Neural Networks 105 (2018) 112–131

Contents lists available at ScienceDirect

Neural Networks

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

Denoising Autoencoder Self-Organizing Map (DASOM)

Christos Ferles [a,b], Yannis Papanikolaou [c], Kevin J. Naidoo [a,b,d,*]

[a] Scientific Computing Research Unit, Faculty of Science, University of Cape Town, Rondebosch, 7701, South Africa
[b] Department of Chemistry, Faculty of Science, University of Cape Town, Rondebosch, 7701, South Africa
[c] Department of Informatics, Aristotle University of Thessaloniki, 54124, Thessaloniki, Greece
[d] Institute for Infections Disease and Molecular Medicine, Faculty of Heath Science, University of Cape Town, Rondebosch, 7701, South Africa

ARTICLE INFO

Article history:
Received 4 September 2017
Received in revised form 18 April 2018
Accepted 25 April 2018
Available online 7 May 2018

Keywords:
Unsupervised learning
Denoising autoencoder
Self-organizing map
Clustering
Visualization

ABSTRACT

In this report, we address the question of combining nonlinearities of neurons into networks for modeling increasingly varying and progressively more complex functions. A fundamental approach is the use of higher-level representations devised by restricted Boltzmann machines and (denoising) autoencoders. We present the Denoising Autoencoder Self-Organizing Map (DASOM) that integrates the latter into a hierarchically organized hybrid model where the front-end component is a grid of topologically ordered neurons. The approach is to interpose a layer of hidden representations between the input space and the neural lattice of the self-organizing map. In so doing the parameters are adjusted by the proposed unsupervised learning algorithm. The model therefore maintains the clustering properties of its predecessor, whereas by extending and enhancing its visualization capacity enables an inclusion and an analysis of the intermediate representation space. A comprehensive series of experiments comprising optical recognition of text and images, and cancer type clustering and categorization is used to demonstrate DASOM's efficiency, performance and projection capabilities.

1. Introduction

The structural and functional nature of biological neural systems provides a developmental template for artificial neural network algorithms. Structurally networks of topographically ordered neurons mimic nerve nets of the visual cortex (Von der Malsburg, 1973) and deep models (Lee, Ekanadham, & Ng, 2008) capture and store higher representations extracted from inherent regularities and structure in data (Ito & Komatsu, 2004). Functionally layered hierarchical architectures imitate sensory signal propagation and preprocessing/processing in regions of the brain (Lee & Mumford, 2003; Lee, Mumford, Romero, & Lamme, 1998). Such findings and advances in neuroscience have fueled numerous aspects of machine learning research, and have paved the way for designing distributed information representation-processing models based on complex layered architectures.

The extensively studied and widely applied clustering paradigm (Jain, 2010; Jain, Murty, & Flynn, 1999; Xu & Wunsch, 2005) forms a substantial part of the unsupervised learning backbone. Cluster analysis lends itself to varying approaches where principally the theme of grouping (separating) data elements based on their similarity/closeness (difference/distance) is common to much of these unsupervised classification algorithms. In a clustering method, partitioning the data into subsets subject to the criteria that the intra-cluster's internal homogeneity is greater than the inter-cluster's external heterogeneity, is the norm. Despite these fundamental commonalities there remains a diverse range of clustering approaches, for addressing generic problems that are enriched by a comprehensive array of problem-specific clustering algorithms.

The Self-Organizing Map (SOM) (Kohonen, 2001, 2013) is a specific type of unsupervised learning algorithm that represents the distribution-characteristics of input samples on planes of topographically ordered nodes and in doing so achieves clustering through dimensionality reduction. The intrinsic nonlinear mapping capabilities of the SOM on its low-dimensional neural surface distinguish it from most of the techniques that fall within the wider class of clustering algorithms. This advantage over more typical clustering approaches has led to the SOM methodology being as a means of visualizing nonlinear relations of data, topology-based cluster analysis, vector quantization and projection of multi-dimensional data. Because of the versatility of the SOM the scope of applications to which it has been applied is vast ranging from pattern recognition, image-text processing, mining, genomics, medical diagnostics, robotics and economics.

A difficulty that arises during pattern recognition from unlabeled data is the presence of large numbers of elements that

* Corresponding author at: Department of Chemistry, Faculty of Science, University of Cape Town, Rondebosch, 7701, South Africa.
E-mail addresses: christos.ferles@gmail.com, Christos.Ferles@uct.ac.za (C. Ferles), ypapanik@csd.auth.gr (Y. Papanikolaou), kevin.naidoo@uct.ac.za (K.J. Naidoo).

https://doi.org/10.1016/j.neunet.2018.04.016
0893-6080/

make no contribution, or contribute marginally, to the data representation (features) of the dataset. The performance of machine learning methods therefore depends on the choice of the data features on which they are applied. The introduction of the autoencoder has been a major innovation in unsupervised learning where through backpropagation key features in the data are discovered. The learning structure is borrowed from the neurological process underpinning global learning and intelligent behavior in hominids. This is where biochemical events respond to input data and induce synaptic changes that are, through self-organization, coordinated to learn from the input data. Biologically this defines what we consider to be perception and ingenuity. Autoencoder algorithms mimic this biology by transforming an input vector into a hidden representation (the encoder) using a deterministic function that is parameterized by a weighting matrix and a bias vector. Through a placement of constraints on the network, such as limiting the number of hidden units or deactivating certain of its parts, structure in the data that is of interest can be discovered. The hidden representation is then mapped back to a reconstruction vector that has the same deterministic functional form as before and so comprises a counterpart to its own weighting matrix and bias vector (the decoder). The parameters are optimized by minimizing the (average) reconstruction error.

During the past decade, deep learning (Bengio, 2009; Hinton, Osindero, & Teh, 2006; Schmidhuber, 2015) has been at the forefront of the development of methods that shift the focus from feature engineering by experts to meaningful representation discovery by algorithms. The discovered distributed layered representations, which build upon lower-level invariant partial features, reveal higher-level abstract concepts and aspects of the data. The induced responses, from discovered correlations within data, depend on the connectivity of the neurons. In an algorithm this is implemented as multiple sequential causative compute events wherein each event transforms (often in a nonlinear way) the aggregate response of the network (Schmidhuber, 2015). Deep learning within this context refers to the accurate adjustment of parameters (weights and bias vectors) across such events.

Denoising Autoencoders (DAs) (Bengio, 2009; Vincent, Larochelle, Bengio, & Manzagol, 2008; Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010) introduce an alternative to deep belief networks by incorporating stochastic corruption during the training process. The premise of introducing the denoising functions is that a good representation is impervious to disruptions of and perturbations to the input data. At the same time these types of representations capture the characteristic aspects and correlations that underpin the distribution of the input data. A procedure of canceling out the artificial corruption leads to devised representations that are closer to the stable structures and invariant features present in the input data space.

Supervised learning tasks, i.e. regression and classification, are the primary objectives of deep learning methodologies. However, a modular approach where unsupervised pre-training replaces the usual initialization schemes during the development of (semi)supervised deep learning algorithms has been shown to be effective and is now the standard approach in many developments (Bengio, Lamblin, Popovici, & Larochelle, 2007; Erhan et al., 2010; Erhan, Manzagol, Bengio, Bengio, & Vincent, 2009; Hinton et al., 2006). In the deep learning paradigm, the unsupervised process contributes to the tuning of each layer. This intervention at each layer adds up to a cascade yielding an improved abstraction of higher-level distributed representations as progress is made bottom-up along the layered hierarchy. In this implementation, unsupervised learning effectively reduces the redundancy by progressively gathering and consolidating the information existent in the original data.

While drawing inspiration from deep learning here we propose a model that deviates from the above principle of layer by layer contribution, instead we hybridize a pure unsupervised learning algorithm (viz. the SOM) with the unsupervised learning module of a deep learning architecture (viz. the DA). The Denoising Autoencoder Self-Organizing Map (DASOM) structurally combines, architecturally stacks and algorithmically fuses its two unsupervised learning components. In the following section we present the basic framework and characteristics of the DASOM model that:

1. Captures and projects the relations between higher representations extracted from input data, and simultaneously maps these onto two-dimensional manifolds (e.g. planes). Throughout the procedure, data clusters and their principal correlations are visually observed and analyzed.
2. By effectively incorporating the use of component planes the influence and the importance of each nonlinear combination of input features are visualized on the neural map. Building upon a mechanism to gain insight into the produced representations, a feature selection scheme is realized based on the highest contributing attributes and inputs of specified representations.

The remainder of this paper is organized into five sections. Section 2 presents in detail the DASOM both architecturally and algorithmically, and subsequently, analyzes the corresponding learning phases and adaptation procedures. Section 3 contains qualitative and quantitative experimental results, performance evaluations and comparisons with different algorithms. A systematic evaluation of the data clustering, visualizations and low-dimensional projections is given in Section 4. In Section 5 a summary is given and conclusions are drawn. We include an Appendix that details the analytical equations for the DASOM learning algorithm as well as the derivation thereof.

2. Prototype

The DASOM is a hybrid model that combines in a unified architectural and algorithmic scheme the denoising autoencoder feature discovery in the input data space and the SOM clustering functionality in the output space. The proposed training procedure signifies the DASOM's combinatorial nature and the feedforward connectivity manifests the operational integration between the two modules. The objective is to introduce a model that is in position to carry out unsupervised classification and clustering tasks based upon the higher-level representations of input data and doing so differently from the more classical approaches that directly use the feature values of the input samples.

2.1. Learning

The training procedure consists of two complementary and adjacent phases. During the first phase a DA is constructed and run to produce higher representations of the input data. Subsequently, during the second phase the higher representations that are encoded by the DA's hidden layer are applied to the inputs of a SOM. Later in the scheme the SOM uses these distributed representations for parameter adjustment and tuning.

2.1.1. First phase

First, we examine a non-recurrent neural network (Fig. 1) with equal numbers of neurons for the inputs (input layer) and the outputs (output layer). Each layer's neuron number coincides with the dimensionality L of the input sample $\mathbf{x}$. The only exception to this is the dimensionality of the hidden representation layer $\mathbf{y}$. Initially, at the layer following the input layer the training sample $\mathbf{x}$ is stochastically corrupted, thus producing the corrupted example $\tilde{\mathbf{x}}$, which is of the same dimensionality as $\mathbf{x}$.

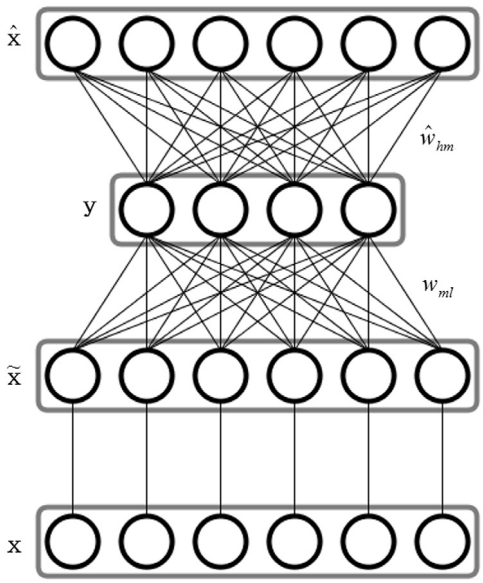

Fig. 1. The DASOM architecture for the first training phase.

encoder

The nonlinear representations are produced at the next layer of M neurons that is constructed with the aid of the weights w_{ml} that denote the contribution of input feature $\widetilde{x}_l$ to the hidden layer neuron m. First, an affine transformation is carried out $z_m = \sum_{l=1}^{L} w_{ml}\widetilde{x}_l + b_m$, where b_m is a bias term. Afterwards, a squashing function f, usually the sigmoid or hyperbolic tangent function, is applied to z_m yielding the activation $y_m = f(z_m)$ of each hidden neuron.

decoder

At the output layer, the sample's reconstructed features are generated by regulating the contribution of representation y_m to the sample feature $\hat{x}_h$ through a set of weights $\hat{w}_{hm}$. These weights and a companion set of biasing parameters $\hat{b}_h$ make up the affine mapping $\hat{z}_h = \sum_{m=1}^{M} \hat{w}_{hm} y_m + \hat{b}_h$ which coupled with a squashing function $\hat{x}_h = f(\hat{z}_h)$ produces the reconstructed sample $\hat{\mathbf{x}}$.

The network's parameter sets can be arranged in a series of matrices, viz. $\mathbf{W}_{M\times L}, \mathbf{b}_{M\times 1}, \hat{\mathbf{W}}_{L\times M}$ and $\hat{\mathbf{b}}_{L\times 1}$ leading to the derivation of the following compact equations:

$$\mathbf{z} = \mathbf{W}\widetilde{\mathbf{x}} + \mathbf{b} \tag{1}$$

$$\mathbf{y} = f(\mathbf{z}) = f(\mathbf{W}\widetilde{\mathbf{x}} + \mathbf{b}) \tag{2}$$

$$\hat{\mathbf{z}} = \hat{\mathbf{W}}\mathbf{y} + \hat{\mathbf{b}} \tag{3}$$

$$\hat{\mathbf{x}} = f(\hat{\mathbf{z}}) = f\left(\hat{\mathbf{W}}\mathbf{y} + \hat{\mathbf{b}}\right). \tag{4}$$

The definition of f is extended to be applied element-wise to vectors. Applying the constraint that ties the weights between $\mathbf{W}$ and $\hat{\mathbf{W}}$ is achieved by setting $\hat{\mathbf{W}} = \mathbf{W}^T$ ($\hat{w}_{hm} = w_{mh}$). The effect of adding this symmetrical assumption is that it reduces the number of free parameters in the model. Within this setup the associated reconstruction error to be optimized (i.e. minimized), with respect to the training example $\mathbf{x}$, is defined as:

$$J\left(\mathbf{W}, \mathbf{b}, \hat{\mathbf{b}}\right) = \frac{1}{2}\left\|\mathbf{x} - \hat{\mathbf{x}}\right\|^2. \tag{5}$$

This is the squared-error function used for real-valued inputs. In the case of binary or $\mathbf{x} \in [0, 1]^L$ inputs the cross-entropy loss function is used instead. Performing gradient descent for updating the network's parameters with the objective of minimizing the reconstruction error is done as:

$$\mathbf{W}^{(next)} = \mathbf{W} - \alpha\nabla_{\mathbf{W}} J\left(\mathbf{W}, \mathbf{b}, \hat{\mathbf{b}}\right) \tag{6}$$

$$\mathbf{b}^{(next)} = \mathbf{b} - \alpha\nabla_{\mathbf{b}} J\left(\mathbf{W}, \mathbf{b}, \hat{\mathbf{b}}\right) \tag{7}$$

$$\hat{\mathbf{b}}^{(next)} = \hat{\mathbf{b}} - \alpha\nabla_{\hat{\mathbf{b}}} J\left(\mathbf{W}, \mathbf{b}, \hat{\mathbf{b}}\right) \tag{8}$$

where α is the learning rate. The partial derivatives of $\mathbf{W}$, $\mathbf{b}$ and $\hat{\mathbf{b}}$ for the gradient descent equations are derived in Appendix A.

2.1.2. Second phase

In a simplistic scheme it may be proposed that the second phase of the learning algorithm could be formed by interleaving the adapted DA network with the SOM (as shown in Fig. 2) by replacing the DA's output layer with the self-organizing neural map. Nonetheless preserving the DA output while directly connecting each neuron of a SOM lattice with the activation of every neuron in the representation layer of the trained DA produces the algorithmic equivalent.

More specifically, consider a low-dimensional array of topologically ordered neurons. Let E be the total number of neurons on the SOM grid. Every neuron e is associated to a codebook vector $\mathbf{u}_e$ consisting of points u_{me} that belong to the distribution domain of each individual hidden layer representation y_m. The codebook vectors (viz. the tunable parameters of the SOM) can be arranged in a matrix $\mathbf{U}_{M\times E}$.

