Machine Learning Methods in Visualisation for Big Data (2022)
D. Archambault, J. Peltonen, and I. Nabney (Editors)

Visual Exploration of Neural Network Projection Stability

C. Bredius[1] and Z. Tian[1] and A. Telea[1]

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

Abstract

We present a method to visually assess the stability of deep learned projections. For this, we perturb the high-dimensional data by controlled sequences and visualize the resulting changes in the 2D projection. We apply our method to a recent deep learned projection framework on several training configurations (learned projections and real-world datasets). Our method, which is simple to implement, runs at interactive rates, sheds several novel insights on the stability of the explored method.

CCS Concepts

• ***Human-centered computing*** → ***Information visualization; Visual analytics; Visualization systems and tools;***

1. Introduction

Multidimensional projections (MPs) are established methods for visualizing high-dimensional data. Within such methods, deep learning approaches, pioneered by [HS06], have several attractive features: They are simple to implement, work generically for any type of high-dimensional data, are fast (especially when using GPU implementations), parametric, and have out-of-sample abilities, *i.e.*, can project unseen data along training data. Several such methods have been proposed [BLS17, vdM09]. Recently, Neural Network Projections (NNP) [EHT20], and its variants [EHT21, KET*22, MEF*20], proposed a deep learning approach which learns the style of any user-supplied projection technique such as t-SNE, UMAP, or PCA, to name a few, with high quality and little training effort.

Yet, NNP and its variants have not been studied for their *stability*. Simply put, their users do not have a clear idea how, and how much, the resulting projection would change when its (unseen) input data changes. This is important to know in practice to assess how a trained NNP would fare on, or *generalize* to, unseen data. How different can this data be from the training data to still obtain plausible projection results? Which types (and amounts) of changes affect the resulting projection the most? And what is the maximum allowed change so that the visual interpretation of a projection would not be affected?

Stability analysis of MP methods was addressed earlier, most notably by Garcia *et al.* [GFVLD13]. However, they studied changes in terms of method hyperparameters and adding more data points, not *changing* the points themselves. Also, they did not study deep learning MP methods. Surveys of MP methods [Yin07, BBH12, KH13, SVPM14, CG15, NA18, EMK*19] mention stability as an important trait but do not provide quantitative evaluations thereof and also do not cover deep learning methods.

We present a method to answer the above questions which is simple to implement, fast to compute, and can be used for any deterministic or parametric MP technique, even beyond the NNP class. We use our method to analyze NNP's stability under a family of changes of different amplitudes. For each change type, we empirically find the maximum change beyond which the projection's visual interpretation for the task of finding the different clusters present in the dataset is affected. We illustrate the above for NNP trained on different projections and real-world datasets.

2. Method

Let $D = \{\mathbf{x}\} \subset \mathbb{R}^n$ be a high dimensional dataset and $P : \mathbb{R}^n \to \mathbb{R}^2$ the (NNP) projection method that maps each $\mathbf{x} \in D$ to a point $P(\mathbf{x}) \in \mathbb{R}^2$. Let $\Delta\mathbf{x}$ be a small change applied to point $\mathbf{x}$, $\Delta D = \{\Delta\mathbf{x} | \mathbf{x} \in D\}$ the set of these changes, and $D' = \{\mathbf{x} + \Delta\mathbf{x} | \mathbf{x} \in D\}$ the set D where each point has been changed accordingly. Let $P(D) = \{P(\mathbf{x}) | \mathbf{x} \in D\}$ be the 2D scatterplot obtaining by projecting D, $P(D')$ the scatterplot from projecting D', and ΔP the *visual* difference between $P(D)$ and $P(D')$. In the following, D is the training set for NNP, but any other (unseen) dataset can be equally used.

Since NNP is a deep learning method, when $\|\Delta D\| \to 0$ (data changes $\Delta\mathbf{x}$ are very small for all points in D), $\|\Delta P\| \to 0$ (the projection will not visibly change). Here, $\|\cdot\|$ denotes the L2 metric. The more ΔD increases, the more, arguably, will the projection change ΔP increase. At some point, $P(D')$ will be so different from $P(D)$ that users looking at it will not see the same *data structure* that P originally depicted. The smaller is the ratio $\Delta P / \Delta D$ (related to the derivative of the projection function P), the more *stable* is the projection, and conversely. This way of measuring stability has been used recently for other high-dimensional datasets such as dynamic projections [VGdS*20] and dynamic treemaps [VSC*20].

