Machine Learning (2025) 114:125
https://doi.org/10.1007/s10994-024-06715-8

Online dimensionality reduction through stacked generalization of spectral methods with deep networks

Juan Carlos Alvarado-Pérez[1,2] · **Miguel Angel Garcia**[3] · **Domènec Puig**[1]

Received: 10 February 2024 / Revised: 26 July 2024 / Accepted: 20 October 2024 /
Published online: 25 March 2025

Abstract
Analyzing large volumes of high-dimensional data poses significant challenges. Dimensionality reduction aims to reveal the most prominent properties of data by embedding them into a low-dimensional representation. Spectral dimensionality reduction methods using kernel matrices have been proven to yield optimal results. Online versions of those methods are desirable to incrementally project new data without recomputing the whole embedding from the complete dataset. In addition, integrating different spectral methods may have a synergistic effect. This paper presents an online dimensionality reduction method based on deep neural networks that integrates embeddings optimized by statistical approximation of neighborhoods and induced by different spectral methods through stacking ensemble learning. In particular, the proposed method first applies a self-supervised stage in order to train a set of deep encoders based on the embeddings induced by different spectral methods applied to a given input dataset. Those basis encoders are optimized and then integrated through a metamodel constituted by a fully connected network. A supervised and an unsupervised approach have been designed depending on whether the final aim is to enforce topological preservation or cluster induction. The proposed method has been experimentally validated on well-known image datasets and compared to some of the most relevant dimensionality reduction techniques by using widely-used quality measures.

Keywords Online dimensionality reduction · Spectral methods · Stacking · Deep networks · Topological preservation · Cluster induction · Manifold approximation

1 Introduction

Processing huge volumes of data has consolidated the field of *big data*. These data can be inconsistent and unpredictable as they often emerge from chaotic phenomena and possess a complex structure. Complexity here implies high dimensionality. The latter is interpreted as the number of variables, features or attributes that characterize an object.

Editor: Qibin Zhao.

Miguel Angel Garcia and Domènec Puig have contributed equally to this work.

Extended author information available on the last page of the article

That complexity becomes a challenge for human understanding and perception, as well as for computational processing in data analysis areas, such as data mining, machine learning, pattern recognition or information visualization.

Dimensionality reduction (DR) (Sarveniazi, 2014) is a feasible and efficient solution to the problems associated with high dimensionality. It aims to reveal the most outstanding and relevant properties of large volumes of data, and ultimately to understand the complexity inherent to a multidimensional and chaotic world. Dimensionality can be reduced either by selecting subsets of features (i.e., dimensions) or by projecting the original features into a low-dimensional space. *Feature projection* methods can be broadly classified into linear and non-linear approaches. Linear approaches are based on criteria related to distance, similarity or divergence. In turn, non-linear approaches, also known as manifold learning methods, use predefined structures, such as adjacency graphs, statistical methods or neural networks.

Spectral methods are a subset of DR feature projection methods characterized by applying eigendecomposition to specially defined matrices in order to determine embeddings that aim to preserve the topology of the original data. A big advantage of spectral methods is that, once the eigendecomposition has been made, the number of dimensions of the projected space is simply determined by the number of top or bottom eigenvectors chosen. These methods include *Principal Component Analysis* (PCA) (Hotelling, 1933), *Classical Multidimensional Scaling* (CMDS) (Borg & Groenen, 2005), *Laplacian Eigenmaps* (LE) (Belkin & Niyogi, 2003), and *Locally Linear Embedding* (LLE) (Roweis & Saul, 2000), among others. PCA and CMDS are linear methods whose embeddings take into account all the given data points (global DR methods), whereas LE and LLE are non-linear methods that consider the neighborhoods of the given points (local DR methods). While global methods yield embeddings that capture the overall structure of the original data sets even preserving distance metrics, local methods better capture the topology of the underlying manifolds and are more computationally efficient by using sparse matrix operations.

The ultimate goal of dimensionality reduction is the preservation in the projected low-dimensional space of the intrinsic data properties fulfilled in the original high-dimensional space, such as cluster distribution or topological manifold structure (i.e., manifold unfolding). However, linear DR methods cannot preserve those properties in case of intricate (non-linear) clusters and manifolds. This limitation may be overcome by applying *kernel methods*, which implicitly project the original data to an even higher dimensional space in which non-linearities can be linearized. Thus, kernel versions of the main DR spectral methods have been proposed with success for linear approaches, such as *Kernel PCA* (KPCA) and *Kernel CMDS* (KCMDS), and even for non-linear schemes, such as *Kernel LE* (KLE) and *Kernel LLE* (KLLE) (Ham et al., 2004; Lee & Verleysen, 2007).

The spectral methods described above are categorized as batch (offline) processing since they require the entire data set for computing the sought embedding. Therefore, they cannot be applied if the whole data set does not fit in main memory. In addition, with the exception of PCA, if new data instances need to be projected, the whole eigendecomposition and its associated matrix must be computed from scratch, which is highly inefficient. In those cases, it is necessary to apply *online processing* (also known as incremental or adaptive processing) that works on a per-instance basis or in mini-batches. Thus, online versions of some DR spectral methods have been proposed in the literature, such as *Incremental PCA* (Zhao et al., 2006), *Incremental KPCA* (Chin & Suter, 2007; Hallgren & Northrop, 2018), *Incremental CMDS* (Basalaj, 1999), *Incremental LLE* (Kouropteva et al., 2005) and *Incremental LE* (Jia et al., 2009), among others.

Since kernel-based spectral methods are based on the eigendecomposition of kernel matrices, it is possible to integrate different methods by combining their associated matrices. The final embedding generated after that integration may thus exhibit properties of the precursor methods. For example, it is possible to combine local and global DR methods in order to inherit beneficial characteristics from both approaches (Gou et al., 2020). This is the foundation of *Multiple Kernel Learning* (MKL) (Gönen & Alpaydın, 2011), in which kernel matrices are averaged with weights that can be learnt as a result of an optimization process.

A well-known paradigm for combining machine learning algorithms is *ensemble learning* (Wang et al., 2018; Denisko & Hoffman, 2018). Its goal is to yield an integrated algorithm that outperforms its constituent learning algorithms. *Stacked generalization* (Wolpert, 1992), also known as *stacking*, is a type of ensemble learning in which a so-called *metamodel* is trained to combine the outputs of a set of heterogeneous learning algorithms pre-trained independently.

This paper proposes an online DR scheme that synergistically combines classical spectral methods with modern deep neural networks. Given a target dataset whose dimensionality is to be reduced, different linear and non-linear DR spectral methods are first applied. The original data points are then projected to the sought low-dimensional space according to the embedding induced by every spectral method. The set of projected points generated by each spectral method along with the original data points are utilized to train a deep encoder (Hinton & Salakhutdinov, 2006) in a self-supervised way. Thus, each encoder learns the embedding induced by its associated spectral method and is able to project new points in an online manner, something that classical spectral methods cannot perform due to their offline nature. A stacking ensemble learning approach is then applied for fusing the low-dimensional outputs of the different encoders through a metamodel implemented as a multilayer fully-connected network.

We propose two metamodel topologies depending on whether the final aim is to enforce topological preservation (Luo et al., 2021; Alswaitti et al., 2022; Jiang et al., 2023) or cluster induction (Guo et al., 2022; Eskandarnia et al., 2022; Wu & Noels, 2022). For topological preservation, the metamodel is a shallow network trained in an unsupervised manner using a graph that represents high-dimensional neighborhoods. In turn, Euclidean distance matrices in high and low dimensions are also used for preserving the global structure of the input data. For cluster induction, the metamodel is a supervised deep network trained by means of a deep *autoencoder* (Hinton & Salakhutdinov, 2006). A deep decoder aims to approximate a one-hot matrix generated from the labels of the different classes by minimizing a cross-entropy cost function.

The proposed method has been validated on seven widely-known image datasets: *Fashion-MNIST*, *COIL-20*, *LEGO*, *Medical-MNIST*, *Imagenette*, *ISL* and *USPS*. Standard implementations of PCA, CMDS, LLE and LE, of their corresponding kernel versions KPCA, KCMDS, KLLE and KLE, and of combinations of them through Multiple Kernel Learning(MKL) and DeepMKL have been considered as precursor spectral methods.

The following dimensionality reduction methods have also been considered in the experimental validation: Linear Discriminant Analysis (LDA), Factor Analysis (FA), GraphEncoder, ScaledPCA, Uniform Manifold Approximation and Projection (UMAP), ISOMAP, Large-scale Dimensionality Reduction Using Triplets (TriMap), DensMap, SliseMap and ParametricUMAP.

Two quality criteria have been used to measure the performance of DR techniques: topological preservation and cluster induction. Topological preservation has been evaluated by means of the $R_{NX}(K)$ curves (Lee et al., 2013), whereas cluster induction has been

evaluated though the homogeneity, completeness and *V*-measure scores (Rosenberg & Hirschberg, 2007).

The rest of this paper is organized as follows: Sect. 3 presents an overview of the main concepts related to classical DR methods, their kernel and online versions, as well as modern neural network approaches. Section 4 discusses the combination of DR methods using MKL and ensemble learning. In Sect. 5, the proposed method, referred to as *NetDR*, is described, including its constituent stages. Sections 6 and 7 present the experimental setup and the obtained results, respectively. Sect. 7.3.6 presents the computational complexity of the proposed methods. Section 2 presents a discussion of the relevance of the method and future work. Finally, conclusions are given in Sect. 8.

2 Relevance of the method and future work

3 Dimensionality reduction

Dimensionality reduction aims at extracting relevant information from high-dimensional data by mapping them into a low-dimensional space. Its application is beneficial for multiple reasons. Firstly, it helps improve the computational performance of learning algorithms by processing fewer dimensions. In addition, overfitting is reduced by decreasing the input dimension. Simpler and more general models are thus learn. It also allows visualizing multidimensional data in an intelligible way by mapping them into 2D or 3D space. It allows topological preservation of intrinsic data structures: nearby points in the original space remain close in the embedded one, whereas distant points remain far away. It allows linear separation of features in the high-dimensional space by minimizing the overlap of points in the projected space, which induces data clusters. Finally, it minimizes noise by suppressing redundant information.

A wide variety of DR methods have been proposed in the literature based on different principles. The following subsections summarize the most representative approaches including classical methods, kernel-based adaptations, online versions, and concluding with modern schemes based on neural networks.

3.1 Classical methods

Dimensionality reduction methods have traditionally followed two approaches: feature selection and feature projection (also known as feature extraction). The goal of feature selection is determining a subset of the original dimensions that optimizes the performance of a certain data analysis algorithm (classification, regression, etc.). This can be done by following either filter, wrapper or embedded strategies (Molina et al., 2002). However, suppressing dimensions may not preserve the underlying structures of the input data in general. The alternative is feature projection, which maps the original dimensions into a lower-dimensional space.

The best-known linear feature projection method is Principal Component Analysis (PCA) (Hotelling, 1933). It maximizes the data variance in the low-dimensional space. Factor Analysis (FA) (Spearman, 1904) is a probabilistic version of PCA that aims to extract a set of d unobserved variables (dimensions), called factors, such that the original

D dimensions can be expressed as a linear combination of those d factors plus a Gaussian noise term, with $d << D$. Every factor represents a group of original dimensions that are highly correlated among them. An iterative implementation of FA using singular value decomposition described in Barber (2012) has been used in this work. In turn, Classical Multidimensional Scaling (CMDS) (Borg & Groenen, 2005) aims at preserving in the projected space the pairwise Euclidean distances among the original data points. ISOMAP (Tenenbaum et al., 2000; Qu et al., 2021) applies CMDS to the geodesic distances between all pairs of points. Geodesic distances are estimated through the shortest path distances between the points represented in a graph of nearest neighbors. Similarly, Stochastic Neighbor Embedding (SNE) (Hinton & Roweis, 2002) preserves pairwise dissimilarities among points. The dissimilarity between two points is defined as their normalized Euclidean distance. Alternatively, when the original data points belong to known classes, *Linear Discriminant Analysis* (LDA) (Izenman, 2013) determines the linear combination of the original dimensions that maximizes inter-class distances while minimizing intra-class distances (Fisher's criterion). LDA is thus a supervised method.

The previous feature projection methods are global as their respective embeddings are generated by considering global properties of all given data points. In many applications, however, it is necessary to reduce the high dimensionality while keeping the structure of the topological manifolds present in the original data sets. This is referred to as manifold learning. That goal can be attained by preserving the local neighborhood of every point in the projected space, which is the principle behind non-linear local feature projection methods, such as Locally Linear Embedding (LLE) (Roweis & Saul, 2000) and Laplacian Eigenmaps (LE) (Belkin & Niyogi, 2003). LLE first determines the k nearest neighbors of every original point and then yields an embedding that preserves the barycentric coordinates of every point with respect to its neighbors. In turn, LE first computes a weighted adjacency graph with as many nodes as original points. Two points are adjacent either if their distance is below a threshold or if one is a k-nearest neighbor of the other. The edge's weight for adjacent points can be either one or their Gaussian radial basis function, and zero for non-adjacent points. The embedding is obtained by computing the eigenvectors of the graph's Laplacian.

3.2 Kernel methods

Linear DR methods are prone to overfitting or underfitting. In addition, real-world data sets do not always exhibit a linear behavior. In turn, non-linear DR methods are computationally intensive on bulky sets involving a large number of dimensions. Furthermore, they often apply convex optimization procedures that may yield suboptimal solutions when falling into local extrema.

Kernel methods overcome the aforementioned disadvantages of both linear and non-linear methods by taking the best of both worlds. In essence, they first map the given data from their original space $\mathcal{X}$ to a very high-dimensional space $\mathcal{V}$, even to an infinite-dimensional Hilbert space (Yuan et al., 2010), where the data points are easily separable. Those mapped data are then projected to the sought low-dimensional space. This allows intricate non-linear data structures to be preserved while applying linear methods.

Fortunately, dealing with such very high dimensional spaces is very efficient thanks to the *kernel trick*. Basically, no explicit mapping $\varphi : \mathcal{X} \rightarrow \mathcal{V}$ is ever done. Instead, the inner product of any pair of data points $\{\mathbf{x}_i, \mathbf{x}_j\} \in \mathcal{X}$ mapped to $\mathcal{V}$ is equivalent to the application of a *kernel function* k to those points: $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle$. The *Mercer's theorem*

(Carmeli et al., 2006) determines which functions can be used as kernels (polynomial, radial basis function, etc.). Given N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathcal{X}$, kernel methods usually represent all pairwise applications of the kernel function k as a *kernel matrix* $\mathbf{K} \in \mathbb{R}^{N \times N}$, such that $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Once $\mathbf{K}$ has been determined, subsequent processing can be kept linear. In this way, kernel methods are able to deal with non-linear data structures in the original space $\mathcal{X}$ by applying linear processing in the very high-dimensional space $\mathcal{V}$.

KPCA (Schölkopf et al., 1998) is a kernelized version of PCA that benefits from the above principles. Based on it, kernel-based versions of other DR spectral methods, such as KCMDS, KLLE and KLE, can be obtained by following two steps. First, a kernel matrix $\mathbf{K}$ is obtained from the definition of the target spectral method. The projection to the sought low-dimensional space is then performed by applying KPCA to $\mathbf{K}$ (Ham et al., 2004; Belanche Muñoz, 2013; Ham et al., 2004).

3.3 Online methods

In general, data sets can be processed by following either an offline (also called discontinuous) or online (also called continuous, incremental or adaptive) approach. Offline processing requires the availability of the whole data set (full batch). Unfortunately, huge datasets may overflow the storage capacity of main computer memory (Fresca & Manzoni, 2022). Furthermore, data may change over time in some applications, hence requiring processing algorithms that continuously adapt to those variations of the data set. Online processing is necessary when a complete data set cannot be processed as a whole or it is directly unavailable. Data points are processed either one at a time or grouped in *mini-batches* (Joshi & Kulkarni, 2012). However, online methods also have drawbacks: their result may depend on the sampling order of the individual points or the mini-batches, and the optimization procedures they may apply can fall into local extrema, not guaranteeing optimal solutions. The latter problem can be tackled by applying stochastic approximation methods, such as the Robbins-Monro algorithm (Robbins & Monro, 1951).

The DR methods summarized above are offline and require the full data set to generate an embedding. If new data points are included, the whole process must be run from scratch. Moreover, limited main memory capacities may prevent their application. Therefore, online versions of those methods have been devised.

