Evaluation of Time Series Clustering on Embedded Sensor Platform

Wenyao Zhu
Division of Electronics and Embedded Systems
KTH Royal Institute of Technology
Stockholm, Sweden
wenyao@kth.se

Zhonghai Lu
Division of Electronics and Embedded Systems
KTH Royal Institute of Technology
Stockholm, Sweden
zhonghai@kth.se

***Abstract*—Clustering is one of the major problems in studying the time series data, while solving this problem on the embedded platform is almost absent because of the limitation of computational resources on the edge. In this paper, two typical clustering algorithms, K-means and Self-Organizing Map (SOM), together with Euclidean distance measurement and dynamic time warping (DTW) are studied to verify their feasibility on an embedded sensor platform. For the given datasets, the models are trained on a computer and moved to an ESP32 microprocessor for inference. It is found that the SOM achieves similar accuracy compared with K-means, while its inference process takes a longer time. The experiment results show that a sample with 300 data points can be clustered into 12 clusters within 40 ms by SOM with the DTW model, while the fastest model can run at around 2 ms using K-means with Euclidean distance model. In other words, it can process the data collected from 40 sensors per second in 680 ms. The clustering function can be scheduled with the real-time data acquisition and transmission tasks. The performance gathered supports that it is feasible to deploy the time series clustering model on the embedded sensor platform.**

***Index Terms*—Time series analysis, K-means clustering, Self-Organizing Map, Machine learning on edge**

I. Introduction

Time series data is a type of common information acquired in the embedded sensor-based system mounted with tens or hundreds of sensors. Clustering is one of the major tasks for time series analysis to reveal the features of sensor data. Edge computing [1] is emerged as a promising paradigm that provides capabilities of sensor data processing locally. We can integrate some of the time series clustering methods on the embedded systems. This may help to achieve the real-time inspection of the data on demand in an environment that has no connection to the cloud.

To find out the feasibility of such time series clustering methods on embedded sensor platform, we need to evaluate their performance, taking both accuracy and data inference time under consideration. Due to the limited computing resources of the embedded microprocessor, these clustering methods need to be trained a prior on computer and then do the inference based on the pre-trained model on edge.

In this paper, we focus on two different clustering methods, the representative partitioning approach of K-means algorithm and the neural network approach of SOM algorithm. Besides, both the Euclidean distance and the Dynamic Time Warping (DTW) algorithm are employed as the similarity measurement. We first implement the K-means and the SOM algorithms on PC with Python and train the models to compare the accuracy performance. Then the C version of the models is written for executing on a microprocessor, specifically the ESP32 System on Chip (SoC) microcontroller [2] in our dedicated sensor platform. Then we evaluate the execution time of the clustering models on the embedded platform. Based on the results, we discuss the feasibility of deploying the time series clustering methods on the embedded sensor platform, in addition to its original tasks of real time data collection and transmission.

II. Related works

In the decade review of the time series clustering, Aghabozorgi *et al.* [3] broadly classified the clustering methods into six groups: partitioning, hierarchical, grid-based, model-based, density-based clustering, and multi-step clustering algorithms. Typical algorithms include K-means [4], fuzzy c-means clustering algorithm (FCM) [5], and SOM [6]. The adaptation can be done by replacing the similarity search kernel in those clustering algorithms to fit the time series dataset. Singh *et al.* [7] and Kamimura [8] have proposed their researches on K-means and SOM with other distance metrics than the Euclidean distance. In our case, DTW [9] is a good replacement for similarity search since it is a kind of elastic measure that allows a comparison of one to many points.

In our evaluation, K-means and SOM algorithms are chosen since these clustering methods have been applied in embedded domains in recent years. For example, Riveros *et al.* compared the clustering precision for diagnosis spinal column patients [10]. There has been novel FPGA accelerator and distributed implementation of the SOM network [11]. The study in deploying K-means using configurable many-core hardware and software architecture on FPGA is proposed by Canilho *et al.* [12]. These researches have achieved a lot of progress in adapting the K-means and the SOM algorithms on different embedded hardware and improving their efficiency. However, there are few works comparing the real-time performance of these clustering algorithms on a general-purpose embedded microprocessor. Considering the application of time series clustering on an existing embedded sensor platform, the general microprocessor with acceptable processing performance is preferred rather than an expensive domain-specific accelerator.

