IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 30, 2020, accepted August 12, 2020, date of publication August 19, 2020, date of current version September 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017884

Non-Parametric Clustering Using Deep Neural Networks

CHRISTOS AVGERINOS, VASSILIOS SOLACHIDIS, NICHOLAS VRETOS, (Member, IEEE), AND PETROS DARAS, (Senior Member, IEEE)

Visual Computing Laboratory, Centre for Research and Technology Hellas-Information Technologies Institute, 57001 Thessaloniki, Greece

Corresponding author: Christos Avgerinos (avgerinos@iti.gr)

This work was supported by the European Commission, digital iNtegrAteD system for the socIal support of migraNts and refugEes (NADINE), under Contract H2020-822601.

ABSTRACT In this paper, a novel algorithm for non-parametric image clustering, is proposed. Non-parametric clustering methods operate by considering the number of clusters unknown as opposed to parametric clustering, where the number of clusters is known a priori. In the present work, a deep neural network is trained, in order to decide whether an arbitrary sized group of elements can be considered as a unique cluster or it consists of more than one clusters. Using this trained neural network as clustering criterion, an iterative algorithm is built, able to cluster any given dataset. Evaluation of the proposed method on several public datasets shows that the proposed method is either on par or outperforms state-of-the-art methods even when compared to parametric image clustering methods. The proposed method is additionally able to correctly cluster input samples from a completely different dataset than the one it has been trained on, as well as data coming from different modalities. Results on cross-dataset clustering show evidence of the generalization potential of the proposed method.

INDEX TERMS Cross-dataset, high dimensional clustering, machine learning, non-parametric.

I. INTRODUCTION

One of the fundamental challenges in computer science is the task of grouping data into categories in an unsupervised manner. An abundance of methods and algorithms on clustering has been proposed thus far in many scientific fields in different areas, such as mathematics, statistics and computer science exploiting traditional analytical methodologies, machine learning and various techniques based on neural networks. Data clustering is used in an excessive number of applications ranging from text mining, video analysis and medical imaging to social science and humanities. The ability to group similar data and distinguish dissimilar ones is essential in broadening and expanding the clustering research field and associated applications.

In particular, extracting underlying connections between high-dimensional data is tackled during the last years with a plethora of different approaches with methods that can be highly distinctive. Subspace clustering algorithms like in [1], [2] and [3] try to extract clusters from multiple and possibly overlapping high-dimensional subspaces. In [4], an algorithm based on sparse representations of the data is presented. This work states that a sparse representation of a data point is essentially a linear or affine combination of data points that belong to the same subspace, a property defined as self-expressiveness that is therefore used to group data together. In Deep Subspace Clustering networks (DSC) [5], the authors presented the idea of blending traditional clustering techniques with modern machine learning ones. They introduced a differentiable, non-linear layer that tries to mimic the self-expressiveness property by learning pairwise affinities between data through standard back-propagation. The idea of employing machine learning mechanisms in support of known clustering algorithms though is not new; Song *et al.* [6] confronted the high dimensionality challenge by using autoencoders rather than typical dimensionality reduction methods such as PCA [7]. Moreover, instead of applying a standard clustering algorithm such as k-means on the encoded data, they proceeded with creating a new objective function, able to fuse loss properties of both the auto encoder and the k-means input and output. In [8] and [9], k-means is combined with neural networks in order to segment medical images and detect brain and kidney abnormalities, respectively. Although methods based on established clustering algorithms excel in ease of use, data adaptation and scalability, the requirement of the number of clusters

The associate editor coordinating the review of this manuscript and approving it for publication was Hong-Mei Zhang.

parameterization is a major restriction on using them in real life applications. Tian *et al.* [10] proposed a method based on [4], where, after acquiring a weight matrix through a kernel method, they used a modified spectral clustering algorithm based on autoencoders. Other approaches have attempted to cluster multimodal data. Abavisani and Patel [11] examined different multimodal fusing techniques and proposed a method for fusing affinities across data modalities based on DSC. An additional clustering approach based on popular clustering algorithms is called Robust Continuous Clustering (RCC) [12], which achieves clustering and dimensionality reduction jointly by optimizing a continuous objective function that uses mutual k-nearest neighbors (m-kNN) information [13].

Density-based clustering algorithms such as DBSCAN [14] have also been proposed in the literature. The main advantage of DBSCAN is the fact that it does not require any parameterization regarding the expected cluster number. DBSCAN is robust to outliers, however it is rather ineffective when applied on high-dimensionality data, especially when the density of each data group is unknown. Other density-based clustering works have been presented, such as [15], which is, however, parametric with respect to the neighborhood size and specialized in handling datasets with various data distribution patterns. Spectral Clustering methods have also been proposed, as in [16], where the pairwise constraints of data points are responsible for generating dynamically adaptive neighborhoods of data points, while preserving a low algorithm complexity. Furthermore, the affinity propagation algorithm [17] and its extensions [18]–[21] are methods that do not make use of the number of clusters as input (non-parametric). Affinity propagation methods initially consider all items as potential centers and then proceed by letting the initial centers exchange messages carrying information on how they should merge. These methods are robust to outliers, however, their greedy strategy results in a high computational complexity of $O(F^2\ G)$, F being the total number of items and G the number of algorithm iterations [22]. Finally, an additional non-parametric method is proposed in [23], in which, a general fuzzy min-max (GFMM) neural network is employed in order to fuse classification and clustering, in a simple yet powerful learning process.

The current work proposes a novel clustering method able to cluster data coming from various classification databases without prior knowledge of the exact number of clusters (non-parametric). To achieve this, a deep neural network is trained in order to identify if the contents of an arbitrary sized group of data belong to the same cluster or not. Therefore, the developed neural network is employed as a clustering criterion by an iterative algorithm in order to cluster any given dataset. Furthermore, the present work is also tested for applications that the data, which are to be clustered, have never been encountered before (cross-dataset clustering). The proposed method is able to correctly cluster input samples from a completely different dataset than the one it has been trained on, as well as data coming from different modalities.

The remainder of this paper is organized as follows: Section II is a brief presentation of relative works on clustering. Section III details the proposed non-parametric clustering method, whereas Section IV describes how data are processed and organized to be ingested to the deep neural network. Section V analyzes the results of the performed experiments, and Section V-F concludes the paper.

II. RELATED WORK

More recent approaches than the ones already described in Section I, have integrated Convolutional Neural Networks (CNN) into clustering. For example, the Joint Unsupervised Learning of Deep Representations and Image Clusters (JULE) [24] method applies agglomerative image clustering, while learning image representations at the same time and yields excellent results on most datasets. JULE also achieves good results in cross-dataset clustering. Additionally, in [25], a k-means clustering in conjunction with classification in an alternating approach that produces soft labels is proposed. This approach highlights the strong relationship between data clustering and classification and how convolutional networks have been able to play a leading role in both fields. Dizaji et al proposed in [26] a lightweight clustering algorithm that projects data into a subspace and then utilizes a stacked multi-layered convolutional autoencoder with a softmax on top to predict clusters. All the aforementioned methods require a known number of clusters to operate. For instance, in [26] the number of clusters is given a priori, whereas in [24] and [25] the number of clusters is estimated by applying a clustering algorithm such as DBSCAN [14] or t-SNE [27] before proceeding to their actual method. The proposed method demonstrates the ability to achieve similar or superior results without knowing the number of desired clusters in advance.