Knowing the point $\|\Delta D\|$where P' is visually assessed as depicting a different data structure than P is useful. Take one key use-case for projections – their ability to depict clusters of similar samples in the data [NA18, TAE*09, TBB*10, SA15]. Knowing how much the data D can change until we see different clusters tells us how 'far away' from the training set D we can expect that NNP generalizes. This next tells us when we can reuse a previously trained NNP or if we must train NNP again since the new data D' is too different from D.

Based on the above, we propose the following workflow for visually exploring the stability of NNP:

- Define a family of perturbations F. A perturbation $f \in F$ is a way to change the data, $D' = f(D, \sigma)$, where σ is an intuitive parameter used to control the amount of change ΔD.
- For each perturbation type $f \in F$
  - generate K increasingly large changes $\{\Delta D_i\}$ so that $\|\Delta D_i\| < \|\Delta D_j\|, \forall i < j, 1 \leq i \leq K, 1 \leq j \leq K$, of NNP's training set D.
  - compute the corresponding projections $\{P(D_i)\}$ for $\{\Delta D_i\}$.

– visually find the largest $i \in [1..K]$ for which $P(D'_i)$ still shows the same clusters as $P(D)$. We call the change amount σ_i the *frontier* of the projection P. Changes of NNP's input beyond this frontier yield projections which users interpret differently in terms of visual clusters than the original projection $P(D)$.

3. Experiments to measure stability

We apply our method to study NNP's stability for different learned projections, real-world datasets, and perturbation types, as follows.

3.1. Perturbation types

Many types of perturbations f are possible. We next propose five such perturbations with different analysis goals. Note that the actual ranges of the data dimensions in D are normalized to the $[0,1]$ interval, following NNP's implementation [EHT20].

Translation: Moves a point $\mathbf{x}$ with a value $\sigma_T \in [0,1]$ along a fraction $\sigma_D \in [0,1]$ of randomly chosen dimensions from the total n ones. Simulates additive bias to a subset of data dimensions.

Scaling: Like translation, but scales the chosen dimensions by a factor $\sigma_S \in [0,1]$. Simulates multiplicative bias.

Jittering: Adds random noise uniformly spread in $[-1,1]$ to a fraction $\sigma_J \in [0,1]$ of randomly chosen dimensions from the n ones.

Dimension permutation: Swaps K randomly-picked dimension-pairs from the n ones, where $K = \sigma_P/n$ for a given $\sigma_P \in [0,1]$. Ideally, NNP (or any projection) should be fully robust to such changes since the order of dimensions is irrelevant.

Dimension removal: Removes a fraction $\sigma_R \in [0,1]$ randomly chosen dimensions from the total n ones, to test how robust is NNP to potential missing dimensions.

After perturbing the data D, we linearly renormalize D' in the $\mathbb{R}^n$ unit hypercube to make it fit the constraints of NNP listed above.

3.2. Datasets and projection techniques

We trained NNP to mimic three popular projection techniques – PCA [Jol02], t-SNE [vdMH08], and UMAP [MHM18]. Its hyperparameters, taken from [EFHT20], are no regularization, ADAM optimization, noise-after-data augmentation, mean absolute error loss, and a three fully-connected hidden layer architecture with 600, 240, and 600 units respectively. We verified that the training converged with small losses in the range of earlier reported ones [EFHT20].

We tested NNP's stability for the perturbations in Sec. 3.1, one perturbation at a time, on three datasets: MNIST [LCB10] (60K 28×28 grayscale images of ten handwritten digits), FashionMNIST (60K images of ten clothing item classes), and Reuters [Tho17] (10K text documents with a vocabulary size of over 35K terms; we used the 10 most frequent classes to make this dataset similar to the first two). Training set sizes for the three datasets were 10K, 10K, and 4.5K samples respectively. Next, we sampled the perturbation parameter space $(\sigma_T, \sigma_D, \sigma_S, \sigma_J, \sigma_R, \sigma_P)$ over the $[0,1]$ range with step size 0.1 and plotted the results, color coded by class to ease interpretation. We repeated our tests several times to account for the random nature of the perturbations and visually confirmed result consistency. We next present a subset (for space reasons) of these results. All images and our method's Python source code are publicly available [Bre20].

