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

A deep learning object detection method to improve cluster analysis of two-dimensional data

**Raphaël Couturier*, Pablo Gregori,
Hassan Noura, Ola Salman and
Abderrahmane Sider**

the date of receipt and acceptance should be inserted later

Abstract Clustering is an unsupervised machine learning method grouping data samples into clusters of similar objects, used as a system support tool in numerous applications such as banking customers profiling, document retrieval, image segmentation, and e-commerce recommendation engines. The effectiveness of several clustering techniques is sensible to the initialization parameters, and different solutions have been proposed in the literature to overcome this limitation. They require high computational memory consumption when dealing with big data. In this paper, we propose the application of a recent object detection Deep Learning model (YOLO-v5) for assisting the initialization of classical techniques and improving their effectiveness on two-variate datasets, leveraging the accuracy and reducing dramatically the memory and time consumption of classical clustering methods.

Keywords Clustering algorithms; Clustering initialization methods; Clustering initialization metrics; Deep Learning object detection model

1 Introduction

Clustering is an efficient solution to split the observations in a dataset into groups, that are characterized by a high similarity of observations within each

R. Couturier (* corresponding author, email: raphael.couturier@univ-fcomte.fr), H. Noura
University of Franche-Comte (UFC), FEMTO-ST Institute, CNRS, France

P. Gregori
Instituto de Matemáticas y Aplicaciones de Castellón, Departamento de Matemáticas, Universitat Jaume I de Castellón, E-12071 Castellón, Spain

O. Salman
American University of Beirut, Electrical and Computer Engineering Department, Lebanon

A. Sider
LIMED Laboratory, Faculty of Exact Sciences, University of Bejaia, 06000 Bejaia, Algeria

group and a high distance between different groups [1,2]. Mainly, the clustering methods are unsupervised machine learning methods that can be parametric [3] (i.e. probability-based) or non-parametric [4], and they serve as support tools for systems (see e.g. [5] for MRI images). The non-parametric clustering methods are often based on an empirical function that measures the similarity or dissimilarity between the data points [6].

In addition, the clustering methods can be classified into partitional and hierarchical ones. We shall deal with the former type, where each observation is initially assigned to a unique cluster, and they are rearranged according to an optimality criterium [7].

Partitional clustering algorithms suffer from several challenges, concerning their reliability and efficiency. The main challenge is that the stability and convergence of the clustering algorithms depend on the initialization parameters. In addition, they are generally applied to unlabeled datasets, where the number of clusters is unknown. However, some clustering methods (as k-means) require the choice of a given number of clusters. In order to estimate the most likely number of clusters, several metrics can be employed (e.g., see Table 1), some of them being computationally expensive when dealing with large-scale datasets.

We propose an efficient solution to estimate the number of clusters and their initialization parameters. It consists of training a supervised object detection neural network in order to (1) detect the number of clusters, and (2) optimize the initialization parameters of the current clustering algorithms. Without it, the efficiency of the classical algorithms is compromised, since it depends on fortunate initial guesses for the parameters, that are unlikely in high dimensional problems.

Our approach is assessed through extensive experimental results with methods such as k-means (in two implementations) and Fuzzy c-Means (FCM). The results show an improvement in both performance and stability of the considered clustering algorithms.

The rest of the paper is organized as follows: in Section 2, we briefly describe the selected clustering algorithms, the internal validation metrics, and the object detection Deep Learning (DL) tool. Then, Section 4 presents the new methodology that we propose for the optimization of the clustering problem. In Section 5, experimental results with real datasets are presented to validate the efficiency of the proposed solution. Finally, conclusions are depicted in Section 6.

2 Related Works

In the sequel $X = \{x_1, x_2, \ldots, x_n\}$ denotes a set of n data points in $\mathbb{R}^d$, the Euclidean space of dimension d. These data points can be clustered in m different clusters with centroids denoted by $C = \{c_1, c_2, \ldots, c_m\}$.

2.1 Clustering Algorithms

Clustering algorithms have been extensively explored in the literature and implemented in a set of substantive areas [1,8,9]. We shall consider algorithms based on the Expectation-Maximisation (EM) algorithm [10]. The EM algorithm is an iterative algorithm, where each iteration consists of two steps: computing the expectation (E-Step) and maximizing it (M-Step). This probabilistic technique is generally used to solve Maximum Likelihood problems.

2.1.1 K-means

The k-means algorithm [11] tries to find the optimal m (given by the user) clusters in a dataset through the following steps:

1. Initialize m centroids by randomly selecting m data-points without replacement.
2. Iterate the following steps, until a maximum number of steps is reached, or until the difference between two successive sets of centers is below some error threshold.
   (a) Update of the cluster membership: for each data-point, compute the Euclidean distance to each one of the present centroids, and assign that data-point (only) to the closest centroid (E step).
   (b) Update the centroids: compute the new centroids as the average of the data-points within the corresponding updated cluster (M step).

