Noname manuscript No.
(will be inserted by the editor)

Self-Supervised Autoencoders for Clustering and Classification

Paraskevi Nousi · Anastasios Tefas

Received: date / Accepted: date

Abstract Clustering techniques aim at finding meaningful groups of data samples which exhibit similarity with regards to a set of characteristics, typically measured in terms of pairwise distances. Due to the so-called curse of dimensionality, i.e., the observation that high-dimensional spaces are unsuited for measuring distances, distance-based clustering techniques such as the classic k-means algorithm fail to uncover meaningful clusters in high-dimensional spaces. Thus, dimensionality reduction techniques can be used to greatly improve the performance of such clustering methods. In this work, we study Autoencoders as Deep Learning tools for dimensionality reduction, and combine them with k-means clustering to learn low-dimensional representations which improve the clustering performance by enhancing intra-cluster relationships and suppressing inter-cluster ones, in a self-supervised manner. In the supervised paradigm, distance-based classifiers may also greatly benefit from robust dimensionality reduction techniques. The proposed method is evaluated via multiple experiments on datasets of handwritten digits, various objects and faces, and is shown to improve external cluster quality measuring criteria. A fully supervised counterpart is also evaluated on two face recognition datasets, and is shown to improve the performance of various lightweight classifiers, allowing their use in real-time applications on devices with limited computational resources, such as Unmanned Aerial Vehicles (UAVs).

Keywords Autoencoders, Clustering, Classification, Dimensionality Reduction, Deep Learning

1 Introduction

Clustering refers to the process of identifying groups of samples from a data set, which exhibit similarity with regards to a set of features [10,38]. Clustering

Aristotle University of Thessaloniki, Department of Informatics, Thessaloniki, 54124, Greece
E-mail: paranous@csd.auth.gr · tefas@aiia.csd.auth.gr

techniques are typically fully unsupervised, in the sense that they don't make use of any prior knowledge on the data, such as labels which might accompany the data points. This distinguishes the clustering task from classification, where the prior knowledge of class labels of known data points is exploited for the purpose of generalizing and making predictions for unfamiliar samples.

Amongst the most widely used clustering techniques, k-means [32] is perhaps the most popular one, although many different clustering algorithms have been proposed since its first introduction, many of which are summarized in [22]. Despite the plethora of algorithms that have been proposed for the purpose of clustering, the popularity of k-means lies in its simplicity as well as its versatility and extensibility. Many variants of the algorithm have also been proposed, including a fuzzy variant named fuzzy c-means [5,17], extensions that facilitate large datasets which may contain categorical values [21,50], and variants that exploit the kernel trick [13,48] among others. As k-means begins by randomly choosing k data points as the initial cluster centers, some studies focused on improving this random initialization [9,24], such as the global k-means method [31], or k-means++ [2].

Clustering techniques based on measuring distances between data points, including the k-means algorithm, are plagued by the dimensionality curse [4]. It has been shown that in high-dimensional spaces the distances between data points may become indistinguishable and thus incapable of providing a valid measure of closeness between the data samples [1]. For this purpose, Principal Component Analysis (PCA) [23], was used in [14] to first reduce the dimensionality of the data points thus transforming the points into uncorrelated, low-dimensional data samples on which k-means clustering is performed. Linear Discriminant Analysis (LDA) [16,33], has also been used in conjunction with k-means clustering to simultaneously reduce the dimensionality of the data points and partition the low-dimensional points into meaningful clusters [15]. More recently, a feature selection approach was studied and presented in [7], as well as two feature extractions methods, for the purpose of dimensionality reduction to facilitate the k-means clustering task.

In this paper, we study Autoencoders (AEs) [28,49] as dimensionality reduction tools while simultaneously performing clustering on the data points, for the purpose of extracting robust features and meaningful clusters in a self-supervised framework. We exploit the low-dimensional subspace produced by an AE to perform clustering and in turn use the clustering result to finetune the learned representations in a self-supervised fashion. The reconstruction space of the AE is shifted to increase intra-cluster compactness and inter-cluster separability, and multiple shifting methods are proposed. Furthermore, we show that when label information is available, it can be exploited in a similar fashion as the cluster indices produced by clustering the data, to manipulate the input space and force the AE to learn better separated low-dimensional representations. Finally, we investigate the effect of multiple gradual shifts of the reconstruction space to the learned space. The proposed supervised and self-supervised autoencoders are evaluated on various datasets for classification and clustering purposes and are shown to improve the quality of clusters

formed in the low-dimensional space in both cases, in terms of their correspondence to the actual classes present in the datasets.

The rest of this paper is organized as follows. Section 2 contains a discussion on previous related work and summarizes the advantages of the proposed methodology. An introduction and analysis of the methods used in the proposed framework is given in Section 3, while Section 4 describes the proposed method in detail. The results of the experiments conducted to showcase the performance of the proposed methods are discussed and analyzed in Section 5, and, finally, conclusions are drawn in Section 6.

2 Related Work

The effect of the curse of dimensionality on proximity-based algorithms, such as k-means clustering or k-NN classification, has been extensively studied in the past [1, 4]. The observation that distances become increasingly meaningless as the dimensionality of data increases has lead researchers to the development of more robust proximity measures, such as metric-learning approaches [52], as well as towards dimensionality reduction methods [8, 35, 39].

More specifically, in [14], the relationship between the k-means clustering objective and the principal components extracted via PCA is studied, and it is shown that the principal components comprise relaxed solutions of the cluster membership indicators of k-means clustering. Moreover, the effectiveness of using PCA for dimensionality reduction and k-means clustering is experimentally validated. Similarly, in [15], LDA is combined with PCA and k-means on a low-dimensional subspace to further reduce dimensionality and select the subspace most suitable for clustering. Finally, a provably accurate feature selection method was presented in [7], along with two feature extraction methods based on Singular Value Decomposition (SVD) and random projections.

With the advent of Deep Learning, Autoencoders were naturally studied as tools for unsupervised, non-linear feature extraction and dimensionality reduction for the tasks of clustering and classification. In [44], a clustering constraint was integrated into the standard autoencoder objective to extract feature representations which encoded cluster compactness as well as reconstruction information. Similarly, in [51], a Kullback-Leibler divergence method is integrated into the standard autoencoder's objective function to optimize clustering performance on the latent low-dimensional subspace that is learned. In [20], an Autoencoder was trained with a locality-preserving criterion and a group sparsity constraint to learn block diagonal feature representations where nonzero groups of elements correspond to clusters. In [47], a non-linear graph embedding is performed using stacked Autoencoder networks, before the k-means clustering procedure, exploiting the similarity between spectral clustering methods and autoencoders in terms of their objectives.

As Autoencoders are trained in an unsupervised fashion, research interest has also steered towards incorporating supervised information into their training process, to extract hidden representations better suited to classifica-

tion tasks. In [40], the label information is incorporated into an AE's training process by augmenting the loss function so as to include the classification error. In [43], discriminative criteria are employed by forcing pairs of representations corresponding to the same face to be closer together in the latent Euclidean subspace than to other representations corresponding to different faces. This is achieved by using a triplet loss method, although, heuristically selecting such triplets is very computationally expensive. Similarly, in [12], gated Autoencoders, which require pairs of samples as inputs, were deployed for the task of measuring similarity between parents and children. The aforementioned methods focus on either incorporating the classification error into the AE's objective, or by utilizing carefully selected tuples of samples. In contrast, the method proposed in this work does not require any complex loss functions, which may disrupt the convergence of the reconstruction error of the AE, or the selection of any specialized tuples, which imposes heavy computational costs during the training process of the AE.