3.3. Translation stability

Figure 1(top) shows how NNP behaves in presence of additive bias. For each (dataset, projection) pair, we show nine combinations of $\sigma_T \in \{0.1, 0.3, 0.7\}$ and $\sigma_D \in \{0.1, 0.5, 0.9\}$. For MNIST, we see a clear trend: The projection 'collapses' towards it center, with clusters increasingly mixing, as both the translation factor σ_T and the number of dimensions σ_D it is applied to grow. This is explained by the fact that translation affects only a *subset* of *randomly* chosen dimensions. Interestingly, the visual degradation for the same σ_T, σ_D varies per learned projection: For t-SNE, this is the largest, clusters get already mixed for $\sigma_T = 0.5, \sigma_D = 0.5$. For PCA, the degradation is the least, the effect being similar to a uniform scaling of the projection. Visual clusters do not become significantly more mixed up when σ_T or σ_D increase, since they are already poorly separated in the original PCA projection. UMAP fares visually between t-SNE and PCA. We believe this is due to the already very strong visual cluster separation in the original UMAP projection, a known characteristic of this method [MHM18, EMK*19]. FashionMNIST has very similar results, likely since it is a similar-nature (grayscale image) dataset – see supplementary material [Bre20].

For Reuters, the situation is starkly different: All methods loose visual cluster separation when σ_T or σ_D are above 0.1 (see also online material for more images). While we do not know why this occurs, this is a very important insight as it tells that NNP's stability *vs* additive noise is strongly influenced by the *nature* of the projected dataset or the training set size (Reuters: 4.5K samples; MNIST and FashionMNIST: 10K samples). As such, users should not assume that if NNP behaves robustly on one type of data trained for the same will hold for other types.

3.4. Scaling stability

Figure 1(bottom) analyzes scaling stability. We see that NNP behaves very stably irrespective of the dataset or learned projection. We also see that the visual cluster separation decreases as the scaling factor approaches zero. This makes sense as the variance captured by the downscaled dimensions is essentially reduced.

3.5. Jittering stability

Figure 2(top) shows that, for the MNIST dataset and all learned projections, NNP is remarkably robust to adding noise to up to 10 to 20% ($\sigma_J \in [0.1, 0.2]$) of the n dimensions, given that the noise bandwidth is the *full* range $[0,1]$ of the dimensions. As noise is added to more dimensions, visual clusters get mixed up, quite similar to the additive noise effect (Fig. 1). For the Reuters dataset, the situation is very different: Even the smallest amount of jitter completely 'collapses' the projection, regardless of the learned technique, even more so than for additive noise. This strengthens our insight that NNP's stability is strongly dependent on the nature of the projected data and/or training set size and far less on the learned projection technique.

3.6. Dimension permutation stability

As Sec. 3.1 noted, we expect NNP to be fully stable *vs* dimension permutation. Figure 2(middle) confirms this for all projections up to roughly 30-40% permuted dimension-pairs (σ_P up to 0.3..0.4). For larger σ_P, projections slowly collapse and the visual clusters mix up. This is an unexpected insight. The cause may be NNP's *fully* connected and *bottleneck* architecture which 'specializes' at training certain parts of some layers to process specific data dimensions. As the n dimensions of a point go one-to-one in the n units of the input layer, permuting dimensions can create confusion, akin to the known sensitivity of deep learning image classifiers to rotated images [SK19]. If so, we could consider data augmentation for NNP's training to remove this unwanted dimension-order sensitivity.

