Graphical Models 108 (2020) 101060

Contents lists available at ScienceDirect

Graphical Models

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

Robust dimensionality reduction for data visualization with deep neural networks

Martin Becker*, Jens Lippel, André Stuhlsatz, Thomas Zielke

Hochschule Düsseldorf – University of Applied Sciences, Germany

ARTICLE INFO

Keywords:
High-dimensional data
Dimensionality reduction
Data visualization
Robust feature extraction
Regularization
Deep neural networks
Machine learning
GerDA
Discriminant analysis
Deep autoencoder
Hierarchical cluster analysis

ABSTRACT

We elaborate on the robustness assessment of a deep neural network (DNN) approach to dimensionality reduction for data visualization. The proposed DNN seeks to improve the class separability and compactness in a low-dimensional feature space, which is a natural strategy to obtain well-clustered visualizations. It consists of a DNN-based nonlinear generalization of Fisher's linear discriminant analysis and a DNN-based regularizer. Regarding data visualization, a well-regularized DNN guarantees to learn sufficiently similar data visualizations for different sets of samples that represent the data approximately equally well. Such a robustness against fluctuations in the data is essential for many real-world applications. Our results show that the combined DNN is considerably more robust than the generalized discriminant analysis alone. We further support this conclusion by examining feature representations from four comparative approaches. As a means of measuring the structural dissimilarity between different feature representations, we propose a hierarchical cluster analysis.

1. Introduction

Mapping high-dimensional data – usually containing many redundant observations – onto 1, 2 or 3 features that are highly informative, often is a useful first step in data analysis, as it allows to generate straightforward data visualizations such as histograms or scatter plots. A fundamental problem arising in this context is that there is no general answer to the question of how one is supposed to choose or even design a mapping that yields these informative features. Finding a suitable mapping typically requires prior knowledge about the given data. At the same time, knowledge is what we hope to be able to derive after mapping the data onto informative features. Frequently, one might know nothing or only very little about the given data. In any case, one needs to be very careful not to mistake crude assumptions for knowledge, as this may lead to a rather biased view on the data. So in summary, it appears as a closed loop "knowledge ⇒ mapping ⇒ informative features ⇒ knowledge", where each part ultimately depends on the given data and the only safe entry point is true knowledge.

Deep neural networks (DNNs) have been proven capable of tackling such problems. A DNN is a model that covers an almost infinite number of mappings, which is realized through millions of adjustable real-valued network parameters. By a learning process, the network parameters are gradually optimized (DNN *learning*) with respect to a criterion that indicates whether or not a mapping of a given data set is informative. Two DNNs that have been shown to be able to successfully learn useful data visualizations are the **Gen***eralized* **D***iscriminant* **A***nalysis* (GerDA) and **D***eep* **A***uto***E***ncoders* (DAEs) as suggested by Stuhlsatz et al. [38] and Hinton and Salakhutdinov [15], respectively. A closer look at these two DNNs reveals that the term "informative" may have quite different meanings.

GerDA is a nonlinear generalization of *Fisher's* **L***inear* **D***iscriminant* **A***nalysis* (LDA) [12] and thus considers discriminative features to be most informative, which appears as a very natural strategy to generate well-clustered visualizations of labeled data sets. DAEs, on the other hand, seek to improve an encoder/decoder mapping

$$f_{\mathrm{DAE}} := f_{\mathrm{dec}} \circ f_{\mathrm{enc}}, \tag{1}$$

where f_{enc} is a dimensionality reducing encoder (the desired feature mapping) and f_{dec} is the associated decoder. Practically, this is achieved by defining a criterion that measures the dissimilarity between the data and the reconstructions obtained by encoding and subsequent decoding. DAEs can therefore be learned without the use of class labels. Here, reconstructable features are considered most informative.

The **Re***gularized* **N***onlinear* **D***iscriminant* **A***nalysis* (ReNDA) [2] presented in this paper uses the combined criterion

$$J_{\mathrm{ReNDA}} := (1-\lambda) J_{\mathrm{GerDA}} + \lambda J_{\mathrm{DAE}} \quad (\lambda \in [0|1]), \tag{2}$$

* Corresponding author.
E-mail addresses: martin.becker@hs-duesseldorf.de (M. Becker), jens.lippel@hs-duesseldorf.de (J. Lippel), thomas.zielke@hs-duesseldorf.de (T. Zielke).

https://doi.org/10.1016/j.gmod.2020.101060
Received 10 August 2018; Received in revised form 18 October 2019; Accepted 12 February 2020
Available online 17 February 2020

where the two subcriteria J_{GerDA} and J_{DAE} are based on GerDA and a DAE, respectively. As the name suggests, we expect the associated ReNDA DNN to be better regularized. Regularization is a well-known technique to improve the generalization capability of a DNN. Regarding dimensionality reduction for data visualization, a good generalization performance is indicated by a reliably reproducible 1D, 2D or 3D feature mapping. In other words, a well-regularized DNN guarantees to learn sufficiently similar feature mappings for different sets of samples that all represent the data equally well. Clearly, such a robustness against fluctuations in the data is essential for many real-world applications.

Based on the belief that a feature mapping learned by a DNN should be as complex as necessary and as simple as possible, regularization of DNNs is traditionally imposed in the form

$$J_{\text{effective}} := J_{\text{obj}} + \lambda J_{\text{reg}} \quad (\lambda \in [0|\infty)), \tag{3}$$

which looks very similar to the combined criterion (2). Here, λ is a hyperparameter that is adjusted to control the impact of a regularization term J_{reg} on the DNN's true objective J_{obj}. Well-known approaches following (3) are *weight decay* (encouraging feature mappings that are more nearly linear) and *weight pruning* (elimination of network parameters that are least needed) [27]. Both these measures are intended to avoid the learning of overly complex mappings by imposing a restriction on the network parameters. In our opinion, one advantage of (2) over these two approaches can be stated as follows:

In (2), the restriction on the network parameters is implicitly imposed through encouraging a feature representation that is informative in two respects, i.e. the informativity of the feature representation is paramount. In the case of weight decay or pruning, only J_{obj} is responsible for the informativity of the learned feature representation. J_{reg} is usually neither designed to yield any outcomes regarding the given data nor meant to directly support or complement J_{obj}. Most often, examining the effects of J_{reg} on a DNN's learning process can only tell us what we might already know: The DNN covers overly complex feature mappings.

1.1. Related work

Dimensionality reduction may serve different purposes. The feature representations obtained can be used

(a) for subsequent classification [9,37].
(b) as compact representations for image retrieval [53].
(c) for data mining and data visualization [10,13].
(d) to analyze machine learning (ML) models [21].
(e) to enhance the robustness of ML models [3].

For each of these categories, there often exist several dimensionality reduction approaches that are suitable to solve the concrete task at hand. Useful overviews over unsupervised approaches are given by Lee and Verleysen [20], Sorzano et al. [35], Van Der Maaten et al. [45]. To the best of our knowledge, Chao et al. [5] is the first publication to present an overview over supervised dimensionality reduction approaches. Its focus lies on approaches that have counterparts in the field of unsupervised dimensionality reduction, e.g. isomap-based and locally linear embedding-based supervised dimensionality reduction.

Two of the most fundamental dimensionality reduction approaches are the *linear* **P***rincipal* **C***omponent* **A***nalysis* (PCA) and Fisher's LDA. While GerDA is a nonlinear generalization of Fisher's LDA, DAEs can be understood as a nonlinear generalization of the linear PCA [15] (a mathematical analysis that reveals the link between the linear PCA and *linear autoencoders* is presented in [1,26]). Therefore, the *hybrid PCA-LDA model* suggested by Zhao et al. [52] comes closest to a linear counterpart of ReNDA. The key to their hybrid model is a combined criterion with the same functional form as (2). They were able to show that any weighting between the PCA and the LDA criterion performs better than the PCA or the LDA alone.

The regularizing effect of a combined encoding/reconstruction criterion has been addressed in other publications that look at autoencoder-like neural networks, e.g. [22,30]. In [49], the term *autoencoder regularization* was introduced. The very recently introduced unsupervised dimensionality reduction approach SCVIS [10] seems to benefit from the described regularizing effect, too. Although SCVIS is introduced as a deep variational autoencoder [16,28] that uses the **t**-*distributed* **S***tochastic* **N***eighbor* **E***mbedding* (t-SNE) criterion [44] as an additional encoding criterion, it can also be viewed as follows: SCVIS is a DNN-based parametric t-SNE [43] that is extended to form a deep variational autoencoder. Because t-SNE has become a state-of-the-art dimensionality reduction approach and because SCVIS bears a certain resemblance to ReNDA, we consider SCVIS as a comparative approach in Section 5.

UMAP (short for **U***niform* **M***anifold* **A***pproximation and* **P***rojection*) has recently drawn much attention because of its outstanding runtime performance and the sound mathematical foundation laid out in [23]. It is competitive with t-SNE when learned unsupervisedly. With respect to ReNDA it is a far better comparative approach than SCVIS for its capability of supervised learning (Section 5).

1.2. Research contributions

We consider this paper to provide useful contributions for applications under the categories (c) and (d) (Section 1.1). Category (c) applies since we aim to further elaborate on an approach to dimensionality reduction that is especially suitable for data visualization. Since Sections 3, 4 and 5 present several strategies on how to assess and compare the robustness of dimensionality reduction approaches, category (d) is covered as well. In Section 4, we also show that ReNDA yields 2D feature representations that allow for a robust hierarchical clustering. The resulting hierarchical tree structures can be used to design hierarchical classifiers, a promising approach in the field of classification [18,33]. This indicates that future applications under category (a) may benefit from ReNDA as a data preprocessing step. In this context, 2D visualizations of the feature representations and hierarchical tree structures can be used to increase the transparency of the overall classification process.

Apart from serving a wide range of data analytics- and ML-related applications, our goal is to draw attention to a largely overlooked gap in ML research: While dimensionality reduction has become a standard data preprocessing step to realize robust ML models (cf. [46,50]), the robustness of the dimensionality reduction itself is rarely questioned. Currently, Chenouri et al. [6] appears to be the only other publication that directly addresses this gap in ML research. The results of our work show that this gap deserves greater attention.

2. ReNDA

As mentioned above, ReNDA is a combination of two different DNNs, GerDA and a DAE. As a matter of fact, both of these DNNs learn feature mappings in a very similar way, which is another reason why we considered this particular combination: They both use a Restricted Boltzmann Machine (RBM) pretraining to determine good initial network parameters, which are then used for subsequent gradient descent-based fine-tuning. The big difference between them is that a DAE has an encoding (f_{enc}) and a decoding (f_{dec}) DNN, whereas GerDA has only an encoding DNN. So contrary to a DAE, GerDA is unable to decode previously learned informative feature representation [2].

The idea behind ReNDA is to equip GerDA with a suitable decoding DNN. We introduce it in such a way that it has a regularizing effect on the encoding GerDA DNN. In the following, we focus on presenting the resulting ReNDA DNN as a well-regularized and therefore robust approach to dimensionality reduction for data visualization. (In [2, Section 3.2.4], we looked at ReNDA's capability to decode its feature representations.) In Fig. 1, we show a detailed data flow graph of ReNDA. In the following four subsections, we give a detailed explanation of all elements depicted in this figure.

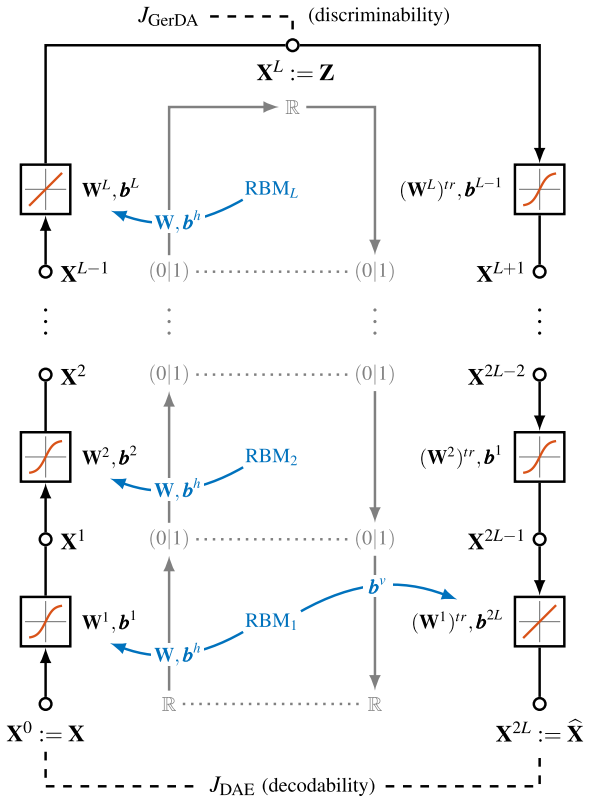


Fig. 1. A data flow graph of the overall $2L$-layered ReNDA DNN. Each layer is depicted as a box containing a symbolic plot of its activation function. The L layers on the left-hand side form the encoding and the L layers on the right-hand side form the decoding DNN (Sections 2.1 and 2.2). The inner "spaces flow graph" along with the RBMs and the curved arrows concern the RBM pre-training (Section 2.3). The GerDA criterion J_{GerDA} is connected to the feature space node by a dashed line, where it takes direct influence during fine-tuning (Section 2.4). Accordingly, J_{DAE} takes direct influence at the original space node and the reconstruction space node. From [2].

