Gradient-Based Competitive Learning: Theory

Cognitive Computation
https://doi.org/10.1007/s12559-023-10225-5

Gradient-Based Competitive Learning: Theory

Giansalvo Cirrincione[1,2] · Vincenzo Randazzo[3] · Pietro Barbiero[4] · Gabriele Ciravegna[5] · Eros Pasero[3]

Received: 1 March 2023 / Accepted: 12 November 2023

Abstract
Deep learning has been recently used to extract the relevant features for representing input data also in the unsupervised setting. However, state-of-the-art techniques focus mostly on algorithmic efficiency and accuracy rather than mimicking the input manifold. On the contrary, competitive learning is a powerful tool for replicating the input distribution topology. It is cognitive/biologically inspired as it is founded on Hebbian learning, a neuropsychological theory claiming that neurons can increase their specialization by competing for the right to respond to/represent a subset of the input data. This paper introduces a novel perspective by combining these two techniques: unsupervised gradient-based and competitive learning. The theory is based on the intuition that neural networks can learn topological structures by working directly on the transpose of the input matrix. At this purpose, the vanilla competitive layer and its dual are presented. The former is representative of a standard competitive layer for deep clustering, while the latter is trained on the transposed matrix. The equivalence of the layers is extensively proven both theoretically and experimentally. The dual competitive layer has better properties. Unlike the vanilla layer, it directly outputs the prototypes of the data inputs, while still allowing learning by backpropagation. More importantly, this paper proves theoretically that the dual layer is better suited for handling high-dimensional data (e.g., for biological applications), because the estimation of the weights is driven by a constraining subspace which does not depend on the input dimensionality, but only on the dataset cardinality. This paper has introduced a novel approach for unsupervised gradient-based competitive learning. This approach is very promising both in the case of small datasets of high-dimensional data and for better exploiting the advantages of a deep architecture: the dual layer perfectly integrates with the deep layers. A theoretical justification is also given by using the analysis of the gradient flow for both vanilla and dual layers.

Keywords Competitive Hebbian Learning · Deep clustering · Duality theory · Gradient-based clustering · Topology · Unsupervised learning

Introduction

Machine learning can be generally referred as extracting information from noisy data. Depending on the paradigm, either unsupervised or supervised, this problem is called clustering or classification, respectively. Both groups of techniques can be seen as an optimization problem where a loss function is minimized. The oldest and most famous clustering technique is *k*-means [1], which iteratively adapts cluster centroid positions in order to minimize the quantization error. This technique has been extensively used and studied to uncover unknown relations in unsupervised problems. However, its main drawback is the definition of the number of cluster centroids (k) beforehand. This is the same issue as other famous techniques such as Gaussian Mixture Models (GMM) [2] and Neural Gas (NG) [3]. To overcome this limitation, several *incremental* algorithms have been proposed in the literature, where the number of neurons is not fixed but changes over time w.r.t the complexity of the problem at hand. This approach adds a novel unit whether certain conditions are met, e.g., the quantization error is too high or data is too far from the existing neurons; in this sense, the new unit should yield a better quantization of the input distribution. Some examples are the adaptive *k*-means

✉ Giansalvo Cirrincione
exin@u-picardie.fr

✉ Vincenzo Randazzo
vincenzo.randazzo@polito.it

1 Lab. LTI, Université de Picardie Jules Verne, Amiens, France

2 University of South Pacific, Suva, Fiji

3 DET - Department of Electronics and Telecommunications, Politecnico Di Torino, Turin, Italy

4 Computer Laboratory, Cambridge University, Cambridge, UK

5 MAASAI, Inria, Université Cote D'Azur, I3S, CNRS, Nice, France

Published online: 23 November 2023

[4] and the Density Based Spatial Clustering of Applications with Noise (DBSCAN) [5]. Furthermore, unsupervised learning is generally capable of finding groups of samples that are similar under a specific metric, e.g., Euclidean distance. However, it cannot infer the underlying data topology. At this purpose, to define a local topology, the *Competitive Hebbian Learning* (CHL) paradigm [6–8] is employed by some algorithms such as Self-Organizing-Map (SOM) by Kohonen [9], the Growing Neural Gas (GNG) [10], and its variants [11–14]. Indeed, given an input sample, the two closest neurons, called first and second winners, are linked by an edge, which locally models the input shape. Hebbian learning is a cognitive/biologically inspired technique, based on a neuropsychological theory claiming that neurons can increase their specialization by competing for the right to respond to/represent a subset of the input data.

All the previously cited techniques suffer from the curse of dimensionality. Distance-based similarity measures are not effective when dealing with highly dimensional data (e.g., images or biological applications like gene expression). Therefore, many methods to reduce input dimensionality and to select the most important features have been employed, such as Principal Component Analysis (PCA) [15] and kernel functions [16]. To better preserve local topology in the reduced space, the Curvilinear Component Analysis (CCA) [17] and its online incremental version, the GCCA [18, 19], proposed a nonlinear projection algorithm. This approach is quite useful for noise removal and when input features are highly correlated, because projection reduces the problem complexity; on the contrary, when features are statistically independent, a smaller space implies worse clustering performance due to the information loss. An alternative way for dealing with high-dimensional data is the use of Deep Neural Networks (DNNs). Indeed, Convolutional Neural Networks (CNNs) [20] have proven to be a valid tool for handling high-dimensional input distribution in the case of supervised learning [21–24]. The strength of CNNs relies on the convolutional filters, which yield an output space that is linearly separable in terms of the output classes. In this sense, CNN filters can also be exploited for clustering. Indeed, CNNs, but also DNNs, can be trained by optimizing a clustering loss function [25–27]. A straightforward approach, however, may lead to overfitting, where data are mapped to compact clusters that do not correspond to data topology [28]. To overcome this problem, weight regularization, data augmentation, and supervised network pre-training have been proposed [28]. The latter technique exploits a pre-trained CNN (e.g., AlexNet on ImageNet [29]) as a feature extractor in a transfer learning way [30]. Otherwise, clustering learning procedures may be integrated with a network learning process, which require employing more complex architectures such as k-means in [31, 32], Autoencoders (AE) [33] as in [34–37], Variational Autoencoders (VAE) [38] as in [39, 40], graph neural networks as in [41], or Generative Adversarial Networks (GAN) [42] as in [43–45]. Such techniques usually employ a two-step learning process: first, a good representation of the input space is learnt through a network loss function and, later, the quantization is fine-tuned by optimizing a clustering-specific loss. The network loss can be either the reconstruction loss of an AE, the variational loss of a VAE, or the adversarial loss of a GAN. To the same purpose, a deep extension of sparse subspace clustering with L1-norm is used in [46]. At last, always taking inspiration from the supervised learning world, attention-based mechanisms have been also employed for deep clustering. Attention mechanisms [47] have been initially introduced for natural machine translation to allow models focus on the most important input data. In deep clustering, it has been used for enhancing the embedded representation in speech separation [48], but also combined with autoencoders for handwritten recognition [49] and molecular similarity [50]. The requirement of a two-step learning process in deep clustering algorithms derives from the different nature of the network and clustering losses, which hinders their integration.

To our knowledge, no previous work suggested to join DNN feature transformation skill with the higher representation capabilities of competitive learning approaches. In this paper, we propose two variants of a neural architecture where competitive learning is embedded in the training loss function. The first variant that we refer to as vanilla layer consists in a gradient-based competitive learning approach, where the weights represent the cluster prototypes, but the outputs are not meaningful. In order to integrate with deep architectures, a novel approach, called dual competitive layer, is here introduced, which directly outputs the prototypes after the presentation of a complete batch of input data. A duality theory is proposed and demonstrated, which highlights the relationships between the two layers.

The "Methods" section presents the vanilla and dual competitive layers together with the corresponding dual theory and the analysis of the loss function. The "Results" section tests the two layers on three synthetic datasets and confirms the validity of the proposed approach. The "Discussion—Theoretical Analysis" section provides a theoretical justification of the results by means of the analysis of the gradient flows, using both the stochastic approximation theory and the evaluation of their dynamics. Finally, the "Conclusion" section concludes the paper and proposes future directions.

Methods

Dual Neural Networks

Multi-layer feedforward neural networks are universal function approximators [51]. Given an input matrix $X \in \mathbb{R}^{d \times n}$

containing a collection of n observations and a set of k supervisions$Y \in \mathbb{R}^{k \times n}$, a neural network with d input and k output units can be used to approximate the target features Y. The relationship between X and Y can be arbitrarily complex; nonetheless, deep neural networks can optimize their parameters in such a way that their predictions $\widehat{Y}$ will match the target Y. In supervised settings, neural networks are used to combine the information of different features (rows of X) in order to provide a prediction$\widehat{Y}$, which corresponds to a nonlinear projection of the observations (columns of X) optimized to match the target Y. Hence, in such scenarios, the neural network will provide one prediction for each observation$i = 1, \ldots, n$.