In particular, various online versions of PCA, generally referred to as Incremental PCA (IPCA) have been proposed. Some approaches reconstruct the significant principal components from the original data set and every new added point, such as Skocaj and Leonardis (2003). The updated eigenspace is generated by using coefficient vectors from the original data. However, these approaches may yield unbounded approximation errors. Other approaches compute the principal components without estimating the covariance matrix, such as CCIPCA (Weng et al., (2003). In the latter, however, the approximation error can be accumulated as each principal component is computed from a previous one. Alternatively, SVDU-IPCA (Zhao et al., 2006) is an online PCA method based on a singular value decomposition updating algorithm. It is advantageous as it yields a bounded approximation error and can be easily extended to KPCA.

Similarly, there are also online versions of LLE, LE and CMDS, such as Incremental Locally Linear Embedding (ILLE) (Kouropteva et al., 2005), Incremental Laplacian Eigenmaps (ILE) (Jia et al., 2009), and Incremental Multidimensional Scaling (IMDS) (Basalaj, 1999), respectively. Different online versions of KPCA have also been proposed, such as Chin and Suter (2007), Hallgren and Northrop (2018).

3.4 Manifold approximation and projection methods

Some methods attempt to model the data intrinsic structure by mapping the high-dimensional data into an embedding through projections or approximations in a way that preserves the topological structure, as well as the local and global relationships among the original data points.

In contrast to traditional PCA, which seeks to maximize the variation between components, ScaledPCA (Huang et al., 2022) is a supervised DR method that improves the process by assigning higher weights to components that have greater predictive ability. It does this by scaling each predictor based on its ability to predict the desired objective.

UMAP (Uniform Manifold Approximation and Projection for Dimension Reduction) (McInnes et al., 2018) models the high-dimensional space by preserving the fuzzy topological structure both locally and globally. The data points are projected into an embedding that maintains the characteristics of their structure in the low dimension. In turn, ParametricUMAP (Sainburg et al., 2021; Xu & Zhang, 2023) is a semi-supervised DR algorithm that extends the UMAP model. Instead of establishing the low-dimensional space directly, the embedding is generated through a neural network that learns the relationships between the high-dimensional data previously established in a graph. This is done by optimizing the weights of a batch-trained deep neural network, thus learning the parametric relationship between the data and the embedding.

Another variety of UMAP is DensMap (Narayan et al., 2020). It locally preserves the relative density of data points and uses this estimate as a regularizer in the embedding optimization. TriMap (Amid & Warmuth, 2019) is a semi-supervised DR method that focuses on preserving the global structure by introducing a global score. The latter quantifies the quality of a low-dimensional embedding of the data by means of triplets, calculating a linear inverse optimal mapping from an initial mapping generated by PCA. SliseMap (Björklund et al., 2023) is a supervised visualization method based on dimensionality reduction that simultaneously maintains local relationships between the data and preserves the global structure of the high-dimensional space. Similarly to TriMap, SliseMap also performs an initial mapping with PCA.

3.5 Neural networks

Unsupervised neural networks have also been successfully applied to dimensionality reduction. One of the best-known schemes is *Self-Organizing Maps* (SOM) (Kohonen, 2001), a non-linear approach that yields a low-dimensional (typically two-dimensional) representation of high-dimensional data while preserving their topological structure. The network is usually defined as a rectangular or hexagonal grid of nodes (neurons). Every node is associated with a weight vector defined in the high-dimensional space. All vectors can be initialized at random or uniformly sampled from the hyperplane spanned by the two largest principal components of the data set. A learning algorithm iteratively maps each dataset point to the grid's node with the closest weight vector. The weight vector of the selected node is updated with a running average in order to move it closer to the point coordinates. Likewise, the weight vectors of the node's neighbors are also updated with a Gaussian decay dependant on their grid distance to the selected node. As a result of this competitive learning process, the network associates groups of

nodes with clusters present in the high-dimensional data set. In the end, every data point is mapped to the node with the closest weight.

Alternatively, *Generative Topographic Mapping* (GTM) (Bishop et al., 1998) is a probabilistic counterpart of SOM. It also assumes a grid of nodes. Every node is associated with a multivariate Gaussian that models a probability distribution of points in the high-dimensional space. The Gaussians of all nodes constitute a mixture model that is fitted to the given data set by applying the Expectation-Maximization algorithm. Every original data point is finally mapped to the node whose Gaussian has the maximum likelihood.

Isotop (Lee et al., 2003) is a DR neural method that overcomes the main disadvantage of both SOM and GTM: their rectangular grid of nodes may not suit the shape of the manifold lying in the original data set. This problem is tackled by generating an adjacency graph of nodes. Every node is an original data point and is connected to its k closest neighbors. If the data set is very large, the nodes can be reduced by applying vector quantization. The generated graph captures the local structure of the original data similarly to LE and LLE. The high-dimensional vector of every node is then replaced by a low-dimensional vector randomly set around zero, referred to as a prototype. A unit variance Gaussian is centered at each prototype. Then, a learning algorithm iteratively draws a random point from that mixture of Gaussians and finds its closest prototype. The coordinates of the latter and its closest neighbors are updated with a running average similarly to SOM. The final embedding is constituted by the low-dimensional coordinates of the prototypes.

Another example in this category is Deep Isometric Maps (Pai et al., 2022), an unsupervised deep learning approach for computing distance-preserving maps for non-linear dimensionality reduction, managing to improve the local and non-local generalization of the isometric mapping.

Nowadays, the impressive evolution of deep neural networks has also benefitted the field of dimensionality reduction by using *autoencoders* (AE) (Hinton & Salakhutdinov, 2006; Shinde et al., 2023). An autoencoder is generated by concatenating two deep networks: an *encoder* network that maps a high-dimensional vector $\mathbf{X} \in \mathbb{R}^D$ into a low-dimensional encoding $\mathbf{Y} \in \mathbb{R}^d$, followed by a *decoder* network that maps this encoding $\mathbf{Y}$ back into a high-dimensional vector $\widehat{\mathbf{X}} \in \mathbb{R}^D$. Both networks are trained end-to-end in an unsupervised way by applying backpropagation in order to minimize the error between $\mathbf{X}$ and $\widehat{\mathbf{X}}$. Once the autoencoder has been trained, the DR method is the encoder network alone.

There are three main types of autoencoders. In the simplest case, both the encoder and decoder are fully-connected multilayer networks. A second type is *variational autoencoders* (VAE) (Kingma and Welling, 2013). In this case, the encoder network does not generate a single encoded vector $\mathbf{Y} \in \mathbb{R}^d$, but a set of d Gaussian functions defined by a mean and a standard deviation. They constitute a multivariate Gaussian that models the probability distribution of the encoded space (usually referred to as *latent space*). Since the decoder must be fed with a single point, a d-dimensional vector $\mathbf{Y}$ is randomly drawn from that multivariate Gaussian (this is known as the *reparameterization trick*). The third type is *convolutional autoencoders* (CAE) (Masci et al., 2011). In this case, both the encoder and decoder are *convolutional neural networks* (CNN). CAEs yield the best results for reducing the dimensionality of images. Multiple variations and combinations of these basic types have been proposed in the literature.

GraphEncoder (Tian et al., 2014; Luo et al., 2021) is an autoencoder-based graph clustering model. It consists of a deep neural network with a sparse autoencoder as its building block. The input graph can be the same weighted adjacency graph with as many nodes as original points defined for Laplacian Eigenmaps. The graph's similarity matrix is the

training set of the neural network. The graph embedding consists of the output features in the deepest layer of the deep network.

Finally, supervised deep networks have also been proposed for dimensionality reduction. They project the high-dimensional space into a low-dimensional one and take advantage of the class labels of the training samples to optimize that mapping. A good example is MMINet (Özdenizci & Erdoğmuş, 2021), which projects high-dimensional features into an output feature space in which lower dimensional representations of features carry maximum mutual information with their associated class labels. MMINet uses a training procedure based on the stochastic estimate of the mutual information gradient.

4 Combined methods

A wide variety of DR methods have been proposed in the literature as summarized in the previous sections. Their different operation principles enforce specific properties in the embeddings they yield, among which topological preservation and cluster induction are the most relevant. *Topological preservation* (Lee & Verleysen, 2007) attempts to preserve neighborhoods by minimizing both extrusions (nearby points in a neighborhood moving away) and intrusions (distant points in a neighborhood moving closer). In turn, *cluster induction* (Blockeel et al., 2000; Chavula & Suleman, 2017) aims at generating dense non-overlapped groupings of data that satisfy certain inter-class and intra-class features (Özdenizci & Erdoğmuş, 2021). However, both properties tend to be mutually exclusive. For instance, cluster induction typically minimizes topological preservation, since inducing a cluster usually involves generating intrusions or extrusions that affect the embedding continuity.

If different DR methods are synergistically combined, the result may yield a rich embedding that preserves their respective properties (Menaga & Revathi, 2021). The two main approaches for combining DR methods are summarized below.

4.1 Multiple Kernel Learning

According to Mercer's theorem (Carmeli et al., 2006), a kernel is a symmetric positive semi-definite function. It is possible to combine different kernel functions provided the resulting function also satisfies that condition. In particular, the addition of kernels also yields a kernel. This is the basic principle behind Multiple Kernel Learning (MKL) (Gönen & Alpaydın, 2011), which is the main approach for combining kernelized versions of DR spectral methods, such as KPCA, KCMDS, KLLE and KLE. In particular, MKL combines the kernel matrices corresponding to M methods, $\{\mathbf{K}_1, \ldots, \mathbf{K}_M\}$, yielding an integrated kernel matrix $\mathbf{K}$ defined as a weighted average:

$$\mathbf{K} = \sum_{m=1}^{M} \alpha_m \mathbf{K}_m, \tag{1}$$

where α_m is a weighting factor associated with the m-th method. Once $\mathbf{K}$ has been obtained, the dimensionality is reduced by applying KPCA (Belanche Muñoz, 2013).

The above weighting factors corresponding to the fused kernel methods are usually determined through an iterative optimization procedure. Many different approaches have been proposed in the literature (Gönen et al., 2011). Among them, three significant

methods have been evaluated in this work: *Scalable Multiple Kernel Learning* (EasyMKL) (Aiolli & Donini, 2015), *Radius-Margin Optimization Based on Gradient Descent* (RMGD) (Lauriola et al., 2017) and *Minimum Effort Maximum Output* (MEMO) (Lauriola et al., 2018).

In addition, various deep learning approaches have also been proposed for MKL. For example, *Deep Multiple Kernel Laearning* (DMKL) (Strobl & Visweswaran, 2013) defines a multi-layer network to combine a set of M basis kernel methods. The first layer applies the given kernel methods to the input data $\mathbf{X} \in \mathbb{R}^{N \times D}$, yielding M kernel matrices. In the next layer, each kernel method is applied again not to the input data but to a kernel matrix (i.e., kernel within a kernel) obtained as a weighted average of the M kernel matrices computed in the previous layer, yielding a single combined kernel matrix per method (M new kernel matrices). This is iterated for a number of layers. In the end, the M kernel matrices obtained at the last layer are weighted averaged to yield the final kernel matrix $\mathbf{K}$ generated by DMKL. A training stage iterates over the different layers setting the coefficients of all those weighted averages through gradient descent in order to minimize the classification error of a SVM classifier fed with $\mathbf{K}$. In the present work, DMKL has been applied to combining four kernelized versions of DR spectral methods: KPCA, KCMDS, KLLE and KLE.

4.2 Ensemble learning

The rationale behind ensemble learning is that combining diversity is synergistic. Its aim is to combine multiple learning methods (usually classifiers) or instances of a same method previously trained on a same data set in order to yield a better performance than when those methods are applied on their own due to the minimization of overfitting and generalization error, which negatively affect the performance of any learning system. The three most-popular ensemble learning methods are bagging, boosting and stacking.

Bootstrap aggregating (or *bagging*) (Efron & Tibshirani, 1994) typically trains multiple instances of a same learning method with different subsets of data randomly drawn from the given data set. The predictions of the different ensemble members are then combined with simple statistics, such as averaging or voting. The Random Forest classification method (Breiman, 2001), which is a referent in ensemble learning, uses bagging to combine multiple independent decision trees. By averaging their predictions, it reduces the overall variance and yields more accurate results.

Boosting (Yao & Doretto, 2010) sequentially trains multiple instances of a same learning method with the entire data set, although every new instance focuses on the data points that have been misclassified by the previous instance. Thus, each new instance aims to correct the predictions made by the previous one. Finally, the predictions of all instances are combined by averaging or voting. AdaBoost (Freund & Schapire, 1997) is an effective and widely-used implementation of boosting.

Stacked generalization (or *stacking*) (Wolpert, 1992) combines the outputs of a set of heterogeneous learning methods (called base-models) independently pre-trained on a same data set by means of a so-called metamodel. The latter can be any machine learning method, such as linear or logistic regression, and is trained based on the predictions of the base-models upon a different data set or different partition of the data set with which those base-models were trained. The goal of the metamodel is to learn what base-models are the most reliable.

5 Stacked generalization of spectral methods with deep networks

In computational terms, the goal of dimensionality reduction from the feature projection standpoint is to determine a structure-preserving continuous map f (referred to as *embedding*), $f : \mathbf{X} \rightarrow \mathbf{Y}$, which projects a given set $\mathbf{X}$ of N data points defined in a space of high dimensionality D, $\mathbf{X} \in \mathbb{R}^{N \times D}$, into a set $\mathbf{Y}$ of N points defined in a space of low dimensionality d, $\mathbf{Y} \in \mathbb{R}^{N \times d}$, with $d \ll D$.

This section describes an online DR method, referred to as NetDR, which synergistically combines classical spectral methods with modern deep neural networks and embedding optimization. NetDR consists of two stages. In the first stage, a set of deep encoders is trained with the embeddings induced by a collection of linear and non-linear DR spectral methods independently applied to the original data. Every encoder inherits the properties of its associated DR method and is able to project new data points in an online manner. Afterwards, the embedding induced by each encoder is the seed of an optimization process that tends to bring closer in the low-dimensional space points that are close in the high-dimensional space, as well as to separate in the low-dimensional space distant points in the high-dimensional space.

In the second stage, a stacking ensemble learning approach is applied in order to combine the low-dimensional outputs of all the encoders through a metamodel implemented as a multilayer fully-connected network with two variations. The first variation enforces topological preservation (Sanodiya & Mathew, 2019) through an unsupervised approach with a shallow network trained from a graph that represents the neighborhoods of the original points, such that those neighborhoods are kept in the low-dimensional space. The second variation enforces cluster induction trough a supervised approach with a deep network that aims to minimize intra-cluster distances and to maximize inter-cluster distances. These two stages are described in detail below.

5.1 Learning DR spectral methods

Given a set of M dimensionality reduction spectral methods $\{DR_1, \ldots, DR_M\}$, the aim of this stage is to train M deep encoders $\{\varepsilon_1, \ldots, \varepsilon_M\}$ such that every encoder learns the embedding induced by its corresponding DR method when applied to the original data set $\mathbf{X}$. Figure 1 outlines the training procedure for every encoder, which is described below.

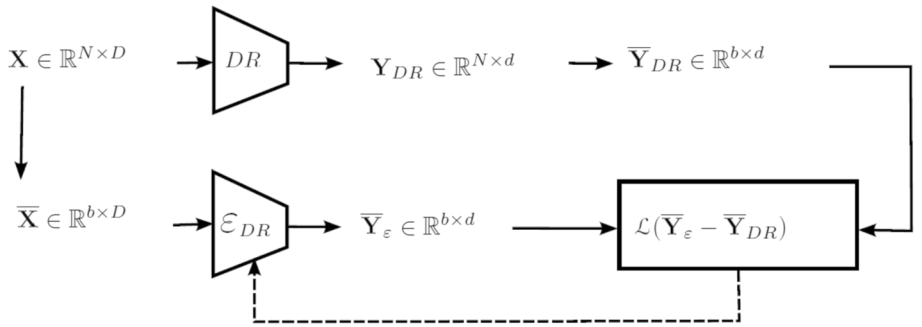

Fig. 1 Learning spectral method DR through associated encoder ε_{DR}

Let ε_{DR} be the deep encoder associated with the corresponding spectral method *DR*. The latter is first applied to the entire data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ in order to obtain the low-dimensional embedding $\mathbf{Y}_{DR} \in \mathbb{R}^{N \times d}$. Both $\mathbf{X}$ and $\mathbf{Y}_{DR}$ are used to train ε_{DR} in a supervised way. First, they are randomly partitioned into *N*/*b minibatches* of *b* data points each. At every training *epoch*, the encoder is sequentially trained with all those minibatches, hence processing the whole data set. After every epoch, all data points are reshuffled before generating the new random partition.

