some similarity of the same-class samples, visible as close points having the same color. Yet, the delineation of same-class points as *clearly separated* same-color regions in the projection, visible with the ShaRP spherical projection (a–e), is now lost. The key problem for t-SNE is that it uses the *entire* 3D space to project data. Given the earlier-mentioned tendencies of t-SNE to create round, organic-like shapes, and also to fill in the projection space with these shapes, this yields a sphere filled in with the projected points. Finding out how such points group into same-class regions, representing class-consistent groupings, is extremely hard, even when examining this 3D scatterplot from multiple viewpoints, due to occlusion and depth effects. Finding out how same-color point groups neighbor each other is even harder for the same reason. The above are fundamental challenges of exploring 3D scatterplots created by projections, *i.e.*, not limited to t-SNE [63]. Additional interactive exploration tools can help finding better viewpoints but cannot eliminate the occlusion and inherent difficulties of understanding a dense 3D scatterplot [64]. In contrast, our spherical space considerably decreases the occlusion problems (maximally two points can be on the same view line). Also, as explained earlier, the spherical space allows for a more flexible point placement to model similarities than Euclidean (3D) space.

One can now ask whether simply avoiding 3D projections and using 2D ones would not solve the above issues. Fig. 7 (top) g,h shows the projections of the same dataset to 2D using ShaRP, respectively t-SNE. We see that ShaRP renders the similarity of same-class points far better than t-SNE due to its shape constraints. We also see an additional advantage of the earlier spherical space: whereas the spherical projection shows that the green and pink classes are similar (since they are close on the sphere, see images (a) and (b)), the 2D projection places these far apart from each other, due to its need to render the other similarities of the respective classes with their neighbors. Clearly, assessing 3D projections — or, more generally, 3D scatterplots — is hard to do in a static 2D view as shown in this paper. For such plots, interaction is paramount, as explained in e.g. [63–66] . Our work is openly available for users to try out such interactive exploration (see GitHub link listed earlier in this section).

Table 3
Mechanisms offered by ShaRP to control the cluster shapes and/or the projection space.

Sampling	Prior	Shape	Space
$\mathbf{z} \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$	$\mathcal{N}(0, I)$	○	$\mathbb{R}^q$
$\mathbf{z} \sim \mathcal{GN}(\mu, \alpha, \omega)$	$\mathcal{GN}(0, 1, \omega)$	□	$\mathbb{R}^q$
$\mathbf{z} \sim \mathrm{Dir}(\alpha_1, \alpha_2, \alpha_3)$ $\mathbf{z} \mapsto \begin{bmatrix} s_x \cos\phi & -s_y \sin\phi \\ s_x \sin\phi & s_y \cos\phi \end{bmatrix} \mathbf{Vz} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$	$\mathrm{Dir}(1, 1, 1)$	△	$\mathbb{R}^2$
$\mathbf{z} \sim \mathrm{vMF}(\mu, \kappa)$	$\mathrm{vMF}(\mu, 0)$	○	$\mathcal{S}^2$

Fig. 7 (bottom) shows the same experiment as above, now done for the MNIST dataset. While we see a clearer separation of the same-color clusters in the t-SNE view, this separation is massively better with spherical ShaRP. The same quality difference is visible in 2D ShaRP *vs* 2D t-SNE.

It is important to note that, for the evaluation of such non-Euclidean projections using quality metrics, we adapt distance calculations to use the appropriate geodesic distance which, on a unit sphere ($\mathcal{S}^2$), is the great-circle distance

$$d_{\mathcal{S}^2}(\mathbf{x}, \mathbf{y}) = 2 \,|\, \arcsin(0.5\, \|\mathbf{x} - \mathbf{y}\|) \,|\, \in [0, \pi].$$

Table 3 summarizes our proposed mechanisms for controlling cluster shapes by sampling distributions in both 2D Euclidean and spherical $\mathcal{S}^2$ spaces.

4. Evaluation

We next discuss how ShaRP gives direct control over cluster shapes while learning to project data (Section 4.1), the quality of ShaRP projections (Section 4.2), and how tuning a single parameter controls the shape regularization strength (Section 4.3). We also show how ShaRP can be used to compute decision maps for exploring trained classification models and compare it with other recent methods for the same task (Section 4.5). Finally, we discuss ShaRP's computational scalability (Section 4.6).

Datasets. We use 15 datasets for evaluation (Table 4) with different classification difficulties, dimensionality, data types (images, motion data, text), all often used in DR evaluations [7].

Techniques. We compare ShaRP with t-SNE, UMAP, and Isomap, due to their wide adoption in the DR arena. We also compare with Auto-Encoders since they are a key building block of our technique; with SSNP since we are extending it; and with NNP [81], a technique that learns to imitate projections, here trained to imitate t-SNE. For both ShaRP and SSNP, we use three different label sources: from the ground truth of the dataset (GT); and pseudolabels created by the K-Means (KM) [82] and Agglomerative (AG) [83] clustering techniques.

Hyperparameter settings. We train ShaRP with the Adam optimizer with default parameter settings. We add L_2 regularization to the network's bottleneck layer with a coefficient of 0.5. We use $\rho = 1$ and $\beta = 0.1$ and mini-batches of 256 data points.

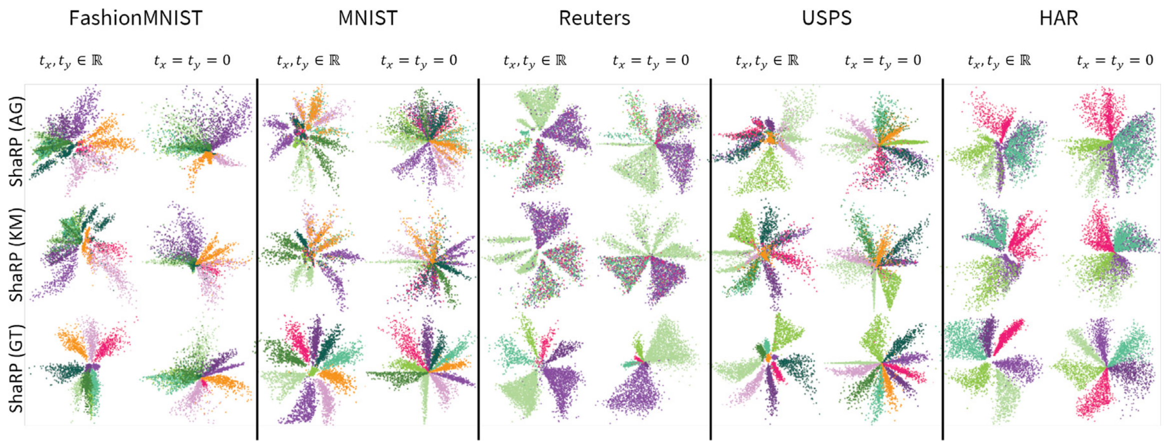


Fig. 6. Triangle-oriented shape regularization demonstrated for five of the studied datasets. We present all three variants of ShaRP and also explore the impact of (dis)allowing the translation of triangles in space. We notice that ShaRP is always able to find a sensible projection where distinct class clusters can be identified, even when we take away some degrees of freedom of the technique (e.g., forcing $t_x = t_y = 0$).

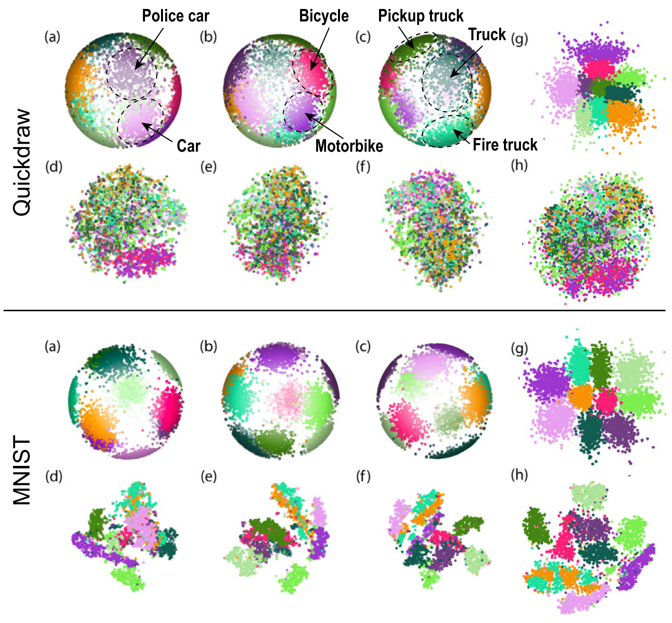


Fig. 7. Projections of the Quickdraw (top) and MNIST (bottom) datasets. Using ShaRP's spherical sampling scheme plots points on the 2D surface of a 3D sphere (different viewpoints shown in a–c). Using such a manifold allows for wrap-around behavior that allows showing more complex similarity relationships of data points. This is in stark contrast with a 3D t-SNE plot (three viewpoints of which are shown in d–f) where similar groups are not easily identifiable. In 2D, a standard t-SNE plot (h) does not make the visualization better; and standard 2D ShaRP (g) can only show groupings in a limited way.