In contrast to the majority of SOM variants, which are based on heuristics approximating Hebbian learning, DASOM has its origins in an appropriately constructed cost function. Consequently, during this second phase, the training algorithm corresponds to a type of gradient descent that seeks for a minimum in each respective modeling task at hand. Starting with the variation proposed by Heskes (1999) the cost function is defined as:

$$K(\mathbf{W}, \mathbf{b}, \mathbf{U}) = \min_d \sum_{e=1}^{E} h_{de} \frac{1}{2}\|\mathbf{y} - \mathbf{u}_e\|^2 \tag{9}$$

where h_{de} is the neighborhood function, a kernel whose width and form control the elasticity of the surface fitted over the lattice neurons. Practically, K introduces a weighted averaging process over the distances between each codebook vector and a given hidden layer representation. The role of the weighting coefficients (neighborhood kernel) is to impose restrictions that are dictated by the grid's topology. The cost function can equally be formulated as:

$$K(\mathbf{W}, \mathbf{b}, \mathbf{U}) = \sum_{c=1}^{E} N(\mathbf{y}, c) \sum_{e=1}^{E} h_{ce} \frac{1}{2}\|\mathbf{y} - \mathbf{u}_e\|^2 \tag{10}$$

where:

$$N(\mathbf{y}, c) = \begin{cases} 1, c = \arg\min_d \sum_{e=1}^{E} h_{de}\|\mathbf{y} - \mathbf{u}_e\|^2 \\ 0, \text{otherwise.} \end{cases} \tag{11}$$

Such cost functions are continuous for finite sets of input samples. As a result, a gradient descent step for minimizing the (generally non-convex) cost function induces the following learning rules:

$$\mathbf{W}^{(next)} = \mathbf{W} - \eta\nabla_{\mathbf{W}} K(\mathbf{W}, \mathbf{b}, \mathbf{U}) \tag{12}$$

$$\mathbf{b}^{(next)} = \mathbf{b} - \eta\nabla_{\mathbf{b}} K(\mathbf{W}, \mathbf{b}, \mathbf{U}) \tag{13}$$

$$\mathbf{U}^{(next)} = \mathbf{U} - \theta\nabla_{\mathbf{U}} K(\mathbf{W}, \mathbf{b}, \mathbf{U}) \tag{14}$$

similarly, η and θ are scalar learning rate factors. During this second phase of the DASOM learning procedure the DA module of

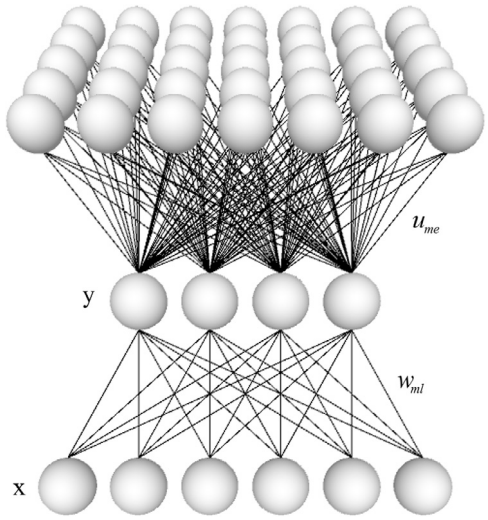

Fig. 2. Paradigm of a DASOM network.

the network has to some extent already been adapted, whereas the SOM unit has not. The learning rate η that regulates the fine-tuning of the weights and the biases should deliver lower values compared with the learning rate θ that initially adjusts the coarse ordering of the mapping. Consequently, during the beginning of the second learning phase the values of η and θ must differ by at least by an order of magnitude to moderate this training imbalance.

Further analysis of the process that combines an already tuned DA with an unadapted SOM highlights the challenge of whether a model should produce clustering results based on a nonlinear manifold that does not change (or changes only slightly), or whether the procedure of cluster modeling should have a larger effect on the curved manifold. In the latter case the curved manifold can potentially be driven towards a direction that is optimal for the neural mapping process but not so for the autoencoding process. The choice is between considering the representations (of the first phase), along with the resulting nonlinear space, ideal for the unsupervised task at hand, and between using the representations as a good starting point to be further refined (during the second phase) with a different optimization target. The proportional balance between the training phases cannot be precisely adjusted because it is not possible to evaluate a representation space, which is being used for an unsupervised task, since the convergence criteria for the overall task itself are both heuristic and approximate.

To address the issues of the two-phase interlaced training a modification is made to the training procedure to give a general network that addresses the competing issues that may arise when combining training phases as described above. The network, termed DASOM-DCT (Denoising Autoencoder Self-Organizing Map with Disjoint-Combined Training), replaces the two-phase DASOM training process with a three-phase learning procedure. Phase one (which is identical to the DASOM first phase) tunes the parameters of the DA and devises the representation space (6)–(8). Phase two adjusts the SOM based on the previous phase's fixed and unaltered representation space (14), it is important to point out that no tuning of the DA module takes place during this phase. This disjoint coarse training of DASOM's components which is carried out independently and separately is followed by the combined in-parallel fine-tuning of all the model parameters. Phase three co-trains the already adjusted components of the DASOM (12)–(14), thus producing a (potentially) more enhanced, fine-tuned and descriptive model. DASOM-DCT avoids extensive distortion of the representation space in the first place but also allows further refinements of the nonlinear manifold towards (curved) directions beneficial for each respective neural mapping problem. At this point it should be reiterated that this change to the basic DASOM construct is not optimal for any given modeling problem. However, it is a generic layout of the DASOM concept that is usable in a wide range of problems and applications. Interestingly this manifestation of DASOM is analogous to the training strategy used in deep learning. Phase one resembles the unsupervised per layer pre-training of a deep architecture. Phase two is similar to the output/classification layer adaptation based on the previous layer's representations. Whereas, phase three identifies with the back-propagation process which runs through the whole network from the output down to the input layer.

Further variants of the DASOM can be derived based on (a) the selection of the above adaptation equations to use during the second phase of the learning algorithm, (b) the replicates of DA networks that are being utilized, and (c) the inclusion/exclusion of the corruption stage. For instance, a naive DASOM would adjust only its codebook vector parameters by simply exploiting training rule (14) and leaving the weight and offset parameters intact. In another instance, each neuron of the lattice could have its own dedicated denoising network and not share a common DA with all the nodes across the grid. Moreover, an AutoEncoder Self-Organizing Map (AESOM) can be easily be implemented by overriding the corruption layer of the DASOM, i.e. by substituting $\hat{\mathbf{x}}$ with $\mathbf{x}$.

2.1.3. Algorithmic and architectural realization

Only the main DASOM learning methodology is being presented since other variants like naive DASOM and DASOM-DCT can be derived in a straightforward way based on the formulation presented below. Essentially, a single pass of the first phase and a single pass of the second phase will be demonstrated since generalizing for the necessary number of first phase iterations followed by the required number of second phase repetitions is an obvious task. Every specific step of the procedure, which is detailed next, for updating the DASOM's parameters is largely based on the derivations contained in Appendix A and on the previously defined learning rules.

first phase

$$w_{ij}^{(next)} = w_{ij} + a\left[\left(x_j - \hat{x}_j\right) f'\left(\hat{z}_j\right) y_i + \sum_{h=1}^{L}\left(x_h - \hat{x}_h\right) f'\left(\hat{z}_h\right) w_{ih} f'\left(z_i\right) \tilde{x}_j\right], \quad 1 \le i \le M,\ 1 \le j \le L \tag{15}$$

$$b_i^{(next)} = b_i + a\sum_{h=1}^{L}\left(x_h - \hat{x}_h\right) f'\left(\hat{z}_h\right) w_{ih} f'\left(z_i\right),\ 1 \le i \le M \tag{16}$$

$$\hat{b}_j^{(next)} = \hat{b}_j + a\left(x_j - \hat{x}_j\right) f'\left(\hat{z}_j\right),\ 1 \le j \le L \tag{17}$$

second phase

$$c = \arg\min_{d} \sum_{e=1}^{E} h_{de} \sum_{m=1}^{M} (y_m - u_{me})^2 \tag{18}$$

$$w_{ij}^{(next)} = w_{ij} + \eta \sum_{e=1}^{E} h_{ce}(u_{ie} - y_i) f'\left(z_i\right) x_j, \quad 1 \le i \le M,\ 1 \le j \le L \tag{19}$$

$$b_i^{(next)} = b_i + \eta \sum_{e=1}^{E} h_{ce}(u_{ie} - y_i) f'\left(z_i\right),\ 1 \le i \le M \tag{20}$$

$$u_{ie}^{(next)} = u_{ie} + \theta h_{ce}(y_i - u_{ie}),\ 1 \le i \le M,\ 1 \le e \le E. \tag{21}$$

2.2. Feedforward operation

After the learning procedure, the outcome is a DASOM much like the one depicted in Fig. 2 that has all the parameters updated and tuned. As a consequence, a trained DASOM can be used for mapping input data onto its innate neural map. More specifically, the image of an input sample on the DASOM plane is defined as the neuron yielding the lowest weighted squared Euclidean distance between the hidden layer representation (of the input) and the corresponding codebook vector. Here the weighting refers to the neighborhood kernel defined over the topology of the DASOM grid. More colloquially expressed this neuron is often called "winner" or "best-matching unit". Algorithmically, this best-matching winner neuron is given by:

$$y_m = f\left(\sum_{l=1}^{L} w_{ml}x_l + b_m\right),\ 1 \le m \le M \tag{22}$$

$$c = \arg\min_{d} \sum_{e=1}^{E} h_{de} \sum_{m=1}^{M} (y_m - u_{me})^2. \tag{23}$$

A further advantage of this type of nonlinear projection is that it can be exploited to expand on data clustering and visualization. Finally, if the unimodal neighborhood function's radius is narrow enough to mostly encompass the closest neighbors, then in the majority of setups the previously detected best-matching neuron will coincide with the winner neuron of the original SOM:

$$c = \arg\min_{d} \|\mathbf{y} - \mathbf{u_d}\|^2. \tag{24}$$

Following this these two expressions can be used interchangeably given that the neighborhood size condition applies.

2.3. Singular value decomposition/principal components analysis SOM

If the nonlinear capabilities of the autoencoder (which play a crucial role for its performance) are removed then the respective devised nonlinear manifold is essentially substituted with a linear mapping. In such cases the representations achieved by an autoencoder span the same (high-dimensional) space as that produced by linear transformation techniques such as Principal Components Analysis (PCA) and Singular Value Decomposition (SVD), although there might be some rotation and/or skewing of the axes. The connection and similarities between neural network representations and PCA have been studied extensively in the past (Becker, 1991; Oja, 1982, 1992). These include: (a) simple linear neurons which have been shown to extract the principal component of an input sequence, (b) algorithms which learn a set of orthogonal projections in the directions of principal variation in the input distribution or a rotated subspace of these directions, and (c) single-layer-feedforward networks that compute PCA in unsupervised mode.

The PCA technique is intimately related to the SVD mathematical technique. To see this let $\mathbf{X}$ denote the data matrix where each row contains a sample and each column represents one of the L input dimensions. Given that $\mathbf{X}$ is centered the PCA can be seen along the lines of eigen-decomposition of the (symmetric) covariance matrix $\mathbf{X}^T\mathbf{X}/(n-1)$, where n represents the total number of samples provided. The result after diagonalization is $\mathbf{X}^T\mathbf{X}/(n-1) = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\mathbf{T}$, where $\mathbf{V}$ is an eigenvector matrix where each column contains a principal direction/axis of the data, and $\mathbf{\Lambda}$ is a diagonal matrix containing eigenvalues λ in decreasing order. In $\mathbf{XV}$ the principal components are given by the columns and the coordinates of each sample on the devised linear manifold are given by the rows.

Similarly, SVD for $\mathbf{X}$ yields $\mathbf{X} = \mathbf{H}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$, where the columns of $\mathbf{H}$ are unit vectors in the directions of maximum data variance ("right" singular vectors of $\mathbf{X}$), and $\mathbf{\Sigma}$ is a diagonal matrix containing singular values σ in decreasing order. Thus, it can be easily seen given that $\mathbf{X}$ is centered the singular values of SVD and the eigenvalues of PCA are related via: $\sigma^2 = (n-1)\lambda$. In what follows the SVD technique is going to be utilized because (a) it is more flexible since it does not implicitly necessitate a centering and scaling by $\left(\sqrt{n-1}\right)^{-1}$ of the data matrix, (b) it can be easily changed to perform PCA and (c) unlike in PCA where calculating the covariance matrix numerically can cause loss of precision SVD directly uses $\mathbf{X}$ thus producing more accurate results.

These SVD and PCA features can be aligned with the DASOM approach making SOMs based on linear manifolds possible. What is of importance, and to the best of our knowledge novel, is the fact that these manifolds continue to be adjusted according to the requirements of the neural mapping objective. Analogous to the core DASOM algorithm the learning rules of these networks are the following:

first phase

$$w_{ij}^{(next)} = w_{ij} + a\left[\left(x_j - \hat{x}_j\right) y_i + \sum_{h=1}^{L}\left(x_h - \hat{x}_h\right) w_{ih}x_j\right],$$
$$1 \le i \le M,\ 1 \le j \le L \tag{25}$$

or alternatively:

$$\mathbf{X} = \mathbf{H}\mathbf{\Sigma}\mathbf{V}^T \tag{26}$$
$$\mathbf{W} = \mathbf{V}^T \tag{27}$$

second phase

$$c = \arg\min_{d} \sum_{e=1}^{E} h_{de} \sum_{m=1}^{M} (y_m - u_{me})^2 \tag{28}$$

$$w_{ij}^{(next)} = w_{ij} + \eta \sum_{e=1}^{E} h_{ce}(u_{ie} - y_i)x_j,\ 1 \le i \le M,\ 1 \le j \le L \tag{29}$$

$$u_{ie}^{(next)} = u_{ie} + \theta h_{ce}(y_i - u_{ie}),\ 1 \le i \le M,\ 1 \le e \le E \tag{30}$$

where nonlinear functions are removed also from the activations $y_m = z_m$ and the reconstructed sample $\hat{x}_h = \hat{z}_h$.

During the first phase, the stochastic corruption of the input has been removed and instead, as in classical autoencoders, the original training sample is used. The main reason justifying this algorithmic decision is that the SVD cannot handle noise and corruption so there is no gain from including them. It should be apparent that the two alternative learning procedures of the first phase lead to two different kinds of models that span potentially the same linear manifold but in the general case vary on their representations. The first (25) and (28)–(30) is based on the linear autoencoder whereas the second (26)–(30) operates on the space created by SVD. In what follows we will call the later SVDSOM (Singular Value Decomposition Self-Organizing Map). In a similar fashion to the DASOM, certain variants can be devised based on (a) whether Eq. (29) is included into the adaptation procedure, and on (b) potential different linear manifolds exclusively assigned to individual neurons of the SOM map. For instance, a naive SVDSOM can result from training SOM on a preset and fixed set of linear representations devised by SVD without incorporating the subsequent adjustment of the eigenvector matrix ($\mathbf{W} = \mathbf{V}^T$).

Comparing the SVD and PCA techniques with the nonlinear (denoising) autoencoder a major advantage of the former is that

they are completely nonparametric and deterministic. With the use of a dataset and without any tweaking a repeatable result will emerge. Another significant advantage of SVD/PCA is the direct exploitability of the first two or three principal components in cluster detection/visualization tasks. While there is no guarantee that the answers will be ideal or useful or meaningful, nonetheless applying SVD/PCA to any dataset will deliver an answer. Finally, in comparison to the autoencoder gradient descent algorithm the SVD/PCA algorithm is typically more computationally efficient given the amount of data is not huge.

However, there are a number of shortcomings using the SVD/PCA as a replacement to a DA. Firstly, the devised linear hyperplanes' dimensionality is smaller in comparison to the sample number or input dimensions (whichever is smaller). Consequently, SVD and PCA are quite frequently outperformed by autoencoders with nonlinear curved manifolds on non-contrived datasets. This is because the autoencoders are more flexible and have enhanced representation capabilities. While the SVD/PCA produce full or reduced rank approximations of the data they cannot go above the problem-predefined number of representations. By comparison the representations for the DA can be set to an arbitrary number possibly higher in comparison to the respective input dimensionality. The linearity of SVD/PCA is a fundamental limitation as it creates a linear map and can only model linear relationships between variables. A further impediment of SVD/PCA approach lies in the structural biasing assumptions that principal components: (a) with larger associated variances represent interesting structure while those with lower variances represent noise, (b) are orthogonal. While these intuitive simplifications render the ensuing models soluble with linear algebra decomposition techniques the resulting model inherits methodological biases that may divorce the model results from the true solution.

2.4. k-means++ (denoising) autoencoder SOM

The employed combinatorial approach between the k-means++ and (denoising) autoencoder SOM algorithms is defined, along the lines described in Arthur and Vassilvitskii (2007), as:

1. Train a DASOM.

2.a. Choose the first cluster center $\mathbf{kc_1}$ randomly from $\mathbf{U}$.

2.b. Take a new cluster center $\mathbf{kc_i}$ by choosing $\mathbf{u_e}$ with probability $\frac{SD(\mathbf{u_e})^2}{\sum_{\varepsilon=1}^{E} SD(\mathbf{u_\varepsilon})^2}$.

2.c. Repeat step 2.b until k centers have been taken selected.

3. For $1 \leq i \leq k$ set the cluster represented by $\mathbf{kc_i}$ to be the set of codebook vectors that are closer to it than with any other cluster $\mathbf{kc_j}, j \neq i$.