Processing minibatches instead of full data sets is advantageous due to the stochastic nature of the network training procedure. The latter first computes the gradient of the network's approximation error (*loss*) with respect to each network's parameter (weight) by applying *backpropagation*, and then minimizes that loss by updating every parameter with a fraction of its negative gradient through *gradient descent*. With small minibatches, gradients are computed more efficiently and parameter updates are somewhat noisy, which may help escape from sharp local minima. The latter reduces overfitting to the training set, hence increasing the network's generalization.

Let $\overline{\mathbf{X}} \in \mathbb{R}^{b \times D}$ be a certain minibatch sampled from $\mathbf{X}$, and $\overline{\mathbf{Y}}_{DR} \in \mathbb{R}^{b \times d}$ the corresponding minibatch obtained from $\mathbf{Y}_{DR}$. The encoder is fed with the data points in $\overline{\mathbf{X}}$ and generates a minibatch of low-dimensional points $\overline{\mathbf{Y}}_{\varepsilon}$. The encoder's approximation error or loss is a real number that expresses the difference between the current output $\overline{\mathbf{Y}}_{\varepsilon}$ and the ground-truth $\overline{\mathbf{Y}}_{DR}$. Such error is obtained by applying a *loss function*: $\mathcal{L}(\overline{\mathbf{Y}}_{\varepsilon} - \overline{\mathbf{Y}}_{DR})$. Since the problem at hand is not classification but regression, any regression loss function can be applied, such as *Mean Square Error* (MSE or L2 loss), *Mean Absolute Error* (MAE or L1 loss), *Smooth Mean Absolute Error* (Huber or Smooth L1 loss), among others Wang et al. (2020). In the experiments conducted in this work we have opted for the Smooth L1 loss (Huber, 1964) in being more robust to outliers.

Once the loss function is evaluated, the encoder is trained to reduce the approximation error by first computing the error gradient through backpropagation and then reducing that error by means of gradient descent. In the present work, we have applied the *Adam optimization* algorithm (Kingma & Ba, 2014), which is an extension to stochastic gradient descent.

Regarding the topology of the encoder, any multilayer feedforward or convolutional neural network can be used provided the last layer is linear (the encoder's outputs are unbounded) and *d*-dimensional vectors are generated. Since the experiments in this work have been conducted on image datasets, encoders have been implemented as CNNs with the structure shown in Fig. 2 and detailed in Sect. 6.3. Skip connections have been introduced in order to reduce vanishing gradients. Notwithstanding, any backbone CNN (e.g., VGG, ResNet, etc.) can be utilized provided the last layer is linear and each input image is reduced to a 1×1 tensor with *d* channels.

The aforementioned process has been applied in order to learn eight DR spectral methods {*PCA*, *CMDS*, *LLE*, *LE*, *KPCA*, *KCMDS*, *KLLE*, *KLE*} through their respective deep encoders: ε_{PCA}, ε_{CMDS}, ε_{LLE}, ε_{LE}, ε_{KPCA}, ε_{KCMDS}, ε_{KLLE} and ε_{KLE}.

Once the encoders have been trained as described above, they are optimized by applying ParametricUMAP (Sainburg et al., 2021; Xu & Zhang, 2023). In this way, the initial embedding induced by each encoder is the seed of an optimization process that tends to bring closer in the low-dimensional (LD) space points that are close in the high-dimensional (HD) space, and to repel in the LD space distant points in the HD space.

ParametricUMAP operates as follows. Firstly, a distance matrix $\mathbf{D}_{\mathbf{X}} \in \mathbb{R}^{N \times N}$ is defined as:

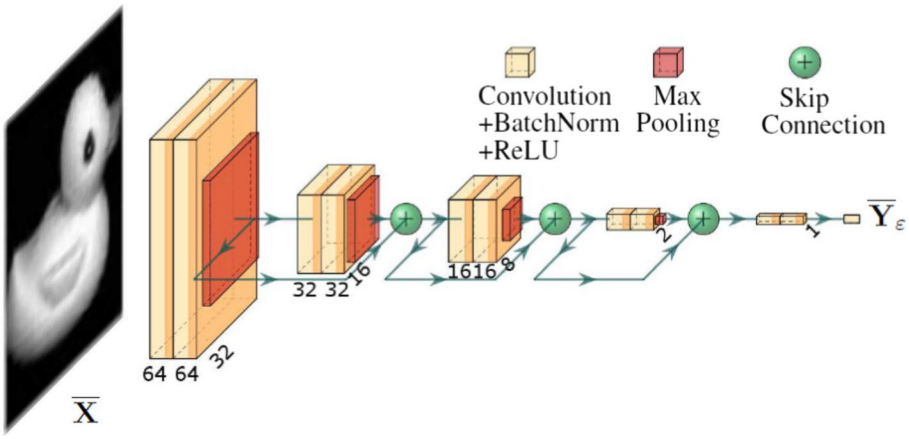

Fig. 2 Structure of convolutional encoders

$$\mathbf{D}_\mathbf{X}(i,j) = \|\mathbf{X}_i - \mathbf{X}_j\|. \tag{2}$$

A graph representing the k nearest neighbors of every point in $\mathbf{X}$ is also generated, with k being a hyperparameter. Based on that graph and the distance matrix $\mathbf{D}_\mathbf{X}$, a matrix of *similarity scores* $\mathbf{P}_\mathbf{X} \in \mathbb{R}^{N\times N}$ between each point and its neighbors is defined as:

$$\mathbf{P}_\mathbf{X}(i,j) = e^{-(\mathbf{D}_\mathbf{X}(i,j)-\rho_i)/\sigma_i}, \tag{3}$$

where ρ_i is the distance to the closest neighbor of $\mathbf{X}_i$ in the graph, and σ_i is a value that normalizes the similarity scores of each point. In particular, σ_i is chosen such that the sum of similarity scores $\mathbf{P}_\mathbf{X}(i,j)$ of $\mathbf{X}_i$ with its k nearest neighbors in the graph is lower than $\log_2(k)$. The similarity scores $\mathbf{P}_\mathbf{X}$ are not symmetric. A new matrix $\mathbf{G}_\mathbf{X} \in \mathbb{R}^{N\times N}$ of symmetric similarity scores is thus defined for the whole dataset, with $\circ$ being the element-wise (Hadamard) product:

$$\mathbf{G}_\mathbf{X} = (\mathbf{P}_\mathbf{X} + \mathbf{P}_\mathbf{X}^T) - \mathbf{P}_\mathbf{X} \circ \mathbf{P}_\mathbf{X}^T \tag{4}$$

Both $\mathbf{D}_\mathbf{X}$ and $\mathbf{G}_\mathbf{X}$ are computed once for the whole dataset. However, the neural encoders are applied to different subsets of b data points $\overline{\mathbf{X}} \in \mathbb{R}^{b\times D}$ randomly sampled from $\mathbf{X} \in \mathbb{R}^{N\times D}$. Therefore, every time an encoder is applied, we consider the distance submatrix and symmetric similarity score submatrix corresponding to the HD points that belong to the minibatch: $\mathbf{D}_{\overline{\mathbf{X}}} \in \mathbb{R}^{b\times b}$ and $\mathbf{G}_{\overline{\mathbf{X}}} \in \mathbb{R}^{b\times b}$. They are submatrices directly extracted from $\mathbf{D}_\mathbf{X}$ and $\mathbf{G}_\mathbf{X}$, respectively.

Secondly, based on the embedding $\overline{\mathbf{Y}}_\varepsilon \in \mathbb{R}^{b\times d}$ generated by each encoder, a second similarity score matrix $\mathbf{Q}_{\overline{\mathbf{Y}}_\varepsilon} \in \mathbb{R}^{b\times b}$ is defined for each minibatch as follows:

$$\mathbf{Q}_{\overline{\mathbf{Y}}_\varepsilon} = \left(1 + \alpha \, \|\overline{\mathbf{Y}}_\varepsilon - \overline{\mathbf{Y}}_\varepsilon^T\|^{2\beta}\right)^{-1} \tag{5}$$

Those LD similarity scores come from a fixed bell-shaped t-distribution curve. Hyperparameter α controls the shape of the t-distribution: as α increases, the t-distribution approaches a standard normal distribution. In turn, hyperparameter β determines how the

contribution of the squared norm is weighted in the overall expression, allowing the sensitivity of the model to be adapted to differences in the data.

The goal of the ParametricUMAP optimization process is to minimize the difference between both similarity scores, $\mathbf{G}_{\overline{\mathbf{X}}}$ and $\mathbf{Q}_{\overline{\mathbf{Y}}_\varepsilon}$. This is done by minimizing the following cross-entropy cost function $\mathcal{C}$ through gradient descent:

$$\mathcal{C}(\mathbf{G}_{\overline{\mathbf{X}}}, \mathbf{Q}_{\overline{\mathbf{Y}}_\varepsilon}) = -\sum_{i \neq j} \mathbf{G}_{\overline{\mathbf{X}}}(i,j) \log \frac{\mathbf{G}_{\overline{\mathbf{X}}}(i,j)}{\mathbf{Q}_{\overline{\mathbf{Y}}_\varepsilon}(i,j)} + (1 - \mathbf{G}_{\overline{\mathbf{X}}}(i,j)) \log \frac{1 - \mathbf{G}_{\overline{\mathbf{X}}}(i,j)}{1 - \mathbf{Q}_{\overline{\mathbf{Y}}_\varepsilon}(i,j)} \tag{6}$$

That cost function is applied to all pairs of points in the minibatch. Pairs with a high similarity score $\mathbf{G}_{\overline{\mathbf{X}}}(i,j)$ are nearby points in the HD space. In turn, pairs with a low $\mathbf{G}_{\overline{\mathbf{X}}}(i,j)$ are distant points in the HD space. By minimizing that cross-entropy function, nearby points in the HD space tend to move closer in the LD space (i.e., attraction), whereas far away points in the HD space tend to separate in the LD space (i.e., repulsion). As a result of this first stage, an improved embedding $\overline{\mathbf{Y}}_\varepsilon \in \mathbb{R}^{b \times d}$ is obtained through the neural encoder associated with each considered DR method.

5.2 Combining DR spectral methods

Let us assume a set of M encoders $\{\varepsilon_1, \ldots, \varepsilon_M\}$ trained as described in the previous section. Their low-dimensional outputs $\{\overline{\mathbf{Y}}_1, \ldots, \overline{\mathbf{Y}}_M\}, \overline{\mathbf{Y}}_i \in \mathbb{R}^{b \times d}$, are integrated by following a stacking ensemble learning approach through a metamodel defined as a multilayer feed-forward neural network (FFN) constituted by several consecutive fully-connected (linear) layers. As a result of the integration, a low-dimensional output $\overline{\mathbf{Y}} \in \mathbb{R}^{b \times d}$ is generated. The set of M encoders along with the FFN metamodel constitute the integrated encoder ε.

The FFN metamodel is trained with the original data $\mathbf{X} \in \mathbb{R}^{N \times D}$ by randomly partitioning them into N/b minibatches $\overline{\mathbf{X}} \in \mathbb{R}^{b \times D}$ of b data points each. At every training epoch, FFN is sequentially trained with all those minibatches, hence processing the whole data set. After every epoch, all data points are reshuffled before generating the new random partition.

DR methods may exhibit topological preservation or cluster induction. As both properties tend to be mutually exclusive, we propose two variations of FFN depending on whether the final aim is to preserve the topology of the original data or to induce non-overlapping clusters.

5.2.1 Topological preservation

In order to enforce topological preservation, the FFN metamodel is configured as a shallow feed-forward network illustrated in Fig. 3. It consists of two consecutive fully-connected layers with a Mish activation function (Misra, 2020) in between. This two-layer topology was found to be optimal for the seven tested datasets as shown in the experiments reported in Sect. 7.3.1. FFN is trained in an unsupervised way by considering a compound loss function $\mathcal{L}$ defined as follows.

Let $\mathbf{D}_{\overline{\mathbf{X}}} \in \mathbb{R}^{b \times b}$ be the distance matrix between the HD points belonging to the current minibatch. $\mathbf{D}_{\overline{\mathbf{X}}}$ is a submatrix of $\mathbf{D}_{\mathbf{X}}$ (2). Let $\mathbf{D}_{\overline{\mathbf{Y}}} \in \mathbb{R}^{b \times b}$ be the corresponding distance matrix between the LD points generated by FFN for the current minibatch. An HD symmetric similarity score matrix $\mathbf{G}_{\overline{\mathbf{X}}} \in \mathbb{R}^{b \times b}$ is obtained by applying (3) and (4) to $\overline{\mathbf{X}}$. In turn, an LD similarity score matrix $\mathbf{Q}_{\overline{\mathbf{Y}}} \in \mathbb{R}^{b \times b}$ is obtained by applying (5) to $\overline{\mathbf{Y}}$. Finally, a

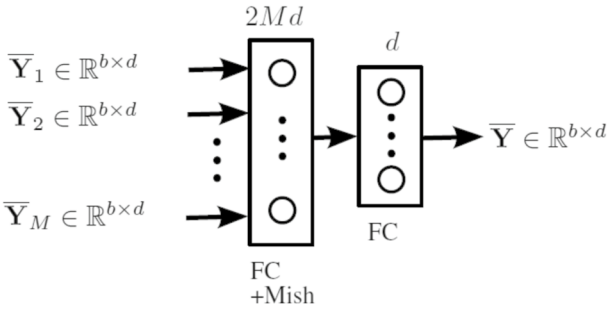

Fig. 3 Structure of feedforward network (FFN) with 2 fully-connected linear layers, which exhibits topological preservation

cross-entropy cost function $\mathcal{L}_L = \mathcal{C}(\mathbf{G}_{\overline{\mathbf{X}}}, \mathbf{Q}_{\overline{\mathbf{Y}}})$ is defined by applying (6) to the previous HD and LD similarity score matrices. As described in Sect. 5.1, by minimizing that cost function, close points in HD will tend to approach in LD, whereas far away points in HD will tend to separate in LD. This aims at preserving the original HD *local topology*.

Conversely, the original HD *global topology* can be preserved my minimizing a regression loss function $\mathcal{L}_G = \mathcal{D}(\mathbf{D}_{\overline{\mathbf{X}}} - \mathbf{D}_{\overline{\mathbf{Y}}})$ that determines the difference between the HD and LD distance matrices. In the experiments conducted in this work, we have opted for the Smooth L1 loss (Huber, 1964) in being more robust to outliers.

Finally, the FFN metamodel is trained by minimizing a compound topological loss function $\mathcal{L}_t$ that depends on both the local and global terms defined above:

$$\mathcal{L}_t = w\mathcal{L}_L + (1-w)\mathcal{L}_G \tag{7}$$

The weighting factor w is a hyperparameter between 0 and 1 that determines the balance between local, $\mathcal{L}_L$, and global, $\mathcal{L}_G$, topological preservation. A tradeoff value $w = 0.5$ was considered in the experiments conducted in this work. The parameters of the M encoders were frozen while training FNN. However, similar results were obtained by unfreezing all encoders. The full process is illustrated in Fig. 4.

5.2.2 Cluster induction

The goal of dimensionality reduction with cluster induction is to yield a discriminant embedding in which low-dimensional points from different classes group into non-overlapping clusters. Since it is necessary to know the class label corresponding to every dataset point, this is a supervised problem. The FFN metamodel is thus trained in a supervised way through a deep autoencoder defined as follows.

The topology of the FFN metamodel is illustrated in Fig. 5. It is constituted by $2L$ consecutive fully-connected layers: L expanding layers followed by L contracting layers. The i-th expanding layer has 2^iMd nodes (neurons). In turn, the i-th contracting layer has $2^{L-i}Md$ nodes except for the last layer, which has d nodes. As shown in Sect. 7.3.1, the optimal number of layers was found to depend on the evaluated dataset.

Let us now assume that the given dataset contains N high-dimensional points, $\mathbf{X} \in \mathbb{R}^{N\times D}$, belonging to C different classes. We define a function $\Lambda : \mathbb{R}^D \to \mathbb{R}^C$ that maps a high-dimensional point $\mathbf{X}_i$ into a one-hot vector $\mathbf{O}_i$, $\mathbf{O}_i = \Lambda(\mathbf{X}_i)$, such that $\sum_{c=1}^{C} \mathbf{O}_i(c) = 1$ and $\mathbf{O}_i(c) = 1$ if $\mathbf{X}_i$ belongs to class c.