Table 4
Datasets used in our evaluation.

Dataset	Dimensionality (n)	# classes (K)
Bank [67]	63	2
CNAE-9 [68]	856	9
COIL20 [69]	400	20
FashionMNIST [70]	784	10
FMD [71]	1536	10
HAR [72]	561	6
Hate Speech [73]	100	3
IMDB [74]	700	2
MNIST [75]	784	10
Quick, draw! [62]	784	10
Reuters [76]	5000	6
Sentiment [77]	200	2
Spambase [78]	57	2
SVHN [79]	1024	10
USPS [80]	256	10

4.1. Generating shape-regularized projections

Figs. 8–16 show how ShaRP regularizes projections for our 15 studied datasets and compares these results with eight other projection techniques. All images were generated using the same hyperparameter values, showing the robustness of ShaRP to different datasets. Instead of producing scatterplots where cluster shapes, sizes, and intercluster spacing are widely different (as with t-SNE and UMAP), ShaRP generates a more similar representation of the high-dimensional data in each 2D projection (intra-projection regularization). Also, the visual signature obtained is consistent throughout datasets (inter-projection regularization). The learned projections do well with respect to quality metrics (see Fig. 9 and its discussion in Section 4.2).

This comparison shows how the choice of projection algorithm has a strong impact on the structures that arise in the projection plots. Worthy of note are:

For the COIL20 dataset (Fig. 8), all of Isomap, t-SNE, and UMAP produce ring-like structures (more deformed in Isomap) with different degrees of spread and intercluster spacing (most visible in UMAP, with low spread and big spacing between clusters). These seem to point to pairwise sample similarities forming a chain, with points on opposite sides of the rings being most different from each other. In contrast, ShaRP is able to effectively project all samples without wasting the space necessary to "draw" a ring; it draws filled circles with the data points instead, and arranges them in 2D space with approximately uniform intercluster spacing. One may wonder who is 'right' here. We point out that PCA, a well-known linear projection method that should have no trouble preserving such (near-)2D topology, does not produce such structures in 2D. Also, changing the t-SNE perplexity parameters can make these rings disappear — see *e.g.* the related analysis of how setting perplexity can be tricky for t-SNE [53]. As such, we conjecture that the existence of such rings in data space is not clear — it is, at the very least, contingent on the representation used. Our method, by design, does not create these rings because of its shape regularization. This behavior is undoubtedly destructive, in the sense that if a dataset *does* possess such clear patterns in the data space, ShaRP *will* erase them and favor whichever shape regularization target is used. Whether this is desirable is left for the user to decide. Note also that our approach comes at the cost of introducing some false neighbors — which leads to a lower trustworthiness value — while producing better DSC and Stress values.

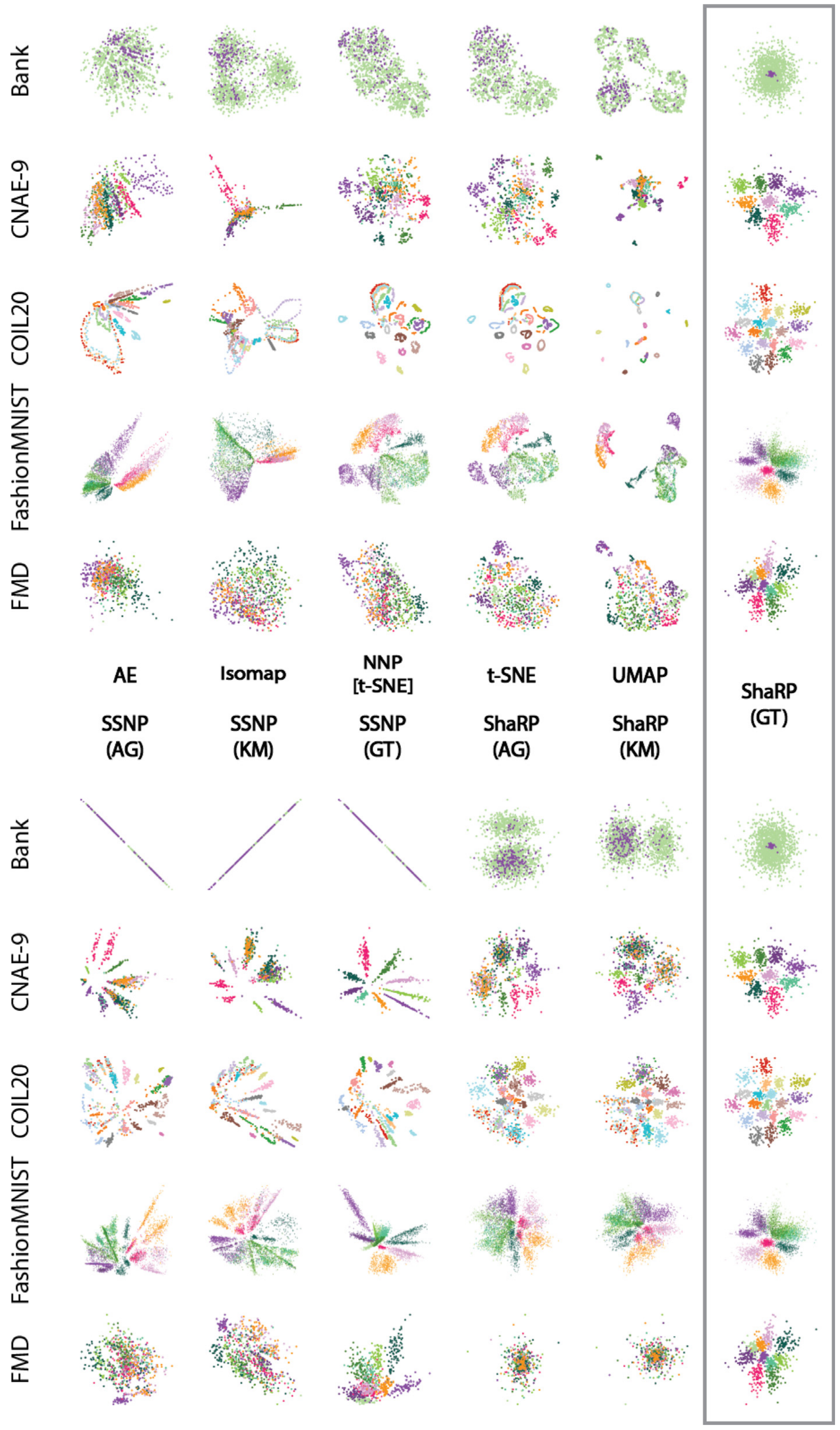


Fig. 8. Projections of 15 datasets by 10 projection techniques (continued in Figs. 15, 16, see end of manuscript). Our ShaRP method produces cluster shapes regularized towards a user-chosen target — here, ellipses — and can handle diverse data distributions, both when using (GT) labels or pseudolabels generated by K-Means (KM) or agglomerative clustering (AG).

Datasets with only two classes (e.g., Bank, IMDB, SpamBase) seem to draw out problematic behavior in SSNP. Namely, as we also discuss next in Section 4.2, the absence of dedicated neurons in the output layer for both positive and negative class *collapses* the projections of different-class data points into a single line. This causes values of Stress that reach 6 orders of magnitude above what is produced by other techniques. ShaRP does not suffer of this problem, being able to reliably place points with different (pseudo-)labels in distinct regions of space.