The k-means algorithm tries to minimize the inertia, which is defined as the sum of the squared distances between data-points and their centroids. We can express it by the Equation

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

where u_{ki} is a membership matrix (with value 1 only if the data-point x_i belongs to the cluster with center c_k, and null otherwise).

The initial centroids are usually chosen at random from the dataset (Lloyd's implementation [12]). k-means++ [13] is a smarter heuristic for setting the initial centroids to achieve faster convergence. In any case, the number of clusters must be given beforehand, and the attainment of the minimum inertia is not ensured, as the algorithm might stick to a local minimum.

The x-means algorithm [14] is a modified version of k-means, which does not require a given number of clusters: it relies on the Bayesian Information Criterion (BIC) in order to decide the number of clusters. However, the stability of x-means still depends on its initialization. Let us mention that the computation of the BIC measure presents a high computation overhead with large datasets.

The Fuzzy c-Means (FCM) algorithm, first presented by Dunn [15] and later improved by Bezdek [16], leverages the fuzzy algebra to express the simultaneous membership of a data-point to different clusters. It computes a *soft* partition of the dataset. In order to get the partition with FCM, it is sufficient to consider that each data-point shall belong to the cluster with maximum membership degree. The FCM algorithm tries to minimize the inertia, given by the equation

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

where z is called the fuzziness parameter, initialized to a value between 2 and 3, and $\|\cdot\|_A^2$ stands for any mathematical distance. It entails the following steps:

1. Initialize a random membership matrix $U^0 = (u_{ki})$ such that $\sum_{k=1}^{m} u_{ki} = 1$ for any data-point x_i, $i = 1, 2, \ldots, n$.
2. Iterate the following steps, until a maximum number of steps is reached, or until the difference between two successive matrices is below some error threshold ($\|U^{t+1} - U^t\|^2 < \epsilon$):
   (a) Update the centroids using the membership matrix U^t (E-Step).
   (b) Update the membership matrix U^{t+1} using the computed centroids in the previous step (M-Step).

Just like k-means, the FCM algorithm is very dependent on the initial membership matrix, which is randomly chosen. Our approach manages to handle this limitation very efficiently since the initial U^0 matrix will be tightly linked to the detected centroids.

2.2 Validation Metrics

When the dataset contains labels specifying the group of each observation, the true labels can be used to validate the performance of the clustering method. Otherwise, there are several different approaches for the best estimation of the number of clusters [25], and we list the most widely used ones in Table 1.

In general, the use of these metrics implies the execution of the clustering methods for a sequence of values (of the number of clusters), and the selection of the most likely value under an optimization criterium.

3 Proposed Model

Object detection is among the classical computer vision problems. It aims to identify which objects are in the image and their corresponding locations. The object detection issue is more complex than the classification problem, which consists of recognizing objects but without indicating their locations in the image.

Table 1: Several metrics to estimate the number of clusters for the k-means clustering algorithm

Reference	Metric	Description
[17]	Adjusted Rand Index (ARI)	It measures the similarity between the cluster assignments by making pair-wise comparisons. A higher score signifies greater similarity.
[18]	Bayesian Information Criterion (BIC)	BIC is a criterion for measuring and selecting models. It relies on the principles of Bayesian inference and probability. The complexity of the model is penalized by the BIC, so more complex models would have a lower score and therefore be less likely to be chosen.
[19]	Fowlkes-Mallows Index (FMI)	FMI performs the external evaluation using labels that are already known. Scores use pairwise precision and recall to assess how correctly cluster assignments were performed. The score is defined as the geometric mean of precision and recall. Higher similarity is indicated by a higher score.
[20]	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.
[15]	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.
[21]	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.
[22]	Silhouette Width (SIL)	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 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.
[23]	Calinski and Harabasz index (CAL)	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.
[24]	Gap statistic (GAP)	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.

In this paper, we use YOLO-v5 [26], a recent update of the YOLO family, the first object recognition model to merge bounding box estimation and object identification in one end-to-end differentiable network. In comparison to previous YOLO models, YOLO-v5 is the first one developed with the `PyTorch` framework [27,28], 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

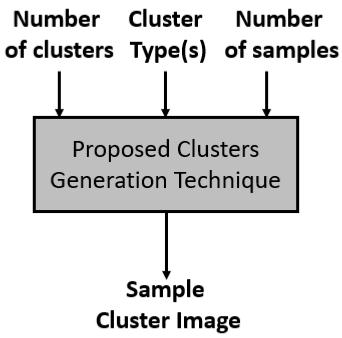


Fig. 1: Proposed Technique to generate label dataset that will be used to training the YOLO-v5 model

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).

4 Experimental Procedure

Let us assume that we want to apply a cluster analysis to 2D datasets $X = \{x_1, \dots, x_n\}$, where $x_i \in \mathbb{R}^2$. In order to apply our model, we need to transform such datasets into images. We have chosen squares of resolution 640×640 pixels.