A different approach is introduced in Deep Embedding Clustering (DEC) [28], where the authors firstly project data into a space of smaller dimension by employing a non-linear mapping function. DEC is learning (in a simultaneous manner) the cluster centers by minimizing the Kullback–Leibler (KL) divergence between the distribution of the items and an auxiliary target distribution. In [29], the authors propose a feature extraction method using deep convolutional neural networks trained in distinct faces from other identities as well as a new cluster-merging algorithm that measures similarity, based on local density. Another approach based on convolutional networks is the one in [30] where the clustering problem is approached by training a deep autoencoder to initially extract features. Then, a density-based algorithm is applied in order to calculate the total number of clusters. Both methods achieve good clustering results without prior knowledge of the number of clusters, though, on a restricted number of datasets. The proposed method is extensively tested against multiple and diverse datasets, yielding comparable results with parametric methods, or even better results when compared with non-parametric ones.

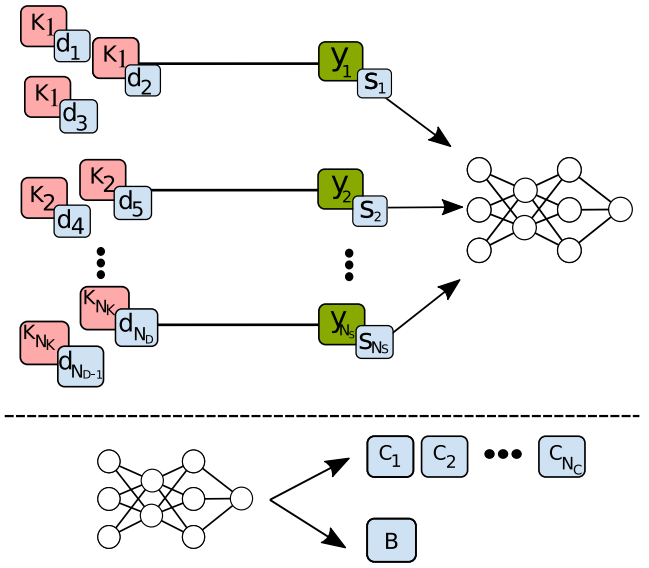

FIGURE 1. Overview of the proposed method. Items of D are organized in subsets based on cluster labels $K_1, K_2, \ldots, K_{N_K}$ (red). Each subset is accompanied with a binary label $y_1, y_2, \ldots, y_{N_S}$ (green), indicating whether the subset can form a single cluster or not. The subsets are first forwarded through a deep neural network, the Evaluation Network (EN), training a clustering criterion. This criterion is employed by the Clustering Process (CP) to create clusters $C_1, C_2, \ldots, C_{N_C}$ and a list of unclustered items B, which will be fed to the network again.

III. PROPOSED METHOD

In this Section the proposed clustering method and the overall framework are described in detail.

A. OVERVIEW

Let D be a set of items $d_i, i = 1, 2, \ldots, N_D$, divided in N_K mutually exclusive partitions $K_l, l = 1, 2, \ldots N_K$,namely $\bigcup_{l=1}^{N_K} K_l = D$ and $K_l \cap K_m = \emptyset, \forall\ 1 \leq l, m \leq N_K$ with $l \neq m$. Let also $S = \{s_1, s_2, \ldots, s_{N_s}\}$ be a random subset of D ($S \subset D$). The elements of S can either belong to the same cluster or not.

The proposed method consists of two components, the Evaluation Network (EN) and the Clustering Process (CP). EN is a binary classification network, trained to distinguish if an input sample-set contains items coming from the same ground truth class and therefore can be recognized as a cluster or not. CP is an iterative procedure that uses EN as a clustering criterion in order to decide whether the random subset S of D forms a single cluster or not. In the latter case, CP proceeds by grouping all similar items of S in a single cluster C_1. This procedure is repeated on the remaining items of S in order to form the next cluster C_2 and so forth. The proposed pipeline is illustrated in Figure 1 and will be further analyzed in the subsequent subsections.

B. EVALUATION NETWORK

To train EN, the data need to be arranged through the following procedure; An input sample-set S_y is essentially a set that is assembled by the union of two distinct subsets. These subsets can contain items from one or multiple clusters, and thus labeled as $y = 1$ (pure) or $y = 0$ (impure), respectively. Either case can be expressed as:

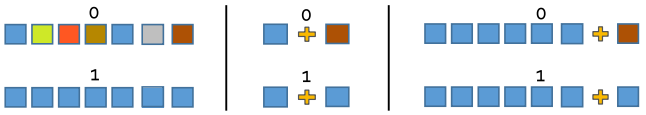

FIGURE 2. Different data combination approaches (left: set-plus-set, center: one-plus-one, right: set-plus-one). Different colors represent different K_l.

$$S_y \triangleq \begin{cases} T_{K_l,N_1} \cup T_{K_l,N_2}, & \text{if } y = 1 \\ T_{K_l,N_1} \cup T_{D-K_l,N_2}, & \text{if } y = 0 \end{cases} \quad (1)$$

where T_{K_l,N_1} is a set of N_1 elements from cluster K_l, T_{K_l,N_2} is a set of N_2 elements from cluster K_l and T_{D-K_l,N_2} stands for a set featuring N_2 items from classes different than K_l.

In the creation of the training, validation and test datasets, the number of S_1 and S_0 samples are equalized to avoid bias towards one or the other case. Three ways of combining data were examined during experimentation: a) set-plus-set, b) one-plus-one and c) set-plus-one. The different cases are obviously defined by the values N_1 and N_2 get.

In the set-plus-set case each subset contains $N_1, N_2 > 1$ items, forming the input sample S_y with cardinality $|S_y| > 2$. The ratio of main (T_{K_l,N_1}) and foreign (T_{D-K_l,N_2}) class items is adjusted by modifying a weight w, $w \in (0, 1)$. w is related to the subset length as $N_1 = wN_s$ and $N_2 = N_s(1 - w)$. In the one-plus-one mode, each subset contains $N_1, N_2 = 1$ item, subsequently forming an input sample S_y with cardinality $|S_y| = 2$. This mode is a simplified variation of the set-plus-set mode where a sample-set S_y is a tuple. Finally, the set-plus-one mode is a hybrid extension of the modes already described. Each sample S_y is produced by the union of a subset T_{K_l,N_1} of size $N_1 > 1$ and a subset T_{D-K_l,N_2} of size $N_2 = 1$. The intuition of this mode is that input samples S_0 with only one item from main class K_l are harder to be distinguished by EN, therefore allowing it to be trained more effectively. Figure 2 illustrates all the above different cases.

In the cases of set-plus-set and set-plus-one, the training, validation and test datasets contain samples S_y of variable length since the system should be able to perform clustering on sets of arbitrary size.

C. CLUSTERING PROCESS

The essence of the proposed clustering method is that, given a sample-set S_y of N_s items, each item $s_i, 1 \leq i \leq N_s$ is sequentially examined against the members of already formed clusters, $C_k, 1 \leq k \leq N_c$ where N_c denotes the total number of produced clusters at the current clustering stage.