3.7. Dimension removal stability

Figure 2(bottom) explores NNP's stability when removing an increasingly large number of data dimensions. Surprisingly, the projection

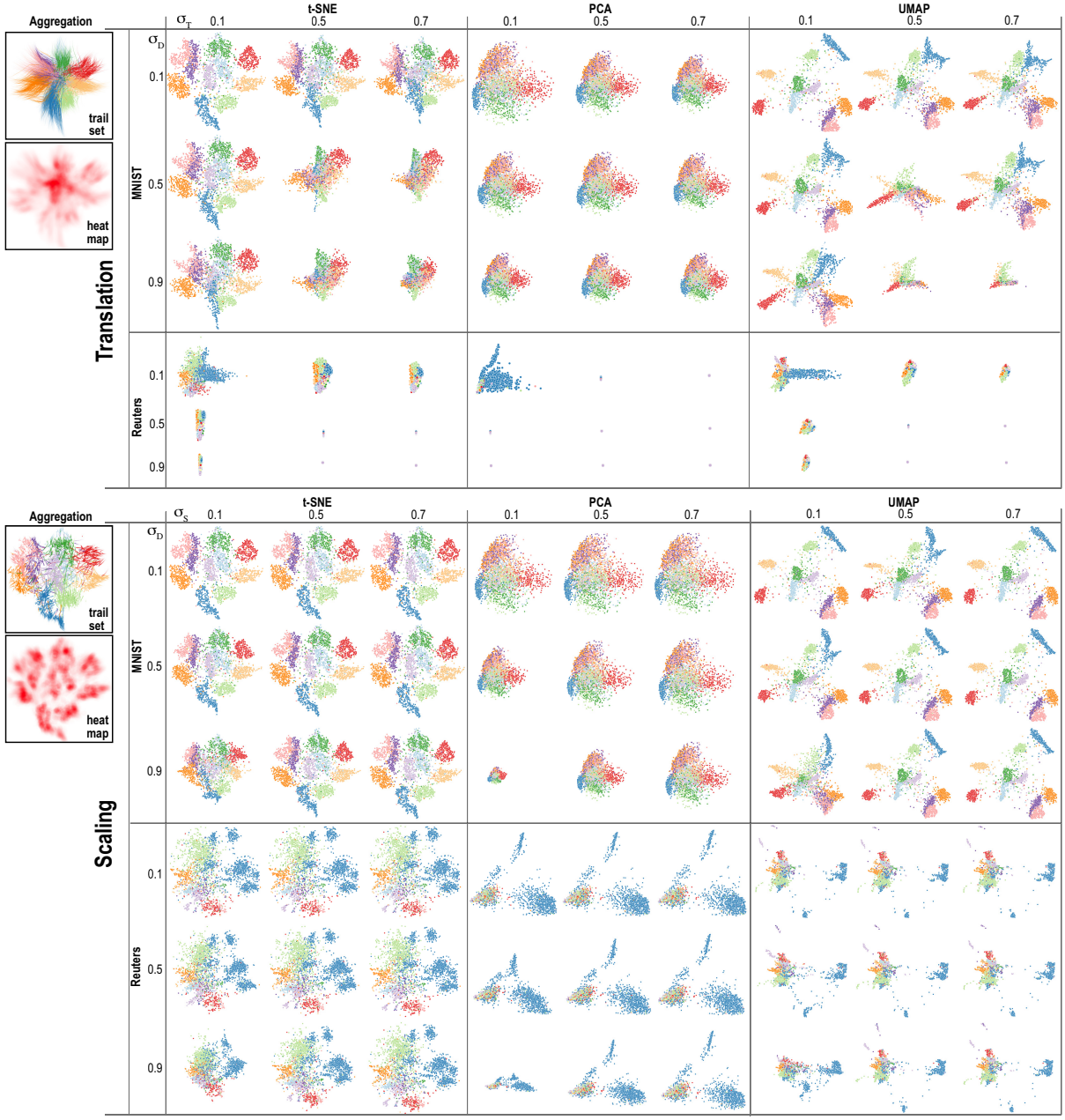

Figure 1: *Translation (top) and scaling (bottom) stability analysis, MNIST and Reuters datasets, NNP trained for t-SNE, PCA, and UMAP.*

changes quite little up to roughly 40% ($\sigma_R = 0.4$) removed dimensions, and this holds for all learned projections and datasets tested. Up to some extent, this can be explained by redundancies being present in the input data or, in other words, the tested datasets having a (far) smaller intrinsic dimensionality than their n dimensions. Still, since NNP does not explicitly aim to detect or use the data's intrinsic dimensionality, and since the removed dimensions are *randomly* picked, this is a quite interesting, and positive, result. We plan to further investigate this effect and its potential connection to projecting random subspaces of a high-dimensional space.