In this paper, we expand upon the findings of a previous work [36,37], extending the basic notion of the target shifting process to the more challenging unsupervised case. We exploit the low-dimensional subspace produced by an AE to perform clustering and use the clustering result to finetune the learned representations in a self-supervised fashion, to extract better-formed clusters. We use the geometrical relationships in the low-dimensional space to manipulate the high-dimensional input space. This manipulation is in turn reflected onto the learned low-dimensional subspace, where external clustering measures are improved. We also propose the use of multiple, gradual shifts in the reconstruction space and investigate their effect on the low-dimensional space.

3 Background

3.1 Autoencoders

Autoencoders are Artificial Neural Networks which map their input to a latent representation typically of lower dimension, through non-linear transformations, and reconstruct their input through this intermediate representation. In its simplest form, an AE may be comprised of only one hidden layer of neurons, an input layer corresponding to its input data samples and an output layer, containing the same number of neurons as the input layer.

Generally, an autoencoder consists of an encoding part and a decoding part, both of which may contain multiple layers for deeper AEs, and the decoding part has a layout that is symmetric to its encoding counterpart. Each layer of neurons $l = 1, \ldots, L$ is accompanied by a weight matrix, a bias vector and a non-linear activation function, which transform the input to produce the output of the neuron. The output of the encoding part of the network, which we denote by $\mathbf{y}$, may be regarded as a compressed version of the network's input, when the number d of neurons it contains is less than the dimension D

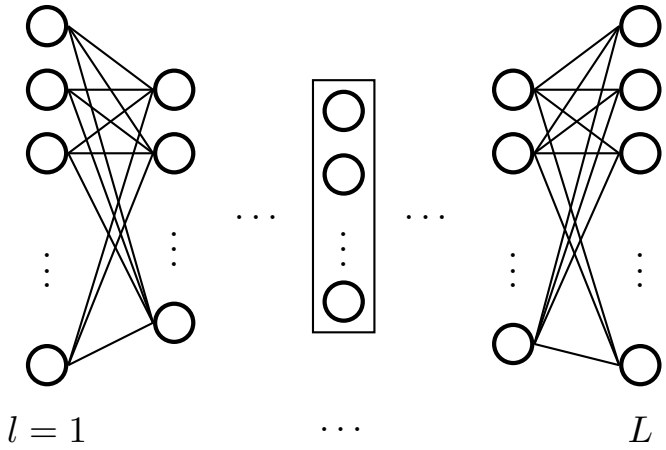


Fig. 1 Typical Deep Autoencoder architecture. The input and output layer consist of the same number of neurons, equal to the dimensionality of the data. Multiple non-linear layers of neurons lead to the final layer of the encoding part (enclosed in a rectangle), which produces low-dimensional representations of the AE's input. After the low-dimensional representation has been fed forward through the decoding part, the network is trained so that its output resembles its input.

of the input, or $d \ll D$. The encoded samples then lie on the low-dimensional space $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^T \in \mathbb{R}^{N \times d}$. The typical architecture of an AE as it was described is illustrated in Figure 1.

Formally, if $\mathbf{x}^{(l-1)}$ denotes the output of the $(l-1)$-th layer, which is the input of the l-th layer of the network, and $\mathbf{x}^{(l)}$ is the output the l-th layer produces for this input, given by:

$$\mathbf{x}^{(l)} = f(\mathbf{A}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}) \tag{1}$$

where $\mathbf{A}^{(l)}$ is the weight matrix and $\mathbf{b}^{(l)}$ is a bias vector which accompany the l-th layer, and $\mathbf{x}^{(1)}$ is the network's input, $\mathbf{x}^{(1)} \equiv \mathbf{x}$.

The weight matrices of the encoding and decoding parts may be tied by certain rules, e.g., the decoding weight matrix may be the transpose of the encoding weight matrix as the respective layer share a symmetric architecture. This constraint significantly reduces the number of free parameters that the network must learn but it is optional.

The goal of an AE is to optimize its parameters ϑ, which include the weights and biases of all layers, i.e., $\vartheta = \{\mathbf{A}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$, so that the reconstruction error is minimized. For example, using the Mean Square Error (MSE) to measure this error, the network's objective:

$$\underset{\vartheta}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_i^{(L)} - \mathbf{x}_i\|_2^2 \tag{2}$$

over all training samples, where $\mathbf{x}_i^{(L)}$ denotes the output of the final layer of the AE for the i-th input sample and can be regarded as a reconstruction of that input.

A network such as the one described above can be trained effectively using a variation of the backpropagation algorithm [41], in combination with a gradient descent optimization method, such as Stochastic Gradient Descent (SGD) [54], or an adaptive optimizer, such as Adam [25], so that it reconstructs its input with a small error.

A variation of the simple Autoencoder, the Denoising Autoencoder [49] corrupts its input with some kind of noise and then attempts to reconstruct the original input without the noise. This approach helps the network discover more robust features instead of simply learning the identity of its input. A popular corruption method choice is dropout noise, which forces a value of zero to a number of features, although other methods may be used, such as adding Gaussian noise to the input. Dropout may also be used to drop features in intermediate layers during the training process, as this has been shown to improve the generalization ability of Neural Networks [45].

3.2 Clustering

Clustering techniques aim at discovering groups of data points which exhibit likeness, indicated by a similarity measure. Samples assigned to the same cluster are regarded as similar to one another, whereas samples assigned to different clusters must differ significantly with regards to said measure.

The simplicity of k-means in combination with its effectiveness and efficiency contribute to its unyielding popularity. The algorithm begins by choosing K samples from the data set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times D}$, either at random or through an initialization algorithm such as k-means++ [2], as the initial cluster centers $\mathbf{v}_k^{(0)}, k = 1, \ldots, K$. Every sample $\mathbf{x}_i \in \mathbb{R}^D$ is then assigned to the cluster whose center lies the closest to it. Various metrics can be used for this step, the most popular being the ℓ^2 norm of their difference, corresponding to the Euclidean distance between two points. The metric used in this step highly affects the formation of the clusters and the final clustering produced, and some studies have focused on learning a suitable metric for this task [11].

After all data samples have been assigned to one of the k clusters, the cluster centers are recomputed as the mean vector of the data points assigned to each of them. The process is repeated iteratively, with the cluster centers at the t-th iteration given by:

$$\mathbf{v}_k^{(t)} = \frac{1}{|\mathcal{V}_k^{(t)}|} \sum_{\mathbf{x}_i \in \mathcal{V}_k^{(t)}} \mathbf{x}_i, \quad k = 1, \ldots, K \tag{3}$$

where $\mathcal{V}_k^{(t)}$ is the set of samples assigned to the k-th cluster on the t-th iteration of the algorithm. The algorithm stops when the cluster centers converge, that is $\mathbf{v}_k^{(t)} = \mathbf{v}_k^{(t-1)}, \forall k$, or after a predefined number of iterations.

3.3 Classification

In this Section we provide an introduction to the classifiers used to evaluate the performance of the features extracted by the supervised proposed AEs. Although more complex and more accurate algorithms have emerged in recent years and excelled at classification tasks, especially those based on deep Convolutional Neural Networks (CNNs) [26], for visual information classification, we focus on more simplistic classifiers which are more lightweight and better suited for deployment on devices of limited capabilities. Furthermore, simpler classifiers rely more on the feature extraction step of the classification pipeline, thus better reflecting the quality of the features used.

