Project 2: Structure in compressed spaces (Unsupervised Learning). Assignment 2.

Student number: 1802493

***Abstract*—Clustering is one of the most common challenges in machine learning. To find useful patterns in unlabeled data is not an easy task. Many techniques have been developed for this purpose over the last century. This project uses a state-of-art method for clustering, Deep K-Means (DKM), a traditional k-means algorithm tuned simultaneously with an autoencoder used to find a new embedded feature space. This allows us to deal with the problems related to high-dimensional and unstructured data. Three data sets of very different nature were chosen to test this method. The first one is a database of images of handwritten numbers (PenDigits). The second is a data set of sensor outputs registered while performing different human activities (HAR). The third is a database of mushrooms types with their descriptive features (Mushrooms). The results of DKM are compared with those of three other techniques: traditional K-Means (KM), separately tuned linear autoencoder (AE) and separately tuned softmax autoencoder (AESoft). To assess their performance, internal and external evaluation metrics are computed: the mean Silhouette Coefficient (MeanSC) , the Completeness Score (CS), the Homogeneity Score (HS) and the Accuracy (ACC). DKM outperforms the other techniques in ACC, CS and HS for all three data sets.**

I. Introduction

UNSUPERVISED learning can be a difficult task because patterns need to be extracted from the data without having a response variable available for training. Traditional methods for clustering, such as k-means, use distance measures that have some limitations. In some cases, when dealing with image data sets for example (where the features are the raw pixels), using the k-means directly on the original set of features is proven to be ineffective (Zhang et al., 2018 [1]). To deal with this kind of challenges, algorithms may be applied prior to the clustering to determine a new feature space.

This project considers using autoencoders as a feature reduction method to apply in combination with traditional clustering techniques. An autoencoder can be considered as a neural network (NN) that is trained with the original features as both, the input and the output layer, and is composed by encoder/decoder functions in the hidden layers. The encoder generates new latent representation of the data, but with a reduction on dimensionality.

The proposed methodology is based on Deep K-Means algorithm (DKM) presented by (Fard et al., 2018 [2]). This methodology estimates the embedded feature space jointly with the clusters' centers to find the "best" encoder for the purpose of clustering. A comparison with three different methods is done: (a) traditional K-Means over the original feature space (KM), (b) K-Means over embedded features obtained using previously trained autoencoders (AE) and (c) a similar approach to AE, but using a softmax layer in middle NN layer (AESoft).

II. Background

Clustering has been the topic of many research studies in the past. Its origins date back to the first half of the 20th century and were driven by researchers in the social sciences, such as anthropology and psychology.[1] Several types of algorithms have been developed since. From k-means (MacQueen et al., 1967 [4]) to DBSCAN (Ester et al., 1996 [5]), many authors have proposed different techniques to find the best clustering method possible. The truth is that all techniques have some advantages and disadvantages and defining which one is best usually depends on the characteristics of each particular problem.

Many of the traditional clustering techniques present problems when dealing with high-dimensionality (Steinbach et al., 2004 [6]) and feature reduction is not a simple task. Principal Component Analysis (Pearson, 1901 [7]) is a traditional approach for feature reduction that transforms the original features into uncorrelated components that are defined in a way that the first components accumulate most of the variance of the data. New approaches propose the use of neural networks, more specifically autoencoders, to deal with the reduction of dimensionality of the data sets. For example, authors such as (Ji et al., 2017 [8]) and (Peng et al., 2017 [9]) propose using a feature subspace obtained using deep neural networks for clustering. However, those methodologies differ from the one proposed in this project is that the feature space reduction is done separately from the cluster assignment.

Methods such as Deep Embedded Clustering (DEC) (Xie et al., 2016 [10]) and Deep K-Means (DKM) (Fard et al., 2018 [2]), that simultaneously define the feature reduction and the cluster assignment, have shown significant improvement over other state-of-the-art clustering techniques. Fard compared the DKM methodology with other nine methodologies, which include variants of the following: traditional k-Means, k-Means over previously embedded feature space, deep clustering network approach (Yang et al, 2017 [11]) and IDEC (Guo et al., 2017 [12]), which is an improved version of the DEC approach proposed by Xie. In this project, an adaptation of DKM is implemented and tested over three different data sets.

Some extensions of these techniques, such as Variational Deep Embedding (VaDE) (Jiang et al., 2016 [13]) and Gaussian Mixture Variational Autoencoder (GMVAE) (Dilokthanakul et al., 2016 [14]), have been proposed and have also shown good results.

[1]The first approach to clustering was done by anthropologists Driver and Kroeber in 1932[3].

III. Methodology

This section presents the method that will be used for clustering and makes a detailed description of the three data sets that will be used.

A. Clustering Methodology

For this project autoencoders will be used with a traditional clustering algorithm, k-means. A simplified explanation of this technique is presented.