Again, we assume a minibatch $\overline{\mathbf{X}} \in \mathbb{R}^{b\times D}$ of b points randomly sampled from $\mathbf{X}$. Let $\mathbf{O} = \Lambda(\overline{\mathbf{X}})$ be the set of b one-hot vectors associated with $\overline{\mathbf{X}}$. The output $\overline{\mathbf{Y}}$ of the

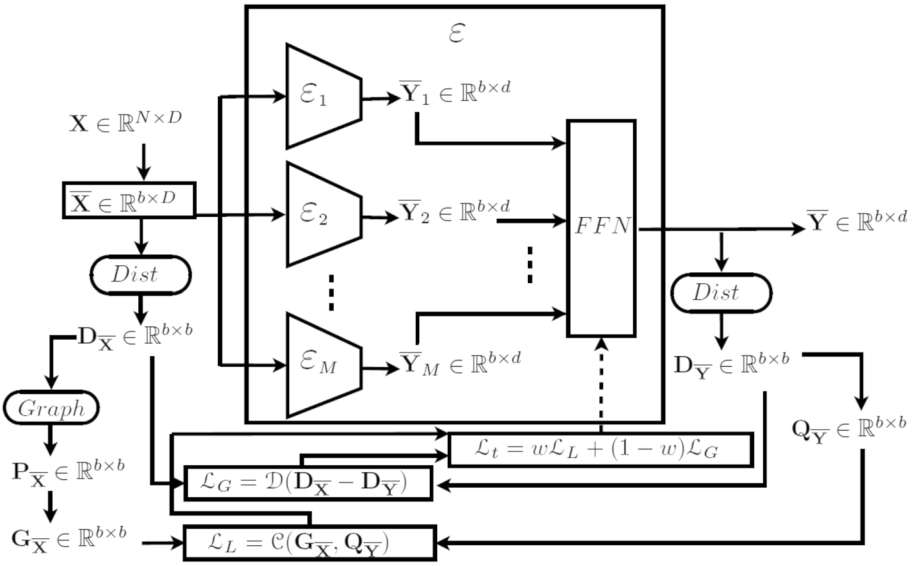

Fig. 4 Combination of DR methods through multilayer feedforward network trained with Euclidean distance matrices for global topological preservation, and with a neighborhood graph for local topological preservation

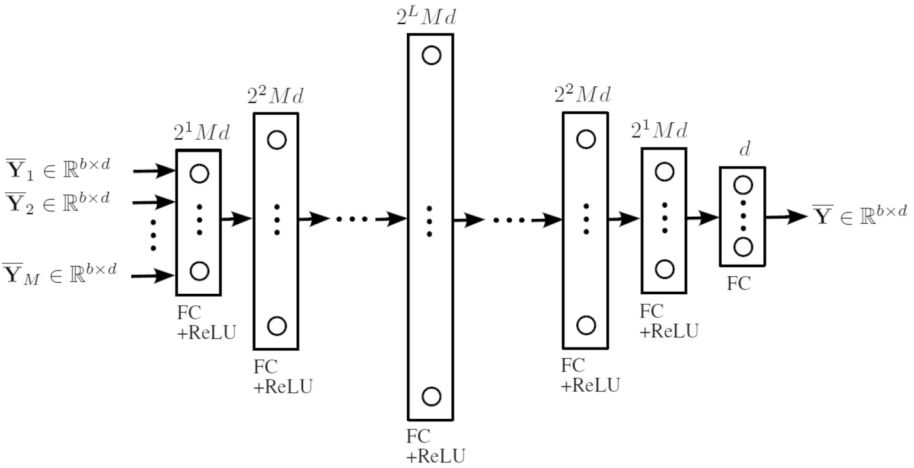

Fig. 5 Structure of feedforward network (FFN) with $2L$ fully-connected linear layers, which exhibits cluster induction

FFN metamodel is fed into a decoder δ that generates b vectors of class probabilities $\mathbf{C} \in \mathbb{R}^{b \times C}$, such that $\mathbf{C}_i(c)$ is the predicted probability that the projected point $\overline{\mathbf{Y}}_i$ (i.e., $\overline{\mathbf{X}}_i$) belongs to class c. The topology of the decoder is shown in Fig. 6. It is a shallow feed-forward network with two consecutive fully-connected layers. A softmax function is applied to the output logits to generate the class probabilities $\mathbf{C}$.

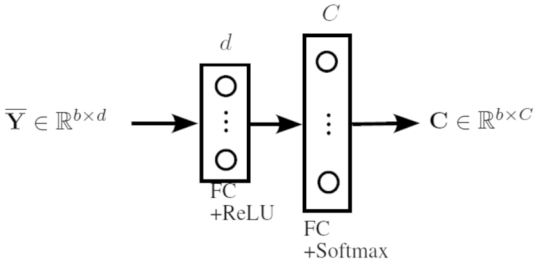

Fig. 6 Structure of decoder δ

Ideally, the class probability vector associated with each point from the minibatch, $\mathbf{C}_i$, should coincide with its corresponding one-hot vector $\mathbf{O}_i$. In order to fulfill this goal, the FFN metamodel is finally trained by minimizing a cross-entropy cost function $\mathcal{L}_C$ defined as:

$$\mathcal{L}_C = \mathcal{C}'(\mathbf{C}, \mathbf{O}) = -\sum_{i=1}^{b}\sum_{c=1}^{C} \mathbf{O}_i(c)\log \mathbf{C}_i(c) + (1-\mathbf{O}_i(c))\log(1-\mathbf{C}_i(c)) \tag{8}$$

The full process is illustrated in Fig. 7. The parameters of the M encoders are frozen, hence only learning the parameters of the FFN metamodel and the decoder. Notwithstanding, similar results have been obtained by unfreezing all encoders.

6 Experimental setup

The proposed dimensionality reduction approach has been experimentally validated on seven widely-known image datasets introduced in Sect. 6.1 using the topological preservation and cluster induction measures described in Sect. 6.2. The topology of encoders and decoders has been defined based on those datasets as detailed in Sect. 6.3.

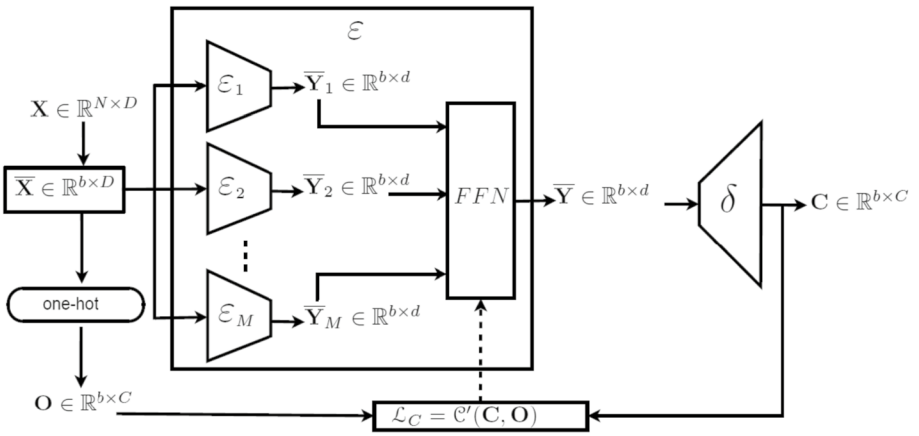

Fig. 7 Combination of DR methods through multilayer feedforward network trained in a supervised manner, which exhibits cluster induction

6.1 Datasets

The experiments in this work have been conducted on seven widely-used image datasets: *Fashion-MNIST*, *COIL-20*, *LEGO Bricks*, *Medical-MNIST*, *USPS*, *Imagenette* and *ISL*. Figure 8 shows examples of images of the seven considered datasets. *Fashion-MNIST* (Xiao et al., 2017) consists of a training set with 60,000 images and a test set with 10,000 images that belong to 10 different categories corresponding to various Zalando's articles. In this case, $N = 60,000$. *COIL-20* (Columbia Object Image Library) (Nene et al., 1996), contains $N = 1,440$ images of 20 simple objects, such that every object was rotated 360 degrees on a turntable in 5-degree steps, yielding 72 images per object. *LEGO Bricks* Hazelzet (2019) contains $N = 12,800$ images of LEGO bricks belonging to 16 different classes. Every class contains 800 images of a same LEGO brick rendered from different angles. Every original image has 400×400 pixels. The *Medical-MNIST* or MedNIST Polanco Lozano (2017) dataset contains 58,954 ($N = 58,954$) medical and radiology images labeled in 6 categories: AbdomenCT (10,000 images), HeadCT (10,000 images), Hand (10,000 images), CXR (10,000 images), BreastMRI (8,954 images) and ChestCT (10,000 images). The Hand-written Digits *USPS* dataset (Hull, 1994a, 1994b) contains digits scanned from envelopes received by the U.S. Postal Service. This dataset contains a total of 9,298 grayscale ($N = 9,298$), centered and normalized images (7,291 training images and 2,007 test images). *Imagenette* (Howard et al., 2020) is a subset of the well-known *ImageNet* dataset. It contains $N = 9,469$ images belonging to ten classes: *tench*, *English springer*, *cassette player*, *chain saw*, *church*, *French horn*, *garbage truck*, *gas pump*, *golf ball*, and *parachute*. They are divided into 70% training images and 30% test images. Finally, Indian Signal Language (*ISL*) (Preksha, 2023) contains the 26 letters of the English alphabet represented by $N = 50,000$ high-resolution images of hand gestures captured using the MediaPipe pose estimation technology, with different acquisition angles, lighting conditions and skin tones.

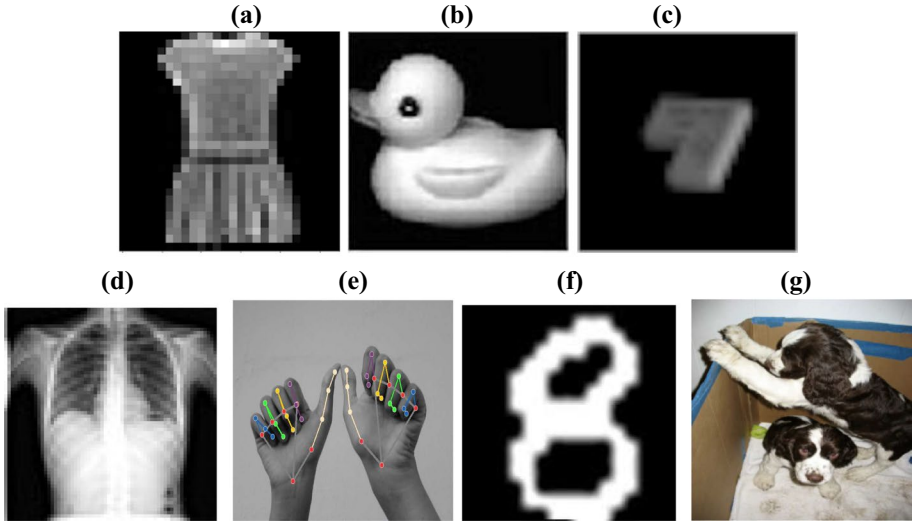

Fig. 8 Examples of images from *Fashion-MNIST* (**a**), *COIL-20* (**b**), *LEGO Bricks* (**c**), *Medical-MNIST* (**d**), *ISL* (**e**), *USPS* (**f**), and *Imagenette* (**g**)

6.2 Quality measures

The proposed method has been experimentally validated and compared to some of the most relevant dimensionality reduction techniques using the topological preservation and cluster induction measures described below.

6.2.1 Topological preservation measures

R_{NX} curves (Lee et al., 2013) are useful to assess the quality of topological preservation attained by the generated low-dimensional embeddings with respect to the original data. These curves express the preservation of K-ary neighborhoods, which is related to the concepts of trustworthiness and continuity (Venna and Kaski, 2007). *Trustworthiness* measures how many neighbors in the low-dimensional space are also neighbors in the original space, whereas *continuity* is the opposite. Distant points that become neighbors (i.e., *intrusion*) reduce trustworthiness, whereas close points that are embedded far away from each other (i.e., *extrusion*) reduce continuity.

In particular, let ν_i^K be the set of K-nearest neighbors of point $\mathbf{x}_i \in \mathbb{R}^D$ in the original high-dimensional space, and n_i^K the K-nearest neighbors of its corresponding projected point $\mathbf{y}_i \in \mathbb{R}^d$ in the low-dimensional space. The number of neighbors K is referred to as *perplexity* (Hinton & Roweis, 2002). If ν_i^K and n_i^K coincide, the cardinality of their intersection is K: $|\nu_i^K \cap n_i^K| = K$. This means that the embedding accurately represents the high-dimensional space given a perplexity K. Based on that, the average normalized agreement between corresponding K-ary neighborhoods of the given N points is defined as the Q_{NX} index (Lee et al., 2013), which ranges between zero (no topological preservation) and one (perfect neighborhood agreement):

$$Q_{NX}(K) = \sum_{i=1}^{N} \frac{|\nu_i^K \cap n_i^K|}{KN}.$$

However, it can be shown that for a random embedding, $Q_{NX}(K) \approx K/(N-1)$ (Lee & Verleysen, 2009). Thus, the $R_{NX}(K)$ measure is defined in such a way that it gives zero for a random embedding and one for perfect neighborhood agreement (Lee et al., 2013):

$$R_{NX}(K) = \frac{(N-1)Q_{NX}(K) - K}{N-1-K}.$$

An R_{NX} curve is generated by plotting $R_{NX}(K)$ for different values of K between 1 and $N-2$. K is shown in a logarithmic scale in order to reflect that errors in large neighborhoods are proportionally less important than in small ones. Finally, the different $R_{NX}(K)$ values are aggregated in a scalar score defined as the *Area Under Curve* (AUC) in the log plot of $R_{NX}(K)$ (Lee et al., 2015). AUC measures the overall topological preservation quality of an embedding by considering all perplexities, with one indicating perfect neighborhood preservation and zero representing random results:

$$AUC_{\ln K}(R_{NX}(K)) = \frac{\sum_{K=1}^{N-2} R_{NX}(K)/K}{\sum_{K=1}^{N-2} 1/K}. \tag{9}$$

6.2.2 Cluster induction measures

In order to evaluate the clustering capabilities of a dimensionality reduction method, it is first necessary to map the given set $\mathbf{X}$ of N data points defined in the high-dimensional space, $\mathbf{X} \in \mathbb{R}^{N \times D}$, into the corresponding set $\mathbf{Y}$ of N embedded points, $\mathbf{Y} \in \mathbb{R}^{N \times d}$. A partition of the original dataset $\mathbf{X}$ into K classes must be available, with every single point belonging to one of those classes. Let $\mathbf{Y}_{GT} = \{\mathbf{Y}_{GT_i}\}_{i=1}^{K}$ be the family of K subsets (classes) induced in $\mathbf{Y}$ by the predefined (ground-truth) partition of $\mathbf{X}$. A clustering method such as *k-means* (Teknomo, 2006) is then applied to $\mathbf{Y}$ in order to generate a second family of K subsets (clusters) $\mathbf{Y}_C = \{\mathbf{Y}_{C_i}\}_{i=1}^{K}$. Both $\mathbf{Y}_{GT}$ and $\mathbf{Y}_C$ can be considered to be random discrete variables and their associated subsets their possible outcomes that occur with probabilities $P(\mathbf{Y}_{GT_i}) = \|\mathbf{Y}_{GT_i}\|/N$ and $P(\mathbf{Y}_{C_i}) = \|\mathbf{Y}_{C_i}\|/N$, respectively, with $\|\cdot\|$ being the cardinality operator.

The Shannon entropy (Shannon, 1948) of both the ground-truth partition, $H(\mathbf{Y}_{GT})$, and the clustering, $H(\mathbf{Y}_C)$, measures the dispersion of the N data points into the K respective classes/clusters. Maximum entropy implies largest dispersion. In turn, the conditional entropy of the ground-truth partition given the clustering, $H(\mathbf{Y}_{GT}|\mathbf{Y}_C)$, measures the dispersion of the points belonging to every cluster among the K classes. Similarly, the entropy of the clustering conditioned on the ground-truth partition, $H(\mathbf{Y}_C|\mathbf{Y}_{GT})$, measures the dispersion of the points belonging to every ground-truth class among the K clusters. Based on the previous definitions, two complementary clustering measures are defined in Rosenberg and Hirschberg (2007): homogeneity and completeness.