Multilayer Perceptron A Multilayer Perceptron (MLP) [6], without hidden layers, maps its input to output neurons which correspond to the various classes describing the data. Thus the input layer has as many neurons as is the dimensionality of the input data, and the output layer has as many neurons as is the number of classes. The softmax function is typically used as the activation function in the output layer of neurons, in order to produce a probability distribution over the possible classes:

$$\sigma_j(\mathbf{x}_i) = \frac{e^{x_{i,j}}}{\sum_{k=1}^{C} e^{x_{i,k}}} \tag{4}$$

where C is the total number of classes and $x_{i,j}$ indicates the j-th element of vector $\mathbf{x}_i$. This allows for the optimization of the network's parameters via the minimization of the categorical cross-entropy loss function:

$$\underset{\varphi}{\operatorname{argmin}} = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j} \log \sigma_j(\mathbf{x}_i) \tag{5}$$

with respect to the network's parameters φ over all training samples, via a variation of gradient descent optimization.

Nearest Centroid The Nearest Centroid (NC) classifier assigns samples to the class whose centroid (i.e., mean of samples belonging to that class) lies the closest to them in space. The dimensionality of the data heavily affects the performance of this classifier, as the distances between very high-dimensional data have been shown to be inefficient for determining neighboring samples [1].

k-Nearest Neighbors Similarly to the NC classifier, the k-Nearest Neighbors (kNN) [3] classifier assigns samples to the class to which the majority of its k nearest neighbors belongs to. The dimensionality of the data affects the performance of this classifier as well, as it requires the computation of distances between all data samples.

Support Vector Machine A Support Vector Machine (SVM) [18] aims to find the optimal hyperplane to separate samples belonging to different classes. The kernel method can be utilized by SVMs to map the input data into a higher-dimensional space which is more easily separable by linear hyperplanes (e.g., Radial Basis Function (RBF) kernel).

4 Proposed Methodology

The latent representation produced by an AE is learned via minimizing the network's reconstruction error. Intuitively, if the target to be learned for each sample is a modified version of itself, such that it is closer to other samples of the same class, the network will learn to reconstruct samples that belong to classes which are more easily separable. This modification should be reflected by the intermediate representation, thus producing well-separated low-dimensional representations of the network's input data.

Let $\tilde{\mathbf{x}}_i^{(t)}$ be the target reconstruction of sample $\mathbf{x}_i$, $i = \{1, \ldots, N\}$, then for $t = 0$, $\tilde{\mathbf{x}}_i^{(0)} \equiv \mathbf{x}_i$ corresponds to the standard AE targets. The *target shifting* process may be repeated multiple times, each time building on top of the previous iteration. The exponent t denotes the current iteration. The objective to be minimized for the proposed autoencoders becomes:

$$\underset{\vartheta}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_i^{(L)} - \tilde{\mathbf{x}}_i^{(t)}\|_2^2 \tag{6}$$

summed over all data samples. Using this objective and given a sample $\mathbf{x}$ from the original dataset, the autoencoder is trained to learn to reconstruct its shifted version. This is reflected in the low-dimensional representation, which is then used for clustering and classification purposes. Furthermore, the AE learns to generalize this transformation, and applies it to unseen test samples, thus moving them towards same-class or same-cluster samples. This is corroborated by the experimental results analyzed in Section 5.

4.1 Improving Cluster Separability

Cluster separability may be improved by increasing either intra-cluster compactness or inter-cluster separability. In the following paragraphs, we propose various sample shifting methods for both cases of cluster separability enhancement.

Intra-cluster compactness We adopt a general sample shifting procedure of the form:

$$\begin{aligned}\tilde{\mathbf{x}}_i^{(t+1)} &= \tilde{\mathbf{x}}_i^{(t)} + \alpha(\mathbf{m}_i^{(t)} - \tilde{\mathbf{x}}_i^{(t)}) \\ &= (1-\alpha)\tilde{\mathbf{x}}_i^{(t)} + \alpha\mathbf{m}_i^{(t)}\end{aligned} \tag{7}$$

where $\mathbf{m}_i^{(t)}$ is chosen to be a linear combination of the shifted versions of all samples in the dataset for the current iteration, i.e.:

$$\mathbf{m}_i^{(t)} = \sum_{j=1}^{N} W_{ij}\tilde{\mathbf{x}}_j^{(t)} \tag{8}$$

Thus, Equation (7) is equivalent to a linear combination of the sample itself and all other samples in the dataset with weights $\mathbf{W}$. The hyperparameter α controls the sharpness of the shift in space, i.e., larger values correspond to larger displacements.

For the task of forming well separated clusters, the weight matrix can be defined to dictate that intra-cluster relationships should be strengthened while inter-cluster ones should be weakened.

Let $\mathcal{V}_k$ be the k-th cluster obtained by performing k-means clustering on a set of data samples $\mathbf{Y}$, which are low-dimensional representations of $\mathbf{X}$ extracted by a standard autoencoder, and $\mathbf{v}_k \in \mathbb{R}^d, k = 1, \ldots, K$ be the center of this cluster. The center of a cluster, i.e., the mean vector of all samples belonging to that cluster, is an obvious choice to shift samples towards, so as to increase the cluster's compactness. In this case, the weight matrix elements W_{ij} for a given sample $\mathbf{x}_i$ have a value of $1/|\mathcal{V}_k|$ for all samples $\mathbf{x}_j$ belonging to the same cluster as $\mathbf{x}_i$, or, formally:

$$W_{ij} = \begin{cases} 1/|\mathcal{V}_k| & \text{if } \{\mathbf{x}_i, \mathbf{x}_j\} \in \mathcal{V}_k \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

For each sample, the weights that affect its optimal target reconstruction sum up to one, as there are $|\mathcal{V}_k|$ such samples, each affecting $\mathbf{x}_i$ with $1/|\mathcal{V}_k|$, where $\sum_j W_{ij}\tilde{\mathbf{x}}_j$ is equivalent to $\mathbf{x}_i$'s cluster center. This observation allows for the direct use of the cluster centers to find the optimal target reconstructions, instead of fully defining the matrix $\mathbf{W}$, which can become prohibitively large for datasets consisting of many samples.

Another intuitive definition for the relationship matrix $\mathbf{W}$ is for only neighboring samples to affect each sample $\mathbf{x}_i$, instead of the entirety of samples belonging to the same cluster. This entails searching for each sample's closest neighbors within their cluster. Let $\mathcal{N}_i$ define the set of $\mathbf{x}_i$'s n closest neighbors within their cluster $\mathcal{V}_k$, then $\mathbf{W}$ can be defined as:

$$W_{ij} = \begin{cases} 1/|\mathcal{N}_i| & \text{if } \mathbf{x}_j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Then, the optimal target reconstruction can be given by Equation (7), with weights given by Equation (10), as they once again sum up to one for each sample. The full definition of the weight matrix can be avoided by finding the neighbors of each samples and computing each target reconstruction separately.

Inter-cluster separability Simultaneously with the cluster-contracting methods described above, penalty weights can be defined to better separate the various clusters from each other. For this task, we adopt the following general shifting transformation:

$$\begin{aligned}\tilde{\mathbf{x}}_i^{(t+1)} &= \tilde{\mathbf{x}}_i^{(t)} - \beta(\mathbf{m}_i^{(t)} - \tilde{\mathbf{x}}_i^{(t)}) \\ &= (1+\beta)\tilde{\mathbf{x}}_i^{(t)} - \beta\mathbf{m}_i^{(t)}\end{aligned} \tag{11}$$

where β is a hyperparameter which controls the weight of the shifting process. Once again, the vector $\mathbf{m}_i^{(t)}$ is chosen to be a linear combination of all data samples with weights given by a matrix $\mathbf{W}$.