In Algorithm 1 the Clustering Process is described. More specifically, the first item of the set forms the initial cluster C_1. The next item s_2 is selected and appended to cluster C_1, forming the temporary subset U_1, which is examined by EN as described in III-B. If the output of EN assesses that the members of the temporary subset U_1 belong to the same cluster, CP can move on by confirming that s_2 belongs to

Algorithm 1 Clustering

$C_1.append(s_1)$
for $i < N_s$ **do** ▷ for all items in S_y
  $counter \leftarrow 0$
  for $k < N_c$ **do** ▷ for all clusters
    for j<N_{C_k} **do** ▷ for all items in cluster
      $U_j \leftarrow \{s_i, c_{k,j}\}$
      if $EN(U_j) = 1$ **then**
        $counter+ = 1$
    if $counter > v_c \cdot N_{C_k}$ **then**
      $C_k.append(s_i)$
    else
      $B.append(s_i)$
  $S_y \leftarrow B$

C_1. Subsequently, the next item s_3 is forming two temporary clusters $U_1 = \{s_1, s_3\}$ and $U_2 = \{s_2, s_3\}$ with every item of C_1. CP decides if s_3 belongs to C_1 by comparing the ratio of the number of positive EN decisions to $|C_1|$ with a voting threshold v_c as follows:

$$\frac{1}{|C_1|}\sum_{j}^{|C_1|} EN(U_j) > v_c \tag{2}$$

The described procedure is repeated for every combination between each s_i and each item already appended to C_k, resulting in an EN decision for each $U_j = \{s_j, c_{k,j}\}, 1 \leq j \leq N_{C_k}$, where $c_{k,j}$ denotes the j-th item in cluster C_k, and N_{C_k} denotes the number of items already added. After the first run all items of S_y have either been assigned to C_1 or left to a remainder list B. When all items of S_y have been visited, Algorithm 1 starts over by selecting the first item from the remainder list B thus initiating C_2. The total number of Algorithm 1 executions equals the total number of clusters produced. The flowchart depicted in Figure 3 complements Algorithm 1 and schematically summarizes CP. Figure 4 illustrates, step-by-step, a clustering example.

D. MERGING

It has been noted that CP tends to produce an excessive number of clusters compared to the ground truth clusters' number (overclustering). To mitigate this behavior, a merging mechanism is introduced in order to combine smaller clusters into larger ones.

The number of clusters at the end of Algorithm 1 is closely linked to the selection of the cluster voting threshold v_c. Since the presented clustering method is non-parametric with respect to the number of clusters, a strict voting threshold v_c usually leads to more precise clusters, however casts the method prone to overclustering. On the other hand, a loose voting threshold v_c provides fewer and larger clusters but bears the risk of falsely accepting irrelevant samples into clusters. To remedy the situation, the proposed method initially adopts a strict voting threshold v_c in order to get a more precise initial clustering, and then utilizes the merging mechanism presented in Algorithm 2.

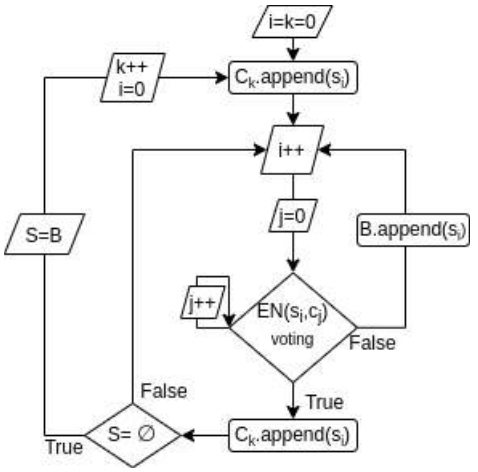

FIGURE 3. Flowchart of the Clustering Process (CP). The EN node describes the voting result after taking into account the criterion's decision for each formed subset U_j, as described in Algorithm 1.

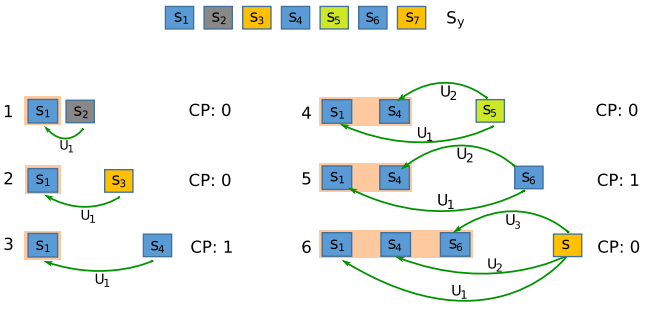

FIGURE 4. Example of the Clustering Process. The first item of the set, S_1, is picked, and all following items are sequentially examined by CP. If an item is recognized as a member of the same cluster (step 3), it is appended to the cluster, and the next CP decision (step 4) will be made after taking into account each EN(U_j) result for each formed subset U_j. The CP decision is shown on the right of each comparison. The green arrows represent the formation of each U_j subset. Different colors represent different K_l.

At the beginning, for each one of the initial clusters, the algorithm calculates a mean vector $\boldsymbol{\mu}_k$ from all associated feature vectors $\mathbf{c}_{k,j}$ of the items contained in cluster C_k. Therefore, for each cluster, the euclidean distance $d(\mathbf{c}_{k,j}, \boldsymbol{\mu}_k)$ between each feature vector $\mathbf{c}_{k,j}$ and the mean $\boldsymbol{\mu}_k$ is computed, as shown in (3). Note that $\boldsymbol{\mu}_k$ is a calculated feature vector not necessarily associated to any item of C_k.

$$d(\mathbf{c}_{k,j}, \boldsymbol{\mu}_k) = \sqrt{(\mathbf{c}_{k,j} - \boldsymbol{\mu}_k)^2} \tag{3}$$

After comparing all $d(\mathbf{c}_{k,j}, \boldsymbol{\mu}_k)$, the smallest distance between a vector $\mathbf{c}_{k,j}$ and $\boldsymbol{\mu}_k$ maps to $c_{k,j}$ element which is therefore considered as the representative item of each cluster k. Subsequently, the inter-cluster distances between the representative items of all produced clusters are calculated and a $N_c \times N_c$ distance matrix Z is generated. Based on distance matrix Z, the method creates tuples between similar clusters, with the smallest cluster considered the candidate and the larger one

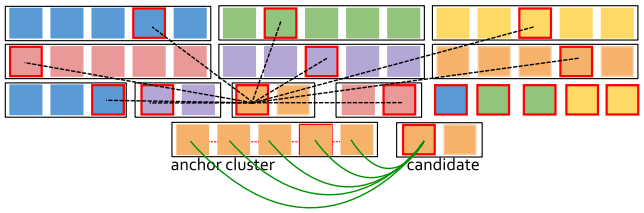

FIGURE 5. Example of merging the output clusters. Cluster representatives are illustrated by red borders. Dotted lines represent the calculated representatives' distances. Green lines picture the U_j subsets formed with each item of the anchor cluster and their voting is shown in red dotted line.