1) Autoencoders: An autoencoder is a type NN where the input is the original list of features (x) and the output is the reconstruction of this list (x'). The intermediate layers are composed by an encoder function and a decoder function. To simplify, we will think about it as a two layers NN, so that $h_1$is the activation function of the encoder and h_2 is the activation function of the decoder. This is represented in the diagram of Figure 1. The encoder is what we will use to reduce the dimensions of the features.

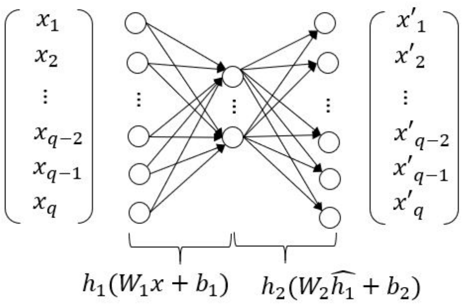

Fig. 1. Autoencoders are NN that have as input the features and as output a reconstruction of them. To do this , the intermediate layers have an encoder and a decoder. By defining he parameters of the encoder function ($W_1$and b_1) the new lower-dimensionality features can be computed.

To determine the values of the parameters, a minimization is done over the squared errors. This minimization problem is presented in Equation (1).

$$\min_{W,b} \sum_i (x_i - x'_i)^2 \tag{1}$$

Where $W = [W_1, W_2, ..., W_l]$ and $b = [b_1, b_2, ..., b_l]$ are the parameters of the autoencoder NN with l layers.

2) K-Means: We use k-means to find the optimal assignment of instances to k clusters. The objective is to find k centroids and assign the instances to the k clusters, so that the sum of the distances of each point to the centroid of the cluster where that point is assigned is minimized. This is presented in Equation (2), where $c_j$is the centroid of cluster $S_j$and x_i is the i-th instance of the cluster S_j.

$$\min_{C} \sum_j \sum_{x_i \in S_j} (x_i - c_j)^2 \tag{2}$$

Where $C = [c_1, c_2, ..., c_k]$ is the set of clusters centroids for k clusters.

3) Autoencoders+K-Means: When using autoencoders to reduce the dimensionality it is important that both optimization problems (1) and (2) are solved simultaneously. If (1) is solved separately some parameters (W and b) will be found, but nothing can assure that those parameters are the ones that will reduce the features' dimensions adequately for our goal, which is to find the clusters.

In order to do the clustering, we need to solve both optimization problems together, as stated in Equation (3).

$$\min_{W,b,C} \sum_i (x_i - x'_i)^2 + \sum_j \sum_{x_i \in S_j} (x_i - c_j)^2 \tag{3}$$

Based on (Fard et al, 2018 [2]) methodology, two elements were added to the optimization problem in (3). First, λ is included as a regularization parameter to trade-off between good feature representation and good features for clustering. The bigger the λ the more important is the part of (3) that represents the loss function for clustering (2). Second, we include a softmax function in the cluster assignment part of (3) to transform it into a continuous generalization. The resulting equation is (4).

$$\min_{W,b,C} \sum_i (x_i - x'_i)^2 + \lambda \sum_j \sum_{x_i \in S_j} (x_i - c_j)^2 G_j(x_i, c_j) \tag{4}$$

with

$$G_j(x_{i,}, c_j) = \frac{e^{-\alpha \sum_j \sum_{x_i \in S_j} (x_i - c_j)^2}}{\sum_j e^{-\alpha \sum_j \sum_{x_i \in S_j} (x_i - c_j)^2}}$$

The function $G_j(x_{i,}, c_j)$ takes values between 0 and 1. The higher the parameter α, the hardest is the cluster assignment.

B. Evaluation Methodology

For evaluation, two forms of metrics are used.

First, an internal evaluation of the clusters' assignment using the mean Silhouette Coefficient (SC) is done. The SC is computed for every observation i. It contrasts the average distance of the instances within the same cluster with the average distance to instances in other clusters. Intuition is that an instance has a high SC if there is a high similarity with other instances of the same cluster and low similarity with instances in other clusters. It takes values from -1 to 1, where a high value indicates that the object is well matched to its own cluster. The mean SC formula is presented in Equation 5, where a_i is the average distance between point i and all other data within the same cluster and b_i is the smallest average distance of i to all other points in other clusters. N is the total number of instances in the sample.

$$MeanSC = \sum_{i=1}^{N} SC_i = \sum_{i=1}^{N} \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{5}$$

The second form of evaluation will be external, comparing the clusters obtained with the labels. In order to do this, three different metrics will be computed: the Completeness Score (CS), the Homogeneity Score (HS) and the Cluster

Accuracy (ACC). The metrics CS and HS are both necessary when evaluating clusters solutions because they run in opposition: Increasing the homogeneity of a clustering solution often results in decreasing its completeness. As explained by (Rosenberg et al., 2007 [15]): “A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class. A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.” The ACC is computed so that the results obtained are comparable to those presented by other authors who used deep clustering on very similar data sets.

