A DEEP LEARNING OBJECT DETECTION METHOD FOR AN EFFICIENT CLUSTERS INITIALIZATION

Raphaël Couturier
Univ. Bourgogne Franche-Comté (UBFC),
FEMTO-ST Institute,
France

Hassan N. Noura
Univ. Bourgogne Franche-Comté (UBFC),
FEMTO-ST Institute,
France

Ola Salman
American University of Beirut,
Electrical and Computer Engineering Department,
Beirut 1107 2020, Lebanon

Abderrahmane Sider
Laboratoire LIMED, University of Bejaia, Algeria

July 6, 2021

ABSTRACT

Clustering is an unsupervised machine learning method grouping data samples into clusters of similar objects. In practice, clustering has been used in numerous applications such as banking customers profiling, document retrieval, image segmentation, and e-commerce recommendation engines. However, the existing clustering techniques present significant limitations, from which is the dependability of their stability on the initialization parameters (e.g. number of clusters, centroids). Different solutions were presented in the literature to overcome this limitation (i.e. internal and external validation metrics). However, these solutions require high computational complexity and memory consumption, especially when dealing with big data. In this paper, we apply the recent object detection Deep Learning (DL) model, named YOLO-v5, to detect the initial clustering parameters such as the number of clusters with their sizes and centroids. Mainly, the proposed solution consists of adding a DL-based initialization phase making the clustering algorithms free of initialization. Two model solutions are provided in this work, one for isolated clusters and the other one for overlapping clusters. The features of the incoming dataset determine which model to use. Moreover, The results show that the proposed solution can provide near-optimal clusters initialization parameters with low computational and resources overhead compared to existing solutions.

Keywords Clustering algorithms; K-means; Clustering initialization parameters; Deep Learning object detection model

1 Introduction

Clustering is an efficient solution to split a data-set into a set of clusters, that are characterized by the high similarity within the same cluster and the high distance between different clusters [1, 2]. Mainly, the clustering methods are unsupervised machine learning methods that can be parametric (probability-based) or non-parametric [3]. The non-parametric clustering methods are often based on an empirical function that measures the similarity or dissimilarity between the data points [4]. On the other hand, the parametric methods assume some mapping functions or probability distributions and try to find their best parameters to optimally cluster the data points. In addition, the clustering methods can be classified into two main types, which are hierarchical and partitional. With partitional clustering methods, which are most employed, the data point belongs to one cluster [5]. However, with hierarchical clustering, there are nested clusters, such that a data point can belong to many clusters at a time.

1.1 Problem Formulation

Unfortunately, the clustering algorithms suffer from several challenges, preventing their reliability and efficiency. A main challenge is that the stability and convergence of the clustering algorithms depend of the initialization parameters. For some clustering algorithms, the number of clusters must be given a priori, as in the case of k-means. In fact, the clusters number is mostly unknown, given that clustering is generally applied on unlabeled data-sets. To determine the number of clusters, several metrics can be employed, from which we include the well-used metrics in Table 2. However, using these metrics is hindered by several limitations, especially in terms of computation and memory overhead, when dealing with large-scale data-sets. Therefore, there is a need for an efficient clustering initialization solution, which is the main contribution of this paper.

1.2 Motivation and Contributions

Proposing a clustering initialization solution that requires low computation overhead is crucial since the existing initialization techniques are the most computation-consuming functions, especially for frequently updated and/or large data-sets. In this paper, an efficient solution for clustering parameters initialization is presented. The proposed solution consists of training a supervised deep neural network (i.e. Yolo-v5) in order to predict the required initialization parameters of existing clustering algorithms. To the best of our knowledge, we are the first to apply object detection algorithms to find appropriate clusters initialization parameters. The proposed solution can predict the number of clusters and their corresponding centroids, in addition to the size of each cluster. Moreover, two models are presented in this work, one for isolated clusters and the other for overlapping clusters. The properties of the input data set determine which model to use.

To assess our approach, we present extensive experimental results on several clustering algorithms, i.e. k-means , Fuzzy C Means (FCM), with and without pre-computed initialization parameters using the proposed approach. The results show an improvement in both performance and stability of many of the considered clustering algorithms.

Finally, it is worth noting that our approach achieves high generalization since our training is done on data with small number of centroids, while at the testing phase we are able to detect the centroids of data presenting higher number of clusters. Furthermore, this paper presents a good application of transfer learning to solve issues of existing unsupervised machine learning algorithms.

1.3 Organization

The rest of this paper is organized as follows: In Section 2, we list and briefly describe some clustering algorithms, the internal validation metrics used for clusters initialization, in addition to the used object detection DL-based approach. Then, Section 3 presents the proposed detection clustering initialisation parameters. In Section 4, experimental results with real data-sets are presented to validate the efficiency of the proposed solution. Finally, the paper is concluded in Section 5.

2 Background and Preliminaries

This section presents a set of well-known clustering algorithms such as k-means and FCM. For each of them, we present their required initialization parameters. Then, the advantages and limitations of existing well-known clustering algorithms are listed. Afterwards, several internal metrics used for initial cluster numbers detection are presented and described. Let us indicate that the proposed solution is also internal since we consider that we do not have any information about data labels. Finally, the employed YOLO-v5 DL-based object detection is described. Let us indicate that YOLO-v5 is used as proof of concept in the proposed solution and any object detection model can be used if it can ensure high detection accuracy.

In the following, we will use these notations : $X = \{x_1, \ldots, x_n\}$ is a set of n data points in a d-Euclidean space of dimension R^d. These data points can be clustered in m different clusters of centroids $C = \{c_1, \ldots, c_m\}$.

2.1 Clustering Algorithms

Clustering algorithms have been extensively explored in the literature and implemented in a set of substantive areas [1, 6, 7]. Before dwelling into more details, let us recall that all the considered algorithms in this work are based

on the Expectation-Maximisation (EM) algorithm [8]. The EM algorithm is an iterative algorithm, where each iteration consists in two steps: computing the expectation (E-Step) and maximizing it (M-Step). This probabilistic technique is generally used to solve Maximum Likelihood problems. It transforms an optimization problem (minimizing an objective function) into a probabilistic problem solved through this simple yet very powerful technique as is the case with k-means, and FCM.