4. For $1 \leq i \leq k$ set $\mathbf{kc_i}$ to be the center of mass of all codebook vectors that belong to it.

5. Repeat steps 3 and 4 until the clusters no longer change.

$SD(\mathbf{u_e})$ represents the shortest distance from a codebook vector to the closest center we have already chosen. Conceptually, k-means++ is set to operate on the same input space as the various SOMs but practically the definition/description of this space is guided by the self-organizing neurons' parameters and not by the samples' features. The actual samples' attribute values come to play afterwards during the computations for evaluation measures/criteria. It should also be noted that a positive side-effect of this combination is that crisp clusters, similar to those produced by typical clustering algorithms, can be constructed.

3. Experiments

We now set out to evaluate the clustering and mapping of the DASOM. In an effort to ensure a broad perspective of the experimental procedure, the basis of conducted comparisons was formed by both internal and external evaluation criteria. This translates into either including or excluding a priori knowledge about the preexisting data categories and class assignments. Assessing the performance of methods is done by evaluating clustering results by comparing the relative organization, compactness or sparseness, isolation and preservation of the clustered data. These are the so-called internal evaluation criteria. In contrast, a more direct measure of the method's effectiveness in analyzing the data relies on a direct comparison with prior knowledge (i.e., external knowledge gained from domain technical expert classification and scientific categorization) of the data. Evidently internal evaluation criteria are the only means for measuring relative clustering performance in all the cases where the datasets' category assignments are absent or unknown. In this context and under certain conditions, clustering becomes an optimization problem and the produced estimations could prove useful in guiding the training of a given model and so tune its parameters accordingly.

We compare the DASOM as a cluster visualization model with algorithms that share similar components and capabilities. Comparing with architectures that have characteristics in common with the DASOM standardizes the evaluation criteria such that the results produced by each of the methods reflect the DASOM's methodological strengths or weaknesses and are not systematic variations caused by architectural and algorithmic variances. Taking this into account, two sets of algorithms formed the basis of the conducted experiments. The first set of experiments uses the classical SOM, the SVDSOM, the AESOM and the DASOMs with different types of data input distortions (e.g. Gaussian/masking noise). The second set of algorithms uses an improved version of k-means (Arthur & Vassilvitskii, 2007) for the top-level component and then the codebook vectors and representations of the SOM, SVDSOM, AESOM and DASOM form the back-end part of the stacked architecture.

We now discuss the quantitative measures of clustering quality. Specifically, we present external clustering criteria such as normalized mutual information, cluster purity and the Rand index. In setting the notation for the equations, let the subscript p denote the partitioning of a set of S samples into P distinct clusters (a posteriori estimated by the model); similarly, let the subscript t denote the assignment of these samples into T categories (a priori defined in the dataset). The formulation of normalized mutual information (Strehl & Ghosh, 2002) requires the definition of a number of additional quantities. In so doing s_p symbolizes the number of samples in the pth group of the devised clustering and s_t represents the number of samples belonging to the $t$thclass of the preexisting categorization. Analogously, $s_{p,t}$ denotes the number of samples that are simultaneously present in the pth partition and the tth classification:

$$NMI = \frac{\sum_{p=1}^{P}\sum_{t=1}^{T} s_{p,t} \log\left(S\frac{s_{p,t}}{s_p s_t}\right)}{\sqrt{\left(\sum_{p=1}^{P} s_p \log\frac{s_p}{S}\right)\left(\sum_{t=1}^{T} s_t \log\frac{s_t}{S}\right)}}. \tag{31}$$

NMI attains values in the range $[0, 1]$ with lower values signifying higher uncertainty.

Purity is possibly the most widely used external measure for clustering tasks:

$$PUR = \frac{1}{S}\sum_{p=1}^{P} \max_{1\leq t\leq T} \left|s_p \cap s_t\right| \tag{32}$$

the resulting values lie in the $[0, 1]$ interval. Given that the majority voting principle is used for labeling each individual cluster it follows that purity is linked to accuracy. Although purity is intuitively straightforward and precise, it tends to favor small (in sample numbers) clusters with singletons scoring the highest values.

For every pair of samples there are four mutually exclusive cases. They are (a) where both samples belong to the same cluster from P and to the same category from T (their total number is denoted as SS), (b) where both samples belong to the same cluster from P but different categories of T, (c) where samples belong to different clusters of P but both belong to the same category from T and (d) where samples belong to different clusters of P and to different categories of T with their total number denoted as DD. The degree of overlap can be estimated by the Rand measure:

$$RAN = (SS + DD)\left(\frac{S}{2}\right)^{-1} \tag{33}$$

with values between [0, 1] where higher values indicate the coincidence of cluster partitioning and category assignments. Interestingly in the latter two criteria a degenerate monolithic clustering will receive a score equal or related to the fraction of the number of samples in the biggest category. In unbalanced datasets it is likely that this score approaches 1. This implies an optimal partitioning even though the actual clustering quality is low. On the other hand, entropy based criteria such as NMI are more robust against such phenomena.

Next we consider two cluster morphology measures; the quantization error and the Davies–Bouldin measure. Quantization error determines the adaptation of the network to the inputs/representations by averaging:

$$QE = \frac{1}{S}\sum_{s=1}^{S}\left\|\mathbf{y}^{(\mathbf{s})} - \mathbf{u}_{\mathbf{c}}^{(\mathbf{s})}\right\|, \tag{34}$$

where $\mathbf{u}_{\mathbf{c}}^{(\mathbf{s})}$ denotes the winner neuron's centroid for representation $\mathbf{y}^{(\mathbf{s})}$ of the sth sample. The resulting values are strictly non-negative $[0, +\infty)$ and lower values indicate a better fit of the neural map to the data. An alternative definition of this internal criterion exists in the form of a root mean squared-error:

$$QE = \left(\frac{1}{S}\sum_{s=1}^{S}\left\|\mathbf{y}^{(\mathbf{s})} - \mathbf{u}_{\mathbf{c}}^{(\mathbf{s})}\right\|^{2}\right)^{\frac{1}{2}}. \tag{35}$$

Since QE decreases when the number of map neurons increases (independently from the quality/distortion of the produced mapping) it cannot be used for comparing maps of different sizes or as the only criterion for adjusting or selecting the parameters of a network.

In essence, the Davies–Bouldin measure provides an estimate of the average of pairwise similarities between closest clusters:

$$DB = \frac{1}{P}\sum_{i=1}^{P}\max_{1\le j\le P, i\ne j}\frac{CD_i + CD_j}{MD_{ij}}, \tag{36}$$

where CD_i is the dispersion of cluster i, and MD_{ij} is the Minkowski distance between the codebook vectors of clusters i and j.

$$CD_i = \left\{\frac{1}{S_i}\sum_{j=1}^{S_i}\left\|\mathbf{y}^{(\mathbf{j})} - \mathbf{u}_{\mathbf{i}}\right\|^{q}\right\}^{1/q}, \tag{37}$$

$$MD_{ij} = \left\{\sum_{k=1}^{M}\left|u_{ki} - u_{kj}\right|^{r}\right\}^{1/r}, \tag{38}$$

where S_i is the number of samples in cluster i and $\mathbf{u_i}$ is the centroid of the same cluster. When $r = 2$ and $q = 1$, MD_{ij} becomes the Euclidean distance between centroids and CD_i represents the average distance between the assigned samples and calculated codebook vector of cluster i. Furthermore, DB produces non-negative values in the range $[0, +\infty)$. Higher dispersions and large distances between closest clusters translate to higher DB values which are indicative of poor or suboptimal clusterings and neural mappings.

For the (experimental) analysis that follows, when NMI, PUR and RAN are being considered as a group they are called external criteria/measures; similarly, when QE and DB are grouped together they form the set of internal measures/criteria.

A series of well-known widely-analyzed datasets formed the basis for all conducted experiments and analyses. This series comprised of coil i.e., Columbia Object Image Library (Nene, Nayar, & Murase, 1996), optdigits i.e., Optical Recognition of Handwritten Digits Dataset (Bache & Lichman, 2013), orl i.e., Database of Faces, usps i.e., Handwritten Text Recognition Database (Hull, 1994), and yale i.e.,Yale Face Database. Their details are summarized in Table 1.

3.1. Quantitative–Qualitative comparative results

The characteristics and the attributes of the SOM's parameters with respect to its behavior and performance have been extensively studied (Kohonen, 2001) across a wide range of problems (Kohonen, 2014). Furthermore models with certain adaptive parameters have been proposed (Berglund & Sitte, 2006; Shah-Hosseini & Safabakhsh, 2003). These parameter selection findings and guidelines can inform the DASOM development to the point where it shares a common set of algorithmic parameters, and also, shares a similar network architecture with the SOM. One such valuable strategy to use is that of a unimodal Gaussian neighborhood function with a wide initial radius that shrinks as the map evolves to include, at the end of the training, only the closest neighboring neurons on the lattice. This implementation averts the phenomenon of sparse and discontinuous projections.

A thorough grid based search has been conducted to discover semi-optimal values for the common parameters shared between the SOM, the SVDSOM, the AESOM and the various DASOMs with the different distortion mechanisms. The output of this procedure is a set of values and functions that yield models demonstrating good performance and a set of parameters that produces a balance between the internal and external measures for each individual dataset. We will elaborate on this last point in further detail later in this text. To compare the noise patterns a slightly different approach was followed since: (a) the corresponding parameters affect only the DASOMs, (b) a uniform comparison over the entire dataset collection is more preferable. Furthermore, the exact values for the Gaussian noise's standard deviation ($\le$1.0) and the masking noise's percentage ($\le$40%) have been chosen from our discoveries using the optdigits dataset as a reference benchmark.

A cursory examination of Table 2 shows that the differences in external criteria performances are marginal; even though the various autoencoders give better performance we are hesitant to argue an across the board superiority of the proposed approach. However, for certain datasets even the incorporation of a typical autoencoder which lacks the denoising aspect (like the AESOM or the SVDSOM) suffices for outperforming the rest of the algorithms. This makes clear the positive impact a representation layer has when inserted between the inputs and the output plane. The improvements introduced by the DASOM are evident both in cluster organization and compactness. Considering that QE and DB both explicitly take the input/representation space into account and that the only point of differentiation is the existence of the hidden representation space; its positive contribution becomes apparent. We believe that these noticeable performance improvements are an additional albeit indirect proof of the efficiency of the deep learning logic (even in a degenerate form i.e. with only one representation layer). Considering all reported results, the number of representations for DASOM was smaller in comparison to the number of inputs in so adding further evidence to the method's

Table 1
Characteristics of the datasets.

	Identifier	No. of samples	No. of features	No. of categories
Columbia object image library	coil	1440	1024	20
Optical recognition of handwritten digits	optdigits	5620	64	10
Database of faces	orl	400	1024	40
Handwritten text recognition database	usps	9298	256	10
Yale face database	yale	165	1024	15

Table 2
Evaluation criteria of (denoising) autoencoder SOM, SVDSOM and SOM paradigms on benchmark datasets.

Identifier	Evaluation measure	DASOM 25%	DASOM-DCT 25%	DASOM 0.5	DASOM-DCT 0.5	AESOM	SVDSOM	SOM
coil	NMI	**0.802 ± 1.23E−2**	0.785 ± 1.47E−2	0.798 ± 8.86E−3	0.788 ± 1.19E−2	0.790 ± 8.53E−3	**0.802 ± 8.66E−3**	0.776 ± 9.69E−3
	RAN	**0.970 ± 2.07E−3**	0.965 ± 2.97E−3	0.969 ± 1.32E−3	0.966 ± 2.70E−3	0.968 ± 1.33E−3	**0.970 ± 1.37E−3**	0.966 ± 1.12E−3
	PUR	0.809 ± 1.91E−2	0.783 ± 2.35E−2	0.799 ± 1.55E−2	0.784 ± 1.76E−2	0.784 ± 1.20E−2	**0.815 ± 1.48E−2**	0.765 ± 1.98E−2
	QE	2.423 ± 5.51E−2	2.286 ± 7.75E−2	2.394 ± 6.32E−2	**2.250 ± 7.07E−2**	2.521 ± 5.35E−2	7.015 ± 7.39E−2	9.026 ± 5.91E−2
	DB	1.951 ± 1.05E−1	1.920 ± 1.02E−1	2.029 ± 1.53E−1	**1.903 ± 1.06E−1**	1.984 ± 1.48E−1	1.935 ± 4.00E−2	2.998 ± 1.65E−1
optdigits	NMI	0.717 ± 3.88E−3	**0.754 ± 9.99E−3**	0.718 ± 5.69E−3	0.747 ± 1.44E−2	0.715 ± 5.70E−3	0.718 ± 4.49E−3	0.703 ± 3.84E−3
	RAN	0.925 ± 1.45E−3	**0.941 ± 2.87E−3**	0.925 ± 8.05E−4	0.939 ± 5.54E−3	0.924 ± 5.58E−4	0.924 ± 6.98E−4	0.922 ± 5.88E−4
	PUR	0.920 ± 5.13E−3	0.921 ± 6.15E−3	**0.923 ± 7.32E−3**	0.905 ± 2.17E−2	0.920 ± 1.07E−2	**0.923 ± 7.07E−3**	0.921 ± 4.52E−3
	QE	1.926 ± 1.75E−2	1.041 ± 2.40E−2	1.809 ± 1.99E−2	**0.959 ± 1.94E−2**	2.007 ± 3.99E−2	2.694 ± 9.74E−3	2.716 ± 9.70E−3
	DB	1.802 ± 3.89E−2	**1.297 ± 6.00E−2**	1.795 ± 5.48E−2	1.299 ± 4.73E−2	1.921 ± 3.95E−2	2.208 ± 6.17E−2	3.301 ± 9.24E−2
orl	NMI	0.788 ± 9.51E−3	0.794 ± 1.23E−2	0.789 ± 1.24E−2	0.786 ± 1.46E−2	0.796 ± 1.02E−2	**0.799 ± 8.32E−3**	0.781 ± 8.28E−3
	RAN	0.978 ± 1.01E−3	0.977 ± 1.80E−3	0.977 ± 1.64E−3	0.976 ± 2.18E−3	**0.979 ± 1.08E−3**	0.978 ± 1.25E−3	0.978 ± 9.75E−4
	PUR	0.671 ± 1.58E−2	**0.680 ± 1.97E−2**	0.672 ± 2.41E−2	0.670 ± 2.43E−2	0.677 ± 2.46E−2	0.678 ± 1.92E−2	0.649 ± 1.69E−2
	QE	2.781 ± 2.87E−2	1.618 ± 5.98E−2	2.569 ± 3.27E−2	**1.438 ± 4.84E−2**	2.853 ± 3.51E−2	5.760 ± 4.08E−2	7.936 ± 2.37E−2
	DB	1.607 ± 3.74E−2	0.978 ± 4.79E−2	1.544 ± 4.96E−2	**0.935 ± 3.72E−2**	1.874 ± 6.44E−2	1.614 ± 2.79E−2	3.123 ± 5.03E−2
usps	NMI	0.597 ± 8.79E−3	0.591 ± 1.03E−2	0.587 ± 6.36E−3	0.582 ± 7.23E−3	0.590 ± 1.19E−2	**0.603 ± 7.66E−3**	0.583 ± 5.10E−3
	RAN	**0.912 ± 1.55E−3**	0.910 ± 1.98E−3	0.910 ± 1.22E−3	0.909 ± 1.39E−3	**0.912 ± 1.24E−3**	**0.912 ± 1.47E−3**	0.910 ± 2.20E−3
	PUR	0.815 ± 1.38E−2	0.804 ± 1.48E−2	0.811 ± 9.14E−3	0.802 ± 9.34E−3	**0.816 ± 1.20E−2**	0.814 ± 9.00E−3	0.784 ± 1.08E−2
	QE	2.613 ± 2.73E−2	2.437 ± 3.25E−2	2.586 ± 2.90E−2	**2.406 ± 2.52E−2**	2.745 ± 2.62E−2	5.080 ± 1.26E−2	5.510 ± 1.54E−2
	DB	2.276 ± 5.33E−2	**2.173 ± 5.11E−2**	2.328 ± 5.05E−2	2.195 ± 4.85E−2	2.399 ± 4.15E−2	2.403 ± 5.35E−2	3.657 ± 4.85E−2
yale	NMI	0.650 ± 1.49E−2	**0.652 ± 1.57E−2**	0.644 ± 1.08E−2	0.643 ± 1.10E−2	0.640 ± 2.45E−2	0.651 ± 1.41E−2	0.643 ± 1.15E−2
	RAN	**0.940 ± 2.23E−3**	**0.940 ± 2.65E−3**	0.939 ± 1.07E−3	0.938 ± 9.30E−4	0.938 ± 3.74E−3	**0.940 ± 2.39E−3**	0.937 ± 2.29E−3
	PUR	0.660 ± 2.51E−2	**0.665 ± 2.75E−2**	0.648 ± 1.94E−2	0.648 ± 1.95E−2	0.641 ± 3.32E−2	0.647 ± 2.31E−2	0.646 ± 1.70E−2
	QE	3.445 ± 3.77E−2	2.810 ± 4.26E−2	3.312 ± 5.06E−2	**2.681 ± 6.60E−2**	3.376 ± 4.52E−2	7.602 ± 5.97E−2	8.958 ± 4.43E−2
	DB	1.716 ± 4.19E−2	1.405 ± 4.58E−2	1.671 ± 3.93E−2	**1.376 ± 4.15E−2**	1.695 ± 4.09E−2	1.696 ± 1.78E−2	2.503 ± 5.06E−2

The corruption percentage for the reported masking noise results is 25% and the Gaussian noise's standard deviation equals 0.5.

efficiency. Information in the form of hidden representations is presented to the output neural layer in a condensed form without noticeable loss or deterioration. The addition of a denoising component to the autoencoder SOM architecture (i.e. DASOM c.f. AESOM) delivers superior performance in most of the cases. Moreover, the DCT versions of the algorithm are frequently top performing with respect to the internal criteria, thus experimentally establishing/verifying the efficiency of the related training procedure. In summary, the interposition of a properly adjusted neural layer of stable and noise robust representations suggests a step forward also within the currently studied self-organizing framework. By using the information provided mainly via QE and DB it is also interesting to note that our findings are in favor of nonlinear activation functions (i.e. AESOM) as opposed to linear ones (i.e. SVDSOM). This result is rather expected since it is aligned with the long-held belief that nonlinear neural networks will continue to outperform linear algebra techniques in tasks that become increasingly complex.