2.1. The encoding DNN

Suppose that the columns of $\mathbf{X} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \in \mathbb{R}^{d_{\mathbf{X}} \times N}$ are $d_{\mathbf{X}}$-dimensional samples and that $\boldsymbol{y} := (y_1, \ldots, y_N)^{tr} \in \{1, \ldots, C\}^N$ is a vector of class labels associated with these samples. ReNDA's objective is to find a DNN-based nonlinear encoding

$$\mathbf{X} \mapsto \mathbf{Z} := f_{\text{enc}}(\mathbf{X}) \in \mathbb{R}^{d_{\mathbf{Z}} \times N} \tag{4}$$

with $d_{\mathbf{X}} > d_{\mathbf{Z}} \in \{1, 2, 3\}$ that is optimal in the sense of an LDA for data visualization, i.e. that the features $\mathbf{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \in \mathbb{R}^{d_{\mathbf{Z}} \times N}$ are both well-clustered with respect to $\boldsymbol{y}$ and visualizable. The layerwise encoding shown on the left-hand side of Fig. 1 is obtained by setting $\mathbf{X}^0 := \mathbf{X}$, $d_0 := d_{\mathbf{X}}$, $\mathbf{X}^L := \mathbf{Z}$, $d_L := d_{\mathbf{Z}}$ and defining

$$\mathbf{X}^\ell := f_\ell(\underbrace{\mathbf{W}^\ell \mathbf{X}^{\ell-1} + \mathbf{B}^\ell}_{=: \mathbf{A}^\ell(\mathbf{X}^{\ell-1})}) \in \mathbb{R}^{d_\ell \times N} \tag{5}$$

for $\ell \in \{1, \ldots, L\}$ and intermediate dimensions $d_1, \ldots, d_{L-1} \in \mathbb{N}$. We refer to $d_0 - d_1 - d_2 - \cdots - d_L$ as the DNN topology. Further, $\mathbf{A}^\ell(\mathbf{X}^{\ell-1}) \in \mathbb{R}^{d_\ell \times N}$ is the ℓth layer's net activation matrix and it depends on the layer's adjustable network parameters: the weight matrix $\mathbf{W}^\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and the bias matrix $\mathbf{B}^\ell := (\boldsymbol{b}^\ell, \ldots, \boldsymbol{b}^\ell) \in \mathbb{R}^{d_\ell \times N}$. The function $f_\ell : \mathbb{R} \to \mathbb{R}$ is called the ℓth layer's activation function and it is applied entrywise, i.e.

$$x^\ell_{k,n} = f_\ell(a^\ell_{k,n}(\mathbf{X}^{\ell-1})) \tag{6}$$

yields the entries of $\mathbf{X}^\ell$. The encoding DNN's activation functions are set to $f_\ell := \text{sigm}$ with $\text{sigm} : \mathbb{R} \to (0|1)$ given by

$$\text{sigm}(x) := \frac{1}{1 + \exp(-x)} \qquad (x \in \mathbb{R}) \tag{7}$$

for $\ell \in \{1, \ldots, L-1\}$ and to $f_L := \text{id}$ with $\text{id} : \mathbb{R} \to \mathbb{R}$ given by

$$\text{id}(x) := x \qquad (x \in \mathbb{R}), \tag{8}$$

respectively. In Fig. 1 the activation functions are depicted as symbolic plots.

Altogether

$$f_{\text{enc}} = \underbrace{f_L \circ \mathbf{A}^L \circ \ldots \circ f_2 \circ \mathbf{A}^2 \circ f_1 \circ \mathbf{A}^1}_{\text{layerwise forward propagation}} \tag{9}$$

and optimizing it with respect to J_{GerDA} (Section 2.4.1) corresponds to the originally proposed GerDA fine-tuning [38]. The dashed link between J_{GerDA} and the $\mathbf{Z}$ node of the data flow graph shown in Fig. 1 is a reminder that $\mathbf{Z}$ is the GerDA feature space. With the decoding DNN presented in the next section, $\mathbf{Z}$ becomes the feature space of the overall ReNDA DNN.

2.2. The decoding DNN

As can be seen on the right-hand side of Fig. 1, the adjustable network parameters of the encoding DNN are reused for the decoding

$$\mathbf{Z} \mapsto \widehat{\mathbf{X}} := f_{\text{dec}}(\mathbf{Z}) \in \mathbb{R}^{d_{\widehat{\mathbf{X}}} \times N} \tag{10}$$

with $d_{\widehat{\mathbf{X}}} := d_{\mathbf{X}}$. The final biases $\boldsymbol{b}^{2L} \in \mathbb{R}^{d_{2L}}$ represent the only additional network parameters of ReNDA compared to GerDA. We summarize by

$$\boldsymbol{\theta} := (\underbrace{\mathbf{W}^1, \boldsymbol{b}^1, \ldots, \mathbf{W}^L, \boldsymbol{b}^L}_{\text{network parameters of the encoding DNN}}, \boldsymbol{b}^{2L}) \tag{11}$$

the network parameters of the ReNDA DNN. One of the main reasons for this kind of parameter sharing is that it connects f_{enc} and f_{dec} at a much deeper level than (2) alone. Observe that J_{GerDA} and J_{DAE} only take direct influence at three points of the ReNDA DNN. Because a DNN typically has millions of adjustable real-valued network parameters, there also lie millions of degrees of freedom between the two criteria. Thus, it is very likely that f_{dec} compensates for a rather poor f_{enc} or vice versa. In this case, the two mappings would not be working together. Considering this, we can specify what we mean by a connection of f_{enc} and f_{dec} at a deeper level: The parameter sharing ensures that the two DNNs work on the very same model. It makes the decoding DNN a supportive and complementing coworker that helps to tackle the existing task rather than causing new, independent problems. The idea of parameter sharing between two neural networks has also been investigated by others with so-called *Siamese architectures* [7].

We conclude this section with the mathematical formulation of the weight sharing as it is depicted in Fig. 1. To provide a better overview, we arranged the layers as horizontally aligned encoder/decoder pairs that share a single weight matrix: Layer $\ell = 2L$ uses the transposed weight matrix $(\mathbf{W}^1)^{tr}$ of the first layer. Layer $\ell = 2L - 1$ uses the transposed weight matrix $(\mathbf{W}^2)^{tr}$ of the second layer. So in general,

$$\mathbf{W}^\ell = \left(\mathbf{W}^{2L-\ell+1}\right)^{tr} \tag{12}$$

and $d_\ell = d_{2L-\ell}$ for $\ell \in \{L+1, \ldots, 2L\}$, which implies $d_{2L} = d_0 = d_{\mathbf{X}} = d_{\widehat{\mathbf{X}}}$. Note that the decoding DNN has the inverse encoding DNN topology $d_L - \ldots - d_0$. We can therefore still write $d_0 - \ldots - d_L$ for the DNN topology of the overall ReNDA DNN. In the case of the biases, we see that

$$\boldsymbol{b}^\ell = \boldsymbol{b}^{2L-\ell} \tag{13}$$

for $\ell \in \{L+1, \ldots, 2L-1\}$. Observe that (13) does not include the additional final decoder bias vector $\boldsymbol{b}^{2L}$ because there is no d_0-dimensional

encoder bias vector that can be reused at this point. The symbolic activation function plots indicate that

$$f_\ell = \begin{cases} \text{sigm} & L+1 \le \ell \le 2L-1 \\ \text{id} & \ell = 2L\,. \end{cases} \tag{14}$$

Finally, we have that

$$f_{\text{dec}} = f_{2L} \circ \mathbf{A}^{2L} \circ \ldots \circ f_{L+1} \circ \mathbf{A}^{L+1} \tag{15}$$

with $\mathbf{A}^{2L}, \ldots, \mathbf{A}^{L+1}$ according to (5). It is

$$\mathbf{X} \mapsto \widehat{\mathbf{X}} = (f_{\text{dec}} \circ f_{\text{enc}})(\mathbf{X}) = f_{\text{DAE}}(\mathbf{X}) \tag{16}$$

and optimizing f_{DAE} with respect to J_{DAE} (Section 2.4.2) corresponds to the originally proposed DAE fine-tuning [15]. J_{DAE} measures the dissimilarity between the samples $\mathbf{X}$ and its reconstructions $\widehat{\mathbf{X}}$. In the data flow graph shown in Fig. 1 this is symbolized by a dashed line from the $\mathbf{X}$ node to J_{DAE} to the $\widehat{\mathbf{X}}$ node.

2.3. RBM pretraining

Both GerDA and DAEs use an RBM pretraining in order to determine good initial network parameters. In this context, "good" means that a subsequent gradient descent-based fine-tuning has a better chance to approach a globally optimal mapping. Randomly picking a set of initial network parameters, on the other hand, almost certainly leads to mappings that are rather poor and only locally optimal [11]. As an in-depth explanation of the RBM pretraining would go beyond the scope of this paper, we only give a brief description of the RBM elements shown in the data flow graph (Fig. 1).

We see that there exists an RBM for each horizontally aligned encoder/decoder layer pair. Each RBM_L for $\ell \in \{1, \ldots, L\}$ is equipped with a weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$, a vector $\boldsymbol{b}^v \in \mathbb{R}^{d_{\ell-1}}$ of visible biases and a vector $\boldsymbol{b}^h \in \mathbb{R}^{d_\ell}$ of hidden biases. Once pretrained, the weights and biases are passed to the DNN as indicated by the curved arrows. This is exactly the same way in which the network parameters of the original GerDA DNN are initialized. Again, the only exception is the final bias vector $\boldsymbol{b}^{2L}$. Here, the bias $\boldsymbol{b}^v$ of RBM_1 is used. The initialization of the remaining network parameters of the decoding DNN follows directly from the parameter sharing (12) and (13) introduced in Section 2.2.

2.4. Fine-tuning

For the gradient descent-based fine-tuning, we need to specify the two criteria J_{GerDA} and J_{DAE}. When combining the two criteria one has to pay attention to their orders of magnitude. We found the following normalized criteria to be best working.

2.4.1. Normalized GerDA criterion

The original GerDA criterion is given by

$$Q_z^\delta := \text{trace}\left(\left(\mathbf{S}_T^\delta\right)^{-1} \mathbf{S}_B^\delta \right). \tag{17}$$

In [38], it has been shown that maximizing Q_z^δ yields well-clustered, visualizable features. The two matrices appearing in (21) are: The weighted total scatter matrix

$$\mathbf{S}_T^\delta := \mathbf{S}_W + \mathbf{S}_B^\delta \tag{18}$$

with the common (unweighted) within-class scatter matrix $\mathbf{S}_W := (1/N)\sum_{i=1}^C N_i \boldsymbol{\Sigma}_i$ of the class covariance matrices $\boldsymbol{\Sigma}_i := (1/N_i)\sum_{n\,:\,y_n=i}(\boldsymbol{z}_n - \boldsymbol{m}_i)(\boldsymbol{z}_n - \boldsymbol{m}_i)^{tr}$ with the class sizes $N_i := \sum_{n\,:\,y_n=i} 1$ and the class means $\boldsymbol{m}_i := (1/N_i)\sum_{n\,:\,y_n=i} \boldsymbol{z}_n$. The weighted between-class scatter matrix

$$\mathbf{S}_B^\delta := \sum_{i,j=1}^C \frac{N_i N_j}{2N^2} \cdot \delta_{ij} \cdot (\boldsymbol{m}_i - \boldsymbol{m}_j)(\boldsymbol{m}_i - \boldsymbol{m}_j)^{tr} \tag{19}$$

with the global symmetric weighting

$$\delta_{ij} := \begin{cases} 1/\|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2 & i \ne j \\ 0 & i = j. \end{cases} \tag{20}$$

Clearly, δ_{ij} is inversely proportional to the distance between the class means $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$. The idea behind this is to make GerDA focus on classes i and j that are close together or even overlapping, rather than ones that are already far apart from each other.

For ReNDA, we modified Q_z^δ as follows:

$$J_{\text{GerDA}} := 1 - \frac{Q_z^\delta}{d_{\mathbf{Z}}} \in (0|1) \tag{21}$$

The division through $d_{\mathbf{Z}}$ is the actual normalization (Appendix A). Subtracting this result from one makes J_{GerDA} a criterion that has to be minimized, which is necessary in order to be able to perform gradient descent for optimization. See Appendix B for the partial derivatives of J_{GerDA}.