$$CS = 1 - \frac{H(K \mid C)}{H(K)} \tag{6}$$

$$HS = 1 - \frac{H(C \mid K)}{H(C)} \tag{7}$$

With:

$$H(C \mid K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{c,k}}{n} log(\frac{n_{c,k}}{n_k})$$

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} log(\frac{n_c}{n_k})$$

Where $H(C \mid K)$ is the conditional entropy of the classes given the cluster assignments and $H(C)$ is the entropy of the classes, with n the total number of samples, n_c and n_k the number of samples respectively belonging to class c and cluster k, and finally $n_{c,k}$ the number of samples from class c assigned to cluster k.

$$ACC = \max_{m} \frac{\sum_{i=1}^{n} 1\{l_i = m(c_i)\}}{n} \tag{8}$$

Where l_i is the ground-truth label, c_i is the cluster assignment produced by the algorithm, and m ranges over all possible one-to-one mappings between clusters and labels.

Both, internal and external evaluation are important. On the one hand, internal evaluation will give insights about the construction of the clusters (how cohesive are the clusters and how separate they are from each other) and the external evaluation shows if the clusters are good at predicting what we wanted to predict.

To evaluate the quality of the results, the evaluation metrics for DKM will be compared to the results obtained for KM, AE and AESoft.

C. Data sets

Three very different data sets are used to evaluate the proposed method. Although these data sets are used for unsupervised learning, all of them are labeled. This label is not used in the training of the models, but is later used in evaluating the completeness metric. Because of their different nature, the analysis and pre-processing of the data is also different in each case. Table I shows a summary of the final data sets used in the cluster analysis. The total number of observations (# of Obs.), Features (# of Feat.) and Classes (# of Class.) is shown. Also, details on these characteristics are presented in the last three rows: the presence of missing values, the type and the scale of the features. These characteristics indicate whether it is necessary to make some pre-processing of the data (like imputations or re-scaling) before using it in the clustering training. A detailed description of every data set is presented next.

TABLE I
DATA SETS SUMMARY

	PenDigits	**HAR**	**Mushrooms**
# of Obs.	1,797	10,299	8,124
# of Feat.	64	560	75
# of Clas.	10	6	2
Missing Val.	No	No	Yes
Type of Feat.	Continuous	Continuous	Binary
Scales of Feat.	Homogen.	Slightly Heterogen.	Homogen.

1) PenDigits: The original Pen-Based Recognition of Handwritten Digits Data Set (Alimoglu, 1996[16]) contains about 11,000 instances and 16 features. Each instance represents an image of a handwritten object. The instances were collected from 44 different writers who were asked to write 250 digits from 0 to 9. The features are integers that represent coordinate information. Each number is associated with a coordinate (x,y). The initial image has 250,000 (500x500) possible coordinate points, but after processing, the data, each image can be represented as an 8x8 bitmap (64 pixels). The processing of the data is shown in Figure 2.

For this project we only use the processed data where every image is represented by 64 pixels. These pixels are the features we use in the clustering algorithms. All features in this data set are continuous and have the same range, as they hold a numerical value from 0 to 16, representing the level of darkness of the gray color in the pixel.

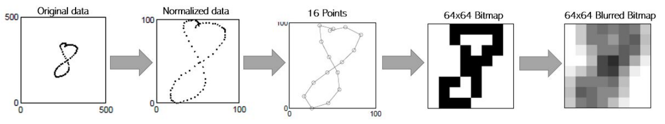

Fig. 2. Processing of original data set. A normalization is done to reduce dimensions from 500x500 to 100x100. Resampling is applied so that each image can be represented by only 16 strategic points. These points are the transformed into a 64x64 bitmap using Bresenham’s Algorithm and then the image is blurred to deal with the small variations on pen’s coordinates that could distort the overall image. More detail on how this transformation from points to bitmap is done can be found in (Alimoglu et al., 1996 [16]).

By processing the data set, a simpler representation of the images is obtained. This is convenient to reduce the processing time when training the clustering model. However, the risk of the machine confusing the digits increases. In Figure 3 the plotting of 10 instances of the data set, each representing one digit from zero to nine, is presented. At simple sight, in this example some digits could be easily mistaken. For example, digits 3 and 5 have similar bitmap representations in this case.

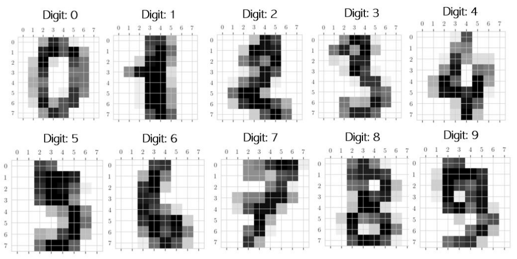

Fig. 3. Plots of a sample of 10 instances of the final database. Due to the low resolution, some digits could easily be confused with one another. In the example, the plots for 3 and 5 look very similar.