The *Homogeneity Score* (HOM) (Rosenberg & Hirschberg, 2007) is defined as $h = 1 - H(\mathbf{Y}_{GT}|\mathbf{Y}_C)/H(\mathbf{Y}_{GT})$. The conditional entropy is zero when all points of every cluster are not dispersed among the ground-truth classes, but concentrated into a single one. This yields the maximum homogeneity, $h = 1$, meaning that every cluster only has points belonging to the same class.

In turn, the *Completeness Score* (COM) (Rosenberg & Hirschberg, 2007) is defined as $c = 1 - H(\mathbf{Y}_C|\mathbf{Y}_{GT})/H(\mathbf{Y}_C)$. In this case, the conditional entropy is zero when all points of every class are not dispersed among the K clusters, but concentrated into a single one. This yields the maximum completeness, $c = 1$, meaning that every ground-truth class only has points belonging to the same cluster.

Both measures are finally combined through the weighted harmonic mean of homogeneity and completeness, which is referred to as the *V-Measure Score* (V) (Rosenberg & Hirschberg, 2007): $V_\beta = (1+\beta)hc/(\beta h + c)$. If β is greater than one, completeness receives more weight. If β is lower than one, homogeneity is reinforced.

6.3 Neural architecture of encoders and decoders

The images of the considered datasets were transformed to 32×32 pixels (i.e., dimensionality $D = 1,024$) in the preprocessing stage, except for the *Imagenette* dataset, whose 160-pixel images were transformed to 128×128 pixels (i.e., dimensionality $D = 16,384$).

Table 1 specifies the detailed structure of the convolutional encoder for the 32×32 images. In turn, Table 2 presents the structure of the encoder for 128×128 images. The

Table 1 Encoder structure for an image size of 32×32 pixels

Encoder, Image size: (32×32)						
Layer	in ch	out ch	kernel	stride	padding	dimension
Conv1+BatchNorm+ReLU	1	64	3	1	1	64×32×32
Conv2+BatchNorm+ReLU	64	64	3	1	1	64×32×32
Max Pool			2	2	0	64×16×16
Conv3+BatchNorm+ReLU	64	32	3	1	1	32×16×16
Conv4+BatchNorm+ReLU	32	32	3	1	1	32×16×16
Max Pool			2	2	0	32×8×8
Conv5+BatchNorm+ReLU	32	16	3	1	1	16×8×8
Conv6+BatchNorm+ReLU	16	16	3	1	1	16×8×8
Max Pool			2	4	0	16×2×2
Conv7+BatchNorm+ReLU	16	d	3	1	1	d×2×2
Conv8+BatchNorm+ReLU	d	d	3	1	1	d×2×2
Max Pool			2	2	0	d×1×1
Conv9+BatchNorm+Sigmoid	d	d	3	1	1	d×1×1
Squeeze						d

Table 2 Encoder structure for an image size of 128×128 pixels (*Imagenette*)

Encoder, Image size: (128×128)						
Layer	in ch	out ch	kernel	stride	padding	dimension
Conv0+BatchNorm+ReLU	1	32	3	1	1	32×128×128
Max Pool			2	2	0	32×64×64
Conv1+BatchNorm+ReLU	32	64	3	1	1	64×64×64
Max Pool			2	2	0	64×32×32
Conv2+BatchNorm+ReLU	64	64	3	1	1	64×32×32
Max Pool			2	2	0	64×16×16
Conv3+BatchNorm+ReLU	64	32	3	1	1	32×16×16
Conv4+BatchNorm+ReLU	32	32	3	1	1	32×16×16
Max Pool			2	2	0	32×8×8
Conv5+BatchNorm+ReLU	32	16	3	1	1	16×8×8
Conv6+BatchNorm+ReLU	16	16	3	1	1	16×8×8
Max Pool			2	4	0	16×2×2
Conv7+BatchNorm+ReLU	16	d	3	1	1	d×2×2
Conv8+BatchNorm+ReLU	d	d	3	1	1	d×2×2
Max Pool			2	2	0	d×1×1
Conv9+BatchNorm+Sigmoid	d	d	3	1	1	d×1×1
Squeeze						d

Table 3 Structure of the decoder δ

Decoder δ		
Layer	Input	Output
Fully connected	d	C

encoder was introduced in Section 5.1, while the δ decoder was presented in Sect. 5.2.2 and illustrated in Fig. 6 and Table 3.

7 Experimental validation

The experimental validation of the proposed technique is organized as follows. Section 7.1 shows the results of the first stage of the proposed technique, in which DR spectral methods are learnt by means of deep encoders. In turn, Sect. 7.2 shows results corresponding to the second stage of the proposed technique, in which the previous deep encoders are combined through a multilayer feedforward network (FFN) trained as described in Sect. 5.2. Finally, Sect. 7.3 presents three groups of experiments that analyze the behavior of the proposed technique for different configurations: by varying the number of layers in the FFN (Sect. 7.3.1), by choosing different combinations of DR spectral methods (Sect. 7.3.2), and by changing the dimensionality d of the low-dimensional space (Sect. 7.3.4). All experiments were run on an Intel Core i7 computer with 16 GB of RAM, 500 GB of SSD, and an NVIDIA GeForce 1060 GPU.

7.1 Performance of deep encoders trained with DR spectral methods

The first experiments illustrate the performance of deep encoders trained with DR spectral methods as described in Sect. 5.1. We initially considered public implementations of the original PCA method and its kernel version by assuming polynomial homogeneous kernel basis functions of degree 1: $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ (see Sect. 3.2), referred to as KPCA[1]. We also considered their incremental variations, IPCA and IKPCA[1], respectively. Two deep encoders were respectively trained from PCA and KPCA[1] by considering the training set of Fashion-MNIST: ε_{PCA} and ε_{KPCA^1}.

Figure 9 shows the R_{NX} log-plot curves and AUC measures for the 60,000 training images of Fashion-MNIST and every tested DR method. These results show a similar behavior of the DR methods regarding the R_{NX} scores except for IKPCA[1], which has a significant downfall in performance. Both encoders capture the structure of their

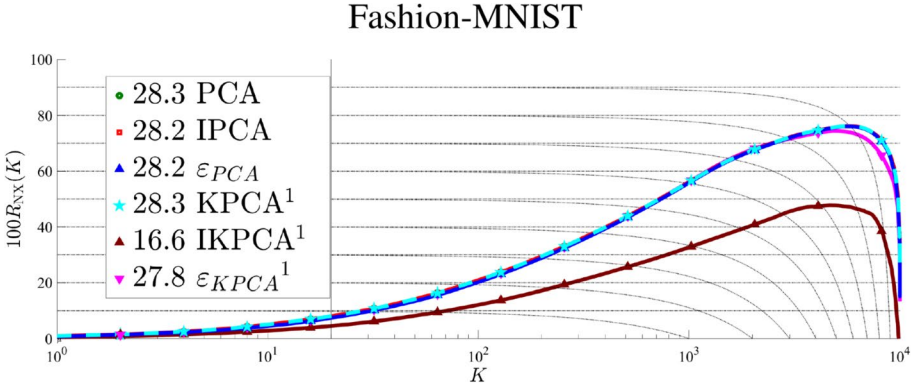

Fig. 9 R_{NX} scores for PCA, KPCA[1], IPCA, IKPCA[1], ε_{PCA} and ε_{KPCA^1}

corresponding embeddings with high accuracy while allowing the mapping of new data points in an online manner. Similar results were obtained for the other evaluated datasets. KPCA using a degree-1 polynomial kernel basis function is equivalent to classical PCA.

We also tested public implementations of the classical DR spectral methods CMDS, LLE and LE, as well as their kernel versions with degree-1 polynomial kernel basis functions (KCMDS^1, KLLE^1, KLE^1) and their incremental variations (IKCMDS^1, IKLLE^1, IKLE^1). Three deep encoders were trained from the above kernel methods by considering the training set of Fashion-MNIST: ε_{KCMDS^1}, ε_{KLLE^1} and ε_{KLE^1}. Figure 10 shows the 2D

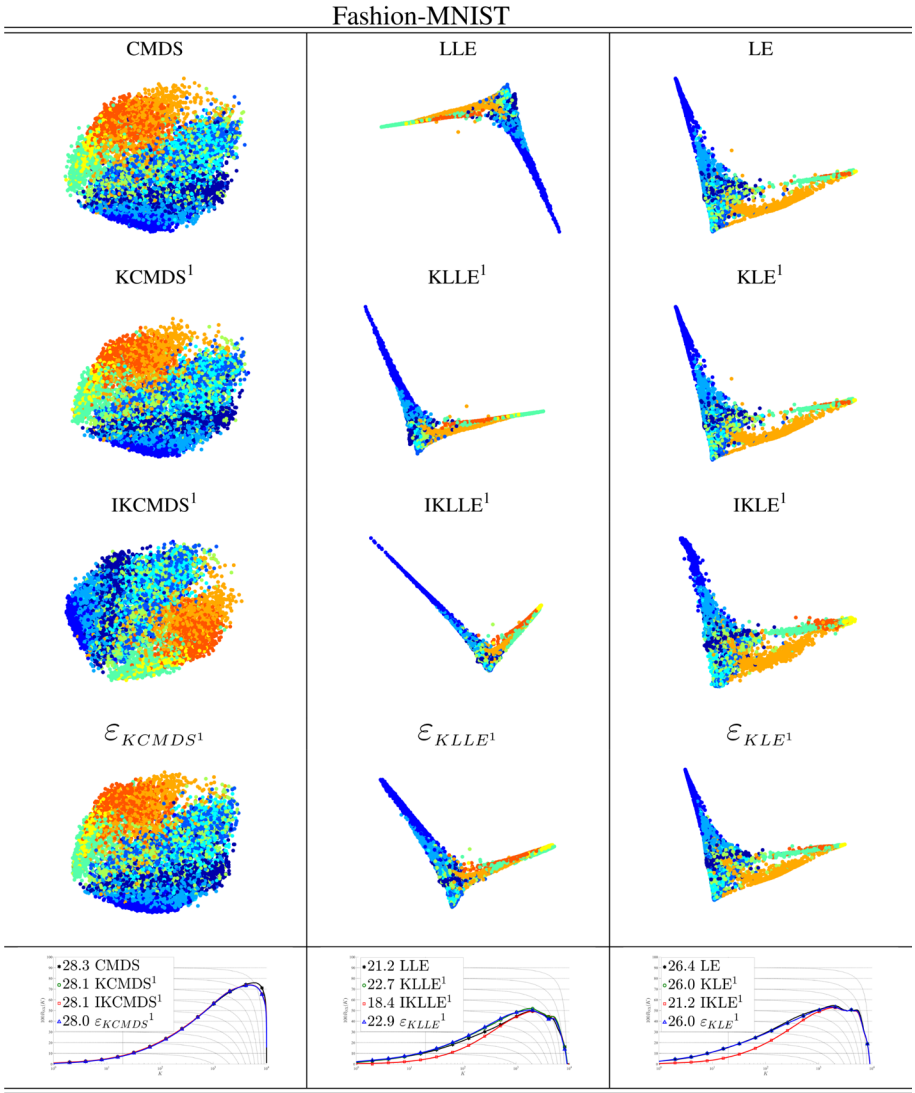

Fig. 10 2D embeddings and R_{NX} scores for CMDS, LLE, LE, their kernel and incremental versions, ε_{KCMDS^1}, ε_{KLLE^1} and ε_{KLE^1}

embeddings and their respective R_{NX} scores for the 60,000 training images of Fashion-MNIST. Again, the encoders accurately reproduce the embeddings of the original methods and the kernel DR methods using degree-1 polynomial kernel basis functions behave similarly to their respective classical counterparts. The results for the other tested datasets were alike. As a consequence, standalone degree-1 kernel methods will be omitted in the remaining experiments as their performance is on par with their corresponding classical methods.

7.2 Performance of combination of DR deep encoders

The second set of experiments illustrates the performance of the proposed methodology to combine DR deep encoders through the FFN metamodel described in Sect. 5.2. In particular, we combined ε_{KPCA^1}, ε_{KCMDS^1}, ε_{KLLE^1} and ε_{KLE^1}, whose performance is reported in the previous section. As indicated above, we consider two configurations of FFN depending on whether the final aim is topological preservation (Fig. 3) or cluster induction (Fig. 5). The combination of those four encoders with the first configuration of FFN will be referred to as NetDR^1_T (i.e., topological NetDR of degree 1), whereas their combination with the second configuration will be denoted as NetDR^1_D (i.e., discriminating NetDR of degree 1). The number of fully-connected layers within the FFN module was chosen according to the performance analysis described in Sect. 7.3.1 and Figs. 11 and 12. For both configurations of the proposed model, we also tested their performance by applying end-to-end training, that is, training the four encoders and the FFN metamodel together from scratch. We refer to those variations as NetDR^1_{e2eT} and NetDR^1_{e2eD}.

We also tested the combination of KPCA^1, KCMDS^1, KLLE^1 and KLE^1 by using both MKL and DMKL (see Sect. 4.1). We refer to them as MKL^1 and DMKL^1, respectively. In particular, we run experiments with the three MKL approaches introduced in Sect. 4.1: EasyMKL (Aiolli & Donini, 2015) for *Fashion-MNIST*, *Medical-MNIST*, *USPS* and *ISL*. RM-GD (Lauriola et al., 2017) for COIL-20 and MEMO (Lauriola et al., 2018) for LEGO. In turn, we tested DMKL (Strobl & Visweswaran, 2013) on the seven datasets.

Finally, we complemented these experiments with eight more DR methods. First, we considered the same four kernel DR methods but using degree-3 polynomial kernel basis functions, referred to as KPCA^3, KCMDS^3, KLLE^3 and KLE^3. As introduced in Sect. 3.2, the kernel trick avoids explicit mappings to high dimensional spaces. However, when polynomial kernel basis functions are used, this implicit dimensionality upscaling only occurs for degrees above one. Kernel methods defined with degree-1 polynomials are only meaningful to represent their associated classical spectral methods in kernel matrix form, such that they can be later combined through MKL approaches as we do in this work. We also tested GraphEncoder, a deep network method introduced in Sect. 3.5, as well as ISOMAP, Factor Analysis (FA) and Linear Discriminant Analysis (LDA), all introduced in Sect. 3.1. LDA is a supervised DR method that favors cluster induction, similarly to NetDR_D. Additionally, the methods referenced in 3.4, which seek to preserve the topological structure of data by applying projections or approximations, were also considered in the experimentation: ScaledPCA, UMAP, TriMAP, DensMap, SliseMap and ParametricUMAP.