2.4.2. Normalized DAE criterion

Directly applying the classical mean squared error

$$\text{MSE} := \frac{1}{N} \, \|\widehat{\mathbf{X}} - \mathbf{X}\|_F^2 \in [0|\infty) \tag{22}$$

with Frobenius norm

$$\|\mathbf{U}\|_F := \sqrt{\sum_{i=1}^m \sum_{j=1}^n |u_{i,j}|^2} \quad (\mathbf{U} \in \mathbb{R}^{m \times n}) \tag{23}$$

as the DAE criterion is problematic because it is typically considerably greater than Q_z^δ. Note that (21) implies $Q_z^\delta \in (0|d_{\mathbf{Z}})$. In the context of dimensionality reduction for data visualization, where $d_{\mathbf{Z}} \in \{1, 2, 3\}$, this difference in order of magnitude is especially large. We therefore modify the DAE criterion in the following way:

$$J_{\text{DAE}} := \frac{\text{MSE}/d_{\mathbf{X}}}{1 + \text{MSE}/d_{\mathbf{X}}} \in [0|1) \tag{24}$$

The division through $d_{\mathbf{X}}$ was arbitrarily introduced. Together with N, the denominator $d_{\mathbf{X}}$ prenormalizes $\|\cdot\|_F^2$ before the final normalization $(\cdot)/[1 + (\cdot)]$. However, there may be better ways of defining a normalized DAE criterion. This is subject to future work. With J_{GerDA} (see (21)) and J_{DAE} having the same bounded codomain, their combination is less problematic. The partial derivatives of J_{DAE} can be found in Appendix C.

3. Experiments

DNNs are able to successfully learn dimensionality-reducing mappings that yield informative, visualizable features. This claim has been experimentally proven for both GerDA and DAEs. In [15,38], respectively, the widely used MNIST database of handwritten digits [19] has been mapped into a 2D feature space. In another example, GerDA has been used for an emotion detection task. Here, 6552 acoustic features extracted from speech recordings were reduced to 2D features that allow to detect and visualize levels of valence and arousal [39].

In the following two sections, we experimentally show that ReNDA is also able to successfully learn feature mappings for data visualization and that these mappings are robust against fluctuations in the data. In order to be able to see this improvement in regularization, we ran all experiments for both ReNDA and GerDA and compared their results.

Throughout all of the ReNDA experiments we set $\lambda = 0.5$, mainly because it avoids a prioritization of any of the two criteria J_{GerDA} and J_{DAE} (see (2)). Although not subject of the work presented in this paper, optimizing λ may improve the results for a particular application. This has been shown in [1].

3.1. Artificial galaxy data set

For a first experiment, we used the artificially generated galaxy-shaped data set shown in Fig. 2. Although it is already very easy to visualize, DNN learning of optimal 1D features is still challenging. The

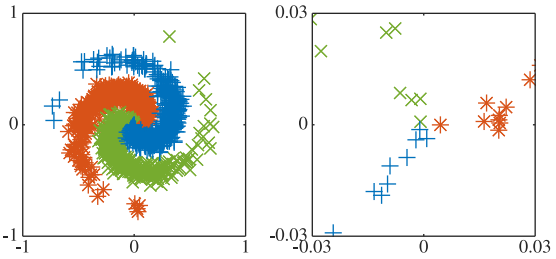


Fig. 2. A scatter plot of the artificial galaxy data set. The plot on the right-hand side shows a zoom of the center point of the galaxy. We see that the 3 classes are in fact non-overlapping but very difficult to separate. From [2].

reason why we chose to use an artificial rather than a real-world data set is that most interesting real-world data sets are far too complex to obtain fast results. In the case of the galaxy data set, the associated DNN parameters are relatively fast to compute, which made it possible to run very extensive experiments but with reasonable computational effort.

3.1.1. Experimental setup

The main goal of this experiment is to investigate the influence of fluctuations in the data on the learned ReNDA and GerDA visualizations. The results allow us to compare these two approaches with regard to their robustness.

We simulated fluctuations in the data by taking 10 distinct sets of samples from the galaxy data set, which were then used for 10 ReNDA and 10 GerDA runs. In detail, each of the 10 galaxy sets contains 1440 samples (480 per class) that were presented for DNN learning, and additional 5118 samples (1706 per class) that were used for validation. Further details on how the samples are presented for DNN learning can be found in Appendix D.

For both ReNDA and GerDA we chose the DNN topology 2-20-10-1. This choice is based on the very similar 3-40-20-10-1 DNN topology that [38] used to learn informative 1D features from a 3-class artificial Swiss roll data set. Removing the intermediate dimension 40 made DNN learning more challenging while reducing the computational effort. In other words, it yielded a less flexible DNN mapping with fewer parameters to optimize.

An important aspect to consider is that the algorithmic implementations of both ReNDA's and GerDA's DNN learning, involve the use of a random number stream. In this experiment we ensured that this stream is the same for all 10 ReNDA and all 10 GerDA runs. The initial network parameters of the RBM pretraining are also based on this stream, which implies that we do not include any potentially biased parameter initializations. As a consequence, any fluctuations in the ReNDA and GerDA results are due to the simulated fluctuations in the data only.

3.1.2. 1D visualization

We now compare the 1D mappings obtained from the 10 ReNDA and the 10 GerDA runs. To that end, we use class-conditional histograms as a straightforward method for 1D visualization. This is best explained by directly discussing the results. We begin with the ReNDA results shown in Fig. 3(a) and discuss the GerDA results (Fig. 3(b)) afterwards.

The top row of small plots in Fig. 3(a) shows the results of the individual ReNDA runs. Each of these plots includes 3 distinct relative histograms that are based on standardized 1D features associated with the validation samples: One that considers the samples in the red or asterisk (∗) class, a second for the green or cross mark (×) class, and a third for the blue or plus mark (+) class. The large plot in Fig. 3(a) represents an overlay of all small plots. Note that the axis limits of all 11 plots are identical. Therefore, the overlay plot indicates a high similarity between the learned 1D mappings. Only the order of the 3 classes changes throughout the different ReNDA runs, which is due to the symmetry of the galaxy data set.

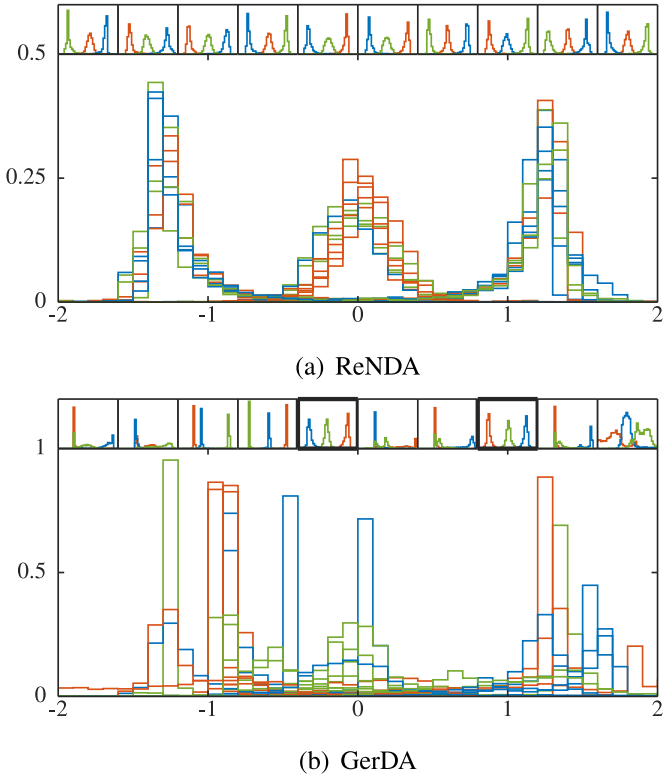


Fig. 3. A comparison of the 1D mappings learned by ReNDA (a) and GerDA (b). The top row of small subplots in (a) and (b), respectively, shows the histograms of the 1D features associated with the validation samples of each of the 10 galaxy data sets. The large plots represent overlays of these 10 subplots. From [2].

The corresponding GerDA histograms shown in Fig. 3(b) are organized in the very same way as in Fig. 3(a), i.e. two small histograms that have the same position in Fig. 3(a) and (b) are based on the same 1440 samples for DNN learning and the same 5118 samples for validation. However, here we used differently scaled vertical axes depending on the maximum bar height of each histogram. Observe that only the two bold-framed histograms are similar to the ReNDA histograms. The GerDA overlay plot shows that the 1D mappings learned by GerDA are significantly less similar to each other than those learned by ReNDA. In the case of GerDA, the three classes are hard to detect, whereas for ReNDA we obtained 3 bump-shaped and easy to separate clusters. This indicates that regularizing GerDA in order to obtain ReNDA yields the desired robustness improvement.

3.2. Handwritten digits

The artificial galaxy data set used above is neither high-dimensional nor an interesting example in terms of practical applications. We therefore decided to carry out further experiments with the well-known MNIST database of handwritten digits [19], a widely used real-world benchmark data set for the testing of DNN learning approaches.

MNIST contains a large number of samples of handwritten digits 0 to 9 stored as grayscale images of 28 × 28 pixels. These samples are organized as two subsets: a *training set* containing 60k samples and a *test set* of 10k samples. With its 28 × 28 pixel images and variations in the handwriting it falls into the category of big dimensionality data sets as discussed in [51]. Nevertheless, there are no visible non-understood fluctuations present, which is important for our experimental setup. As before, we want to simulate the fluctuations in order to see their effect on the feature mappings.

3.2.1. Experimental setup

The setup of this experiment slightly differs from the previous one. Again we considered fluctuations in data but also fluctuations in the random number stream that both ReNDA and GerDA depend on (Section 3.1.1). In practice, the latter fluctuations are especially present when DNN learning is performed on different computer architectures: Although we have not studied these effects systematically yet, we believe that different rounding procedures may lead to significantly dissimilar feature representations even if the same samples are used for DNN learning. Instead of conducting experiments on distinct computer architectures, we pursued the following strategy in order to obtain a cleaner experimental setup:

In this experiment, we simulated these fluctuations in the random number stream simply by generating 3 distinct random number streams with a single random number generator. The fluctuations in the data were simulated via 3 distinct random partitions of the 60k training samples into 50k samples presented for DNN learning, and 10k samples for validation. Finally, we combined each of these 3 partitions with each of the 3 random number streams, which then allowed us to realize 9 ReNDA and 9 GerDA runs. Further details on how the samples are presented for DNN learning can be found in Appendix D.

For both ReNDA and GerDA we selected the DNN topology 784-1500-375-750-2 that was also used in [38], allowing a direct comparison of our results.

3.2.2. 2D visualization

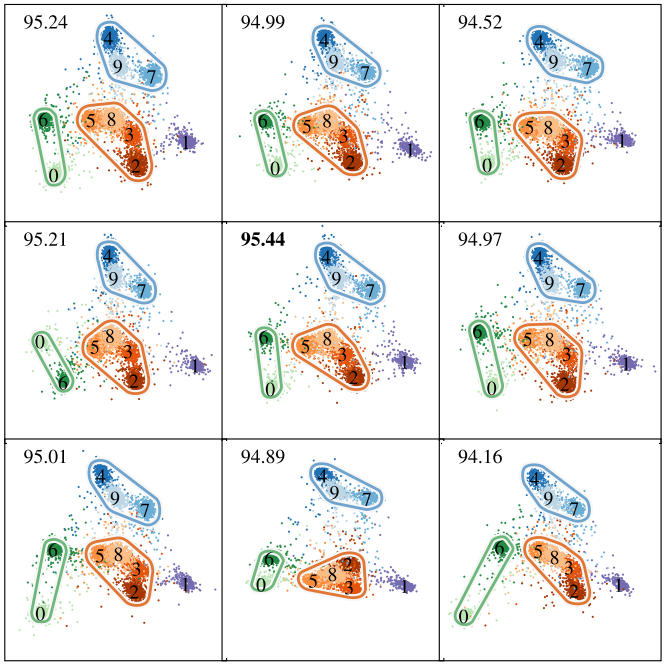
In the following, we demonstrate ReNDA's improved robustness compared to its predecessor, GerDA, by two means: We look at 2D scatter plot visualizations and the *distance consistency measure* DSC suggested by Sips et al. [34] to assess the quality and the robustness of the underlying 2D mappings. The DSC score has been shown to be a first choice in terms of the ability to imitate human perception [32]. Recently an improvement of the DSC score was suggested, the density-aware DSC (dDSC) [47]. Although possibly relevant for our work too, we used the original DSC score from [32].

The scatter plots in Fig. 4(a) show the results of the 9 ReNDA runs. Each column corresponds to 1 of the 3 partitions of the 60k training samples and each row corresponds to 1 of the 3 random number streams as described in the previous section. The 2D features depicted are based on the 10k validation samples of the respective run. Fig. 4(b) shows the associated GerDA scatter plots and it is organized in the very same way, i.e. two scatter plots that have the same position in Fig. 4(a) and (b) are based on the same combination of a training set partition and a random number stream.