We now evaluate the introduction of the hybrid *k*-means++ (denoising) autoencoder SOM paradigm (Table 3) and observe that the performance patterns are aligned with those found for SOM, SVDSOM, AESOM and DASOM (Table 2). The remarks regarding the utility of the different evaluation measures remain the same. The improved clustering quality along the autoencoder route and the comparably better performance characteristics of the DASOM, apply unaltered to this case also. Although numerically the reported results (of the employed combinatorial scheme) are inferior, the positive influence of the inclusion of a hidden representation layer in cluster organization and compactness is visible. It is also interesting to note that, except for only one dataset, the *k*-means++ that operates on the nonlinear manifolds produced by the DASOMs yields the best results. This provides additional proof of the robustness and improved characteristics of such noise incorporating representations. Even though, the examined *k*-means++ self-organizing approach should not be preferred when the objective is an optimal neural map, the fact that it provides the capability to derive crisp clusterings (according to the predefined cluster number), and not clustering results and visualizations in the broader sense as the SOM does, should not be ignored. It goes without saying that the choice of method/approach depends on each respective modeling problem.

3.2. Number of hidden representations, effect of noise distortion

Compared with the classical SOM there are additional parameters that are not adjusted automatically in SVDSOM, AESOM and DASOM. These are (a) the number of used representations (i.e. the count of neurons in the hidden layer) and (b) the type and degree of stochastic corruption (i.e. the type of noise) applied to the inputs during the first learning phase of the model. These parameters were systematically adjusted for SVDSOM, AESOM, DASOM and DASOM-DCT and are summarized in a series of figures (Figs. 3–8) that depict the quality of the neural map as a function of hidden representations. The effect of isotropic Gaussian noise with varying standard deviations is illustrated (Figs. 9 and 10) as well as the impact that masking noise has on the networks' performance (Figs. 11 and 12). All points plotted and values reported were taken from averages over 20 repetitions to ensure statistical validity. The optdigits dataset is the basis upon which this series of experiments

Table 3
Evaluation criteria of hybrid *k*-means++ (denoising) autoencoder SOM paradigms on benchmark datasets.

Identifier	Evaluation measure	DASOM 25%	DASOM-DCT 25%	DASOM 0.5	DASOM-DCT 0.5	AESOM	SVDSOM	SOM
coil	NMI	**0.751 ± 2.33E−2**	0.751 ± 1.69E−2	0.748 ± 1.35E−2	0.750 ± 1.62E−2	0.725 ± 1.83E−2	0.733 ± 1.21E−2	0.717 ± 1.95E−2
	RAN	0.948 ± 6.02E−3	**0.949 ± 6.76E−3**	0.945 ± 6.08E−3	0.943 ± 1.02E−2	0.941 ± 8.12E−3	0.944 ± 4.96E−3	0.942 ± 7.14E−3
	PUR	0.644 ± 3.48E−2	0.644 ± 3.83E−2	0.634 ± 2.54E−2	**0.645 ± 2.62E−2**	0.604 ± 4.31E−2	0.601 ± 1.39E−2	0.596 ± 3.93E−2
	QE	3.191 ± 1.23E−1	3.177 ± 1.01E−1	**3.131 ± 1.25E−1**	3.168 ± 1.17E−1	3.277 ± 7.51E−2	9.102 ± 1.39E−1	10.346 ± 1.26E−1
	DB	2.029 ± 1.13E−1	2.006 ± 1.16E−1	**1.986 ± 1.60E−1**	2.002 ± 1.13E−1	2.045 ± 1.12E−1	2.033 ± 9.33E−2	2.880 ± 1.48E−1
optdigits	NMI	0.682 ± 3.12E−2	0.711 ± 2.72E−2	0.693 ± 2.90E−2	**0.716 ± 3.12E−2**	0.674 ± 1.84E−2	0.676 ± 3.04E−2	0.677 ± 2.55E−2
	RAN	0.914 ± 1.41E−2	0.912 ± 1.40E−2	**0.919 ± 1.21E−2**	0.914 ± 1.13E−2	0.913 ± 1.01E−2	0.910 ± 1.55E−2	0.915 ± 9.65E−3
	PUR	0.718 ± 5.17E−2	0.708 ± 4.39E−2	**0.724 ± 4.15E−2**	0.721 ± 4.24E−2	0.706 ± 2.82E−2	0.700 ± 4.63E−2	0.707 ± 3.53E−2
	QE	2.544 ± 3.52E−2	1.614 ± 5.78E−2	2.398 ± 5.38E−2	**1.494 ± 9.36E−2**	2.582 ± 6.31E−2	3.280 ± 3.83E−2	3.305 ± 3.72E−2
	DB	1.957 ± 1.27E−1	1.380 ± 1.25E−1	1.918 ± 7.90E−2	**1.258 ± 1.07E−1**	2.049 ± 7.08E−2	2.253 ± 1.05E−1	2.631 ± 1.46E−1
orl	NMI	0.760 ± 8.09E−3	0.764 ± 1.23E−2	0.760 ± 1.31E−2	0.757 ± 1.24E−2	**0.774 ± 6.47E−3**	0.767 ± 6.64E−3	0.760 ± 1.14E−2
	RAN	0.970 ± 9.15E−4	0.967 ± 4.11E−3	0.970 ± 2.12E−3	0.966 ± 2.60E−3	**0.974 ± 1.10E−3**	0.971 ± 1.29E−3	0.969 ± 1.97E−3
	PUR	0.595 ± 1.60E−2	0.607 ± 1.72E−2	0.604 ± 2.73E−2	0.600 ± 2.64E−2	**0.621 ± 1.65E−2**	0.604 ± 2.20E−2	0.592 ± 1.93E−2
	QE	3.004 ± 2.76E−2	1.923 ± 4.04E−2	2.774 ± 2.86E−2	**1.715 ± 4.59E−2**	3.018 ± 4.06E−2	6.178 ± 4.70E−2	7.804 ± 6.22E−2
	DB	1.699 ± 4.58E−2	1.113 ± 3.76E−2	1.626 ± 4.81E−2	**1.078 ± 5.40E−2**	1.923 ± 5.26E−2	1.682 ± 3.46E−2	2.148 ± 5.17E−2
usps	NMI	**0.561 ± 2.44E−2**	0.552 ± 2.29E−2	0.553 ± 2.38E−2	0.545 ± 3.05E−2	0.542 ± 4.58E−2	0.540 ± 2.97E−2	0.535 ± 2.34E−2
	RAN	0.887 ± 1.69E−2	0.873 ± 2.42E−2	**0.891 ± 8.55E−3**	0.878 ± 1.67E−2	0.887 ± 1.68E−2	0.882 ± 1.30E−2	0.888 ± 7.53E−3
	PUR	**0.652 ± 3.66E−2**	0.643 ± 2.45E−2	0.648 ± 3.09E−2	0.631 ± 4.43E−2	0.649 ± 4.89E−2	0.628 ± 3.51E−2	0.624 ± 3.00E−2
	QE	3.231 ± 8.34E−2	3.103 ± 1.11E−1	3.194 ± 4.18E−2	**3.069 ± 6.83E−2**	3.365 ± 1.06E−1	6.147 ± 9.66E−2	6.307 ± 4.83E−2
	DB	2.411 ± 1.53E−1	2.291 ± 8.60E−2	2.417 ± 1.80E−1	**2.274 ± 1.30E−1**	2.620 ± 2.25E−1	2.434 ± 1.41E−1	3.103 ± 1.45E−1
yale	NMI	0.511 ± 2.15E−2	**0.528 ± 2.56E−2**	0.509 ± 2.77E−2	0.511 ± 3.09E−2	0.506 ± 2.17E−2	0.510 ± 2.36E−2	0.497 ± 2.89E−2
	RAN	0.901 ± 7.71E−3	**0.903 ± 6.76E−3**	**0.903 ± 7.29E−3**	0.898 ± 1.13E−2	0.894 ± 1.12E−2	0.896 ± 6.83E−3	0.891 ± 1.49E−2
	PUR	0.470 ± 2.81E−2	**0.473 ± 2.83E−2**	0.456 ± 3.18E−2	0.465 ± 3.41E−2	0.447 ± 2.80E−2	0.451 ± 3.28E−2	0.438 ± 3.15E−2
	QE	4.272 ± 6.31E−2	3.817 ± 9.39E−2	4.128 ± 5.36E−2	**3.643 ± 7.53E−2**	4.169 ± 5.42E−2	9.358 ± 1.63E−1	10.380 ± 8.31E−2
	DB	2.240 ± 7.67E−2	1.963 ± 5.60E−2	2.271 ± 1.03E−1	**1.910 ± 6.15E−2**	2.256 ± 1.31E−1	2.142 ± 9.83E−2	2.656 ± 1.01E−1

The corruption percentage for the reported masking noise results is 25% and the Gaussian noise's standard deviation equals 0.5. Parameter k is set to coincide with the number of categories existing in each respective dataset.

is built. This dataset is representative of the datasets used in this study. Moreover, the choice of optdigits upon which to perform this battery of experiments was because this dataset's input space dimensionality allows for a thorough and detailed examination of the potential number of representations ranging from degenerate values (e.g. 2) through to counts exceeding the input dimensionality by approximately 50%. Each set of figures is composed of two frames where the left-hand side frames contain graphs of the internal criteria and the right-hand side frames depict values for external measures. More specifically the right-hand side frames show the normalized mutual information (31), the purity (32), and the Rand measure (33). The left-hand side frames in these figures depict the quantization error (34) and (35), and the Davies–Bouldin measure (36).

The AESOM and DASOM algorithms have similar behavior and comparable performance. The outputs gradually stabilize after the leading degenerate cases (that involve extremely small numbers of representations) were similar oscillations and abrupt jumps are observed. This is on display in the graphs where all the plots become concave and monotonically increasing with increasing number of hidden representations. The RAN, PUR and NMI evaluation measures' plateau (Figs. 3–8) in each of the cases is around 10, 15 and 20 hidden representations respectively. This asymptotic increase is similar for DB stabilizing at 30 to 40 representations whereas for QE the convergence to a maximum value is slower. Compared to the input samples' feature count which is 64 the DASOM's hidden layer representation of the input space is more compact while preserving the descriptive details and good levels of performance. The behavior and performance for the DASOM-DCT variants are comparably better, since not only do they demonstrate significant improvements for QE and DB but they continue to ameliorate across a wide spectrum of hidden representations. More significantly, they appear to gradually converge (except from PUR) at an optimal level as the number of neurons in the hidden layer continues to increase. Interestingly, the graphs of AESOM and DASOM are similar in shape and values but there appears to be a shift between them. This displacement indicates that fewer representations are required for a certain level of clustering quality (viz. RAN, PUR and NMI) that is dictated by an external measure. However, this gain comes at the expense of the internally assessed clustering quality (viz. QE and DB) deteriorating more quickly.

The contradicting optimization directions are revealed by examining the external and the internal measures in isolation. What is seen is that an improvement in the former leads to a deterioration in the latter and vice versa with the only exception to this observation being for DASOM-DCT. The internal and external measures are based on different principles and evaluate aspects of cluster composition versus the shape and relationship of elements within a cluster of a neural map. Arriving at an optimal balance that compromises the opposing tensions essentially suggests a heuristic technique for parameter selection and tuning. In attempting to arrive at this balance we observe that the internal evaluation criteria show greater sensitivity to parameter change because they reveal differences between the models. These model differences can be obscured with an exclusively optimal choice of parameters based on external measures. On the contrary, when QE and DB are taken into consideration it is revealed that DASOMs perform better than the AESOM and the SVDSOM. Further, for the optdigits dataset the Gaussian noise gives marginally better results when compared to the masking noise. However, this observation is reversed for some of the remaining datasets and so cannot be argued as a general result.

A demonstration of the effect of the different types of noise is further studied (see Figs. 9–12). In all cases the number of representations is equal to half the number of inputs since it has been observed during the previous analysis that such a selection performs adequately. Exceeding a threshold (approximately 1.5 in this case) for the Gaussian noise's standard deviation leads to deteriorating results as is evident from the fast decreasing NMI and PUR values. As noted earlier a less structured and descriptive output plane corresponds to a more compact representation of the input space and vice versa. Nevertheless, of greater interest is the invariance of NMI and PUR for values of noise below 1.0 which is then followed by a decrease in QE and DB. This implies that the

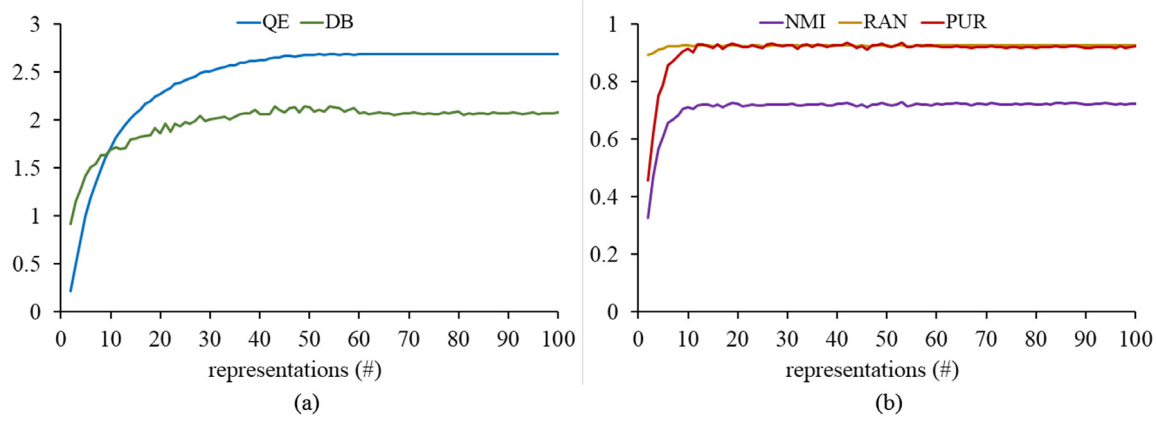

Fig. 3. The SVDSOM clustering capability as a function of hidden representations measured by (a) quantization and compactness using the QE and DB methods and (b) accuracy using measures NMI, RAN and PUR.