Figure 11 and Table 4 show the 2D embeddings and scores corresponding to the proposed models, NetDR^1_T and NetDR^1_D, as well as the other tested methods when applied to Fashion-MNIST. In Table4 and the subsequent numerical tables (from Tables 5 to 10), the highest R_{NX} score is highlighted in bold, whereas the highest V score is

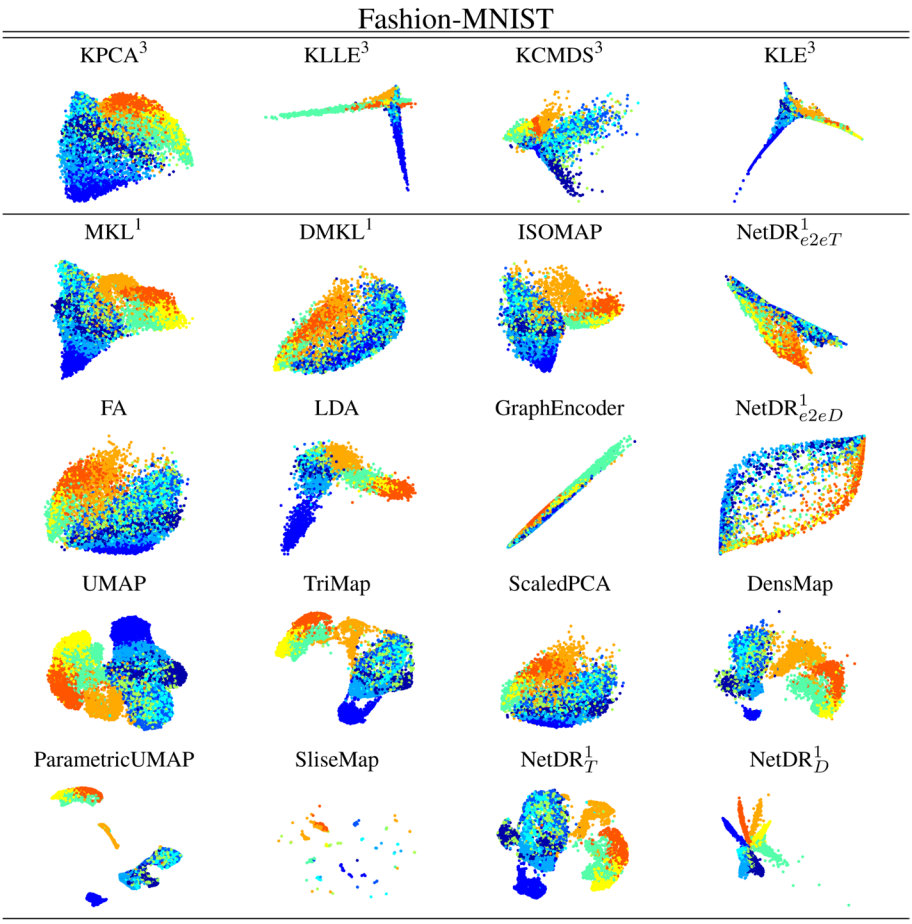

Fig. 11 2D embeddings of tested DR methods applied to the Fashion-MNIST dataset

Table 4 R_{NX} and Cluster induction measures of tested DR methods applied to the Fashion-MNIST dataset

Method	R_{NX}	V	Method	R_{NX}	V
PCA	28.2	0.421	GraphEncoder	12.9	0.368
CMDS	28.2	0.422	MKL[1]	25.3	0.508
LE	26.4	0.557	DMKL[1]	25.0	0.355
LLE	21.2	0.511	FA	27.9	0.432
KPCA[3]	25.2	0.427	LDA	14.2	0.561
KCMDS[3]	24.0	0.400	NetDR$^1_{e2eT}$	16.0	0.463
KLE[3]	20.9	0.479	NetDR$^1_{e2eD}$	12.3	0.472
KLLE[3]	14.5	0.366	NetDR^1_T	**38.8**	0.590
ISOMAP	26.6	0.486	NetDR^1_D	16.1	***0.725***
UMAP	31.4	0.538	DensMap	33.6	0.599
TriMap	33.2	0.589	SliseMap	13.4	0.631
ScaledPCA	27.7	0.415	ParametricUMAP	34.0	0.632

Table 5 R_{NX} and cluster induction measures of tested DR methods applied to the COIL-20 dataset

Method	R_{NX}	V	Method	R_{NX}	V
PCA	39.6	0.643	GraphEncoder	28.8	0.505
CMDS	39.6	0.628	MKL[1]	35.3	0.646
LE	30.5	0.558	DMKL[1]	33.8	0.659
LLE	35.1	0.626	FA	39.7	0.661
KPCA[3]	38.6	0.645	LDA	17.4	0.883
KCMDS[3]	33.7	0.630	NetDR^1_{e2eT}	34.1	0.542
KLE[3]	31.0	0.571	NetDR^1_{e2eD}	26.8	0.619
KLLE[3]	26.8	0.632	NetDR^1_{T}	**48.9**	0.839
ISOMAP	29.0	0.649	NetDR^1_{D}	30.6	***0.960***
UMAP	40.1	0.707	DensMap	44.5	0.878
TriMap	43.6	0.863	SliseMap	14.9	0.661
ScaledPCA	39.6	0.637	ParametricUMAP	44.6	0.882

highlighted in both bold and italic. Notice that NetDR^1_T yields the best AUC score (38.8) followed by ParametricUMAP (34.0) and DensMap (33.6). Neither MKL[1] or DMKL[1] manage to improve the topological preservation of their combined methods. In addition, the degree-3 kernel methods yield a worse topological preservation than their degree-1 counterparts shown in Figs. 9 and 10. This is reasonable since mapping the original points to a higher dimensional space distorts their topology. The lowest topological preservation is attained by NetDR^1_{e2eD} (12.3) and SliseMap (13.4). Notice that the end-to-end versions of the proposed network model yield scores significantly lower than when the encoders are trained before the FFN metamodel.

In order to determine the clustering capabilities of the proposed models and the other tested methods, we also computed their cluster induction measures. They are shown in Table 4. In this case, the proposed discriminating model, NetDR^1_D, yields the best V measure (0.725), significantly higher than ParametricUMAP (0.632) and SliseMap (0.631). Again, degree-3 kernel methods are not superior to their degree-1 counterparts, and the end-to-end versions of the proposed network model yield scores significantly lower than with the proposed methodology.

The same experiments were also run upon the 1,440 images of COIL-20.

Table 5 shows the obtained R_{NX} scores. Again, the proposed NetDR^1_T yields the best AUC score (48.9) followed by ParametricUMAP (44.6) and DensMap (44.5). Neither MKL[1] or DMKL[1] manage to improve the topological preservation of their combined methods. In addition, the degree-3 kernel methods yield a worse topological preservation than their degree-1 counterparts shown in Figs. 9 and 10. This is reasonable since mapping the original points to a higher dimensional space distorts their topology. The lowest topological preservation is attained by SliseMap (14.9).

In turn, Table 5 shows the clustering capabilities of the tested methods on COIL-20. In this case, the proposed discriminating model, NetDR^1_D, also yields the best V measure (0.960). The next best performance was obtained by LDA (0.883) and ParametricUMAP (0.882).

The same experiments were also run upon the 12,800 images of LEGO. Table 6 shows the obtained R_{NX} scores. NetDR^1_T keeps providing the best AUC score (39.1), followed by ParametricUMAP (36.9) and DensMap (36.3). The lowest topological

Table 6 R_{NX} Cluster induction measures of tested DR methods applied to the LEGO dataset

Method	R_{NX}	V	Method	R_{NX}	V
PCA	28.1	0.455	GraphEncoder	10.6	0.446
CMDS	27.6	0.510	MKL[1]	24.7	0.463
LE	24.6	0.600	DMKL[1]	16.1	0.444
LLE	18.8	0.425	FA	23.1	0.454
KPCA[3]	23.6	0.464	LDA	15.7	0.691
KCMDS[3]	19.9	0.566	NetDR^1_{e2eT}	19.8	0.430
KLE[3]	24.8	0.478	NetDR^1_{e2eD}	14.7	0.475
KLLE[3]	17.2	0.473	NetDR^1_{T}	**39.1**	0.600
ISOMAP	27.7	0.497	NetDR^1_{D}	16.6	***0.851***
UMAP	29.7	0.525	DensMAP	36.3	0.640
TriMap	35.6	0.604	SliseMap	8.5	0.487
ScaledPCA	21.9	0.555	ParametricUMAP	36.9	0.581

Table 7 R_{NX} Cluster induction measures of tested DR methods applied to the USPS dataset

Method	R_{NX}	V	Method	R_{NX}	V
PCA	22.2	0.439	GraphEncoder	23.7	0.125
CMDS	25.5	0.408	MKL[1]	19.2	0.577
LE	22.1	0.600	DMKL[1]	19.1	0.565
LLE	20.1	0.211	FA	21.7	0.359
KPCA[3]	17.7	0.412	LDA	13.6	0.604
KCMDS[3]	16.3	0.244	NetDR^1_{e2eT}	22.1	0.478
KLE[3]	23.5	0.712	NetDR^1_{e2eD}	20.3	0.565
KLLE[3]	20.4	0.342	NetDR^1_{T}	**34.4**	0.869
ISOMAP	23.9	0.600	NetDR^1_{D}	18.5	***0.923***
UMAP	32.4	0.766	DensMap	33.8	0.851
TriMap	32.7	0.852	SliseMap	15.6	0.737
ScaledPCA	22.5	0.386	ParametricUMAP	32.9	***0.922***

preservation is attained by SliseMap (8.5). Table 6 shows that NetDR^1_{D} gives the best V measure (0.851), which is significantly higher than the other tested techniques. LDA (0.691) yields the second best result.

Table 7 shows the obtained R_{NX} and V measure scores obtained in the experiments carried out on the USPS dataset. The best topological preservation was achieved by NetDR^1_{T} (34.4), while DensMap achieves a close performance of 33.8. On the other hand, the lowest score was achieved by LDA (13.6). In terms of cluster induction, NetDR^1_{D} and ParametricUMAP have the best performances, 0.923 and 0.922, respectively.

Table 8 shows the results for the *Medical-MNIST* dataset. In this case, NetDR^1_{T} also achieves a significant result in terms of structure preservation (45.9), followed by TriMap (38.8). However, the results for cluster induction show that TriMap yields the lowest performance (0.074), whereas NetDR^1_{D} achieves the best discrimination (0.957).

Table 9 presnts results for the ISL dataset. The best R_{NX} results were obtained by NetDR^1_{T} (48.8), ParametricUMAP (46.8) and TriMap (45.8), respectively. That order

Table 8 R_{NX} Cluster induction measures of tested DR methods applied to the Medical-MNIST dataset

Method	R_{NX}	V	Method	R_{NX}	V
PCA	32.4	0.804	GraphEncoder	37.6	0.635
CMDS	29.6	0.659	MKL[1]	22.8	0.825
LE	30.5	0.906	DMKL[1]	21.3	0.744
LLE	22.2	0.939	FA	26.3	0.758
KPCA[3]	29.1	0.651	LDA	23.1	0.945
KCMDS[3]	24.9	0.447	NetDR^1_{e2eT}	26.5	0.535
KLE[3]	25.7	0.852	NetDR^1_{e2eD}	23.1	0.662
KLLE[3]	21.3	0.812	NetDR^1_{T}	**45.9**	0.890
ISOMAP	26.9	0.780	NetDR^1_{D}	21.0	**0.957**
UMAP	33.4	0.911	DensMAP	34.8	0.924
TriMap	38.8	0.074	SliseMap	26.3	0.931
ScaledPCA	31.3	0.791	ParametricUMAP	36.8	0.938

[1] Footnote content not provided. [3] Footnote content not provided

Table 9 R_{NX} Cluster induction measures of tested DR methods applied to the ISL dataset

Method	R_{NX}	V	Method	R_{NX}	V
PCA	29.2	0.640	GraphEncoder	22.2	0.547
CMDS	32.6	0.650	MKL[1]	24.6	0.654
LE	34.5	0.779	DMKL[1]	23.8	0.617
LLE	26.0	0.580	FA	26.7	0.803
KPCA[3]	30.1	0.655	LDA	17.2	0.784
KCMDS[3]	26.6	0.525	NetDR^1_{e2eT}	26.4	0.615
KLE[3]	21.5	0.530	NetDR^1_{e2eD}	21.3	0.711
KLLE[3]	21.0	0.533	NetDR^1_{T}	**48.8**	0.839
ISOMAP	27.4	0.589	NetDR^1_{D}	29.0	***0.959***
UMAP	42.7	0.802	DensMAP	40.5	0.871
TriMap	45.8	0.894	SliseMap	14.2	0.765
ScaledPCA	29.4	0.657	ParametricUMAP	46.8	0.919

[1] footnote description here - marks omitted; descriptive footnote missing

is repeated for the V measure: NetDR^1_{D} (0.959), ParametricUMAP (0.919) and TriMap (0.894).

Finally, Table 10 presents the results for the *Imagenette* dataset. No preprocessing, such as background removal or image segmentation was applied. The latter explains the relatively low performances obtained in general. In terms of R_{NX}, NetDR^1_{T} yields the largest score (21.8), followed by ParametricUMAP (20.8). For cluster induction, NetDR^1_{D} yields the largest V measure (0.863), followed by SliseMAP (0.831).

Table 10 R_{NX} Cluster induction measures of tested DR methods applied to the *Imagenette* dataset

Method	R_{NX}	V	Method	R_{NX}	V
PCA	21.0	0.219	GraphEncoder	17.3	0.194
CMDS	19.6	0.223	MKL[1]	16.2	0.208
LE	17.6	0.237	DMKL[1]	15.6	0.211
LLE	18.9	0.244	FA	19.9	0.232
KPCA[3]	18.8	0.236	LDA	7.74	0.278
KCMDS[3]	17.6	0.237	NetDR^1_{e2eT}	18.9	0.238
KLE[3]	17.1	0.243	NetDR^1_{e2eD}	9.3	0.827
KLLE[3]	16.4	0.221	NetDR^1_{T}	**21.8**	0.236
ISOMAP	18.4	0.212	NetDR^1_{D}	10.5	***0.863***
UMAP	20.7	0.222	DensMAP	20.1	0.230
TriMap	19.2	0.245	SliseMap	3.8	0.831
ScaledPCA	20.2	0.224	ParametricUMAP	20.8	0.246

7.3 Additional experiments

Three sets of experiments were carried out to further assess the topology and performance of the proposed model. The first experiment aimed at finding the optimal number of layers of the FFN metamodel described in Sect. 5.2. The second experiment evaluated the performance of the proposed dimensionality reduction model for different combinations of spectral methods. The third experiment tested the proposed model for 4D and 8D embeddings in addition to the 2D embeddings shown in the previous sections. The last experiment compared the online capabilities of the proposed model against the best online versions of spectral methods.

7.3.1 Optimal topology of the FFN metamodel

As described in Sect. 5.2, M encoders associated with corresponding DR spectral methods are integrated by a metamodel defined as a multilayer feedforward neural network (FFN).

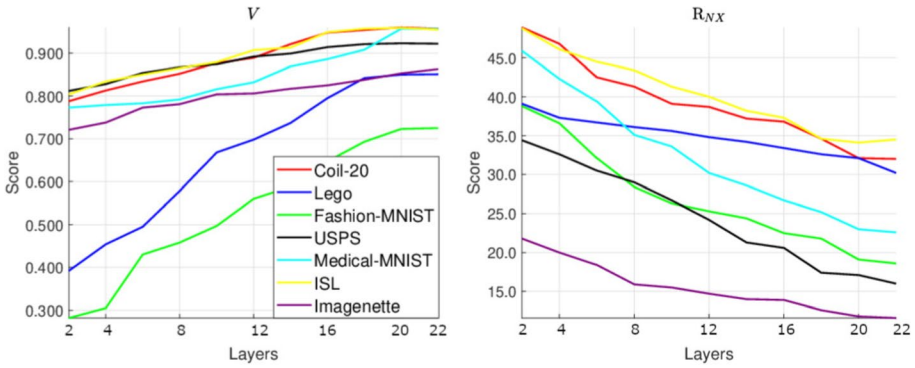

Fig. 12 Evaluation of R_{NX} score and V measure for different depths of the FFN module

The current experiments aimed at finding the optimal depth (i.e., number of fully-connected layers) of the FFN module according to the topology depicted in Fig. 5. Figure 12 shows both the R_{NX} score and V measure for the seven tested datasets. The X-axis represents a different number of layers defined by increasing values of L by 2 from 2 to 22. The Y-axis represents the score obtained for each measure.

Similarly to the experiments conducted in the previous sections, NetDR^1_T integrates ε_{KPCA^1}, ε_{KCMDS^1}, ε_{KLLE^1} and ε_{KLE^1} with the FFN module trained as depicted in Fig. 4, whereas NetDR^1_D integrates the same encoders with the FFN trained as shown in Fig. 7. Additionally, the FFN in NetDR^1_D applies ReLU activation functions, whereas Mish functions are applied in NetDR^1_T as they were assessed to yield superior performance for topological preservation.

Figure 12 shows that the best R_{NX} score (i.e., topological preservation) for the seven datasets was attained by NetDR^1_T with the lowest depth of two layers as shown in Fig. 3. As expected, the best V measure (i.e., cluster induction) was attained by NetDR^1_D. However, the optimal number of layers in this case depends on the chosen dataset: 20 layers for *Fashion-MNIST*, 16 layers for *COIL-20*, 18 layers for *LEGO*, 16 layers for *USPS*, 20 layers for *Medical-MNIST*, 22 layers for *Imagenette*, and 16 layers for *ISL*. These were the topologies of the FFN module in the experiments reported in Sect. 7.2 for NetDR^1_D.

7.3.2 Alternative combinations of DR methods and effect of training epochs

Although the experiments conducted in Sect. 7.2 tested the proposed neural model on a combination of four spectral methods (KPCA^1+KCMDS^1+KLE^1+KLLE^1), any other combination is possible. For example, Table 11 shows the R_{NX} scores and V measures obtained with both NetDR^1_T and NetDR^1_D for COIL-20 by considering additional combinations: KPCA^1+KLE^1, KCMDS^1+KLLE^1, KCMDS^1+KPCA^1, KLE^1+KLLE^1, KCMDS^1+KPCA^1 +KLE^1, and KCMDS^1+KLE^1+KLLE^1.