Algorithm 2 Merging

for $k < N_c$ **do** ▷ for all clusters
  for $j < N_{C_k}$ **do** ▷ compute intra cluster distances
    $dist \leftarrow d(\mathbf{c}_{k,j}, \boldsymbol{\mu}_k)$
    $r.append(argmin(dist))$ ▷ representatives
for $i, j < size(r)$ **do** ▷ get all representative distances
  $Z.append(d(\mathbf{r_i}, \mathbf{r_j})$
for $k < N_c$ **do** ▷ compute inter cluster distances
  $candidate \leftarrow argmin(Z)$
  $counter \leftarrow 0$
  for $j < N_{C_a}$ **do** ▷ for all items in anchor cluster
    $U_j \leftarrow \{candidate, C_{a,j}\}$
    if $EN(U_j) = 1$ **then**
      $counter+ = 1$
  if $counter > v_m$ **then**
    $C_a.extend(C_k)$

the anchor. This convention is assumed so that the smaller cluster is always appended to the larger one and not vice-versa. It has to be noted that due to the nature of EN, a large and precise (having items only from one ground truth cluster) cluster is expected to be more robust than a smaller equivalently precise cluster.

Clusters are then merged by comparing the representative of the candidate cluster with all items of the anchor cluster through a voting mechanism similar to the one described in Algorithm 1. More precisely, if the total number of votes surpasses the ratio of the cluster cardinality $|C_k|$ to a merging threshold v_m, then the candidate cluster is appended to the anchor. During experimentation, various v_m values were tested and it was observed that a strict initial threshold applied on clustering yields the best results when relaxed during merging. Figure 5 illustrates how each candidate cluster is matched with an anchor cluster, based on the features' distance of their representative items.

Nonetheless, the merging procedure is exhaustive in nature and can render the whole procedure cumbersome when dealing with large datasets. To avoid unnecessary calculations, the decision on whether a new item should be accepted in an existing cluster or not is taken when the total number of votes for either case surpasses the required number of votes, defined by $N_{C_k} \cdot v_m$. Algorithm 2 describes the merging procedure in a more systematic manner, accompanied with the flowchart in Figure 6.

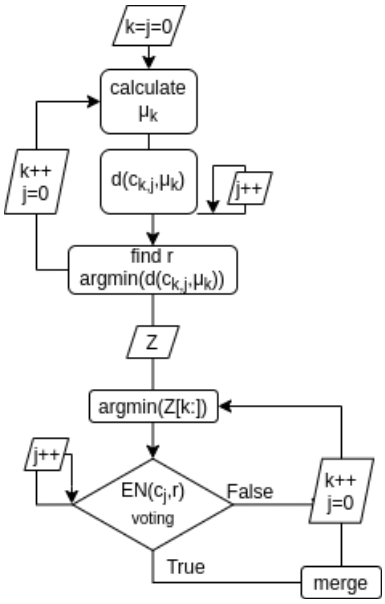

FIGURE 6. Flowchart of the merging mechanism as described in Algorithm 2. An iterative process calculates the matrix of representative distances **Z**, which is then used to decide which clusters may merge with larger ones. The EN is employed again, in order to double check if the candidate cluster should be merged with the respective anchor cluster.

IV. DATA PROCESSING

In this Section the input data organization in order to be processed by both EN and CP is described. Moreover, an extended list of the used datasets is provided, accompanied with the individual data processing conducted for each one.

A. FEATURE EXTRACTION AND INPUT TRANSFORMATION

Instead of using the original data as input for the network, the proposed method utilizes the feature map embeddings of a pre-trained ResNet-50 classification network and more precisely the implementation of torchvision [31], by excluding the fully connected and softmax layers in order to only extract 2048×1 feature maps.

The system's ability to process samples of variable length is desirable but beckons the adversity of inconsistent input size. To alleviate this, instead of the original input data, the network is fed with the data outer product matrix $\mathbf{P}$. Let s_i be an item that belongs to an input sample-set S_y and $\mathbf{S}$ be a matrix that includes the feature vector representation of each s_i, namely $\mathbf{S} = [\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_{N_s}}]$. Given that the total number of items is N_s and the dimension of each feature vector is t, then the dimension of $\mathbf{S}$ is $N_s \times t$. Consequently, the dimension of $\mathbf{S}'$ is $t \times N_s$. The $t \times t$ outer product of $\mathbf{S}$ is therefore acquired by multiplying $\mathbf{S}'$ by $\mathbf{S}$. Finally, the resulting outer product is:

$$\mathbf{P} = \mathbf{S}'\mathbf{S} = \sum_{i=1}^{N_s} \mathbf{s_i'}\mathbf{s_i} \tag{4}$$

To normalize the resulting values, the mean value of elements of $\mathbf{P}$, denoted as p_μ is firstly subtracted and then divided by N_S. (5).

$$\mathbf{P_{norm}} = \frac{P - p_\mu J_t}{N_S}, \tag{5}$$

where J_t denotes a $t \times t$ all-ones matrix. The described data transformation shifts input samples from a length variant input to an input of fixed dimensions, essential for feeding each sample-set S_y to the EN. This shape transformation is crucial when input samples are created by the set-plus-set or the set-plus-one modes, i.e when $N_s > 2$. Alternatively, if a sample is a tuple constructed by the one-plus-one mode, the combination of the two items could either be a special case of (4), where $N_s = 2$, or a simple tensor concatenation.

B. LIST OF DATABASES

- **Columbia Object Image Library (COIL)** [32]: There are two versions of the COIL database, COIL20 and COIL100. COIL20 is a collection of 1440, 128 x 128 grayscale images acquired from 20 objects, whereas COIL100 is a set of objects with a wider variety of complex geometric and reflectance characteristics compared to COIL20. The database consists of 7200 color (RGB) images of 100 objects captured in the same manner as COIL20.
- **MNIST** [33]: The MNIST database is one of the most known and easily recognizable image databases. MNIST features 70000 hand written numbers of 10 classes split in 60000 train and 10000 test samples.
- **Fashion-MNIST** [34]: Fashion-MNIST is designed as a more challenging MNIST dataset for benchmarking machine learning algorithms and is structured identically to MNIST in order to be able to substitute the latter in experiments; it consists of a training set of 60000 examples and a test set of 10000 28 x 28 grayscale examples of 10 classes.
- **USPS Handwritten Digits (USPS)** [35]: Another well-known set of handwritten digits featuring 7291 training and 2007 8 x 8 testing examples organized in 10 categories.
- **YouTube Faces Database (YTF)** [36]: A database of frames captured from face videos designed for unconstrained face recognition. The YTF database contains 3425 videos of 1,595 different people. The proposed approach is evaluated on the first 40 subjects of the dataset as in [12], [24], [29], [37], which roughly contain 10000 images.

Regarding the way data are fed to EN, the one-plus-one mode was finally selected after experimenting with the different variants of data processing and different mixture weights w as defined in Section III-B. The set-plus-set mode proved to be a high resource consuming option with questionable results due to the vast amount of different data combinations. Furthermore, a dataset featuring so diverse samples can potentially catapult the dimensionality of the problem and subsequently the network fails to generalize and produce certain task specific rules. Finally, the set-plus-one mode would need a far deeper network to understand the narrow differences between cluster and non-cluster samples.