Let $\mathcal{V}_{-k}$ denote the union of all clusters but the k-th, e.g, $\mathcal{V}_{-3} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_4 \cup \cdots \cup \mathcal{V}_K$. We can then formally define the shift of a sample $\mathbf{x}_i$ away from all rivaling samples by defining $\mathbf{W}$ as:

$$W_{ij} = \begin{cases} -1/|\mathcal{V}_{-k}| & \text{if } \mathbf{x}_j \in \mathcal{V}_{-k} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Another option would be to only shift samples away from their closest rival-cluster samples instead of the entirety of their rivals. Let $\mathcal{R}_i$ define the set of samples which belong to clusters different than the cluster that $\mathbf{x}_i$ belongs to and constitute its r such closest rivals. In this case, the penalty weights can be defined as:

$$W_{ij} = \begin{cases} -1/|\mathcal{R}_i| & \text{if } \mathbf{x}_j \in \mathcal{R}_i \\ 0 & \text{otherwise} \end{cases}. \tag{13}$$

4.2 Improving Class Separability

In the supervised case, the new targets may be shifted so that the samples are moved towards their class centroids with weights given by:

$$W_{ij} = \begin{cases} 1/|\mathcal{C}_k| & \text{if } \{\mathbf{x}_i, \mathbf{x}_j\} \in \mathcal{C}_k \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

applied to Equation (7), where $\mathcal{C}_k$ is set of samples belonging to the k-th class.

Respectively, the distances between samples and centroids of rival classes could be enlarged by moving each sample away from the mean of all samples belonging to other classes by defining weights:

$$W_{ij} = \begin{cases} -1/|\mathcal{Q}_i| & \text{if } \mathbf{x}_j \in \mathcal{Q}_i \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $\mathcal{Q}_i$ is the set of samples belonging to a class different than the class $\mathbf{x}_i$ belongs to. In practice, Equation (15) could be modified to only include the neighbors of rival classes within a given range of each sample, or only the top k nearest rivaling neighbors to the i-th sample, instead of the mean of all rival class samples.

4.3 Cluster Contracting & Class Separating Autoencoders

To perform k-means clustering on the low-dimensional representations of the data, first a standard autoencoder is trained to convergence to extract $\mathbf{Y}$. Then k-means can be run on the entirety of the dataset. New targets can then be derived using Equation (7) with weights given by Equations (9) or (10) to increase intra-class compactness, in combination with Equation (11) with weights given by Equations (12) or (13) to increase inter-class separability.

The Autoencoder is trained to optimize Equation (6) over all training samples, starting from its previous point of convergence. Then, k-means can be run on the dataset using the previously found cluster centers as the initial centers. The shifting process can be repeated multiple times, followed by training the Autoencoder with the new targets and performing k-means on the obtained low-dimensional representations. The entire procedure is summarized in Algorithm 1. After each convergence of the AE to the new targets, k-means is performed again starting from the previously found centers to adjust the clusters to the new positions of the samples.

In the fully-supervised case where class labels are available, the new target reconstructions may be obtained before training the AE, and thus begin training with the shifted versions. This variation is summarized in Algorithm 2.

It should be noted that in the clustering scenario, due to its unsupervised nature and the risk of extracting entangled clusters, large values for the hyperparameters α and β will further deteriorate the performance of clustering by debilitating the representation learning capability of the AE. Thus, the final clusters will be well-formed, in terms of cluster separability and compactness, but their homogeneity in terms of the natural classes present in them will have been compromised.

In contrast, in the classification scenario, where groundtruth labels are available, larger values can be used safely. This is especially the case for α, which controls the intra-class compactness. The inter-class separability procedure controlled by β however, is not as informative: shifting a sample away from its rivals, no matter the certainty of their dissimilarity, may lead the sample to a new position where it is still falsely. This is because the rivals of a sample may lie in adversarial positions.

Finally, one should note that computing a full shifting matrix $\mathbf{W}$, can quickly become inefficient as its size is $N \times N$, where N is the number of samples in a dataset. However, the full computation of these matrices can be avoided in all cases, by updating each sample online. This entails first computing all cluster/class centers or every samples nearest neighbors and rivals and secondly shifting each sample towards the mean of its center or neighbors and away from the mean of rival centers or neighbors, as discussed.

Algorithm 1 Cluster Separating Autoencoder Training Procedure

Input Dataset $\mathbf{X} \in \mathbb{R}^{N \times D}$
Output Low-dimensional representations $\mathbf{Y} \in \mathbb{R}^{N \times d}$, Clusters $\{\mathcal{V}_k\}_{k=1}^{K}$

function SampleShift($\tilde{\mathbf{X}}, \mathbf{W}_{intra}, \mathbf{W}_{inter}, \alpha, \beta$)
  $\tilde{\mathbf{X}} \leftarrow (1-\alpha)\tilde{\mathbf{X}} + \alpha \mathbf{W}_{intra}\tilde{\mathbf{X}}$
  $\tilde{\mathbf{X}} \leftarrow (1+\beta)\tilde{\mathbf{X}} - \beta \mathbf{W}_{inter}\tilde{\mathbf{X}}$
end function

Obtain $\mathbf{Y}$ by optimizing Equation (2)
$\{\mathcal{V}_k\}_{k=1}^{K} \leftarrow k$-means($\mathbf{Y}$, random)
for $t = 0, \ldots, T-1$ **do**
  $\mathbf{W}_{intra} \leftarrow$ Equation (9) or (10)
  $\mathbf{W}_{inter} \leftarrow$ Equation (12) or (13)
  $\tilde{\mathbf{X}}^{(t+1)} \leftarrow$ SampleShift($\tilde{\mathbf{X}}^{(t)}$, $\mathbf{W}_{intra}$, $\mathbf{W}_{inter}$, α, β)
  Retrain by optimizing Equation (6)
  $\{\mathcal{V}_k\}_{k=1}^{K} \leftarrow k$-means($\mathbf{Y}, \{\mathbf{v}_k\}_{k=1}^{K}$)
end for

Algorithm 2 Class Separating Autoencoder Training Procedure

Input Dataset $\mathbf{X} \in \mathbb{R}^{N \times D}$
Output Low-dimensional representations $\mathbf{Y} \in \mathbb{R}^{N \times d}$

function SampleShift($\tilde{\mathbf{X}}, \mathbf{W}_{intra}, \mathbf{W}_{inter}, \alpha, \beta$)
  $\tilde{\mathbf{X}} \leftarrow (1-\alpha)\tilde{\mathbf{X}} + \alpha \mathbf{W}_{intra}\tilde{\mathbf{X}}$
  $\tilde{\mathbf{X}} \leftarrow (1+\beta)\tilde{\mathbf{X}} - \beta \mathbf{W}_{inter}\tilde{\mathbf{X}}$
end function

for $t = 0, \ldots, T-1$ **do**
  $\mathbf{W}_{intra} \leftarrow$ Equation (14)
  $\mathbf{W}_{inter} \leftarrow$ Equation (15)
  $\tilde{\mathbf{X}}^{(t+1)} \leftarrow$ SampleShift($\tilde{\mathbf{X}}^{(t)}$)
  Obtain $\mathbf{Y}$ by optimizing Equation (6)
end for

5 Experimental Results

Our experiments were made using an NVIDIA GTX 1080 GPU and an Intel Core i7-4930K CPU at 3.40GHz. The training time is no different from that

of a standard AE, although the sample shifting process causes delays between successive training processes. Moving data from GPU to CPU and vice versa is quite time consuming and inefficient, but can be avoided by running the shifting methods and k-means directly on GPU.