2.1.1 K-means

In this part, the k-means algorithm, which is one of the most famous and ancient unsupervised learning algorithms [9], is described.

The k-means algorithm consists of the following steps :

1. Set the number of clusters (centers) m

2. Initialize centroids by shuffling the data-points and then randomly selecting m centers without replacement

3. Iterate until the maximum number of allowed iterations is reached or the difference between two successive sets of centers is under some error threshold

   (a) Updating the membership matrix: for each data-point, compute the euclidean distance to each old centroid, find the minimum distance for each data-point, assign that data-point to the centroid with which it has minimum distance (E step)
   (b) Updating the centroids: find the new centroids by computing the average of each newly determined cluster according to the updated membership matrix (M Step)

Each data point x_i, $i = 1, 2, \ldots, n$ is considered as a member of only one cluster k of center c_k, with $k = 1, 2, \ldots, m$, if it has a minimal euclidean distance to c_k compared to other cluster centers. The k-means algorithm minimizes the inertia, which is defined as the sum of the distances between data-points and their centroids, as can be seen in Eq. 1.

The big issue with k-means is that the number of clusters m should be given a priori, but usually, this number is unknown in real applications, where the data is unlabeled. In addition, another limitation of k-means is that its stability is affected by the initialization parameters because the initial centroids are usually randomly chosen from the data-set in the so-called LLoyd's implementation. A more sophisticated approach, called "k-means++", uses a smarter heuristic for setting the initial number of centroids to achieve faster convergence, but the number of clusters should in all cases be known.

The recent x-means algorithm [10], a modified version of k-means, does not necessitate the a priori knowledge of the number of clusters, but instead, it relies on the Bayesian Information Criterion (BIC) measure to define the number of clusters. However, the stability of x-means is still in function of the initialization parameters as k-means. In addition, calculating the BIC measure presents a high computation overhead, especially with large data-sets. By contrast, our proposed solution, based on the object detection concept, detects efficiently the clusters number, possible centroids, and clusters sizes. Let us note that k-means minimizes the objective function defined by Eq. 1, which is simply the sum of the distances from each data point to its corresponding centroid.

$$J_m(X, c) = \sum_{i=1}^{n} \sum_{k=1}^{m} \|x_i - c_k\|^2 \tag{1}$$

2.1.2 Fuzzy C Means

With k-means, each data-point may be a member of at most one cluster. While this might be well-adapted for many applications, in some cases, a data-point might belong to several clusters. The FCM algorithm, first presented by Dunn [11] and later improved by Bezdek [12], leverages the fuzzy algebra to express the simultaneous membership of a data-point to different classes (clusters). The FCM algorithm computes what is called a soft partition of the data-set in contrast to k-means which computes a hard data partition. In order to compute a hard partition with FCM, it is sufficient to consider the maximum membership degree of a data-point as its unique cluster. The FCM algorithm tries to minimize

the objective function given in Eq. 2, where z is called the fuzziness parameter initialized to a value between 2 and 3, and $\|.\|_A^2$ stands for any mathematical distance.

$$J_z(X, c) = \sum_{i=1}^{n} \sum_{k=1}^{m} (u_{ki})^z \|x_i - c_k\|_A^2 \qquad (2)$$

The FCM algorithm consists of the following steps :

1. Randomly initialize the membership matrix U such that $\sum_{k=1}^{m} u_{ki} = 1$ for any data-point x_i

2. Update the centroids using the membership matrix U (E-Step)

3. Update the membership matrix U using the last computed centroids in the previous step (M-Step)

4. Stop if $\|U^{t+1} - U_t\|_A^2 < \epsilon$ or maximum number of allowed iterations is reached, otherwise make another iteration of steps 2 and 3.

The FCM algorithm needs only the set of initial centroids but it must be fed with a proper membership matrix. A workaround may consist of running one extra "M-Step", after feeding the algorithm with a set of centroids. This helps to compute a proper membership matrix, that is consistent with the input centroids. Just like k-means, the FCM algorithm stability is very dependent on the initial membership matrix, which is randomly chosen. Our approach promise is to handle this limitation very efficiently since the initial U matrix will be tightly linked to the detected centroids.

Table 1 presents a set of well-known clustering algorithms with their advantages and disadvantages. Their main issue, as indicated in this table, is that they require information about the number of clusters or the clusters sizes.

Table 1: Overview of the advantages and disadvantages of commonly used clustering algorithms

Clustering Algorithm	Advantages	Disadvantages
k-means	• This algorithm is simple and has a linear relative computational complexity.	• The user must specify the number of clusters to be used. • With a cluster of irregular maps, output is poor performance.
Mean Shift	• The number of clusters or classes is not required.	• The user must specify the window size.
DBSCAN	• The number of clusters or classes is not required. • It can detect outliers and irregularly shaped clusters.	• The user must specify the window size. • For different clusters densities, the performance is reduced.
Expectation-Maximization	• It is capable of detecting clusters with ellipsoidal forms. • Each point is given a membership probability.	• The user must specify the number of clusters beforehand.

To fix this challenge, several metrics (internal or external: with labels) that can be used to detect cluster numbers are presented in Table 2. However, these methods suffer to deal with big data-set and they are described in the next.