The YOLO-v5 learner has been trained with 1000 of such datasets, each one presenting between 2 and 12 clusters, made of between 20000 and 50000 points, generated from different bivariate Gaussian densities (as sketched in Fig. 1). We point out that the flexibility of this model would be enhanced if the training phase would cover a larger family of cluster shapes.

Fig. 2 presents the workflow of the proposed solution that consists of:

1. The labeled dataset will be processed to convert samples to image representations in 2D.
2. Split the dataset into training and testing sets
3. Training the YOLO-v5 DL-object detection model by using the training set. After fitting this model, the trained model can be used to predict the cluster parameters (number of clusters and their corresponding centroid).
4. The trained model will be evaluated using the test set, and if it reaches good performance, the trained model can be used with new datasets to quickly detect the cluster initialization parameters. Otherwise, the retraining process will be applied after modifying the DL parameters.

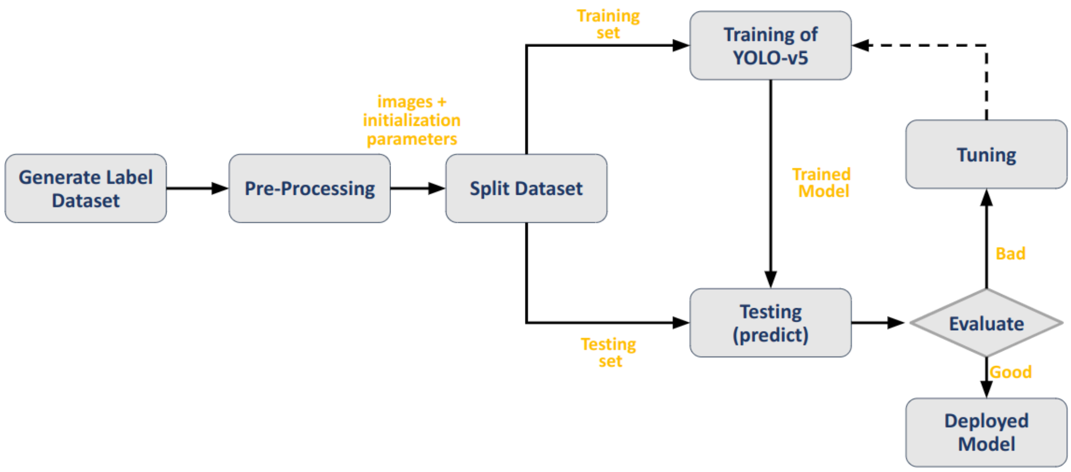


Fig. 2: The workflow process of the proposed solution

We show in Fig. 3, a visual show of the proposed scheme with a random training image (correspondent to dataset samples in 2D image representation) and another one for testing: after the training step, for each new test dataset, the trained learner detects its clusters, and it feeds the clustering method (k-means, FCM , and x-means) with the initial parameters, improving their efficiency.

We present a few examples of the training images in Fig. 4.

We have used the YOLO-v5 model implementation of [26], trained with the SGD optimizer by using an initial value of 10^{-2}, a batch of the size of 16, and the rest of the parameters with the default values.

In the training phase, the YOLO-v5 model learns to detect 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. YOLO-v5 is very efficient and lightweight; it quickly detects the objects (clusters) and can be implemented on GPU or CPU (after being trained).

5 Results

In this section, we compare the performance of k-means [29], x-means [30], and FCM [31] under their usual initialization, against our DL-based initialization, over a set of 100 test datasets (i.e. images).

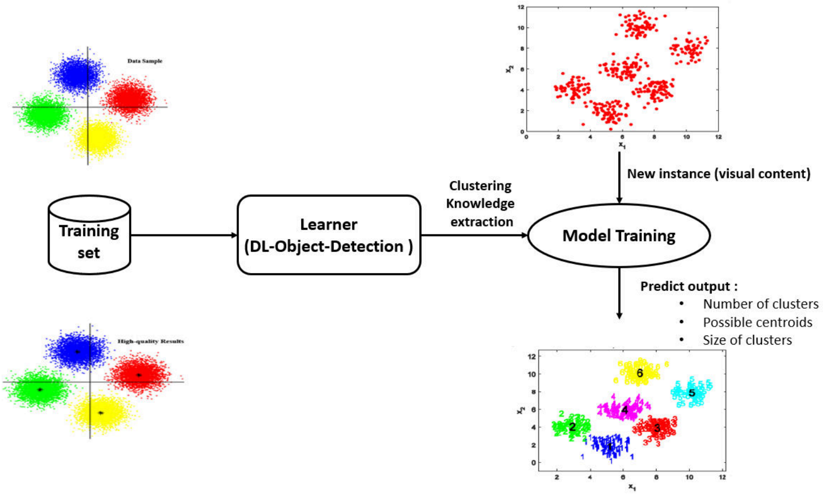
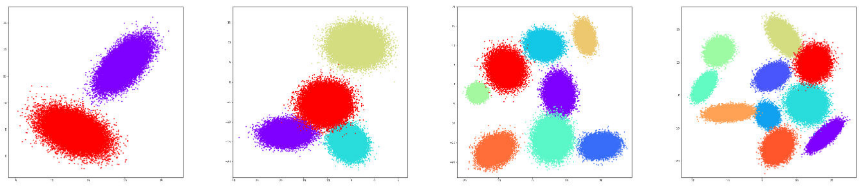