5.1 Datasets

We evaluate the proposed method on the MNIST [29] test set, for hand-written digit recognition, which consists of 10000 grayscale images of size 28×28 making the original dimension equal to 784. We also evaluate the method on the COIL-20 dataset [34], containing 1440 grayscale 32×32 sized images of various objects belonging to 20 categories. Finally, the ORL [42] and Yale B [30] datasets are used. ORL consists of 400 grayscale images of 40 subjects while Yale B consists of 165 grayscale images of 15 subjects. The images are resized to 32×32 making the original dimension equal to 1024.

For the MNIST dataset, we set the dimensionality of the hidden representation equal to $d = 10$, equal to the number of naturally occurring classes in these datasets. For the other datasets, we set the dimensionality of the hidden representation to be $d = 32$. The number of clusters for k-means clustering is set equal to the number of classes K for each dataset. This information is summarized in Table 1 for all the datasets used.

Dataset	**N**	**K**	**D**	**d**
MNIST	70000	10	784	10
COIL-20	1440	20	1024	32
ORL	400	40	1024	32
Yale	2452	38	1024	32

Table 1 Summary of the datasets used in the conducted *clustering* experiments, with N being the number of samples present, K being the number of classes and number of clusters extracted by k-means, D being the original dimension corresponding to the number of pixels and d being the dimension of the representation learned by the proposed autoencoders.

In the supervised scenario, the proposed method is evaluated on the ORL faces dataset as well as the Extended Yale B dataset [30]. For both datasets, the grayscale images depicting the faces to be recognized are resized to 32×32, meaning the original data dimension is 1024. The dimension is downscaled by a factor of 4, down to 256, by the AEs.

For the ORL dataset, five-fold cross-validation is commonly used for the conduction of experiments with this dataset, where five experiments are conducted using 80% of the images per person as training data and selecting a different portion of the dataset for each fold such that all images serve as training and testing data at different runs.

The cropped version of the Yale dataset is used in our experiments, which contains images depicting 38 individuals under severe lighting variations and slight pose variations. Typically, half of the images per person are selected

as training data and evaluation is performed on the remaining half images. We follow the same dataset splitting methodology performed five times and average the results over all folds.

5.2 Metrics

To evaluate the performance of the proposed method we use two standard external clustering metrics, accuracy and normalized mutual information. The clustering accuracy measure requires the computation of the best one-to-one mapping of cluster assignments to class labels for each labels, which can be found efficiently by using the Hungarian algorithm [27]. Formally, it is defined as:

$$ACC = \frac{1}{N} \sum_{i=1}^{N} \delta[l_i - m(c_i)] \tag{16}$$

where m denotes the best one-to-one mapping of cluster assignments c_i for all samples to real labels l_i, and $\delta[\cdot]$ denotes the Dirac delta function.

The normalized mutual information score [46], is defined as the mutual information between the cluster assignments and real class labels over the square root of the product of the entropies of the two assignments:

$$NMI = \frac{\mathcal{I}(c, l)}{\sqrt{\mathcal{H}(c)\mathcal{H}(l)}} \tag{17}$$

where c denotes the final clustering assignment (i.e., an integer $1, \ldots, K$ for each sample) and l denotes the real class assignment.

For the classification experiments we report the mean classification accuracy and standard deviation over five runs for both datasets, as aforementioned.

5.3 Results

In both the supervised and self-supervised scenario, the same architecture, number of epochs and initialization was used for the standard AE and the proposed AEs for fair comparison between the obtained results. In total, the target shifting process is applied five times over the training process of the AE. The results for both cases are discussed below.

5.3.1 Clustering

For all datasets we use a D-512-d encoder architecture with sigmoid non-linearities and a symmetrical decoder with a linear final layer. We perform five iterations of the proposed algorithm, thus splitting the training process into five parts: in the first iteration, a standard AE is trained to convergence.

Then, new targets are computed using combinations of the shifting methods described in Section 4.1, and the training process picks up with these targets from its last point of convergence, and this is repeated four times to gradually form more compact clusters. We use the same initialization method for all experiments, which is k-means++ with the same random seed, ensuring the reported results are fair.

The accuracy and normalized mutual information scores for all datasets are summarized in Table 2. The results indicate that the proposed method improves the performance of k-means clustering for all of the evaluated datasets and shifting methods. We evaluate the following settings:

1. the original D-dimensional feature vectors, corresponding to pixel intensities (No AE)
2. the d-dimensional latent representations achieved by a standard AE (AE)
3. the d-dimensional latent representations achieved by the proposed AEs where the targets were shifted:
   (a) towards their cluster centers, using Equation (9) (CCAE-1)
   (b) towards their cluster center as well as away from the nearest rival-cluster center, using Equations (9) and (12) (CCAE-2)
   (c) towards their same-cluster neighbors, using Equation (10) (CCAE-3)
   (d) towards their same-cluster neighbors as well as away from their nearest rival-cluster neighbors, using Equations (10) and (13) (CCAE-4)

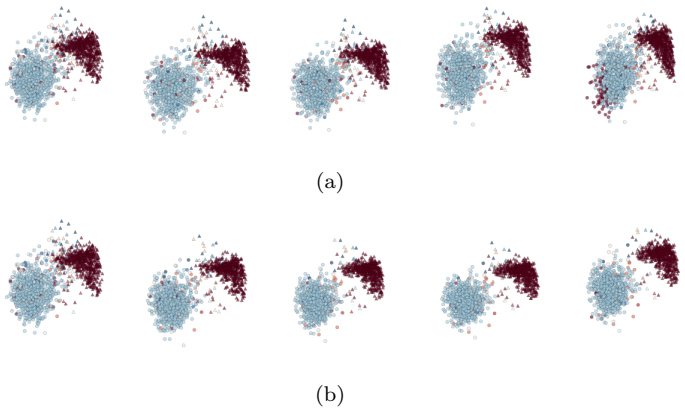
For all datasets, the clustering performance is improved even when using a standard AE to produce low-dimensional representations over the performance achieved when using the high-dimensional pixel intensity representations. Furthermore, the proposed CCAEs improve upon the standard AEs in all cases. It is also worth noting that different types of shifts work better for different datasets, as well as for different metrics. The type of method that works best in each case depends on the form of the low-dimensional representations learned by the AE, e.g., how tangled the formed clusters are, how spread out each cluster is etc. As the AE procedure is unsupervised, there is no way of knowing this information beforehand. However, from the reported results we conclude that CCAE-1 and CCAE-3 are the safest choices, always leading to improved clustering measures. CCAE-2 and CCAE-4 may yield even better results for the MNIST and ORL datasets than their safe counterparts, but are more heavily influenced by the quality of the formation of the clusters in the low-dimensional space.

Figure 2 shows 2-dimensional PCA projections of the hidden representations obtained for two clusters of the MNIST dataset with a standard Autoencoder (a), and the proposed CCAE-2 (b), over the five iterations of the algorithm. The samples belonging to each of the two clusters are denoted by circles and triangles and the variation in color corresponds to different classes. Although both clusters consist mostly of one class of digits, the representations obtained by a standard Autoencoder lead to larger overlap between the two clusters, both spatial and semantic. In contrast, using the proposed Autoencoders and gradually shifting samples towards their cluster centers and

	MNIST		COIL-20		ORL		Yale	
Method	*ACC*	*NMI*	*ACC*	*NMI*	*ACC*	*NMI*	*ACC*	*NMI*
No AE	0.5385	0.5050	0.7069	0.8002	0.5925	0.7765	0.3757	0.4731
AE	0.7571	0.6555	0.7333	0.8060	0.6700	0.8259	0.4303	0.4754
CCAE-1	0.7653	0.6642	0.7361	**0.8132**	0.6775	0.8303	0.4545	**0.5106**
CCAE-2	**0.7700**	**0.6737**	0.7312	0.8089	0.6775	0.8359	0.4606	0.5016
CCAE-3	0.7597	0.6568	**0.7381**	0.8113	0.6800	0.8324	0.4484	0.4961
CCAE-4	0.7654	0.6651	0.7326	0.8067	**0.6825**	**0.8413**	**0.4667**	0.5050

Table 2 Accuracy and normalized mutual information scores for all evaluated datasets and methods.

away from rival cluster centers, the two clusters seem to be better spatially separated as well as more homogeneous in terms of the class they represent.