Table 2: Existing metrics to detect clusters number for the k-means clustering algorithm

Reference	Metric	Description
[13]	Bayesian Information Criterion (BIC)	BIC is a criterion for measuring and selecting models. It relies on the principles of Bayesian inference and probability. The model complexity is penalized by BIC, so more complex models would have a lower score and therefore be less likely to be chosen.
[14]	Akaike Information Criterion (AIC)	It is suitable for models that fit into the maximum likelihood estimation system, like BIC. The lower are the AIC and BIC, the better is the clustering performance.
[11]	Dunn's index (DN)	DN defines sets of clusters that are compact, with a very small variation between cluster members, and large separation between clusters. The higher is the value of Dunn's index, the better is the clustering performance. The optimal amount of clusters is the number of clusters that maximises the Dunn's index.
[15]	Davies-Bouldin index (DB)	It calculates the average similarity between each cluster and its most similar one. The DB validity index aims to maximize the distances between clusters while minimizing distances between the cluster centroid and its data objects.
[16]	Silhouette Width (SW)	It is a statistic that measures how similar an object is to its own cluster versus other clusters. The silhouette value ranges from -1 to 1. A high silhouette value is well suited to its own cluster but poorly related to neighboring clusters. Positive and negative high silhouette widths (SW) indicate the objects that are correctly clustered and those that are incorrectly clustered, respectively. It is well known that objects with a SW validity index of zero or less are difficult to be clustered.
[17]	Calinski and Harabasz index (CH)	This metric is the ratio of the sum of between-cluster dispersion and inter-cluster dispersion for all clusters. It is known also as the Variance Ratio Criterion. The higher is this score, the better is the clustering performance.
[18]	Gap statistic	It is a statistical hypothesis test-based cluster validity measure. At each value of the cluster number, the gap statistic compares the variation in within-cluster dispersion to that predicted under an appropriate reference null distribution. The smallest is the number of clusters, the best is the clustering performance.

2.2 Validation Metrics

In this part, we present the state-of-the-art methods for clusters number estimation. To find the appropriate number of clusters k, several validity metrics were presented and they can be divided into two main classes [19]: external and internal.

1. External metrics (data labels) are employed to evaluate a clustering method performance by comparing the obtained cluster memberships with their initial class labels.

2. Internal metrics are used to evaluate the correctness of the obtained clusters structure by focusing on the intrinsic information of the data itself.

In general, the most advanced methods of estimating the cluster numbers can be described as follows.:

1. For $k = 1, 2, \ldots, k_{max}$, recognize k clusters in the specified data-set using the clustering algorithm.

2. Apply the cluster number estimation method to all of the output clusters, for $k = 1, 2, \ldots, k_{max}$.

3. To estimate the number of clusters, use the appropriate selection criteria (max, min, etc.).

2.3 YOLO Deep Learning Object Detection Model

Object detection is among the classical computer vision problems to identify which objects are in the image and their corresponding locations. The object detection issue is more complex than the classification problem that consists of recognizing objects but without indicating their locations in the image. Moreover, images containing more than one object cannot be classified.

In this paper, we use YOLO-v5, a recent update of the YOLO family. YOLO was the first object recognition model to merge bounding box estimation and object identification in one end-to-end differentiable network. Darknet is the environment under which YOLO is written and maintained. In comparison to previous YOLO models, YOLO-v5 is the first YOLO model developed with the PyTorch framework, and it is more lightweight and simple to use compared to previous YOLO variants.

YOLO-v5 is based on a smart Convolutional Neural Network (CNN) for real-time object detection. This algorithm divides the image into regions and calculates the bounding boxes and probabilities for each region. The predicted probabilities are used to weight these bounding boxes. The algorithm needs only one forward propagation pass through the neural network to make predictions, so it "only looks once" at the image. It then outputs known objects together with the bounding boxes after a non-max suppression (which ensures that the object detection algorithm only recognizes each object once).

3   Proposed Model

The structure of the proposed solution, illustrated in Figure 1, uses YOLO-v5 as a learner to detect clusters (as objects) with their centroids and sizes. The YOLO-v5 DL-based object detection method is used in this paper to construct a model that can make the clustering algorithm free of initialization. This model finds automatically the appropriate clustering initialization parameters such as the number of clusters, possible centroids and clusters sizes. In the following, we present the implementation details of the proposed solution.
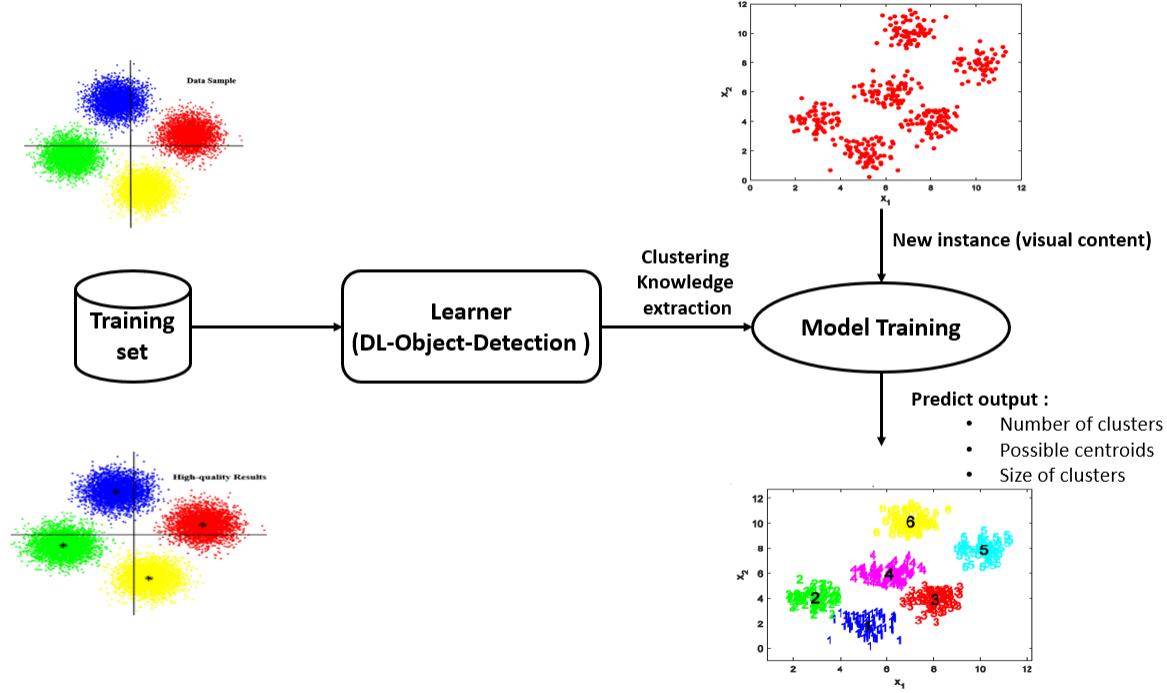