Fig. 3: The proposed solution consists of training the YOLO-v5 model to detect the number of clusters and centroids, and later the trained model will be used to extract initialization parameters (the number of clusters and centroids) for new unknown clustering datasets



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

Fig. 4: Examples of simulated datasets used to train the YOLO-v5 object detection model

174 5.1 Detection of the number of clusters

Our test set has been generated with varying numbers of clusters, that we have labeled in order to test the accuracy of the clustering algorithms. We show in Fig. 5 the percentage of correct cluster number identification, in the test set of 100 images, for: (1) the k-means algorithm with initialization, assisted with the metrics of Table 1; (2) the x-means algorithm; and (3) the k-means algorithm assisted with our proposal (i.e., with the number of clusters estimated in the object detection phase).

We can see that the k-means algorithm assisted with our proposal, as well as with the random initialization plus the metrics BIC, AIC, FMI, ARI, and CAL, have yielded the exact number of clusters in all the instances. The metrics DB, SIL, and GAP have produced a very high rate of correct guesses too, and only the x-means has shown a poor rate of correct guesses. Let us mention that the advantage of our proposed solution, compared to the other ones, using the metrics, is that we require less computation and memory overhead (see Fig. 12).

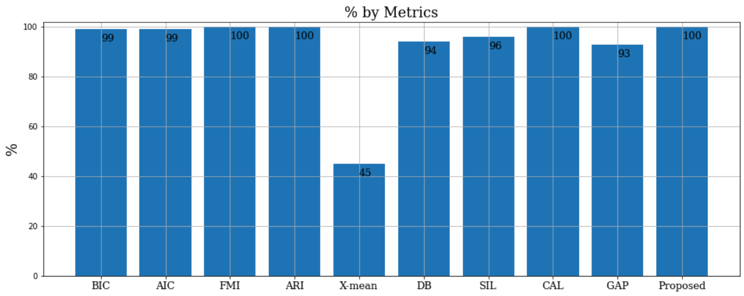


Fig. 5: Percentage of correctly detected number of clusters for 100 tests, where clusters for each test iteration have been generated randomly with different cluster numbers, sizes, and locations.

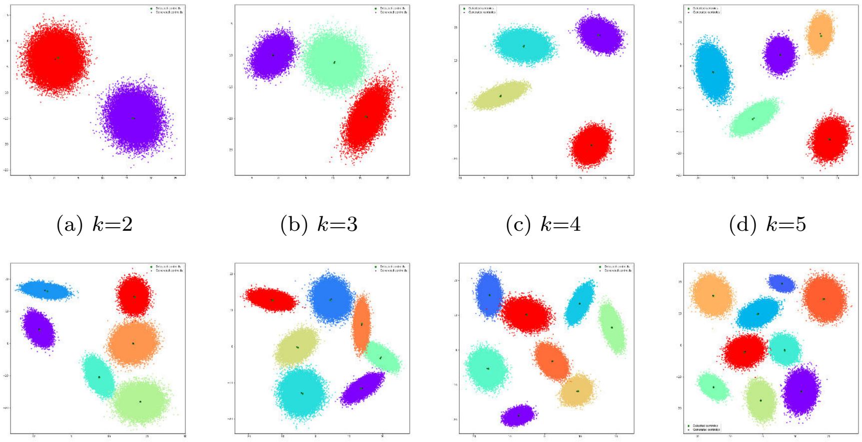
5.2 Centroids Detection Correctness

In this part, we have analyzed the correctness of the detected centroids. Fig. 6 presents a few of the images in our test set. We can remark visually that the identified centroids are very close to the generated ones for the different clusters (see Fig. 9). Moreover, we measure the Euclidean distance between the identified and true centroids, and we have obtained the results presented in Fig. 7, showing that the generated and detected centroids are very close in almost all the instances.