C. DATA SPLITTING AND AUGMENTATION

The first step of data handling is to obtain the transformed samples that will augment the existing dataset in case overfitting is observed while training. The original data are transformed by applying 32 $\times$ 32 pixel random crops with a padding of 4 pixels and random horizontal and vertical flips. After obtaining the feature embeddings of both original and augmented data as described in IV-A, both are managed identically but independently to avoid any conflict. For COIL20, COIL100 and YouTube Faces datasets that do not provide train and test sets, the original and augmented features are split by keeping 70% of data for training and 30% for testing purposes. The same ratio is applied again on the training data in order to check the training progress on a constant validation set. In case a dataset training is evaluated by using cross validation, this is achieved in five folds; one fold is validated while training on the other four.

V. EXPERIMENTAL RESULTS

The proposed method is established on a binary classifier that needs to be able to process high dimensional data, coming from multiple, diverse sources on a minimum resource overhead. Three ResNet [38] architectures were tested, featuring 18, 34 or 50 network layers, respectively. Each residual block in the 50 layer version comprises three layers, whereas residual blocks of the smaller ResNet18 and ResNet34 networks are two-layer deep.

A. PERFORMANCE AND SPEED TRADE-OFF

In order to inspect all three architectures in terms of performance and execution time, the proposed method is applied on the COIL20 database, which is the smallest and simplest one in terms of number of items, image size and color information. All three experiments have been performed on a machine equipped with an nVidia GeForce GTX 1080 GPU, 128GB of RAM and an Intel Xeon E5-2620 v4 @ 2.10GHz CPU. In Table 1, the performance and clustering duration of each architecture are shown. Performance is measured in F-score and clustering duration in minutes. The fastest implementation is achieved when EN utilizes a ResNet18 architecture, however its performance is inferior to the 34 layer version, which achieves a perfect clustering. Finally, the 50 layer network fails to yield results on par with the smaller architectures as it fails to generalize. This can be seen, since during training, EN is processing each sample-set independently with an abundance of ways, thus leading to overfitting. Overfitting can also occur when the training sample-sets are so distinct that the model struggles to fit them all, a phenomenon which is referred as the curse of dimensionality in the machine learning context [39]. Apart

TABLE 1. Performance and duration of the three examined ResNet architectures during clustering on the COIL20 database. The most complex architecture needs 2.5× more time in order to achieve equal performance to the simplest one, and, is therefore rejected.

	ResNet18	ResNet34	ResNet50
Performance	0.932	**0.999**	0.933
Duration	**32.5**	54.3	99.1

from the performance results, the most complex version of the network needs more time to implement clustering.

The first convolution layer of all ResNet implementations is applied with a large stride in order to scale down parameters in the following layers. During training, the learning rate is scheduled to decrease, starting from a value of $lr = 10^{-3}$, and the sigmoid layer is omitted. Instead, the sigmoid function is calculated in conjunction with the network losses by the binary cross entropy of the network output and the target label, as this approach is reported to be more numerically stable [40]. The sigmoid layer is activated again during inference.

B. CLUSTERING AND MERGING THRESHOLDS

As described in III-C and III-D, CP depends on the clustering and merging thresholds v_c and v_m, which adjust how rigorous the method is towards accepting new items to an already formed cluster, or merging smaller clusters with larger ones. These threshold values essentially indicate the number of required votes prior to accepting or rejecting a clustering or merging candidate. Large v_c and v_m values represent strict threshold values, as more votes are required to make a decision. For instance, $v_c = 1$ and $v_m = 1/2$ denote that all items of a formed cluster must vote positively for the insertion of a new candidate into the cluster, whereas only half of the votes are required for merging.

Tables 4, 5 and 6 illustrate how different v_c and v_m value combinations affect the performance of the method on the COIL20, COIL100 and MNIST-test databases. For this ablation study, only 50% of the overall dataset has been used, which justifies the slight variation in performance scores when compared to the overall final results in V-D. As it can be seen in all tables, a very strict threshold value combination produces an excessive number of small clusters, many of which comprise only one item (singleton clusters). This behavior is expected for the following two reasons; Firstly, a large v_c value is responsible for preventing new items from being appended to existing clusters easily, which can heavily affect the precision score and secondly, even if some small clusters are formed, a large v_m value restrains the merging mechanism from combining them into larger ones. In the opposite case, small threshold values result in less clusters comprising more items, that may, however, falsely include items of multiple classes, thus negatively affecting the recall score.

C. METRICS

The proposed clustering method is evaluated by adopting three measures. The normalized mutual information (NMI) [46], accuracy (ACC) as proposed in [37], and an F-score, as proposed in [47]. Although NMI has been broadly used in comparing different clustering approaches, it is a measure that requires the number of clusters that the method yields to be equal to the one of the ground truth. Since the presented method is based on the absence of this parameterization, the total number of clusters and consequently the number of items contained in a cluster is unknown. Thus, the NMI measure fails to precisely judge clusters with more or fewer items than the respective ground truth cluster. To calculate the NMI score of the proposed clustering method, an approach similar to the implementation of [37] is adopted after modifying the cardinality of a produced cluster $|C_k|$ to match the ground truth cluster's cardinality $|K_l|$ or vice versa by discarding items from either C_k or K_l. The decision on which cluster items will be discarded is established on the distance matrix between each item and the cluster's representative, generated for every cluster, as described in Section III-D. Equation (6) shows how NMI is calculated for a number of clusters that has not been predefined, after modifying the produced cluster's or ground truth's cardinality to q (where $q = min(|C_k|, |K_l|)$).

$$NMI(C, K) = \frac{1}{N_c\, N_k} \sum_{k=1}^{N_C} \sum_{l=1}^{N_K} \frac{MI(C'_k, K'_l)}{\sqrt{H(C'_k)H(K'_l)}}, \tag{6}$$

where $H(\cdot)$ expresses the entropy, $MI(\cdot, \cdot)$ is the mutual information of a cluster C'_k and ground truth class K'_l after modifying their cardinality to $q_{k,l}$. Despite the fact that NMI is considered a standard clustering metric, some related work uses the adjusted mutual information (AMI) score instead, because of the known drawback of NMI to favor fine-grained partitions [48]. AMI is defined as:

$$AMI(C, K) = \frac{1}{N_c\, N_k} \sum_{k=1}^{N_C} \sum_{l=1}^{N_K} \frac{MI(C'_k, K'_l)E(C'_k, K'_l)}{\sqrt{H(C'_k)H(K'_l)} - E(C'_k, K'_l)} \tag{7}$$

Another popular clustering measure is the cluster accuracy (ACC). ACC of a cluster to the ground truth is defined as

$$ACC = \frac{1}{N_c\, N_k} \sum_{k=1}^{N_C} \sum_{l=1}^{N_K} 1(C'_k = K'_l) \tag{8}$$

where $1(C'_k = K'_l) = 1$ if $C'_k = K'_l$ and 0, otherwise. Since there is no information provided on how the ground truth clusters are matched to the produced clusters, the best permutation is found by first constructing an $N_c \times N_c$ cost matrix and then solving the linear sum assignment problem. The fact that the ACC and NMI metrics are inadequate to provide accurate results is the intuition behind residing to a metric that simultaneously measures performance both quantitatively and qualitatively. For each produced cluster C_k its precision is computed with respect to a ground truth cluster K_l as $prec(C_k, K_l) = |C_k \cap K_l|/|C_k|$, and the recall of C_k with respect to K_l is defined as $rec(C_k, K_l) = |C_k \cap K_l|/|K_l|$.