The objective of competitive learning consists in studying the underlying structure of a manifold by means of prototypes, i.e., a set of positions in the feature space representative of the input observations. Each prototype p_k is a vector in $\mathbb{R}^d$ as it lies in the same feature space of the observations. Hence, competitive learning algorithms can be described as functions mapping an input matrix $X \in \mathbb{R}^{d \times n}$ in an output matrix $\widehat{P} \in \mathbb{R}^{d \times k}$ where the j-th column represents the prototype p_j. Indeed,

$$X \rightarrow \widehat{P} = \left[p_1 \ldots p_k\right] \tag{1}$$

is the relationship implemented by competitive learning. In deep clustering, it is used in a feedforward way. However, it directly computes the prototypes as its own weights and the output is not meaningful. Indeed, vanilla competitive neural networks [52–54] are composed of a set of competing neurons described by a vector of weights p_j, representing the position of neurons (a.k.a. *prototypes*) in the input space. The inverse of the Euclidean distance between the input data x_i and the weight vector p_j represents the similarity between the input and the prototype. For every input vector x_i, the prototypes *compete* with each other to see which one is the most similar to that particular input vector. By following the Competitive Hebbian Learning (CHL) rule [6, 7], the two closest prototypes to x_i are connected using an edge, representing their mutual activation. Depending on the approach, the closest prototypes to the input sample move towards it, reducing the distance between the prototype and the input. As a result, the position of the competing neurons in the input space will tend to cluster centroids of the input data. As a consequence, the feedforward representation of the vanilla algorithm is not justified. Instead, as it will be proved in the following sections, the most natural way of using a feedforward neural network for this kind of task is the transposition of the input matrix X while optimizing a prototype-based loss function. This approach derives from the idea of requiring the prototypes as outputs, and not as weights. This leads to the dual competitive layer (DCL, see "Duality Theory for Single-Layer Networks" and "Clustering as a Loss Minimization" sections), i.e., a fully connected layer trained on X^T, thus having n input units corresponding to observations and k output units corresponding to prototypes (see Fig. 1). Thus, the mapping of DCL is given by:

$$X^T \rightarrow \widehat{P}^T = \left[{p_1}^T \ldots {p_k}^T\right] \tag{2}$$

where, unlike the vanilla algorithm, the prototypes are the output of the network. Instead of combining different features to generate the feature subspace $\mathbb{R}^k$ where samples will be projected as for classification or regression tasks, in this case the neural network combines different samples to generate a synthetic summary of the observations, represented by a set of prototypes. Resuming, compared with the architecture of a vanilla competitive layer (VCL) [52] where prototypes correspond to the set of weight vectors of the fully connected layer, the dual approach naturally fits in a deep learning framework as the fully connected layer is actually used to apply a transformation of the input.

The DCL outputs an estimation of the prototypes after each batch of samples by means of a linear transformation represented by its weights. At this aim, a gradient-based minimization of a loss function is used, by using the whole batch. This reminds the centroid estimation of the generalized Lloyd algorithm (*k-means*, [55, 56]), which, instead, uses only the Voronoi sets. This is an important difference, because the error information can be backpropagated to the previous layer, if any, by exploiting all the observations, thus providing a relaxation of the Voronoi constraint. The underlying DCL analysis can be found in the "Discussion—Theoretical Analysis" section.

In order to estimate the parameters of DCL, a loss function representing the quantization error of the prototypes is used. It requires the computation of the Voronoi sets, which are deduced by means of the Euclidean distance matrix (edm). This is the same requirement of the second iteration of the generalized Lloyd algorithm. However, the latter uses this information for directly computing the centroids. The former, instead, only yields the error to be backpropagated. The analysis and choice of the loss function is illustrated in the "Clustering as a Loss Minimization" section.

By training on the transposed input, DCL looks at observations as features and vice versa. As a consequence, increasing the number of observations n (rows of X^T) enhances the capacity of the network, as the number of input units corresponds to n. Providing a higher number of features, instead, stabilizes the learning process as it expands the set of input vectors of DCL. After training, once prototype positions have been estimated, the dual network is no longer needed. Indeed, test observations can be evaluated finding the closest prototype for each sample. This means that the amount of information required to employ this approach in production environments corresponds just to the prototype matrix $\widehat{P}$.

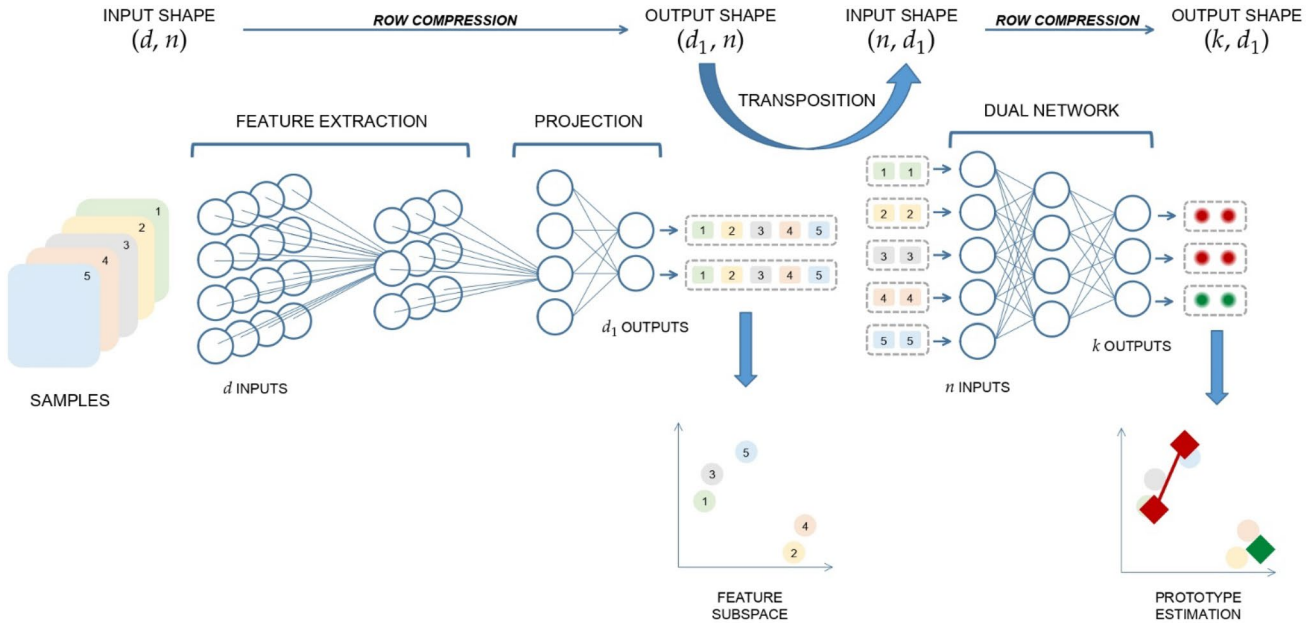

Fig. 1 Representation of a deep architecture where a dual competitive network is used to estimate cluster centroids. The first network executes a feature extraction and then maps training observations into a d1-dimensional feature subspace. This output is transposed and used to feed the dual network to estimate prototype positions [67]

Duality Theory for Single-layer Networks

The intuitions outlined in the previous section can be formalized in a general theory that considers the duality properties between a linear single-layer neural network and its dual, defined as a network, which learns on the transpose of the input matrix and has the same number of output neurons.

Consider a single-layer neural network whose outputs have linear activation functions. There are d input units and k output units which represent a continuous signal in the case of regression or class membership (posterior probabilities for cross entropy error function) in the case of classification. A batch of n samples, say X, is fed to the network. The weight matrix is W_1, where the element w_{ij} represents the weight from the input unit j to the neuron i. The single-layer neural network with linear activation functions in the lower scheme is here called the dual network of the former one. It has the same number of outputs and n inputs. It is trained on the transpose of the original X database. Its weight matrix is W_2 and the output batch is Y_2. The following theorems state the duality conditions of the two architectures. Figure 2 represents the two networks and their duality.

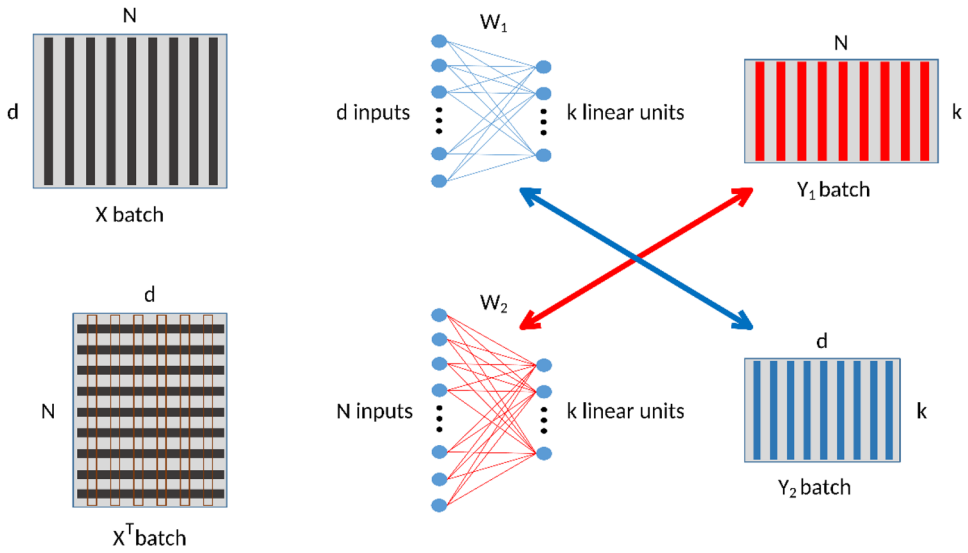

Fig. 2 Base (top) and dual (bottom) single-layer neural networks. The red double arrow shows the equivalence between the columns (in red) of the Y_1 batch and the weight vectors of W_2. The blue double arrow shows the dual equivalence (W_1 – Y_2) [67]

Theorem 2.1 (Network duality in competitive learning). *Given a loss function for competitive learning based on prototypes, a single linear network (base), whose weight vectors associated to the output neurons are the prototypes, is equivalent to another (dual) whose outputs are the prototypes, under the following assumptions:*

1. *The input matrix of the dual network is the transpose of the input matrix of the base network;*
2. *The samples of the input matrix X are uncorrelated with unit variance.*

Proof. Consider a loss function based on prototypes, whose minimization is required for competitive learning. From the assumption on the inputs (rows of the matrix X), it results $XX^T = I_d$. A single-layer linear network is represented by the matrix formula:

$$Y = WX = \left[prototype_1 \dots prototype_k\right]^T X \tag{3}$$

By multiplying on the right by X^T, it holds:

$$WXX^T = YX^T \tag{4}$$

Under the second assumption:

$$W = \left[prototype_1 \dots prototype_k\right]^T = YX^T \tag{5}$$

This equation represents a (dual) linear network whose outputs are the prototypes W. Considering that the same loss function is used for both cases, the two networks are equivalent.