5.3 Accuracy Rate

The clustering accuracy rate (AR) is the proportion of correctly classified observations:

$$AR = \sum_{i=1}^{k} \frac{n(c_k)}{n} \tag{3}$$



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

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

Fig. 6: An example of validation datasets with generated and identified centroids (consequently of each cluster)

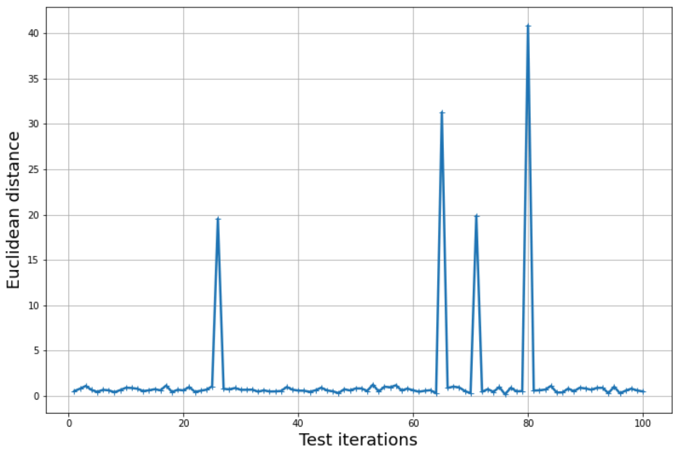


Fig. 7: Average Euclidean distance between generated and detected centroids for the 100 test images with k-means of `scikit-learn` and our initialization

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 detection.

Fig. 8 represents the AR results of k-means with and without the identified centroids in the test set. These results validate that the clustering accuracy (at the level of the samples) 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 iterations to converge and consequently low delay) by using the proposed initialization parameters detection method. For example, as shown in Fig. 9-(e) (detected $nc = 6$), the accuracy is enhanced from 0.854 to 0.999 by using the identified clusters number and centroids.

5.4 Iterations for convergence and time consumption

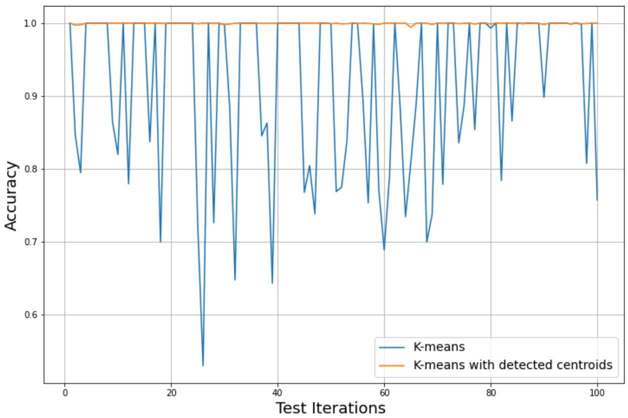
Fig. 11 confirms the reduction in the number of iterations until convergence, for k-means and FCM, when used with the proposed approach, compared to the traditional clustering approaches, for the 100 test set images. As an example, in the case of $nc = 7$ (Fig. 9-f), the required number of iterations to converge is decreased from 4 to 2 by using the identified centroids.

Regarding the time delay of the clustering algorithms, we remark that it depends on three factors:

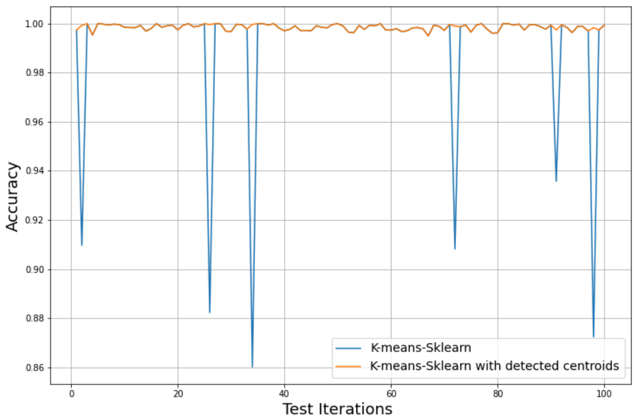
1. The number of iterations in the algorithm.
2. The amount of data points.
3. The time needed to find out the clustering centers and data points partitions.

We have examined the computer time of the proposed solution to detect cluster initialization parameters. Fig. 12 shows the execution times (in seconds) for the k-means assisted by two of the well-known metrics (AIC and BIC), as well as our proposed method, versus the dataset size. We observe a lower execution time for our proposal, compared to the other methods, making it the best choice when working on large datasets. The proposed solution has the advantage of requiring very low computational demand, and consequently, low delay, since it is practically independent of the number of observations in the dataset.

Moreover, we quantify the effectiveness of using these initialization parameters to help the clustering algorithm to converge fast. Fig. 13 shows the execution time ratio between using clustering algorithms with the identified centroids and without it. These results show clearly that using the identified centroids reduces the testing time to half on average. This indicates that the proposed solution reduces significantly the clustering testing time and makes it suitable for large tabular datasets.