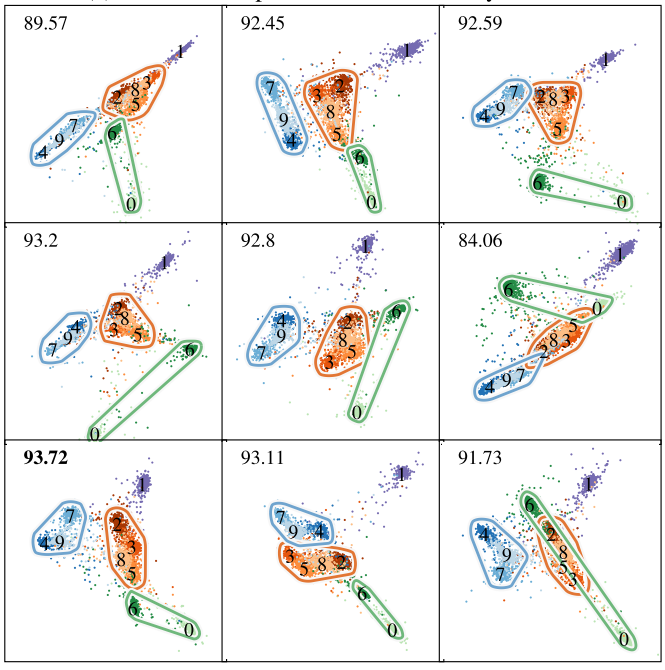
The value given in the top left corner of each scatter plot is the associated DSC score. It is based on the respective 10k validation samples. A DSC of 100 means that all data points have a smaller Euclidean distance to their own class centroid than to any other. It is a good measure of the compactness of class-related clusters that can be directly applied to any low-dimensional feature representation even if the underlying original set of samples is not available. Table 3 shows a comparison of the DSC scores of ReNDA, GerDA and 4 comparative approaches (Section 5).

As can be seen in Fig. 4(a), ReNDA yields reliably reproducible feature representations, too. In order to make this more evident, we pursued the following strategy:

First, we standardized each feature representation as a means to remove any translatory and scaling differences. Secondly, a mirroring and a rotation were applied if needed to obtain similarly orientated feature representations. In the second step, the top left scatter plot in Fig. 4(a) served as the reference. The same steps were applied to the GerDA feature representations shown in Fig. 4(b). For each scatter plot, the applied overall affine transformation is determined based on the 50k samples presented for DNN learning. The transformation of the feature representations does not have any effect on the DSC scores. This can be easily verified by comparing the DSC scores to those published in [2].



(a) 2D feature representations learned by ReNDA



(b) 2D feature representations learned by GerDA

Fig. 4. A comparison of the 2D feature representations learned by ReNDA (a) and GerDA (b). In both (a) and (b), the class centroids are marked with the associated digits. The coloring of the data points is based on the easy-to-spot meta-clusters **2-3-8-5, 7-9-4** and **0-6** in (a). We applied a single-hue color scheme per meta-cluster and added a matching border per meta-cluster. The selected colors were transferred to (b). In (a) and (b), each column corresponds to 1 of the 3 partitions of the training samples and each row corresponds to 1 of the 3 random number streams. The value in the top left corner of each scatter plot states the corresponding distance consistency measure DSC [34]. Please note that Fig. 7 shows a larger version of the top left scatter plot of Fig. 4(a).

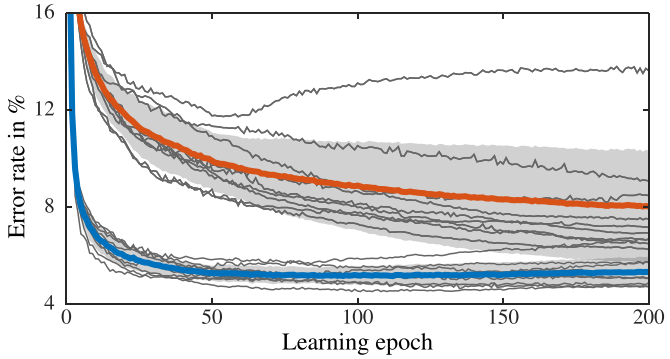


Fig. 5. A comparison of the learning curves associated with **ReNDA** and **GerDA**. Each learning curve shows the validation classification error (error for short) as a function of the learning epoch. The lower **blue** and the upper **red** curve represent the average errors. The light gray ribbon surrounding each of the two indicates the corresponding standard deviations per epoch. The thinner dark gray curves show the actual errors per epoch of the 9 ReNDA runs and the 9 GerDA runs. From [2]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To further highlight the high degree of similarity of the 9 ReNDA scatter plots, we colored the data points based on the easy-to-spot meta-clusters **2-3-8-5, 7-9-4** and **0-6**. We applied a single-hue color scheme per meta-cluster and also added a matching border per meta-cluster (Fig. 4(a)). The selected colors were transferred to Fig. 4(b). The coloring very well reflects the extent to which the ReNDA approach enhances the GerDA approach. In most cases, GerDA yields the same meta-clusters. However, their relative positioning appears less reproducible than in the case of ReNDA. Also, in the case of GerDA, there appears to be a bit more variation between the individual digit clusters contained in the **2-3-8-5** meta-cluster. They differ in their order and in the overall shape they produce.

3.2.3. Robust learning behavior

It is natural to assume that the above final results are due to a more robust, more efficient and more targeted learning behavior. To test this, we compare the two learning curves depicted in Fig. 5. The curves show the Bayesian linear classification error (error for short) as a function of learning epochs; the iterative steps of DNN learning. The error is a measure of class separability and we calculated it based on the respective 10k validation samples. The reason we use it is to show that ReNDA also performs well on classification tasks.

In detail, we see the average ReNDA error (lower emphasized, **blue** curve) and the average GerDA error (upper emphasized, **red** curve). Both are surrounded by a light gray ribbon indicating the corresponding standard deviation per epoch. The thinner dark gray curves represent the errors of the 9 ReNDA and the 9 GerDA runs, respectively.

The assumed more robust learning behavior of ReNDA is evident because throughout all learning epochs its standard deviation is significantly smaller than that of GerDA. Also its average learning curve is almost constant after epoch 50 whereas GerDA's average learning curve is still falling at epoch 200, which surely can be interpreted as a more efficient and more targeted learning behavior.

3.2.4. Unsupervised cluster detection

Yet another way to assess the learned dimensionality reductions is to simulate the absence of class labels and examine how far an unsupervised clustering algorithm can detect clusters that are consistent with the known class information. In contrast to the previous two sections, we ran this analysis using the feature representations corresponding to the 50k samples presented for DNN learning. The reason for this is that we do not consider it an analysis in its own right. It will be the first step of a future hierarchical classification approach where the validation samples must be saved for a final overall performance assessment.

In this section, we focus on the necessary initial task of finding the number of clusters that appears to be most consistent with a given feature representation. In close to ideal scenarios, feature representations that are well-clustered with respect to their associated class labels would consist of exactly one cluster per class. Here, each individual cluster would be easy to distinguish from all other clusters (*separability*), and all samples that are assumed to form a cluster would not deviate much from the respective cluster mean (*compactness*). Simultaneously assessing the overall separability and compactness of a set of clusters is key to several clustering validity indices (CVIs).

As can be seen in Table 1, we use 4 CVIs (*Caliński-Harabasz* [4], *Davies-Bouldin* [8], *gap statistic* [42] and *silhouette* [29]) to validate 3 clustering algorithms: The first is the well-known K-means algorithm. The second can be viewed as a variation of the K-means algorithm. Instead of assuming $K \in \mathbb{N}$ cluster centroids, a K-components *Gaussian Mixture Model* (GMM) is fitted to the data. The third is a hierarchical clustering algorithm using *Ward's method* [48]. It yields a hierarchical tree structure where each leaf represents a cluster. We denote by $K \in \mathbb{N}$ the number of leaves of such a hierarchical tree structure. The following experimental setup can be easily realized using `evalclusters` – a functionality provided by MATLAB®'s Statistics and Machine Learning Toolbox.

For each combination of a clustering algorithm with a CVI we identified the number of clusters $K^* \in \{1, \ldots, 20\}$ with the best validity index. Clearly, in the case of the MNIST feature representations, $K^* \neq 10 = C$ can be seen as a false prediction of the number of classes. The number of false predictions per combination is stated by the first value in each table cell. The best prediction result is 0. Because we carried out

Table 1
Cluster evaluation results obtained for the ReNDA and GerDA feature representations shown in Fig. 4(a) and (b), respectively. The first value in each table cell states the number of false predictions where a false prediction is indicated by $K^* \neq 10 = C$. The best prediction result is 0. Since we carried out 9 ReNDA and 9 GerDA runs, the worst prediction result is 9. The value in parentheses states the average absolute deviation of the false predictions. It is not defined if the number of false predictions is 0.

Clustering validity index	Dimensionality reduction approach	Clustering algorithms		
		K-means	K-comp. GMM	Ward's method
Caliński-Harabasz	ReNDA	0	0	1(1.0)
	GerDA	3 (1.0)	4 (1.5)	5 (1.2)
Davies-Bouldin	ReNDA	0	0	2 (1.0)
	GerDA	3 (5.0)	4 (5.5)	9 (3.8)
Gap Statistic	ReNDA	0	0	1 (1.0)
	GerDA	3 (1.7)	2 (1.0)	3 (1.3)
Silhouette	ReNDA	1 (1.0)	5 (1.6)	3 (1.0)
	GerDA	6 (2.3)	6 (4.0)	7 (3.1)

9 ReNDA runs and 9 GerDA runs the worst prediction result is 9. The value in parentheses states the associated average absolute deviation of these false predictions. It is not defined if the number of false predictions is 0.

Let us focus on the left-most table cell of the first GerDA row to get an idea of how to interpret the second value: It is easy to see that the false predictions are either 9 or 11, i.e. they deviate by either +1 or −1. If any absolute deviation had been > 1, the second value would also have been > 1. For instance, the left-most table cell of the third GerDA row clearly reveals that K^* deviated by ± 1 once, and by ± 2 twice, since $(1 + 2 + 2)/3 \approx 1.7$.

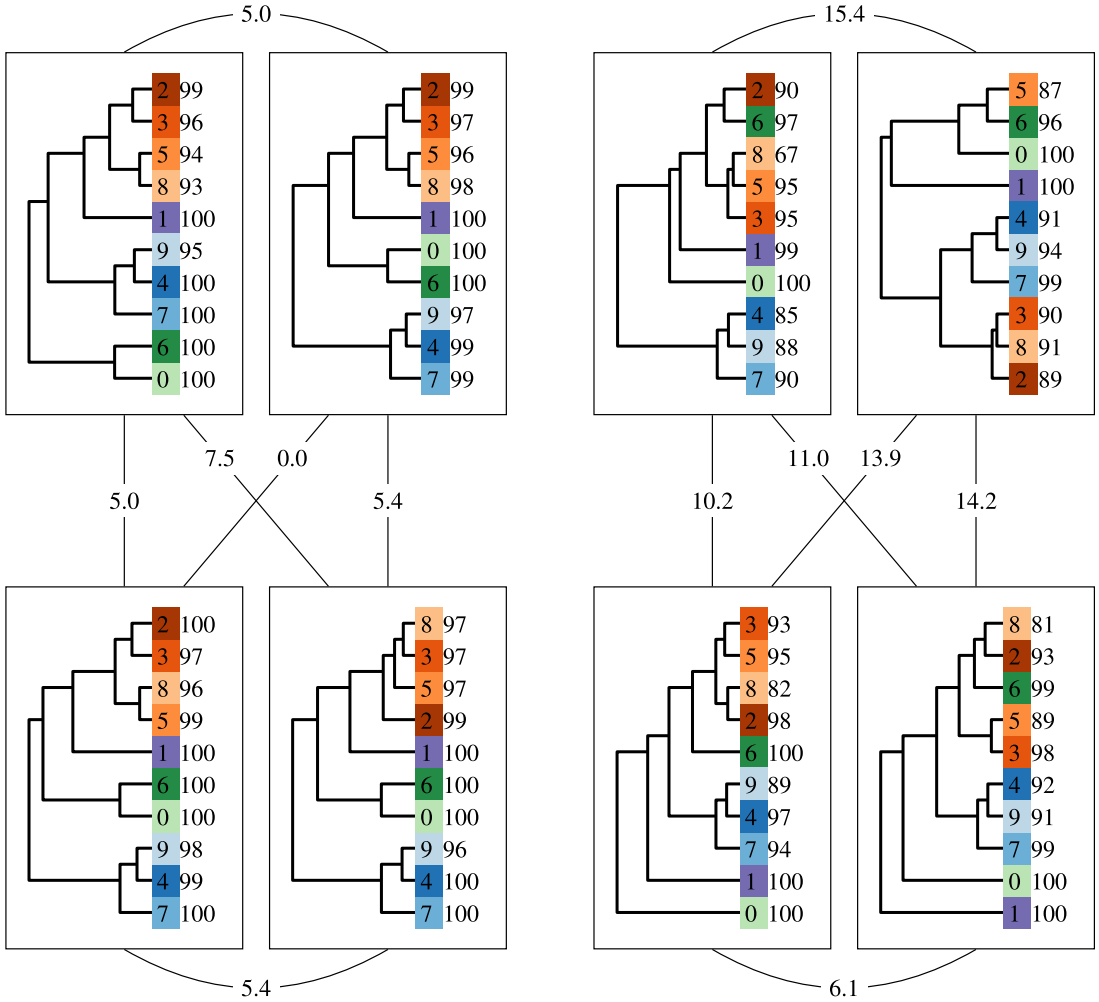
It is striking that in the case of ReNDA, half of the different combinations of a clustering algorithm and a CVI fully agree upon the correct number of clusters of $K^* = 10 = C$. The only true exception is the combination of the GMM clustering algorithm with the silhouette validity index. The second highest number of false predictions is 3 (combination of Ward's method and the silhouette validity index), which appears to be about the best result that one can expect for a feature representation provided by its predecessor, GerDA.