This theorem directly applies to the VCL (base) and DCL (dual) neural networks if the assumption 2 holds for the training set. If not, a pre-processing, e.g., batch normalization, can be performed.

This theorem justifies the dual approach, in the sense that this novel architecture directly outputs the prototypes by using the weights for building the solution. It can be said that DCL is more "neural" than VCL (whose output is not meaningful). It also requires an input normalization (uncorrelation with unit variance), which is a standard requirement in data pre-processing (batch normalization).

Theorem 2.2 (Impossible complete duality). *Two dual networks cannot share weights as $W_1 = Y_2$ and $W_2 = Y_1$(complete dual constraint), except if the samples of the input matrix X^T are uncorrelated with unit variance.*

Proof. From the duality of networks and their linearity, for an entire batch it follows:

$$\begin{cases} Y_1 = W_1 X \\ Y_2 = W_2 X^T \end{cases} \Rightarrow W_1 = Y_1 X^T \Rightarrow W_1 = W_1 X X^T \Rightarrow XX^T = I_d \tag{6}$$

$$\begin{cases} Y_1 = W_1 X \\ Y_2 = W_2 X^T \end{cases} \Rightarrow W_2 = Y_2 X^T \Rightarrow W_2 = W_2 X^T X \Rightarrow X^T X = I_n \tag{7}$$

where I_d and I_n are the identity matrices of size d and n, respectively. These two final conditions are only possible if the samples of the input matrix X are uncorrelated with unit variance, which is not the case in (almost all) machine learning applications.

Theorem 2.3 (Half duality I). *Given two dual networks, if the samples of the input matrix X^T are uncorrelated with unit variance and if $W_1 = Y_2$ (first dual constraint), then $W_2 = Y_1$ (second dual constraint).*

Proof. From the first dual constraint (see Fig. 3), for the second network it stems:

$$Y_2 = W_1 = W_2 X^T \tag{8}$$

Hence:

$$Y_1 = W_1 X \Rightarrow Y_1 = W_2 X^T X \tag{9}$$

under the given assumption on X^T, which implies $X^T X = I_n$, the result follows.

Theorem 2.4 (Half duality II). *Given two dual networks, if the samples of the input matrix X are uncorrelated with unit variance and if $W_2 = Y_1$ (second dual constraint), then $W_1 = Y_2$(first dual constraint).*

Proof. From the second dual constraint (see Fig. 3), for the second network it stems:

$$Y_1 = W_2 = W_1 X \tag{10}$$

From the assumption on the inputs (rows of the matrix X), it results $XX^T = I_d$. The first neural architecture yields:

$$Y_2 = W_2 X^T \Rightarrow Y_2 = W_1 X X^T = W_1 \tag{11}$$

Theorem 2.4 completes the analysis of duality, by highlighting the relationships between the VCL weights and DCL outputs, and justifies the use of backpropagation in a straight way in DCL. Indeed, the meaningfulness of the DCL output allows to estimate the cost-function, which will be backpropagated, from the output. In this sense, DCL can be integrated in a deep architecture, as one of its layers.

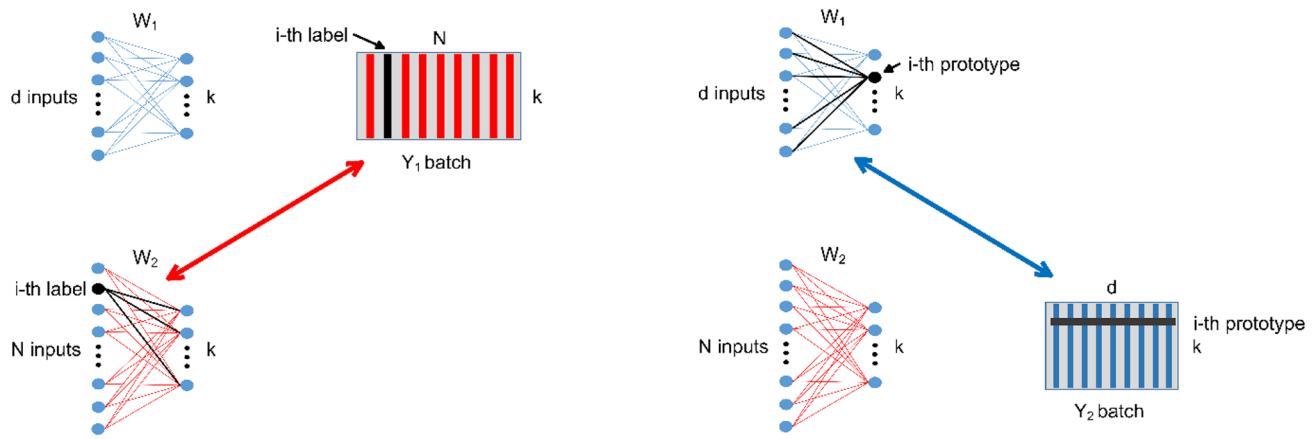

Fig. 3 Half dualities

Corollary 2.4.1 (Self-supervised learning). *The assumption of Theorem* 2.4 *implies the construction of labels for the base network.*

Proof. As sketched in Fig. 3, under the assumption of the equivalence between the training of the dual network (building of prototypes) and the architecture of the base network (output neurons as prototypes), the previous theorem implies the second dual constraint, which means the construction of a self-organized label.

Thanks to this corollary, the base network can work in a self-supervised way, by using the results of the dual self-organization, to infer information on the dataset. This results in a new approach to self-supervised learning.

Clustering as a Loss Minimization

The theoretical framework developed in the "Duality Theory for Single-Layer Networks" section can be easily adapted to accommodate for a variety of unsupervised learning tasks by designing a suitable loss function. One of the most common prototype-based loss functions employed for clustering aims at minimizing the expected squared quantization error [57]. Depending on the feature subspace, some clusters may have complex shapes; therefore, using only one prototype per cluster may result in a poor representation. To overcome this limitation, each cluster can be represented by a graph composed of a collection of connected prototypes. The corresponding loss function can be written as:

$$\mathcal{L} = \mathcal{Q} + \lambda \|E\|_2 \tag{12}$$

where $\mathcal{Q}$ is the classical quantization error, given by the sum of the squares of the Euclidean distances between the data and their closest prototypes; and E is the adjacency matrix describing the connections between prototypes. The $\mathcal{Q}$ term is estimated from the Voronoi sets of the prototypes, which require the evaluation of the edm between X and Y. The E term uses the CHL rule, which implies the estimation of the first and second winner w.r.t. each sample by means of the same edm. By using the Lagrangian term $\lambda\|E\|_2$, the complexity of the graph representing connections among prototypes can be minimized, in order to learn the minimal topological structure. Lonely prototypes (i.e., prototypes without connections) may represent outliers and can be easily pruned or examined individually.

The minimization of Eq. (12) can be exploited for analyzing the topological properties of the input manifolds. While this is out of the scope of this paper, it allows both the detection of clusters by means of the connectedness of the graphs and the best number of prototypes (pruning from a user-defined number of output units), as it has been shown in [10, 14]. This technique addresses the problem of the choice of prototypes in k-means.

The "Duality Theory for Single-Layer Networks" section established a set of conditions for the duality of two single-layer feedforward neural networks only in terms of their architecture. Instead, the choice of the learning process determines their application. In the case of clustering, they correspond to the VCL and DCL respectively, if they are both trained by the minimization of Eq. (12). However, as it will be shown in the "Discussion—Theoretical Analysis" section, the equivalence in the architecture does not imply an equivalence in the training process, even if the loss function and the optimization algorithm are the same. Indeed, in a vanilla competitive layer, there is no forward pass as Y_1 is neither computed nor considered and the prototype matrix is just the weight matrix W_1:

$$\widehat{P_1} = \left[prototype_1 \dots prototype_k\right] = W_1 \tag{13}$$

where $\text{prototype}_i \in \mathbb{R}^d$. In a dual competitive layer, instead, the prototype matrix corresponds to the output Y_2; hence,

the forward pass is a linear transformation of the input X^T through the weight matrix W_2:

$$\widehat{P_2} = \left[prototype_1 \dots prototype_k\right]^T = Y_2 = W_2 X^T$$
$$= \begin{bmatrix} w_1^T \\ w_2^T \\ \dots \\ w_k^T \end{bmatrix} \begin{bmatrix} f_1 & f_2 & \dots & f_d \end{bmatrix} = \begin{bmatrix} w_1^T f_1 & w_1^T f_2 & \dots & w_1^T f_d \\ w_2^T f_1 & w_2^T f_2 & \dots & w_2^T f_d \\ \dots & \dots & \ddots & \vdots \\ w_k^T f_1 & w_k^T f_2 & \dots & w_k^T f_d \end{bmatrix} \quad (14)$$

where w_i is the weight vector of the i-th output neuron of the dual network and f_i is the i-th feature over all samples of the input matrix X. The components of i-th prototype are computed using the same weight w_i, because each row is a rank one outer product. Besides, each component is computed as it was a one-dimensional learning problem. For instance, the first component of the prototypes is $\begin{bmatrix} w_1^T f_1 & \dots & w_k^T f_1 \end{bmatrix}^T$, which means that the first component of all the prototypes is computed by considering just the first feature f_1. Hence, each component is independent from all the other features of the input matrix, allowing the forward pass to be just like a collection of d columnwise one-dimensional problems.