Furthermore, Table 11 also shows the effect of the number of epochs on the performance measures when training NetDR^1_T (see Fig. 4) and NetDR^1_D (see Fig. 7). Notice that NetDR^1_D yields the best cluster induction after 200 training epochs. In this case, the minibatch size was set to $b = 64$. However, NetDR^1_T yields the best topological preservation between 100 and 200 training epochs and the largest minibatch size allowed by RAM in order to process meaningful Euclidean distance matrices: b=5,000 for *Fashion-MNIST*, b=1,440 for *COIL-20*, b=3,000 for *LEGO*, b=5,000 for *USPS*, b=5,000 for *Medical-MNIST* and b=5,000 for *ISL*. All encoders were trained with b=64.

7.3.3 Ablation study

We have carried out an ablation study to assess the contribution of the different stages of the proposed technique. In particular, Figs. 13 and 14 show quantitative results of three alternative configurations applied to the seven tested datasets.

The first configuration (blue bars), referred to as ε_{DR} (encoder), only applies a single neural encoder to reduce the dimensionality of the input high-dimensional data, as described in Sect. 5.1. We separately run the four encoders considered in this work: ε_{KPCA^1}, ε_{KCMDS^1}, ε_{KLLE^1} and ε_{KLE^1}. Only the best result is shown for each dataset. The second configuration (orange bars), referred to as FFN (direct), only applies the FFN metamodel described in Sect. 5.2 to the original high-dimensional data, without applying any

Table 11 R_{NX} score, V measure and run time for alternative combinations of *DR* methods and training iterations of NetDR upon COIL-20

NetDR Combination	Measure	Number of training epochs							
		50		100		200		300	
		NetDR^1_T	NetDR^1_D	NetDR^1_T	NetDR^1_D	NetDR^1_T	NetDR^1_D	NetDR^1_T	NetDR^1_D
PCA/KPCA[1]	V	0.761	0.882	0.815	0.924	0.829	***0.925***	0.836	0.934
LE/KLE[1]	R_{NX}	42.0	29.7	**44.9**	31.6	44.5	32.4	42.3	33.0
	Time (min)	1.4	0.6	2.92	1.90	3.78	3.17	5.20	4.0
CMDS/KCMDS[1]	V	0.766	0.907	0.811	0.930	0.821	***0.934***	0.830	0.943
LLE/KLLE[1]	R_{NX}	41.7	29.6	44.4	30.8	**45.0**	32.1	43.4	33.5
	Time (min)	1.3	0.62	2.92	1.91	3.79	3.18	5.019	4.01
CMDS/KCMDS[1]	V	0.758	0.876	0.814	0.917	0.819	***0.920***	0.833	0.929
PCA/KPCA[1]	R_{NX}	41.8	30.4	**44.8**	31.1	43.6	31.9	41.4	33.2
	Time (min)	1.4	0.6	2.94	1.90	3.80	3.19	5.18	4.0
LE/KLE[1]	V	0.775	0.919	0.808	0.952	0.816	***0.956***	0.828	0.954
LLE/KLLE[1]	R_{NX}	42.6	29.7	44.1	31.8	**44.5**	32.6	43.0	33.3
	Time (min)	1.4	0.6	2.93	1.92	3.81	3.19	5.19	4.01
CMDS/KCMDS[1]	V	0.769	0.895	0.793	0.955	0.824	**0.955**	0.835	0.963
PCA/KPCA[1]	R_{NX}	41.6	32.1	**45.3**	32.3	44.8	33.4	43.2	34.6
LE/KLE[1]	Time(min)	1.8	0.71	3.11	2.10	4.03	3.41	5.41	4.21
LLE/KLLE[1]	V	0.779	0.915	0.823	0.941	0.830	**0.958**	0.835	0.968
LE/KLE[1]	R_{NX}	42.6	31.2	**46.6**	32.4	45.0	31.9	43.3	34.7
CMDS/KCMDS[1]	Time(min)	1.8	0.71	3.09	2.11	4.01	3.42	5.40	4.21
LLE/KLLE[1]	V	0.781	0.902	0.811	0.949	0.816	***0.960***	0.836	0.964
LE/KLE[1]	R_{NX}	42.2	30.1	**48.9**	31.3	45.0	30.6	43.8	35.6
CMDS/KCMDS[1]	Time(min)	2.3	0.9	3.31	2.15	4.26	3.68	5.71	4.31
PCA/KPCA[1]									

previous embedding through neural encoders. The topological version of the metamodel (Sect. 5.2.1) was used for the results shown in Fig. 13, whereas the discriminant version (Sect. 5.2.2) was used for the results in Fig. 14. Finally, the last configuration (yellow bars) is the full proposed model, that is, the four neural encoders plus the FFN metamodel. The topological version, NetDR^1_T, was considered in Fig. 13, whereas the discriminant version, NetDR^1_D, was used in Fig. 14.

The results shown in both figures indicate that the lowest performances correspond to the individual encoders. The direct application of the FFN metamodel to the input high-dimensional data yields better performances, particularly for cluster induction, whose process is guided in a supervised way. Finally, the full proposed model gives the best overall performance. This shows that the synergistic combination of neural encoders trough the proposed neural metamodel is advantageous.

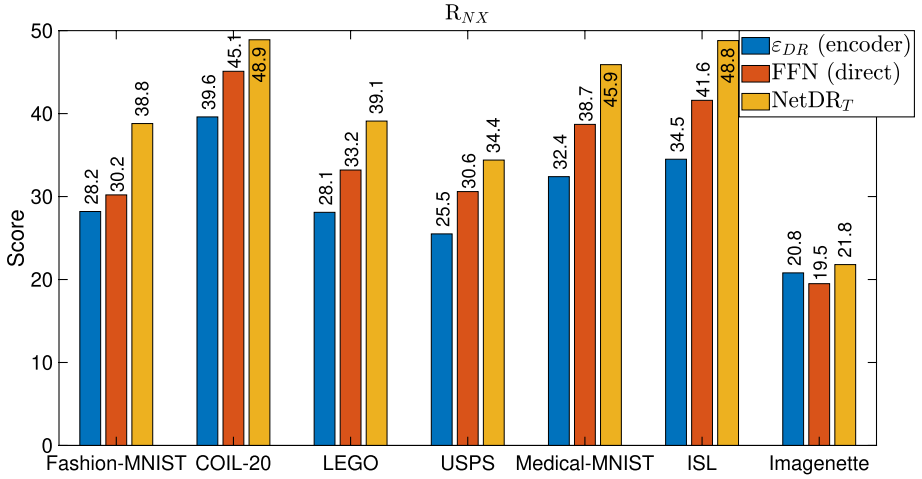

Fig. 13 Topological results for the proposed model and two ablated configurations

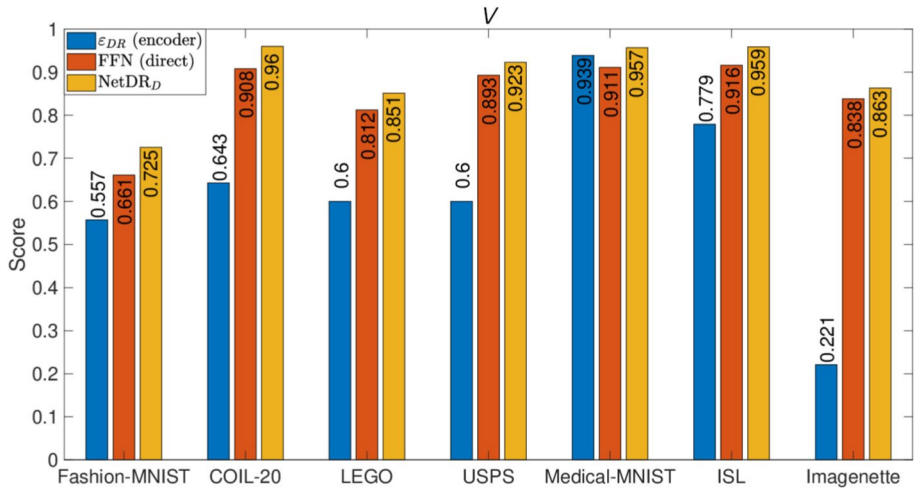

Fig. 14 Cluster induction results for the proposed model and two ablated configurations

7.3.4 Embedding performance for higher dimensions

The experiments presented in Sect. 7.2 were conducted for a 2D embedding, $d = 2$. Table 12 also shows the performance of both NetDR^1_T and NetDR^1_D for higher low-dimensional spaces ($d = 4$ and $d = 8$). These results show that the performance of NetDR^1_T for topological preservation increases as the embedding dimension increases. Conversely, the higher the embedding dimension, the lower the cluster induction score V for the NetDR^1_D.

Table 12 R_{NX} score and V measure for NetDR by considering embeddings to 4 and 8 dimensions

d	Dataset	NetDR	R_{NX}	V
4	*COIL-20*	NetDR^1_T	53.3	0.575
		NetDR^1_D	38.0	0.588
	LEGO	NetDR^1_T	46.8	0.481
		NetDR^1_D	29.7	0.483
	Fashion-MNIST	NetDR^1_T	40.2	0.398
		NetDR^1_D	30.2	0.391
	USPS	NetDR^1_T	40.4	0.405
		NetDR^1_D	30.3	0.432
	Medical-MNIST	NetDR^1_T	49.6	0.325
		NetDR^1_D	32.0	0.344
	ISL	NetDR^1_T	52.0	0.516
		NetDR^1_D	37.6	0.518
	Imagenette	NetDR^1_T	29.6	0.355
		NetDR^1_D	18.7	0.392
8	*COIL-20*	NetDR^1_T	57.1	0.570
		NetDR^1_D	38.5	5.76
	LEGO	NetDR^1_T	48.5	0.479
		NetDR^1_D	32.9	0.483
	Fashion-MNIST	NetDR^1_T	42.0	0.397
		NetDR^1_D	32.6	0.381
	USPS	NetDR^1_T	44.4	0.405
		NetDR^1_D	34.0	0.391
	Medical-MNIST	NetDR^1_T	49.8	0.326
		NetDR^1_D	32.8	0.306
	ISL	NetDR^1_T	55.8	0.511
		NetDR^1_D	41.1	0.516
	Imagenette	NetDR^1_T	35.4	0.338
		NetDR^1_D	26.1	0.376

[1] Footnotes might be provided here

7.3.5 Online performance

The online capabilities of the proposed dimensionality reduction neural models were finally assessed by comparing them against the best online versions of classical spectral methods introduced in Sect. 3.3: IKPCA, ICMDS, ILE, and ILLE. In order to generate new data points compatible with the original manifolds, a DCGAN network (Radford et al. 2015) was trained for *Fashion-MNIST*, *COIL-20* and *LEGO*. Each DCGAN was able to generate synthetic images that mimicked the original images of its associated dataset, although being sufficiently different due to random rotations and noise. All methods were initially applied to the original datasets as described in the previous

Table 13 R_{NX} and V-measure scores applied to NetDR$_T$, NetDR$_D$ and the incremental versions of classical sepctral methods for GAN-generated variations of the datasets: *COIL-20*, *LEGO* and *Fashion-MNIST*, as well as for the *USPS*, *Medical-MNIST*, *Imagenette*, and *ISL* validation datasets

Dataset	Measure	IKPCA	ICMDS	ILE	ILLE	NetDR^1_T	NetDR^1_D
COIL-20	V	0.603	0.549	0.511	0.623	0.769	0.918
	R_{NX}	37.8	33.4	27.0	26.3	42.4	27.4
LEGO	V	0.430	0.491	0.455	0.461	0.576	0.818
	R_{NX}	27.7	26.3	22.4	16.9	35.0	13.2
Fashion-MNIST	V	0.410	0.407	0.537	0.501	0.554	0.688
	R_{NX}	25.7	25.3	19.4	16.4	32.9	14.4
USPS	V	0.403	0.385	0.586	0.201	0.806	0.892
	R_{NX}	20.3	22.3	19.8	18.7	31.3	14.6
Medical-MNIST	V	0.776	0.614	0.856	0.893	0.848	0.925
	R_{NX}	28.7	27.1	27.2	19.4	41.6	18.4
ISL	V	0.592	0.587	0.708	0.514	0.794	0.903
	R_{NX}	26.8	29.8	31.1	23.6	44.2	26.2
Imagenette	V	0.203	0.209	0.225	0.216	0.208	0.827
	R_{NX}	20.2	18.5	16.8	18.1	19.3	8.8

sections. Then, their performance was measured after applying them to the new GAN-generated datasets. However, we used their own validation datasets for the *USPS*, *Medical-MNIST*, *Imagenette*, and *ISL* datasets. Table 13 shows the obtained results in terms of R_{NX} and V measures. The behavior of tested measures is similar with respect to the results reported in the previous section. NetDR^1_T had the best topological performance (R$_{NX}$ score) for *COIL-20* (42.4), *LEGO* (35.0), *Fashion-MNIST* (32.9), *USPS* (31.3), *Medical-MNIST* (41.6), *ISL* (44.2), and *Imagenette* (19.3). In turn, NetDR^1_D gave the best clustering performance (*V*-measure): *COIL-20* (0.918), *LEGO* (0.818), *Fashion-MNIST* (0.688), *USPS* (0.892), *Medical-MNIST* (0.925), *ISL* (0.903), and *Imagenette* (0.827).

7.3.6 Computational complexity

We have applied the open-source package *Ptflops* [93] for estimating the computational complexity and number of trainable parameters of the different neural models that have been proposed in this work. The computational complexity is estimated in terms of MACs

Table 14 Computational complexity and number of trainable parameters of the different neural models of the proposed technique (32×32 images)

Model	Computational complexity (MACs)	Trainable parameters
Encoder ϵ_{DR}	46.61×10^6	73.4×10^3
FFN topological	214	178
FFN discriminant	1.41×10^6	1.46×10^6
NetDR$_T$	46.62	73.6×10^3
NetDR$_D$	48.02×10^6	1.53×10^6

(Multiply-Accumulate Calculations). One MAC is roughly equivalent to two FLOPs (Floating Point Operations). The results are presented in Table 14.

8 Conclusions

An online dimensionality reduction (DR) scheme that synergistically combines classical spectral methods and uniform manifold approximation with deep neural networks has been proposed. Given a target dataset whose dimensionality is to be reduced, different linear and non-linear DR spectral methods are first applied. The original data points are then projected to the sought low-dimensional space according to the embedding induced by every spectral method, which is optimized by approximating points of similar characteristics based on a probability distribution of group membership. The set of projected points generated by each spectral method along with the original data points and their respective optimization are utilized to train a deep encoder in a self-supervised way. Those encoders are able to project new data points in an online manner, something that classical spectral methods cannot perform due to their offline nature. A stacking ensemble learning approach is then applied for fusing the low-dimensional outputs of the different available encoders through a metamodel implemented as a multilayer feed-forward network. Two metamodel topologies have been proposed depending on whether the final aim is to enforce topological preservation or cluster induction. For topological preservation, the metamodel is unsupervised. A shallow network is trained with a composite cost function that preserves the local topology by using a neighborhood network, and the global topology by means of Euclidean distance matrices associated with both the original and low-dimensional points. For cluster induction, the metamodel is supervised. A deep network is trained by means of a cross-entropy cost function.

Experimental results with widely-used image datasets have shown that the proposed deep learning scheme yields significantly higher performance measures than the original spectral methods and state-of-the-art dimensionality reduction approaches in terms of both topological preservation and cluster induction. The proposed methodology also exhibits a much higher performance than when the encoders and the FFN metamodel that combines them are directly trained from scratch in an end-to-end manner.

The main contributions of the proposed technique are summarized below:

- *Online Learning of Embeddings*: The first phase of NetDR allows online learning of any embedding generated by a dimension reduction method. This allows new data to be projected based on the learned model. This approach improves computational efficiency, optimizing the use of memory and reducing processing time for large volumes of data.
- *Synergistic Combination of Methods*: The second phase of NetDR synergistically combines different dimension reduction methods to achieve a holistic embedding that incorporates the features of its precursor methods. This combination capability provides a more robust and representative embedding, thus improving the quality of data representations.
- *Versatility*: NetDR comprises two specialized variations: NetDR$_T$, which optimizes topology preservation, and NetDR$_D$, which focuses on cluster induction. This allows users to choose the variation that best suits their specific needs, making NetDR a versatile and adaptable method for complex data analysis.