We compactly visualize translation, scaling, and dimension removal by 2D *trails* that connect the projections $P(\mathbf{x}+\Delta\mathbf{x})$ of each sample $\mathbf{x} \in D$, see left insets in Figs. 1 and 2, computed for MNIST projected by NNP trained with t-SNE and $\sigma_{D,S,P,R} \in [0, 0.5]$, $K = 50$ sample steps. We see here the *nonlinear* nature of NNP: Translations and scalings (linear) map to curved trails. We also see that translation and dimension permutation keep the visual clusters better separated (less color mixing) than scaling and dimension removal and also the strong 'projection collapsing' effect of translation. To better see the *locality* of the perturbations, we also aggregate them using kernel density plots [Sil86] of the projected perturbed points, visualized by a white-to-red heatmap (left insets, Figs. 1 and 2). The far stronger 'collapse' effect of translation *vs* the more local behavior of the other perturbations is visible in the central red hotspot of the translation heatmap plot.

Given NNP's high inference speed [EHT20], all above visualizations are created in *real time* for datasets of hundreds of thousands of samples and hundreds of dimensions on a commodity PC while interactively modifying the perturbation parameters. This allows users to freely explore NNP's stability in any desired direction.

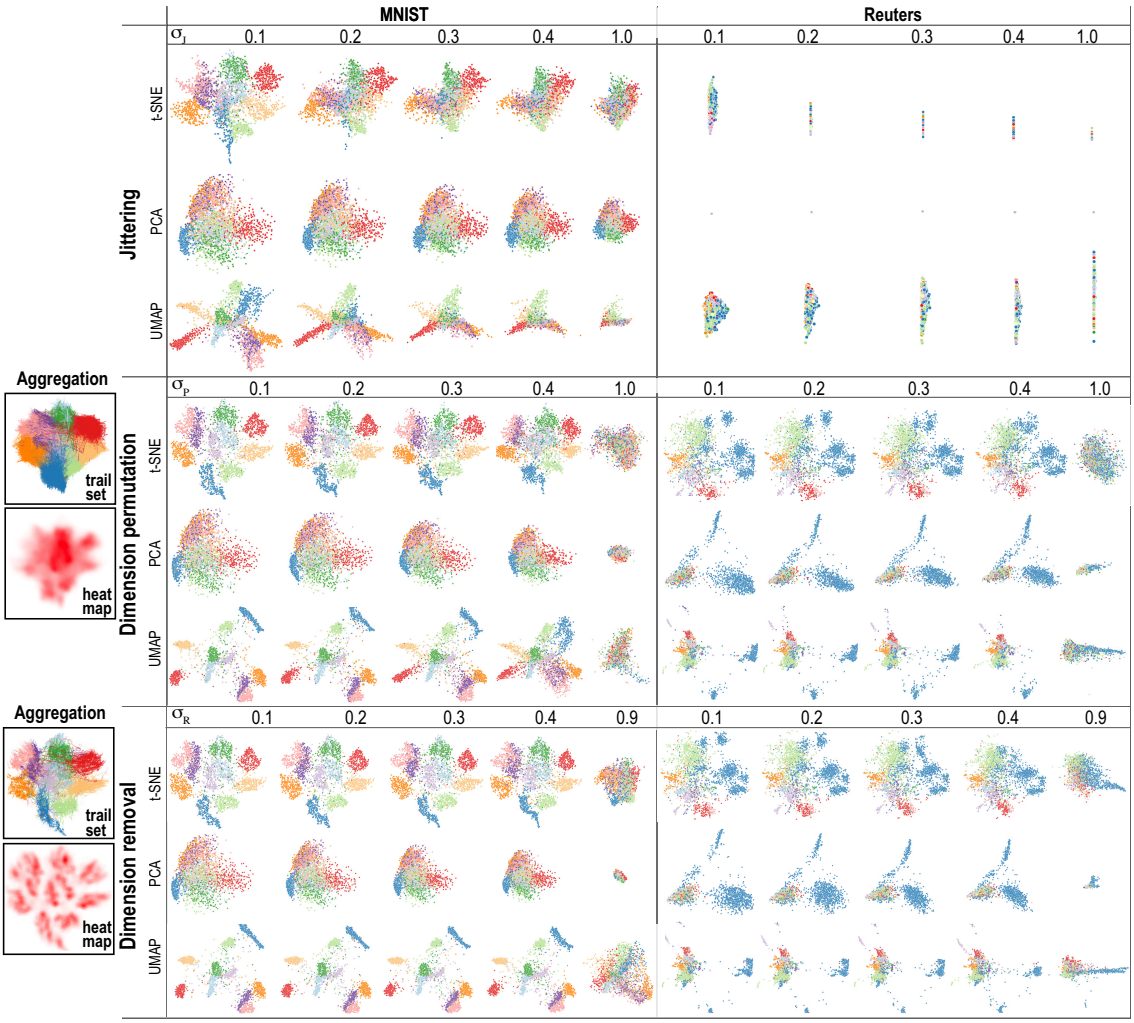

Figure 2: *Jittering (top), dimension permutation (middle), and dimension removal (bottom) stability analysis, MNIST and Reuters datasets, NNP trained for t-SNE, PCA, and UMAP.*

4. Conclusion

Our results show that NNP is in most cases stable even to significant amounts of perturbations of various types. As perturbations increase in amplitude or number of affected dimensions, the NNP projection 'collapses' inwards smoothly in terms of the shapes and relative positions of the visual clusters. This can be explained by NNP's deterministic nature – though not *all* deterministic projection techniques show an equally graceful degradation. Yet, this does not happen in all cases – for the Reuters dataset, the collapse occurs even for small perturbations. Finding out which characteristics (*a.k.a.* traits [NA18, EMK*19]) in the nature of a given dataset causes this particular behavior can lead to refining NNP to exhibit higher stability, a direction we want to explore next.

Other further directions of work are possible, as follows.