Although Ward's method did not prove as reliable as the K-means and the K-components GMM clustering algorithm, it is worthwhile to take a closer look at the hierarchical tree structures provided. In the subsequent section, we show how they can be used to assess the structural dissimilarities between different feature representations. Furthermore, we present a suitable strategy on how to examine whether a suggested set of clusters is in fact class-related.

4. Structural dissimilarity

In the following subsections, we show that the hierarchical tree structures obtained by Ward's method [48] reflect structural dissimilarities between different feature representations. By structural dissimilarities we mean any dissimilarities regarding the relative positioning of a set of clusters. In this context, assessing the detectability of clusters in terms of separability and compactness has been a useful first step (Section 3.2.4). We assume that the structural dissimilarity assessment presented works with any hierarchical clustering algorithm and it is part of our future work to carry out and present a comparison.

Since we are already familiar with the feature representations obtained for the MNIST data set, we continue using them as an example. In both sections, we assume $K^* = 10 = C$ to be the number of clusters, regardless of the fact that it has not been reliably predicted in the case of the GerDA feature representations. The sections describe the different elements depicted in Fig. 6(a) and (b), respectively. They contain *dendrograms* (Section 4.1) visualizing the hierarchical tree structures cor-



(a) ReNDA (b) GerDA

Fig. 6. Dendrograms (Section 4.1) corresponding to the four top left scatter plots in Fig. 4(a) and (b), respectively. The leaves are colored applying the same single-hue color scheme as in the case of the scatter plots. The value stated right of each dendrogram leaf indicates the reliability (in %) of the label assigned to that leaf. The labeled links connecting the rectangular boxes indicate the pairwise tree dissimilarity measures (TDMs) presented in Section 4.2.

responding to the four top left scatter plots shown in Fig. 4(a) and (b), respectively. The labeled links indicate the pairwise structural dissimilarity of the shown tree representations. The considered dissimilarity measure has been suggested by Saunders et al. [31]; a brief explanation is given in Section 4.2.

Like the previously performed cluster evaluation (Section 3.2.4), the hierarchical tree structures discussed in the following are based on the 50k samples presented for DNN learning. Again, this is done with regard to a future hierarchical classification approach where the validation samples must be saved for a final overall performance assessment.

4.1. Dendrograms

The result of a hierarchical clustering is often displayed graphically using a tree-like diagram called a dendrogram. It reveals both the cluster-subcluster relationships and the order in which the clusters were merged. In Fig. 6, the dendrograms are shown inside rectangular boxes. In each dendrogram, the 10 leaves are labeled with the respective MNIST digit they represent. The strategy we pursued to label the leaves also provides a good way to examine how far a suggested set of clusters is in fact class-related – an issue we left open in Section 3.2.4.

For each leaf, we looked up the class labels of all data points of the respective cluster and determined the class label occurring with the highest percentage. The resulting percentages are stated to the right of each leaf's label in Fig. 6. Clearly, a high value per leaf indicates a reliable overall labeling. Of course, each individual label's percentage should only be the highest at exactly one leaf. Moreover, a unique highest percentage per leaf is required. In the case of ReNDA and GerDA, the above strategy yielded unambiguous labelings.

We applied the same single-hue color scheme to the leaves as in the case of the scatter plots (Fig. 4), in order to facilitate the comparison of scatter plots and dendrograms belonging together. In the ReNDA dendrograms shown in Fig. 6(a), note how the single-hue color scheme per meta-cluster agrees with the final leaf positions. From this, we can conclude that this meta-class-based visual assessment was encoded into the hierarchical tree structures. As a consequence, an automatic detection of such meta-classes should be realizable by exploiting the underlying cluster-subcluster relationships of a hierarchical representation.

Fig. 7 shows an example of how the cluster-subcluster relationships can be transferred back to a 2D scatter plot visualization. The scatter plot corresponds to the top left scatter plot in Fig. 4(a). Accordingly, the cluster-subcluster relationships, here annotated via encircling the meta-clusters at all different levels of merging, can be seen in the top left dendrogram of Fig. 6(a). The borders of the meta-clusters **2-3-8-5, 7-9-4** and **0-6** – including their colors – are identical to those in the top left scatter plot of Fig. 4(a).

The fact that the ReNDA dendrograms differ slightly from each other indicates that subtle dissimilarities between the feature representations were encoded into the corresponding set of hierarchical tree structures. In the case of GerDA, we see that evidently different 2D feature representations (Fig. 4(b)) yield to evidently different dendrograms (Fig. 6(b)). We thus conclude that applying hierarchical clustering algorithms can lead to suitable tools for the robustness assessment of DNN-based dimensionality reductions.

4.2. A Tree Dissimilarity measure

Recently, Saunders et al. [31] proposed a tree stability measure they called *minimal containing clade* (MCC), a term that is biologically motivated and does not fit well into the context of our work. A closer look at its definition reveals that the term *tree dissimilarity measure* (TDM) is suitable and also more generally applicable: The idea behind it is to first characterize each hierarchical tree structure through the vector containing all leaf-to-leaf edge distances, where a leaf-to-leaf edge distance is simply the number edges forming the path from a first leaf to a second leaf. Given two such vectors the TDM of the corresponding hierarchical tree structure is defined to be the Euclidean distance of those vectors.

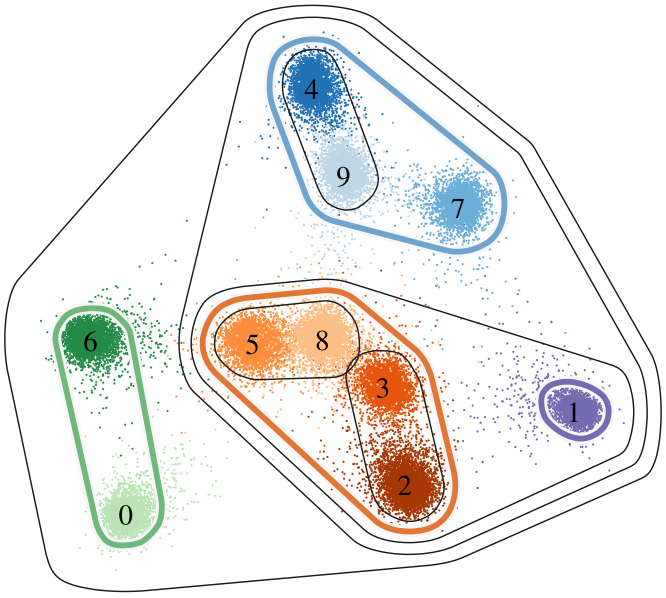


Fig. 7. An example of how the cluster-subcluster relationships can be transferred back to the top left scatter plot of Fig. 4(a). The cluster-subcluster relationships shown in the top left dendrogram of Fig. 6(a) are annotated via encircling the meta-clusters at all different levels of merging. The borders of the meta-clusters **2-3-8-5, 7-9-4** and **0-6** – including their colors – are identical to those in the top left scatter plot of Fig. 4(a).

In Fig. 6, the resulting pairwise TDMs are displayed as labeled links between the boxes containing the dendrograms. The average TDM and its standard deviation based on all 9 ReNDA and all 9 GerDA runs are given by 4.9 ± 3.0 and 11.1 ± 2.8, respectively. The fact that these values agree with the visual robustness assessment presented so far, leads to interesting future perspectives in the context of the robustness assessment of DNN-based dimensionality reduction approaches, not only in the field of data visualization.

For instance, having access to a scalar measure allows for a learning curve-like visualization that can be used to monitor DNN learning processes. In the case of the TDM, this form of visualization would consist of comparative curves each showing how two distinct DNN learning processes converge or diverge. It is subject to our future work to explore the opportunities of such robustness assessment approaches.

5. Comparative results

So far, we have only compared ReNDA to its direct predecessor GerDA. In order to be able to provide further comparative results, we ran our 9 MNIST experiments (Section 3.2) with 4 other dimensionality reduction approaches. We look at the *deep belief net* (DBN) [40,41] that we already considered in [2] and UMAP [24]; both were learned supervisedly. In addition, we look at 2 unsupervised approaches: the plain DAE obtained for $\lambda = 1$ in (2) and SCVIS [10]. An overview over all relevant experiment details is given in Appendix D.

In Section 5.3, we give a summary of the results that covers all approaches including ReNDA and GerDA. In addition, the section states the approximate computation time and memory usage per approach.

5.1. 2D visualization

Fig. 8 shows the respective scatter plot visualizations. The 2D feature representations shown are based on our first 2 partitions of the 60k training samples and 2 different random number streams. Fig. 8(a) to (f)

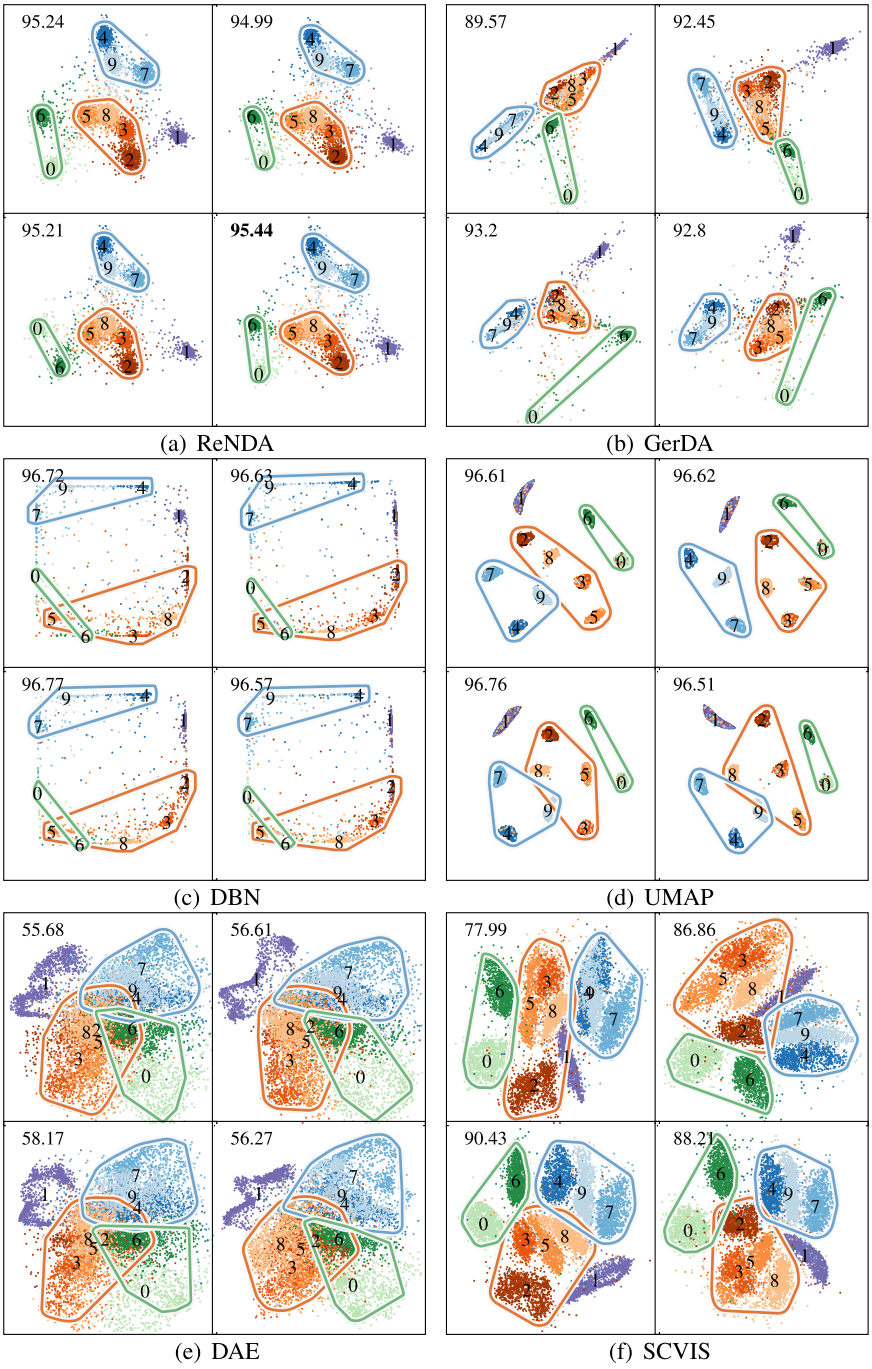


Fig. 8. A comparison of the 2D feature representations learned by the ReNDA (a), GerDA (b), DBN (c), UMAP (d), the plain DAE (e) and SCVIS (f). The ReNDA/GerDA results from Fig. 4 are repeated to enable a faster comparison. In all scatter plots, the class centroids are marked with the associated digits. The coloring of the data points is based on the easy-to-spot meta-clusters **2-3-8-5, 7-9-4** and **0-6** in (a). We applied the same single-hue color scheme per meta-cluster and added a matching border per meta-cluster as in Section 3.2.2. In (a) to (f), the columns correspond to the first and the second of our 3 partitions of the training samples and the rows correspond to 2 different random number streams. The value in the top left corner of each scatter plot states the corresponding distance consistency measure DSC [34].

are organized as Fig. 4(a) and (b), respectively: The columns correspond to the partitions, the rows to the random number streams. The DSC scores [34] are stated in the top left corner of each scatter plot. Again, all results are based on the 10k validation samples of the respective partition. To remove any translatory, scaling and rotational differences between the scatter plots, we applied the affine transformation introduced in Section 3.2.2. To simplify the direct comparison of these and the ReNDA and GerDA scatter plots, we reuse the color scheme from the previous sections. In addition to the scatter plots, we show all corresponding dendrograms in Figs. 9 and 10. They were generated as those in Fig. 6 (Section 4.1).