For simplicity, in this project a sample of only 1,797 instances from the 11,000 available is used.[2] Also, features are normalized to take values between 0 and 1, by dividing each value by the variable range (16). A variable description of the final data set can be found in Table II. The distribution of the classes (digits) in the final data set is presented in Table III.

TABLE II
PENDIGITS VARIABLES DESCRIPTION

Variable	Description
pixel_i	Pixel in position i of the 8x8 bitmap. After normalization, it takes float values from 0 to 1 which indicate the scale of gray of the pixel (being 0 the lightest and 1 the darkest). i=0,1,..,63.
label	The number which the image represent. It could be any number from 0 to 9.

TABLE III
PENDIGITS CLASS DISTRIBUTION

Class	0	1	2	3	4
Perc.	9.9%	10.1%	9.8%	10.2%	10.1%
Class	5	6	7	8	9
Perc.	10.1%	10.1%	10.0%	9.7%	10.0%

2) HAR: The Human Activity Recognition Using Smartphones Data Set[3] contains the sensor outputs from smartphones of people performing different activities. The activities include: laying, standing, sitting, walking, walking upstairs and walking downstairs. The data set contains the readings of two motion sensors for each type of device: accelerometer and gyroscope. The data set consists of 560 variables obtained from the accelerometer signals in the time and frequency domain (e.g. mean, standard deviation, signal magnitude area, entropy, signal-pair correlation, etc.) and 10,299 instances. The features are named according to what they represent, using a nomenclature that specifies the following characteristics: transformation applied - acceleration origin - acceleration type - statistic - axis. For example, the variable f-Body-Acc-Mean-X is the mean of the body acceleration in axis X with a Fourier transformation applied. Table IV shows the characteristics of the features of the data set.

TABLE IV
HAR VARIABLES CHARACTERISTICS DESCRIPTION

Variable Characteristic	Description
Transformation	Transformation applied to the variable. t=no transformation, f=fast Fourier transformation.
Acceleration Origin	Origin of the acceleration signal. Body=body signal, Gravity=gravity signal.
Acceleration Type	Type of acceleration. Acc=linear, Gyro=angular, AccJerk=jerk signal derived from linear acceleration, Gyro=jerk signal derived from angular acceleration.
Statistic	Summary statistic computed.[4] mean=mean value, std=standard deviation, mad=median absolute deviation, max=maximum, min=minimum, sma=signal magnitude area, energy=energy measure, iqr=interquartile range, entropy=signal entropy, arCoeff=autoregresssion coefficient, correlation= correlation between 2 signals, maxInds=Index of the largest magnitude. meanFreq=Weighted avg. of the frequency components, skewness=skewness of the frequency domain signal, kurtosis= kurtosis of the frequency domain signal, bandsEnergy=Energy of a frequency interval , angle=angle between two vectors.
Axis	Axis of the variable. X=axis x, Y=axis y, Z=axis Z, Mag=Magnitude of the 3-dimensional signals calculated using the Euclidian norm.

The right image of Figure 4 shows is a boxplot of the variable t-Body-Acc-Mag-mean. It is clear that differences can be spotted between stationary activities and moving activities. Just by taking this variable and setting a threshold of -0.7, we could predict if the activity is stationary or not depending if the value is above or under this threshold. Other variables can be useful to differentiate between other activities. For example, in the left image of Figure 4 the boxplot for the variable angle-Y-gravity-Mean is presented. This variable represents the angle between the gravity acceleration vector and axis y and it is useful to differentiate the activity Laying from the rest.

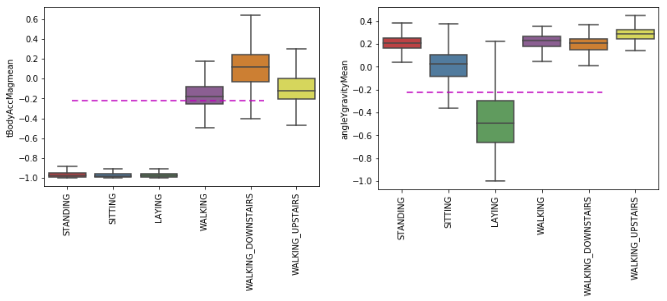

Fig. 4. t-Body-Acc-Mag-mean distribution per activity. The mean linear body acceleration is lower for stationary activities then for moving activities. Also, the distribution for moving activities

[2]This sample is available as the “digits” data set in the scikit learn library.
[3](Anguita et al., 2013 [17])

[4]The description of the statistics used in this data set are explained in (Figo et al., 2010 [18]).

The distribution of the classes (activities) in the final data set is presented in Table V.

TABLE V
HAR CLASS DISTRIBUTION

Class	lay	stand	sit	walk	upstairs	downstairs
Perc.	18.9%	18.5%	17.3%	16.7%	15.0%	13.7%

3) Mushrooms: The Mushroom Data Set[5] contains the description of samples corresponding to 23 species of mushrooms from the Agaricus and Lepiota Family. It has 8,124 instances and 23 variables (22 features + 1 label). The label is a binary variable that indicates if the mushroom is edible or not. Since the edibility of the mushrooms can usually be determined by its physical characteristics, we use deep clustering to see if these two groups can be identified by only looking at the features. A diagram of the mushrooms' different parts is presented in Figure 5.