Evaluation: More datasets and learned projections can be used to test NNP's stability. Quantitative measurement of the projection stability is another low-hanging fruit to explore. We also aim to understand the *reasons* for instability and adapt NNP (or its training) to correct for this. As any (deep learned) regressor, NNP inherently changes its output upon its input change. Yet, not all input changes are equal. We foresee ways to refine NNP's training to become less sensitive to certain types of input changes which are deemed less important. For this, we plan to design *custom* perturbations to model the actual variability, or noise, present in a given application domain, and improve NNP to handle these specifically per-domain, thereby extending earlier work [EFHT20] which aimed at making NNP more robust by effectively exploring how *specific* data changes induce projection changes.

Comparison: Comparing NNP's stability with that of the *actual* projection techniques that NNP learns, *e.g.*, t-SNE or UMAP, and adding such results to public benchmarks [EMK*19] will help users choose suitable projection techniques from a stability viewpoint.

Explanation: Annotating projections with explanations of their visual patterns is a powerful tool for understanding high-dimensional data [TZvD*21, dSRM*15]. Using explanations with our perturbation analysis can help finding which dimensions are behind the appearance of (un)stable parts of a projection, thus show why points come together or get separated. This can help us next to understand how stability depends on the dataset type. Secondly, studying how explanations themselves are (un)stable with respect to data perturbations is useful for getting more trust in the explanations themselves. All in all, this will help designing visual explanations more robust to noise, gauging the actual effect on the user's perception of the data implied by such projection instabilities, and also for using explanations to actually measure the stability of a projection technique.

References

[BBH12] BUNTE K., BIEHL M., HAMMER B.: A general framework for dimensionality reducing data visualization mapping. *Neural Computation 24*, 3 (2012), 771–804. 1

[BLS17] BECKER M., LIPPEL J., STUHLSATZ A.: Regularized non- linear discriminant analysis – an approach to robust dimensionality reduction for data visualization. In *Proc. VISIGRAPP* (2017), SciTePress, pp. 116–127. 1

[Bre20] BREDIUS C.: Nnp stability exploration code and data, 2020. URL: https://github.com/CarloBredius/NNP. 2

[CG15] CUNNINGHAM J., GHAHRAMANI Z.: Linear dimensionality reduction: Survey, insights, and generalizations. *JMLR 16* (2015), 2859–2900. 1

[dSRM*15] DA SILVA R., RAUBER P., MARTINS R., MINGHIM R., TELEA A.: Attribute-based visual explanation of multidimensional projections. In *Proc. EuroVA* (2015), pp. 69–75. 4