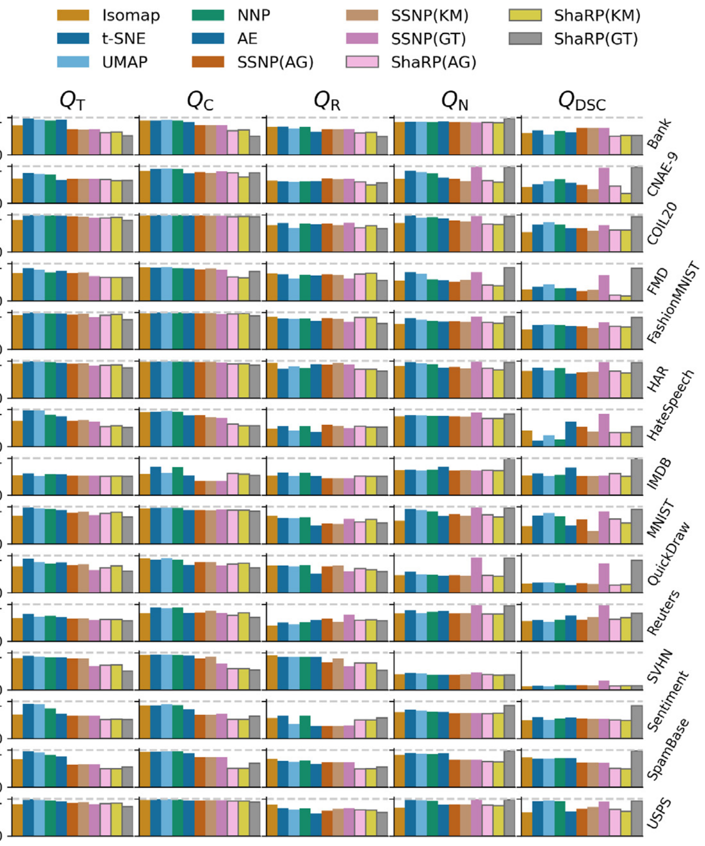


Fig. 9. Performance of studied projection algorithms according to standard quality metrics: Trustworthiness Q_T, Continuity Q_C, Shepard Goodness Q_R, Neighborhood Hit Q_N, and Distance Consistency Q_{DSC} (higher is better, see Table 1). We observe a loss in terms of quality metrics due to shape regularization. Q_{DSC} is also highest for the (GT) variants of SSNP and ShaRP, which is expected as *true* class information is being used. For detailed measurements including the Stress metric, see supplementary material.

Finally, for the Hate Speech dataset, all studied techniques have difficulty coming up with an easily visually separable projection. However, the patterns present in different projections are qualitatively distinguishable: for Isomap, we see lots of intermixing around the middle of the projection, but not towards the edges; in t-SNE, we see small groups of same-class samples, mixed with each other; UMAP seems to collapse most of the samples towards the same spatial region. ShaRP is able to create some "pure" clusters even in absence of ground truth labels (see ShaRP(KM)). If ground truth labels are provided, the two classes become even more distinguishable, even if they are projected concentrically.

Across all datasets, we consistently see ShaRP's ability to regularize cluster shapes towards a desired form, without the need to change hyperparameters to adapt to new datasets.

4.2. Measuring the projection quality

We evaluate ShaRP by trustworthiness, continuity, Shepard correlation, normalized stress, neighborhood hit, and distance consistency (see Section 2.5). We compute these metrics over all datasets using a Gaussian sampling layer which produces ellipse-like clusters. Fig. 9 shows the metric values over all datasets for ShaRP and the other six evaluated projection techniques. Since Stress (Q_S) has values in an unbounded range, we chose not to visualize it here along the other metrics. We report Q_S values, together with detailed numerical values for all other metrics, in the supplementary material.

For the Q_S metric, ShaRP avoids very high values (present in t-SNE, *all* studied datasets; UMAP and AEs, some datasets). We do, however, have higher Stress than SSNP, since we *force* clusters into desired shapes, which can require projected (2D) distances to be quite different from data-space distances. As explained in Section 1, if, for example, data is spread into Gaussian clusters, and we use ShaRP with the elliptical cluster constraint, then we expect to see good quality metric values (Stress and others). Conversely, if the data is spread in different ways than the visual signature selected by the user for ShaRP, then some quality metrics may decrease for ShaRP. Given that our Stress is still lower than t-SNE, UMAP, and AE, we believe this is a reasonable trade-off. For the same reason, Q_T and Q_C tend to have (even if slightly) lower values in ShaRP than in reference projection algorithms — since shaping can introduce (resp. miss) data space neighbors. Overall, we claim that ShaRP offers its capability of shape regularization with slight impacts to quality metrics — stronger for high values of β, see Eq. (11). It is worth noting that, for their AG and KM versions, both SSNP and ShaRP can be held back by the clustering algorithm's ability to properly group the dataset into classes.

To test how ShaRP's support of different regularization shapes affects projection quality, we ran ShaRP to produce clusters in five shapes — ellipses (using Gaussian sampling); rectangles ($\omega = 5$ and $\omega = 15$, see Section 3.2); and triangles (translated in projected space and respectively forced to $t_x = t_y = 0$), for all 5 tested datasets. Table 5 shows the quality metrics for these settings, for five of the tested datasets (for brevity; for the other 10 datasets, the values are similar). We see little variation in these metrics. This points to the robustness of ShaRP and further supports our claim that controlling the visual signatures of projections can be done without (strongly) influencing quality metric values.

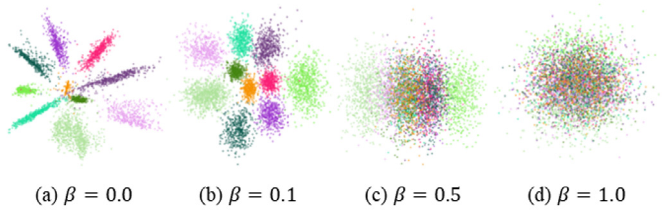
4.3. Control of shape regularization intensity

We adjust the amount of shape regularization through the β multiplier in the loss function (Eq. (11)). Fig. 10 shows this: larger β values force clusters to conform to the shape generated in the sampling layer

Table 5
Non-aggregated quality metrics for different sampling schemes within ShaRP. Rightmost column shows mean square error (MSE) for training.

Dataset	Shape	Q_T	Q_C	Q_R	Q_S	Q_N	Q_{DSC}	MSE
Fashion	○	0.835	0.907	0.463	0.880	0.829	0.766	0.048
	□[$\omega = 5$]	0.825	0.915	0.485	0.892	0.842	0.816	0.050
	□[$\omega = 15$]	0.829	0.916	0.495	0.891	0.813	0.771	0.049
	△	0.826	0.898	0.482	0.875	0.895	0.781	0.050
	△†	0.807	0.808	0.208	0.866	0.834	0.641	0.053
	$\mathcal{S}^2$	0.815	0.910	0.308	0.751	0.952	0.951	0.049
HAR	○	0.823	0.810	0.521	0.749	0.965	0.829	0.010
	□[$\omega = 5$]	0.825	0.880	0.582	0.749	0.952	0.936	0.010
	□[$\omega = 15$]	0.829	0.886	0.513	0.768	0.928	0.909	0.010
	△	0.828	0.854	0.563	0.662	0.978	0.921	0.010
	△†	0.830	0.824	0.426	0.717	0.980	0.928	0.010
	$\mathcal{S}^2$	0.823	0.898	0.423	0.522	0.990	0.990	0.009
MNIST	○	0.732	0.897	0.251	0.859	0.977	0.962	0.055
	□[$\omega = 5$]	0.735	0.896	0.271	0.878	0.979	0.961	0.055
	□[$\omega = 15$]	0.735	0.900	0.230	0.864	0.968	0.948	0.055
	△	0.747	0.874	0.119	0.850	0.986	0.896	0.054
	△†	0.738	0.799	0.148	0.844	0.959	0.769	0.056
	$\mathcal{S}^2$	0.735	0.907	0.219	0.722	0.993	0.994	0.053
Reuters	○	0.554	0.701	0.104	0.676	0.962	0.851	0.001
	□[$\omega = 5$]	0.556	0.699	0.300	0.714	0.975	0.933	0.001
	□[$\omega = 15$]	0.558	0.715	0.256	0.760	0.977	0.953	0.001
	△	0.560	0.696	0.265	0.581	0.981	0.953	0.001
	△†	0.561	0.637	0.002	0.665	0.977	0.793	0.001
	$\mathcal{S}^2$	0.555	0.709	0.283	0.411	0.981	0.971	0.001
USPS	○	0.802	0.922	0.285	0.774	0.973	0.931	0.044
	□[$\omega = 5$]	0.799	0.918	0.350	0.793	0.975	0.948	0.045
	□[$\omega = 15$]	0.798	0.919	0.329	0.792	0.962	0.934	0.045
	△	0.823	0.898	0.363	0.775	0.992	0.887	0.043
	△†	0.804	0.826	0.185	0.763	0.972	0.837	0.045
	$\mathcal{S}^2$	0.826	0.935	0.296	0.559	0.993	0.993	0.040

Shape	Sampling scheme
○	Ellipses, Gaussian sampling
□[$\omega = k$]	Squares, generalized Normal sampling with $\omega = k$
△	Triangles, Dirichlet sampling
△†	Triangles, Dirichlet sampling with $t_x = t_y = 0$
$\mathcal{S}^2$	Spherical, von Mises-Fisher sampling



β =	0.0	0.05	0.1	0.25	0.5	1.0
Trustworthiness	0.87	0.81	0.79	0.77	0.66	0.54
Continuity	0.95	0.93	0.85	0.74	0.70	0.56
Shepard Corr.	0.39	0.30	0.27	0.26	0.21	0.09
Stress	0.93	0.84	0.78	0.66	0.57	0.54
Neighborhood Hit	0.99	0.99	0.97	0.91	0.60	0.45

Fig. 10. The β coefficient (Eq. (11)) controls the shape regularization strength, shown here on the USPS dataset. $\beta = 0$ approximately reproduces SSNP (a). Increasing it (b, c) progressively forces the learned clusters into circular shapes, up to the point where the projection is of low quality (d).