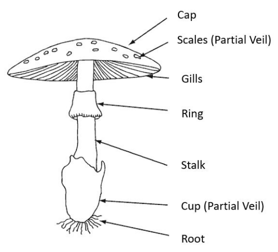

Fig. 5. The parts of the mushrooms that are described by the features of the data set are shown in this diagram.

All the variables of the original data set are categorical.[6] Table VI has a description of these features and their categories. For the clustering, these initial 22 features are transformed into 75 dummies variables, one for each feature-category pair.

When analyzing the frequencies of the categories for each feature[7], we can see that for two of the variables more than 95% of the instances are concentrated in only one category: veil_type (100% belongs to category "p") and veil_color (98% belongs to category "w"). All the mushrooms in the data set have partial veil and most of them are white. These variables are of no use for discriminating among clusters and are removed prior to the training. Also, the feature stalk_shape has 31% of its values missing. These observations with missing values were removed when training the clusters. For this reason the final data set is 5,644 observations.

The distribution of the classes in the final data set is presented in Figure VII.

[5](Schlimmer et al., 1987 [19])

[6]Only ring number could be considered continuous, but it is coded as categorical an is used as such.

[7]The distribution of categories for each variable can be found in the Jupyter notebook attached.

TABLE VI
MUSHROOMS VARIABLES DESCRIPTION

Variable	**Description**
label	Binary variable that indicates edibility. poisonous=p, edible=e.
cap_shape	Shape of the cap. bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s.
cap_surface	Surface of the cap texture. fibrous=f, grooves=g, scaly=y, smooth=s.
cap_color	Color of the cap. brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y .
bruises	Binary variable that indicates the presence of bruises. bruises=t, no=f.
odor	Odor of the mushroom. almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s.
gill_attachment	Type of attachment of gills. attached=a, descending=d, free=f, notched=n
gill_spacing	Spacing of the gills. close=c, crowded=w, distant=d
gill_size	Size of the gills. broad=b, narrow=n.
gill_color	Color of the gills. black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y.
stalk_shape	Shape of the stalk. enlarging=e, tapering=t.
stalk_root	Type of root of the stalk. bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?.
stalk_surface_above	Stalk surface above the ring. fibrous=f, scaly=y, silky=k, smooth=s.
stalk_surface_below	Stalk surface below the ring. fibrous=f, scaly=y, silky=k, smooth=s.
stalk_color_above	Stalk color above the ring. brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y.
stalk_color_below	Stalk color below the ring. brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
veil_type	Veil type. partial=p, universal=u
veil_color	Veil color. brown=n, orange=o, white=w, yellow=y
ring_number	Number of rings. none=n, one=o, two=t
ring_type	Type of rings. cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
spore_print_color	Spore print color. black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
population	Type of population. abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
habitat	Habitat. grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

TABLE VII
MUSHROOMS CLASS DISTRIBUTION

Class	poisonous	edible
Perc.	38.2%	61.8%

IV. EXPERIMENTS

A. *Experiments Setup*

The DKM approach is tested in the three data sets previously described. Also, three additional methods are processed for comparison: KM, AE and AESoft.

The number of clusters is defined according to the number of classes of the labels for each data set. Since k-means is sensitive to the initial centroids, for all four techniques 10 random restarts are performed and the run with the best objective function (accuracy) is then selected. Ten random seeds are defined and used across the different techniques.

The encoders used in DKM, AE and AESoft are fully-connected multilayer perceptron with dimension *f-500-500-2000-k,* where f is the number of features of the database and k is the number of clusters. This configuration is the same they used in our reference studies, (Xie et al., 2016 [10]) and (Fard et al., 2018 [2]). All layers, expect the last encoder layer and the output layer, use a ReLU activation function. The output layer is the default linear activation function for DKM and the sigmoid activation function for AE and AESoft[8]. The last encoder layer (middle layer of the autoencoder) uses the default linear activation function for DKM and AE, and softmax function for AESoft.

AE and AESoft are trained with 50 epochs and a batch size of 256 each. DKM is pretrained with 50 epochs and the final parameters are obtained by fine tuning, using 100 epochs[9]. For DKM, the loss function (Equation (4)) is solved by using the Adam Optimizer. For AE and AESoft, we use AdaDelta Optimizer.

For DKM, the parameters λ and α are set to 0.1 and 1000 respectively, both were taken from the experimental setup used by (Fard et al., 2018 [2]) for implementing the deep clustering with pretraining variant of their experiment. This variant of DKM was the one that presented the highest performance (accuracy and normalized mutual information scores) of the ten different techniques tested by Fard.

