DOI: 10.1111/cgf.70101
Eurographics Conference on Visualization (EuroVis) 2025
W. Aigner, N. Andrienko, and B. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 3

Necessary but not Sufficient: Limitations of Projection Quality Metrics

A. Machado[1], M. Behrisch[1], A. Telea[1]

[1]Department of Information and Computing Sciences, Utrecht University, Netherlands

Abstract

High-dimensional data analysis often uses dimensionality reduction (DR, also called projection) to map data patterns to human-digestible visual patterns in a 2D scatterplot. Yet, DR methods may fail to show true data patterns and/or create visual patterns that do not represent any data patterns. Projection Quality Metrics (PQMs) are used as objective measures to gauge the above process: the higher a projection's scores in PQMs, the more it is deemed faithful to the data it represents. We show that, while PQMs can be used as exclusion criteria — low values usually mean poor projections — the converse does not always hold. For this, we develop a technique to automatically generate projections that score similar or even higher PQM values than projections created by well-known techniques, but show different, often confusing, visual patterns. Our results show that accepted PQMs cannot be used as an exclusive way to tell whether a projection yields accurate and interpretable visual patterns — in this sense, PQMs play a role akin to that of summary statistics in exploratory data analysis. We also show that not all studied metrics can be fooled equally well, suggesting a ranking of metrics in their ability to reliably capture quality.

CCS Concepts

• ***Mathematics of computing*** → *Dimensionality reduction;* • ***Computing methodologies*** → *Machine learning;* • ***Human-centered computing*** → *Information visualization;*

1. Introduction

Dimensionality Reduction (DR, also called Projection) is a choice tool for depicting high-dimensional data, scaling well in the number of data samples and data dimensions, with ease of use and many implementations. DR scatterplots encode data so that users can perceive *visual patterns* — *e.g.*, grouping, spacing, proximity, outliers — to infer the same aspects for the data distribution.

In general, no DR method can map data with a high intrinsic dimensionality to a 2D scatterplot while perfectly preserving all data structure. Projections can both *fail* to show true data patterns and also *create* visual patterns that do not represent data patterns — that is, at a high level, distort the data depiction [NA18]. Some authors strongly warn about DR distortions affecting exploration tasks [CP23] while others support opposite views [LBK24]. This points to the need for objectively quantifying such aspects in a projection before it is used in practice.

DR addresses this by so-called *Projection Quality Metrics* (PQMs) [JCJ*23, EMK*21], each measuring a specific data-pattern preservation, *e.g.*, neighborhood matching and pairwise distance distortion. Projections with high PQM values are deemed to faithfully reflect data patterns, so, are used for visual exploration; projections with low PQM values are discarded since they can be misleading.

Previous work has provided evidence that some quality metrics can produce *false positives* when evaluating projections [STMT12]. In this work, we show that several commonly-used quality metrics fall short of gauging how well a projection captures data patterns in this same sense; much like summary statistics in exploratory data analysis, PQMs are *necessary* but *not sufficient* to characterize a faithful data depiction [Ans73, MF17]. To this end, we develop an algorithm that generates 'fake' projections containing arbitrary noise-like visual patterns scoring close to, or even higher than, projections created using standard DR techniques. We post-process our fake projections to introduce even more controlled visual patterns and still score high in quality metrics. While users can dismiss noise-like fake projections as unnatural, our post-processed fake projections have higher potential to mislead users to believe that the shown visual patterns actually represent data ones. We next show that it is easier to 'fool' certain metrics than others, suggesting a ranking of PQMs in terms of their ability to gauge a good projection.

2. Background and Related Work

A dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is a collection of n data points in a D-dimensional space ($\mathbf{x}_i \in \mathbb{R}^D$). Points can be optionally paired with a label $c_i \in \{1, \ldots, K\}$. We denote the label set of $\mathbf{X}$ by $C = \{c_1, \ldots, c_n\}$ and the average of a set of m scalars by $\bar{\mu} = \frac{1}{m}\sum_i \mu_i$. Let $\mathbf{Y} = \mathcal{P}(\mathbf{X}) = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be a *projection* of $\mathbf{X}$, where $\mathbf{y}_i \in \mathbb{R}^d$ with $d \ll D$. We next restrict ourselves to $d = 2$ since we consider projections visualized as 2D scatterplots. Except where stated otherwise, our scatterplots show class labels by point colors.

2.1. Dimensionality Reduction

DR techniques create $\mathcal{P}(\mathbf{X})$ by aiming to preserve specific aspects of $\mathbf{X}$ that are relevant to understanding this data. For example, MDS [Kru64] aims to minimize the squared error between pairwise Euclidean distances computed in $\mathbb{R}^D$ with respect to those in $\mathbb{R}^2$. Isomap [TSL00] replaces Euclidean with geodesic distances along an approximation of the manifold containing $\mathbf{X}$. t-SNE [MH08] and UMAP [MH18] aim to preserve neighborhoods in $\mathbf{X}$ and achieve good results for data with high intrinsic dimensionality.

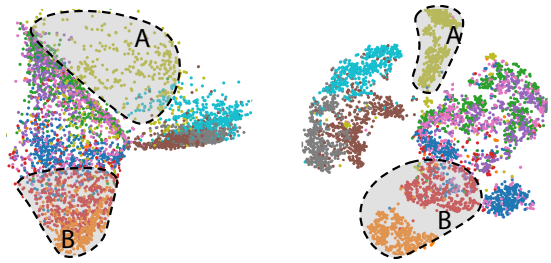

Figure 1: *Isomap (left) and t-SNE (right) projections of the FashionMNIST dataset showing contradictory visual patterns. In (A), Isomap shows same-class points spread out, suggesting they are not very similar; t-SNE packs these points tightly, suggesting the opposite. In (B), Isomap shows the red and orange classes overlapping; t-SNE shows these classes clearly separated.*

Users typically study $\mathcal{P}(\mathbf{X})$, with points optionally enriched to show a few dimensions via color, size, shape, or annotations, looking for salient *visual patterns*. These include (1) clusters of densely packed points with similar values for a dimension, shown, *e.g.*, via color coding; (2) clusters with different point densities; (3) overlaps of clusters with different point colors; (4) clusters of different elongations or shapes; or (5) outlier points far from the scatterplot bulk. Such visual patterns have been studied in detail by *scagnostics* works for selecting informative scatterplots [TT88, WAG05, WAG06, YWS*14]. Visual patterns help inferring data insights, *e.g.*, (1) similar samples sharing a common (dependent) attribute, *e.g.*, label; (2) data subsets of different internal similarities; (3) mixed distribution modes yielding different values for a dependent attribute; (4) latent variables along which data has different variance; and (5) samples not obeying a standard distribution. Many projection techniques exist with different optimization aims and parameters, so one can easily create many different scatterplots $\mathcal{P}(\mathbf{X})$ from the same data $\mathbf{X}$ [WVJ16, VBF21, CHAS18]; some authors even proposed to *control* visual patterns in projections [MTB24]. This has two direct implications for our work, as follows.

Picking a DR technique: To explore a dataset $\mathbf{X}$, we can use several techniques $\mathcal{P}_i$, *e.g.*, MDS and t-SNE. For this, we must know which data patterns (and how well) each $\mathcal{P}_i$ preserves. Preserving a data pattern means mapping it to a specific *visual pattern* in the scatterplot. If we see that visual pattern, we can 'reverse' the mapping to infer the associated data pattern [Mun14, Tel14]. Yet, visual patterns created by different $\mathcal{P}_i$ might contradict each other (see Fig. 1). As we generally do not know how techniques $\mathcal{P}_i$ *exactly* map all types of data patterns to visual ones, how to **pick** a 'suitable' $\mathcal{P}_i$?

Validating new techniques: When developing a new projection technique $\mathcal{P}_{new}$, we need to objectively **compare** $\mathcal{P}_{new}$ to existing techniques $\mathcal{P}_i$ to tell whether $\mathcal{P}_{new}$ better maps some data patterns to visual ones than $\mathcal{P}_i$; how much better that is; and for which patterns.

2.2. Projection Quality Metrics

Projection Quality Metrics (PQMs) are the currently accepted way to address the above two points. PQMs are functions that measure how well some data patterns in $\mathbf{X}$ are mapped to some visual patterns in $\mathbf{Y} = \mathcal{P}(\mathbf{X})$. **Global** PQMs do this by computing a single score

$$z = \mathcal{M}(\mathbf{X}, \mathbf{Y}, C). \tag{1}$$

For example, Normalized Stress [JCC*11] computes the distortions in pairwise distances introduced by $\mathcal{P}$ *vs* the same distances in $\mathbf{X}$. The Procrustes statistic [GR09] measures how well $\mathcal{P}$ represents the data after rescaling, rotating, and translating neighborhoods in $\mathbf{Y}$ to match the corresponding ones in $\mathbf{X}$. Distance Consistency [SNLH09] assesses visual separation as the fraction of points in $\mathbf{Y}$ that are closer to the centroid of their own class than to that of another class. **Local** quality metrics measure how well a specific data pattern at (or close to) data point $\mathbf{x}_i$ is preserved by the scatterplot points at, or close to, the projection $\mathbf{y}_i$ of $\mathbf{x}_i$, outputting *per-data point* scores as

$$\mathbf{Z} = \mu(\mathbf{X}, \mathbf{Y}, C), \mathbf{Z} = \{\mu_i\} \in [0,1]^n. \tag{2}$$

Local metric scores μ_i can be aggregated to a global average score $\bar{\mu} = \frac{1}{n}\sum_i \mu_i(\mathbf{X}, \mathbf{Y}, C)$. Trustworthiness and Continuity [VK06] define $\mathbf{S}_D^k$ and $\mathbf{S}_2^k$ — the k-nearest neighbors of $\mathbf{x}_i \in \mathbf{X}$, respectively $\mathcal{P}(\mathbf{x}_i) \in \mathcal{P}(\mathbf{X})$ — and compute how many points in $\mathbf{S}_D^k$ are in $\mathbf{S}_2^k$ and how many points in $\mathbf{S}_2^k$ are in $\mathbf{S}_D^k$, respectively. Neighborhood Hit [PNML08] measures how many points in $\mathbf{S}_2^k$ have the same label as each $\mathbf{x}_i \in \mathbf{X}$. Other local metrics include projection precision score [SvLB10], stretching and compression [Aup07, LA11], cluster separation metrics [SA15, STMT12], and the Shepard diagram [SC88, JCC*11]. Tens of such metrics exist for projections. For recent surveys and implementations, we refer to [EMK*21, JCJ*23].