[EFHT20] ESPADOTO M., FALCAO A., HIRATA N., TELEA A.: Improving neural network-based multidimensional projections. In *Proc. IVAPP* (2020), SciTePress. 2, 4

[EHT20] ESPADOTO M., HIRATA N., TELEA A.: Deep learning multidimensional projections. *Information Visualization 9*, 3 (2020), 247–269. 1, 2, 3

[EHT21] ESPADOTO M., HIRATA N., TELEA A.: Self-supervised dimensionality reduction with neural networks and pseudo-labeling. In *Proc. IVAPP* (2021), SciTePress. 1

[EMK*19] ESPADOTO M., MARTINS R., KERREN A., HIRATA N., TELEA A.: Toward a quantitative survey of dimension reduction techniques. *IEEE TVCG 27*, 3 (2019), 2153–2173. 1, 2, 4

[GFVLD13] GARCIA-FERNANDEZ F. J., VERLEYSEN M., LEE J. A., DIAZ I.: Stability comparison of dimensionality reduction techniques attending to data and parameter variations. In *Proc. EuroVis (short papers)* (2013). 1

[HS06] HINTON G. E., SALAKHUTDINOV R. R.: Reducing the dimensionality of data with neural networks. *Science 313*, 5786 (2006), 504–507. Publisher: AAAS. 1

[Jol02] JOLLIFFE I. T.: *Principal Component Analysis*. Springer, 2002. 2nd edition. 2

[KET*22] KIM Y., ESPADOTO M., TRAGER S., ROERDINK J., TELEA A.: SDR-NNP: Sharpened dimensionality reduction with neural networks. In *Proc. IVAPP* (2022), SciTePress. 1

[KH13] KEHRER J., HAUSER H.: Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE TVCG 19*, 3 (2013), 495–513. 1

[LCB10] LECUN Y., CORTES C., BURGES C.: MNIST handwritten digit database, 2010. AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist. 2

[MEF*20] MODRAKOWSKI T. S., ESPADOTO M., FALCÃO A. X., HIRATA N. S. T., TELEA A.: Improving deep learning projections by neighborhood analysis. In *Communication in Computer and Information Science* (2020), Springer. 1

[MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform manifold approximation and projection for dimension reduction, 2018. arXiv:1802.03426v2 [stat.ML]. 2

[NA18] NONATO L., AUPETIT M.: Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG 8*, 25 (2018), 2650–2673. 1, 4

[SA15] SEDLMAIR M., AUPETIT M.: Data-driven evaluation of visual quality measures. *Comp Graph Forum 34*, 3 (2015), 545–559. 1

[Sil86] SILVERMAN B. W.: *Density estimation for statistics and data analysis*. Chapman & Hall / CRC, 1986. 3

[SK19] SHORTEN C., KHOSHGOFTAAR T. M.: A survey on image data augmentation for deep learning. *Journal of Big Data 6*, 60 (2019). 2

[SVPM14] SORZANO C., VARGAS J., PASCUAL-MONTANO A.: A survey of dimensionality reduction techniques, 2014. arXiv:1403.2877 [stat.ML]. 1

[TAE*09] TATU A., ALBUQUERQUE G., EISEMANN M., SCHNEIDEWIND J., THEISEL H., MAGNOR M., , KEIM D.: Combining automated analysis and visualization techniques for effective exploration of high dimensional data. In *Proc. IEEE VAST* (2009), pp. 59–66. 1

[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. 1

[Tho17] THOMA M.: The Reuters dataset, 2017. URL: https://martin-thoma.com/nlp-reuters. 2

[TZvD*21] TIAN Z., ZHAI X., VAN DRIEL D., VAN STEENPAAL G., ESPADOTO M., TELEA A.: Using multiple attribute-based explanations of multidimensional projections to explore high-dimensional data. *Computers and Graphics 98* (2021), 93–104. 4

[vdM09] VAN DER MAATEN L.: Learning a parametric embedding by preserving local structure. In *Proc. AI-STATS* (2009). 1

[vdMH08] VAN DER MAATEN L., HINTON G. E.: Visualizing data using t-sne. *JMLR 9* (2008), 2579–2605. 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. 1

[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. 1

[Yin07] YIN H.: Nonlinear dimensionality reduction and data visualization: A review. *Intl. Journal of Automation and Computing 4*, 3 (2007), 294–303. 1