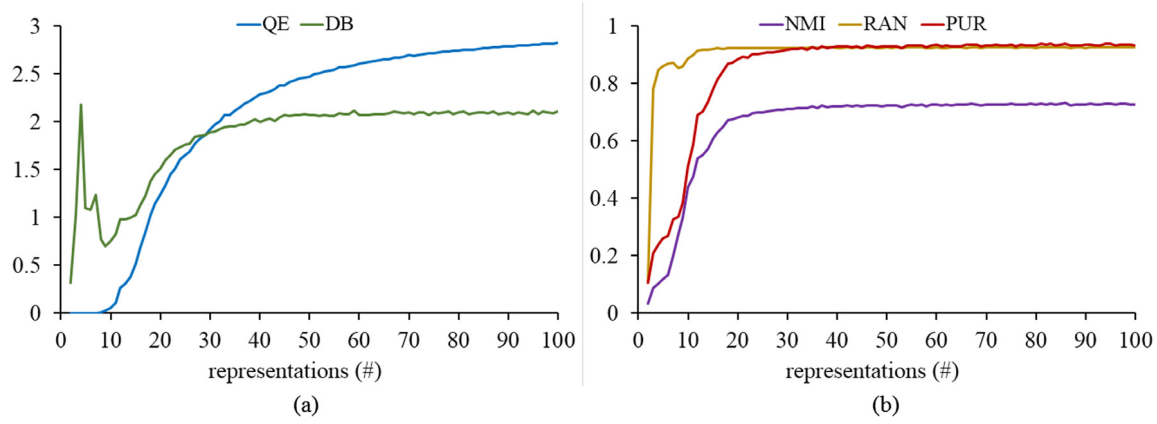

Fig. 4. The AESOM clustering capability as a function of hidden representations measured by (a) quantization and compactness using the QE and DB methods and (b) accuracy using measures NMI, RAN and PUR.

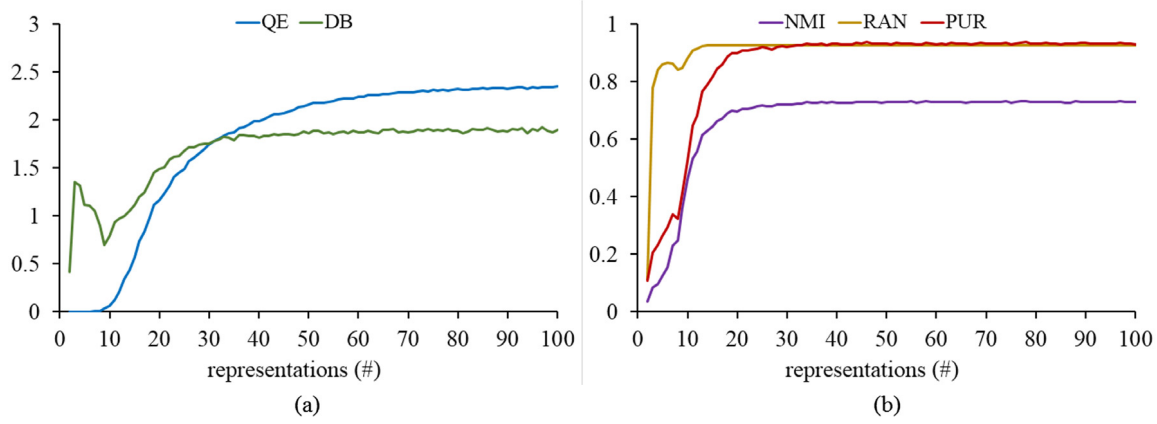

Fig. 5. The DASOM clustering capability as a function of hidden representations, with Gaussian noise's standard deviation kept constant and equal to 0.5, measured by (a) quantization and compactness using the QE and DB methods and (b) accuracy using measures NMI, RAN and PUR.

representations that have been devised by the denoising method along with the dependencies existing in the corresponding hidden space are more robust and descriptive compared to the AESOM approach that is devoid of input corruption.

Before examining the results obtained with the second type of noise it is important to clarify the different mechanisms that are invoked for processing them. The mechanisms that process Gaussian noise are straightforward. The learning algorithm detects the invariant dependencies in the data and infers representations that are robust against the introduced distortions. In contrast, in the masking noise implementation the strategy is to replace the values of features by a default value (i.e. 0 for this series of experiments). Switching off certain components of the network and thus removing the corresponding parts of information effectively forces the denoising training procedure to fill in the missing parts. Therefore producing representations that rely on "dependencies between dimensions in high dimensional distributions" (Vincent et al., 2010).

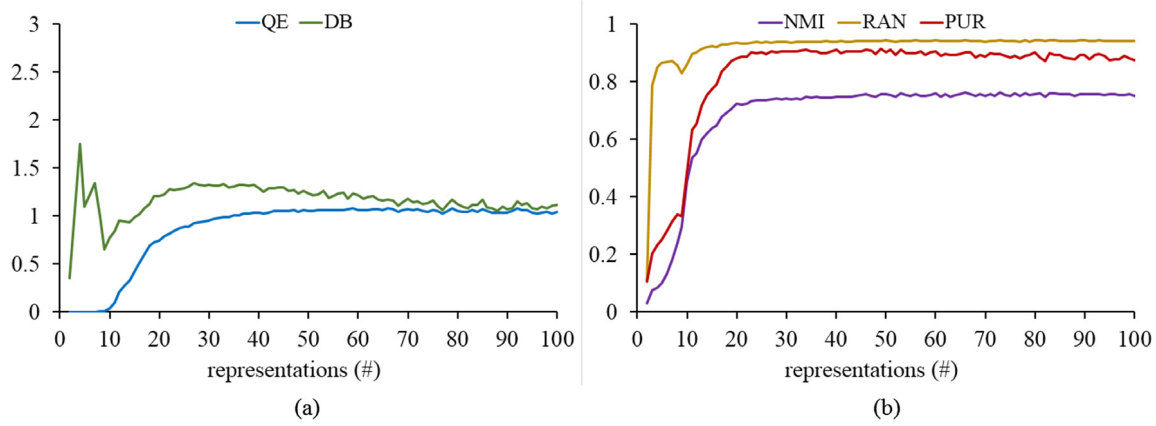

Fig. 6. The DASOM-DCT clustering capability as a function of hidden representations, with Gaussian noise's standard deviation kept constant and equal to 0.5, measured by (a) quantization and compactness using the QE and DB methods and (b) accuracy using measures NMI, RAN and PUR.

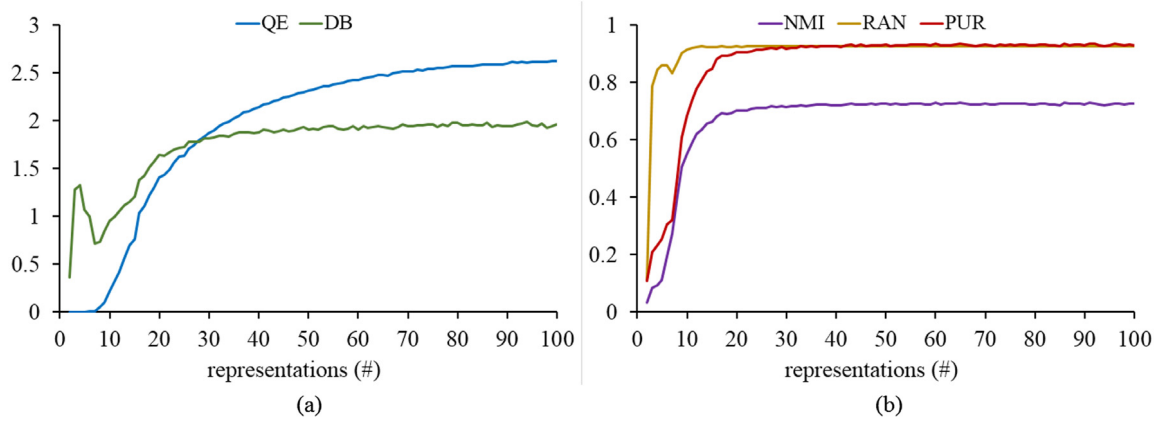

Fig. 7. The DASOM clustering capability as a function of hidden representations, with the incorporation of masking noise, measured by (a) quantization and compactness using the QE and DB methods and (b) accuracy using measures NMI, RAN and PUR. The percentage of zero-valued features for the masking noise scheme remained fixed at 25%.

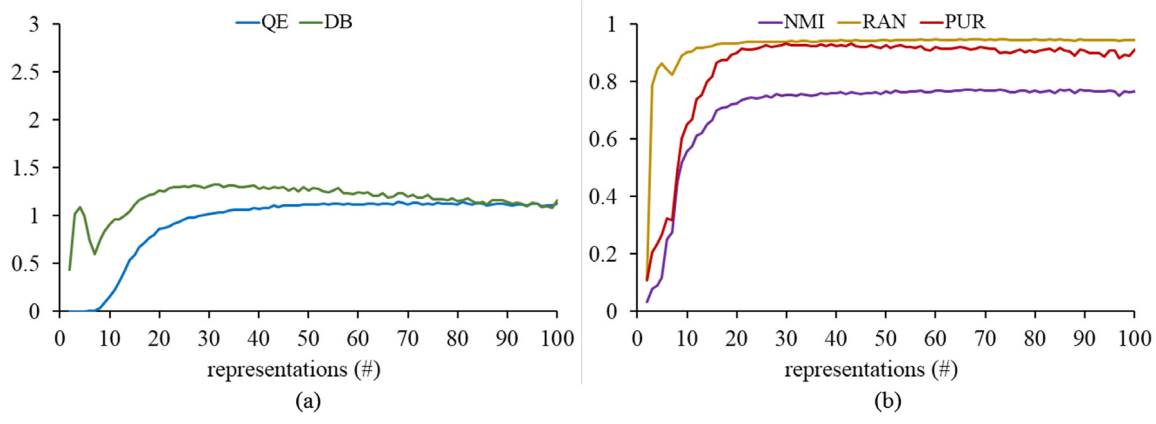

Fig. 8. The DASOM-DCT clustering capability as a function of hidden representations, with the incorporation of masking noise, measured by (a) quantization and compactness using the QE and DB methods and (b) accuracy using measures NMI, RAN and PUR. The percentage of zero-valued features for the masking noise scheme remained fixed at 25%.

While the conclusions drawn through an examination of the masking noise results (Figs. 11 and 12) are comparable with the analysis of the Gaussian noise results, the effect of increasing percentages of zero-valued features is less prominent. Nonetheless, clustering quality (NMI and PUR) deteriorates as noise increases with only small improvements to the adaptation of the network (QE and DB). What is clear from our attempt to find a middle ground between internal and external measures is that there is a limit for masking noise percentages (approximately 40% for the optdigits dataset) that both preserves an acceptable level of clustering efficiency and at the same time allows the improvement of internal evaluation measures. In a final note: the abrupt jumps observed with 100% masking noise are an aftereffect of canceling out the first learning phase of the DASOM algorithm (since all

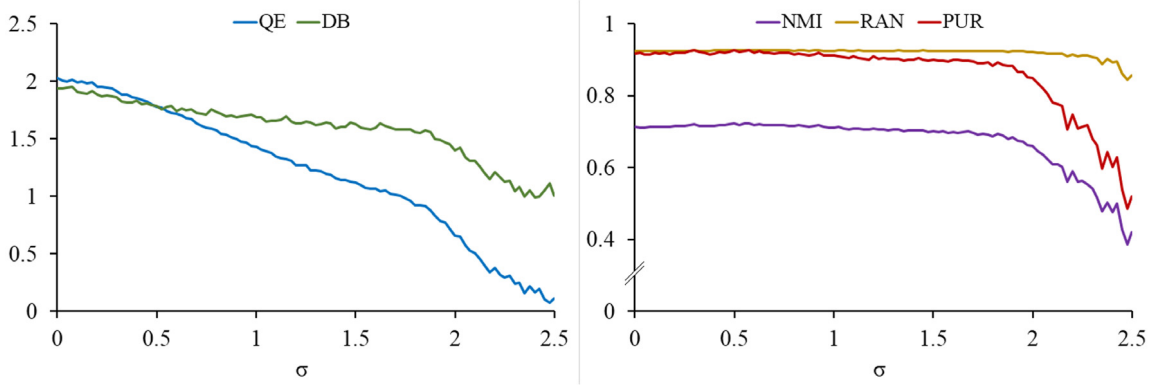

Fig. 9. The effect of increasing Gaussian noise distortion on a DASOM with 32 hidden neurons.

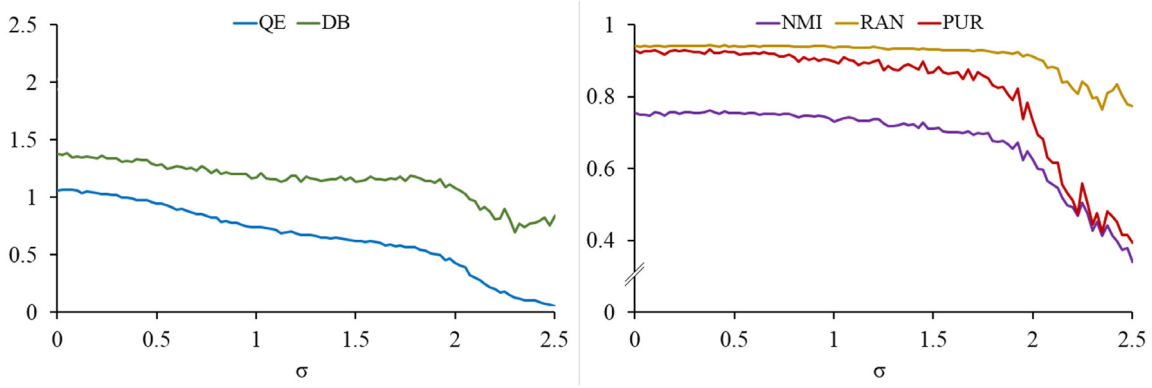

Fig. 10. The effect of increasing Gaussian noise distortion on a DASOM-DCT with 32 hidden neurons.

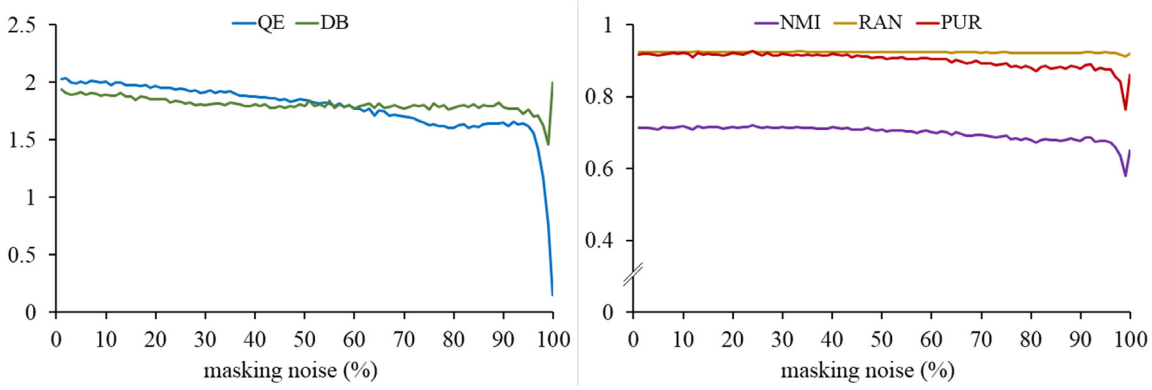

Fig. 11. The impact of varying percentages of masking noise corruption on a DASOM using 32 hidden representations.

inputs are turned off) and using only the adaptations designated by the rest learning phases.

3.3. Manifold of representations

Earlier we discussed the synergy and potential co-adaptation of the hidden representation and neural map spaces. The question then is in which direction and to what extent should the devised (nonlinear) manifold be affected and transformed by the modeling problem. This matter remains unresolved even in the case of deep architectures in supervised learning. The premise is that since the classification rate is improved the representations at each layer are most likely improved during the deep learning phase where the information of the output error is propagated back across the whole architecture down to the input, providing estimates of the weight parameters along the way. Based on the outcome this is without doubt the case but, as has already been mentioned, this is not proof that for a different adjustment to the magnitudes or directions of the weights the result would be worse. For instance, the Extreme Learning Machines (ELMs) comprise a single hidden layer network with randomly parameterized input — hidden layer weights and analytically determined hidden layer — output parameters. Although ELMs use such a completely different architecture that is shallow and wide they produce equally good results at certain problems. These considerations only make clear that the machine learning framework lends itself to continued debates on whether optimality and uniqueness-of-solution is definitive and achievable.

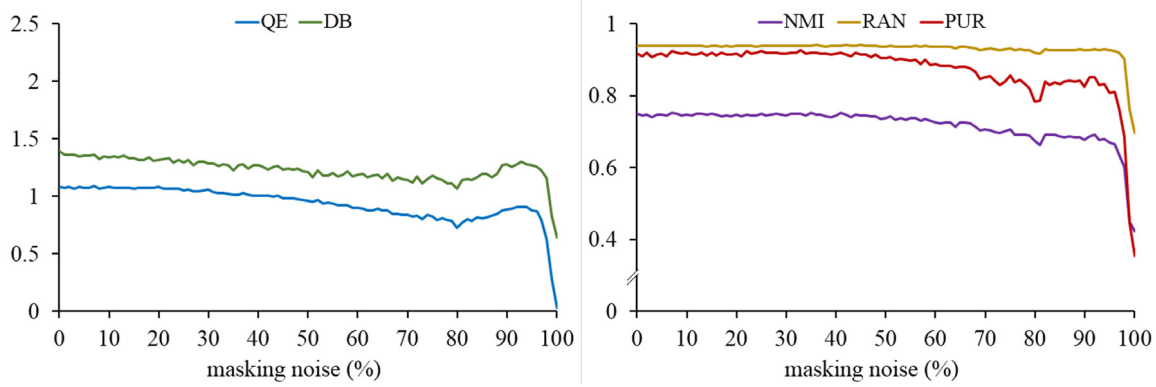

Fig. 12. The impact of varying percentages of masking noise corruption on a DASOM-DCT using 32 hidden representations.