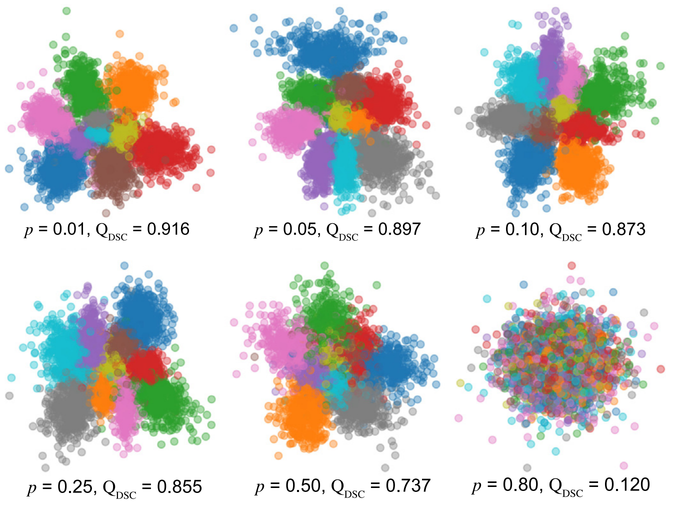


Fig. 11. Progressively mixing classes in the MNIST dataset (i.e., increasing the mixing fraction p) leads to the progressive degradation of the projection produced by ShaRP, as visible above and also measured by the class separability metric Q_{DSC}. Even though ShaRP has access to the (noisy) labels, they do not artificially create separated clusters — as is expected from projection methods.

— ellipses, in this case. Exaggerated shape regularization (high β), however, makes ShaRP favor 'shape over data' too much and creates projections which cannot properly depict data — sampling from a distribution similar to the prior overshadows producing a sensible embedding. The table below in Fig. 10 gives the quality metrics for the different β settings. In our tests, we have found a value of $\beta = 0.1$ to give consistently good results.

4.4. Faithfulness to data class structure

As ShaRP is a method that uses (pseudo-)labels $\bar{Y}$, it is important to assess whether the creation of clusters in a projection is artificial, and therefore less trustworthy, similar to earlier studies that aim to assess class faithfulness in projections [84,85]. To this end, we conduct an experiment in which we randomly assign a different label $y_i' \in \{1, \dots, K\}$ to increasing fractions $p \in [0, 1]$ of the samples of each class in a dataset — we choose MNIST due to expected reader familiarity. This *class mixing* should produce correspondingly *mixed* projections as we increase p. Fig. 11 shows exactly this effect: As we increase p, the observed visual separation of same-label clusters decreases; also, the class separability metric Q_{DSC} *monotonically* decreases.

This experiment provides evidence that ShaRP does not artificially create clusters based on the (pseudo-)label information it is supplied. That is to say, ShaRP's shape regularization effect still preserves intra-class similarities, and therefore data class structure, as we would expect from a sensible DR algorithm.

4.5. Generating decision boundary maps

Visualizing how a classifier partitions the input space into regions corresponding to different classes can give insight into its correctness and robustness. For high-dimensional inputs, this visualization uses a projected space through so-called Decision Boundary Maps (see Section 2.4). A key factor to the quality of DBMs is the inverse projection used. We next focus on comparing with SDBM given that, from all existing techniques in this area (Standard DBM [15], SDBM [26], DeepView [30]), this technique strikes the best balance of quality and computational scalability.

Fig. 13 compares SDBM and ShaRP for visualizing the decision boundaries of a given, (per-dataset) fixed, classifier. We chose as classifier model a shallow Multilayer Perceptron — 3 layers with sizes 512, 128, 32 — with ReLU activations. We additionally encode the confidence of the classifier — *i.e.*, the probability assigned to the most probable class — in the saturation of the DBM colors, giving extra insights into the classification quality per region. Note that, since ShaRP learns to jointly project *and* classify, we could use the classifier head of

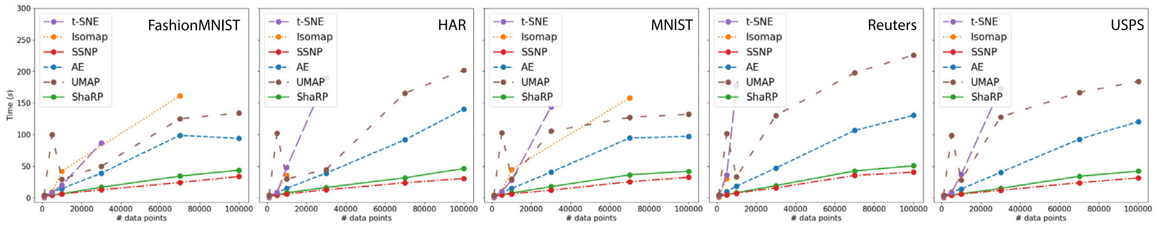


Fig. 12. Times for a call to `fit_transform` (that is, both training and inference), all studied projection techniques, five datasets. Runs were stopped after 5 min. We see that ShaRP is faster than all other tested DR techniques, being only marginally slower than SSNP, which does not offer visual pattern control. See Section 4.6.

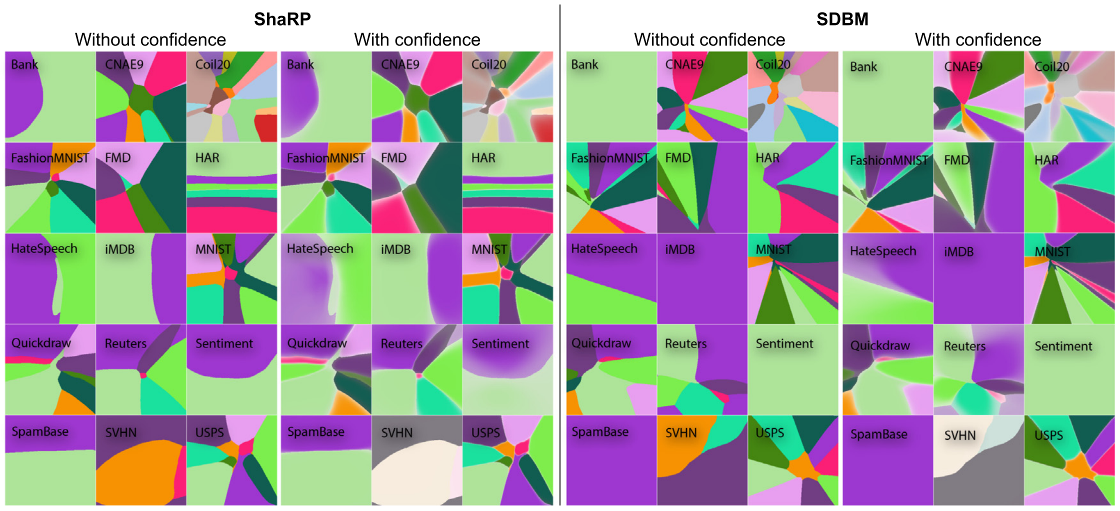


Fig. 13. DBMs created by ShaRP and SDBM for all studied datasets, visualized without and with saturation-encoded confidence. The classifier is frozen per dataset. Immediately noticeable are the 2-class cases where SDBM produces degenerate DBMs (Sentiment, SpamBase datasets). We also see SSNP's starburst pattern leaking into SDBM (FashionMNIST, MNIST datasets), which ShaRP avoids.

its decoder over a grid of points in the latent space, immediately obtaining information on which class is projected to which region, without any additional training. Also, ShaRP outputs a *probability distribution* over classes, enabling the visualization of uncertainty — *e.g.*, entropy — in the DBMs, without any additional effort. However, as mentioned above, we choose not to use ShaRP's built-in classifier, but an externally given one. This decouples how the DBM is generated from what it is used to visualize — which, we argue, makes full sense. Indeed, users likely want to use a DBM visualization to inspect their own classifiers, and not the one in ShaRP.

In Fig. 13, we see that SDBM seems to fail to generate sensible DBMs for cases where there are only two classes in the dataset (see *e.g.* datasets Bank and SpamBase). We believe this is due to a common practice in Deep Learning models: using a single neuron in the output layer of the classification network instead of naively using two, one per class. This absent neuron would indeed be redundant for classification since the output layer's values must sum to one (as they form a probability distribution). We argue that this unnecessarily and strongly limits the power of SDBM to learn projection and inverse projection functions that place elements of different classes in significantly different regions of space. ShaRP does not possess that limitation, since it uses explicitly *two* neurons in its output layer for projections where only two classes of points exist.