We draw a distinction between PQMs and Visual Quality Metrics (VQMs) [BS06]. While the latter can be used to assess the purely-visual patterns of a scatterplot — which might represent a projection $\mathcal{P}$ — they are not adequate to judge $\mathcal{P}$ *itself*, since projections need to preserve patterns *from the data*; *e.g.*, if the data presents poor class separation, this should be reflected in $\mathcal{P}$, leading to a low VQM value for that pattern, even if $\mathcal{P}$ has good quality.

2.3. Are Quality Metrics Enough?

PQMs have several advantages for assessing projections: they are quantitative (and typically deterministic); they evaluate easily, rapidly, and automatically. They allow a simple ranking of projections with respect to quality [EMK*21]: If $\mathcal{P}_1$ scores higher than $\mathcal{P}_2$ on several such metrics, then $\mathcal{P}_1$ is found better than $\mathcal{P}_2$ (relative assessment); the closer metric values are to the maximum, the better a projection is (absolute assessment). Such criteria help filter out or rank projections before users examine them [JJ09, Hub85, WM17] and to holistically evaluate projection techniques [MBC*23, EMK*21]. Yet, PQMs do not *truly* capture the essence of a good projection, namely its ability to consistently map given *data patterns* to given *visual patterns*, due to several factors:

Level of detail: Current PQMs measure how $\mathcal{P}$ works *locally* or *globally* (see Sec. 2.2). Using only such levels is not enough since humans examine scatterplots (or actually any image) over *many* scales [AEM11, SA15, EMK*21]. Most PQMs model pattern scale by the size of k-nearest neighborhoods in data or projection spaces; or by clustering the data or projection, *e.g.*, by k-means and analyzing the resulting so-called motifs [SSB*16]. Such relatively simple approaches cannot capture the full richness of pattern *shapes* that humans see, and reason about, when viewing a projection scatterplot. In fact, finding *all* such patterns in a plot $\mathbf{Y}$ is an open problem. Also, we lack ways to find which data patterns are *missing* from a projection $\mathbf{Y}$. We will show that we can add to a projection salient, but arbitrary, visual patterns, on different scales, without changing its computed PQMs. This proves our point that current PQMs are not enough to fully capture projection quality.

Perception studies: Several studies have been conducted to find

which quality metrics better align with how humans perceive projections [SNLH09, SA15]. This typically uses a correlation proxy: if users are more likely to pick a projection when a given metric is high, one may infer that such a metric aligns well with the human processing of projections. Such work is often task-based, *i.e.*, uses a scatterplot to solve a given problem or answer a given question. Such tasks include the identification of class clusters and their visual separation [EMdSP*15, SA15]. Tatu et al. [TBB*10] studied the perceived quality of 2D projections based on cluster separability and cluster density. Rensink et al. [RB10] studied how correlation is perceived in similar scatterplots. Other authors studied metrics that capture scatterplot perceptual similarity [PKF*16, DW14, AEM11]. While useful to understand how humans interpret visual patterns in scatterplots, such studies do not answer how well these visual patterns actually capture *data patterns.*

Continuity: A metric that aims to rank some phenomenon on a *continuous* scale should itself be continuous [AY79]. This property has been studied earlier under the name 'stability' for treemaps [VSC*20], decision maps [OEJT23], projections [VGdS*20, BTT22], and inverse projections [EAS*23]. In our context, continuity means that small changes Δ in a projection should imply small changes in the PQM of that projection:

$$\mathcal{M}(\mathbf{X}, \mathcal{P}(\mathbf{X}) + \Delta) \approx \mathcal{M}(\mathbf{X}, \mathcal{P}(\mathbf{X})) + \mathcal{O}(\|\Delta\|).$$

The further we are from this goal, the less can $\mathcal{M}$ tell something *useful* about $\mathcal{P}(\mathbf{X})$. Indeed, if $\mathcal{M}$ strongly changes while $\mathcal{P}(\mathbf{X})$ stays about the same, then its values cannot characterize $\mathcal{P}(\mathbf{X})$ robustly — see the work of Wang et al. [WWL*20]. Conversely, if $\mathcal{P}(\mathbf{X})$ changes a lot (large Δ), then users will see different visual patterns in it, and in turn infer different data patterns. However, if $\mathcal{M}$ gives about the same value for the changed projection, then it will be wrongly labeled as 'showing the same'. In our work, we will show that we can achieve the latter for many existing PQMs, which questions their ability to capture projection quality.

3. Fooling Projection Quality Metrics

While PQMs have several limitations (Sec. 2.3), projections are typically assessed by using precisely such metrics. Extensive evaluations have shown that projections scoring *low values* of PQMs are practically useless for data exploration [EMK*21]. Indeed, if a projection cannot preserve even the *basic* patterns current PQMs capture, then it is likely a poor projection. The converse is unclear: *Do* high values *for PQMs mean that the visual patterns in a projection map data patterns clearly and accurately?* To show this is not the case, we propose a systematic procedure to create projections that contain a scale of visual patterns which make them hard to interpret, or are very likely misleading, and still obtain high PQM values. Just like summary statistics for data, PQMs do not tell the whole story about a projection's ability to map data patterns [MF17, Ans73]. While PQM's use as exclusion criteria is justified — low PQM values mean poor projections — the converse does not generally hold.

Our approach works as follows (see also Fig. 3): Given a dataset $\mathbf{X}$ and its projection $\mathbf{Y} = \mathcal{P}(\mathbf{X})$ computed by any user-chosen technique $\mathcal{P}$, we aim to modify, or 'fool', $\mathcal{P}(\mathbf{X})$ in random or determined ways while keeping $\mathcal{M}(\mathbf{X}, \mathcal{P}(\mathbf{X}))$ constant or even increasing it. This yields 'fake' projections of $\mathbf{X}$ that show arbitrary visual patterns in $\mathbf{Y}$ with no PQM cost decrease; hence, the said PQMs are not good to gauge how well visual patterns in $\mathbf{Y}$ capture data patterns in $\mathbf{X}$. One way to distort $\mathbf{Y}$ while increasing PQM $\mathcal{M}$ would be to use some sort of gradient ascent: compute $\nabla_{\mathbf{Y}}\mathcal{M}(\mathbf{X}, \mathbf{Y})$ and move points in $\mathbf{Y}$ according to it. Yet, many PQMs are not differentiable, so we cannot directly compute $\nabla_{\mathbf{Y}}\mathcal{M}$. To address this, we

1. approximate $\mathcal{M}$ with a continuous model differentiable with respect to $\mathbf{Y}$: a feed-forward neural network Q_θ with parameters θ minimizing a regression loss;
2. use (as approximation target μ) the local version of $\mathcal{M}$ (Eqn.2) as a proxy, since we need one target value per $(\mathbf{x}_i, \mathbf{y}_i)$ pair during training, whereas a global metric such as Shepard goodness yields one target value for the entire $\mathcal{P}(\mathbf{X})$. This limits our method to target metrics of the type given by Eqn. 2. The loss function for Q_θ then reads

$$\mathcal{L}_{\text{Metric}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \left(Q_\theta(\mathbf{x}_i, \mathbf{y}_i) - \mu_i(\mathbf{X}, \mathbf{Y}, C)\right)^2.$$

This learning setup does not capture a PQM in a way that generalizes to new projections of the same data. Since PQMs are typically discontinuous or at least non-differentiable, changes to one $\mathbf{y}_i$ do not smoothly change the value of $\mathcal{M}$. We do not propose, and do not need, a continuous approximation of PQMs and refrain from using Q_θ as a *true* approximation to $\mathcal{M}$. Figure 2 shows this for the μ =Trustworthiness metric. We see that μ and Q_θ have different variations when we change $\mathcal{P}(\mathbf{X})$. Despite this, the approximation Q_θ is still useful for our purposes (see Sec. 3.1).

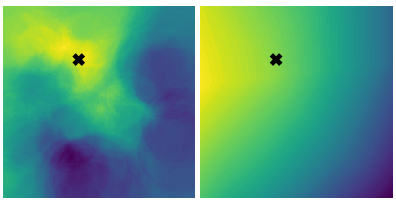

Figure 2: *Left: Each pixel* $\mathbf{p}$ *shows the* $\mathcal{M}$ *= Trustworthiness value of a projection obtained by moving* a single *point (marked X) of a reference* $\mathcal{P}(\mathbf{X})$ *to* $\mathbf{p}$*. Here,* $\mathcal{P}$ *= t-SNE and* $\mathbf{X}$ *= MNIST. Right: predicted* Q_θ *after training to approximate Trustworthiness for this projection. While most of the true behavior of* $\mathcal{M}$ *is not captured by* Q_θ*, the approximation captures correctly the high/low value zones.*

3.1. Pattern-destructive phase

We set out to create projections $\hat{\mathbf{Y}}$ which are evidently poor but still score well according to a target metric μ by using a modified version of NNP [EHT19]. NNP outputs a projection $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$ by training a neural network P_ϕ to mimic a reference projection $\mathcal{P}$ of a subset $\mathbf{X}' \subset \mathbf{X}$. NNP's loss function is a simple Mean Squared Error

$$\mathcal{L}_{\text{NNP}}(\phi) = \frac{1}{|\mathbf{X}'|}\sum_{i=1}^{|\mathbf{X}'|} \left\|P_\phi(\mathbf{x}'_i) - \mathcal{P}(\mathbf{x}'_i)\right\|^2.$$

Beyond what NNP does, we wish to create a projection that has high values of one or more PQMs *and* remove visual patterns that exist in $\mathbf{Y}$ (thereby creating type-I errors). For this, we pick a metric μ to fool and use Q_θ, trained to approximate μ, as an extra loss to *maximize*. This extra loss can destabilize training, likely due to our approximator inability to fully capture μ. To fix this, we add an auxiliary learning goal that reconstructs $\mathbf{X}$ from the learned projection $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$, following [MTB24, MSJG15]. For this, we augment NNP with a network $\mathcal{I}_\psi : \mathbb{R}^2 \to \mathbb{R}^D$ with parameters ψ outputting $\hat{\mathbf{X}} = \mathcal{I}_\psi(P_\phi(\mathbf{X}))$ and minimizing a *reconstruction* loss.