We now attempt to consider a very specific and rather limited aspect of this unresolved matter as is relevant to the model and its training processes presented here. The role of the first training phase is to devise a transformation of the input space onto a nonlinear manifold, something that is achieved by employing a reconstruction criterion. Subsequently, based on a neural mapping cost function, the second phase seeks an ordered, smooth, representative projection of the transformed inputs onto a low-dimensional lattice. It immediately becomes apparent that there is not any a priori theoretical guarantee on the optimality of the devised representations. Even though experimental results are in favor of an interjected representation layer, we aim to investigate whether and how much the initially produced hidden representation manifold changes in the transition from the autoencoder to the self-organizing technique. Nonetheless the investigation that follows here is being carried under certain and specific conditions. subsequently any attempt to generalize or interpolate the findings to deeper or more complex architectures does not follow naturally.

The first condition to consider is that the weights of the encoding and decoding process should be tied. This is particularly important since the encoder parameters, that are being adjusted during the second learning phase, rely on this architectural symmetry with the decoder parameters to preserve the integrity of the results. Simply, the input — hidden layer connections cannot continue to change independently from the hidden layer — output connections that remain fixed. Similarly, the employed autoencoder network should not incorporate a bias or offset parameter since $\mathbf{b}$ and $\hat{\mathbf{b}}$ cannot be tied (since they have different dimensionalities, in the general case) and moreover parameters cannot be tuned independently from ones that remain static.

The four architectures: SVDSOM, AESOM and DASOMs with masking and Gaussian types of corruption that are potentially suited for this series of experiments are now elaborated upon. The manifold's dimensionality is set smaller compared to the input layer's dimensionality because the effective rank of the SVDSOM cannot go above the sample size or dimensionality (whichever is smaller). Further the hidden representation size of the AESOM should be smaller than the dimensionality of the inputs, if not it overfits and approximates the identity function. To simplify the graphs and make the results easier to examine we have opted for equal durations of training steps for all three learning phases. Notwithstanding the shortcoming that it is not the optimal epoch hyper-parameter option. Each graph illustrates three ordered and successive training phases. Phase A identifies with the first learning phase of the AESOM and DASOM, as well as identifies with the alternative (linear algebra derived) first phase of the SVDSOM. During phase B only the codebook vectors are being adjusted ($\theta > 0$) where the weights are kept constant ($\eta = 0$). Last, throughout phase Γ the parameters of each model are tuned in an interwoven and concurrent way according to the proposed second phase learning rules (so both $\eta > 0$ and $\theta > 0$). It can be easily seen that combinations of these phases are the building blocks of the naive, standard and DCT versions of the proposed algorithms. The same metrics that have been used in all previous experimental investigations are used in the current setup. For estimating or measuring the performance of the autoencoder module the (averaged over all inputs) reconstruction error is used:

$$SQER = \frac{1}{S}\sum_{s=1}^{S}\left\|\mathbf{x_s} - \hat{\mathbf{x}}_\mathbf{s}\right\|^2 \tag{39}$$

where S symbolizes the overall number of available training samples. Excluding the training phases' durations, all experimental parameters have been kept the same as with the previous experimental setups to ensure coherency and compatibility between the reported results (e.g. 32 hidden representations, 25% masking noise corruption and Gaussian distortion with 0.5 standard deviation). Characteristic or indicative results on the optdigits dataset for the four models are displayed in graphs (Figs. 13 and 14). Each plot triplet demonstrates the learning trajectory of each one of the SVDSOM, AESOM and DASOM models. We present the results in this way since the value ranges and the limits of the quality measures are vastly different, and to include them in a single graph would conceal important details. Nevertheless, as previously discussed the x-axes are shared between all graphs within a triplet.

The design and optimization criteria of the learning algorithm are visible in the training progress plots. The autoencoder is trained during phase A. Following this, by keeping the representation manifold fixed, the output neural mapping is tuned during phase B. Lastly, throughout phase Γ both components of the model co-adapt yielding the final trained model. It should be noted that since class/category information is not used during the training of the model (in contrast to supervised approaches) the observed convergence is approximate and is obviously guided by the shapes and values across time of the training functions. For instance, as is common practice, the neighborhood radius and learning rate functions are both monotonically decreasing functions of time. As a result, the achieved level of performance is not guaranteed to reach the global optimum (given this exists) since at least with regards to the external quality criteria certain better states have been missed in favor of more stable ones. Nevertheless, this observation is visible through the internal quality measures that reveal a different behavior. As previously discussed the balance between the opposing tendencies during the final stages of training remains under the control of the modeler who is guided by the demands of the application. It is also reasonable that during these final stages (phase Γ) the system is more sensitive to hyper-parameter variations because a more complex and intertwined

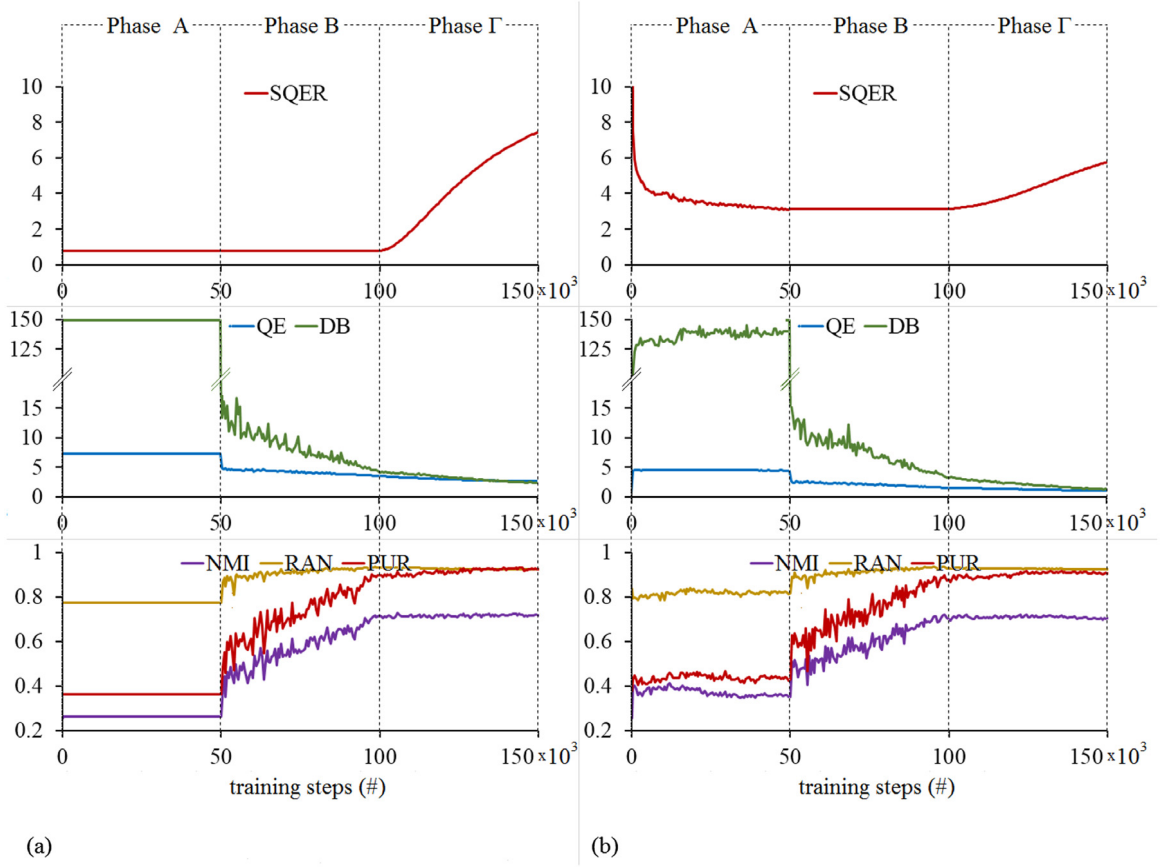

Fig. 13. (a) Training progress of the SVDSOM, alongside the trajectories of the respective quality measures. (b) Training progress of the AESOM, alongside the trajectories of the respective quality measures.

learning process which involves all the components takes place. As a side note, the nearly horizontal lines are the outcome either of zero learning rates during specific training time intervals or of the alternate linear algebra based learning technique in the case of SVDSOM.

With regards to the main question under consideration the outcome is somewhat expected: even though the autoencoder reconstruction error (SQER) increases or is comparably higher, the overall performance of the model improves or is comparably better. Two observations support the resulting conclusion. (a) Although SVDSOM achieves by far the lowest SQER, during phase B its performance (according to QE and DB measures) is evidently worse in comparison to AESOM and DASOM. To be precise, NMI, RAN and PUR show SVDSOM to perform similarly to the rest but still the lower SQER does not seem to play an analogously important role. (b) Observing the first few thousand training steps of phase Γ, when the models' efficiencies are still improving one can easily see that this happens inversely from the various autoencoder modules' performances. The SQER of the linear and nonlinear autoencoder modules get (monotonically) worse whereas the overall models become (asymptotically) better. Nevertheless, as it becomes evident at the end of phase Γ where most of the external criteria start to drop, a very high detuning of the autoencoders gradually starts to have a negative impact. If the training algorithm were to run for longer then this behavior will become more prominent.

In summary, there in not a one-to-one correspondence between the efficiency of the autoencoder (measured by the reconstruction error) and the performance of the overall model (as assessed by the external and internal clustering criteria). In fact, deviations (i.e. deterioration) from the optimal initial representation space seem to augment the eventual neural mapping. On the other hand, substantial increases in the reconstruction error certainly affect the model in a negative way. A rule of thumb emerges from these experiments: that is as long as the autoencoder's fluctuations remain within reasonable limits (with regards to the initially reached optimum), the various autoencoder SOMs have room for potential improvement.

4. Visualizing

The visualization of multidimensional complex data is a useful means with which to analyze and summarize domain space information. Visual inspection is often used to interpret results, study hidden relations and draw conclusions about the underlying structures of the data.

One of the main functions of the SOM is data clustering which is frequently coupled with a variety of graphical representations of the final outcome. The clustering is done in a so called overloading manner such that there is not a one-to-one correspondence between a lattice neuron and a cluster center or descriptor since nearly always clusters are formed by variable-size sets of neighboring neurons. In this context the DASOM can be thought of as an emergent system (Ultsch, 2005) in the sense that high-level structures (such as clusters) arise from the synergy and cooperation of elementary modules (viz. neurons). These abstract descriptions of complex data emerge along (partially) unsupervised procedures that at varying degrees incorporate both the whole (i.e. the neuron's topology) as well as the parts (i.e. the hidden representations).

The intrinsic arrangement of neurons by SOMs on a two-dimensional plane (and less frequently a three-dimensional surface) provides the overarching framework for visualizing the

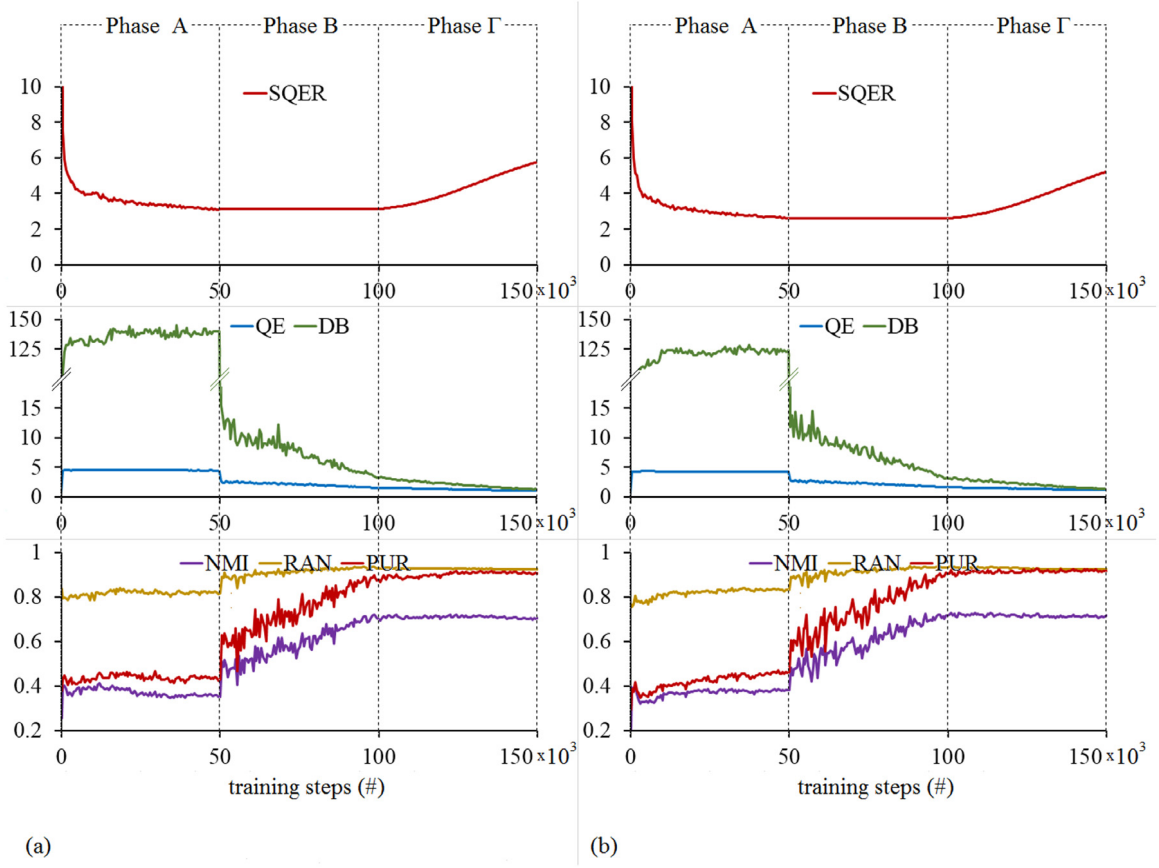

Fig. 14. (a) Training progress of the DASOM (which uses Gaussian noise corruption), alongside the trajectories of the respective quality measures. (b) Training progress of the DASOM (which uses masking noise), alongside the trajectories of the respective quality measures.

Table 4
List of characteristic SOM visualization techniques and methodologies.

Method	Synopsis	Reference
U-matrix	Local cluster boundaries are revealed by depicting pairwise distances of neighboring prototype vectors	Ultsch (2003)
P-matrix	Shows a landscape of density relationships estimated by the number of samples that are within a sphere with a certain radius around the prototype vectors, the radius is a quantile of the pairwise distances of the data vectors	
U*-matrix	A combination of U-matrix and P-matrix, the sum of distances of each node to its direct neighbors is multiplied by a scaling factor induced by the local density of the data points around the corresponding prototype vector	Ultsch (2005)
Connectivity strength matrix visualization	Integrates data distribution into a weighted Delaunay triangulation which, when displayed on the SOM grid, enables visualization of the manifold structure, and also, guides the capture of cluster boundaries based on how strongly (or weakly) various parts of the data manifold are connected	Tasdemir and Merényi (2009)
Clusot surfaces	Use both the Euclidean distance between the codebook vectors of the SOM and the neurons' sample assignment frequencies for estimating the cluster distribution	Brugger, Bogdan, and Rosenstiel (2008)
Gradient fields	Flow diagram where the length and direction of each arrow indicate towards where the cluster centers are most likely located, the entirety of the arrows forms a smooth vector field	Pölzlbauer, Dittenbach, and Rauber (2006)
Borderlines visualization	Derived from the gradient fields as an equivalent that emphasizes at showing likely cluster boundaries	
Visualization induced SOMs	Regularize the distance between a neighboring neuron to the winner such that the distances between the neurons on the map reflect the corresponding distances in the data space, this produces a smooth, quantitatively measurable, evenly graded mesh through the data points	Yin (2002)
Smoothed data histograms	By assigning a weighted membership of data vectors to the map, a visualization of the data density distribution on the output is produced	Pampalk, Rauber, and Merkl (2002)
Component planes Response surfaces	Projections of the values of a single vector component (of the codebook) in all map units. Show the (averaged) prototype vectors' responses for the data sample(s)	Vesanto (1999)

data structures. Exactly, this characteristic provides the foundation for analogous DASOM visualizations. A considerable number of generic SOM visualization techniques have been proposed during the past years. The list found in Table 4 is indicative of their polymorphy. Disregarding the specifics and the details of the individual algorithms the common theme that binds them is that

they augment and refine the low-dimensional projection of the data space, whereupon cluster structures might be detected.