TABLE 2. Performance of the proposed method against other state-of-the-art non-parametric methods. * marks methods evaluated by AMI instead of NMI. Empty cells denote that no results were reported by the respective methods on the specific database.

	COIL20				COIL100				MNIST-test			
	F	NMI	ACC	Time	F	NMI	ACC	Time	F	NMI	ACC	Time
[12] (2017)						0.957*				0.893*		
[37] (2018)						0.962*	0.858			0.913*	0.963	
[30] (2018)										0.927	0.970	
[29] (2018)												
Ours-34	**0.996**	**1.000**	**1.000**	54	**0.937**	**0.978**	**0.991**	2280	0.963	0.779	0.989	22057
Ours-18	0.932	0.625	0.919	52	0.838	0.945	0.990	1346	**0.967**	0.852	**0.999**	9568

	fashion-MNIST				YTF				USPS			
	F	NMI	ACC	Time	F	NMI	ACC	Time	F	NMI	ACC	Time
[12] (2017)						0.850*						
[37] (2018)						0.903*	0.699					
[30] (2018)		0.661	0.609							0.939	0.977	
[29] (2018)					0.906	0.960						
Ours-34	0.789	**0.903**	0.956	15771	0.923	**0.993**	**0.999**	1020	**0.926**	0.817	**0.999**	7779
Ours-18	**0.941**	0.882	**0.991**	8764	**0.963**	0.980	0.993	540	0.918	0.864	0.993	1963

A high precision score implies that most items in the cluster are correctly grouped together in that cluster, but the algorithm may have missed samples that should have been included too. On the other hand, a high recall score means that the method has accurately clustered most images of a specific class but has also included samples that belong to other classes. Given the precision and recall of two clusterings, their F-score is defined as:

$$F(C_k, K_l) = 2 \times \frac{prec(C_k, K_l) \times rec(C_k, K_l)}{prec(C_k, K_l) + rec(C_k, K_l)}. \quad (9)$$

It is safe to assume that for each C_k, K_l pair the highest F-score value is located where the mutual information is maximized. To conclude to a single score, an overall F-score is measured as:

$$overallF = \sum_{l=1}^{N_K} \frac{|K|}{|S|} \times \max_{i:1,\ldots,N_C} F(C_k, K_l) \quad (10)$$

where N_K is the number of ground truth clusters, N_C is the number of clusters that the method extracted.

D. RESULTS AND DISCUSSION

Tables 2 and 3 show how the proposed work compares to other clustering methods. For each experiment, the clustering and merging threshold values, v_c and v_m, were selected based on the ablation study presented in V-B. Since there is a abundance of databases on which clustering methods are evaluated on, and the employed databases do not always match, the proposed method is evaluated on as many common databases as possible. More specifically, Table 2 displays the performance of the proposed method against other non-parametric methods. The significant increase in total process time across experiments indicates that the proposed method copes with handling larger datasets; CP needs to exclusively scan all items of the given dataset at least once ($O(n)$) and then repeat the process on the set of unclustered items as many times as needed, with the worst-case scenario being that all items remain unclustered, as the method yields only singleton clusters ($O(n!)$). However, the exhaustive nature of the proposed method leads to notable results, as it exploits the advances in classification methods and the increasing improvements in available hardware. As a future work, new strategies will be investigated towards avoiding the worst-case scenario, by either random cluster initialisation or clustering preprocessing for initial clusters.

Table 3 illustrates the performance of our work against methods that also exploit the knowledge of the number of clusters (parametric methods). Despite the significant advantage of the parametric clustering methods, the method presented in this work is capable of being on par with their performance, and moreover, even surpassing them at certain datasets. In order to be able to compare to all methods, the efficiency of this approach is also calculated using F-score, ACC and where noted with *, AMI. For the parametric methods presented on Table 3, the selection of NMI, ACC or AMI is expected since the number of clusters is predefined. However, for non parametric methods, in order to employ these metrics, the conventions described in V-C should be applied. These conventions question the reliability of the NMI, AMI and ACC measures, when the expected number of clusters is unknown, thus this work considers the F-score as a more fair clustering metric.

E. CROSS DATASET CLUSTERING

In addition to the favorable results presented in Section V-D, the proposed method is further tested on cases where the Evaluation Network is trained on a database but the learned weights are used by the Clustering Process on another

TABLE 3. Performance of the proposed method against state-of-the-art methods where the number of clusters is known a priori. Empty cells denote that no results were reported by the respective methods on the specific database.

	COIL20				COIL100				MNIST-test			
	F	NMI	ACC	Time	F	NMI	ACC	Time	F	NMI	ACC	Time
[28] (2015)										0.830	0.856	
[24] (2016)		**1.000**	**1.000**			**0.978**	0.916			0.913	0.961	
[26] (2017)										**0.915**	0.963	
[41] (2017)		0.895	0.793			0.905	0.775			0.917	0.964	
[42] (2019)										**0.963**	0.987	
[43] (2019)										0.957	0.985	
[44] (2020)										0.882	0.948	
[45] (2018)										0.885	0.871	
Ours-34	**0.996**	**1.000**	**1.000**	54	**0.937**	**0.978**	**0.991**	2280	0.963	0.779	0.989	22057
Ours-18	0.932	0.625	0.919	52	0.838	0.945	0.990	1346	**0.967**	0.852	**0.999**	9568

	fashion-MNIST				YTF				USPS			
	F	NMI	ACC	Time	F	NMI	ACC	Time	F	NMI	ACC	Time
[28] (2015)		0.546	0.518							0.767	0.762	
[24] (2016)						0.848	0.684			0.915	0.950	
[26] (2017)						0.802	0.621			**0.927**	0.964	
[41] (2017)										0.724	0.743	
[42] (2019)		0.642	0.591							**0.948**	0.981	
[43] (2019)		0.662	0.586							**0.948**	0.981	
[44] (2020)		0.684	0.672							0.901	0.958	
[45] (2018)						0.801	0.606			0.948	0.979	
Ours-34	0.789	**0.903**	0.956	15771	0.923	**0.993**	**0.999**	1020	**0.926**	0.817	**0.999**	7779
Ours-18	**0.941**	0.882	**0.991**	8764	**0.963**	0.980	0.993	540	0.918	0.864	0.993	1963

TABLE 4. Performance (F-score) and number of extracted clusters of the method with various clustering and merging threshold combinations, using the ResNet18 architecture on the COIL20 database.

v_m \ v_c	1		1/2		1/3		1/4		1/5	
	F	N_C	F	N_C	F	N_C	F	N_C	F	N_C
1	0.425	78	0.900	39	0.921	35	0.944	30	0.946	32
1/2	0.895	32	0.976	26	0.937	31	0.956	29	0.957	27
1/3	0.951	27	0.976	26	0.930	32	0.970	26	0.954	27
1/4	0.963	23	**0.992**	22	0.910	24	0.931	32	0.956	22
1/5	0.904	31	0.988	21	0.985	22	0.982	24	0.943	30