Figure 1: The proposed solution consists of training the YOLO-V5 model to detect clusters initialization parameters such as the number of clusters and centroids

3.1   Model Description

The input of the proposed model is a transformed data in 2D feature space. The proposed solution assumes that a high-dimensional data-set can be represented in a lower dimension space, where an inter-cluster pairwise distance is preserved among clusters. Given a transformed data-set of n points $X = x_1, \ldots, x_n$, where $x_i \in R^2$, the clustering algorithms, such as k-means, partition the data-set into k clusters $C_1, C_2, \ldots, C_k$, where each cluster C_j is represented by a cluster center c_j. Let k denotes the correct number of clusters in a data-set. To build accurate clustering models

that can achieve high clustering accuracy, k and other clustering parameters (e.g. centroids and clusters sizes) are required as input. Getting a good estimate of these parameters and especially k is not a straightforward task. Several metrics were presented previously to determine the number of clusters. However, computing these metrics requires a high computation and resources overhead.

The input of the proposed solution is an image representing the clustered data. Thus, with large data-set, the computation overhead of the proposed solution will remain the same, contrarily to the existing techniques that require an exponential computation complexity in function of the data size, as shown in Figure 10.

3.2 Data Description

The YOLO-v5 model was trained using a generated data-set. This data-set consists of a large set of images generated with random configuration, as illustrated in Figure 3. Each image contains a number of clusters (between 2 and 12) of random size, having each a number of samples between 20000 and 50000 points. Moreover, the form of clusters can be of one of the following types: Gaussian Mixture Model, noisy moon, blob, no structure, etc.

The obtained training images have a size equal to 640×640. These images are processed by scaling them between $[0;\ 1]$. The testing images consist also of a set of generated clusters with random configuration. The importance of the proposed solution is that it is flexible (can add new cluster forms) and takes into account different conditions (dense or not). Let us indicate that the generated training/testing images are pre-processed to obtain the required labels files that are considered as input to YOLO-v5.

In total, the data-set contains 1000 images. Let us indicate that the generated clusters with random configurations can have equal or different number of points. Also, the obtained clusters can be well-separated or very close, in addition to having equal or variable variances.

In Figure 2, we present several examples of training images generated from the 2-variate with variable mixture components (2 dimensions with different clusters number).
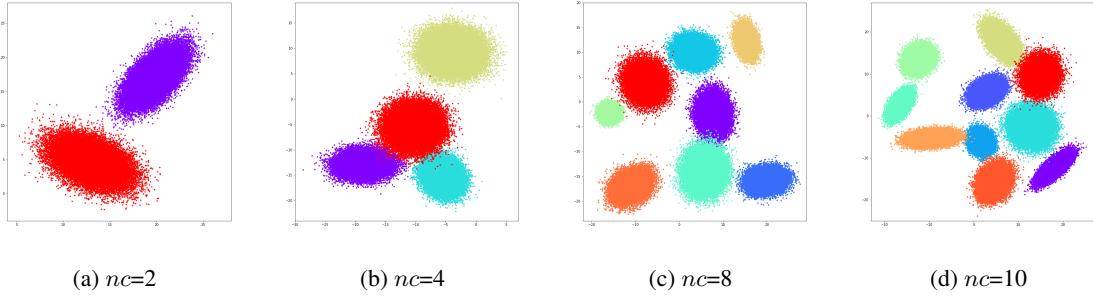

(a) nc=2   (b) nc=4   (c) nc=8   (d) nc=10

Figure 2: Examples of generated data-sets used to train YOLO-v5 object detection model

3.3 Model Implementation

The YOLO-v5 model implementation of [20] was used to build our proposed learner model. To train this model, the SGD optimizer was used with an initial value of 10^{-2} and a batch of size 16. All the other parameters are the standard parameters of the Yolo-V5 code.

In the training phase, the Yolo-V5 model learns to recognize all the clusters with 20 epochs. At the inference phase, once the model detects the bounding boxes, the center for each cluster is computed and this value will be the initial value for the initial cluster. As Yolo-V5 being very efficient and lightweight, it can detect quickly objects (clusters) and can be implemented on GPU or CPU (after being trained).
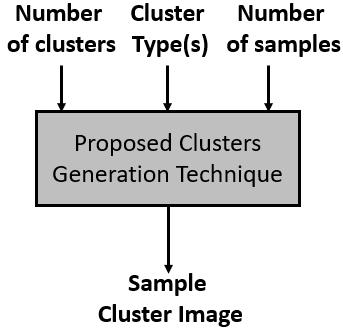
Figure 3: Proposed clusters generation method

4 Performance Analysis

In this section, we present several performance and robustness tests that were applied on 100 testing iterations (images) to demonstrate the efficiency and robustness of the proposed solution. We have trained the object detection model (supervised learning) to obtain the initial clustering parameters such as the clusters number k, centroids and clusters sizes.

Thus, we will first compare our obtained results with the results obtained when applying the listed metrics in Section 2.2 to validate the robustness and efficiency of the proposed solution. Furthermore, the proposed solution was applied with different clustering algorithms: k-means [21], x-means [22], and FCM [23] with or without initialization parameters. For each test iteration, the clusters image consists of a number of clusters varying between 2 to 12, with random positions, directions, and sizes. We choose the random configuration of the clusters generation to check if that affects the performance of the proposed method. Furthermore, all clusters in the training and testing phases have random number of points, variance and also clusters separation distance.

4.1 Testing Methodology: Notations and Settings

In this subsection, the testing methodology and the various parameters that might affect the effectiveness of the proposed approach are presented.

When testing the proposed method, two possible cases should be considered:

1. The case of overlapping clusters data, where the data-set contains overlapping clusters.
2. The case of of separated clusters data, where the data-set contains non-overlapping or separated clusters.

overlapping separated clusters Therefore, two different trained models should be built, and the selection of which model to be used depends on the input data-set. Ideally, data clustering aims to satisfy two conditions, which are:

- Compactness: where the cluster around each center is dense.
- Separation: where different clusters are far apart.