(a)

(b)

Fig. 2 Hidden representations obtained by a standard AE (top) and the proposed CCAE-2 (bottom) for two clusters of the MNIST dataset over five iterations of the proposed method, projected in a two-dimensional subspace via PCA for visualization.

We furthermore perform a set of experiments using a custom dataset consisting of images of cyclists, football players and rowers, for the purpose of investigating the use of the proposed CCAEs for semi-automated annotation of the cropped images, when only the bounding boxes are provided and the class of the cropped object isn't. The images are gathered from an aerial source, and serve to facilitate the process of training models whose purpose is to detect and classify targets belonging to the aforementioned classes. The dataset used consists of 30645 images in total, with about ten thousand instances for each class. All images were resized to 224×224 and fed to a MobileNet [19] pretrained on the ImageNet dataset to extract 1280-dimensional feature vectors. This architecture was chosen due to its lightweight structure, which uses depthwise separable convolutions, reducing the number of operations required for a forward pass. Then, the proposed CCAEs were used to extract 12-dimensional features and perform clustering. The results are presented in Table 3. The proposed

CCAEs, and especially CCAE-1, offer improved clustering and could aid in the annotation process of the cropped boxes.The silhouette score is also provided in this case for the compared methods, as an internal clustering measure, to provide insight into the geometrical relationships between the formed clusters. The silhouette score is defined as:

$$SIL = \frac{b - a}{\max(a, b)}$$

where b is the distance between a sample and the nearest cluster that the sample is not a part of, and a is the mean intra-cluster distance for each sample. The best silhouette score is achieved by CCAE-1, which contracts intra-cluster relationships using the cluster centroids as attraction points. This is unsurprising, as the clusters are explicitly contracted by the CCAE-1. The two neighbor-based CCAEs however (CCAE-3 and CCAE-4), only marginally improve all metrics. This may attributed to the fact that in these cases each sample is shifted towards different points in space, defined by its set of nearest neighbors. In contrast, in the case of shifting samples towards their cluster centers, all samples belonging to the same cluster are shifted towards the same single point. This produces monomodal clusters, whereas shifting samples towards their neighbors produces multimodal clusters, where each mode may be compact but the cluster as a whole may be widespread.

Method	ACC	NMI	SIL
No AE	0.6248	0.3564	0.6746
AE	0.6786	0.3621	0.7803
CCAE-1	**0.6959**	**0.3736**	**0.8628**
CCAE-2	0.6945	0.3737	0.8602
CCAE-3	0.6877	0.3684	0.7865
CCAE-4	0.6788	0.3629	0.7852

Table 3 Accuracy, normalized mutual information and silhouette scores for the cyclists, rowers and football players dataset.

Although the proposed method relies on the quality of the low-dimensional representations learned by Autoencoders in terms of cluster formation, we argue that in a very complex problem where a deep network will be unable to produce discriminative representations, any unsupervised method will fail to uncover meaningful clusters. Shifting the samples towards the cluster centers in such a scenario will severely affect the learned representations to the point where the network might not be able to recover and produce well-formed clusters. However, shifting samples towards their nearest neighbors is a much less severe transformation which affects samples locally and may be better suited in such situations. Furthermore, even when a clustering criterion is optimized in addition to the representation, as in [53], the clustering derived heavily depends on the representations learned by the network especially in the early stages. This is a natural and implication of the unsupervised nature of the task at hand.

5.3.2 Classification

The accuracy achieved by the above classifiers is evaluated and compared for six types of inputs:

1. the original 1024-dimensional feature vectors, corresponding to pixel intensities (No AE)
2. the 256-dimensional latent representations achieved by a standard AE (AE)
3. the 256-dimensional latent representations achieved by the proposed AEs where the targets were shifted:
   (a) towards their class centers, using Equation (14) (dAE-1)
   (b) towards their class center as well as away from the nearest rival-class center, using Equations (14) and (15) (dAE-2)

The performance achieved by the evaluated classifiers for all input representations for the ORL dataset is summarized in Table 4. The dAE-2 method yields the best improvement for all classifiers, even though using the dAE-1 method the results still surpass those achieved by using the pixel intensities and the low-dimensional representations obtained by the standard AE.

Although the performance achieved by the MLP using the pixel intensities representation is quite high, the high dimensionality of the data imposes higher computational costs both in training and deployment. More importantly, the performance achieved by the less computationally intensive NC classifier is very close to the performance achieved by the MLP, and yields 10% and 15% improved accuracy results over the accuracy achieved when using the pixel intensities representation and the representation obtained by the standard AE respectively.

	MLP	NC	kNN	SVM
No AE	96.25 ± 1.58	85.25 ± 1.22	88.25 ± 5.51	90.75 ± 1.69
AE	92.75 ± 2.42	79.50 ± 1.87	82.25 ± 4.35	88.75 ± 2.50
dAE-1	96.00 ± 2.29	94.75 ± 2.29	95.25 ± 2.42	95.50 ± 1.69
dAE-2	$\mathbf{97.00 \pm 1.50}$	$\mathbf{95.50 \pm 1.87}$	$\mathbf{96.25 \pm 2.09}$	$\mathbf{96.50 \pm 1.87}$

Table 4 ORL dataset accuracy results.

Table 5 summarizes the accuracy achieved by all classifiers and input representations for the ExtYale dataset. The proposed methods outperform the baselines by a large margin. The disadvantage of the NC and kNN classifiers when data dimensionality is high becomes very clear in this dataset when the pixel intensities are used as the data representation, indicated by their extremely inaccurate predictions and low accuracy results.

The progress of the results over progressive training and shifting steps can be seen in Figure 3 and Figure 4 for the ORL and ExtYale dataset for all classifiers. This analysis indicates the importance of multiple shifts as the accuracy increases with each iteration of the proposed training method. Although the accuracy converges to a low point when using a standard AE, in

	MLP	NC	kNN	SVM
No AE	93.52 ± 1.07	10.94 ± 1.61	54.91 ± 1.54	71.19 ± 1.40
AE	88.25 ± 1.48	63.24 ± 2.06	73.70 ± 1.10	85.07 ± 1.48
dAE-1	$\mathbf{94.40 \pm 0.78}$	$\mathbf{89.93 \pm 0.68}$	91.12 ± 1.33	94.51 ± 0.63
dAE-2	94.30 ± 0.79	89.64 ± 0.91	$\mathbf{91.43 \pm 0.88}$	$\mathbf{94.61 \pm 0.74}$

Table 5 YALE dataset accuracy results.

the case of our proposed dAEs the accuracy improves with each iteration of the algorithm before reaching its point of convergence, which lies much higher than the baseline for all classifiers and both datasets.