B. *Experiments Results*

The results of running the different clustering techniques are presented in Table VIII. The performance scores presented in section 3.2 are computed. Each algorithm is ran 10 times and the iteration with the highest accuracy is presented. The highest ACC among the four techniques is highlighted using bold font. For all the external evaluation performance metrics, DKM shows the highest scores. However, the best value for the internal evaluation score used (SC) varies depending on the data set.

TABLE VIII
RESULTS SUMMARY

		MeanSC	CS	HS	ACC
PenDigits	KM	0.18	0.75	0.74	79.4%
	AE	0.14	0.62	0.62	71.3%
	AESoft	0.05	0.39	0.39	50.5%
	DKM	**0.19**	**0.77**	**0.76**	**80.7%**
HAR	KM	0.13	0.60	0.58	59.9%
	AE	0.11	0.65	0.62	66.3%
	AESoft	**0.21**	0.68	0.56	56.7%
	DKM	0.08	**0.75**	**0.76**	**84.4%**
Mushrooms	KM	**0.27**	0.54	0.44	85.4%
	AE	0.17	0.49	0.51	84.1%
	AESoft	0.19	0.19	0.20	75.9%
	DKM	0.17	**0.62**	**0.64**	**92.6%**

Figures 6, 7 and 8 show the confusion matrices for PenDigits, HAR and Mushrooms respectively. These matrices are useful to establish the classes that the algorithms have a harder time identifying. To facilitate reading, a color scale has been defined, so that each cell of the matrix is painted according to the magnitude of the number of observations in the cell.

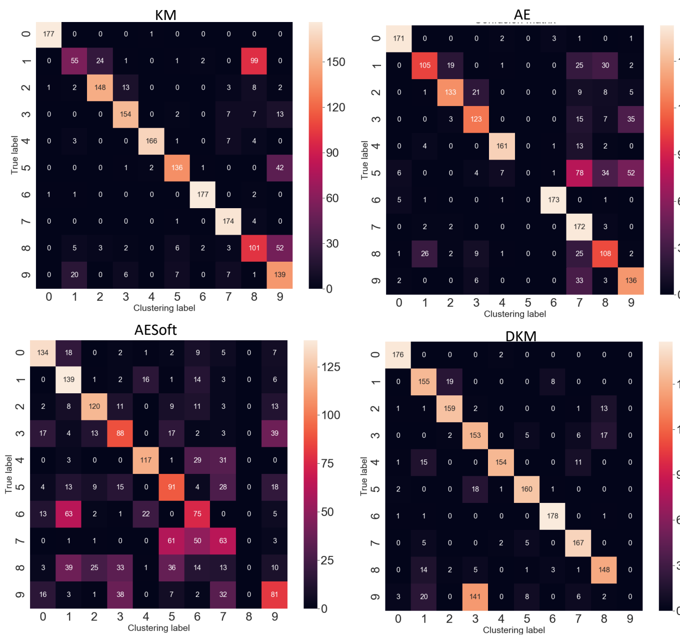

Fig. 6. Confusion Matrix for PenDigits. The two techniques with higher accuracy are DKM and KM. For digits 0 to 9, DKM is clearly the algorithm that identifies the classes better. However, this technique fails to identify digit 9, it mistakes it for a 3 in most of the cases.

[8]We used as reference the autoencoder built in https://blog.keras.io/building-autoencoders-in-keras.html. Also, in the three data sets, all features were normalized to take values between 0 and 1, so it makes sense that the output takes values in this range.

[9]max_n $\times$ n_finetuning_epochs

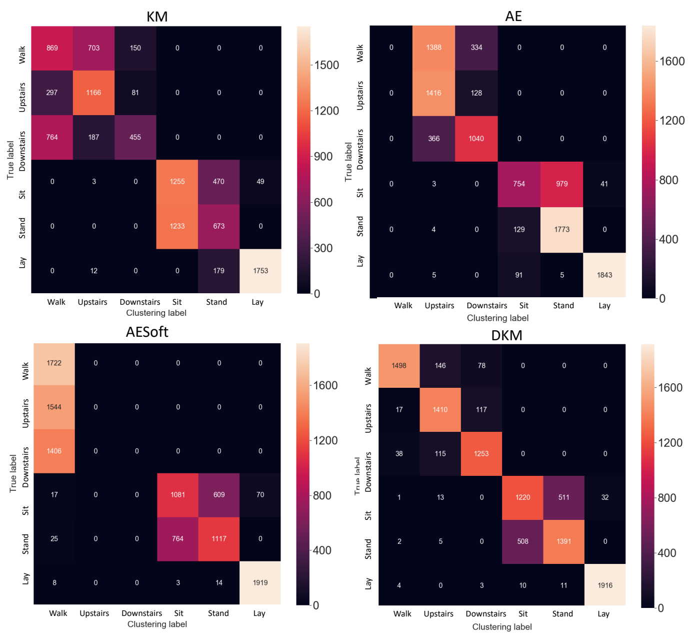

Fig. 7. Confusion Matrix for HAR. There is a significant improvement in using DKM over the other three techniques. With DKM all of the activities have a very high percentage of correct predictions, between 73% (for standing) and 99% (for laying). The two stationary activities, sitting and standing are the most commonly mistaken with each other.