When these conditions are met, the first trained model can be used and a set of experimental results that validate the effectiveness and the robustness of this model will be presented in the new section (see Figure 7).

Moreover, for the data-set that cannot ensure these conditions (see Figure 8), we use the second trained model that can combine detected clusters according to a threshold of common points. The appropriate threshold depends on the data-set structure and can be detected by brute force.

Therefore, the data-set is analyzed firstly to determine if clusters are overlapping or not. Based on the result, we select the appropriate trained model to detect its corresponding initialization parameters.

The unsupervised algorithms that are studied have different initialization parameters that should be configured carefully in the experiments. Thus, two versions of k-means were used. The first one is the naive k-means, that was tested with: the EM approach using random initial centroids, and the proposed solution. The second k-means implementation is the scikit-learn implementation, that was tested with: the k-means++ algorithm to choose the initial centroids, and the proposed solution.

4.2 Clusters Number Correctness

In this part, we compare the performance of the existing clusters number estimation methods with our proposed solution. First, we use the YOLO-v5 trained model to detect the number of clusters. Figure 4 presents the percentage of correctly predicted number of clusters for 100 test iterations, where clusters for each testing image (test iteration) have been generated randomly with different number of samples, sizes and locations.

The results indicate clearly that the proposed approach can detect correctly the clusters number. The clusters number detection rate with other validity metrics such as CH, SW, DB, Gap statistic, are shown in Figure 4. Certain metrics present high accuracy in detecting the number of clusters, similarly to our proposed solution. However, the advantage of our proposed solution compared to existing internal metrics is that the proposed approach requires less computation and memory overhead. Moreover, the proposed solution can work with partial data (a part of the whole clusters identified randomly, e.g. 20% of samples). This leads to reduce more and more the required latency and resource overhead without degrading the accuracy of the clusters number estimation.
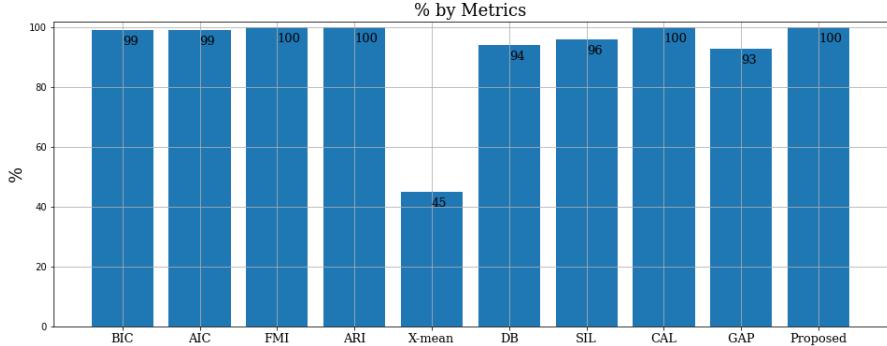

Figure 4: Percentage of correctly detected number of clusters for 100 tests iterations, where clusters for each test iteration have been generated randomly with different clusters number, sizes and locations

4.3 Centroids Detection Correctness

In this part, we analyze the correctness of the detected centroids. Figure 5 presents a set of random generated testing images, where each one consists of a set of clusters produced using random configurations (centroids, sizes, and locations). We can remark visually that the identified centroids are very close to the generated ones for the different clusters (see Figure 5). Moreover, we measure the euclidean distance between identified and generated centroids. The obtained results, presented in Figure 6, indicate clearly that the generated and detected centroids are very close given that overall the obtained distances are less than one. Let us indicate that the euclidean distance between two points p and q in the d space can be computed according to the following equation:

$$d\left(p,q\right) = \sqrt{\sum_{i=1}^{d}\left(q_i - p_i\right)^2} \tag{3}$$

The lower is $d\left(p,q\right)$, the closer are the points p and q. When d is equal to 0, p and q are collocated.

These results show that the proposed method provides better approximation of the correct centroids and consequently this will result in increasing the accuracy of the clustering algorithm, in addition to reducing the memory consumption and the number of iterations for the clustering algorithms to converge.

9

(a) *k*=2   (b) *k*=3   (c) *k*=4   (d) *k*=5

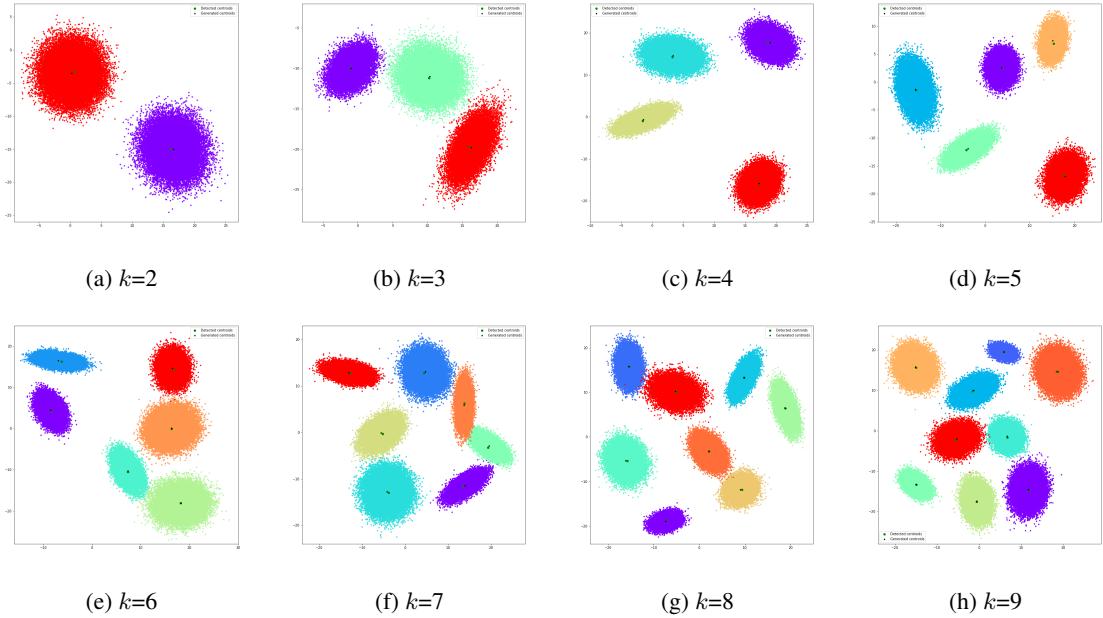(e) *k*=6   (f) *k*=7   (g) *k*=8   (h) *k*=9