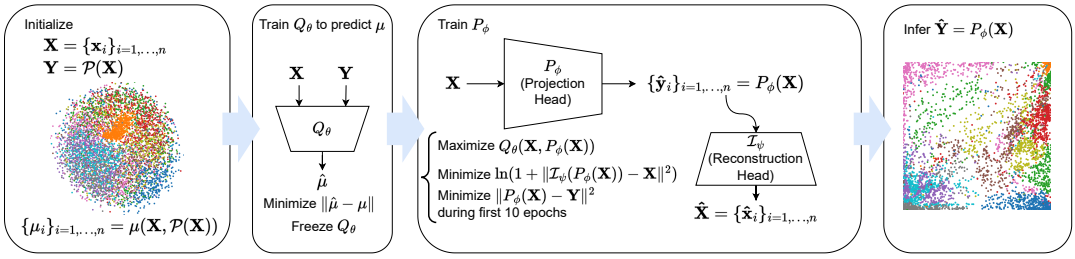

Figure 3: *Architecture of our projection fooling method. From a given dataset* $\mathbf{X}$ *and a reference projection* $\mathbf{Y} = \mathcal{P}(\mathbf{X})$*, we compute a local quality metric* μ *and train a neural network* Q_θ *to infer* μ *at all points* $\mathcal{P}(\mathbf{X})$*. We then freeze* Q_θ *and use it, with* $\mathbf{X}$ *and* $\mathbf{Y}$*, to train the neural network* P_ϕ *to project* $\mathbf{X}$ *as to maximize* Q_θ *while minimizing a reconstruction error. We also force* P_ϕ *to learn the reference projection* $\mathbf{Y}$ *for the first 10 training epochs. After training, we generate the fooling projection* $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$ *and discard* $P_\phi, \mathcal{I}_\psi$*, and* Q_θ*.*

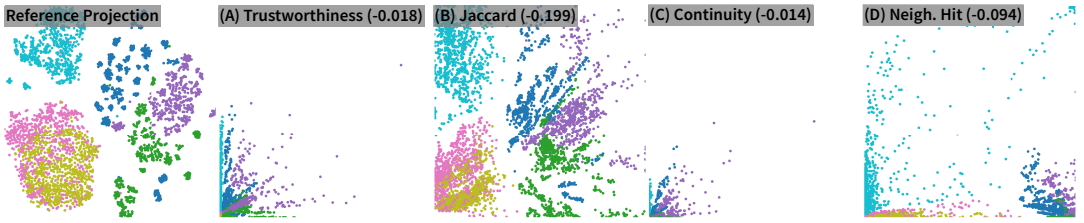

Figure 4: *Fooled projections* $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$ *for the HAR dataset starting from the same t-SNE projection* $\mathcal{P}$ *(left) for different target metrics* μ*. We annotate each projection with its target* μ *and the difference in* $\mathcal{M} = \bar{\mu}$ *using* $\mathcal{P}$ *as reference (in brackets). We see only slight losses in Trustworthiness and Continuity despite the projections looking degenerate. We see higher losses in Neighborhood Hit and Jaccard Similarity. For the latter case, the visual patterns we obtain are more in line with those usually expected in projections.*

The complete loss to be minimized reads

$$\mathcal{L}_{\text{Fooler}}(\phi, \psi) = -\frac{1}{n}\sum_{i=1}^{n}\Big(Q_\theta(\mathbf{x}_i, P_\phi(\mathbf{x}_i)) - \kappa\|P_\phi(\mathbf{x}_i) - \mathbf{y}_i\|^2 - \ln(1 + \|\mathcal{I}_\psi(P_\phi(\mathbf{x}_i)) - \mathbf{x}_i\|^2)\Big), \tag{3}$$

where $x \mapsto \ln(1+x)$ squashes down high values of the reconstruction loss. We set $\kappa = 10$ for the first 10 training epochs and next set it to zero. This makes the model first learn the reference projection, much like NNP; next, the model can freely modify the learned projection to get higher μ values as predicted by Q_θ. This process naturally generates projections with poor visual patterns, since point positions are optimized independently, and according to Q_θ, an approximation of limited accuracy. Figure 3 shows our main fooling pipeline.

The differentiability of Q_θ with respect to the projection $\mathbf{Y}$ enables guiding P_ϕ towards regions with higher Q_θ values. This is because gradients can flow *through* Q_θ into P_ϕ; this would not happen if we used μ (or $\mathcal{M} = \bar{\mu}$) instead of Q_θ since μ is not differentiable. Indeed, the first term on the right hand side of

$$\frac{\partial \mu}{\partial \phi} = \frac{\partial \mu}{\partial \mathbf{y}_i}\Big|_{\mathbf{y}_i = P_\phi(\mathbf{x}_i)} \frac{\partial \mathbf{y}_i}{\partial \phi}$$

does not exist, while $\partial Q_\theta / \partial \mathbf{y}_i$ (using $Q_\theta \approx \mu$) is always defined.

After training, we run P_ϕ in inference mode with $\mathbf{X}$ as input to get our object of interest $\hat{\mathbf{Y}}$ and *discard* all neural networks. Figure 4 shows examples of our fooled projections $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$ for the HAR dataset and four quality metrics. The learned projections for Trustworthiness and Continuity are visually poor (Fig. 4a,c): Points collapse towards a scatterplot corner; class separation is low; yet, both target metrics suffer only minor penalties of 0.02 and 0.01, telling that they are *not* sensitive to the artificial patterns our fooling introduces. The learned projection for Neighborhood Hit is also degenerate (Fig. 4d) but less than (a) or (c); here, the target metric suffers a heavier penalty. Finally, when aiming to fool the Jaccard metric, we see the least degenerate projection of all (Fig. 4b); quality has now a heavy drop of almost 0.2.

3.2. Pattern-constructive phase

Figure 4 shows that P_ϕ creates projections with random visual patterns but with PQM values close to those of the ground-truth projection. Yet, one could argue that users might quickly feel that such projections are fabricated and, even if they have high quality values, dismiss them from further exploration. We next present three postprocessing techniques that modify fooled projections to keep high PQM values *and* contain visual patterns typically seen in good-quality projections that do not reflect the underlying data (thereby creating type-II errors).

Postprocessing aims to counteract what a salient aspect of $\hat{\mathbf{Y}}$ — the collapse of the scatterplot onto the projection boundaries (Fig. 4). We cannot change $\hat{\mathbf{Y}}$ arbitrarily since we want to affect PQMs as little as possible. This suggests some sort of 'semi-rigid' scatterplot transformation that preserves neighborhoods. Our postprocessing techniques all work by building a graph from the projection $\hat{\mathbf{Y}}$ or from the dataset $\mathbf{X}$ (see Fig. 5) in different ways, as explained next.

Data Nearest Neighbors: We build a weighted k-nearest neighbors graph G_{DataNN} of $\mathbf{X}$, with weights given by Euclidean distances between data points — we use $k = 7$ in our experiments.

Projection Nearest Neighbors: We build the unweighted k-nearest neighbors graph G_{ProjNN} of $\hat{\mathbf{Y}}$ — we use $k = 7$ in our experiments. We use an unweighted graph as the distances in projection space are not necessarily meaningful after the fooling procedure.

Delaunay Triangulation: We build the Delaunay triangulation $G_\triangle$ of the projection $\hat{\mathbf{Y}}$ forcing simplicial output.

On all these graphs, we apply 5 iterations of the standard force-directed layout [FR91] to tweak point positions in $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$. For

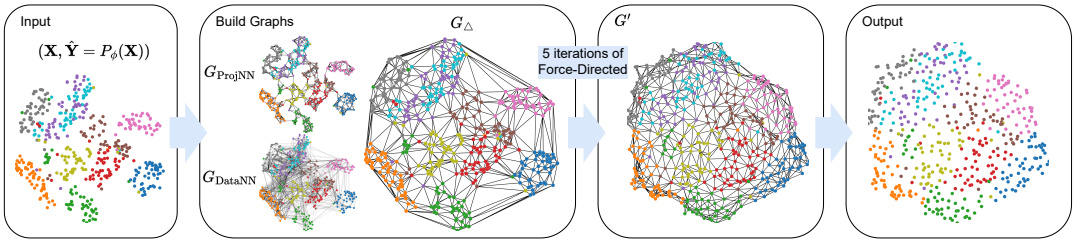

Figure 5: *Architecture of our post-processing pipeline. Starting from a given dataset* $\mathbf{X}$ *and fooled projection* $\hat{\mathbf{Y}}$*, we build three different graphs —* G_{ProjNN}*,* G_{DataNN}*, and* $G_{\triangle}$*. Each of those three graphs is then used to perform 5 iterations of a force-directed layout, producing three different outputs. To avoid clutter, we visualize here only the output using* $G_{\triangle}$*.*

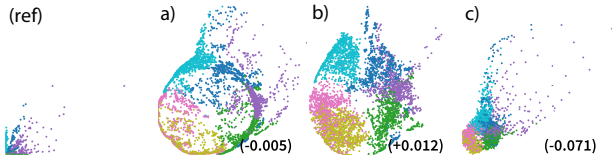

Figure 6: *Postprocessing with (a)* G_{ProjNN}*, (b)* G_{DataNN}*, (c)* $G_{\triangle}$ *of the Continuity-fooling projection of the HAR dataset (ref). Annotations show the difference in Continuity* vs *the unprocessed output. While (c) shows a more important loss, (a) shows barely any change in Continuity. (b) shows an* increase *in Continuity and visual patterns similar to those typical in good projections.*

the first two graphs, we use the default ideal edge length $l = 1/\sqrt{n}$; for $G_{\triangle}$, we set l to the 40th-percentile of pairwise projection distances. Figure 6 shows the results of our three postprocessing methods for the HAR projection (Fig. 4c). Scatterplot points spread better as compared to the unprocessed version. The resulting scatterplots also appear more similar to typical projections in DR literature. Large-scale visual patterns appear, *e.g.*, circular structures which could fool users to believe that they are data-driven. Postprocessing only lightly affects the PQMs (see Fig. 7) and can even *increase* these. All in all, we see that introducing *significant* visual patterns that are unrelated to the data is not detected by the evaluated PQMs.