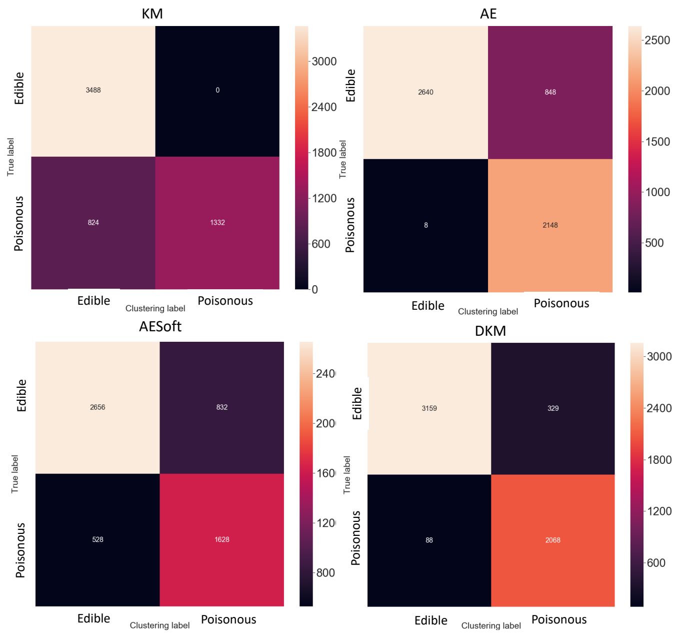

Fig. 8. Confusion Matrix for Mushrooms. The algorithm with the best performance in identifying the classes is DKM, 93% accuracy. It recognizes a poisonous mushrooms correctly 96% of the times (91% in recognizing the edible mushrooms correctly). KM, that has the second highest ACC and the highest MeanSC, labels correctly all the edible mushrooms but fails in labeling 38% of the poisonous mushrooms.

V. Discussion

In this section, a discussion on the performance scores obtained is presented. First, the external evaluation metrics are analyzed, comparing the results between the four techniques and also commenting on the results obtained by other authors who have implemented similar methodologies. Then, an analysis over the internal evaluation is done, by examining the MeanSC computed for each data set.

The DKM methodology has the highest ACC scores in all three data sets. In other words, this algorithm is the most effective of the four in successfully identifying the classes. However, the improvement of ACC from the next best technique varies from one data set to the other. The largest improvement by using DKM is obtained in the HAR data set, which ACC is 18.1% higher than the next best technique, AE. The DKM on the Mushrooms and PenDigits data sets show an improvement of 7.20% and 1.20% respectively over the next best method, KM. HAR is the data set that seems to benefit the most from the construction of a new embedded feature space because both, AE and DKM, show better performance scores than KM, which does the clustering over the original feature space. We attribute this effect to the fact that HAR is considerably largest in terms of number of features than the other two data sets. This phenomenon has been addressed as “the curse of high dimensionality” (Bellman, 2015[20]). According to (Beyer et al., 1999[21]), a large number of features causes problems when using a distance or similarity measure to find the clusters, because in high dimensional spaces, distances between points become relatively uniform and the notion of the nearest neighbor of a point becomes meaningless.

The CS and HS also reach their highest values when using DKM for clustering. Considering that these scores move in a 0 to 1 range, obtaining values above 0.7 (for PenDigits and HAR) is a good result. A high homogeneity indicates that each cluster contains mostly members of a single class, while a high completeness indicates that all members of a given class are assigned to the same cluster. As stated in section 3.2, these metrics can run in opposition, so it is important that both are maximized simultaneously to obtain a good clustering solution. This is the case in all three data sets, because for every one of them HS and CS are balanced. The Mushrooms data set has the lower CS and HS of the three and this could seem contradictory with the fact that this data set is also the one with the highest ACC score. However, it is necessary to point out that ACC score depends on the number of classes to predict and the balance of those classes, and for this reason it is not right to compare ACC across different data sets. For example, in Mushrooms if we assign the total of observations to the predominant class, “edible”, we would still obtain an AAC of 61.8%, while if we did the same with PenDigits by assigning all of the observations to class “3”, we would only obtain a 10.2% accuracy. So, the “base ACC” differs from one data set to the other. CS and HS, however, are independent of the number of classes (Rosenberg et al., 2007 [15]) so we can compare the scores across data sets. Thus, it is correct to say that the clusters for Mushrooms are less homogeneous and complete that the ones obtained for PenDigits and HAR.