Figure 5: An example of validation data-sets with generated and identified centroids (consequently of each cluster)
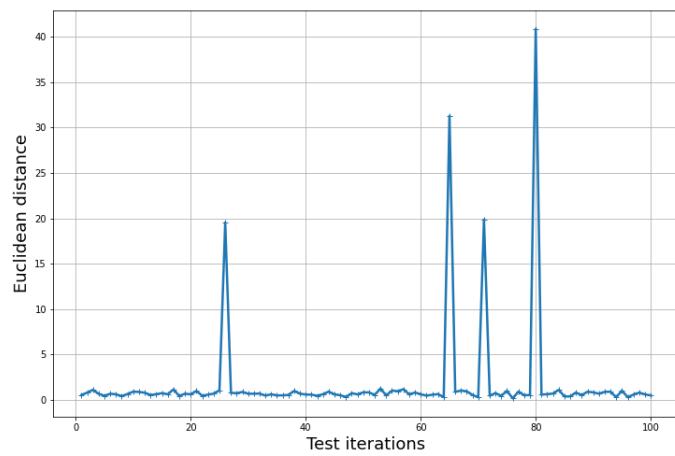

Figure 6: Euclidean distance between generated and detected centroids for 100 test iterations: each test iteration consists of a set of clusters that have generated with random parameters such number of samples, size and location

4.4 Accuracy Rate

In this part, the accuracy of the clustering algorithms is analyzed by using the proposed detected initialized parameters for k-means and other clustering algorithms.

The clustering accuracy rate (AR) can be computed as follows:

$$AR = \sum_{i=1}^{k} \frac{n(c_k)}{n} \tag{4}$$

where $n(c_k)$ is the number of data points that were correctly included in cluster k, and n is the total number of data points. The higher is the AR, the better is the clustering correctness.

Figure 9 represents the AR results of k-means with and without the identified centroids. These results validate that the clustering accuracy (at the samples level) is enhanced (to be close to 1) by using the identified centroids for a naive implementation of k-means and also for the optimized scikit-learn k-means implementation. Therefore, k-means (or any other clustering algorithm) can provide better clustering accuracy and low computational overhead (low number of iteration to converge and consequently delay) by using the proposed initialization parameters detection method. For example, as shown Figure 7-e (detected nc =6), the accuracy is enhanced from 0.854 to 0.999 by using the identified clusters number and centroids.

| (a) nc=2 | (b) nc=3 | (c) nc=4 | (d) nc=5 |

| (e) nc=6 | (f) nc=7 | (g) nc=8 | (h) nc=9 |

Figure 7: The k-means clustering results for the testing data-sets, listed in Figure 5, by using the detected cluster number and identified centroids (proposed method)
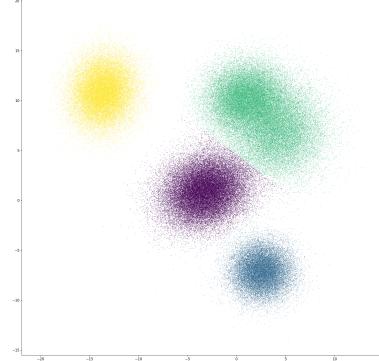
On the other hand, in case of $nc = 7$ (Figure 7-f), the required number of iterations to converge is decreased from 4 to 2 by using the identified centroids. Moreover, Figure 11 confirms this reduction since the required number of iterations to converge is reduced for k-means and FCM when using the proposed approach compared to the traditional clustering approaches (for 100 test images).
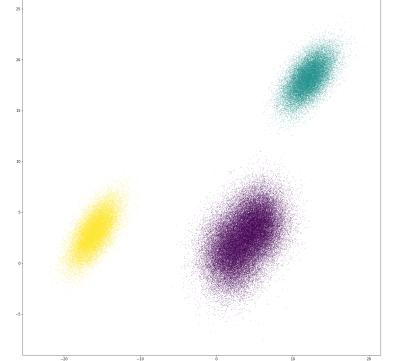
4.5 Execution Time

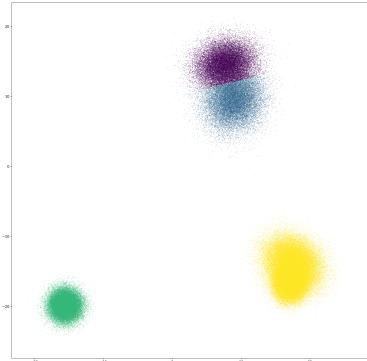In general, the time delay of any clustering algorithm depends on three factors:
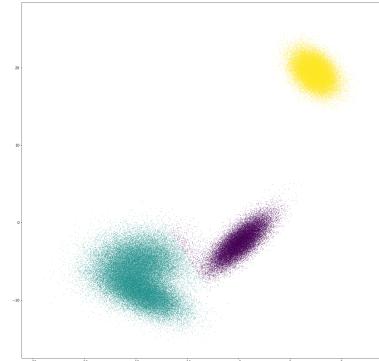
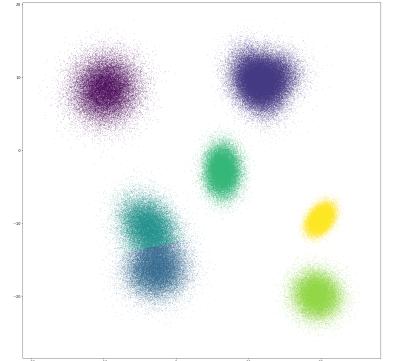(a) Detected nc=5, BIC=6, AIC=7