Such differences in the forward pass have an impact on the backward pass as well, even if the form of the loss function is the same for both systems. However, the parameters of the optimization are not the same. For the base network:

$$\mathcal{L} = \mathcal{L}(X, W_1) \quad (15)$$

while for the dual network:

$$\mathcal{L} = \mathcal{L}(X^T, Y) \quad (16)$$

where Y is a linear transformation (filter) represented by W_2. In the base competitive layer, the gradient of the loss function with respect to the weights W_1 is computed directly as:

$$\nabla \mathcal{L}(W_1) = \frac{d\mathcal{L}}{dW_1} \quad (17)$$

On the other hand, in the dual competitive layer, the chain rule is required to compute the gradient with respect to the weights W_2 as the loss function depends on the prototypes Y_2:

$$\nabla \mathcal{L}\left(W_2\right) = \frac{d\mathcal{L}}{dW_2} = \frac{d\mathcal{L}}{dY_2} \cdot \frac{dY_2}{dW_2} \quad (18)$$

As a result, despite the architecture of the two layers is equivalent, the learning process is quite different.

Results

In order to rigorously assess the main characteristics of the learning process, several metrics are evaluated while training the VCL and DCL networks on three synthetic datasets containing clusters of different shapes and sizes. Table 1 summarizes the main characteristics of each experiment. While these experiments deal with maximum two clusters, there is no theoretical reason to limit this study to only two clusters. For DCL, the number of output units corresponds to the number of clusters (just like the parameter k in the k-means algorithm), as shown in [58]. The first dataset is composed of samples drawn from a two-dimensional Archimedean spiral (*Spiral*). The second dataset consists of samples drawn from two half semicircles (*Moons*). The last one is composed of two concentric circles (*Circles*). Each dataset is normalized by removing the mean and scaling to unit variance before fitting neural models. For all the experiments, the number of output units k of the dual network is set to 30. A grid-search optimization is conducted for tuning the hyper-parameters. The learning rate is set to $\epsilon = 0.008$ for VCL and to $\epsilon = 0.0008$ for DCL. Besides, for both networks, the number of epochs is equal to $\eta = 400$, while the Lagrangian multiplier to $\lambda = 0.01$. For each dataset, both networks are trained 10 times using different initialization seeds in order to statistically compare their performance.

Table 1 Synthetic datasets used for the simulations (s.v. stands for singular value)

DATASET	SAMPLES	FEATURES	CLUSTERS	MAX S.V	MIN S.V
SPIRAL	500	2	1	23.43	21.24
MOONS	500	2	2	26.97	16.51
CIRCLES	500	2	2	22.39	22.34

Figure 4 shows for each dataset the dynamics of three key metrics for both VCL and DCL: the quantization error, the topological complexity of the solution (i.e., $\|E\|$), and the number of valid prototypes (i.e., the ones with a non-empty Voronoi set). By looking at the quantization error, both networks tend to converge to similar local minima in all scenarios, thus validating their theoretical equivalence. Nonetheless, the single-layer dual network exhibits a much faster rate of convergence compared to the vanilla competitive layer. The most significant differences are outlined (i) by the number of valid prototypes as DCL tends to employ more resources and (ii) by the topological complexity as VCL favors more complex solutions.

Figure 5 shows topological clustering results after 800 epochs. As expected, both neural networks yield an adequate estimation of prototype positions, even though the topology learned by DCL is far more accurate in following the underlying manifolds w.r.t. to VCL.

Figure 6 shows the trajectories of the prototypes during the training for both networks. The parameters in both networks have been initialized by means of the Glorot initializer [59], which draws small values from a normal distribution centered on zero. For VCL, these parameters are the

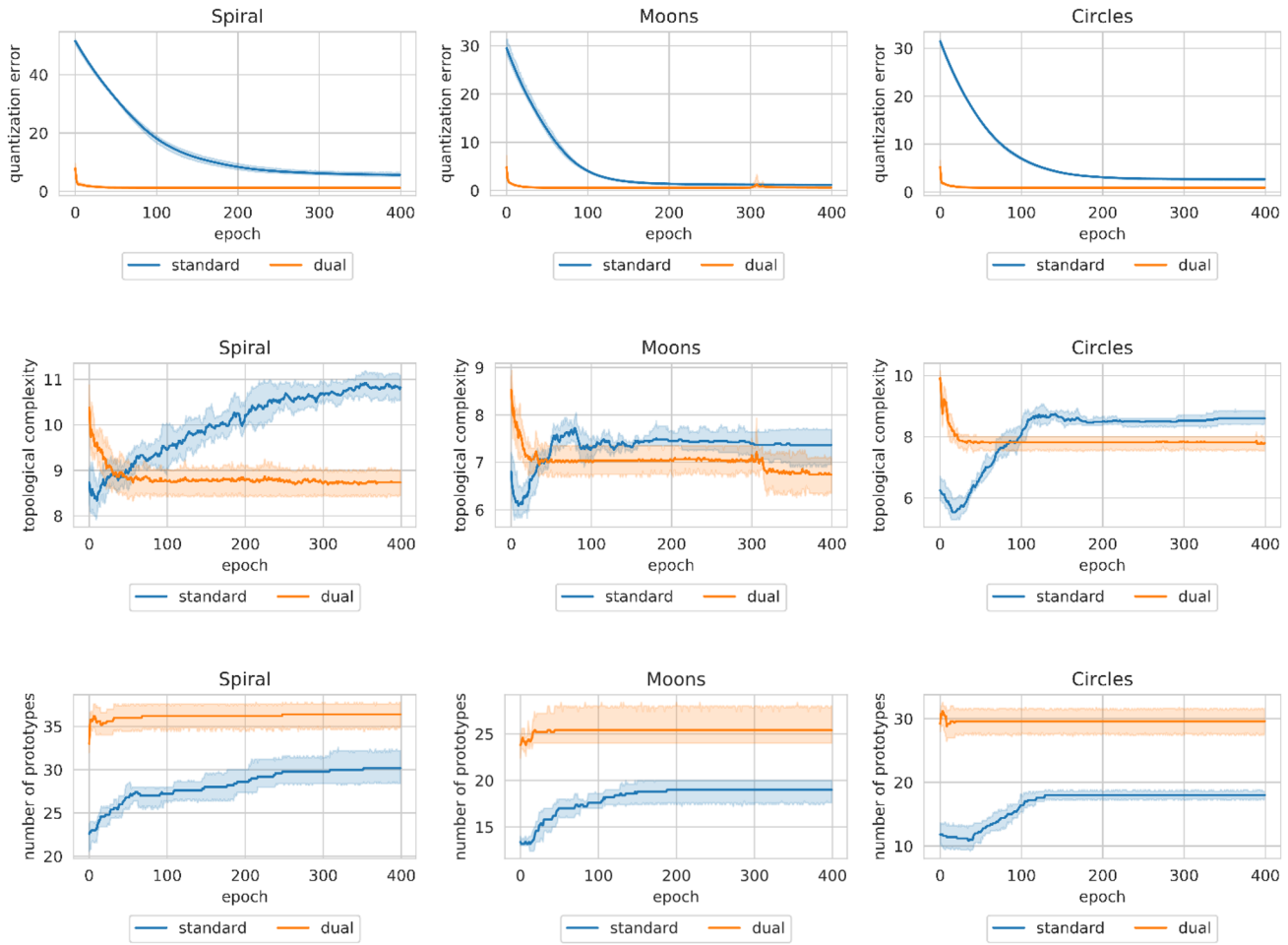

Fig. 4 Comparison of three key metrics between the vanilla single-layer network and its dual over 10 runs. The metrics are the quantization error (top row), the norm of the matrix of the edges (middle row), and the number of valid prototypes (bottom row). The metrics are computed on three different datasets: Spiral (left column), Moons (middle column), and Circles (right column). Error bands represent the standard error of the mean

prototypes and they are initially clustered around the origin, as expected. For DCL, instead, the initial prototypes are an affine transformation of the inputs parameterized by the weight matrix. This implies the initial prototypes are close to a random choice of the input data. The VCL trajectories tend towards the closest to the initial (close to the origin) cluster and then some of them spread towards the furthest manifolds. The DCL trajectories are much shorter because of the closeness of the initial prototypes to the input clusters. These considerations reveal the better suitability of DCL to deep learning traditional initializations.

The performance of the vanilla competitive layer and its dual network in tackling high-dimensional problems is assessed through numerical experiments. Sure enough, standard distance–based algorithms generally suffer the well-known curse of dimensionality when dealing with high-dimensional data. The MADELON algorithm proposed in [60] is used to generate high-dimensional datasets with an increasing number of features and fixed number of samples. This algorithm creates clusters of points, normally distributed about vertices of an n-dimensional hypercube. An equal number of cluster and data is assigned to two different classes. Both the number of samples (n_s) and the dimensionality of the space (n_f) in which they are placed can be defined programmatically. More precisely, the number of samples is set to $n_s = 100$ while the number of features ranges in $n_f \in [1000, 2000, 3000, 5000, 10000]$. The number of required centroids is fixed to one-tenth the number of input samples. Table 2 summarizes the hyper-parameter settings. Three different networks are compared (see Fig. 7): VCL, DCL, and a deep variant of DCL with two hidden layers of 10 neurons each (deep-DCL). Results are averaged over 10 repetitions on each dataset. Accuracy for each cluster is calculated by considering true positive those samples belonging to the class more represented, and false positive the remaining data.