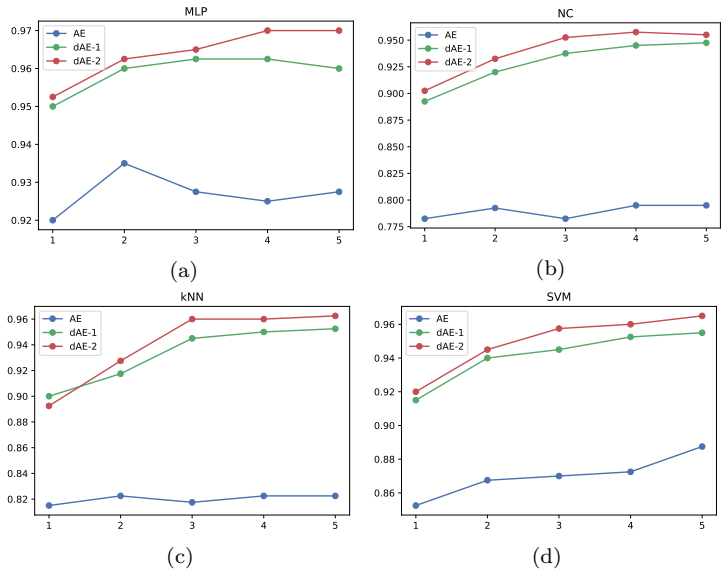


Fig. 3 Measured accuracy progress over five iterations of the proposed training method for the ORL dataset. The mean accuracy over five-fold cross validation is shown for (a) the MLP classifier, (b) Nearest Centroid, (c) k-Nearest Neighbors and (d) RBF-kernel SVM.

Overall, the results indicate that the proposed AEs are capable of generalizing well and implicitly applying the shifting process to unknown test samples. Figures 5 and 6 illustrate and ascertain this hypothesis. The left plot in both Figures is a 3-dimensional projection of the hidden representation learned by the standard AE, obtained via PCA. The middle plot is a 3-dimensional PCA projection of the representation learned by the dAE where the targets of the AE have been shifted five times in total towards their class centers. Finally, the plot on the right in both Figures is the 3-dimensional PCA projections of the hidden representation of the test samples, obtained by the same dAE as the middle projection. For both datasets, the 3D projections of the AE representation are difficult to unfold into separable formations. Through iterative repetitions of the target shifting process however, well-formed and separated

classes become obvious. Furthermore, the test samples appear at positions close to their counterparts used in the training process in the 3D projection, meaning that the AE learns to map those samples closer to their manifold.

The projections indicate that the distribution shift that occurs to the training samples also affects the test samples belonging to the same classes. This can be attributed to the fact that the testing samples follow more or less the distribution of the training samples in the input space as well as on the generalization ability of the AEs. However, in the original input space as well as the subspace produced by the standard AE, the various class distributions are not well separated at all, which makes classification by lightweight classifiers very difficult. This is also reflected by the extremely low accuracy results achieved by the NC and kNN classifiers especially in the Yale dataset.

For the MLP classifier, although the improvement in accuracy is quite small, it is worth noting that the reduction in dimensionality leads to faster computation times but also reduces the capability of the classifier: the learned weights matrix will only be of size $d \times K$ in the AE cases, whereas it is of size $D \times K$ in the No AE case, where d is the dimension of the learned representation, i.e. 256, D is the original dimension, i.e. 1024, and K is the number of classes. This means that in the No AE case, the MLP has four times as many trainable parameters as in the AE cases. Thus, the improvement, although small, is still significant.

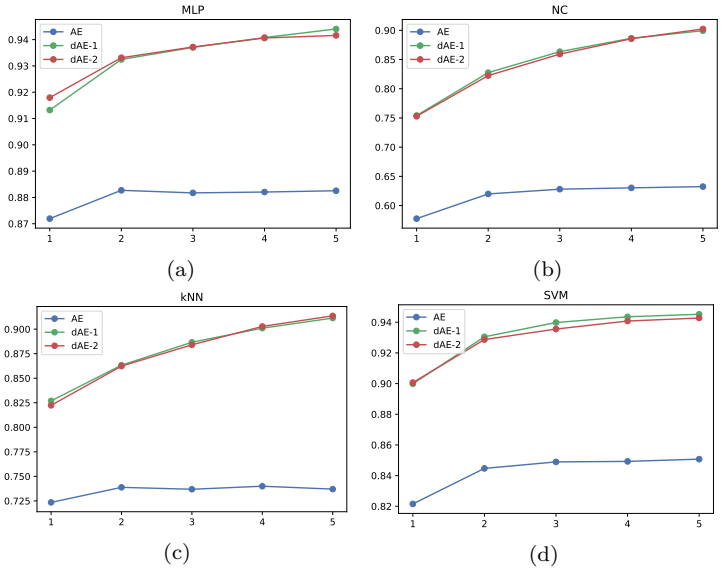
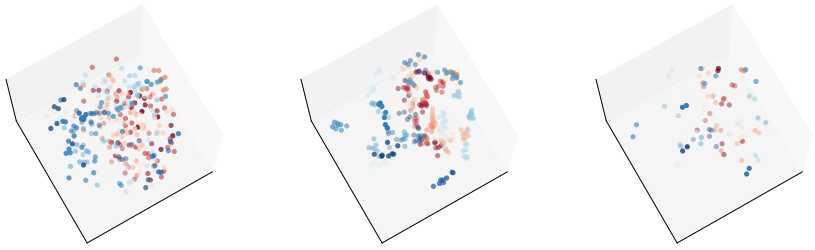
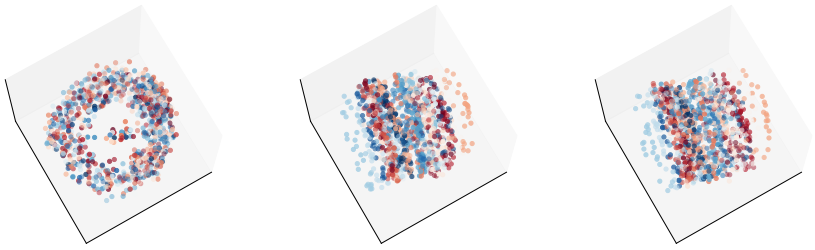


Fig. 4 Measured accuracy progress over five iterations of the proposed training method for the ExtYAle dataset. The mean accuracy over five runs each with a different 50-50 split of the dataset is shown for (a) the MLP classifier, (b) Nearest Centroid, (c) k-Nearest Neighbors and (d) RBF-kernel SVM.



(a) (b) (c)

Fig. 5 ORL hidden representation 3-dimensional projection by PCA: (a) hidden representation of the training data obtained by standard AE, (b) hidden representation of the training data obtained by dAE-1, and (c) hidden representation of the test data also obtained by dAE-1.



(a) (b) (c)

Fig. 6 YALE hidden representation 3-dimensional projection by PCA: (a) hidden representation of the training data obtained by standard AE, (b) hidden representation of the training data obtained by dAE-1, and (c) hidden representation of the test data also obtained by dAE-1.

In conclusion, the proposed dAEs significantly improve the performance of the more lightweight classifiers, allowing their use in real-time applications from devices with limited computational resources, such as Unmanned Aerial Vehicles (UAVs). Thus, they may, for example, be used in conjunction with a lightweight classifier to identify persons of interest and facilitate the tracking procedure.

6 Conclusions

We have presented several methods of manipulating the reconstruction space of an AE to guide the low-dimensional representation to form more compact and meaningful clusters. By exploiting the meaningfulness of proximity between samples in the low-dimensional space, the Autoencoder's objective is altered so that it learns to reconstruct samples shifted in space so as to lie closer to samples belonging to the same cluster and away from samples of other clusters.

This results in clusters which are more compact and well separated in space, as well as more homogeneous and complete in terms of the natural classes of the samples they contain.