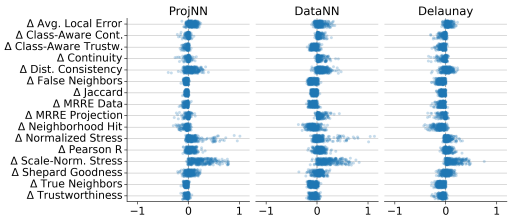

Figure 7: *Effect of post-processing on studied quality metrics across our experiments. We measure the difference* $\Delta M = M_{Post} - M^{\dagger}$ *between metrics learned by our fooling pipeline* $M^{\dagger}$ *and their values after post-processing* M_{Post}*. Points are mostly concentrated around 0, meaning there is no strong (negative or positive) influence in terms of quality in the constructive phase of the pipeline.*

4. Evaluation

We evaluate the ability of our fooling mechanism to create projections with high values of various PQMs. As many such PQMs exist, we focus next on metrics which are

- **easy to compute:** metrics should have few hyperparameters;
- **deterministic:** for a given $\mathbf{X}, \mathbf{Y}$, and C, the metric value should be the same; non-deterministic metrics are less useful in practice for assessing the quality of a given study object;
- **common:** fooling a metric well-established in DR literature has arguably stronger impact than fooling an infrequently used one.

With these criteria, we selected 17 quality metrics (Tab. 1). Their definitions, value ranges, and optimal values are listed in the supplementary material.

Table 1: *Projection quality metrics used in the evaluation.*

Metric	Parameters	Introduced in
Average Local Error	—	[MCMT14]
Continuity and Trustworthiness	k	[VK06]
Class-Aware Continuity and Trustworthiness	k	[CPA*20]
Distance Consistency (DSC)	—	[SNLH09]
Proportion of False (resp. True) Neighbors	k	[MCMT14]
Jaccard Similarity of Neighbor Sets	k	[Jac01]
Mean Relative Ranking Errors	k	[LV09]
Neighborhood Hit	k	[PNML08]
Normalized Stress	—	[JCC*11]
Pearson Correlation of Distances	—	[GZZ05]
Procrustes Statistic	k	[GR09]
Scale-Normalized Stress	—	[SMK24]
Shepard Goodness	—	[SC88]

4.1. Experiment Setup

Projections: We tested four commonly-used projection techniques, namely t-SNE [MH08], UMAP [MH18], MDS [Kru64], and Isomap [TSL00]. These techniques are chosen due to their wide adoption, readily-available implementation [PVG*11], and diversity in terms of mathematical grounding and algorithm design.

Datasets: We used six datasets (Tab. 2) with a stratified subsample of at most 5000 samples from each dataset in all experiments.

Target metrics: For the *local* metric to fool (Fig. 3, Sec. 3), we explore four cases: Continuity, Jaccard set similarity, Neighborhood hit, and Trustworthiness, and also fooling all four metrics jointly. All these metrics have one hyperparameter, the neighborhood size k, which we set to $k \in \{1, 7, 21, 51\}$. In total, we consider 6 datasets $\times$ 4 projections $\times$ $(4 + 1)$ fooling targets $\times$ 4 hyperparameter settings $= 480$ total experiments. For each setting, we measure the 17 chosen metrics — identically setting k when applicable — leading to 8160 measured values in total. Given this amount, we next show different kinds of aggregation and focus on specific instances. Further details are given in the supplementary material. All results can be explored interactively online [MBT24b] (see also Fig. 9).

Implementation details: We have a large experimentation space, so computing metrics must be fast. We implement the 17 studied metrics using TensorFlow [ABC*16] as a PyPI [MBT24a] package. Our code is openly available [MBT24c].

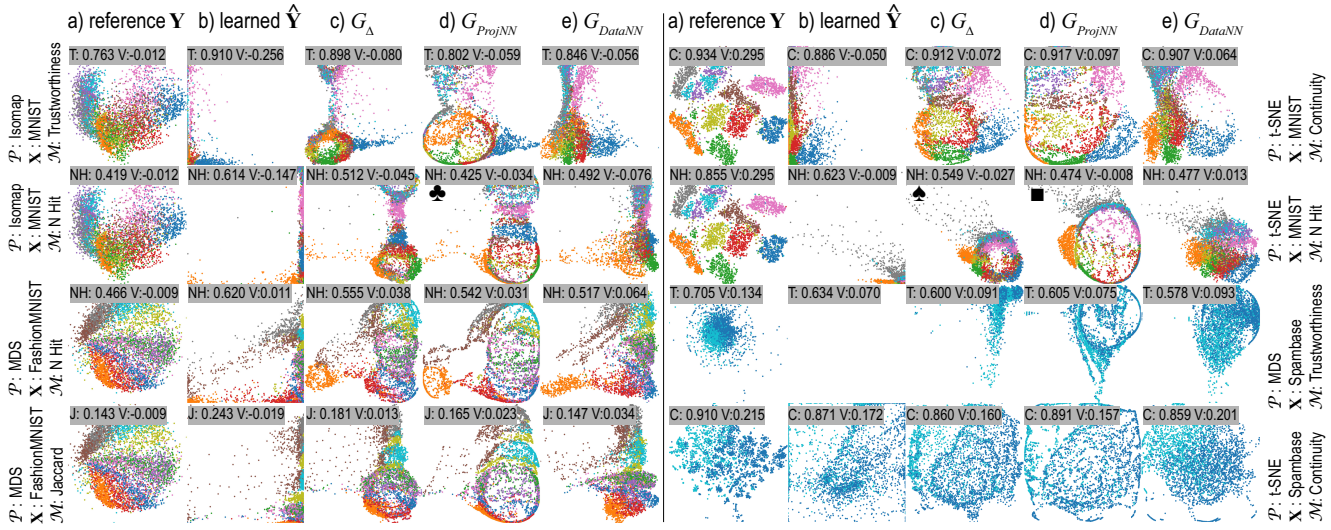

Figure 8: *Samples of our fooling pipeline for different projection techniques $\mathcal{P}$, datasets $\mathbf{X}$, and target metrics $\mathcal{M}$. Each projection is annotated with the value of the target metric $\mathcal{M}$, as well as of CIVM [LLX*10], a VQM measuring cluster preservation (V in each plot). For further discussion on the elements marked with glyphs, see Section 4.2.*

Hyperparameter settings: All our studies use the same hyperparameters. We train all our neural networks with a learning rate of 10^{-3}, mini-batches of 256 data points, and the Adam [KB15] optimizer, 100 epochs (Q_θ), and 1000 epochs (P_ϕ together with $\mathcal{I}_\psi$). Full architecture details are given in the supplementary material.

Table 2: *Datasets used for experimentation.*

Dataset	# Dimensions (D)	# Classes (K)
FashionMNIST [XRV17]	784	10
Human Action Recognition (HAR) [AGO*12]	561	6
MNIST [LBBH98]	784	10
Reuters [Tho17]	5000	6
Spambase [HRFS99]	57	2
USPS [Hul94]	256	10

4.2. Studying the created projections

We start exploring by looking at the outputs generated by our fooling pipeline (fooled projections and their quality scores) and compare these with the reference $\mathcal{P}$. Figure 8 shows a selected subset of such results. All experiments are available in the online material.

Figure 8a shows the projections $\mathbf{Y}$ used as reference by our fooling pipeline, each annotated with the value of the metric targeted for fooling. Columns (b) show the learned $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$ before post-processing. These projections have undesirable visual properties (point overlap, cluster collapse, and points with extreme x and/or y coordinates), so they clearly cannot preserve any meaningful data patterns — they actually show no human-digestible visual patterns overall. Yet, the evaluated metrics score most of these fake projections *highly* — see columns (b) — often *higher* than the reference projection. Columns (c-e) show the outcomes of our post-processing $G_\triangle$, G_{ProjNN}, and G_{DataNN}. The post-processed projections all look more plausible, *i.e.*, contain visual patterns one typically sees in DR scatterplots. Yet, these patterns are *misleading*: For example, the plot for Isomap-MNIST-NH created by $G_{\text{ProjNN}}(\clubsuit)$ looks like a symmetric dumbbell while the ground-truth $\mathbf{Y}$ shows there is no such structure in MNIST; the plot for t-SNE-MNIST-NH created by $G_{\text{ProjNN}}(\blacksquare)$ shows a circular structure which, again, does not exist in MNIST; the plot for t-SNE-MNIST-NH created by $G_\triangle(\spadesuit)$ shows a cloud of outliers top-left which do not exist in MNIST. Post-processing weakly affects quality values, sometimes positively, sometimes negatively — another instance of insensitivity of the quality metrics to introducing arbitrary patterns into a projection. Indeed, of all post-processing approaches, only G_{DataNN} uses ground-truth data, and still does not always lead to a gain in quality metrics.

4.3. Correlations in fooling behavior

We next explore all metric values of the fooled projections $\hat{\mathbf{Y}}$ before post-processing (Sec. 3.2) to find which metrics tend to move the same way during fooling. For this, we measure all 17 metrics for a reference projection $\mathbf{Y} = \mathcal{P}(\mathbf{X})$, obtaining $M = [\mathcal{M}_1(\mathbf{X},\mathbf{Y}),\ldots,\mathcal{M}_{17}(\mathbf{X},\mathbf{Y})]$. We run our fooling pipeline with a fooling target $\mathcal{M}^\dagger$ and obtain $\hat{\mathbf{Y}} = P_\phi(\mathbf{X})$. We measure again all 17 metrics on $\hat{\mathbf{Y}}$, obtaining $M^\dagger = [\mathcal{M}_1(\mathbf{X},\hat{\mathbf{Y}}),\ldots,\mathcal{M}_{17}(\mathbf{X},\hat{\mathbf{Y}})]$. The value $\Delta M = M^\dagger - M$ tells us how much fooling the *single* target $\mathcal{M}^\dagger$ affected *all* 17 metrics. Doing this for all experiment settings yields a set of 480 different $(M, M^\dagger, \Delta M)$. With these data, we set out to answer two questions:

1. Which metrics behave *similarly* across projections? Figure 10a answers this by showing the correlations in M.
2. Which metrics does our fooling affect *as a group*, *i.e.*, when fooling some $\mathcal{M}^\dagger$, are consistently fooled too. Figure 10b answers this by showing the correlations in ΔM.