Additionally, the starburst visual signature of SSNP leaks through SDBM, generating DBMs that have long, line-like regions (see datasets FashionMNIST, FMD, MNIST). ShaRP *might* generate line-like regions in DBMs as well, usually when the training process makes the learned clusters stack vertically or horizontally (see dataset HAR), but this effect is less frequent and the regions are not as thin as those in SDBM, which poses a smaller hindrance to understanding the visualization. The above show that ShaRP's regularizing clusters in projections also regularizes the generated DBMs, while preserving the between-class structures as demonstrated in Section 4.4.

It is worthy of note that ShaRP's classifier head (see Fig. 1) is able to predict classes directly for 2D points, thereby "painting" a DBM, *i.e.*, computing the labels that lead to colors for each DBM pixel. However, this classifier runs *directly on projected space coordinates*. In order to classify a new, unseen data point, this point must be projected by ShaRP and then classified, a process through which information may be lost.

In Section 3.3, we showed that ShaRP can create projections on the S^2 sphere, which can better show certain neighboring relations between sample clusters. The same idea applies to decision maps. Fig. 14 shows decision maps created with ShaRP for a Multilayer Perceptron classifier trained on the Quickdraw and MNIST datasets. We see, just as we do for the 2D counterparts (Fig. 13), different decision zones of the classifier, with an additional degree of freedom given by the wrap-around nature of the S^2 space. The decision zones naturally enclose the training samples (small white dots) and have decreasing confidence as one approaches their boundaries. Note that our goal here to create DBMs which better capture the underlying neighborhoods of high-dimensional data points (which we achieve by using the S^2 sphere) is related conceptually to goals of preserving the data topology via projections [86]. The key difference is that we aim to generate DBMs, which involve *extrapolating* a projection from a given dataset, whereas [86] focuses on preserving topology as defined by a given dataset.

4.6. Computational performance

Fig. 12 shows how ShaRP fares compared to other projection techniques *vs.* computational time. Tests were run on a PC with an AMD Ryzen 9 5900HX 3.3 GHz 8-core processor and an NVIDIA RTX 3080

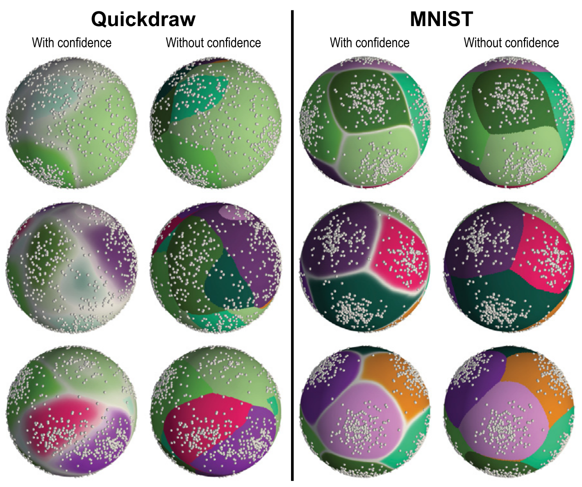


Fig. 14. Decision maps for the Quickdraw and MNIST datasets using the $\mathcal{S}^2$ space. Bottom row maps classifier confidence to color saturation. White dots show training samples. Compare with Fig. 7.

GPU. ShaRP is much faster than t-SNE (5 to 10 times speedup) and Isomap (5 to 6 times speedup). It is also faster than AE, UMAP, and only slightly slower than SSNP, its predecessor. The used batch size (256 data points) is largely responsible for ShaRP's speed and, according to our experiments, does not negatively impact the quality metrics of the obtained projections, meaning it could be increased by a user whose priority is fast iteration. For ease of comparison, we use evaluation settings similar to those in other works advocating analogous Deep Learning architectures [14,81,87,88]. We refer the reader to [87,88] for experiments on how the batch size affects performance for a wide variety of datasets.

5. Discussion

We next discuss several key aspects of our proposed method.

Genericity: ShaRP can handle high-dimensional dataset of any nature (*e.g.*, text, images, or tabular) as long as the data values are quantitative, as such data directly works with autoencoders. One-hot encoding or Categorical Variational Auto-Encoders [89] can overcome this limitation with only slight adaptions to our network architecture.

Inverse projections: Since using an autoencoder architecture, ShaRP automatically provides the inverse projection P^{-1} from the embedding to the data space at no additional cost or complexity.

Scalability: Since based on a deep learning architecture, ShaRP's running time is linear in the number of samples and dimensions and considerably faster than classical methods such as t-SNE, Isomap, or UMAP (see Section 4.6). After training, ShaRP's operation is deterministic and also has the out-of-sample property, *i.e.*, it can project unseen samples along the training ones. This allows its simple usage for constructing decision maps, for which the inverse projection P^{-1} must be applied to unseen points (pixels).

Shape and space control: ShaRP allows controlling the shapes of similar-sample clusters (provided by (pseudo-)labels) to match ellipses, triangles, rectangles, or more generally any convex polygon. Doing this can decrease certain projection quality metrics as compared to using 'free' projections which do not constrain cluster shapes. We see this as an application-dependent trade-off that users can settle based on tuning a single, simple, parameter (see Section 3.2). As explained in Section 1, this trade-off means that, if users select, say, ShaRP with rectangles, they will (a) get a projection containing mainly rectangles and no other patterns; and (b) know that these patterns are due to their choice. In contrast, when using other DR techniques, one can expect a wide set of patterns in the projection (also depending on the technique's parameter values). We argue that this can be confusing since (a) one has no clear control of the appearing patterns; and, more importantly, (b) the spontaneous appearance of such patterns can wrongly suggest they actually depict data patterns [53]. We remove such uncertainties by allowing the user to control the visual patterns in the projection. On the flip side, this means that if cluster shape patterns do exist in the data space, ShaRP will explicitly undo them and favor the user-chosen shape target instead. This is a relevant decision point for whether ShaRP is an adequate projection technique for a given task.

Shape-regularized projections can help visualization tasks. For example, rectangular shaped clusters ease the task of adding (rectangular) thumbnails or similar annotations atop of the respective clusters (see Fig. 3).

Independently on the selected cluster shape, ShaRP can project data to Euclidean spaces (like the 2D plane) or spherical ones (like $\mathcal{S}^2$). This is an additional global degree of control of the projection shape.

Decision maps: ShaRP can trivially create decision maps of any classification model trained on its input data samples. Compared to other decision map visualizations (*e.g.* SDBM or DeepView), ShaRP offers the additional aforementioned control of shape and space. Simply put, if this control enables one to generate a good *projection* for understanding a dataset, the same control (and projection) can be used to create a decision map for a classifier trained on that dataset.

Limitations: While flexible, ShaRP also has several limitations.

The sampling schemes we presented in Section 3 support shaping clusters into ellipses and convex polygons. A wider variety of shapes can be obtained by devising new sampling schemes. However, obtaining log-probabilities for samples of complex sampling schemes can be computationally intensive.

As many other DR methods relying on clustering, e.g. SSNP [14,88], ShaRP relies on a reasonable estimate of the number of clusters to use if no ground-truth labels exist. If ground-truth labels exist, the problem is solved — one either uses them directly or sets the cluster count to the number of different classes. Evidently, the quality of the clustering used in this case is important as well. This is a standard limitation of similar DR methods that use clustering.

More importantly, the issue of shape control raises some hard questions. On the one hand, no control allows a projection technique to *freely* place samples to optimally follow the data values. Yet, as we argued, this also allows the technique to enforce its own 'visual signature' which typically does not depend on the data but on the technique's design, so it can be visually misleading. On the other hand, controlling shapes largely removes this algorithmic bias. Yet, this puts the responsibility of selecting 'good' shapes to further interpret the data on the user. All taken into account, we believe the latter option to be better. Yet, further studies *e.g.* involving user experiments are needed to see how shapes affect the perception of a ShaRP projection.

Another open question is how to control ShaRP in *intuitive* ways. So far, we have shown that the user can specify the shape of visual patterns ro emerge in a projection, while taking some loss in projection quality in return. Yet, users may want to tune this, e.g., specify how much they want to control shapes *vs* how much quality loss they agree with, beyond setting the ω parameter. A way forward would be to show local projection errors for ShaRP (following e.g. [38,90]) and provide users with interactive tools to tune such errors *vs* the desired visual cluster shapes. A separate interesting open issue is how to quantify and display how well ShaRP enforces the desired visual shapes.