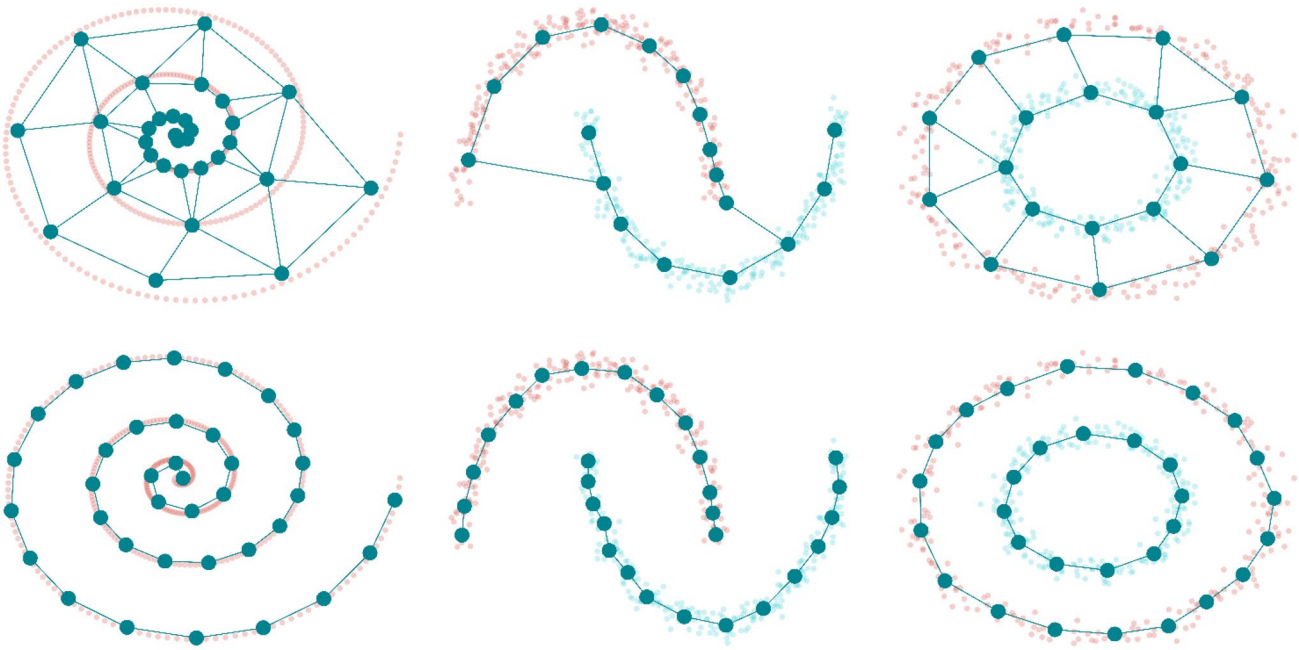

Fig. 5 Experiments on synthetic datasets. From left to right: Spiral, Moons, and Circles datasets

The "Discussion—Theoretical Analysis" section yields a theoretical explanation for the observed results. All the code for the experiments has been implemented in Python 3, relying upon open-source libraries [61, 62]. All the experiments have been run on the same machine: Intel® Core™ i7-8750H 6-Core Processor at 2.20 GHz equipped with 8 GiB RAM. To enable code reuse, the Python code for the mathematical models including parameter values and documentation is freely available under Apache 2.0 Public License from a GitHub repository1[1] [63]. The whole package can also be downloaded directly from PyPI[2]. Unless required by applicable law or agreed to in writing, software is distributed on an "as is" basis, without warranties or conditions of any kind, either express or implied. The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Discussion—Theoretical Analysis

Stochastic Approximation Theory of the Gradient Flows

In the following, the gradient flows of the vanilla and the dual single-layer neural networks are formally examined when trained using the quantization error, one of the most common loss functions used for training unsupervised neural networks in clustering contexts. The following theory is based on the assumption of $\lambda = 0$ in Eq. (12). Taking into account the edge error only relaxes the analysis, but the results remain valid. Under the stochastic approximation theory, the asymptotic properties of the gradient flows of the two networks can be estimated.

Base Layer Gradient Flow

For each prototype j, represented in the base layer by the weight vector $W_1^j \in \mathbb{R}^d$ of the j-th neuron (it is the j-th row of the matrix W_1), the contribution of its Voronoi set to the quantization error is given by:

$$E^j = \sum_{i=1}^{n_j} \|x_i - W_1^j\|_2^2 = \sum_{i=1}^{n_j} (\|x_i\|_2^2 + \|W_i^j\|_2^2 - 2x_i^T W_1^j) \quad (19)$$

where n_j is the cardinality of the j-th Voronoi set. The corresponding gradient flow of the base network is the following:

$$W_1^j(t+1) = W_1^j(t) - \epsilon \nabla_{W_1^j} E^j = W_1^j(t) - \epsilon \sum_{i=1}^{n_j} (W_1^j - x_i) \quad (20)$$

being ϵ the learning rate. The averaging ODE holds:

$$\frac{dW_1^j}{dt} = -W_1^j + \mu_j \quad (21)$$

where $\mu_j = \mathbb{E}[x_i]$ is the expectation in the limit of infinite samples of the j-th Voronoi set, and corresponds to the centroid of the Voronoi region. The unique critical point of the ODE is given by:

[1] https://github.com/pietrobarbiero/cola

[2] https://pypi.org/project/deeptl/1.0.0/

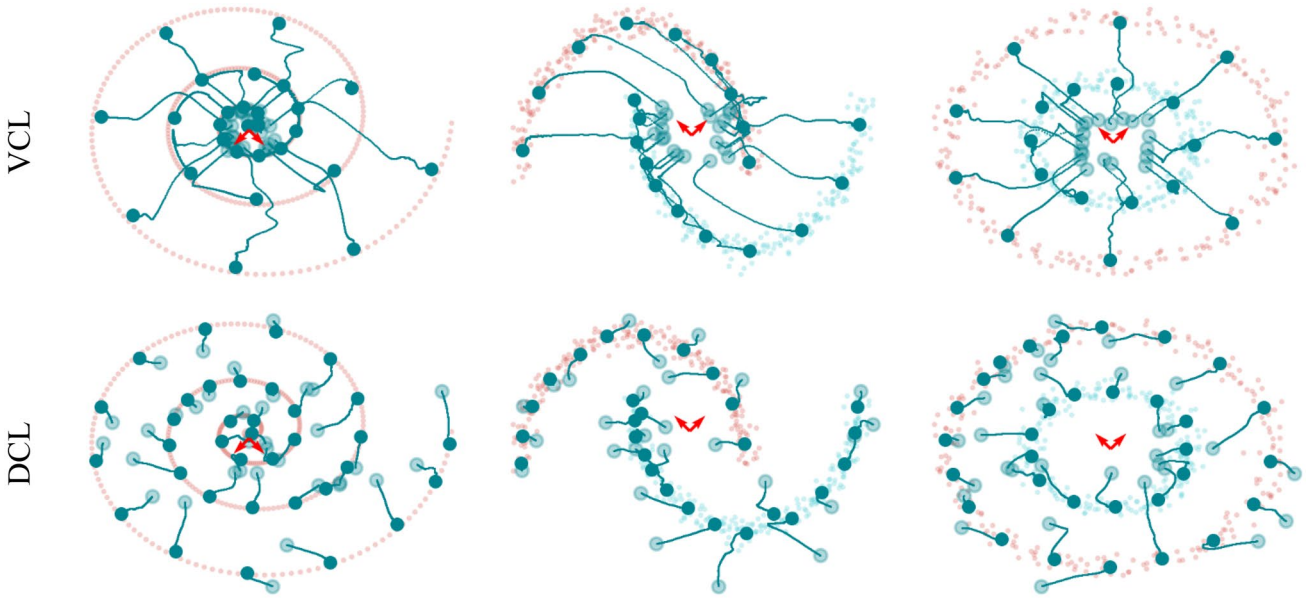

Fig. 6 Dynamical simulations. From left to right: Spiral, Moons, and Circles datasets

$$W^j_{1,crit} = \mu_j \tag{22}$$

and the ODE can be rewritten as:

$$\frac{dw^j_1}{dt} = -w^j_1 \tag{23}$$

under the transformation $w^j_1 = W^j_1 - W^j_{1,crit}$ in order to study the origin as the critical point. The associated matrix is $-I_d$, whose eigenvalues are all equal to -1 and whose eigenvectors are the vectors of the standard basis. Hence, the gradient flow is stable and decreases in the same exponential way, as e^{-t}, in all directions. The gradient flow of one epoch corresponds to an approximation of the second step of the generalized Lloyd iteration, as stated before.

Dual Layer Gradient Flow

In the dual layer, the prototypes are estimated by the outputs, in such a way that they are represented by the rows of the Y_2 matrix. Indeed, the j-th prototype is now represented by the row vector $\left(Y^j_2\right)^T$, from now on called y^T_j for sake of simplicity. It is computed by the linear transformation:

$$y^T_j = \left(W^j_2\right)^T [x_1 \cdots x_d] = \left(W^j_2\right)^T X^T = \Omega^T_j X^T \tag{24}$$

where $x_i \in \mathbb{R}^n$ is the i-th row of the training set X and $W^j_2 \in \mathbb{R}^n$ is the weight vector of the j-th neuron (it is the j-th row of the matrix W_2), and is here named as Ω_j for simplicity. Hence, the j-th prototype is computed as:

$$y_j = X\Omega_j \tag{25}$$

and its squared (Euclidean) 2-norm is:

$$\|y_j\|^2_2 = \Omega^T_j X^T X \Omega_j \tag{26}$$

For the j-th prototype, the contribution of its Voronoi set to the quantization error is given by:

$$E^j = \sum_{i=1}^{n_j} \|x_i - y_j\|^2_2 = \sum_{i=1}^{n_j} (\|x_i\|^2_2 + \|y_i\|^2_2 - 2x^T_i y_j) \tag{27}$$

with the same notation as previously. The gradient flow of the dual network is computed as:

$$\Omega_j(t+1) = \Omega_j(t) - \epsilon \nabla_{\Omega_j} E^j \tag{28}$$

being ϵ the learning rate. The gradient is given by:

$$\nabla_{\Omega_j} E^j = \nabla_{\Omega_j} \sum_{i=1}^{n_j} \left(x^T_i x_i + \Omega^T_j X^T X \Omega_j - 2\Omega^T_j X^T x_i\right) = 2\left(X^T X \Omega_j - X^T x_i\right) \tag{29}$$

Table 2 Parameters for high-dimensional simulations using MADELON (s.v. stands for singular value)