(b) Detected nc=4, BIC=5, AIC=5

(c) Detected nc=3, BIC=4, AIC=4
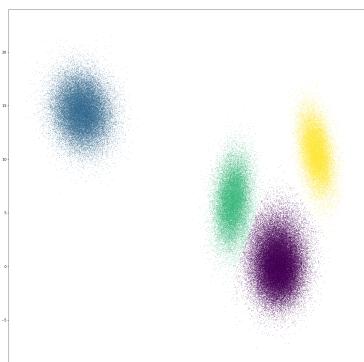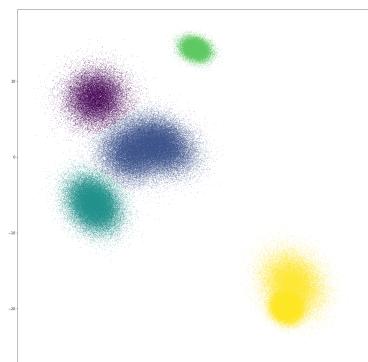
(d) Detected nc=4, BIC=6, AIC=8

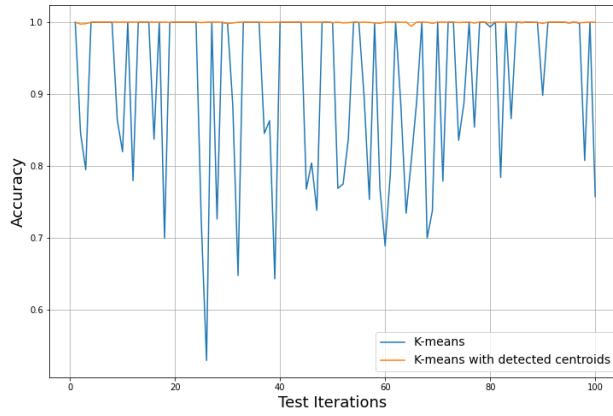(e) Detected nc=3, BIC=4 AIC=4

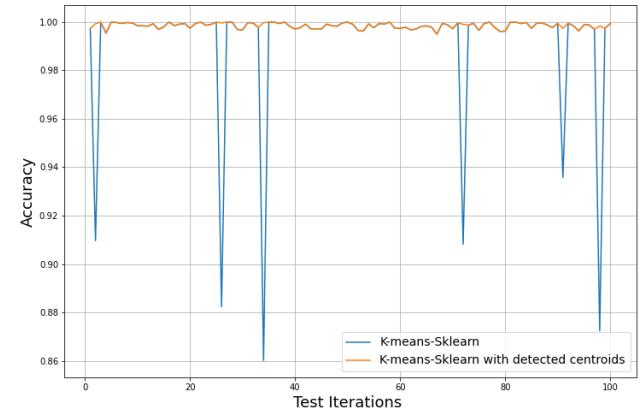(f) Detected nc=7, BIC=8, AIC=8

(g) Detected nc=4, BIC=5, AIC=5

(h) Detected nc=5, BIC=8, AIC=8

Figure 8: Results of the overlapping model for which the number of centroids detected by the proposed approach is different compared to other classical approaches (AIC, and BIC)

(a) Naive implementation   (b) Scikit-learn implementation

Figure 9: Accuracy of k-means clustering algorithms with and without using the proposed approach for two k-means implementations

1. The number of the algorithm iterations
2. The amount of data points
3. The time needed to find out the clustering centers and data points partitions
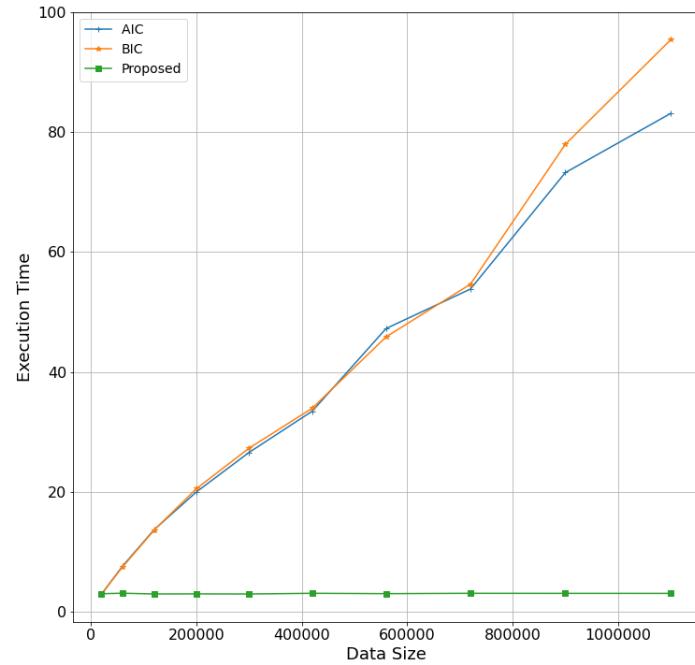

Figure 10: Variation of execution time (in seconds) to detect correct number of clusters by using the proposed approach, AIC, and BIC

First of all, we computed the testing time of the proposed solution to detect cluster initialization parameters. Figure 10 shows the execution times (in seconds) of two well-known internal indices methods (AIC and BIC) and the proposed method versus data sizes. We observe lower execution time for the proposed solution compared to the existing indices methods, making it the best choice when working on large data-sets. The proposed solution has the advantage of requiring very low computation complexity and consequently testing delay since it is independent of the number of data points. However, with the increase in the data-set size, the running time for all internal metrics is significantly increased.