(a) k-means naive

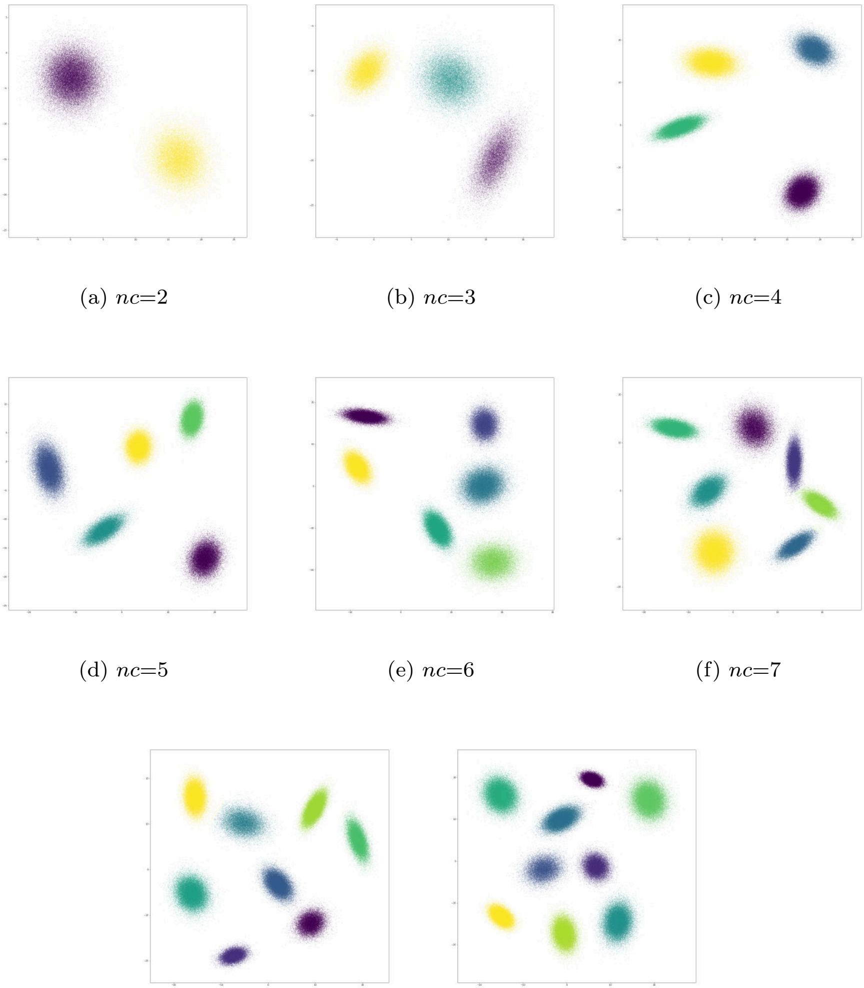


(b) k-means of `scikit-learn`

Fig. 8: Accuracy of k-means clustering algorithms with and without using the proposed approach for two k-means implementations

6 Discussion

The proposed solution involves the implementation of a Deep Learning (DL) approach, utilizing Yolo-v5 with its default configuration to identify cluster initialization parameters. This method demonstrates superior results in comparison to statistical approaches, showcasing enhanced efficiency. Yolo-v5, employed as an object detection model in this context, serves as a proof of con-



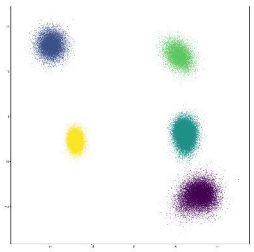
(a) nc=2 (b) nc=3 (c) nc=4

(d) nc=5 (e) nc=6 (f) nc=7

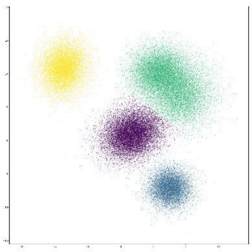
(g) nc=8 (h) nc=9

Fig. 9: The k-means clustering results for the test datasets, listed in Fig. 6, by using the detected cluster number and identified centroids (proposed method)

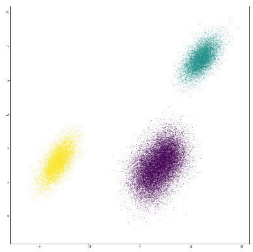
cept. Consequently, based on the achieved outcomes, any proficient DL object detection model could substitute YOLO for detecting initial clustering parameters. The solution leverages the capabilities of DL models for image analysis, emphasizing that the contribution lies in the overall framework rather than the specific DL model used. Additionally, the work includes a comprehensive comparison between classical initialization methods and the proposed DL YOLO-v5 initialization, validating its efficiency and robustness.