- *Efficient Management of Complex Data*: In contexts of dynamic and constantly changing data, the ability to learn and update embeddings in an online manner is crucial. NetDR addresses this need by providing an efficient mechanism to keep models up-to-date without extensive recalculations. This feature is particularly useful for big data applications, and in scenarios where the efficient processing of new data is critical, such as real-time surveillance and environmental monitoring.
- *Improved Data Representation*: The combination of multiple dimensionality reduction methods makes it possible to capture different aspects of data structure, resulting in more complete and accurate representations. This holistic approach improves data interpretation and analysis, facilitating tasks such as pattern discovery.

Immediate work aims at exploring new topologies for the multilayer feed-forward network corresponding to the cluster inducing metamodel (i.e., NetDR_D), as the optimal number of layers currently depends on the target dataset. We also aim at adapting the proposed DR scheme to other multidimensional problems beyond images. In the medium term, we also aim to explore the application of state-of-the-art AI models, such as SpectralGPT (Hong et al., 2023), to dimensionality reduction. SpectralGPT has been shown to optimize tasks related to scene classification, semantic segmentation, and detection and discovery of complex spectral patterns. Indeed, spectral image analysis is a promising research field. Spectral variability can originate from several factors, such as atmospheric conditions, changes in illumination, and intrinsic characteristics of materials. A recent study (Hong et al., 2018) addresses this problem by introducing an improved linear mixing model to handle spectral variability.

NetDR can also be applied as a preprocessing stage to mitigate the effects of noise and variabilities. Techniques such as PCA are known for their ability to reduce noise by capturing the principal components that represent the greatest variability in the data. However, nonlinear techniques, such as LE and LLE can be more sensitive to noise. Our neural ensemble can be trained to assign appropriate weights to each technique, hence optimizing dimensionality reduction in the presence of variability. Finally, future directions in AI, such as advanced foundation models, transferable learning, multimodal data fusion, edge computing, and model interpretability, offer exciting opportunities for expanding the capabilities of NetDR.

Author contributions J.A.: conceptualization, data curation, investigation, software, validation, visualization, writing—original draft. M.G.: conceptualization, investigation, methodology, software, supervision, writing—review and editing. D.P.: Conceptualization, funding acquisition, project administration, resources, supervision. All authors reviewed the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability Fashion-MNIST: https://www.kaggle.com/datasets/zalando-research/fashionmnist Columbia object image library (coil-20): https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php Lego bricks dataset: https://www.kaggle.com/datasets/joosthazelzet/lego-brick-images Medical-MNIST: https://www.kaggle.com/datasets/andrewmvd/medical-mnist/data USPS: https://www.kaggle.com/datasets/bistaumanga/usps-dataset Indian Sign Language (ISL): https://www.kaggle.com/datasets/prekshapalva/indian-sign-language.

Declarations

Conflict of interest The authors declare no competing interests.

References

Aiolli, F., & Donini, M. (2015). EasyMKL: A scalable multiple kernel learning algorithm. *Neurocomputing, 169*, 215–224.

Alswaitti, M., Siddique, K., Jiang, S., Alomoush, W., & Alrosan, A. (2022). Dimensionality reduction, modelling, and optimization of multivariate problems based on machine learning. *Symmetry, 14*(7), 1282.

Amid, E., & Warmuth, M.K. (2019). TriMap: Large-scale dimensionality reduction using triplets. arXiv preprint arXiv:1910.00204.

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. United Kingdom: Cambridge University Press.

Basalaj, W. (1999). Incremental multidimensional scaling method for database visualization. In *Visual data exploration and analysis VI* (vol. 3643, pp. 149–158). International Society for Optics and Photonics.

Belanche Muñoz, L. A. (2013). Developments in kernel design. In *ESANN 2013 proceedings: European symposium on artificial neural networks, computational intelligence and machine learning: Bruges (Belgium)* (pp. 369–378).

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation, 15*(6), 1373–1396.

Bishop, C. M., Svensén, M., & Williams, C. K. (1998). GTM: The generative topographic mapping. *Neural Computation, 10*(1), 215–234.

Björklund, A., Mäkelä, J., & Puolamäki, K. (2023). SLISEMAP: Supervised dimensionality reduction through local explanations. *Machine Learning, 112*(1), 1–43.

Blockeel, H., De Raedt, L., & Ramon, J. (2000). Top-down induction of clustering trees. arXiv preprint cs/0011032.

Borg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Science & Business Media: Springer.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Carmeli, C., De Vito, E., & Toigo, A. (2006). Vector valued reproducing kernel Hilbert spaces of integrable functions and mercer theorem. *Analysis and Applications, 4*(04), 377–408.

Chavula, C., & Suleman, H. (2017). Morphological cluster induction of bantu words using a weighted similarity measure. In *Proceedings of the South African institute of computer scientists and information technologists* (pp. 1–9).

Chin, T.-J., & Suter, D. (2007). Incremental kernel principal component analysis. *IEEE Transactions on Image Processing, 16*(6), 1662–1674.

Denisko, D., & Hoffman, M. M. (2018). Classification and interaction in random forests. *Proceedings of the National Academy of Sciences, 115*(8), 1690–1692.

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC Press.

Eskandarnia, E., Al-Ammal, H. M., & Ksantini, R. (2022). An embedded deep-clustering-based load profiling framework. *Sustainable Cities and Society, 78*, 103618.

Fresca, S., & Manzoni, A. (2022). POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDES by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering, 388*, 114181.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Gönen, M., & Alpaydın, E. (2011). Multiple kernel learning algorithms. *The Journal of Machine Learning Research, 12*, 2211–2268.

Gönen, M., Alpaydın, E., & Bach, F. (2011). Multiple kernel learning algorithms. *The Journal of Machine Learning Research, 12*, 2181–2238.

Gou, J., Yang, Y., Yi, Z., Lv, J., Mao, Q., & Zhan, Y. (2020). Discriminative globality and locality preserving graph embedding for dimensionality reduction. *Expert Systems with Applications, 144*, 113079.
Guo, T., Yu, K., Aloqaily, M., & Wan, S. (2022). Constructing a prior-dependent graph for data clustering and dimension reduction in the edge of AIoT. *Future Generation Computer Systems, 128*, 381–394.
Hallgren, F., & Northrop, P. (2018). Incremental kernel PCA and the Nystrom method. arXiv preprint arXiv:1802.00043 (2018)
Ham, J., Lee, D., Mika, S., & Schölkopf, B. (2004). *A kernel view of the dimensionality reduction of manifolds.*[SPACE]https://doi.org/10.1145/1015330.1015417
Ham, J., Lee, D. D., Mika, S., & Schölkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first international conference on machine learning* (p. 47). ACM.
Hazelzet, J. (2019). Images of LEGO bricks dataset. https://www.kaggle.com/joosthazelzet/lego-brick-images
Hinton, G. E., & Roweis, S. T. (2002). Stochastic neighbor embedding. In *Advances in neural information processing systems* (pp. 833–840).
Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science (New York), 313*, 504–507. https://doi.org/10.1126/science.1127647
Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504–507.
Hong, D., Zhang, B., Li, X., Li, Y., Li, C., Yao, J., Yokoya, N., Li, H., Jia, X., & Plaza, A., et al. (2023). Spectralgpt: Spectral foundation model. arXiv preprint arXiv:2311.07113
Hong, D., Yokoya, N., Chanussot, J., & Zhu, X. X. (2018). An augmented linear mixing model to address spectral variability for hyperspectral unmixing. *IEEE Transactions on Image Processing, 28*(4), 1923–1938.
Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology, 24*(6), 417–441. https://doi.org/10.1037/h0071325
Howard, J., et al. (2020). Imagenette. **2**. https://github.com/fastai/imagenette
Huang, D., Jiang, F., Li, K., Tong, G., & Zhou, G. (2022). Scaled PCA: A new approach to dimension reduction. *Management Science, 68*(3), 1678–1695.
Huber, P. J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics, 35*(1), 73–101.
Hull, J. J. (2023). Hand-written digits USPS dataset. Accessed on November 27. https://www.kaggle.com/datasets/andrewmvd/medical-mnist/data
Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 16*(5), 550–554.
Izenman, A. J. (2013). Linear discriminant analysis. In *Modern multivariate statistical techniques* (pp. 237–280). Springer.
Jiang, L., Fang, X., Sun, W., Han, N., & Teng, S. (2023). Low-rank constraint based dual projections learning for dimensionality reduction. *Signal Processing, 204*, 108817.
Jia, P., Yin, J., Huang, X., & Hu, D. (2009). Incremental Laplacian eigenmaps by preserving adjacent information between data points. *Pattern Recognition Letters, 30*(16), 1457–1463.
Joshi, P., & Kulkarni, P. (2012). Incremental learning: Areas and methods-a survey. *International Journal of Data Mining & Knowledge Management Process, 2*(5), 43.
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. http://arxiv.org/abs/1412.6980
Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
Kohonen, T. (2001). Self-organizing maps of massive databases. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications, 9*(4), 179–186.
Kouropteva, O., Okun, O., & Pietikäinen, M. (2005). Incremental locally linear embedding algorithm. In *Scandinavian conference on image analysis* (pp. 521–530). Springer.
Lauriola, I., Polato, M., & Aiolli, F. (2017). Radius-margin ratio optimization for dot-product Boolean kernel learning. In *International conference on artificial neural networks*.
Lauriola, I., Polato, M., & Aiolli, F. (2018). The minimum effort maximum output principle applied to multiple kernel learning. In *European symposium on artificial neural networks*.
Lee, J. A., Archambeau, C., & Verleysen, M., et al. (2003). Locally linear embedding versus isotop. In *ESANN* (pp. 527–534).
Lee, J. A., Peluffo-Ordonez, D. H., & Verleysen, M. (2015). Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. *Neurocomputing, 169*, 246–261.

Lee, J. A., Renard, E., Bernard, G., Dupont, P., & Verleysen, M. (2013). Type 1 and 2 mixtures of Kullback-Leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing, 112*, 92–108.

Lee, J. A., & Verleysen, M. (2007). *Nonlinear Dimensionality Reduction*. Science & Business Media: Springer.

Lee, J. A., & Verleysen, M. (2009). Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing, 72*(7), 1431–1443.

Luo, Z., Xu, C., Zhang, Z., & Jin, W. (2021). A topology-preserving dimensionality reduction method for single-cell RNA-SEQ data using graph autoencoder. *Scientific Reports, 11*(1), 20028.

Masci, J., Meier, U., Cireşan, D., & Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In T. Honkela, W. Duch, M. Girolami, & S. Kaski (Eds.), *Artificial neural networks and machine learning–ICANN 2011* (pp. 52–59). Berlin: Springer.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426

Menaga, D., & Revathi, S. (2021). Probabilistic principal component analysis (PPCA) based dimensionality reduction and deep learning for cancer classification. In *Intelligent computing and applications: Proceedings of ICICA 2019* (pp. 353–368). Springer.

Misra, D. (2020). Mish: A self regularized non-monotonic neural activation function. In *The British machine vision conference*.

Molina, L. C., Belanche, L., & Nebot, A. (2002). Feature selection algorithms: a survey and experimental evaluation. In *2002 IEEE international conference on data mining, 2002. Proceedings* (pp. 306–313). https://doi.org/10.1109/ICDM.2002.1183917

Narayan, A., Berger, B., & Cho, H. (2020). Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability. bioRxiv, 2020-05.

Nene, S. A., Nayar, S. K., & Murase, H. (1996). Columbia object image library (coil-20). Department of Computer Science, Columbia University, New York. http://www. cs. columbia. edu/CAVE/coil-20. html **62**

Özdenizci, O., & Erdoğmuş, D. (2021). Stochastic mutual information gradient estimation for dimensionality reduction networks. *Information Sciences, 570*, 298–305.

Pai, G., Bronstein, A., Talmon, R., & Kimmel, R. (2022). Deep isometric maps. *Image and Vision Computing, 123*, 104461.

Polanco Lozano, A. (2017). Medical MNIST. Accessed on November 27, 2023. https://www.kaggle.com/datasets/andrewmvd/medical-mnist/data

Preksha, A. (2023). Indian sign language. Accessed on November 27. https://www.kaggle.com/datasets/prekshapalva/indian-sign-language

Qu, H., Li, L., Li, Z., & Zheng, J. (2021). Supervised discriminant Isomap with maximum margin graph regularization for dimensionality reduction. *Expert Systems with Applications, 180*, 115055.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics, 22*, 400–407.

Rosenberg, A., & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)* (pp. 410–420).

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326.

Sainburg, T., McInnes, L., & Gentner, T. Q. (2021). Parametric UMAP embeddings for representation and semisupervised learning. *Neural Computation, 33*(11), 2881–2907.

Sanodiya, R. K., & Mathew, J. (2019). A novel unsupervised globality-locality preserving projections in transfer learning. *Image and Vision Computing, 90*, 103802.

Sarveniazi, A. (2014). An actual survey of dimensionality reduction. *American Journal of Computational Mathematics, 04*, 55–72. https://doi.org/10.4236/ajcm.2014.42006

Schölkopf, B., Smola, A. J., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation, 10*(5), 1299–1319.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal, 27*, 379–423.

Shinde, K., Itier, V., Mennesson, J., Vasiukov, D., & Shakoor, M. (2023). Dimensionality reduction through convolutional autoencoders for fracture patterns prediction. *Applied Mathematical Modelling, 114*, 94–113.

Skocaj, D., & Leonardis, A. (2003). Weighted and robust incremental method for subspace learning. In *Proceedings ninth IEEE international conference on computer vision* (pp 1494–15012). https://doi.org/10.1109/ICCV.2003.1238667

Sovrasov, V. (2018). PTFLOPS: A flops counting tool for neural networks in pytorch framework. https://github.com/sovrasov/flops-counter.pytorch
Spearman, C. (1904). "General intelligence,'' objectively determined and measured. *American Journal of Psychology, 15*, 201–293.
Strobl, E. V., & Visweswaran, S. (2013). Deep multiple kernel learning. In *2013 12th international conference on machine learning and applications* (vol. 1, pp. 414–417).
Teknomo, K. (2006). K-means clustering tutorial. *Medicine, 100*(4), 3.
Tenenbaum, J. B., Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science, 290*(5500), 2319.
Tian, F., Gao, B., Cui, Q., Chen, E., & Liu, T. -Y. (2014). Learning deep representations for graph clustering. In *Proceedings of the AAAI conference on artificial intelligence* (vol. 28).
Venna, J., & Kaski, S.: Nonlinear dimensionality reduction as information retrieval. In *Artificial intelligence and statistics* (pp. 572–579).
Wang, Q., Ma, Y., Zhao, K., & Tian, Y. (2020). A comprehensive survey of loss functions in machine learning. *Annals of Data Science*. https://doi.org/10.1007/s40745-020-00253-5
Wang, L., Wang, Z., Qu, H., & Liu, S. (2018). Optimal forecast combination based on neural networks for time series forecasting. *Applied Soft Computing, 66*, 1–17.
Weng, J., Zhang, Y., & Hwang, W.-S. (2003). A fast algorithm for incremental principal component analysis. In J. Liu, Y.-M. Cheung, & H. Yin (Eds.), *Intelligent data engineering and automated learning* (pp. 876–881). Berlin: Springer.
Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241–259.
Wu, L., & Noels, L. (2022). Recurrent neural networks (RNNS) with dimensionality reduction and break down in computational mechanics; application to multi-scale localization step. *Computer Methods in Applied Mechanics and Engineering, 390*, 114476.
Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747
Xu, B., & Zhang, G. (2023). Robust parametric UMAP for the analysis of single-cell data. bioRxiv, 2023-11.
Yao, Y., & Doretto, G. (2010). Boosting for transfer learning with multiple sources. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 1855–1862). IEEE.
Yuan, M., Cai, T. T., et al. (2010). A reproducing kernel Hilbert space approach to functional linear regression. *The Annals of Statistics, 38*(6), 3412–3444.
Zhao, H., Yuen, P. C., & Kwok, J. T. (2006). A novel incremental principal component analysis and its application for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36*(4), 873–886.

Authors and Affiliations

Juan Carlos Alvarado-Pérez[1,2] · Miguel Angel Garcia[3] · Domènec Puig[1]

✉ Juan Carlos Alvarado-Pérez
jcalvaradop@sgc.gov.co

Miguel Angel Garcia
miguelangel.garcia@uam.es

Domènec Puig
domenec.puig@urv.cat

[1] Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Paisos Catalans, 26, 43007 Tarragona, Spain

[2] Servicio Geológico Colombiano (SGC), Pasto, Colombia

[3] Department of Electronic and Communications Technology, Autonomous University of Madrid, Francisco Tomas y Valiente 11, 28049 Madrid, Spain