Yet, our 17 metrics have values inconsistent in meaning: Stress and Procrustes range from 0 (best) to infinity (worst); Trustworthiness and Continuity range in 0 (worst) to 1 (best). This creates negative correlations where pairs of metrics would agree on improvements. To fix this, we re-map all metrics so higher values mean better quality (Tab. 3). Any increase $\Delta M_i > 0$ now means an increase in quality. In Figure 10a, blue (resp. red) show positively (resp. negatively) correlated metric values for the *reference* projections. That is, a blue cell for a metric pair tells that these metrics move in roughly the same way — a projection scoring, say, high for Trustworthiness also scores high on Continuity. We see that most of

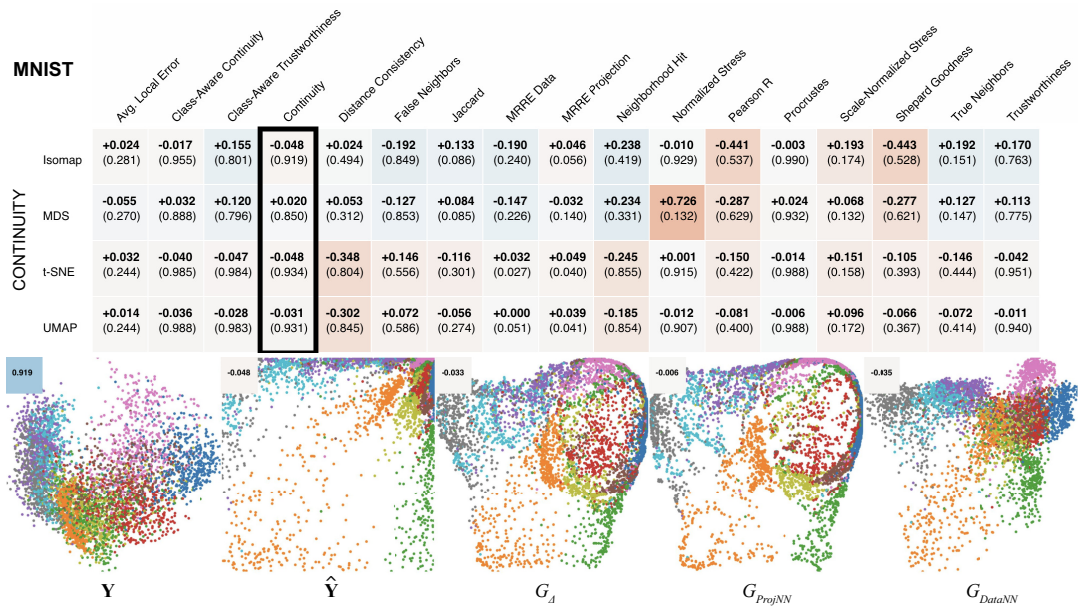

Figure 9: *Exploratory online tool for studying projection fooling results. Top: Results of fooling Continuity (outlined in black) for four projection techniques (Isomap, MDS, t-SNE, UMAP) on the MNIST dataset. Table cells are colored to show PQM gain resp. loss* vs *the reference projection (blue: fooling increases PQM; white: similar values; red: fooling decreases PQM). Fooling yields practically identical Continuity, but also very similar values for the other 17 metrics, which we did* not *optimize for. The main losses for these 17 metrics are for Stress (MDS), Distance Consistency (t-SNE, UMAP), and Pearson R and Shepard (Isomap). Bottom, from left to right: Reference projection* $\mathbf{Y}$*, learned projection* $\hat{\mathbf{Y}}$*, and post-processed outputs* $G_{\triangle}$*,* G_{ProjNN}*, and* G_{DataNN}*, annotated with the difference in Continuity* vs $\mathbf{Y}$*. Fooling produces both random-like patterns (*$\hat{\mathbf{Y}}$*) but also salient structures which actually have no meaning (*G_{ProjNN}*,* G_{DataNN}*).*

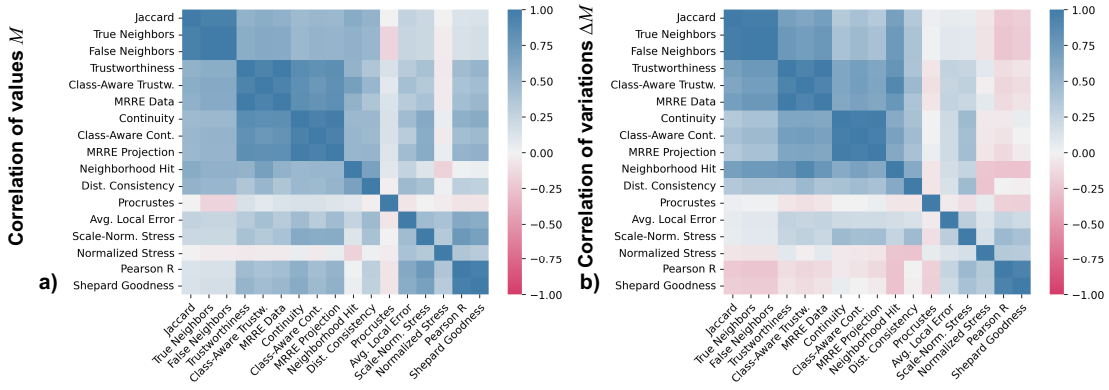

Figure 10: *Correlations in metric values M (a) and value variations ΔM (b). Blue: metrics that vary in the same direction (a), resp. metrics showing the same fooling behavior (b). We use remapped values (see Tab. 3) such that positive metric values always mean high quality.*

Table 3: *Re-mapping metric values for comparison purposes. The best value in each scenario is shown in bold.*

Range of $m = \mathcal{M}(\mathbf{X}, \mathbf{Y})$	Transformation	Range of $f(m)$
$[\mathbf{0}, +\infty)$	$f(m) = -m$	$(-\infty, \mathbf{0}]$
$[a, \mathbf{b}]$	$f(m) = (m-a)/(b-a)$	$[0, \mathbf{1}]$

the metric pairs are positively correlated — meaning that high values (after re-mapping) consistently indicate high quality. This provides us the baseline for addressing question 2 as follows. Figure 10b shows how fooling a single metric affects the other metrics. Blue (resp. red) cells tell that our fooling produces similar (resp. opposite) variations in metric values. We see that, even though most experiments target fooling one metric, this affects other metrics as well. Metrics appear split into two types: The large blue block (top-left in Fig. 10b) shows metrics that can be easily fooled together – we can generate fake projections having large values of any metric in this block by fooling a single metric herein. More interestingly, the light red cells tell us that Normalized Stress, Pearson Correlation, and Shepard Goodness decrease when we try to fool any of the other 17 metrics. Conversely, fooling these three metrics will decrease some of the remaining ones. This means that using metrics from the two types *together* can provide a more robust evaluation of projection quality than using only metrics of a single type.

4.4. Ease of fooling

We next study how well can our fooling yield *higher metric values* while creating arbitrary patterns in projections. For all the metrics we can fool — Continuity, Jaccard, Neighborhood hit, Trustworthiness,

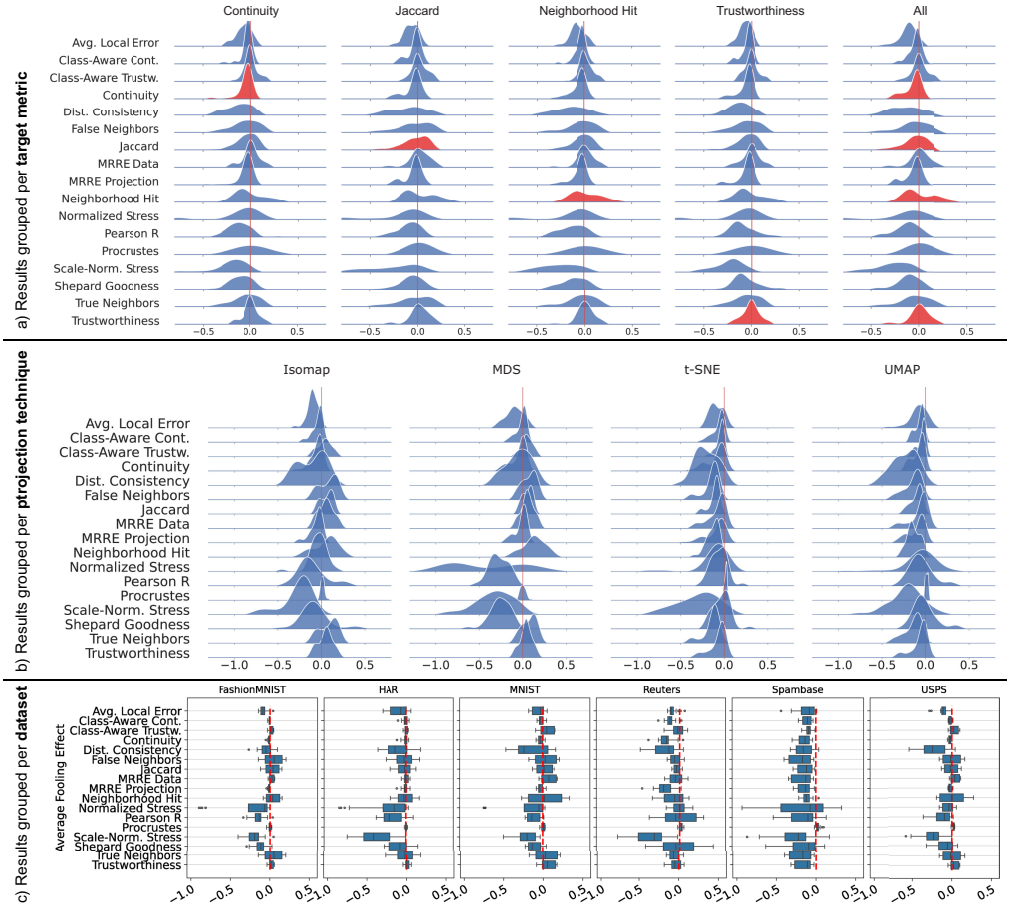

Figure 11: *Effect of fooling (i.e. distribution of ΔM) grouped (see column headers) per target metric (a), per reference projection (b), and per dataset (c). Metrics are remapped (higher = better). In (a), metrics used as fooling target are highlighted.*

and all jointly (Sec. 4.1) — we plot the distribution of the difference ΔM for all measured 17 metrics for all experiments (Fig. 11a). Distributions concentrated around zero tell that the respective metric (see names in the left labels) are kept well irrespective which target metric we fool — *e.g.*, Class-Aware Continuity. Distributions skewed to *right* tell that metric values increase after fooling — so, given our 'higher is better' remapping, fooling actually *increases* quality measured by that metric — *e.g.*, Trustworthiness and Jaccard. *Left*-skewed distributions tell that our fooling slightly loses quality as measured by that metric — *e.g.*, Distance Consistency and Normalized Stress. Figure 2 (supplementary material) shows the same data using box-plots. Overall, these figures tell us that fooling any of the target metrics is quite successful, not only in the target values, but also in affecting all other metrics.