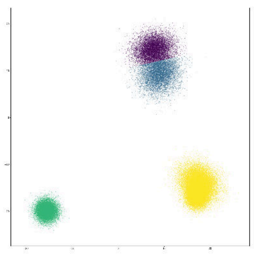
(a) Detected nc=5, BIC=6, AIC=7



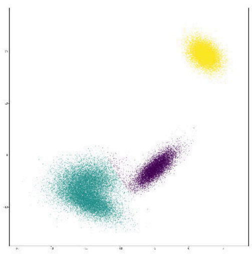
(b) Detected nc=4, BIC=5, AIC=5



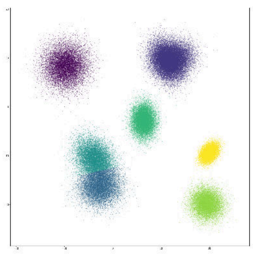
(c) Detected nc=3, BIC=4, AIC=4



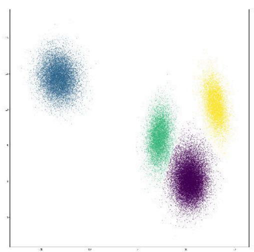
(d) Detected nc=4, BIC=6, AIC=8



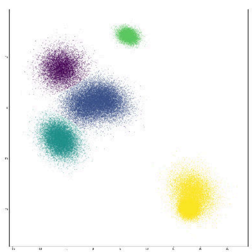
(e) Detected nc=3, BIC=4 AIC=4



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



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



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

Fig. 10: 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)

The proposed work is a new clustering initialization method that can determine the number of clusters of a 2D dataset, in addition to their possible centroids and sizes, improving the speed and results of the most popular clustering algorithms by using the DL object detection model, which is the main contribution of this work. The advantages of the proposed solution are lightness, speed, and robustness with different cluster volumes, shapes, and noise. The proposed solution has been tested with several configurations, proving its efficiency compared to other existing approaches, especially in terms of accuracy, time consumption, and resource overhead, as can be seen in Figs. 8, 11, 12 and 13.

Therefore, this work presents a new simple, and fast way to set up the initial clustering parameters by using the DL object detection models such as YOLO-v5.

7 Conclusion

The proposed solution has proved to outperform the classical setups of cluster analysis, in accuracy as well as in time delay. It should also be noted that even if YOLO-v5 has been used in this paper as proof of work, any future efficient object detection model can be adapted and used instead of YOLO-v5 in the proposed solution.

As future work, we aim to design a new DL-based data transformation model for assisting clustering algorithms in higher-dimension data analysis, as well as envisaging a wider family of cluster shapes (further than Gaussian blobs).

Conflict of Interest

The authors declare that they have no conflict of interest.

Data availability

The code to generate data is available here: `https://github.com/rcouturier/data4clustering`

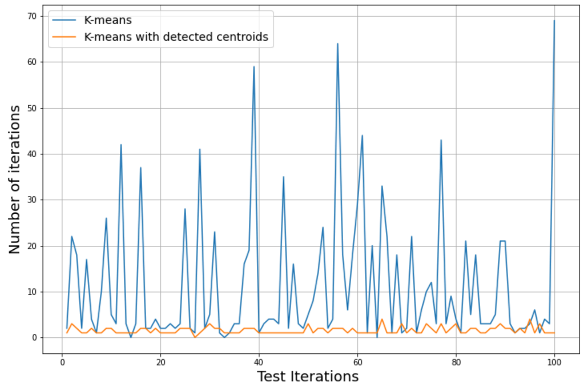
Acknowledgement

This work was partially funded by project ANER 2022 AGRO-IA-LIMENTAIRE and the EIPHI Graduate School (contract ANR-17-EURE-0002). The Mesocentre of Franche-Comté provided the computing facilities. This work was also partially sponsored by the General Directorate for Scientific Research and Technological Development, Ministry of Higher Education and Scientific Research (DGRSDT), Algeria.

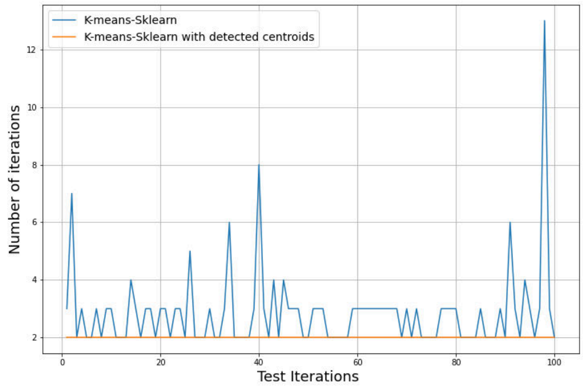
References

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. Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. A robust em clustering algorithm for gaussian mixture models. *Pattern Recognition*, 45(11):3950–3961, 2012.
4. Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE Access*, 8:80716–80727, 2020.
5. Myat Thet Nyo, F. Mebarek-Oudina, Su Su Hlaing, and Nadeem A. Khan. Otsu's thresholding technique for mri image brain tumor segmentation. *Multimedia Tools and Applications*, 81(30):43837–43849, 2022.
6. 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.
7. 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.
8. Mohammad Alhawarat and M Hegazi. Revisiting k-means and topic modeling, a comparison study to cluster arabic documents. *IEEE Access*, 6:42740–42749, 2018.
9. 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.
10. 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.
11. 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, 1967.
12. Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.
13. David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab, June 2006.
14. Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, page 727. Citeseer, 2000.
15. Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.
16. James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer US, 1981.
17. Ka Yee Yeung and Walter L Ruzzo. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.
18. Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995.
19. Tomasz Dziopa. Clustering validity indices evaluation with regard to semantic homogeneity. In *FedCSIS (Position Papers)*, pages 3–9, 2016.
20. Hamparsum Bozdogan. Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, 1987.
21. David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.
22. 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.