6. Conclusion

We have presented ShaRP, a new method for creating projections of high-dimensional data. ShaRP leverages variational autoencoders and (pseudo-)labeled datasets to allow users to control the shapes of clusters

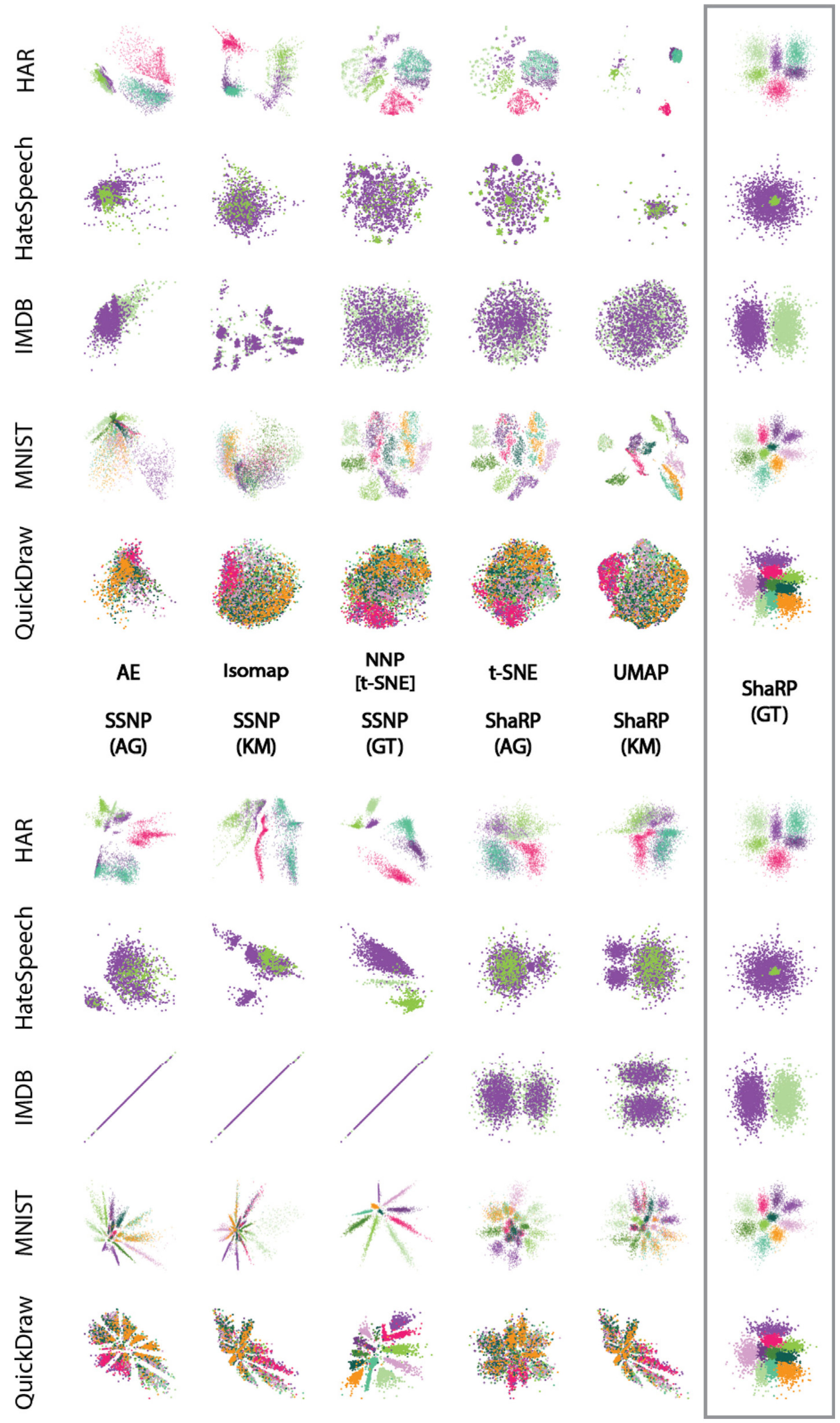


Fig. 15. Continuation of Fig. 8.

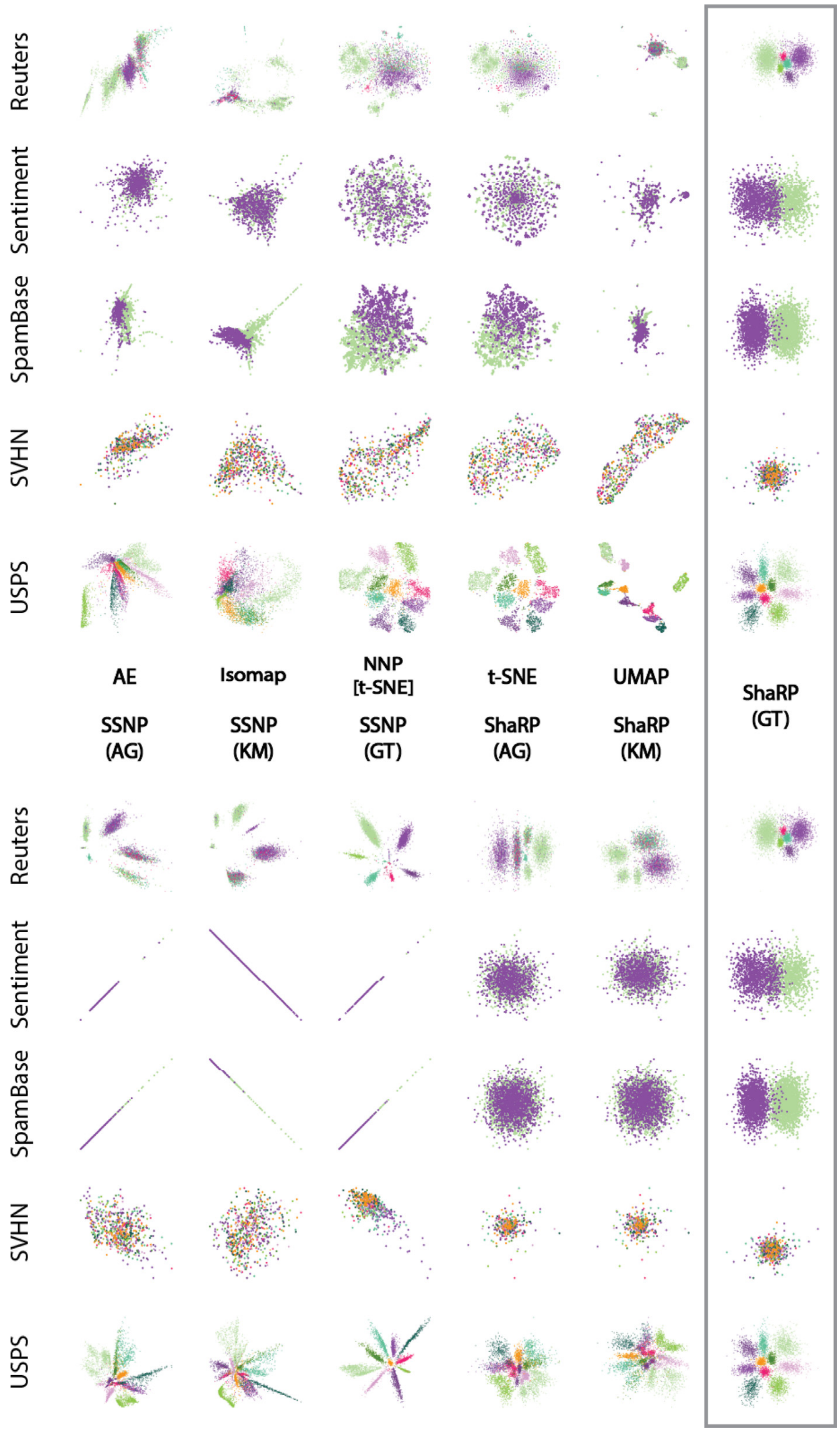


Fig. 16. Continuation of Fig. 8.

created in Euclidean or spherical spaces. Alongside this, ShaRP keeps desirable features of projection techniques such as genericity, computational scalability, ease of use and implementation, and out-of-sample ability.

Several future work directions are possible. First, exploring new shape as well as space constraints would allow users an even greater control of the resulting projections. This can prove useful to *e.g.* create projections onto specific 2D or 3D manifolds that match known priors of the data distribution. Secondly, we aim to study sampling schemes that naturally encourage further visual aspects of interest of projections, *e.g.*, cluster separability. Finally, we aim to explore applying ShaRP recursively to create an interactively navigable hierarchy of clustered data in projections.

CRediT authorship contribution statement

Alister Machado: Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Alexandru Telea:** Writing – review & editing, Writing – original draft, Supervision, Conceptualization. **Michael Behrisch:** Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to the code in the main text.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cag.2024.104093.