As already pointed out in [2], the DBN learns a highly robust dimensionality reduction. The fact that the digit clusters **3** and **8** change places in the top left scatter plot of Fig. 8(c) seems to be the only irregularity worth mentioning. Other new aspects, e.g. the formation of the large **2-3-8-6-5-0** meta-cluster, are consistent between the 4 scatter plots. Characterized by similarly good DSC scores as the DBN, the supervised UMAP learns 2D feature representations where the digit clusters **0** to **9** are considerably more separable and more compact. However, the used UMAP hyperparameter setup (Appendix D) yields no consistent relative positioning of the digit clusters. Moreover, it does not encourage a consistent meta-cluster structure. This visual impression is more pronounced in the corresponding dendrograms (Fig. 9(b)).

Since the plain DAE and SCVIS were learned unsupervisedly, they both yield significantly less separable and less compact digit clusters. Observe however that the relative positioning of the centroids of the digit clusters is quite consistent in the case of the plain DAE. This can be seen as a first experimental confirmation that ReNDA's DAE-like topol-

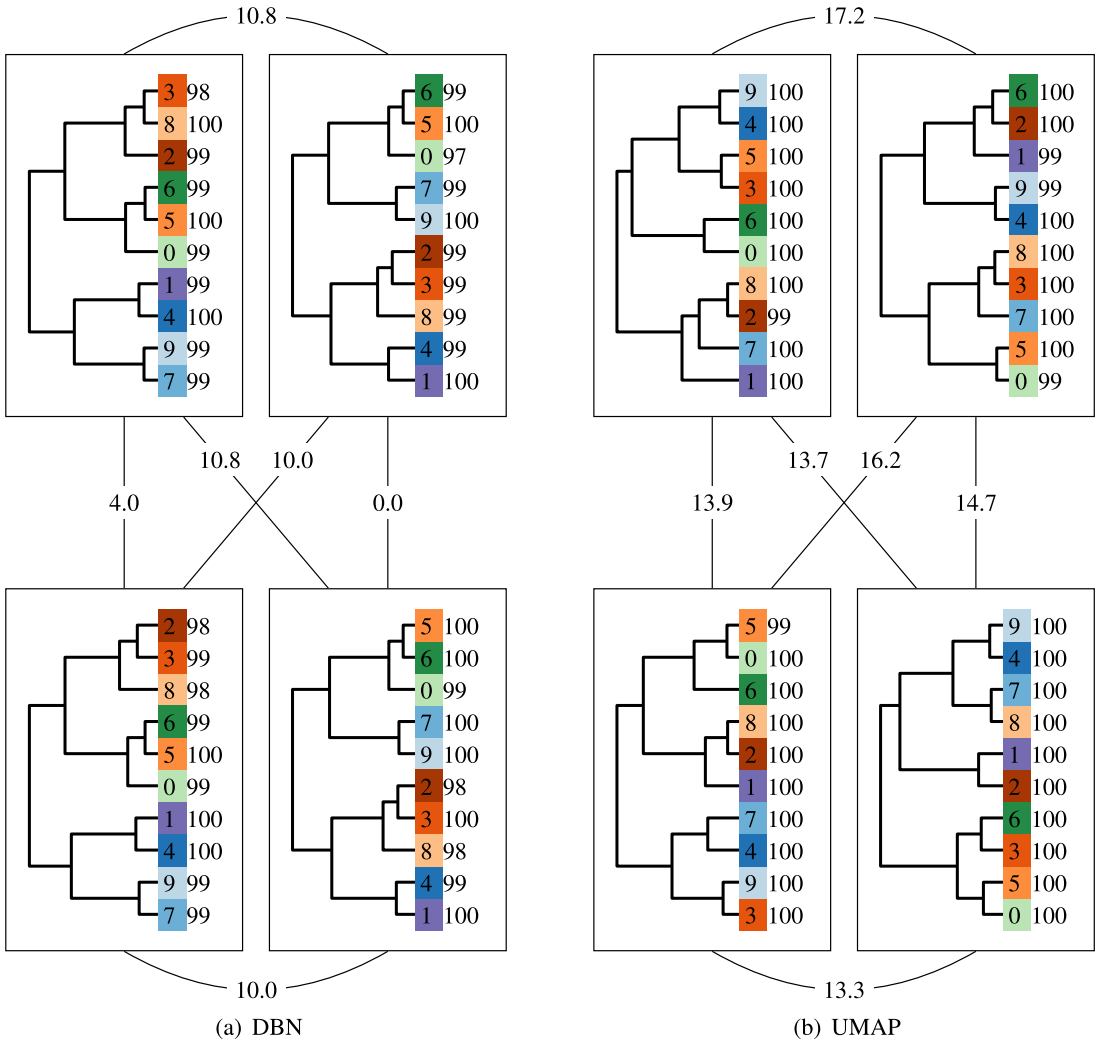


Fig. 9. Dendrograms (Section 4.1) corresponding to the scatter plots in Fig. 8(c) and (d), respectively. The leaves are colored applying the same single-hue color scheme as in the case of the scatter plots. The value stated right of each dendrogram leaf indicates the reliability (in %) of the label assigned to that leaf. The labeled links connecting the rectangular boxes indicate the pairwise tree dissimilarity measures (TDMs) presented in Section 4.2.

ogy is the reason for its robustness. As pointed out in Section 1.1, SCVIS can be interpreted as a parametric t-SNE [43] that is extended to form a deep variational autoencoder [16]. Although SCVIS is not equipped with a parameter sharing scheme, Ding et al. [10, including its supplementary online material] provides an extensive experimental proof that it is more robust against simulated data fluctuations than the plain parametric t-SNE implementation by Krijthe [17]. We find it very interesting that our plain DAE yields a more consistent relative positioning of the digit cluster centroids. We therefore plan to implement a regularized parametric t-SNE that is set up as ReNDA, i.e. an extended parametric t-SNE with a DAE topology and the parameter sharing relations (12) and (13). The results will be presented in a future publication.

5.2. Unsupervised cluster detection

In Section 3.2.4, we demonstrated how to utilize unsupervised clustering algorithms to assess a supervisedly learned dimensionality reduction. Table 2 is organized as Table 1 and summarizes the results for the DBN and UMAP – the *corresponding ReNDA results* (Table 1) are shown to enable a faster comparison.

To interpret the results in the right way, it is important to note that there exists no generally applicable rule on how to select the right CVI for a given feature representation. For instance, in the case of the DBN, only the Davies-Bouldin validity index is able to reliably confirm the true number of clusters $K^* = 10 = C$. In the case of UMAP, this can be achieved with the Davies-Bouldin and the silhouette validity index. Considering the UMAP scatter plots shown in Fig. 8(d), it is a bit surprising that the Caliński-Harabasz and the gap statistic validity index fail to confirm the true number of clusters. Currently, we believe that this is due to a form of *within-cluster noise* [23] but an adequate explanation requires further research. So far, we see it as a positive result that the 2D feature representations provided by ReNDA allows for an application of all CVIs considered.

5.3. Summary of results

Table 3 states the DSC scores and the TDMs (Section 4.2) of all 6 approaches considered in this paper. As in all previous sections, the averages and standard deviations of the DSC scores are based on the respective 10k validation samples of the 9 runs per approach. The averages and standard deviations of the TDMs, on the other hand, are based on the respective 50k training samples. The hierarchical tree structures obtained through Ward's method will be further used for the design of a hierarchical classifier. Therefore, the validation samples must be saved for a final overall performance assessment.

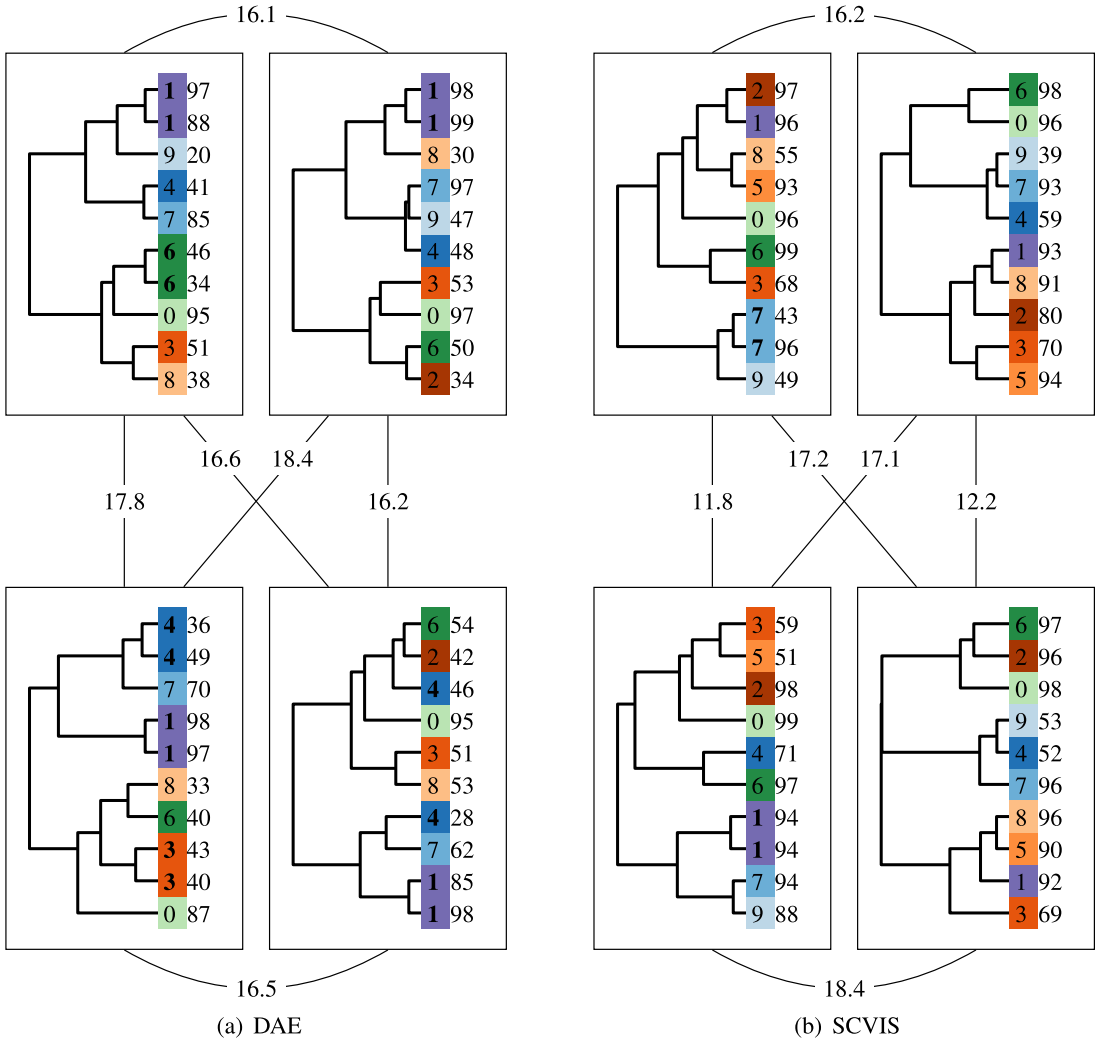


Fig. 10. Dendrograms (Section 4.1) corresponding to the scatter plots in Fig. 8(e) and (f), respectively. The leaves are colored applying the same single-hue color scheme as in the case of the scatter plots. The value stated right of each dendrogram leaf indicates the reliability (in %) of the label assigned to that leaf. The labeled links connecting the rectangular boxes indicate the pairwise tree dissimilarity measures (TDMs) presented in Section 4.2.