Figure 11b shows the same information grouped per *projection technique*. We do not see any strong bias of success or failure of fooling *vs* a technique. Yet, we see quality losses for stress metrics for MDS after fooling. This is expected: The MDS cost function aims to optimize *precisely* a formulation of stress. Finally, Fig. 11c shows the same information grouped per *dataset*. We do not see any strong bias of success or failure of fooling *vs* a specific dataset. The only exception is Spambase, where quality consistently decreases after fooling. We attribute this to the *low dimensionality* of the data ($D = 57$): the reconstruction head $\mathcal{I}_\psi$ uses a 3-layer neural network with 512 neurons in the last hidden layer, almost 10 times larger than D, leading to an overparameterized regime that causes data points to collapse to the same place in the projection.

4.5. Defining a set of guardrail metrics for projection quality

Our experiments showed that (a) quality metrics score high on projections with random or misleading visual patterns and that (b) we can automatically create such projections. So, how can we *still* use metrics to best gauge projection quality? We saw that not all metrics can be fooled equally well — we can, *e.g.*, often deceive Trustworthiness and Neighborhood Hit but not Shepard Goodness (Fig. 11 and related text). We also saw that many metrics are correlated, both in their values for different projections (Fig. 10a) as well as in their increase or decrease due to fooling (Fig. 10b). Metrics that are highly correlated in both these figures measure, to a degree, the same thing. This motivates us to select a small set of *diverse* quality metrics that cover different aspects of a projection. Scoring well in such metrics should be a stronger indication of projection quality. To create this set, we transform the correlation matrix $\mathbf{C} \in [-1, 1]^{17\times 17}$ of metric values (Fig. 10a) into a distance matrix $\mathbf{D} = \frac{1}{2}(\mathbf{1} - \mathbf{C})$, where $\mathbf{1}$ is a matrix where all entries are 1, such that low distances correspond to highly correlated metrics. We then run HDBSCAN [CMS13] with default parameters using $\mathbf{D}$ as input. This produces four clusters of strongly correlated metrics (Distance Consistency, Neighborhood Hit, Normalized Stress, and Procrustes Statistic are not assigned to clusters):

Table 4: *Quality Metric values (high value = high quality) for the subset of metrics found to be least correlated (see Sec. 4.5). Each row compares the value after fooling with the pre-fooling value (in brackets). We highlight the 2 strongest quality losses per row.*

Dataset	Fooling target	$\mathcal{P}$	Cont.	Trust.	True Neigh.	Pearson r	DSC	Neigh. Hit	Procrustes	Stress
MNIST	Trustworthiness	Isomap	0.870 (0.919)	0.910 (0.763)	0.292 (0.151)	**0.588** (0.768)	**0.272** (0.494)	0.609 (0.419)	-0.963 (-0.990)	-0.921 (-0.929)
		UMAP	0.894 (0.931)	0.931 (0.940)	0.348 (0.414)	0.694 (0.700)	**0.526** (0.845)	**0.669** (0.854)	-0.977 (-0.988)	-0.880 (-0.907)
		t-SNE	0.903 (0.934)	0.935 (0.951)	0.338 (0.444)	0.725 (0.711)	**0.515** (0.804)	**0.675** (0.855)	-0.961 (-0.988)	-0.981 (-0.915)
Spambase	Neighborhood Hit	Isomap	0.729 (0.872)	0.632 (0.709)	0.130 (0.191)	**0.653** (0.817)	**0.483** (0.804)	0.619 (0.765)	-0.928 (-0.983)	-0.375 (-0.547)
		UMAP	0.658 (0.878)	0.555 (0.851)	**0.042** (0.402)	0.598 (0.648)	0.616 (0.782)	0.628 (0.780)	-0.979 (-0.992)	**-1.193** (-0.554)
		t-SNE	0.640 (0.910)	0.574 (0.873)	**0.046** (0.444)	0.591 (0.684)	0.790 (0.790)	0.641 (0.790)	-0.975 (-0.985)	**-1.155** (-0.275)
USPS	Continuity	Isomap	0.921 (0.946)	0.943 (0.855)	0.334 (0.231)	**0.631** (0.847)	0.669 (0.627)	0.736 (0.625)	-1.000 (-0.991)	**-0.978** (-0.883)
		UMAP	0.929 (0.960)	0.933 (0.962)	0.315 (0.461)	0.611 (0.738)	**0.465** (0.949)	**0.742** (0.928)	-0.963 (-0.995)	-0.863 (-0.837)
		t-SNE	0.916 (0.962)	0.926 (0.967)	0.343 (0.486)	0.748 (0.748)	**0.604** (0.948)	**0.737** (0.932)	-0.960 (-0.989)	-0.791 (-0.849)

1. Jaccard, False Neighbors, True Neighbors;
2. Avg. Local Error, Pearson r of Distances, Shepard Goodness, and Scale-Normalized Stress;
3. Class-Aware Continuity, Continuity, and MRRE Projection;
4. Class-Aware Trustworthiness, Trustworthiness, and MRRE Data.

Metrics in the same (resp. different) clusters capture similar (resp. different) quality aspects. We pick one metric per cluster to capture different quality aspects — specifically, True Neighbors, Scale-Normalized Stress, Continuity, and Trustworthiness, given their computing simplicity and/or wide adoption in projection evaluation; to these, we add the four metrics which could not be clustered. We next explore how this set of eight 'guardrail' metrics can reliably judge projection quality. Table 4 compares the pre- and post-fooling values of these metrics, highlighting the two largest losses in each row. Metrics do not operate on the same scale — smaller losses for one metric may 'mean more' than larger losses in other metrics. We see that *no* single metric reliably identifies that a projection is of low quality by returning a low score. In fact, all metrics can even *improve* after our fooling. The metrics showing more dramatic reductions — signaling fooling — are **Distance Consistency** (DSC), **Neighborhood Hit**, the **Pearson correlation** r, and the **True Neighbors** rate. We recommend — within the limitations of our fooling framework — the adoption of this set of quality metrics as a means to increase the reliability of data pattern preservation judgment. Although Stress also signals quality reduction, we do not propose its use due to its unboundedness and scale-sensitivity [SMK24].

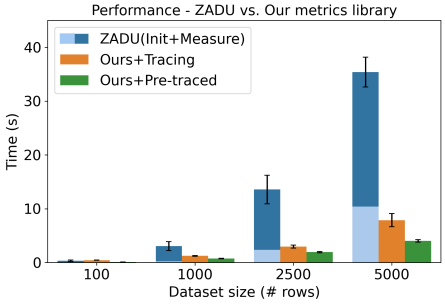

Figure 12: *Performance of our metrics library* vs *ZADU [JCJ*23] on 12 metrics common to both. We are* 30% *slower for datasets of 100 samples due to Tensorflow's overhead, but see speedups of 2.5-4.5× for datasets with* $>1K$ *rows including Tensorflow graph creation cost ('tracing'). Larger speedups (2.8-8.8×) occur when retracing is not needed. Averaged over 7 runs, error bars show* $\pm 1\sigma$.

4.6. Performance

Generating a fooling projection for a dataset from a given projection $\mathcal{P}$ with a target metric $\mathcal{M}$ takes 3-4 minutes on average on a commodity PC. This includes training Q_θ, P_ϕ, and $\mathcal{I}_\psi$, running all three post-processing algorithms, and computing the 17 quality metrics for all generated projections. Figure 12 compares our PQM computing costs with ZADU [JCJ*23], to our knowledge the best effort to provide a comprehensive implementation of PQMs. Our library (Sec. 4.1) is 3 to 9 times faster than ZADU.

5. Discussion

Likelihood: While we can automatically fool PQMs, one may wonder how likely that is to be done in practice. The chances are slim, but not negligible: Most projection algorithms must be initialized to a setting $\mathbf{Y}_0$. Say we initialize t-SNE to a $\mathbf{Y}_0$ that has been optimized to fool Trustworthiness/Continuity — measures of neighborhood preservation, which t-SNE is designed to do well. Such an initialization could make t-SNE get stuck in that local minimum of good neighborhood preservation, but poor visual quality.

Fooling effect: Our fooled projections have a very different visual signature than the reference ones, yet high PQM values. While arguably such projections can lead users to different conclusions about the depicted data than the reference ones, we aim to test this. Since testing this by user experiments is difficult and of limited coverage (in terms of datasets, projections, tasks to be solved, trained users to recruit as participants), we compare our fooled projections with the reference ones by the Clustering Internal Validation Measure [LLX*10] VQM with the Silhouette coefficient. This VQM measures aspects such as compactness and cluster separation. Figures 8 and 14 show the results with the QM and VQM metrics displayed as annotations. Our fooled projections (both before and after postprocessing) have very similar QM values; in contrast, VQM values are very different. This indicates that our fooled projections would convey *different* insights to their users as compared to the reference ones (up to what each VQM measures). We illustrate this in Fig. 13 where we propose a simple task: assess which class has the most *similar* elements — which *visually* corresponds to finding the most tightly-packed cluster. Each projection — MDS and a fooling thereof — shows a different most-dense cluster, illustrating how a user might be led to *distinct* conclusions about the *same* data.