SAMPLES	FEATURES	CLUSTERS	MAX S.V	MIN S.V
100	1000	2	112	3E − 14
100	2000	2	120	7E − 14
100	3000	2	126	4E − 14
100	5000	2	139	5E − 14
100	10,000	2	154	7E − 14

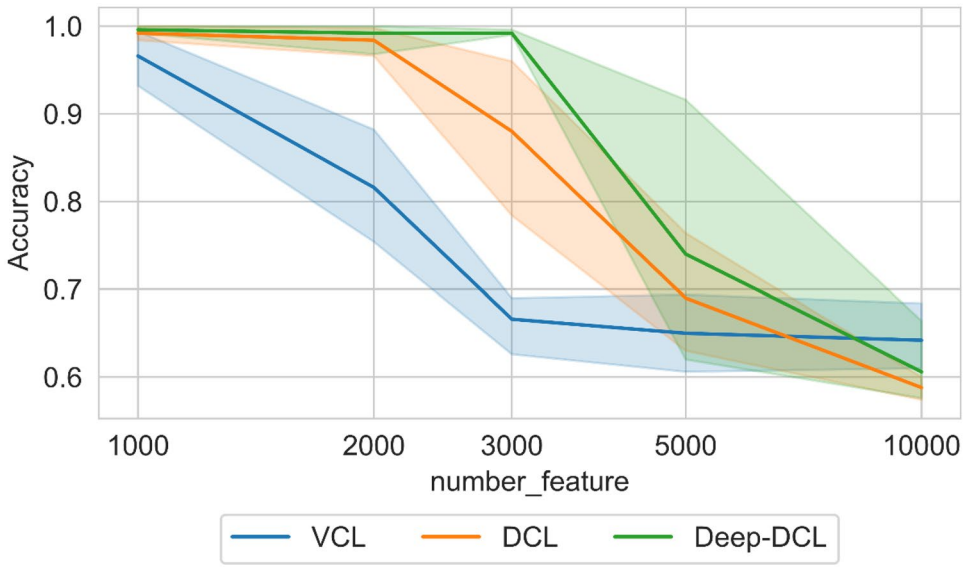

Fig. 7 High-dimensional simulations: accuracy as a function of the dimensionality of the problem. Error bands correspond to the standard error of the mean [67]

The averaging ODE is estimated as:

$$\frac{d\Omega_j}{dt} = -\left(X^T X\Omega_j - X^T \mu_j\right) \tag{30}$$

The unique critical point of the ODE is the solution of the normal equations:

$$X^T X\Omega_j = X^T \mu_j \tag{31}$$

The linear system can be solved only if $X^T X \in \mathbb{R}^{n\times n}$ is full rank. This is true only if $n \leq d$ (the case $n = d$ is trivial and, so, from now on the analysis deals with $n < d$) and all columns of X are linearly independent. In this case, the solution is given by:

$$\Omega_{j,crit} = (X^T X)^{-1} X^T \mu_j = X^+ \mu_j \tag{32}$$

where X^+ is the pseudoinverse of X. The result corresponds to the least squares solution of the overdetermined linear system:

$$X\Omega_j = \mu_j \tag{33}$$

which is equivalent to:

$$\Omega_j^T X^T = \mu_j^T \tag{34}$$

This last system shows that the dual layer asymptotically tends to output the centroids as prototypes. The ODE can be rewritten as:

$$\frac{dw_j}{dt} = -X^T X w_j \tag{35}$$

under the transformation $w_j = \Omega_j - \Omega_{j,crit}$ in order to study the origin as the critical point. The associated matrix is $-X^T X$. Consider the singular value decomposition (SVD) of $X = U\Sigma V^T$ where $U \in \mathbb{R}^{d\times d}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal and $\Sigma \in \mathbb{R}^{d\times n}$ is diagonal (nonzero diagonal elements named singular values and called σ_i, indexed in decreasing order). The i-th column of V (associated to σ_i) is written as v_i and is named right singular vector. Then:

$$X^T X = \left(U\Sigma V^T\right)^T U\Sigma V^T = V\Sigma^2 V^T \tag{36}$$

is the eigenvalue decomposition of the sample autocorrelation matrix of the inputs of the dual network. It follows that the algorithm is stable and the ODE solution is given by:

$$w_j(t) = \sum_{i=1}^{n} c_i v_i e^{-\sigma_i^2 t} \tag{37}$$

where the constants depend on the initial conditions. The same dynamical law is valid for all the other weight neurons. If $n > d$ and all columns of X are linearly independent, it follows:

$$rank(X) = rank\left(X^T X\right) = d \tag{38}$$

and the system $X\Omega_j = \mu_j$ is underdetermined. This set of equations has a nontrivial nullspace and so the least squares solution is not unique. However, the least squares solution of minimum norm is unique. This corresponds to the minimization problem:

$$min\left(\Omega_j\right)\ s.t. X\Omega_j = \mu_j \tag{39}$$

The unique solution is given by the normal equations of the second kind:

$$\begin{cases} XX^T z = \mu_j \\ \Omega_j = X^T z \end{cases} \tag{40}$$

that is, by considering that XX^T has an inverse:

$$\Omega_j = X^T\left(XX^T\right)^{-1}\mu_j \tag{41}$$

Multiplying on the left by XX^T yields:

$$\left(X^TX\right)\Omega_j = \left(X^TX\right)X^T\left(X^TX\right)^{-1}\mu_j = X^T\left(X^TX\right)\left(X^TX\right)^{-1}\mu_j = X^T\mu_j \tag{42}$$

that is Eq. (31), which is the system whose solution is the unique critical point of the ODE (setting the derivative in Eq. (30) to zero). Resuming, both cases give the same solution. However, in the case $n > d$ and $rank(X) = d$, the output neuron weight vectors have minimum norm and are orthogonal to the nullspace of X, which is spanned by $v_{n-d+1}, v_{(n-d+2)}, \ldots, v_n$. Indeed, X^TX has $n-d$ zero eigenvalues, which correspond to centers. Therefore, the ODE solution is given by:

$$w_j(t) = \sum_{i=1}^{n-d} c_i v_i e^{-\sigma_i^2 t} + \sum_{i=n-d+1}^{n} c_i v_i \tag{43}$$

This theory proves the following theorem.

Theorem 3.1 (Dual flow and PCA). *The dual network evolves in the directions of the principal axes of its autocorrelation matrix (see Eq. (36)) with time constants given by the inverses of the associated data variances.*

This statement claims the dual gradient flow moves faster in the more relevant directions, i.e., where data vary more. Indeed, the trajectories start at the initial position of the prototypes (the constants in Eq. (43) are the associated coordinates in the standard framework rotated by V) and evolve along the right singular vectors, faster in the directions of more variance in the data. It implies a faster rate of convergence because it is dictated by the data content, as already observed in the numerical experiments (see Fig. 4).

Dynamics of the Dual Layers

For the basic layer, it holds:

$$W_1^j - W_{1,crit}^j = le^{-t} \tag{44}$$

where $l \in \mathbb{R}^d$ is a vector of constants. Therefore, W_1^j tends asymptotically to μ_j, by moving in $\mathbb{R}^d$. However, being μ_j a linear combinations of the columns of X, it can be deduced that, after a transient period, the neuron weight vectors tend to the range (column space) of X, say $R(X)$, i.e.:

$$\forall j, \forall t > t_0 W_1^j \in R(X) = span\ (u_1, u_2, \ldots, u_r) \tag{45}$$

where t_0 is a certain instant of time and $r = rank(X) = min\{d, n\}$ under the assumption of samples independently drawn from the same distribution, which prevents from the presence of collinearities in data. It follows

$$W_1^j = W_{1,crit}^j + Uce^{-t} \tag{46}$$

where $l \in \mathbb{R}^d$ is another vector of constants. Then, W_1^j can be considered the output of a linear transformation represented by the matrix X, i.e., $W_1^j = X_p$, being $p \in \mathbb{R}^n$ its preimage. Hence, $(W_1^j)^T = p^TX^T$, which shows the duality. Indeed, it represents a network whose input is X^T, and the output $(W_1^j)^T$ and parameter weight vector p^T are the interchange of the corresponding ones in the base network. Notice, however, that the weight vector in the dual network corresponds only through a linear transformation, that is, by means of the preimage. Under the second duality assumption $XX^T = I_d$, it holds:

$$XX^T = U\Sigma V^T\left(U\Sigma V^T\right)^T = U\Sigma\Sigma^TU^T = I_d \Rightarrow U\Sigma\Sigma^T = U \Rightarrow \begin{cases} UI_d = Ud \leq n \\ U\begin{bmatrix} I_n & 0_{n,d-n} \\ 0_{d-n,n} & 0_{d-n,d-n} \end{bmatrix} d > n \end{cases} \tag{47}$$

where $0_{r,s}$ is the zero matrix with r rows and s columns. Therefore, this assumption implies there are d singular values all equal to 1 or -1. In the case of remaining singular values, they are all null and of cardinality $d-n$. For the dual layer, under the second duality assumption, in the case of singular values all equal to -1 or 0, it follows:

$$\Omega_j - \Omega_{j,crit} = \begin{cases} Vqe^{-t} d \geq n \\ Vq\begin{bmatrix} e^{-t}1_d \\ 1_{n-d} \end{bmatrix} d < n \end{cases} \tag{48}$$

where $q \in \mathbb{R}^n$. Therefore, Ω_j tends asymptotically to $\Omega_{j,crit}$, by moving in $\mathbb{R}^n$. Hence, it can be deduced that, after a transient period, the neuron weight vectors tend to the range (column space) of X, say $R(X^T)$, i.e.:

$$\forall j, \forall t > t_0 \Omega^j \in R\left(X^T\right) = span\ (v_1, v_2, \ldots, v_r) \tag{49}$$

where t_0 is a certain instant of time and $r = rank(X) = min\{d, n\}$ under the same assumption of noncollinear data.