While the external evaluation scores show that DKM outperforms the other techniques, the internal evaluation score, MeanSC, is not that clear in defining which technique is better. We observe that KM shows a good MeanSC for all three data sets (if not the best score, the second best). This can be explained on the reason that both, KM and the MeanSC are obtained over the original feature space. In other words, a high SC depends on the distances (or similarities) between observations in the original feature space and when KM is applied to these features directly, it defines clusters by minimizing those distances[10]. This phenomenon can be observed in the Mushrooms data set, where KM shows the highest MeanSC, with a value of 0.27 (0.08 points higher than the next best MeanSC, obtained by AESoft). For the PenDigits data set, the MeanSC obtained by KM and DKM are alike, which indicates that the embedded feature space obtained with DKM can be successfully used to identify instances with similar characteristics in the original features. Lastly, for HAR the technique that shows the highest MeanSC is the AESoft. As mentioned when explaining the "curse of dimensionality", in high dimensional spaces the notion of the nearest neighbor of a point becomes meaningless, so it is reasonable that for this data set, even if KM works on directly on the original feature space, an encoded feature space could give better results. It is interesting that the highest MeanSC is found for AESoft, which presents the lowest ACC. This shows that the most successful algorithms at predicting the labels (high accuracy) are not necessarily the ones that produce the most "dense" clusters in terms of the original features.

VI. Conclusion

This project analyzes the impact of using autoencoders to deal with the high dimensionality in clustering problems. In particular, it consists of an implementation of the DKM methodology for clustering first proposed by (Fard et al., 2018[2]). Various methodologies that use neural networks to deal with this kind of problems have been presented in the background section. Among them, the DKM methodology was chosen for this project because Fard showed that, when compared with other techniques over the same data sets, DKM outperforms in most of the performance metrics. In this project, a comparison with other techniques (that included KM, AE and AESoft) was also done, and the results are consistent with Fard in showing that this technique has better results in predicting the labels than the other three. This is specially true for high dimensionality data sets, such as HAR, in which the benefit of using autoencoders is more evident than in other cases because it helps dealing with the challenges associated with "the curse of high dimensionality".

When doing an internal evaluation of the clusters obtained with different techniques, we conclude that the best techniques for predicting labels are not necessarily the ones that provide the most cohesive clusters in terms of the original features. For example, applying K-Means directly on the original data can generate clusters with better MeanSC but worse accuracy in the predictions.

Next steps could include testing other techniques that have shown good results in clustering and comparing them to DKM. Some of them could be, using Convolutional Neural Networks for the autoencoders used on image data sets, because evidence shows that this type of network performs better for this kind of data, or testing other state-of-the-arts approaches like variational autoencoders (VAE), like the ones used by (Jiang et al., 2016 [13]) and (Dilokthanakul et al., 2016 [14]), which were not tested in this project.

[10]Obtaining local minima.

References

[1] D. Zhang, Y. Sun, B. Eriksson, and L. Balzano, "Deep unsupervised clustering using mixture of autoencoders," *ArXiv*, 2018.

[2] M. M. Fard, T. Thonet, and E. Gaussier, "Deep k-means: Jointly clustering with k-means and learning representations," 2018.

[3] H. Driver and A. Kroeber, "Quantitative expression of cultural relationships," *University of California Publications in America Archaeology and Ethnology*, 1932.

[4] J. M. et al., "Some methods for classification and analysis of multivariate observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1996.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[6] M. Steinbach, L. Ertoz, and V. Kumar, *The challenges of Clustering High Dimensional Data*. Springer, Berlin, Heidelberg, 2004.

[7] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, 1901.

[8] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," 2017.

[9] X. Peng, J. Feng, S. Xiao, J. Lu, Z. Yi, and S. Yan, "Deep sparse subspace clustering," 2017.

[10] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," *International Conference on Machine Learning*, 2016.

[11] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, pp. 3861–3870. [Online]. Available: http://dl.acm.org/citation.cfm?id=3305890.3306080

[12] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation." in *IJCAI*, 2017, pp. 1753–1759.

[13] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: A generative approach to clustering," *CoRR*, vol. abs/1611.05148, 2016. [Online]. Available: http://arxiv.org/abs/1611.05148

[14] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *CoRR*, vol. abs/1611.02648, 2016. [Online]. Available: http://arxiv.org/abs/1611.02648

[15] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

[16] F. Alimoglu, "Combining multiple classifiers for pen-based handwritten digit recognition," Master's thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University, 1996.

[17] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Energy efficient smartphone-based activity recognition using fixed-point arithmetic," vol. 19, no. 9, pp. 1295–1314, may 2013.

[18] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. P. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010. [Online]. Available: https://doi.org/10.1007/s00779-010-0293-9

[19] J. Schlimmer, "Concept acquisition through representational adjustment," Ph.D. dissertation, Department of Information and Computer Science, University of California, 1987, as reported by [1].

[20] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015, vol. 2045.

[21] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *Proceedings of the 7th International Conference on Database Theory*, ser. ICDT '99. London, UK, UK: Springer-Verlag, 1999, pp. 217–235. [Online]. Available: http://dl.acm.org/citation.cfm?id=645503.656271