Limitations: While our results show that one can create projections

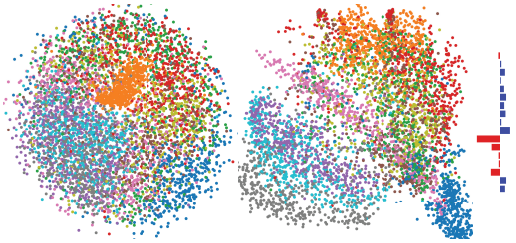

Figure 13: *Two projections of MNIST (left: MDS; right: our fooling projection for that input). For the task of finding which cluster contains the most similar data points, the two images give different answers — left, the orange cluster (top-middle); right, the dark blue cluster (bottom-right). Hence, our fooling may lead users to different conclusions about the same data. The bar plot to the right shows variation in PQMs, most showing* improvements *in the fooled projection; the largest loss is for the Stress metric.*

that are unfaithful to the data while scoring high quality metrics, our findings have some limitations. Firstly, our learning approach has issues handling datasets with only tens of dimensions. We believe this is due to our fixed architecture and may be alleviated by adapting hidden layer sizes when dealing with low-dimensional datasets. Secondly, there is a key mismatch between the assumptions of the deep learning method we use and the semantics of quality metrics. PQMs — even local ones — need access to the *entire* dataset $\mathbf{X}$ and its projection $\mathbf{Y}$ for their computation, since the (point-wise) quality value depends not only on the point itself but on (many) other data points. Yet, in our neural networks, each data point $\mathbf{x}_i$ together with its projection $\mathbf{y}_i$ flows through the network *independently* of all other data points. This trait of Multi-Layer Perceptrons makes them unable to truly approximate a quality metric. While our postprocessing partially fixes this, the problem remains.

We show experimentally that our suggested set of 'guardrail' metrics provides more 'coverage' of assessing projection quality. Yet, this conclusion is *not final*: There is a need to (1) develop further metrics that strongly correlate with the human understanding of projections by, for example, combining VQMs based on [HUB*24, AS16, JQL*24] (among others) with data-driven metrics like Silhouette coefficient [JKA*24, BKBS24]; (2) see how well such metrics can resist our fooling (or similar attempts at that); and (3) develop new DR techniques that score high on such metrics, extending the approach of Wang et al. [WFC*18] beyond linear projections.

6. Conclusions and Future Work

We presented a method to automatically create projections of high-dimensional data that exhibit data-unrelated visual patterns — both random and structured ones — which score well on 17 standard quality metrics (PQMs). This provides evidence of the limitations of PQMs in assessing projection quality. We also show that 9 of the studied metrics get fooled even when they are not explicitly targeted by our fooling pipeline. Hence, we claim that the remaining 8 metrics, which are not implicitly fooled, are a good subset to consider in projection quality evaluation.

Our work shows the need of better ways to adequately measure whether/which data patterns a projection preserves. We see several avenues for future work. Our pipeline relies on the model Q_θ which cannot truly approximate a PQM due to its architecture (Sec. 5). Using architectures where each $(\mathbf{x}_i, \mathbf{y}_i)$ tuple can integrate information

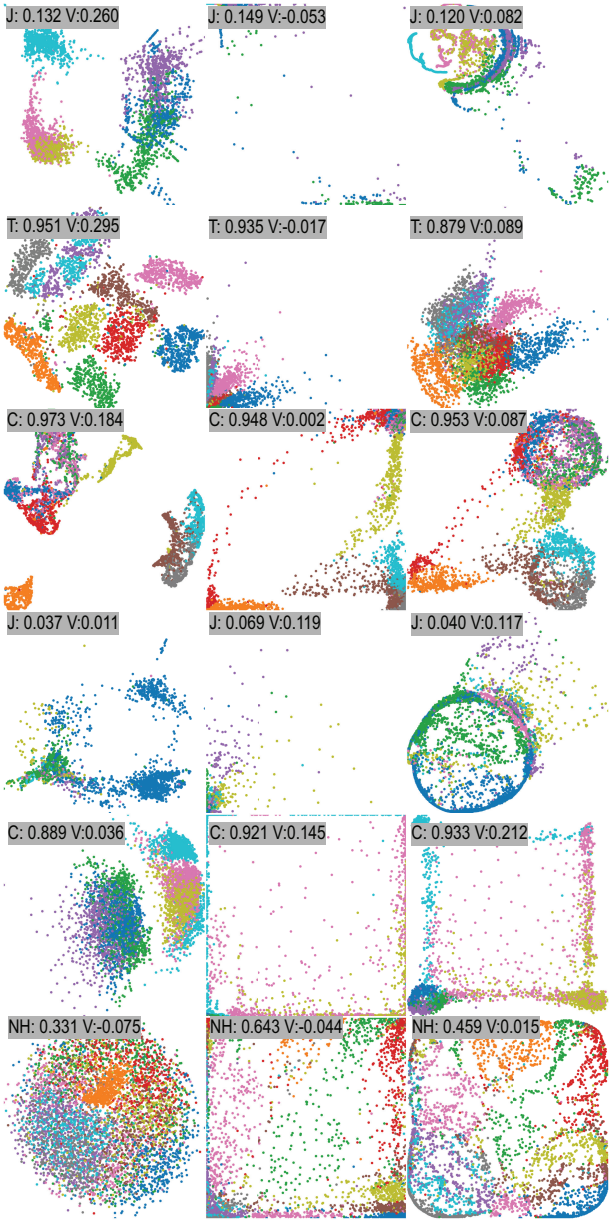

Figure 14: *Behavior of PQM and VQM (CIVM [LLX*10]) for reference projections (left column), fooled projections (middle) and postprocessed outputs (right). Annotations show the PQM used to create the fooled projection and the value V of the CIVM. The top three rows show relatively stable PQM values, with clear losses in CIVM. In the bottom 3 rows, PQMs are stable or even increase, and V increases after the destructive and constructive phases, so the arbitrary patterns are not detected by either PQM or CIVM.*

on the projection of its neighbors could make Q_θ better approximate a given $\mathcal{M}$. Transformers [VSP*17] or Nearest-Neighbor Networks [PR18] are potential solutions for this problem. Separately, the 'strict' faithfulness of the projection to the data is not always the main goal. Examples include very high-dimensional datasets where volumes of clusters become small — due to the curse of dimensionality — and points tend to lie closer to boundaries; showing such data in a 2D projection would yield highly-clumped data in very thin clusters, something hardly informative. How to measure what a 'good' projection in such cases is open for exploration.

References

[ABC*16] ABADI M., BARHAM P., CHEN J., CHEN Z., DAVIS A., ET AL.: TensorFlow: A system for large-scale machine learning. In *USENIX Symp. on Operating Systems Design and Implementation* (2016), pp. 265–283. 5

[AEM11] ALBUQUERQUE G., EISEMANN M., MAGNOR M.: Perception-based visual quality measures. In *Proc. IEEE VAST* (2011), pp. 13–20. 2, 3

[AGO*12] ANGUITA D., GHIO A., ONETO L., PARRA X., REYES-ORTIZ J. L.: Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Proc. IWAAL* (2012), vol. 7657 of *Lecture Notes in Computer Science*, Springer, pp. 216–223. 6

[Ans73] ANSCOMBE F. J.: Graphs in statistical analysis. *The American Statistician 27* (1973), 17–21. 1, 3

[AS16] AUPETIT M., SEDLMAIR M.: SepMe: 2002 new visual separation measures. In *Proc. IEEE PacificVis* (2016). 10

[Aup07] AUPETIT M.: Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing 10*, 7-9 (2007), 1304–1330. 2

[AY79] ALLEN M., YEN W.: *Introduction to measurement theory*. Waveland Press, 1979. 3

[BKBS24] BLASILLI G., KERRIGAN D., BERTINI E., SANTUCCI G.: Towards a visual perception-based analysis of clustering quality metrics. In *Proc. VDM* (2024). 10

[BS06] BERTINI E., SANTUCCI G.: Visual quality metrics. In *Proc. AVI workshop on BEyond time and errors: novel evaluation methods for information visualization* (2006). 2

[BTT22] BREDIUS C., TIAN Z., TELEA A.: Visual exploration of neural network projection stability. In *Proc. Machine Learning Methods in Visualisation for Big Data* (2022), Eurographics. 3

[CHAS18] CUTURA R., HOLZER S., AUPETIT M., SEDLMAIR M.: Viscoder: A tool for visually comparing dimensionality reduction algorithms. In *Proc. ESANN* (2018). 2

[CMS13] CAMPELLO R. J. G. B., MOULAVI D., SANDER J.: Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining* (2013), Springer, pp. 160–172. 8

[CP23] CHARI T., PACHTER L.: The specious art of single-cell genomics. *PLoS Comput Biol 19*, 8 (2023). 1

[CPA*20] COLANGE B., PELTONEN J., AUPETIT M., DUTYKH D., LESPINATS S.: Steering distortions to preserve classes and neighbors in supervised dimensionality reduction. In *Proc. NIPS* (2020), vol. 33. 5

[DW14] DANG T. N., WILKINSON L.: ScagExplorer: Exploring scatterplots by their scagnostics. In *Proc. IEEE PacificVis* (2014), pp. 73–80. 3

[EAS*23] ESPADOTO M., APPLEBY G., SUH A., CASHMAN D., LI M., SCHEIDEGGER C., ANDERSON E., CHANG R., TELEA A.: UnProjection: Leveraging inverse-projections for visual analytics of high-dimensional data. *IEEE TVCG 29*, 2 (2023), 1559–1572. 3

[EHT19] ESPADOTO M., HIRATA N., TELEA A.: Deep learning multidimensional projections. *Information Visualization 19* (2019), 247 – 269. 3

[EMdSP*15] ETEMADPOUR R., MOTTA R., DE SOUZA PAIVA J. G., MINGHIM R., DE OLIVEIRA M. C. F., LINSEN L.: Perception-based evaluation of projection methods for multidimensional data visualization. *IEEE TVCG 21*, 1 (2015), 81–94. 3

[EMK*21] ESPADOTO M., MARTINS R. M., KERREN A., HIRATA N. S. T., TELEA A. C.: Toward a quantitative survey of dimension reduction techniques. *IEEE TVCG 27*, 3 (2021), 2153–2173. 1, 2, 3

[FR91] FRUCHTERMAN T. M. J., REINGOLD E.: Graph drawing by force-directed placement. *Software: Practice and Experience 21* (1991). 4

[GR09] GOLDBERG Y., RITOV Y.: Local procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Mach. Learn. 77*, 1 (2009), 1–25. 2, 5