The Euclidean distance $d(\mathbf{x_1}, \mathbf{x_2}) : \Re^L \rightarrow \Re$ and the dot product $\mathbf{x_1} \cdot \mathbf{x_2} : \Re^L \rightarrow \Re$ (i.e. angle) are the most effective mathematical metrics that measure characteristics such as equality, similarity and proximity in numerical vector spaces. The hidden representations that are linked to the input layer and the lattice parameters that are associated with the output are numerical (i.e. not categorical or nominal), thus allowing the DASOM to operate in Euclidean space. Measuring the characteristics of the clusters therefore follows simply from the SOM analytics and renders a host of visualization techniques (Table 4) directly applicable to the DASOM prototype with no or very minor adaptations required.

Having now argued that a convenient inheritance of methodology exists we pause for a moment on the codebook vectors as they are used to encode and quantize the features and attributes of input data in the classical SOMs, but certain differences come about when they are used in the DASOM. In DASOM a hidden layer is interjected between the inputs and the output layer therefore the codebook vectors summarize and interpolate the devised representations. This leads to two correlated aftereffects arising from this phenomenon. First, the internal projection or mapping mechanism is based on higher robust representations of data and not merely on the data itself. Second, the produced clustering visualizations depend on and are guided by these representations. The formed cluster structures and data relations therefore capture and depict information retrieved by this higher level of abstraction and as a result subsequent interpretations of the obtained results ought to account for this characteristic.

4.1. Cumulative representation planes, stacked representation planes

We now conduct DASOM experiments on tumor gene expression data collated by The Cancer Genome Atlas Research Network (http://cancergenome.nih.gov/). The data (https://data.mendeley.com/datasets/sf5n64hydt/draft?a=2cc3a3c0-8261-41f0-8e2b-e50e7cacfeb5) consists of the RNA sequencing values from tumor samples belonging to five separate cancer types. In total 2086 samples of Breast Invasive Carcinoma (BRCA), Kidney Renal Clear Cell Carcinoma (KIRC), Lung Adenocarcinoma (LUAD), Lung Squamous Cell Carcinoma (LUSC) and Uterine Corpus Endometrial Carcinoma (UCEC) tumors have been retrieved. From each sample's full RNA sequencing expressions we extracted those of the landmark genes (*GEO: GPL20573*) that could be traced in the original gene set. The DASOM which has been trained on this data forms the baseline for the presented mappings. The visualizations use an 8×11 lattice of output neurons that are arranged in an orthogonal grid. While in the clear majority of cases and applications (including the experimental results of the previous section) a hexagonal grid is used, it became evident that the orthogonal mesh is the best option for the proposed three-dimensional visualization.

The trained DASOM which incorporates 100 representations yields a distribution of samples on its neural output where the different types of cancer occupy different regions separated by well-defined boundaries (Fig. 15). The lung carcinomas' (LUAD-LUSC) are an exception as these clusters have overlapping areas even though the majorities of their respective samples are placed in non-neighboring neurons. This finding persists with different initialization and training parameters and the main reason for this could be related to the fact that these two types of cancer are found in the same system, whereas the remaining types of tumors affect different organs of the human body. This RNA expression-based proximity and overlap between the two lung cancers is detected automatically through a purely unsupervised procedure (ignoring any kind of category information). Arriving at this result in this way lowers the probability of its occurrence being artificially induced.

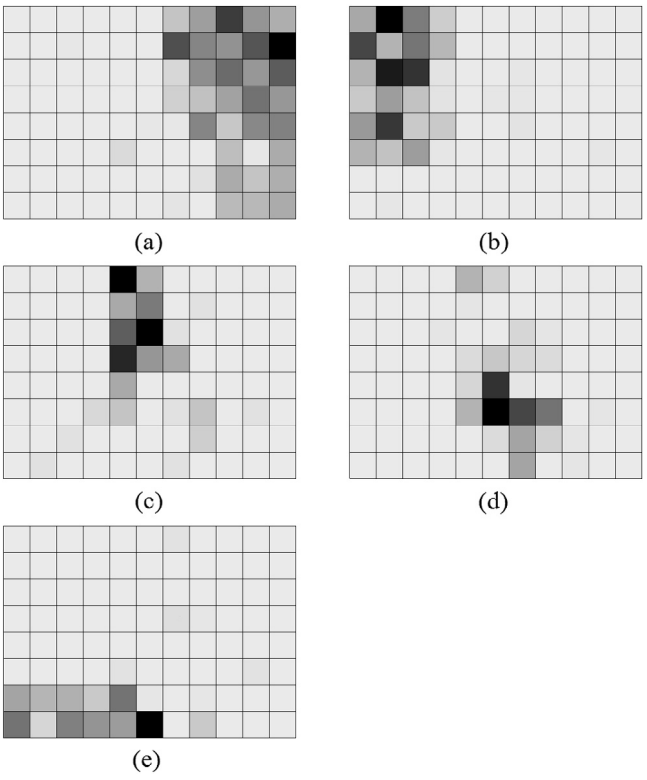

Fig. 15. Characteristic neural mappings for the cancer type dataset. Each tumor's sample distribution on the DASOM output plane (i.e. the number of samples that are assigned to each individual neuron) is illustrated by grayscale squares. Darker and lighter shades indicate higher and lower numbers of assigned samples respectively. Density mappings shown for (a) BRCA, (b) KIRC, (c) LUAD, (d) LUSC, and (e) UCEC.

Previously we mentioned that the component planes visualization technique is directly applicable to the DASOM case. A significant difference when implementing this technique in the DASOM framework is that the devised planes are composed of values of a single representation (instead of an input component) across all map neurons. Upon inspection of a given representation an insight into the distribution of its values across the neural map as well as a revelation of its correlation with specific regions or clusters is gained. Consider two representation planes of the previously trained DASOM (Fig. 16) as indicative cases of a representation plane with widespread values (across clusters) and of a representation plane demonstrating high values in a sub-region of the enclosing BRCA cluster. In Fig. 16 the grayscale squares represent the distance (viz. fitting) of each output neuron with respect to a specific representation.

While an analysis of the representation planes provides an insight into the contribution of the hidden representations in the formation of the clusters, the more important question of the actual inputs' participation and contribution to the observed clusters cannot be answered in this way. The proposed Cumulative Representation Planes (CRPs) fill this gap by revealing the correlation degree between an input feature and the neurons of the DASOM mapping. This information is gained from the weighting of all the representation planes (making up a hidden layer) according to the contribution an input feature makes to each hidden layer neuron. The process for calculating the CRP of input feature j is:

$$wu_e = \sum_{m=1}^{M} w_{mj} u_{me},\ 1 \leq e \leq E. \tag{40}$$

where $\mathbf{wu}$ is the vector containing the necessary values for constructing the CRP of a given input. Now consider two characteristic

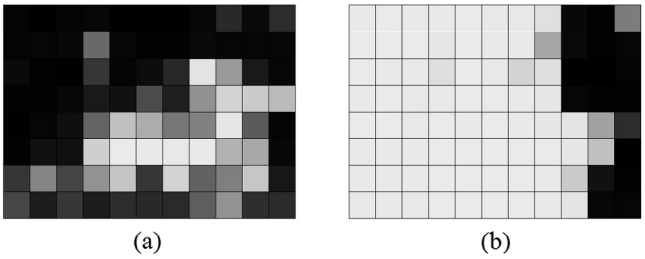

Fig. 16. Representation planes for two different representations from the DASOM's hidden layer. The darker and lighter grid elements denote higher and lower values of the corresponding codebook variables. (a) Representation plane corresponding to a hidden representation that extends along and describes clusters KIRC, LUAD and BRCA. (b) Representation plane of a different hidden representation that is closely related to the BRCA cluster.

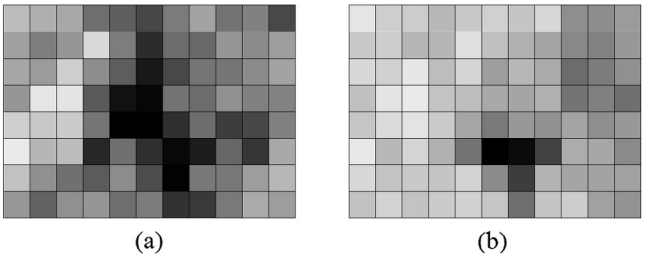

Fig. 17. CRPs for two different input features. The darker and lighter grid elements denote higher and lower values of the weighted codebook values respectively. These symbolize increased and decreased correlation between the respective regions of the mapping and the specific input. (a) CRP of a feature producing high values for the LUAD and LUSC lung carcinoma clusters. (b) CRP that demonstrates a strong association with neurons belonging to the LUSC cluster.

examples of CRPs (Fig. 17) that contain (a) an input that contributes to or correlates with broader cluster regions on the neural map and (b) a feature closely associated with a well-defined cluster. From these it can be concluded that the qualitative information provided by and the functionality of the CRPs are analogous to the component plane visualization of the original SOM model.

As it has been shown the CRP visualization exploits the information existing in the representation planes without providing an insight into the representations' contribution to the final outcome. The Stacked Representation Plane (SRP) visual tool addresses this shortcoming. While the relationship between the input feature and the output neural mapping remains the main priority analytics on the importance and influence of the most prominent representation planes present an additional advantage. As the name suggests, an SRP consists of successive ordered layers that are stacked one on top of the other forming a three-dimensional visual construct. Cubes of appropriate color, transparency and size are the fundamental elements of each layer (i.e., representation plane) of the SRP. The most important representation layers according to each input's weights (w_{mj}, $1 \leq m \leq M$) are positioned lower (or at the bottom), whereas the less important ones occupy the higher levels. Further the w_{mj} parameters are used for weighting the values of each representation plane. The w_{mj} parameters therefore play a dual role. Firstly, they define the ordering of the representation layers and secondly, they adjust the value range of each individual representation plane. For certain input features the outcome of this procedure is the formation of "low clouds" (viz. concentrations of high-valued cubes located around the base of the visualization) that coincide with the previously identified cluster regions. In cases like these one can observe and study not only the correlation between an input feature and a neural mapping but the representations that play an important role in the produced mapping as well. Of even greater analytical significance is the possibility of a visual examination of the underlying mechanism that constructs more complex representations which extend across clusters (Fig. 18).

Characteristic paradigms of SRPs that correspond to the same input features which have been employed for the CRPs (Fig. 17) are presented here (Figs. 18 and 19). We now have an opportunity to unearth the phenomenon of the overlapping LUSC and LUAD clusters that we unsatisfactorily explained using the lower dimension visual analytics. An inspection of the determinant role of the first two (lower) representation layers in Fig. 19 implies that the feature under consideration is closely related to the LUSC cluster. By contrast a low cloud that involves more representation layers containing high values either in the LUAD cluster area or in the LUSC cluster region is seen in Fig. 18. More specifically, the most important representation layer (situated at the bottom) appears highly correlated with the LUAD cluster and the second most important representation layer (which lies exactly above it) closely follows the distribution of the LUSC cluster. What is of significance here is not the previously observed proximity or overlap of the corresponding clusters but the visual verification (through the SRP visual analytics of the DASOM structures) of the richer and more elaborate modeling capabilities achieved by the interjection of a hidden layer between the inputs and the output of the model. The combination of the layered visualization with the DASOM opens areas of data structure analysis that the DA or the SOM independently or disjointly cannot provide. By following a direct unsupervised learning approach the DASOM inferred a higher more abstract representation (i.e. the relation between an input feature and the estimated lung carcinomas' supercluster) based on more basic and elementary representations (i.e. the relations of this input with the LUAD and LUSC subclusters). What this illustrates is that a result that combines the appropriate partials to produce a meaningful whole was achieved without any background information or domain knowledge. We have included animations (Video 1 and Video 2) of the data presented in Figs. 18 and 19 to better provide the reader with perspectives and angles of the respective SRPs that the static presentation lacks.

5. Conclusion

This view that stratification of several levels of nonlinearity is the key for addressing certain weaknesses in machine learning algorithms is supported by a number of findings and advancements that are emerging in the deep learning research field. The general argument is that the use of stratification provides the capability to model complex relationships and discover correlations between variables, tackle difficult recognition tasks, infer higher-level abstract aspects and representations of data, and, in general, mimic and approximate the way human perception and ingenuity function.

Adopting this strategy, the DASOM implements the stratification of layers of nonlinearity to the self-organizing class of algorithms. In comparison to its supervised reference model (i.e. the stacked denoising autoencoder) employing a single hidden layer of neurons could be considered simplistic (or even degenerate). However, this simplification advantages DASOM as it can inherit and extend the visualization and projection virtues of the original SOM. By capturing the characteristic aspects and invariant attributes (of the underlying distributions) of the input data the denoising element seeks representations that are robust and tolerant against corruptions and changes in the input. The autoencoder technique adjusts DASOM's hidden layer which builds upon lower-level distributed partial features and so produces richer descriptions of the data and detects important structures and correlations

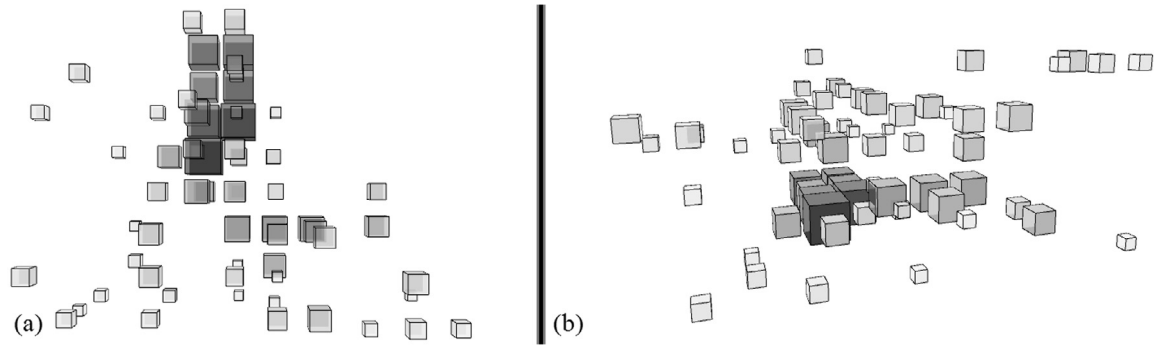

Fig. 18. Different views of an SRP visualization, of the same input feature as that of Fig. 17(a), that demonstrate high values in the lung carcinomas (i.e. LUAD and LUSC) cluster regions. (a) Ground plan and (b) a three-point perspective of the SRP visualization. The color, transparency and size of each cube are defined according to the respective weighted codebook element. Higher values correspond to darker colors, greater opacity and larger sizes.

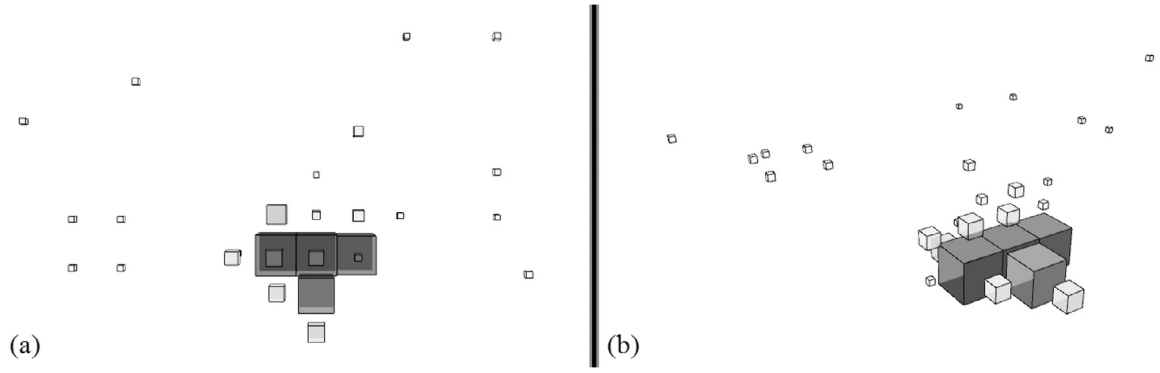

Fig. 19. SRP visualization perspectives corresponding to the same input feature as used in Fig. 17(b). Strong correlation with a specific type of lung carcinoma (i.e. LUSC) can be observed. Higher values correspond to cubes with darker colors, greater opacity and larger sizes, whereas lower values are denoted by lighter color gradually transparent cubes with smaller sizes. (a) Ground plan and (b) a three-point perspective of the SRP visualization.

within them. Finally, at the upper level of the hierarchy a low-dimensional spatially ordered array of output neurons is overlaid above the devised hidden representations to provide topology-driven clustering and visualization.