TABLE 5. Performance (F-score) and number of extracted clusters of the method with various clustering and merging threshold combinations, using the ResNet34 architecture on the COIL100 database.

v_m \ v_c	1		1/2		1/3		1/4		1/5	
	F	N_C	F	N_C	F	N_C	F	N_C	F	N_C
1	0.358	496	0.894	203	0.902	181	0.898	197	0.908	185
1/2	0.859	189	0.909	182	0.927	166	0.910	169	0.933	159
1/3	0.896	156	0.938	140	0.939	154	0.937	138	0.960	141
1/4	0.921	137	0.942	144	0.946	139	0.940	137	0.935	149
1/5	0.915	144	**0.962**	132	0.946	135	0.935	139	0.948	139

dataset. The results on Table 7 illustrate that the proposed method is capable of providing adequate results even when CP is evaluated on data very different than the ones EN has been trained on. The MNIST and FashionMNIST databases are identical in terms of size and structure, they, however, feature very different context. The fact that the FashionMNIST database is a collection of clothing images and the MNIST-test database contains handwritten digits, justifies EN's ability to extract patterns from dissimilar modalities.

TABLE 6. Performance (F-score) and number of extracted clusters of the method with various clustering and merging threshold combinations, using the ResNet18 architecture on the MNIST-test database. Very strict threshold values v_c and v_m, depicted in the first column, force CP to produce an excessive amount of singleton or two-item clusters, and consequently, a very low F-score.

v_m \ v_c	1		1/2		1/3		1/4		1/5	
	F	N_C	F	N_C	F	N_C	F	N_C	F	N_C
1	0.007	2503	0.957	141	0.963	165	0.963	174	0.963	188
1/2	0.119	822	0.967	170	0.963	161	0.964	157	0.954	171
1/3	0.077	1197	0.964	161	0.958	154	0.962	174	0.896	136
1/4	0.556	188	0.964	163	0.963	160	0.942	155	**0.965**	169
1/5	0.565	163	0.901	159	0.912	155	**0.965**	141	0.963	168

TABLE 7. Cross dataset clustering; EN trained on fashionMNIST whereas CP clusters MNIST and vice-versa.

EN \ CP	fashionMNIST	MNIST-test
fashionMNIST	x	0.940
MNIST-test	0.922	x

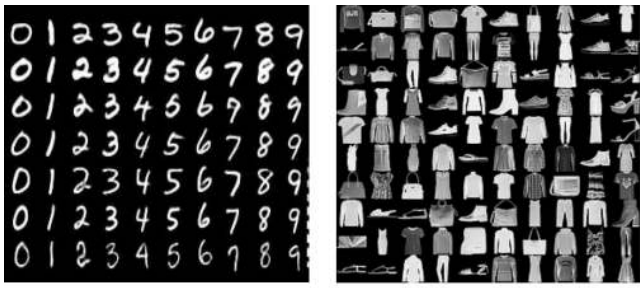

FIGURE 7. Samples from the cross dataset tests performed on databases of disparate modalities; MNIST-test (left) features images of hand written digits, FashionMNIST (right) features images of clothing.

Samples of both databases are illustrated in Figure 7. For both tests, the accuracy is calculated with F-Score and the weights model is produced by training EN on the lightweight 18-layer ResNet network. Cross dataset clustering tests are a real-world challenge, as data are not always available in volumes, as in public datasets.

F. CONCLUSION AND FUTURE WORK

This work presented a novel clustering methodology based on convolutional neural networks that does not require any number of clusters information to be provided beforehand. The approach demonstrated on this work outperforms most approaches that do not utilize this knowledge. In addition, it is on par or even outperforms the state-of-the-art regarding most parametric clustering approaches. Finally, the proposed method demonstrates the ability to cluster images from modalities that it has never encountered before.

Regarding future work, the proposed method can extend to apply to data of additional modalities such as sound or text, by employing related networks and fine-tuning the methodology accordingly. To ameliorate the extended runtime complexity, alternative algorithmic strategies can be approached for CP, as already examined in the literature, by combining the proposed method with well-known algorithms such as DBSCAN or t-SNE. Fusing effective clustering algorithms with the proposed pipeline does not modify the core methodology and objective of this work, that is, as described in III-B, the development of a strong, reliable clustering criterion. Finally, it has to be noted that the method's runtime speed is closely linked to the architecture of the employed network, as already shown in V-A. Although utilizing a simpler and more efficient network than the ResNet equivalent architectures will not reduce the algorithm's overall complexity, it can definitely help achieving a faster inference.

REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 1998, pp. 94–105.

[2] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and scalable subspace clustering for very large data sets," Center Parallel Distrib. Comput., Dept. Elect. Comput. Eng., Northwestern Univ. Technol. Inst., Evanston, IL, USA, Tech. Rep. CPDC-TR-9906-010, 1999.

[3] C.-H. Cheng, A. W. Fu, and Y. Zhang, "Entropy-based subspace clustering for mining numerical data," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 1999, pp. 84–93.

[4] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.

[5] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. D. Reid, "Deep subspace clustering networks," in *Proc. NIPS*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds. 2017, pp. 24–33. [Online]. Available: http://dblp.uni-trier.de/db/conf/nips/nips2017.html#JiZLSR17

[6] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Proc. Iberoamer. Congr. Pattern Recognit.*, 2013, pp. 117–124.

[7] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1986.

[8] N. Arunkumar, M. A. Mohammed, M. K. Abd Ghani, D. A. Ibrahim, E. Abdulhay, G. Ramirez-Gonzalez, and V. H. C. de Albuquerque, "K-means clustering and neural network for object detecting and identifying abnormality of brain tumor," *Soft Comput.*, vol. 23, no. 19, pp. 9083–9096, Oct. 2019.

[9] A. Nithya, A. Appathurai, N. Venkatadri, D. R. Ramji, and C. A. Palagan, "Kidney disease detection and segmentation using artificial neural network and multi-kernel k-means clustering for ultrasound images," *Measurement*, vol. 149, Jan. 2020, Art. no. 106952.

[10] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. Nat. Conf. Artif. Intell.*, vol. 2, 2014, pp. 1293–1299.

[11] M. Abavisani and V. M. Patel, "Deep multimodal subspace clustering networks," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 6, pp. 1601–1614, Dec. 2018, doi: 10.1109/jstsp.2018.2875385.

[12] S. A. Shah and V. Koltun, "Robust continuous clustering," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 37, pp. 9814–9819, Sep. 2017.

[13] M. R. Brito, E. L. Chávez, A. J. Quiroz, and J. E. Yukich, "Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection," *Statist. Probab. Lett.*, vol. 35, no. 1, pp. 33–42, Aug. 1997.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*. Palo Alto, CA, USA: AAAI Press, 1996, pp. 226–231.

[15] M. Parmar, D. Wang, X. Zhang, A.-H. Tan, C. Miao, J. Jiang, and Y. Zhou, "REDPC: A residual error-based density peak clustering algorithm," *Neurocomputing*, vol. 348, pp. 82–96, Jul. 2019.

[16] T. Semertzidis, D. Rafailidis, M. G. Strintzis, and P. Daras, "Large-scale spectral clustering based on pairwise constraints," *Inf. Process. Manage.*, vol. 51, no. 5, pp. 616–624, Sep. 2015.