References

[1] Hoffman P, Grinstein G. A survey of visualizations for high-dimensional data mining. Inf Vis Data Min Knowl Discov 2002;104:47–82.
[2] Yin H. Nonlinear dimensionality reduction and data visualization: A review. Int J Autom Comput 2007;4(3):294–303.
[3] Bunte K, Biehl M, Hammer B. A general framework for dimensionality reducing data visualization mapping. Neural Comput 2012;24(3):771–804.
[4] Sorzano C, Vargas J, Pascual-Montano A. A survey of dimensionality reduction techniques. 2014, arXiv:1403.2877 [stat.ML].
[5] Liu S, Maljovec D, Wang B, Bremer PT, Pascucci V. Visualizing high-dimensional data: Advances in the past decade. IEEE TVCG 2015;23(3):1249–68.
[6] Cunningham J, Ghahramani Z. Linear dimensionality reduction: Survey, insights, and generalizations. JMLR 2015;16:2859–900.
[7] Espadoto M, Martins RM, Kerren A, Hirata NST, Telea AC. Toward a quantitative survey of dimension reduction techniques. IEEE Trans Vis Comput Graph 2021;27(3):2153–73.
[8] Nonato L, Aupetit M. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. IEEE TVCG 2018. http://dx.doi.org/10.1109/TVCG.2018.2846735.
[9] Behrisch M, Blumenschein M, Kim NW, Shao L, El-Assady M, Fuchs J, et al. Quality metrics for information visualization. Comput Graph Forum 2018;37(3):625–62.
[10] Pandey AV, Krause J, Felix C, Boy J, Bertini E. Towards understanding human similarity perception in the analysis of large sets of scatter plots. In: Kaye J, Druin A, Lampe C, Morris D, Hourcade JP, editors. Conference on human factors in computing systems CHI. ACM; 2016, p. 3659–69. http://dx.doi.org/10.1145/2858036.2858155.
[11] Wang Y, Feng K, Chu X, Zhang J, Fu C, Sedlmair M, et al. A perception-driven approach to supervised dimensionality reduction for visualization. IEEE Trans Vis Comput Graph 2018;24(5):1828–40. http://dx.doi.org/10.1109/TVCG.2017.2701829.
[12] Tenenbaum JB, Silva Vd, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science 2000;290(5500):2319–23.
[13] Machado A, Telea A, Behrisch M. ShaRP: Shape-regularized multidimensional projections. In: Proc. euroVA. 2023.
[14] Espadoto M, Hirata NST, Telea AC. Self-supervised dimensionality reduction with neural networks and pseudo-labeling. In: Hurter C, Purchase HC, Braz J, Bouatouch K, editors. Proc. IVAPP (2021). SCITEPRESS; 2021, p. 27–37. http://dx.doi.org/10.5220/0010184800270037.
[15] Rodrigues FCM, Hirata R, Telea AC. Image-based visualization of classifier decision boundaries. In: 31st SIBGRAPI conference on graphics, patterns and images. IEEE Computer Society; 2018, p. 353–60. http://dx.doi.org/10.1109/SIBGRAPI.2018.00052.
[16] van der Maaten L, Postma E. Dimensionality reduction: A comparative review. Tech. rep., Netherlands: Tilburg University; 2009, Tech. report TiCC TR 2009-005.
[17] Li P, Pei Y, Li J. A comprehensive survey on design and application of autoencoder in deep learning. Appl Soft Comput 2023;138(C).
[18] KP FRS. Liii. on lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci 1901;2(11):559–72. http://dx.doi.org/10.1080/14786440109462720.
[19] van der Maaten L, Hinton GE. Visualizing high-dimensional data using t-SNE. J Mach Learn Res 2008;9:2579–605.
[20] van der Maaten L. Accelerating t-sne using tree-based algorithms. J Mach Learn Res 2014;15(1):3221–45. http://dx.doi.org/10.5555/2627435.2697068.
[21] Ulyanov D. Multicore-TSNE. 2016, https://github.com/DmitryUlyanov/Multicore-TSNE.
[22] McInnes L, Healy J. UMAP: uniform manifold approximation and projection for dimension reduction. 2018, CoRR abs/1802.03426. arXiv:1802.03426.
[23] dos Santos Amorim EP, Brazil EV, Daniels J, Joia P, Nonato LG, Sousa MC. iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In: 2012 IEEE conference on visual analytics science and technology. 2012, p. 53–62.
[24] Espadoto M, Rodrigues FCM, Hirata NST, Hirata R. Deep Learning Inverse Multidimensional Projections. In: Proc. euroVA. 2019, p. 5.
[25] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–7.
[26] Oliveira AA, Espadoto M, Hirata Jr R, Telea AC. SDBM: Supervised Decision Boundary Maps for Machine Learning Classifiers. In: VISIGRAPP (3: IVAPP). 2022, p. 77–87.
[27] Wang Y, Machado A, Telea A. Quantitative and qualitative comparison of decision map techniques for explaining classification models. Informatics 2023;16(9).
[28] Espadoto M, Appleby G, Suh A, Cashman D, Li M, Scheidegger CE, et al. UnProjection: Leveraging inverse-projections for visual analytics of high-dimensional data. IEEE Trans Vis Comput Graph 2021;29(2):0.
[29] Rodrigues FCM, Espadoto M, Jr. RH, Telea A. Constructing and visualizing high-quality classifier decision boundary maps. Information 2019;10(9):280–97.
[30] Schulz A, Gisbrecht A, Hammer B. Using discriminative dimensionality reduction to visualize classifiers. Neural Process Lett 2015;42:27–54.
[31] Schulz A, Hinder F, Hammer B. DeepView: Visualizing classification boundaries of deep neural networks as Scatter Plots Using Discriminative Dimensionality Reduction. In: Proceedings of the twenty-ninth international joint conference on artificial intelligence. International joint conferences on artificial intelligence organization;. 2020, p. 2305–11.
[32] Rauber P, Falcao A, Telea A. Projections as visual aids for classification system design. Inf Vis 2017;17(4):282–305.
[33] Douglas Carroll J, Arabie P. Chapter 3 - multidimensional scaling. In: Birnbaum MH, editor. Measurement, judgment and decision making. Handbook of perception and cognition. 2nd ed.. San Diego: Academic Press; 1998, p. 179–250. http://dx.doi.org/10.1016/B978-012099975-0.50005-1, URL: https://www.sciencedirect.com/science/article/pii/B9780120999750500051.
[34] Venna J, Kaski S. Local multidimensional scaling. Neural Netw 2006;19(6):889–99.
[35] Martins RM, Coimbra DB, Minghim R, Telea A. Visual analysis of dimensionality reduction quality for parameterized projections. Comput Graph 2014;41:26–42. http://dx.doi.org/10.1016/j.cag.2014.01.006, URL: https://www.sciencedirect.com/science/article/pii/S0097849314000235.
[36] 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 Trans Vis Comput Graphics 2008;14(3):564–75.
[37] Schreck T, von Land esberger T, Bremm S. Techniques for precision-based visual analysis of projected data. Inf Vis 2010;9(3):181–93. http://dx.doi.org/10.1057/IVS.2010.2.
[38] Aupetit M. Visualizing distortions and recovering topology in continuous projection techniques. Neurocomputing 2007;70(7–9):1304–30. http://dx.doi.org/10.1016/j.neucom.2006.11.018.
[39] Lespinats S, Aupetit M. Checkviz: Sanity check and topological clues for linear and non-linear mappings. Comput Graph Forum 2011;30. URL: https://api.semanticscholar.org/CorpusID:13570201.
[40] Martins R, Minghim R, Telea A. Explaining neighborhood preservation for multidimensional projections. In: Proc. CGVC . eurographics. 2015.

[41] Albuquerque G, Eisemann M, Magnor MA. Perception-based visual quality measures. In: 6th IEEE conference on visual analytics science and technology. IEEE Computer Society; 2011, p. 13–20. http://dx.doi.org/10.1109/VAST.2011.6102437.

[42] Motta R, Minghim R, de Andrade Lopes A, de Oliveira MCF. Graph-based measures to assist user assessment of multidimensional projections. Neurocomputing 2015;150:583–98. http://dx.doi.org/10.1016/J.NEUCOM.2014.09.063.

[43] Sedlmair M, Munzner T, Tory M. Empirical guidance on scatterplot and dimension reduction technique choices. IEEE Trans Vis Comput Graph 2013;19(12):2634–43. http://dx.doi.org/10.1109/TVCG.2013.153.

[44] Benato BC, Falcão AX, Telea AC. Measuring the quality of projections of high-dimensional labeled data. Comput Graph 2023;116:287–97. http://dx.doi.org/10.1016/j.cag.2023.08.023, URL: https://www.sciencedirect.com/science/article/pii/S0097849323001929.

[45] Sedlmair M, Aupetit M. Data-driven evaluation of visual quality measures. Comput Graph Forum 2015;34(3):201–10.

[46] Sips M, Neubert B, Lewis JP, Hanrahan P. Selecting good views of high-dimensional data using class consistency. Comput Graph Forum 2009;28(3):831–8.