The DASOM's input space is transformed by the denoising autoencoder component in a space defined by hidden representations with improved and augmented characteristics. We have experimentally verified that this procedure results in increased efficiency and better performance. But perhaps the most significant characteristic of the method is the visualization techniques that can be used as sensors to extract information, gain insight, interpret throughout the DASOM operation and deconstruct the composition of the outputs. Through this a defensible indication of the formation of higher representations and mappings that comprise of low-level partial features and attributes is provided.

Undoubtedly, the next logical step along the current research direction would be the realization of a deep SOM, viz. of an AESOM or a DASOM that integrate several levels or nonlinearity by utilizing successive stacked hidden representation layers. It is our strong belief that a deep SOM of this kind should be complemented with appropriate visualization techniques providing both insight in its hidden layer representations, and, means for interpreting and analyzing its output neural mapping. We are undertaking this work which we will present in a separate publication.

Acknowledgments

This work is based in part upon research supported by the South African Research Chairs Initiative (SARChI) (grant no. 48103) of the Department of Science and Technology (DST) and National Research Foundation (NRF) grant 81025 (K.J.N). C.F. thanks the University of Cape Town for a URC Postdoctoral Research Fellowship.

The authors would like to thank the anonymous reviewers for their constructive comments and insightful remarks.

Appendix A

Initially, we rewrite the reconstruction squared-error objective function as: 0

$$J\left(\mathbf{W}, \mathbf{b}, \hat{\mathbf{b}}\right) = \frac{1}{2}\sum_{h=1}^{L}\left(\hat{x}_h - x_h\right)^2. \tag{A.1}$$

We first calculate the partial derivative of (A.1) with respect to the weight parameter:

$$\frac{\partial J\left(\mathbf{W}, \mathbf{b}, \hat{\mathbf{b}}\right)}{\partial w_{ij}} = \frac{1}{2}\sum_{h=1}^{L}\frac{\partial\left(\hat{x}_h - x_h\right)^2}{\partial w_{ij}} = \sum_{h=1}^{L}\left(\hat{x}_h - x_h\right)\frac{\partial \hat{x}_h}{\partial w_{ij}}. \tag{A.2}$$

By using the chain rule, $\hat{x}_h = f(\hat{z}_h)$ and $\hat{z}_h = \sum_{m=1}^{M} w_{mh} y_m + \hat{b}_h$ in the right hand side we get:

$$\begin{aligned}\frac{\partial J\left(\mathbf{W}, \mathbf{b}, \hat{\mathbf{b}}\right)}{\partial w_{ij}} &= \sum_{h=1}^{L}\left(\hat{x}_h - x_h\right)\frac{\partial \hat{x}_h}{\partial w_{ij}} = \sum_{h=1}^{L}\left(\hat{x}_h - x_h\right)\frac{\partial \hat{x}_h}{\partial \hat{z}_h}\frac{\partial \hat{z}_h}{\partial w_{ij}} \\ &= \sum_{h=1}^{L}\left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right)\frac{\partial \sum_{m=1}^{M} w_{mh} y_m}{\partial w_{ij}}\end{aligned} \tag{A.3}$$

where f' is the total derivative of f. The outer summation can be split in two parts:

$$\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial w_{ij}} = \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right) \frac{\partial \sum_{m=1}^{M} w_{mj} y_m}{\partial w_{ij}} + \sum_{h=1,h\neq j}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \frac{\partial \sum_{m=1}^{M} w_{mh} y_m}{\partial w_{ij}}. \quad \text{(A.4)}$$

The second term contains a w_{ij} parameter in the corresponding y_i activation function, whereas the first term contains a w_{ij} weight also in the summation:

$$\begin{aligned}\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial w_{ij}} &= \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right) y_i \\ &\quad + \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right) \sum_{m=1}^{M} w_{mj} \frac{\partial y_m}{\partial w_{ij}} \\ &\quad + \sum_{h=1,h\neq j}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \sum_{m=1}^{M} w_{mh} \frac{\partial y_m}{\partial w_{ij}} \\ &= \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right) y_i \\ &\quad + \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \sum_{m=1}^{M} w_{mh} \frac{\partial y_m}{\partial w_{ij}}. \end{aligned} \quad \text{(A.5)}$$

Once again by using the chain rule, $y_m = f(z_m)$ and $z_m = \sum_{l=1}^{L} w_{ml}\widetilde{x}_l + b_m$ for the second term in the right hand side we obtain:

$$\begin{aligned}\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial w_{ij}} &= \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right) y_i \\ &\quad + \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \sum_{m=1}^{M} w_{mh} \frac{\partial y_m}{\partial z_m} \frac{\partial z_m}{\partial w_{ij}} \\ &= \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right) y_i + \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \\ &\quad \times \sum_{m=1}^{M} w_{mh} f'\left(z_m\right) \frac{\partial \sum_{l=1}^{L} w_{ml}\widetilde{x}_l}{\partial w_{ij}} \\ &= \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right) y_i \\ &\quad + \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) w_{ih} f'\left(z_i\right) \widetilde{x}_j. \end{aligned} \quad \text{(A.6)}$$

In a similar way we derive the partial derivative of (A.1) with respect to the hidden layer bias parameter:

$$\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial b_i} = \frac{1}{2}\sum_{h=1}^{L} \frac{\partial\left(\hat{x}_h - x_h\right)^2}{\partial b_i} = \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) \frac{\partial \hat{x}_h}{\partial b_i}. \quad \text{(A.7)}$$

The chain rule in combination with $\hat{x}_h = f(\hat{z}_h)$ and $\hat{z}_h = \sum_{m=1}^{M} w_{mh} y_m + \hat{b}_h$ yields:

$$\begin{aligned}\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial b_i} &= \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) \frac{\partial \hat{x}_h}{\partial \hat{z}_h} \frac{\partial \hat{z}_h}{\partial b_i} \\ &= \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \sum_{m=1}^{M} w_{mh} \frac{\partial y_m}{\partial b_i}. \end{aligned} \quad \text{(A.8)}$$

Likewise, by incorporating $y_m = f(z_m)$ and $z_m = \sum_{l=1}^{L} w_{ml}\widetilde{x}_l + b_m$ we get:

$$\begin{aligned}\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial b_i} &= \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \sum_{m=1}^{M} w_{mh} \frac{\partial y_m}{\partial z_m} \frac{\partial z_m}{\partial b_i} \\ &= \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \\ &\quad \times \sum_{m=1}^{M} w_{mh} f'\left(z_m\right) \frac{\partial\left(\sum_{l=1}^{L} w_{ml}\widetilde{x}_l + b_m\right)}{\partial b_i} \\ &= \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) w_{ih} f'\left(z_i\right). \end{aligned} \quad \text{(A.9)}$$

Last, the partial derivative of (A.1) with respect to the output layer offset parameter is given by:

$$\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial b_j} = \frac{1}{2}\sum_{h=1}^{L} \frac{\partial\left(\hat{x}_h - x_h\right)^2}{\partial \hat{b}_j} = \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) \frac{\partial \hat{x}_h}{\partial \hat{b}_j}. \quad \text{(A.10)}$$

After replacing the expressions $\hat{x}_h = f(\hat{z}_h)$ and $\hat{z}_h = \sum_{m=1}^{M} w_{mh} y_m + \hat{b}_h$ we obtain:

$$\begin{aligned}\frac{\partial J\left(\mathbf{W},\mathbf{b},\hat{\mathbf{b}}\right)}{\partial \hat{b}_j} &= \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) \frac{\partial \hat{x}_h}{\partial \hat{z}_h} \frac{\partial \hat{z}_h}{\partial \hat{b}_j} \\ &= \sum_{h=1}^{L} \left(\hat{x}_h - x_h\right) f'\left(\hat{z}_h\right) \frac{\partial\left(\sum_{m=1}^{M} w_{mh} y_m + \hat{b}_h\right)}{\partial \hat{b}_j} \\ &= \left(\hat{x}_j - x_j\right) f'\left(\hat{z}_j\right). \end{aligned} \quad \text{(A.11)}$$

For ease of reference we reiterate the alternative expression of the cost/energy function:

$$\begin{aligned}K\left(\mathbf{W},\mathbf{b},\mathbf{U}\right) &= \sum_{c=1}^{E} N\left(\mathbf{y},c\right) \sum_{e=1}^{E} h_{ce} \frac{1}{2} \left\|\mathbf{y} - \mathbf{u}_e\right\|^2 \\ &= \frac{1}{2}\sum_{c=1}^{E} N\left(\mathbf{y},c\right) \sum_{e=1}^{E} h_{ce} \sum_{m=1}^{M} (y_m - u_{me})^2. \end{aligned} \quad \text{(A.12)}$$

The first task is estimating the partial derivative of (A.12) with respect to the weight parameter:

$$\begin{aligned}\frac{\partial K\left(\mathbf{W},\mathbf{b},\mathbf{U}\right)}{\partial w_{ij}} &= \frac{1}{2}\sum_{c=1}^{E} \frac{\partial N\left(\mathbf{y},c\right)}{\partial w_{ij}} \sum_{e=1}^{E} h_{ce} \sum_{m=1}^{M} (y_m - u_{me})^2 \\ &\quad + \frac{1}{2}\sum_{c=1}^{E} N\left(\mathbf{y},c\right) \sum_{e=1}^{E} \frac{\partial h_{ce}}{\partial w_{ij}} \sum_{m=1}^{M} (y_m - u_{me})^2 \\ &\quad + \frac{1}{2}\sum_{c=1}^{E} N\left(\mathbf{y},c\right) \sum_{e=1}^{E} h_{ce} \sum_{m=1}^{M} \frac{\partial (y_m - u_{me})^2}{\partial w_{ij}}. \end{aligned} \quad \text{(A.13)}$$

The first term in the right hand side vanishes, refer to Hammer, Micheli, Sperduti, and Strickert (2004) for a detailed justification. The second term also vanishes because h_{ce} is a function of the neurons' geometrical coordinates on the lattice.

$$\frac{\partial K\left(\mathbf{W},\mathbf{b},\mathbf{U}\right)}{\partial w_{ij}} = \sum_{c=1}^{E} N\left(\mathbf{y},c\right) \sum_{e=1}^{E} h_{ce} \sum_{m=1}^{M} (y_m - u_{me}) \frac{\partial y_m}{\partial w_{ij}}. \quad \text{(A.14)}$$

After replacing $y_m = f(z_m)$ and $z_m = \sum_{l=1}^{L} w_{ml} x_l + b_m$ we get:

$$\frac{\partial K(\mathbf{W}, \mathbf{b}, \mathbf{U})}{\partial w_{ij}} = \sum_{c=1}^{E} N(\mathbf{y}, c) \sum_{e=1}^{E} h_{ce} \sum_{m=1}^{M} (y_m - u_{me}) f'(z_m) \times \frac{\partial \sum_{l=1}^{L} w_{ml} x_l}{\partial w_{ij}} = \sum_{c=1}^{E} N(\mathbf{y}, c) \sum_{e=1}^{E} h_{ce} (y_i - u_{ie}) f'(z_i) x_j. \quad (A.15)$$

By following once again the previous steps the expression for the derivative with respect to the bias parameter equals:

$$\frac{\partial K(\mathbf{W}, \mathbf{b}, \mathbf{U})}{\partial b_i} = \sum_{c=1}^{E} N(\mathbf{y}, c) \sum_{e=1}^{E} h_{ce} \sum_{m=1}^{M} (y_m - u_{me}) f'(z_m) \times \frac{\partial \left(\sum_{l=1}^{L} w_{ml} x_l + b_m \right)}{\partial b_i} = \sum_{c=1}^{E} N(\mathbf{y}, c) \sum_{e=1}^{E} h_{ce} (y_i - u_{ie}) f'(z_i). \quad (A.16)$$

In a similar manner, the calculations for the codebook vectors' parameters yield:

$$\frac{\partial K(\mathbf{W}, \mathbf{b}, \mathbf{U})}{\partial u_{ij}} = -\sum_{c=1}^{E} N(\mathbf{y}, c) \sum_{e=1}^{E} h_{ce} \sum_{m=1}^{M} (y_m - u_{me}) \frac{\partial u_{me}}{\partial u_{ij}} = \sum_{c=1}^{E} N(\mathbf{y}, c) h_{cj} (u_{ij} - y_i). \quad (A.17)$$

It is important to point out that in all the cost/energy function derivatives we use the actual feature values x_l instead of the corrupted ones $\widetilde{x}_l$. In addition, it should be noted that the $N(\mathbf{y}, c)$ coefficient yields the winner neuron definition.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2018.04.016.

References

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 1027–1035). Society for Industrial and Applied Mathematics.

Bache, K., & Lichman, M. (2013). *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science, [http://archive.ics.uci.edu/ml].

Becker, S. (1991). Unsupervised learning procedures for neural networks, *International Journal of Neural Systems*, *2*, 17–33.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundation and Trends® in Machine Learning*, *2*, 1–127.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, *19*, 153.

Berglund, E., & Sitte, J. (2006). The parameterless self-organizing map algorithm. *IEEE Transactions on Neural Networks*, *17*, 305–316.

Brugger, D., Bogdan, M., & Rosenstiel, W. (2008). Automatic cluster detection in Kohonen's SOM. *IEEE Transactions on Neural Networks*, *19*, 442–459.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, *11*, 625–660.

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., & Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS, Vol. 5* (pp. 153–160).

Hammer, B., Micheli, A., Sperduti, A., & Strickert, M. (2004). A general framework for unsupervised processing of structured data. *Neurocomputing*, *57*, 3–35.

Heskes, T. (1999). Energy functions for self-organizing maps. In E. Oja, & S. Kaski (Eds.), *Kohonen maps* (pp. 303–315). Amsterdam: Elsevier Science B.V..

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*, 550–554.

Ito, M., & Komatsu, H. (2004). Representation of angles embedded within contour stimuli in area V2 of macaque monkeys. *The Journal of Neuroscience*, *24*, 3313–3324.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*, 651–666.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys (CSUR)*, *31*, 264–323.

Kohonen, T. (2001). *Springer series in information sciences*: *vol. 30. Self-organizing maps*. ed: Springer Berlin.

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Networks*, *37*, 52–65.

Kohonen, T. (2014). *MATLAB implementations and applications of the self-organizing map*. (pp. 11–23). Helsinki, Finland: Unigrafia Oy.

Lee, H., Ekanadham, C., & Ng, A. Y. (2008). Sparse deep belief net model for visual area V2. *Advances in Neural Information Processing Systems*, 873–880.

Lee, T. S., & Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, *20*, 1434–1448.

Lee, T. S., Mumford, D., Romero, R., & Lamme, V. A. (1998). The role of the primary visual cortex in higher level vision. *Vision Research*, *38*, 2429–2454.

Nene, S. A., Nayar, S. K., & Murase, H. (1996). Columbia object image library (COIL-20). In: Technical report CUCS-005-96.

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*, 267–273.

Oja, E. (1992). Principal components, minor components, and linear neural networks. *Neural Networks*, *5*, 927–935.

Pampalk, E., Rauber, A., & Merkl, D. (2002). Using smoothed data histograms for cluster visualization in self-organizing maps. In *International conference on artificial neural networks* (pp. 871–876). Springer.

Pölzlbauer, G., Dittenbach, M., & Rauber, A. (2006). Advanced visualization of self-organizing maps with vector fields. *Neural Networks*, *19*, 911–922.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Shah-Hosseini, H., & Safabakhsh, R. (2003). TASOM: a new time adaptive self-organizing map. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *33*, 271–282.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research (JMLR)*, *3*, 583–617.

Tasdemir, K., & Merényi, E. (2009). Exploiting data topology in visualization and clustering of self-organizing maps. *IEEE Transactions on Neural Networks*, *20*, 549–562.

Ultsch, A. (2003). Maps for the visualization of high-dimensional data spaces. In: *Proc. Workshop on self organizing maps*, (pp. 225–230).

Ultsch, A. (2005). Clustering wih som: U* c. In: *Proceedings of the 5th workshop on self-organizing maps, Vol. 2*, (pp. 75–82).

Vesanto, J. (1999). SOM-based data visualization methods. *Intelligent Data Analysis*, *3*, 111–126.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on machine learning* (pp. 1096–1103). ACM.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*, *11*, 3371–3408.

Von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, *14*, 85–100.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, *16*, 645–678.

Yin, H. (2002). ViSOM-a novel method for multivariate data projection and structure visualization. *IEEE Transactions on Neural Networks*, *13*, 237–243.