The unsupervised nature of the proposed methodology renders it subject to the quality of the extracted low-dimensional representations, in terms of the compactness, separability and correspondence to meaningful classes of the clusters that form in the latent space. When the clusters are tangled and ill-formed, perhaps as the result of suboptimal convergence of the AE, improving their quality with unsupervised methods remains an open and very challenging problem. However, when the produced representations are fairly well-separated, the proposed methods can yield significantly improved clusters. Autoencoders, as Deep Learning tools, have the capability to produce such a space, where the naturally occurring clusters in a dataset are well-formed and well-separated. Thus, although the proposed method relies on the quality of the low-dimensional representations learned by Autoencoders in terms of cluster formation, the low-dimensionality of the representation in combination with its non-linear nature and the Autoencoder's ability to extract robust features in an unsupervised manner aid in the extraction of well separated clusters, which can be further polished by the proposed methodology.

In the supervised scenario, directly using the class labels leads to low-dimensional representations which better separate the classes allowing simple classifiers to achieve great classification performances. Experiments on multiple domains demonstrate the effectiveness of the proposed method and serve to show that the proposed Cluster Compacting Autoencoders extract features biased towards forming well-defined clusters.

Acknowledgments

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional spaces. In: ICDT, vol. 1, pp. 420–434. Springer (2001)
2. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 1027–1035. Society for Industrial and Applied Mathematics (2007)
3. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on pattern analysis and machine intelligence **19**(7), 711–720 (1997)
4. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful? In: International conference on database theory, pp. 217–235. Springer (1999)

5. Bezdek, J.C., Ehrlich, R., Full, W.: Fcm: The fuzzy c-means clustering algorithm. Computers & Geosciences **10**(2-3), 191–203 (1984)
6. Bhuiyan, A.A., Liu, C.H.: On face recognition using gabor filters. World academy of science, engineering and technology **28**, 51–56 (2007)
7. Boutsidis, C., Zouzias, A., Mahoney, M.W., Drineas, P.: Randomized dimensionality reduction for k-means clustering. IEEE Transactions on Information Theory **61**(2), 1045–1062 (2015)
8. Bouzas, D., Arvanitopoulos, N., Tefas, A.: Graph embedded nonparametric mutual information for supervised dimensionality reduction. IEEE transactions on neural networks and learning systems **26**(5), 951–963 (2015)
9. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Systems with Applications **40**(1), 200–210 (2013)
10. Chrysouli, C., Tefas, A.: Spectral clustering and semi-supervised learning using evolving similarity graphs. Applied Soft Computing **34**, 625–637 (2015)
11. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Proceedings of the 24th international conference on Machine learning, pp. 209–216. ACM (2007)
12. Dehghan, A., Ortiz, E.G., Villegas, R., Shah, M.: Who do i look like? determining parent-offspring resemblance via gated autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1757–1764 (2014)
13. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 551–556. ACM (2004)
14. Ding, C., He, X.: K-means clustering via principal component analysis. In: Proceedings of the twenty-first international conference on Machine learning, p. 29. ACM (2004)
15. Ding, C., Li, T.: Adaptive dimension reduction using discriminant analysis and k-means clustering. In: Proceedings of the 24th international conference on Machine learning, pp. 521–528. ACM (2007)
16. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of eugenics **7**(2), 179–188 (1936)
17. Ghosh, S., Dubey, S.K.: Comparative analysis of k-means and fuzzy c-means algorithms. International Journal of Advanced Computer Science and Applications **4**(4) (2013)
18. Guo, G., Li, S.Z., Chan, K.: Face recognition by support vector machines. In: Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on, pp. 196–201. IEEE (2000)
19. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
20. Huang, P., Huang, Y., Wang, W., Wang, L.: Deep embedding network for clustering. In: Pattern Recognition (ICPR), 2014 22nd International Conference on, pp. 1532–1537. IEEE (2014)
21. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data mining and knowledge discovery **2**(3), 283–304 (1998)
22. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern recognition letters **31**(8), 651–666 (2010)
23. Jolliffe, I.: Principal component analysis. Wiley Online Library (2002)
24. Khan, S.S., Ahmad, A.: Cluster center initialization algorithm for k-means clustering. Pattern recognition letters **25**(11), 1293–1302 (2004)
25. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
27. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics (NRL) **2**(1-2), 83–97 (1955)
28. Le, Q.V.: Building high-level features using large scale unsupervised learning. In: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pp. 8595–8598. IEEE (2013)

29. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
30. Lee, K.C., Ho, J., Kriegman, D.J.: Acquiring linear subspaces for face recognition under variable lighting. IEEE Transactions on pattern analysis and machine intelligence **27**(5), 684–698 (2005)
31. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern recognition **36**(2), 451–461 (2003)
32. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, pp. 281–297. Oakland, CA, USA. (1967)
33. Mika, S., Ratsch, G., Weston, J., Scholkopf, B., Mullers, K.R.: Fisher discriminant analysis with kernels. In: Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop., pp. 41–48. IEEE (1999)
34. Nene, S.A., Nayar, S.K., Murase, H., et al.: Columbia object image library (coil-20) (1996)
35. Nikitidis, S., Tefas, A., Pitas, I.: Maximum margin projection subspace learning for visual data analysis. IEEE Transactions on Image Processing **23**(10), 4413–4425 (2014)
36. Nousi, P., Tefas, A.: Deep learning algorithms for discriminant autoencoding. Neurocomputing (2017)
37. Nousi, P., Tefas, A.: Discriminatively trained autoencoders for fast and accurate face recognition. In: International Conference on Engineering Applications of Neural Networks, pp. 205–215. Springer (2017)
38. Passalis, N., Tefas, A.: Information clustering using manifold-based optimization of the bag-of-features representation. IEEE transactions on cybernetics (2016)
39. Passalis, N., Tefas, A.: Dimensionality reduction using similarity-induced embeddings. IEEE transactions on neural networks and learning systems (2017)
40. Rolfe, J.T., LeCun, Y.: Discriminative recurrent sparse auto-encoders. arXiv preprint arXiv:1301.3775 (2013)
41. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Tech. rep., DTIC Document (1985)
42. Samaria, F.S., Harter, A.C.: Parameterisation of a stochastic model for human face identification. In: Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, pp. 138–142. IEEE (1994)
43. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
44. Song, C., Liu, F., Huang, Y., Wang, L., Tan, T.: Auto-encoder based data clustering. In: Iberoamerican Congress on Pattern Recognition, pp. 117–124. Springer (2013)
45. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
46. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of machine learning research **3**(Dec), 583–617 (2002)
47. Tian, F., Gao, B., Cui, Q., Chen, E., Liu, T.Y.: Learning deep representations for graph clustering. In: AAAI, pp. 1293–1299 (2014)
48. Tsapanos, N., Tefas, A., Nikolaidis, N., Pitas, I.: A distributed framework for trimmed kernel k-means clustering. Pattern recognition **48**(8), 2685–2698 (2015)
49. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning, pp. 1096–1103. ACM (2008)
50. Wang, J., Wang, J., Ke, Q., Zeng, G., Li, S.: Fast approximate k-means via cluster closures. In: Multimedia Data Mining and Analytics, pp. 373–395. Springer (2015)
51. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International Conference on Machine Learning, pp. 478–487 (2016)
52. Xing, E.P., Jordan, M.I., Russell, S.J., Ng, A.Y.: Distance metric learning with application to clustering with side-information. In: Advances in neural information processing systems, pp. 521–528 (2003)

53. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5147–5156 (2016)
54. Zhang, T.: Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Proceedings of the twenty-first international conference on Machine learning, p. 116. ACM (2004)