[17] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[18] Z. Lu and M. A. Carreira-Perpinan, "Constrained spectral clustering through affinity propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[19] I. Givoni and B. J. Frey, "A binary variable model for affinity propagation," *Neural Comput.*, vol. 21, no. 6, pp. 1589–1600, 2009.

[20] F. Shang, L. C. Jiao, J. Shi, F. Wang, and M. Gong, "Fast affinity propagation clustering: A multilevel approach," *Pattern Recognit.*, vol. 45, no. 1, pp. 474–486, Jan. 2012.

[21] D.-Y. Xia, F. Wu, X.-Q. Zhang, and Y.-T. Zhuang, "Local and global approaches of affinity propagation clustering for large scale data," *J. Zhejiang Univ.-Sci. A*, vol. 9, no. 10, pp. 1373–1381, Oct. 2008, doi: 10.1631/jzus.a0720058.

[22] R. Refianti, A. Mutiara, and S. Gunawan, "Time complexity comparison between affinity propagation algorithms," *J. Theor. Appl. Inf. Technol.*, vol. 95, pp. 1497–1505, Apr. 2017.

[23] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 769–783, May 2000.

[24] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 5147–5156.

[25] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Computer Vision—ECCV* (Lecture Notes in Computer Science). Springer, 2018, pp. 139–156, doi: 10.1007/978-3-030-01264-9_9.

[26] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proc. ICCV*. IEEE Computer Society, 2017, pp. 5747–5756. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2017.html#DizajiHDCH17
[27] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[28] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. ICML*, in JMLR Workshop and Conference Proceedings, vol. 48, M.-F. Balcan and K. Q. Weinberger, Eds. JMLR.org, 2016, pp. 478–487. [Online]. Available: http://dblp.uni-trier.de/db/conf/icml/icml2016.html#XieGF16
[29] W.-A. Lin, J.-C. Chen, C. D. Castillo, and R. Chellappa, "Deep density clustering of unconstrained faces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8128–8137.
[30] Y. Ren, N. Wang, M. Li, and Z. Xu, "Deep density-based image clustering," in *Knowledge-Based Systems*, vol. 197. Amsterdam, The Netherlands: Elsevier, Jun. 2020, p. 105841, doi: 10.1016/j.knosys.2020.105841.
[31] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proc. Int. Conf. Multimedia (MM)*, New York, NY, USA, 2010, pp. 1485–1488.
[32] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-20)," Dept. Comput. Sci., Columbia Univ., New York, NY, USA, Tech. Rep. CUCS-006-96, 1996.
[33] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," *ATT Labs*, vol. 2, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist
[34] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747
[35] A. Seewald, "Digits—A dataset for handwritten digit recognition," Österreichisches Forschungsinstitut Artif. Intell., Vienna, Austria, Tech. Rep. TR-2005-27, 2005.
[36] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. CVPR*, Jun. 2011, pp. 529–534.
[37] S. A. Shah and V. Koltun, "Deep continuous clustering," *CoRR*, vol. abs/1803.01449, 2018. [Online]. Available: http://arxiv.org/abs/1803.01449
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA: IEEE Computer Society, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[39] R. Bellman, *Dynamic programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957, p. 342.
[40] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS Workshop Autodiff*, Long Beach, CA, USA, 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ
[41] F. Li, H. Qiao, B. Zhang, and X. Xi, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognit.*, vol. 83, pp. 161–173, 2018, doi: 10.1016/j.patcog.2018.05.019.
[42] N. Mrabah, N. Mefraz Khan, R. Ksantini, and Z. Lachiri, "Deep clustering with a dynamic autoencoder: From reconstruction towards centroids construction," 2019, *arXiv:1901.07752*. [Online]. Available: http://arxiv.org/abs/1901.07752
[43] W. Zhou and Q. Zhou, "Deep embedded clustering with adversarial distribution adaptation," *IEEE Access*, vol. 7, pp. 113801–113809, 2019.
[44] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock, "N2D: (Not too) deep clustering via clustering the local manifold of an autoencoded embedding," in *Proc. 25th Int. Conf. Pattern Recognit.* Piscataway, NJ, USA: IEEE Press, Aug. 2019.
[45] M. Jabi, M. Pedersoli, A. Mitiche, and I. Ben Ayed, "Deep clustering: On the link between discriminative models and K-means," *IEEE Trans. Pattern Anal. Mach. Intell.*, p. 1, 2020, doi: 10.1109/tpami.2019.2962683.
[46] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Is a correction for chance necessary?" in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2009, pp. 1073–1080.
[47] C. Chrysouli, N. Vretos, and I. Pitas, "Face clustering in videos based on spectral clustering techniques," in *Proc. 1st Asian Conf. Pattern Recognit.*, Nov. 2011, pp. 130–134.
[48] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Jan. 2010.

CHRISTOS AVGERINOS received the B.Sc. degree from the Department of Applied Informatics, University of Macedonia, in February 2013, and the M.Sc. degree in digital media from the Aristotle University of Thessaloniki, in September 2015. Since January 2017, he has been a Research Assistant with the Visual Computing Laboratory (VCL), Centre for Research and Technology Hellas-Information Technologies Institute. His research interests include computer vision, computational intelligence, image analysis, and deep learning.

VASSILIOS SOLACHIDIS was a Researcher in 16 European and national research projects with the Informatics Department, Information Technologies Institute, Aristotle University of Thessaloniki, and the Applied Electronics Department, Università degli Studi Roma Tre. He was also an Adjunct Lecturer with the Aristotle University of Thessaloniki, the University of Western Macedonia, and the Technological Institutes of Thessaloniki, Kavala, Serres, and Western Macedonia. He has eight articles in international scientific peer-review journals and 35 papers in international and national conferences. His research interests include technologies of image and video analysis and multimedia watermarking.

NICHOLAS VRETOS (Member, IEEE) received the B.Sc. degree in computer sciences from the University Pierre et Marie Curie (Paris VI), in 2002, and the Ph.D. degree from the Aristotle University of Thessaloniki, in 2012. During elaboration of his thesis, he taught as an Assistant and worked as a Research Assistant with the Artificial Intelligence and Information Analysis Laboratory. He has worked in more than ten European projects as a Technical Manager and/or WP Leader and/or researcher. He is a currently a Research Fellow with CERTH-ITI. He has published more than 70 articles in scientific journals and conference proceedings and two book chapters. His main research interests include image and video processing, semantic analysis, neural networks, and 3-D data processing. He has committed as a reviewer for several journals and conferences in the field of image, video, and 3D processing. Moreover, he has been the pointed chair and/or co-chair in several conferences and/or workshops of his field of interest.

PETROS DARAS (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Aristotle University of Thessaloniki, Greece, in 1999, 2002, and 2005, respectively. He is currently a Research Director and the Chair of the Visual Computing Laboratory. He has been involved in more than 50 projects funded by EC and the Greek Ministry of Research and Technology. His involvement with research areas has led to the coauthoring of more than 300 articles in refereed journals and international conferences. His main research interests include visual content processing, multimedia indexing, search engines, recommendation algorithms, and relevance feedback.