Resuming, the base and dual gradient flows, under the two duality assumptions, except for the presence of centers, are given by:

$$\begin{cases} w_1^j = Uce^{-t} \\ \omega_j = Vqe^{-t} \end{cases} \Rightarrow X\omega_j = XVqe^{-t} \Rightarrow X\omega_j = U\Sigma qe^{-t} \Rightarrow X\omega_j = Uce^{-t} \Rightarrow w_1^j = X\omega_j \tag{50}$$

because $XV = U\Sigma$ from the SVD of X and $c = \Sigma q$ for the arbitrariness of the constants. This result claims the fact that

the base flow directly estimates the prototype, while the dual flow estimates its preimage. This confirms the duality of the two layers from the dynamical point of view and proves the following theorem.

Theorem 3.2 (Dynamical duality). *Under the two assumptions of 2.1, the two networks are dynamically equivalent. In particular, the base gradient flow evolves in $R(X)$ and the dual gradient flow evolves in $R(X^T)$.*

More in general, the fact that the prototypes are straightly computed in the base network implies a more rigid dynamics of its gradient flow. On the contrary, the presence of the singular values in the exponentials of the dual gradient flow originates from the fixed transformation (matrix X) used for the prototype estimation. They are exploited for a better dynamics, because they are suited to the statistical characteristics of the training set, as discussed before. Both flows estimate the centroids of the Voronoi sets, like the centroid estimation step of the Lloyd algorithm, but the linear layers allow the use of gradient flows and do not require the a priori knowledge of the number of prototypes (see the discussion on pruning in the "Clustering as a Loss Minimization" section). However, the dual flow is an iterative least squares solution, while the base flow does the same only implicitly. In the case$d > n$,$rank(X) = rank(X^T) = n$, and the base gradient flow stays in $\mathbb{R}^d$, but tends to lie on the n-dimensional subspace$R(X)$. Instead, the dual gradient flow is n-dimensional and always evolves in the n-dimensional subspace$R(X^T)$. Figure 8 shows both flows and the associated subspaces for the case $n = 2$ and$d = 3$. The following lemma describes the relationship between the two subspaces.

Lemma 3.3 (Range transformation). *The subspace $R(X)$ is the transformation by X of the subspace $R(X^T)$.*

Proof. The two subspaces are the range (column space) of the two matrices X and X^T:

$$R(X) = \{z : z = Xu \text{ for a certain } u\} \tag{51}$$

$$R\left(X^T\right) = \left\{y : y = X^T x \text{ for a certain } x\right\} \tag{52}$$

Then:

$$XR\left(X^T\right) = \left\{u = Xy : y = X^T x \text{ for a certain } x\right\} = R(X) \tag{53}$$

More in general, multiplying X by a vector yields a vector in $R(X)$.

All vectors in $R(X^T)$ are transformed by X in the corresponding quantities in $R(X)$. In particular:

$$u_i = \frac{1}{\sigma_i} X v_i \ \ \forall i = 1, \ldots, n \tag{54}$$

$$\mu_j = X\Omega_{j,crit} \tag{55}$$

This analysis proves the following theorem.

Theorem 3.4 (Fundamental on gradient flows, part I). *In the case $d > n$, the base gradient flow represents the temporal law of a d-dimensional vector tending to an n-dimensional subspace containing the solution. Instead, the dual gradient flow always remains in an n-dimensional subspace containing the solution. Then, the least squares transformation X^+ yields a new approach, the dual one, which is not influenced by d, i.e., the dimensionality of the input data.*

This assertion is the basis of the claim the dual network is a novel and very promising technique for high-dimensional clustering. However, it must be considered that the underlying theory is only approximated and gives an average behavior. Figure 7 shows a simulation comparing the performances of VCL, DCL, and a deep variant of the DCL model in tackling high-dimensional problems with an increasing number of features. The simulations show how the dual methods are more

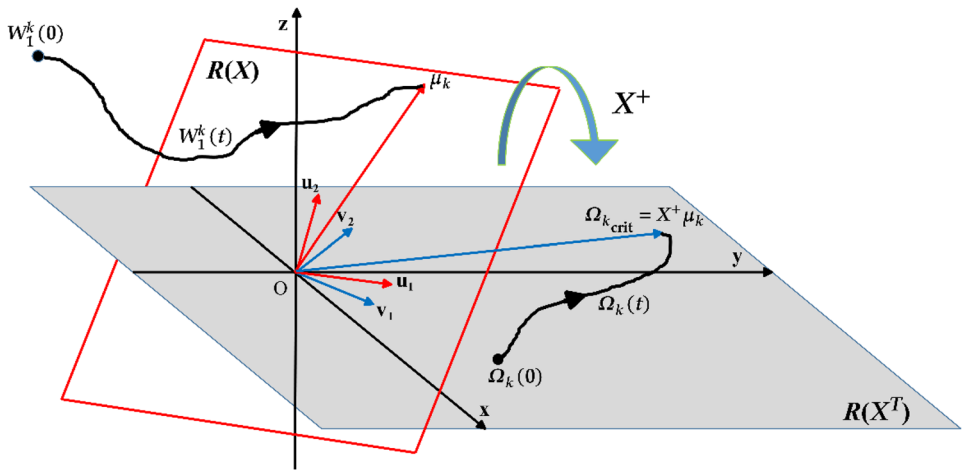

Fig. 8 Gradient flows and subspaces ($n = 2$ and $d = 3$) [67]

capable to deal with high-dimensional data as their accuracy remains near 100% until 2000 − 3000 features. Obviously, the deep version of DCL (deep-DCL) yields the best accuracy because it exploits the nonlinear transformation of the additional layers.

In the case $n \geq d$, instead, the two subspaces have dimension equal to d. Then, they coincide with the feature space, eliminating any difference between the two gradient flows. In reality, for the dual flow, there are $n - d$ remaining modes with zero eigenvalue (centers) which are meaningless, because they only add $n - d$ constant vectors (the right singular vectors of X) which can be eliminated by adding a bias to each output neuron of the dual layer.

Theorem 3.5 (Fundamental on gradient flows, part II). *In the case $d \leq n$, both gradient flows lie in the same (feature) space, the only difference being the fact that the dual gradient flow temporal law is driven by the variances of the input data.*

The Voronoi Set Estimation

Consider the matrix$X^T Y^T \in \mathbb{R}^{n \times j}$, which contains all the inner products between data and prototypes. From the architecture and notation of the dual layer, it follows $= \Omega X^T$, which yields:

$$X^T Y^T = X^T X \Omega^T = G \Omega^T \tag{56}$$

where the sample autocorrelation data matrix G is the Gram matrix. The Euclidean distance matrix$edm(X, Y) \in \mathbb{R}^{n \times j}$, which contains the squared distances between the columns of X and Y, i.e., between data and prototypes, is given by [64]:

$$edm(X, Y) = diag\left(X^T X\right) 1_j^T - 2X^T Y^T + 1_n diag\left(YY^T\right)^T \tag{57}$$

where $diag(A)$ is a column vector containing the diagonal entries of A and 1_r is the r-dimensional column vector of all ones. It follows:

$$edm(X, Y) = diag(G) 1_j^T - 2G\Omega^T + 1_n diag\left(YY^T\right)^T \tag{58}$$

and considering that $YY^T = \Omega X^T \left(\Omega X^T\right)^T = \Omega G \Omega^T$, it holds:

$$edm(X, Y) = f(G, \Omega) = 1_n diag\left(\Omega G \Omega^T\right)^T - 2G\Omega^T + diag(G) 1_j^T \tag{59}$$

as a quadratic function of the dual weights. This function allows the straight computation of the edm from the estimated weights, which is necessary in order to evaluate the Voronoi sets of the prototypes for the quantization loss.

Conclusion

This work opens a novel field in neural network research where unsupervised gradient-based learning joins competitive learning. Two novel layers, VCL, as a representative of the competitive layer, and DCL, its dual, are introduced for unsupervised deep learning applications. Despite VCL is just an adaptation of a standard competitive layer for deep neural architectures, DCL represents a completely novel approach. The relationship between the two layers has been extensively analyzed and their equivalence in terms of architecture has been proven. Nonetheless, the advantages of the dual approach justify its employment. Unlike all other clustering techniques, the parameters of DCL evolve in a n-dimensional submanifold which does not depend on the number of features d as the layer is trained on the transposed input matrix. As a result, the dual approach is natively suitable for tackling high-dimensional problems. The limitation of the proposed theory follows from the choice of using the stochastic approximation theory, which only yields the asymptotic properties of the gradient flows of the two networks. For this reason, the analysis of the dynamics of two flows has been added. The other important advantage of DCL is the fact that it outputs the prototypes. This requires either a batch or minibatch learning. This works the same as the classical neural module outputs and can be naturally embedded in the backpropagation rule. Hence, unlike VCL, and, of course, the traditional deep clustering approaches, DCL can be perfectly integrated in a deep neural framework, thus allowing to exploit the advantages of both.

The flexibility and the power of the approach pave the way towards more advanced and challenging learning tasks; an upcoming paper will compare DCL on renowned benchmarks against state-of-the-art clustering algorithms. Further extensions of this approach may include topological nonstationary clustering [65], hierarchical clustering [12–14], core set discovery [66], incremental and attention-based approaches, or the integration within complex architectures such as VAEs and GANs, and will be studied in the future.

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement. Dr. Randazzo acknowledges funding from the research contract no. 32-G-13427–2 (DM 1062/2021) funded within the Programma Operativo Nazionale (PON) Ricerca e Innovazione of the Italian Ministry of University and Research.

Data Availability The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Consent to Participate Not applicable.

Conflict of Interest The authors declare no competing interests.