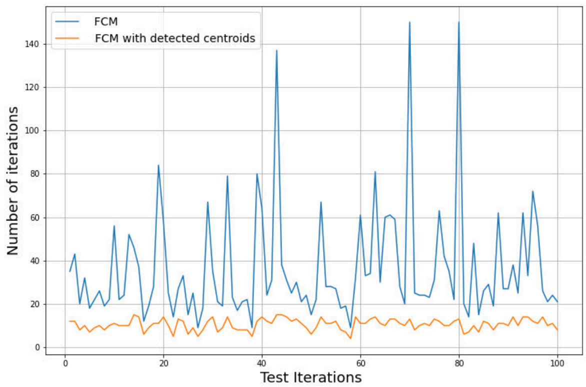
23. Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.
24. Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
25. 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.
26. Yolov5 in pytorch. `https://github.com/ultralytics/yolov5`, 06 2020.
27. Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. In *Programming with TensorFlow*, pages 87–104. Springer, 2021.
28. Nikhil Ketkar and Jojo Moolayil. Introduction to pytorch. In *Deep learning with python*, pages 27–91. Springer, 2021.
29. Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.
30. 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.
31. Pradipta Maji and Sankar K Pal. Rfcm: a hybrid clustering algorithm using rough and fuzzy sets. *Fundamenta Informaticae*, 80(4):475–496, 2007.



(a) Naive K-means implementation



(b) Scikit-learn K-means implementation



(c) FCM

Fig. 11: Variation of the number of iterations required to converge for k-means with random start (a), with `kmeans++` start (b), and for FCM with random start (c), along the images in the test set. It can be seen that our solution to detect centroids is very efficient for all the tested algorithms.

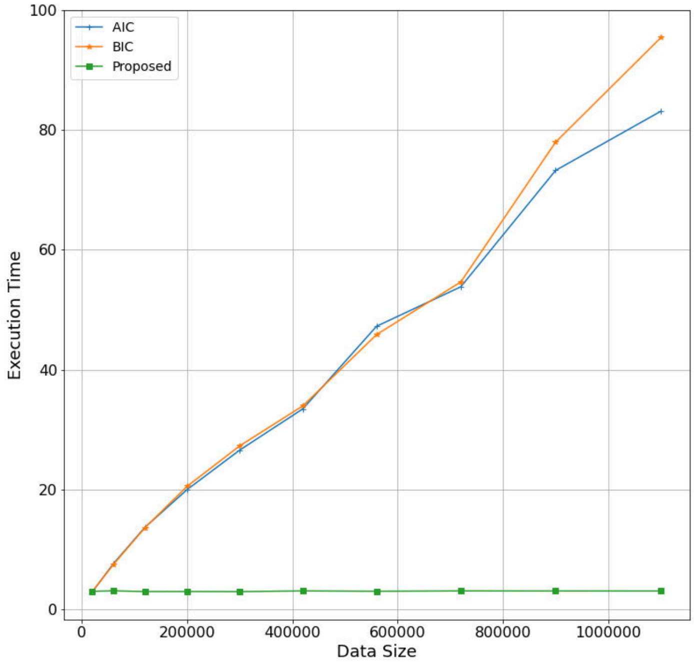


Fig. 12: Comparison of the execution time for the detection of the correct number of clusters, under AIC, BIC, and the proposed initialization method used with the k-means version of `scikit-learn`. It can be observed that the proposed approach is very efficient and scalable compared to other approaches.

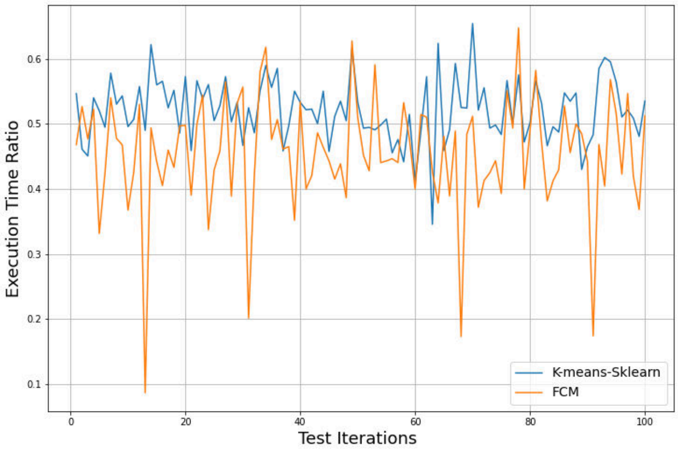


Fig. 13: Ratio of the execution time for k-means and FCM (between "with" over "without" the detected clusters initialization)