Table 2
Cluster evaluation results obtained for the 9 DBN and 9 UMAP feature representations; 4 of these feature representations are shown in Fig. 8(c) and (d), respectively. The first value in each table cell states the number of false predictions where a false prediction is indicated by $K^* \neq 10 = C$. The best prediction result is 0. Since we carried out 9 DBN and 9 UMAP runs, the worst prediction result is 9. The value in parentheses states the average absolute deviation of the false predictions. It is not defined if the number of false predictions is 0. The *ReNDA* cluster evaluation results from Table 1 are included to enable a faster comparison.

Clustering validity index	Dimensionality reduction approach	Clustering algorithms		
		K-means	*K*-comp. GMM	Ward's method
Caliński-Harabasz	DBN	9 (5.9)	9 (4.3)	9 (8.3)
	UMAP	9 (9.4)	9 (7.1)	9 (9.0)
	ReNDA	*0*	*0*	*1(1.0)*
Davies-Bouldin	DBN	0	1 (1.0)	0
	UMAP	0	1 (1.0)	0
	ReNDA	*0*	*0*	*2 (1.0)*
Gap Statistic	DBN	9 (6.1)	2 (1.0)	9 (7.9)
	UMAP	9 (9.0)	9 (8.9)	9 (8.9)
	ReNDA	*0*	*0*	*1 (1.0)*
Silhouette	DBN	9 (4.7)	9 (4.9)	9 (6.6)
	UMAP	0	1 (1.0)	0
	ReNDA	*1 (1.0)*	*5 (1.6)*	*3 (1.0)*

Table 3
DSC scores and TDMs, and approximate computation time and memory usage of all approaches considered in this paper. All stated averages and standard deviations are based on the 9 runs per approach. The computation times are either stated as one value indicating the total computation time, or as follows: total = pretraining + fine-tuning. Note that we did not observe significant differences in memory usage between pretraining and fine-tuning. The stated value is the maximum observed memory usage.

	DSC score see [34]	TDM see Section 4.2	Computation time in hours	Memory usage in GB
ReNDA	94.94 ± 0.39	4.93 ± 3.03	32 = 2 + 30	1.3
GerDA	91.47 ± 3.03	11.11 ± 2.78	14 = 1 + 13	0.7
DBN	96.62 ± 0.22	6.48 ± 4.33	77 = 53 + 24	n/a
UMAP	96.50 ± 0.24	14.16 ± 2.48	< 0.1	1.2
DAE	55.78 ± 1.74	16.55 ± 1.72	17 = 1 + 16	1.1
SCVIS	83.88 ± 5.85	15.89 ± 2.75	52	1.1

As can be seen, ReNDA yields 2D feature representations with relatively good DSC scores that are also reliably reproducibly. Moreover, the obtained TDMs are outstanding as compared to those achieved by the other approaches. Considering the use-case of designing a hierarchical classifier, this means that the final classifier design does not vary much with small fluctuations in the data. Thus, there is a better chance that researchers in this field who work with comparable data sets can exchange their experience in a meaningful way. In the context of this use case, all other approaches are less well suited. The dendrograms in Figs. 6 (Section 4.1), 9 and 10 confirm this result.

Of course, in practice, the robustness of a dimensionality reduction approach is not the only aspect to consider. Column 3 of Table 3 shows that the supervised learning of a UMAP feature mapping based on 50k MNIST digits takes less than 6 minutes, which is about 320 times faster than ReNDA. This makes UMAP the better choice for straightforward data visualization. Note that our ReNDA/GerDA implementation is not well-optimized with regard to computation time and memory usage. For this, we largely relied on MATLAB®'s core routines, and focused on a high transparency of the learning processes instead. Despite of this, ReNDA, GerDA and also the corresponding plain DAE appear to have a better runtime performance than the DBN and SCVIS. It is however important to note that such performance differences are seldom entirely approach-related. Other design choices such as the programming language and style, the use of APIs, etc. must always be considered equally responsible.

In summary, columns 3 and 4 of Table 3 show that the approximate computation times vary much between the 6 different approaches and that all approaches require a moderate amount of memory that lies around 1 GB. For the sake of completeness, we present additional details on all experimental setups in Appendix D. This appendix section also covers relevant information about the workstation that was used for all experiments.

6. Conclusion and outlook

In this paper, we assessed the robustness of a deep neural network (DNN) approach to dimensionality reduction for data visualization. The DNN used for this work is a combination of the **Ge***ner**alized* **D***iscriminant* **A***nalysis* (GerDA) and a **D***eep* **A***uto***E***ncoder* (DAE) as suggested by Stuhlsatz et al. [38] and Hinton and Salakhutdinov [15], respectively. The combined DNN is called ReNDA (short for **Re***gularized* **N***onlinear* **D***iscriminant* **A***nalysis*). The experiments presented in this paper show that ReNDA can reliably produce and reproduce feature representations for visualization. To further support this conclusion, we looked at four comparative approaches: the DBN [41] already considered in [2], UMAP [24], a plain DAE and SCVIS [10].

We presented and discussed various visualizations of the results obtained from two extensive experiments. The forms of visualization ranged from straightforward histograms and scatter plots of the feature representations to dendrograms showing hierarchical cluster structures derived from these feature representations. As part of the hierarchical cluster analysis we carried out in this context, we proposed a measure that expresses the structural dissimilarity between two feature representations.

The hierarchical cluster analysis presented in Sections 3.2.4 and 4 motivates for future research: One is the derivation of a hierarchical classifier, a promising approach in the field of classification [18,33]. Another is the development of visualizations that can be used to monitor DNN learning processes with respect to their robustness (Section 4.2). Further interesting future tasks are comparisons of the DAE-based regularization and other regularization approaches (e.g. the recently proposed *dropout regularization* [36]), and also experiments testing the semi-supervised learning capabilities of ReNDA. The latter makes sense since GerDA is a supervised and a DAE is an unsupervised machine learning approach.

In conclusion, ReNDA has proven to be a suitable approach to robust dimensionality reduction for data visualization. The strategies applied to measure this robustness show that hierarchical classification is a promising future application. Finally, we would like to point out that the hierarchical cluster analysis presented in this paper can also be applied to assess the robustness of other approaches to dimensionality reduction. We consider the comparison of ReNDA, GerDA and the four comparative approaches (Section 5) to be a successful first test of this approach.

Declaration of Competing Interest

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

Appendix A. On the normalized GerDA criterion

In Section 2.4.1, we stated that the GerDA criterion (21) is normalized, i.e. that $J_{\text{GerDA}} \in (0|1)$. As this is not straightforward to see, we give a proof in this appendix section.

Let λ_k for $k \in \{1, \dots, d_{\mathbf{Z}}\}$ denote the eigenvalues of $(\mathbf{S}_T^{\delta})^{-1}\mathbf{S}_B^{\delta}$. Then $\operatorname{trace}((\mathbf{S}_T^{\delta})^{-1}\mathbf{S}_B^{\delta}) = \sum_{k=1}^{n} \lambda_k$ and we need to show that $0 < \lambda_k < 1$ for all k.

Therefore, let μ_k for $k \in \{1, \dots, d_{\mathbf{Z}}\}$ denote the eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B^{\delta}$ and let $\boldsymbol{x}_k \in \mathbb{R}^{d_{\mathbf{Z}}}$ denote an eigenvector to the eigenvalue μ_k. Then

$$\begin{aligned} & \mathbf{S}_W^{-1}\mathbf{S}_B^{\delta}\,\boldsymbol{x}_k = \mu_k \boldsymbol{x}_k \\ \Leftrightarrow\ & \boldsymbol{x}_k^{tr}\mathbf{S}_B^{\delta}\boldsymbol{x}_k = \mu_k \cdot \boldsymbol{x}_k^{tr}\mathbf{S}_W\boldsymbol{x}_k\,. \end{aligned} \tag{25}$$

Since both $\mathbf{S}_B^{\delta}$ and $\mathbf{S}_W$ (see (19) and (20)) are positive definite, we have that $\mu_k > 0$ for all k. We use (18) to rewrite the characteristic eigenvalue equation associated with (25) as follows:

$$\begin{aligned} & \left(\mathbf{S}_W^{-1}\mathbf{S}_B^{\delta} - \mu_k \mathbf{I}_{d_{\mathbf{Z}}}\right)\boldsymbol{x}_k = \mathbf{0} \\ \Leftrightarrow\ & \left(\mathbf{S}_W^{-1}(\mathbf{S}_T^{\delta} - \mathbf{S}_W) - \mu_k \mathbf{I}_{d_{\mathbf{Z}}}\right)\boldsymbol{x}_k = \mathbf{0} \\ \Leftrightarrow\ & \Big(\mathbf{S}_W^{-1}\mathbf{S}_T^{\delta} - \underbrace{(\mu_k + 1)}_{=:\,\kappa_k}\mathbf{I}_{d_{\mathbf{Z}}}\Big)\boldsymbol{x}_k = \mathbf{0} \end{aligned} \tag{26}$$

Clearly, κ_k for $k \in \{1, \dots, d_{\mathbf{Z}}\}$ denote the eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_T^{\delta}$ and it is $\kappa_k > 1$ for all k. The eigenvalues of $(\mathbf{S}_W^{-1}\mathbf{S}_T^{\delta})^{-1}$ are simply $0 < \kappa_k^{-1} < 1$ for all k. We finally use that $(\mathbf{S}_W^{-1}\mathbf{S}_T^{\delta})^{-1} = \mathbf{I}_{d_{\mathbf{Z}}} - (\mathbf{S}_T^{\delta})^{-1}\mathbf{S}_B^{\delta}$ which is equivalent to (18). With these last two statements we further convert (26) to

$$\begin{aligned} &\Leftrightarrow \ \left((\mathbf{S}_W^{-1}\mathbf{S}_T^{\delta})^{-1} - \kappa_k^{-1}\mathbf{I}_{d_{\mathbf{Z}}}\right)\boldsymbol{x}_k = \mathbf{0} \\ &\Leftrightarrow \ \Big((\mathbf{S}_T^{\delta})^{-1}\mathbf{S}_B^{\delta} - \underbrace{\left(1 - \kappa_k^{-1}\right)}_{=\lambda_k}\mathbf{I}_{d_{\mathbf{Z}}}\Big)\boldsymbol{x}_k = \mathbf{0}. \end{aligned} \tag{27}$$

Here, it is easy to see that $0 < \lambda_k < 1$ for all k, which is what we intended to show.

Appendix B. Partial derivatives of J_{GerDA}

Let $\ell \in \{1, \dots, L\}$. The partial derivatives of J_{GerDA} (see (21)) are given by

$$\frac{\partial J_{\text{GerDA}}}{\partial \mathbf{W}^{\ell}} = \boldsymbol{\Delta}^{\ell}(\mathbf{X}^{\ell-1})^{tr} \tag{28}$$

$$\frac{\partial J_{\text{GerDA}}}{\partial \boldsymbol{b}^{\ell}} = \boldsymbol{\Delta}^{\ell} \cdot \mathbf{1}_N \tag{29}$$

with $\mathbf{1}_N := (1, 1, \dots, 1)^{tr} \in \mathbb{R}^N$ and

$$\boldsymbol{\Delta}^{\ell} := f'_{\ell}(\mathbf{A}^{\ell}) \odot \begin{cases} \partial J_{\text{GerDA}}/\partial \mathbf{Z} & \ell = L \\ (\mathbf{W}^{\ell+1})^{tr}\boldsymbol{\Delta}^{\ell+1} & 1 \le \ell < L \end{cases}. \tag{30}$$

It is

$$\frac{\partial J_{\text{GerDA}}}{\partial \mathbf{Z}} = -\frac{1}{d_{\mathbf{Z}}} \cdot \frac{\partial Q_z^{\delta}}{\partial \mathbf{Z}}. \tag{31}$$

A computable expression for $\partial Q_z^{\delta}/\partial \mathbf{Z}$ along with its derivation is given in [38].

Appendix C. Partial derivatives of J_{DAE}

In the case of J_{DAE} (see (24)), the partial derivatives with respect to the weight matrices are given by

$$\frac{\partial J_{\text{DAE}}}{\partial \mathbf{W}^{\ell}} = \boldsymbol{\Lambda}^{\ell}(\mathbf{X}^{\ell-1})^{tr} + \left(\boldsymbol{\Lambda}^{2L-\ell+1}(\mathbf{X}^{2L-\ell})^{tr}\right)^{tr} \tag{32}$$

for $\ell \in \{1, \dots L\}$. The partial derivatives with respect to the bias vectors are given by

$$\frac{\partial J_{\text{DAE}}}{\partial \mathbf{b}^{\ell}} = \begin{cases} \boldsymbol{\Lambda}^{2L} \cdot \mathbf{1}_N & \ell = 2L \\ (\boldsymbol{\Lambda}^{\ell} + \boldsymbol{\Lambda}^{2L-\ell}) \cdot \mathbf{1}_N & 1 \le \ell < L \end{cases}. \tag{33}$$

with $\mathbf{1}_N := (1, \dots, 1)^{tr} \in \mathbb{R}^N$. For $\ell \in \{1, \dots 2L\}$ the matrices $\boldsymbol{\Lambda}^{\ell}$ are defined by

$$\boldsymbol{\Lambda}^{\ell} := f'_{\ell}(\mathbf{A}^{\ell}) \odot \begin{cases} \partial J_{\text{DAE}}/\partial \hat{\mathbf{X}} & \ell = 2L \\ (\mathbf{W}^{\ell+1})^{tr}\boldsymbol{\Lambda}^{\ell+1} & 1 \le \ell < 2L \end{cases}. \tag{34}$$

with

$$\frac{\partial J_{\text{DAE}}}{\partial \hat{\mathbf{X}}} = \frac{2\left(\hat{\mathbf{X}} - \mathbf{X}\right)}{d_{\mathbf{X}} \cdot N \cdot \left(1 + \text{MSE}/d_{\mathbf{X}}\right)}, \tag{35}$$

which is straightforward to prove.