References

1. MacQueen J, others. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Oakland, CA, USA. 1967;281–97.
2. McLachlan GJ, Basford KE. Mixture models: inference and applications to clustering. M. Dekker New York. 1988.
3. Martinetz T, Schulten K, others. A "neural-gas" network learns topologies. Artif Neural Netw. 1991;397–402.
4. Bhatia SK, others. Adaptive K-means clustering. FLAIRS conference. 2004;695–9.
5. Ester M, Kriegel H-P, Sander J, Xu X, others. A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd. 1996;226–31.
6. Hebb DO. The organization of behavior: a neuropsychological theory. Psychology Press; 2005.
7. Martinetz T. Competitive Hebbian learning rule forms perfectly topology preserving maps. International conference on artificial neural networks. Springer. 1993;427–34.
8. White RH. Competitive Hebbian learning. IJCNN-91-Seattle Int Jt Conf Neural Netw. 1991;949 vols.2–.
9. Kohonen T. Self-organized formation of topologically correct feature maps. Biol Cybern. 1982;43:59–69.
10. Fritzke B. A growing neural gas network learns topologies. Advances in neural information processing systems. 1995;625–32.
11. Fritzke B. A self-organizing network that can follow non-stationary distributions. International conference on artificial neural networks. Springer. 1997;613–8.
12. Palomo EJ, López-Rubio E. The growing hierarchical neural gas self-organizing neural network. IEEE Trans Neural Netw Learn Syst. 2017;28:2000–9.
13. Barbiero P, Bertotti A, Ciravegna G, Cirrincione G, Cirrincione M, Piccolo E. Neural biclustering in gene expression analysis. Int Conf Comput Sci Comput Intell. 2017;1238–43.
14. Cirrincione G, Ciravegna G, Barbiero P, Randazzo V, Pasero E. The GH-EXIN neural network for hierarchical clustering. Neural Netw. 2020;121:57–73.
15. Pearson KLIII. On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci. 1901;2:559–72.
16. Schölkopf B, Smola A, Müller K-R. Kernel principal component analysis. International conference on artificial neural networks. Springer. 1997;583–8.
17. Demartines P, Hérault J. Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. IEEE Trans Neural Networks. 1997;8:148–54.
18. Cirrincione G, Randazzo V, Pasero E. The growing curvilinear component analysis (GCCA) neural network. Neural Netw. 2018;103:108–17.
19. Cirrincione G, Randazzo V, Pasero E. Growing Curvilinear Component Analysis (GCCA) for dimensionality reduction of nonstationary data. Multidiscip Approach Neural Comput. Springer. 2018;151–60.
20. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989;1:541–51.
21. Lovino M, Urgese G, Macii E, Di Cataldo S, Ficarra E. A deep learning approach to the screening of oncogenic gene fusions in humans. Int J Mol Sci. 2019;20:1645.
22. Lovino M, Ciaburri MS, Urgese G, Di Cataldo S, Ficarra E. DEEPrior: a deep learning tool for the prioritization of gene fusions. Bioinformatics. 2020;36:3248–50.
23. Roberti I, Lovino M, Di Cataldo S, Ficarra E, Urgese G. Exploiting gene expression profiles for the automated prediction of connectivity between brain regions. Int J Mol Sci. 2019;20:2035.
24. Lovino M, Montemurro M, Barrese VS, Ficarra E. Identifying the oncogenic potential of gene fusions exploiting miRNAs. J Biomed Inform. 2022;129: 104057.
25. Hu W, Miyato T, Tokui S, Matsumoto E, Sugiyama M. Learning discrete representations via information maximizing self-augmented training. arXiv preprint arXiv:170208720. 2017.
26. Yang J, Parikh D, Batra D. Joint unsupervised learning of deep representations and image clusters. Proc IEEE Conf Com Vis Pattern Recognit. 2016;5147–56.
27. Chang J, Wang L, Meng G, Xiang S, Pan C. Deep adaptive image clustering. Proc IEEE Int Conf Comput Vis. 2017;5879–87.
28. Min E, Guo X, Liu Q, Zhang G, Cui J, Long J. A survey of clustering with deep learning: from the perspective of network architecture. IEEE Access. 2018;6:39501–14.
29. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst. 2012;1097–105.
30. Hsu C-C, Lin C-W. Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data. IEEE Trans Multimedia. 2017;20:421–9.
31. Fard MM, Thonet T, Gaussier E. Deep k-means: jointly clustering with k-means and learning representations. Pattern Recogn Lett. 2020;138:185–92.
32. Jabi M, Pedersoli M, Mitiche A, Ayed IB. Deep clustering: on the link between discriminative models and k-means. IEEE Trans Pattern Anal Mach Intell. 2019;43:1887–96.
33. Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. AIChE J. 1991;37:233–43.
34. Huang Q, Zhang Y, Peng H, Dan T, Weng W, Cai H. Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning. Neurocomputing. 2020;404:340–50.
35. Opochinsky Y, Chazan SE, Gannot S, Goldberger J. K-autoencoders deep clustering. ICASSP 2020 - 2020. IEEE Int Conf Acoust Speech Signal Process (ICASSP). 2020;4037–41.
36. Li K, Ni T, Xue J, Jiang Y. Deep soft clustering: simultaneous deep embedding and soft-partition clustering. J Ambient Intell Humaniz Comput. 2021;1–13.
37. Roselin AG, Nanda P, Nepal S, He X. Intelligent anomaly detection for large network traffic with optimized deep clustering (ODC) algorithm. IEEE Access. 2021;9:47243–51.
38. Kingma DP, Welling M. Auto-encoding variational bayes. arXiv preprint arXiv:13126114. 2013.
39. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H. Variational deep embedding: an unsupervised and generative approach to clustering. arXiv preprint arXiv:161105148. 2016.
40. Dilokthanakul N, Mediano PA, Garnelo M, Lee MC, Salimbeni H, Arulkumaran K, et al. Deep unsupervised clustering with gaussian mixture variational autoencoders. arXiv preprint arXiv:161102648. 2016.

41. Bo D, Wang X, Shi C, Zhu M, Lu E, Cui P. Structural deep clustering network. Proceedings of The Web Conference. 2020;2020:1400–10.
42. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Adv Neural Inf Process Syst. 2014;2672–80.
43. Springenberg JT. Unsupervised and semi-supervised learning with categorical generative adversarial networks. arXiv preprint arXiv:151106390. 2015.
44. Chen X, Duan Y, Houthooft R, Schulman J, Sutskever I, Abbeel P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. Proc 30th Int Conf Neural Inf Process Sys. 2016;2180–8.
45. Harchaoui W, Mattei P-A, Bouveyron C. Deep adversarial Gaussian mixture auto-encoder for clustering. 2017.
46. Peng X, Feng J, Zhou JT, Lei Y, Yan S. Deep subspace clustering. IEEE transactions on neural networks and learning systems. 2020;31:5509–21.
47. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. 2014 [cited 2022 Nov 4]; Available from: https://arxiv.org/abs/1409.0473
48. Jin Y, Tang C, Liu Q, Wang Y. Multi-head self-attention-based deep clustering for single-channel speech separation. IEEE Access. 2020;8:100013–21.
49. Chen Z, Ding S, Hou H. A novel self-attention deep subspace clustering. Int J Mach Learn Cyb. 2021;1–11.
50. Shrivastava AD, Kell DB. FragNet, a contrastive learning-based transformer model for clustering, interpreting, visualizing, and navigating chemical space. Molecules. 2021;26:2065.
51. Hornik K, Stinchcombe M, White H, others. Multilayer feedforward networks are universal approximators. Neural Netw. 1989;2:359–66.
52. Rumelhart DE, Zipser D. Feature discovery by competitive learning. Cogn Sci. 1985;9:75–112.
53. Barlow HB. Unsupervised learning. Neural Comput. 1989;1:295–311.
54. Haykin S. Neural networks: a comprehensive foundation. Inc.: Prentice-Hall; 2007.
55. Lloyd S. Least squares quantization in PCM. IEEE Trans Inf Theory. 1982;28:129–37.
56. Sabin M, Gray R. Global convergence and empirical consistency of the generalized Lloyd algorithm. IEEE Trans Inf Theory. 1986;32:148–55.
57. Gray R. Vector quantization IEEE Assp Magazine. 1984;1:4–29.
58. Lovino M, Randazzo V, Ciravegna G, Barbiero P, Ficarra E, Cirrincione G. A survey on data integration for multi-omics sample clustering. Neurocomputing [Internet]. 2021 [cited 2021 Dec 10]; Available from: https://www.sciencedirect.com/science/article/pii/S0925231221018063
59. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proc 13th Int Conf Artif Intell Stat. 2010;249–56.
60. Guyon I. Design of experiments of the NIPS 2003 variable selection benchmark. NIPS 2003 workshop on feature extraction and feature selection. 2003;1–7.
61. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. Tensorflow: a system for large-scale machine learning. 12th ${$USENIX$}$ symposium on operating systems design and implementation (${$OSDI$}$ 16). 2016;265–83.
62. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
63. Barbiero P. pietrobarbiero/cola: Absolutno. 2020.
64. Dokmanic I, Parhizkar R, Ranieri J, Vetterli M. Euclidean distance matrices: essential theory, algorithms, and applications. IEEE Signal Process Mag. 2015;32:12–30.
65. Randazzo V, Cirrincione G, Ciravegna G, Pasero E. Nonstationary topological learning with bridges and convex polytopes: the G-EXIN neural network. 2018 Int Jt Conf Neural Netw (IJCNN). IEEE. 2018;1–6.
66. Ciravegna G, Barbiero P, Cirrincione G, Squillero G, Tonda A. Discovering hierarchical neural archetype sets. Prog Artif Intell Neural Syst. Springer. 2019;255–67.
67. Cirrincione G, Randazzo V, Barbiero P, Ciravegna G, Pasero E. Dual deep clustering. In: Esposito A, Faundez-Zanuy M, Morabito FC, Pasero E, editors. Applications of artificial intelligence and neural systems to data science [Internet]. Singapore: Springer Nature; 2023 [cited 2023 Oct 13]. p. 51–62. Available from: https://doi.org/10.1007/978-981-99-3592-5_5