[GZZ05] GENG X., ZHAN D.-C., ZHOU Z.-H.: Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics) 35* (2005), 1098–1107. 5

[HRFS99] HOPKINS M., REEBER E., FORMAN G., SUERMONDT J.: Spambase. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C53G6X. 6

[Hub85] HUBER P. J.: Projection pursuit. *Ann. Statist. 13*, 2 (1985), 435–475. 2

[HUB*24] HAMZA M., 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 23*, 2 (2024). 10

[Hul94] HULL J.: A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell. 16* (1994), 550–554. 6

[Jac01] JACCARD P.: Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bulletin de la Societe Vaudoise des Sciences Naturelles 37* (01 1901), 241–72. 5

[JCC*11] JOIA P., COIMBRA D., CUMINATO J. A., PAULOVICH F. V., NONATO L. G.: Local affine multidimensional projection. *IEEE TVCG 17*, 12 (2011), 2563–2571. 2, 5

[JCJ*23] JEON H., CHO A., JANG J., LEE S., HYUN J., KO H.-K., JO J., SEO J.: ZADU: a Python library for evaluating the reliability of dimensionality reduction embeddings. *IEEE Visualization and Visual Analytics* (2023), 196–200. 1, 2, 9

[JJ09] JOHANSSON S., JOHANSSON J.: Interactive dimensionality reduction through user-defined combinations of quality metrics. *IEEE TVCG 15* (2009). 2

[JKA*24] JEON H., KUO Y.-H., AUPETIT M., MA K.-L., SEO J.: Classes are not clusters: Improving label-based evaluation of dimensionality reduction. *IEEE TVCG 30*, 1 (2024), 781–791. 10

[JQL*24] JEON H., QUADRI G. J., LEE H., ROSEN P., SZAFIR D. A., SEO J.: CLAMS: a cluster ambiguity measure for estimating perceptual variability in visual clustering. *IEEE TVCG 30*, 1 (2024), 770–780. 10

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *Proc. ICLR* (2015). 6

[Kru64] KRUSKAL J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29*, 1 (1964), 1–27. 1, 5

[LA11] LESPINATS S., AUPETIT M.: CheckViz: Sanity check and topological clues for linear and non-linear mappings. *Computer Graphics Forum 30*, 1 (2011), 113–125. 2

[LBBH98] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P.: Gradient-based learning applied to document recognition. *Proc. IEEE 86*, 11 (1998), 2278–2324. 6

[LBK24] LAUSE J., BERENS P., KOBAK D.: The art of seeing the elephant in the room: 2D embeddings of single-cell data do make sense. *PLoS Comput Biol 20*, 10 (2024). 1

[LLX*10] LIU Y., LI Z., XIONG H., GAO X., WU J.: Understanding of internal clustering validation measures. In *ICDM* (2010), IEEE Computer Society, pp. 911–916. 6, 9, 10

[LV09] LEE J., VERLEYSEN M.: Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing 72* (2009), 1431–1443. 5

[MBC*23] MORARIU C., BIBAL A., CUTURA R., FRÉNAY B., SEDLMAIR M.: Predicting user preferences of dimensionality reduction embedding quality. *IEEE TVCG 29*, 1 (2023), 745–755. 2

[MBT24a] MACHADO A., BEHRISCH M., TELEA A.: Accelerated projection quality metrics, 2024. https://pypi.org/p/tensorflow-projection-qm/. 5

[MBT24b] MACHADO A., BEHRISCH M., TELEA A.: Online experimental results of fooling projection quality metrics, 2024. https://amreis.github.io/fool-proj-metrics/. 5

[MBT24c] MACHADO A., BEHRISCH M., TELEA A.: Projection fooler source code, 2024. https://github.com/amreis/fooling-projection-metrics. 5

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

[MF17] MATEJKA J., FITZMAURICE G.: Same stats, different graphs: Generating datasets with varied appearance and identical statistics through simulated annealing. *Proc. ACM CHI* (2017). 1, 3

[MH08] MAATEN L., HINTON G. E.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605. 1, 5

[MH18] MCINNES L., HEALY J.: UMAP: Uniform manifold approximation and projection for dimension reduction. *ArXiv abs/1802.03426* (2018). 1, 5

[MSJG15] MAKHZANI A., SHLENS J., JAITLY N., GOODFELLOW I. J.: Adversarial autoencoders. *CoRR abs/1511.05644* (2015). 3

[MTB24] MACHADO A., TELEA A., BEHRISCH M.: Controlling the scatterplot shapes of 2D and 3D multidimensional projections. *Computers & Graphics* (2024). 2, 3

[Mun14] MUNZNER T.: *Visualization Analysis and Design: Principles, Techniques, and Practice*. CRC Press, 2014. 2

[NA18] NONATO L. G., AUPETIT M.: Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG 25*, 8 (2018), 2650–2673. 1

[OEJT23] OLIVEIRA A., ESPADOTO M., JR R. H., TELEA A.: Stability analysis of supervised decision boundary maps. *SN Computer Science 4*, 226 (2023). 3

[PKF*16] PANDEY A. V., KRAUSE J., FELIX C., BOY J., BERTINI E.: Towards understanding human similarity perception in the analysis of large sets of scatter plots. In *Proc. ACM CHI* (2016), pp. 3659–3669. 3

[PNML08] PAULOVICH F. V., NONATO L. G., MINGHIM R., LEVKOWITZ H.: Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE TVCG 14*, 3 (2008), 564–575. 2, 5

[PR18] PLÖTZ T., ROTH S.: Neural nearest neighbors networks. In *Proc. NeurIPS* (2018), pp. 1095–1106. 10

[PVG*11] PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., ET AL.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830. 5

[RB10] RENSINK R. A., BALDRIDGE G.: The perception of correlation in scatterplots. *Computer Graphics Forum 29*, 3 (2010), 1203–1210. 3

[SA15] SEDLMAIR M., AUPETIT M.: Data-driven evaluation of visual quality measures. *Computer Graphics Forum 34*, 3 (2015), 201–210. 2, 3

[SC88] SIEGEL S., CASTELLAN N. J.: *Nonparametric statistics for the behavioral sciences*. McGraw Hill, 1988. 2, 5

[SMK24] SMELSER K., MILLER J., KOBOUROV S.: "Normalized Stress" is Not Normalized: How to Interpret Stress Correctly . In *Proc. IEEE BELIV* (2024), IEEE Computer Society, pp. 41–50. 5, 9

[SNLH09] SIPS M., NEUBERT B., LEWIS J. P., HANRAHAN P.: Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum 28*, 3 (2009), 831–838. 2, 3, 5

[SSB*16] SHAO L., SCHLEICHER T., BEHRISCH M., SCHRECK T., SIPIRAN I., KEIM D.: Guiding the exploration of scatter plot data using motif-based interest measures. *J. Vis. Lang. Comput. 36* (2016), 1–12. 2

[STMT12] SEDLMAIR M., TATU A., MUNZNER T., TORY M.: A taxonomy of visual cluster separation factors. *Computer Graphics Forum 31*, 3 (2012), 1335–1344. 1, 2

[SvLB10] SCHRECK T., VON LANDESBERGER T., BREMM S.: Techniques for precision-based visual analysis of projected data. *Inf. Vis. 9*, 3 (2010), 181–193. 2

[TBB*10] TATU A., BAK P., BERTINI E., KEIM D., SCHNEIDEWIND J.: Visual quality metrics and human perception: an initial study on 2D projections of large multidimensional data. In *Proc. AVI* (2010), ACM, pp. 49–56. 3

[Tel14] TELEA A. C.: *Data Visualization: Principles and Practice, Second Edition*, 2nd ed. A. K. Peters, Ltd., USA, 2014. 2

[Tho17] THOMA M.: The Reuters dataset, July 2017. 6

[TSL00] TENENBAUM J., SILVA V., LANGFORD J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science 290 5500* (2000), 2319–23. 1, 5

[TT88] TUKEY J. W., TUKEY P. A.: *Computer graphics and exploratory data analysis: An introduction*, vol. 5. CRC Press, 1988. 2

[VBF21] VU V. M., BIBAL A., FRÉNAY B.: Constraint preserving score for automatic hyperparameter tuning of dimensionality reduction methods for visualization. *IEEE Trans. on Artificial Intelligence 2*, 3 (2021), 269–282. 2

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

[VK06] VENNA J., KASKI S.: Local multidimensional scaling. *Neural Networks 19*, 6 (2006), 889–899. Advances in Self Organising Maps - WSOM'05. 2, 5

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

[VSP*17] VASWANI A., SHAZEER N. M., PARMAR N., USZKOREIT J., JONES L., ET AL.: Attention is all you need. In *Proc. NeurIPS* (2017), pp. 5998–6008. 10

[WAG05] WILKINSON L., ANAND A., GROSSMAN R. L.: Graph-theoretic scagnostics. In *Proc. IEEE InfoVis* (2005), pp. 157–164. 2

[WAG06] WILKINSON L., ANAND A., GROSSMAN R.: High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. *IEEE TVCG 12*, 6 (2006), 1363–1372. 2

[WFC*18] WANG Y., FENG K., CHU X., ZHANG J., FU C.-W., SEDLMAIR M.: A perception-driven approach to supervised dimensionality reduction for visualization. *IEEE TVCG 24*, 5 (2018), 1828–1840. 10

[WM17] WANG B., MUELLER K.: The subspace voyager: Exploring high-dimensional data along a continuum of salient 3D subspaces. *IEEE TVCG 24*, 2 (2017), 1204–1222. 2

[WVJ16] WATTENBERG M., VIÉGAS F., JOHNSON I.: How to use t-sne effectively. *Distill 1*, 10 (2016), e2. 2

[WWL*20] WANG Y., WANG Z., LIU T., CORRELL M., CHENG Z., DEUSSEN O., SEDLMAIR M.: Improving the robustness of scagnostics. *IEEE Trans. Vis. Comput. Graph. 26*, 1 (2020), 759–769. 3

[XRV17] XIAO H., RASUL K., VOLLGRAF R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR abs/1708.07747* (2017). 6

[YWS*14] YATES A., WEBB A., SHARPNACK M. F., CHAMBERLIN H., HUANG K., MACHIRAJU R.: Visualizing multidimensional data with glyph SPLOMs. *Computer Graphics Forum 33*, 3 (2014), 301–310. 2