Appendix D. Experimental setups in detail

Table 4 states further details on our ReNDA and GerDA experiments (Section 3). The setup is the original setup from [2]. The comparative results presented in Section 5 are based on the following experimental setups.

In the case of the *DBN*, we considered the topology 784-1200-1200-*2*-10. As pointed out in [2], the intermediate dimension *2* was added to the original topology [41] to obtain the desired 2D feature representations. Other than that, we adopted the example code [40]. It has not been changed since we ran our DBN experiments for [2].

Table 4
Overview of our experimental setups for the ReNDA and GerDA runs. The batchsizes and the number of epochs were chosen as in the experiments for [38]. From [2].

ReNDA and GerDA		
Setup / Property	Galaxy	MNIST
Data dimensionality	2	784
Feature dimensionality	1	2
Number of classes	3	10
Number of samples		
for DNN learning	1440	50,000
for validation	5118	10,000
in total	65,580	60,000
Pretraining		
Batchsize	144	2000
Number of Epochs	10	50
Fine-tuning		
Batchsize	288	5000
Number of Epochs	1000	200

In the case of *UMAP*, we used the `random_state` parameter of the *UMAP software* [25] to initialize the 3 different random number streams. For all other parameters, we simply used the default values. Our current UMAP version is 0.3.9.

As mentioned in Section 5, the *plain DAE* considered is the one obtained for $\lambda = 1$ in (2). It was therefore set up as ReNDA and GerDA (including the details stated in Table 4). Recall that our DAE criterion is given by (24). See (35) for its derivative.

Because the default topology of *SCVIS* is too small, we set its encoder and decoder topology to those of ReNDA, i.e. 784-1500-375-750-2 and 2-750-375-1500-784, respectively. To further increase the comparability of the SCVIS and the ReNDA/GerDA results, the SCVIS mappings were learned over 200 epochs using a batchsize of 5000. Note that SCVIS does not involve an RBM pretraining but the uniformly distributed random initialization proposed in [14]. We used the `seed` parameter in the YAML configuration file to initialize the 3 different random number streams. Our current SCVIS version is 0.1.0.

For all experiments, we used a workstation with *2 x Intel Xeon E5-2687W v4 @ 3.00GHz, 12 cores, 256 GB RAM* and an *NVMe SSD*. In the case of ReNDA, GerDA, the DBN and the plain DAE, we used a *MATLAB® Parallel Server* to ensure that each of the 9 runs per approach was performed on only 1 of the 24 cores. Currently UMAP only supports single-core execution, so each of the 9 runs was performed on only 1 of the 24 cores by default. In the case of SCVIS, a multi-core execution seems likely since it is implemented using TensorFlow – in order to avoid the introduction of unforeseeable side effects, we ran SCVIS without limiting its computations to a single core.

With 256 GB RAM, we were sure not to experience any RAM-related bottleneck effects in the individual runs per approach. Furthermore, the I/O operations per run only make up a small proportion of the approximate computation times stated in Table 3, which is due to the NVMe SSD as a storage for logging information, saving intermediate models, etc.

References

[1] M. Becker, A Deep Neural Network for Robust Feature Extraction, Hochschule Düsseldorf–University of Applied Sciences, 2017 Master's thesis, doi:10.13140/RG.2.2.35523.14887.
[2] M. Becker, J. Lippel, A. Stuhlsatz, Regularized nonlinear discriminant analysis - an approach to robust dimensionality reduction for data visualization, in: Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: IVAPP, (VISIGRAPP 2017), SciTePress, 2017, pp. 116–127, doi:10.5220/0006167501160127.
[3] A.N. Bhagoji, D. Cullina, C. Sitawarin, P. Mittal, Enhancing robustness of machine learning systems via data transformations, in: 52nd Annual Conference on Information Sciences and Systems (CISS), 2018, pp. 1–5, doi:10.1109/CISS.2018.8362326.
[4] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat. Simul. Comput. 3 (1) (1974) 1–27.

[5] G. Chao, Y. Luo, W. Ding, Recent advances in supervised dimension reduction: asurvey, Mach. Learn. Knowl. Extr. 1 (1) (2019) 341–358, doi:10.3390/make1010020.

[6] S. Chenouri, J. Liang, C.G. Small, Robust dimension reduction, Wiley Interdiscip. Rev. 7 (1) (2015) 63–69, doi:10.1002/wics.1331.

[7] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 1, 2005, pp. 539–546, doi:10.1109/CVPR.2005.202.

[8] D.L. Davies, D.W. Bouldin, A cluster separation measure., IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 224–227.

[9] K. Delac, M. Grgic, S. Grgic, Independent comparative study of PCA, ICA, and LDA on the FERET data set, Int. J. Imaging Syst. Technol. 15 (5) (2006) 252–260, doi:10.1002/ima.20059.

[10] J. Ding, A. Condon, S.P. Shah, Interpretable dimensionality reduction of single cell transcriptome data with deep generative models, Nat. Commun. 9 (1) (2018), doi:10.1038/s41467-018-04368-5.

[11] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11 (2010) 625–660.

[12] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (1936) 179–188.

[13] A. Gisbrecht, B. Hammer, Data visualization by nonlinear dimensionality reduction, Wiley Interdiscip. Rev. 5 (2) (2015) 51–73, doi:10.1002/widm.1147.

[14] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics, 2010.

[15] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507.

[16] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings, 2014.

[17] J.H. Krijthe, Rtsne: T-Distributed Stochastic Neighbor Embedding using Barnes-Hut Implementation, 2015. R package version 0.15.

[18] S. Kumar, J. Ghosh, M.M. Crawford, Hierarchical fusion of multiple classifiers for hyperspectral data analysis, Pattern Anal. Appl. 5 (2) (2002) 210–220, doi:10.1007/s100440200019.

[19] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: Proc. of the IEEE, 1998, pp. 1–46.

[20] J.A. Lee, M. Verleysen, Unsupervised dimensionality reduction: overview and recent advances, in: The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–8, doi:10.1109/IJCNN.2010.5596721.

[21] S. Liu, X. Wang, M. Liu, J. Zhu, Towards better analysis of machine learning models: avisual analytics perspective, Vis. Inform. 1 (1) (2017) 48–56, doi:10.1016/j.visinf.2017.01.006.

[22] Y. Ma, C. Shang, F. Yang, D. Huang, Latent subspace clustering based on deep neural networks, in: 6th International Symposium on Advanced Control of Industrial Processes, 2014, pp. 502–507.

[23] L. McInnes, [Question] Clustering on UMAP output, 2017, (https://github.com/lmcinnes/umap/issues/25}issuecomment-346659130). Last updated: Nov 2017, retrieved: Jul 2019, last checked: Jul 2019.

[24] L. McInnes, J. Healy, UMAP: Uniform manifold approximation and projection for dimension reduction, CoRR abs/1802.03426(2018).

[25] L. McInnes, J. Healy, N. Saul, L. Grossberger, Umap: uniform manifold approximation and projection, J. Open Source Softw. 3 (29) (2018) 861.

[26] E. Plaut, From principal subspaces to principal components with linear autoencoders, arXiv:abs/1804.10253(2018).

[27] R. Reed, Pruning algorithms-a survey, IEEE Trans. Neural Netw. 4 (5) (1993) 740–747, doi:10.1109/72.248452.

[28] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: E.P. Xing, T. Jebara (Eds.), Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research, 32, PMLR, Bejing, China, 2014, pp. 1278–1286.

[29] P. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1) (1987) 53–65 https://doi.org/10.1016/0377-0427(87)90125-7.

[30] R. Salakhutdinov, G. Hinton, Learning a nonlinear embedding by preserving class neighbourhood structure, in: Proc. of the Intern. Conference on Artificial Intelligence and Statistics, 11, 2007.

[31] A. Saunders, D. Ashlock, S.K. Houghten, Hierarchical clustering and tree stability, in: 2018 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2018, Saint Louis, MO, USA, May 30,–June 2, 2018, 2018, pp. 1–8, doi:10.1109/CIBCB.2018.8404978.

[32] M. Sedlmair, M. Aupetit, Data-driven evaluation of visual quality measures, in: Computer Graphics Forum, 34, Wiley Online Library, 2015, pp. 201–210.

[33] C.N. Silla, A.A. Freitas, A survey of hierarchical classification across different application domains, Data Min. Knowl. Discov. 22 (1–2) (2011) 31–72, doi:10.1007/s10618-010-0175-9.

[34] M. Sips, B. Neubert, J.P. Lewis, P. Hanrahan, Selecting good views of high-dimensional data using class consistency, Comput. Graph. Forum 28 (3) (2009) 831–838.

[35] C.O.S. Sorzano, J. Vargas, A.D. Pascual-Montano, A survey of dimensionality reduction techniques, arXiv:abs/1403.2877(2014).

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.

[37] A. Stuhlsatz, J. Lippel, T. Zielke, Discriminative feature extraction with deep neural networks, in: Proc. of the 2010 Intern. Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 2010.

[38] A. Stuhlsatz, J. Lippel, T. Zielke, Feature extraction with deep neural networks by a generalized discriminant analysis, IEEE Trans. Neural Netw. 666 (2012) 1–666.

[39] A. Stuhlsatz, C. Meyer, F. Eyben, T. Zielke, G. Meier, B. Schuller, Deep neural networks for acoustic emotion recognition: raising the benchmarks, in: Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 2011.

[40] M. Tanaka, Deep neural network, MATLAB Central File Exchange (# 42853)(2016). Last updated: Aug 2016, retrieved: Dec 2016, last checked: Jul 2019.

[41] M. Tanaka, M. Okutomi, A novel inference of a restricted Boltzmann machine, in: 22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24–28, 2014, 2014, pp. 1526–1531, doi:10.1109/ICPR.2014.271.

[42] R. Tibshirani, W. Guenther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, J. R. Stat. Soc. Ser. B: Stat. Meth. 63 (2) (2001) 411–423, doi:10.1111/1467-9868.00293.

[43] L. van der Maaten, Learning a parametric embedding by preserving local structure, in: Proc. 12th Intern. Conf. on Artificial Intelligence and Statistics (AISTATS), in: JMLR: W&CP 5, 5, 2009.

[44] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.

[45] L. Van Der Maaten, E. Postma, J. Van den Herik, Dimensionality reduction: a comparative review, J. Mach. Learn. Res. 10 (2009) 66–71.

[46] F. Wang, C. Rudin, Dimension reduction for robust covariate shift correction, CoRR abs/1711.10938(2017).

[47] Y. Wang, K. Feng, X. Chu, J. Zhang, C. Fu, M. Sedlmair, X. Yu, B. Chen, A perception-driven approach to supervised dimensionality reduction for visualization, IEEE Trans. Vis. Comput. Graph. 24 (5) (2018) 1828–1840, doi:10.1109/TVCG.2017.2701829.

[48] J.H. Ward, Hierarchical grouping to optimize an objective function, J. Am. Stat. Assoc. 58 (301) (1963) 236–244, doi:10.1080/01621459.1963.10500845.

[49] W. Yu, G. Zeng, P. Luo, F. Zhuang, Q. He, Z. Shi, Embedding with autoencoder regularization, in: Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III, in: ECMLPKDD'13, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 208–223, doi:10.1007/978-3-642-40994-3_14.

[50] D. Zabihzadeh, M.H. Moattar, Manifold learning based speaker dependent dimension reduction for robust text independent speaker verification, Int. J. Speech Technol. 17 (2014) 271–280.

[51] Y. Zhai, Y.S. Ong, I.W. Tsang, The emerging "big dimensionality", IEEE Comput. Intell. Mag. 9 (3) (2014) 14–26, doi:10.1109/MCI.2014.2326099.

[52] N. Zhao, W. Mio, X. Liu, A hybrid PCA-LDA model for dimension reduction, in: The 2011 International Joint Conference on Neural Networks, 2011, pp. 2184–2190, doi:10.1109/IJCNN.2011.6033499.

[53] L. Zheng, Y. Yang, Q. Tian, Sift meets CNN: a decade survey of instance retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 40 (5) (2018) 1224–1244, doi:10.1109/TPAMI.2017.2709749.