[47] Tatu A, Bak P, Bertini E, Keim D, Schneidewind J. Visual quality metrics and human perception: An initial study on 2d projections of large multidimensional data. In: Proceedings of the international conference on advanced visual interfaces. Association for Computing Machinery; 2010, p. 49–56.

[48] Joia P, Paulovich FV, Coimbra D, Cuminato JA, Nonato LG. Local affine multidimensional projection. IEEE Trans Vis Comput Graph 2011;17(12):2563–71.

[49] Etemadpour R, Motta R, de Souza Paiva JG, Minghim R, de Oliveira MCF, Linsen L. Perception-based evaluation of projection methods for multidimensional data visualization. IEEE Trans Vis Comput Graph 2015;21(1):81–94. http://dx.doi.org/10.1109/TVCG.2014.2330617.

[50] Cutura R, Morariu C, Cheng Z, Wang Y, Weiskopf D, Sedlmair M. Hagrid: using hilbert and gosper curves to gridify scatterplots. J Vis 2022. http://dx.doi.org/10.1007/s12650-022-00854-7.

[51] Abbas MM, Aupetit M, Sedlmair M, Bensmail H. ClustMe: A visual quality measure for ranking monochrome scatterplots based on cluster patterns. Comput Graph Forum 2019;38(3):225–36. http://dx.doi.org/10.1111/cgf.13684.

[52] Abbas MM, Ullah E, Baggag A, Bensmail H, Sedlmair M, Aupetit M. ClustML: A measure of cluster pattern complexity in scatterplots learnt from human-labeled groupings. Inf Vis 2021;23:105–22, URL: https://api.semanticscholar.org/CorpusID:235266213.

[53] Wattenberg M, Fernanda V, Johnson I. How to use t-SNE effectively. Distill 2016. http://dx.doi.org/10.23915/distill.00002.

[54] Appleby G, Espadoto M, Chen R, Goree S, Telea AC, Anderson EW, et al. HyperNP: Interactive visual exploration of multidimensional projection hyperparameters. 2021, CoRR abs/2106.13777. arXiv:2106.13777.

[55] Makhzani A, Shlens J, Jaitly N, Goodfellow IJ. Adversarial autoencoders. 2015, CoRR abs/1511.05644. arXiv:1511.05644.

[56] Chollet F, et al. Keras. 2015, https://keras.io.

[57] Dillon JV, Langmore I, Tran D, Brevdo E, Vasudevan S, Moore D, et al. Tensorflow distributions. 2017, http://dx.doi.org/10.48550/ARXIV.1711.10604.

[58] Bengio Y, Courville AC, Vincent P. Representation learning: A review and new perspectives. IEEE Trans Pattern Anal Mach Intell 2013;35(8):1798–828. http://dx.doi.org/10.1109/TPAMI.2013.50.

[59] Kingma DP, Welling M. Auto-encoding variational bayes. In: Bengio Y, LeCun Y, editors. 2nd international conference on learning representations. 2014.

[60] van Wijk JJ. Unfolding the earth: Myriahedral projections. Cartogr J 2008;45(1):32–42.

[61] Davidson TR, Falorsi L, Cao ND, Kipf T, Tomczak JM. Hyperspherical variational auto-encoders. In: Globerson A, Silva R, editors. Proceedings of the thirty-fourth conference on uncertainty in artificial intelligence. AUAI Press; 2018, p. 856–65, URL: http://auai.org/uai2018/proceedings/papers/309.pdf.

[62] Ha D, Eck D. A neural representation of sketch drawings. In: 6th international conference on learning representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3 2018, conference track proceedings. 2018, OpenReview.net, URL: https://openreview.net/forum?id=Hy6GHpkCW.

[63] Tian Z, Zhai X, Steenpaal Gvan, Yu L, Dimara E, Espadoto M, et al. Quantitative and qualitative comparison of 2D and 3D projection techniques for high-dimensional data. Information 2021;12(239).

[64] Castelein W, Tian Z, Mchedlidze T, Telea A. Viewpoint-based quality for analyzing and exploring 3D multidimensional projections. In: Proc. IVAPP. 2023.

[65] Poco J, Etemadpour R, Paulovich F. A framework for exploring multidimensional data with 3D projections. Comput Graph Forum 2011;30(3):1111–20.

[66] Coimbra D, Martins R, Neves T, Telea A, Paulovich F. Explaining three-dimensional dimensionality reduction plots. Inf Vis 2016;15(2):154–72.

[67] Moro S, Cortez P, Rita P. A data-driven approach to predict the success of bank telemarketing. Decis Support Syst 2014;62:22–31. http://dx.doi.org/10.1016/J.DSS.2014.03.001.

[68] Ciarelli PM, Oliveira E. Agglomeration and elimination of terms for dimensionality reduction. In: Ninth international conference on intelligent systems design and applications. IEEE Computer Society; 2009, p. 547–52. http://dx.doi.org/10.1109/ISDA.2009.9.

[69] Nene SA, Nayar SK, Murase H. Columbia object image library (coil-20). Tech. rep. CUCS-005-96, Department of Computer Science, Columbia University; 1996.

[70] Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017, CoRR abs/1708.07747. arXiv:1708.07747.

[71] Sharan L, Rosenholtz R, Adelson E. Material perception: What can you see in a brief glance? J Vis 2010;9:784. http://dx.doi.org/10.1167/9.8.784.

[72] Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz JL. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: Ambient assisted living and home care. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012, p. 216–23.

[73] Davidson T, Warmsley D, Macy MW, Weber I. Automated hate speech detection and the problem of offensive language. In: Proceedings of the eleventh international conference on web and social media. AAAI Press; 2017, p. 512–5, URL: https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665.

[74] Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C. Learning word vectors for sentiment analysis. In: Lin D, Matsumoto Y, Mihalcea R, editors. The 49th annual meeting of the association for computational linguistics: Human language technologies, proceedings of the conference. The Association for Computer Linguistics; 2011, p. 142–50, URL: https://aclanthology.org/P11-1015/.

[75] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE 1998;86(11):2278–324. http://dx.doi.org/10.1109/5.726791.

[76] Thoma M. The reuters dataset. 2017.

[77] Kotzias D, Denil M, de Freitas N, Smyth P. From group to individual labels using deep features. In: Cao L, Zhang C, Joachims T, Webb GI, Margineantu DD, Williams G, editors. Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2015, p. 597–606. http://dx.doi.org/10.1145/2783258.2783380.

[78] Hopkins M, Reeber E, Forman G, Suermondt J. Spambase. 1999, http://dx.doi.org/10.24432/C53G6X, UCI Machine Learning Repository.

[79] Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY. Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning 2011. 2011, URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[80] Hull JJ. A database for handwritten text recognition research. IEEE Trans Pattern Anal Mach Intell 1994;16(5):550–4. http://dx.doi.org/10.1109/34.291440.

[81] Espadoto M, Hirata NST, Telea AC. Deep learning multidimensional projections. Inf Vis 2020;19(3):247–69. http://dx.doi.org/10.1177/1473871620909485.

[82] Lloyd SP. Least squares quantization in PCM. IEEE Trans Inform Theory 1982;28(2):129–36. http://dx.doi.org/10.1109/TIT.1982.1056489.

[83] Kaufman L, Rousseeuw PJ. Finding groups in data: an introduction to cluster analysis. John Wiley; 1990, http://dx.doi.org/10.1002/9780470316801.

[84] Colange B, Peltonen J, Aupetit M, Dutykh D, Lespinats S. Steering distortions to preserve classes and neighbors in supervised dimensionality reduction. In: Proc. neurIPS. 2020, p. 214–25.

[85] Jeon H, Kuo YH, Aupetit M, Ma KL, Seo J. Classes are not clusters: Improving label-based evaluation of dimensionality reduction. IEEE TVCG 2024;30(1):781–91.

[86] Doraiswamy H, Tierny J, Silva PJS, Nonato LG, Silva C. TopoMap: A 0-dimensional homology preserving projection of high-dimensional data. IEEE TVCG 2021;27(2):561–71.

[87] Espadoto M, Falcao A, Hirata N, Telea A. Improving neural network-based multidimensional projections. In: Proc. IVAPP. SciTePress; 2020.

[88] Oliveira A, Espadoto M, Hirata R, Hirata N, Telea A. Improving self-supervised dimensionality reduction: Exploring hyperparameters and pseudo-labeling strategies. In: Springer CCIS 1691. 2023, p. 135–61.

[89] Morton JT, Silverman J, Tikhonov G, Lähdesmäki H, R Bonneau. Scalable estimation of microbial co-occurrence networks with variational autoencoders. 2021, http://dx.doi.org/10.1101/2021.11.09.467939, bioRxiv.

[90] Martins RM, Coimbra DB, Minghim R, Telea AC. Visual analysis of dimensionality reduction quality for parameterized projections. Comput Graph 2014;41:26–42. http://dx.doi.org/10.1016/j.cag.2014.01.006.