(a) Naive K-means implementation   (b) Scikit-learn K-means implementation   (c) FCM
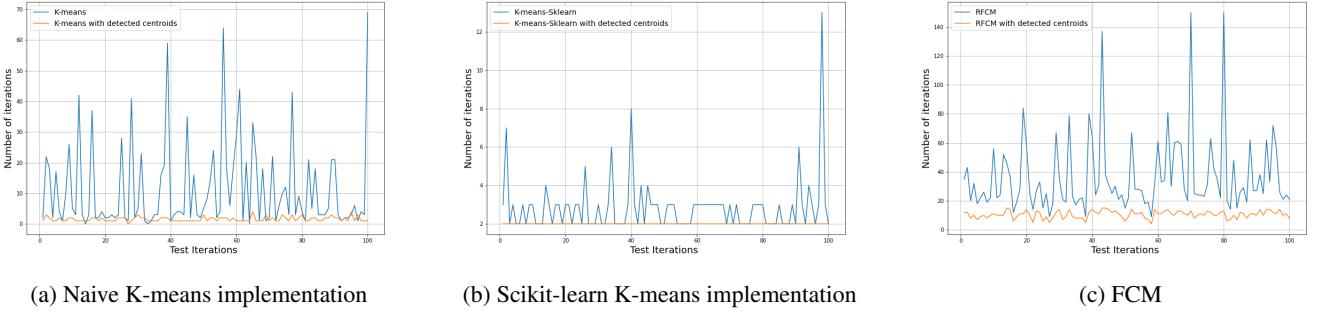
Figure 11: Variation of number iterations required to converge for k-means(a), optimized implementation with scikit-learn(b) and for FCM(c) in function of test iterations
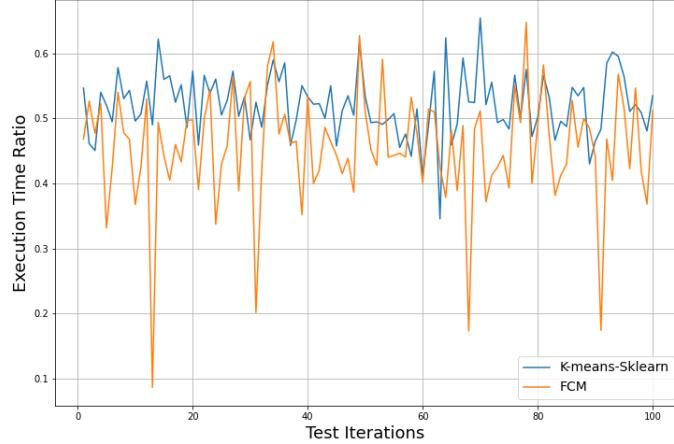

Figure 12: Ratio of the execution time for k-means and FCM between with and without the detected clusters initialization

Moreover, we quantify the effectiveness of using these initialization parameters to help clustering algorithm converging fast. Figure 12 shows the execution time ratio between using clustering algorithms with identified centroids and without. These results show clearly that using the identified centroids reduces the testing time to half in average. This indicates that the proposed solution reduces the clustering testing time and makes it suitable for real time applications.

4.6   Flexibility

Another important property, that the proposed solution can provide, is the flexibility to deal with different clusters forms. The proposed solution can be extended by adding existing cluster formats such as: blobs, blobs with varied variances, noisy circles, noisy moons, no structure, and Anisotropicly distributed data, which are described and presented in [24].

14

5 Conclusion

This paper proposes a new clustering initialization method that can determine the number of clusters in addition to their possible centroids and sizes. The proposed solution uses a DL-based object detection model (YOLO-v5) to detect the initial clustering parameters. The advantages of the proposed solution is that it is: lightweight, fast, and robust with different cluster volumes, shapes, and noise. The proposed solution has been realized with several configurations, demonstrating its efficiency compared to existing approaches, especially in terms of time complexity and resources overhead. Two models are presented. The first one is for the separated clusters and the second one is for overlapping clusters. The choice of which model to use is determined based on the characteristics of the input dataset. As future work, we aim to design a new DL-based data transformation model towards getting well-separated clusters. As well, we aim to work with 3D object detection models to be applied on 3D clusters representations, in addition to consider other cluster forms.

Acknowledgment

References

[1] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[2] Christophe Guyeux, Stéphane Chrétien, Gaby Bou Tayeh, Jacques Demerjian, and Jacques Bahi. Introducing and comparing recent clustering methods for massive data management in the internet of things. *Journal of Sensor and Actuator Networks*, 8(4):56 (25), dec 2019.

[3] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE Access*, 8:80716–80727, 2020.

[4] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y Zomaya, Sebti Foufou, and Abdelaziz Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing*, 2(3):267–279, 2014.

[5] Maria Camila N Barioni, Humberto Razente, Alessandra MR Marcelino, Agma JM Traina, and Caetano Traina Jr. Open issues for partitioning clustering methods: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3):161–177, 2014.

[6] Mohammad Alhawarat and M Hegazi. Revisiting k-means and topic modeling, a comparison study to cluster arabic documents. *IEEE Access*, 6:42740–42749, 2018.

[7] Yinfeng Meng, Jiye Liang, Fuyuan Cao, and Yijun He. A new distance with derivative information for functional k-means clustering algorithm. *Information Sciences*, 463:166–185, 2018.

[8] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society : Series B*, 39(1):1–38, 1977.

[9] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. Berkeley, University of California Press.

[10] Dan Pelleg and Andrew Moore. X-means: Extending k-means with e cient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, page 727. Citeseer.

[11] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[12] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer US, 1981.

[13] Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995.

[14] Hamparsum Bozdogan. Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, 1987.

[15] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[16] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[17] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[18] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[19] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi, and Elvia M Quiroz. Internal versus external cluster validation indexes. *International Journal of computers and communications*, 5(1):27–34, 2011.

[20] Glenn Jocher. Yolov5 in pytorch. `https://github.com/ultralytics/yolov5`, 06 2020. (undefined 28/4/2021 10:16).

[21] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[22] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734, 2000.

[23] Pradipta Maji and Sankar K Pal. Rfcm: a hybrid clustering algorithm using rough and fuzzy sets. *Fundamenta Informaticae*, 80(4):475–496, 2007.

[24] Comparing different clustering algorithms on toy datasets scikitlearn 0241 documentation. `https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html`, 05 2012.