III. Theoretical Background

The main idea of K-means algorithm [4] is to minimize the total distance between all objects inside cluster while maximizing the distance between clusters. The principle solution of K-means is based on the Expectation-Maximization algorithm [13]. Each iteration of the K-means algorithm is composed of two steps: the first step is the expectation step, in which the time series are labeled according to the centroids; the second step is the maximization step, in which the centroids are moved to the newly labeled time series. During the iterations, the cluster converges to the optimal result.

Self-Organizing Map (SOM) [6] is a kind of neural network designed to solve clustering tasks, and it originates to show the geometry relationship of the objects in the dataset. SOM has a two-layer neural network structure, which contains the input layer and output layer (competitive layer). The input layer simulates the perception channel, through which the original time series pass. The output layer simulates the response channel. The neurons in this layer represents the clusters we desire. The training process is a competitive process. Each input is assigned to the best matching unit (BMU) as the winning neuron. Then we can use stochastic or batch training to update the weight of the winning neuron, and appropriately update its neighbors' weights according to the distance of the winning neuron.

K-means algorithm has the advantage of fast convergence speed. The clustering result is easy to compute and explain. However, it may fall into the local minimum, which depends on the selection of K and initial centroids (weights). Compared with K-means, the model-based SOM doesn't require that each neuron of the output layer contains an element. SOM has a cooperation process, which can reduce the effect of noise data, while this may also reduce the final accuracy.

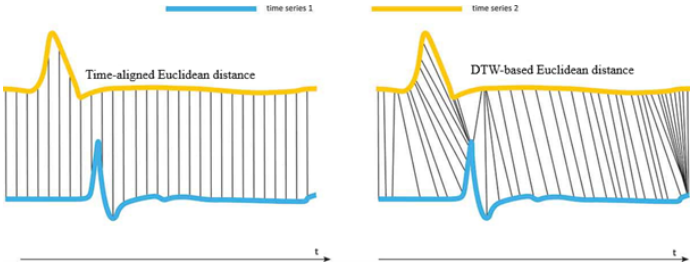

Fig. 1. Distance measure by Euclidean and DTW of two time-series [14].

DTW [9] is a kind of elastic measure of the distance between two discrete sequences. Time series data collected from any embedded sensor system are discrete sequences, so DTW can work on the similarity search task for them. DTW can do one-to-many comparisons, which means it has adaptability on the extension or compression of the two time series. Figure 1 shows the difference between this feature and the one-to-one comparison by the Euclidean distance. In the embedded sensor platform, the sensor data are not perfectly aligned depending on the position and reading order of the sensors, then the DTW algorithm can help on eliminating the distortion on the data.

IV. Methodology

A. Evaluation Process

In this section, the evaluation process for both the selected clustering model, the K-means and SOM algorithms, on the embedded sensor platform is presented. The accuracy and execution time of the models are evaluated based on public datasets. After that, we will discuss the time performance of these models on our target sensor platform to seek the feasibility to bring the clustering function to an embedded platform, which handles the data from its own sensors.

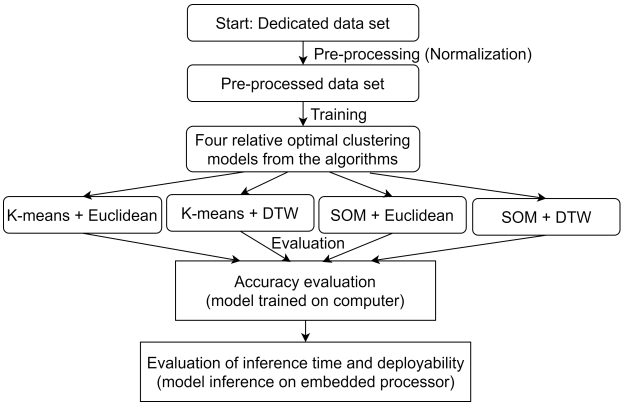

Fig. 2. The evaluation process.

Figure 2 summarizes the implementation and experiment process. For a dedicated dataset, we would do normalization before feeding it into the clustering models. Then models are trained on a computer to obtain the relative optimal ones. These trained models are then implemented on the embedded processor for the time evaluation of the inference process. The distance comparison in K-means and SOM algorithms can be replaced by DTW from the original Euclidean distance. Hence in total, four combinations of the algorithms, K-means with Euclidean metric, K-means with DTW, SOM with Euclidean metric and SOM with DTW, are under evaluation as shown in the diagram.

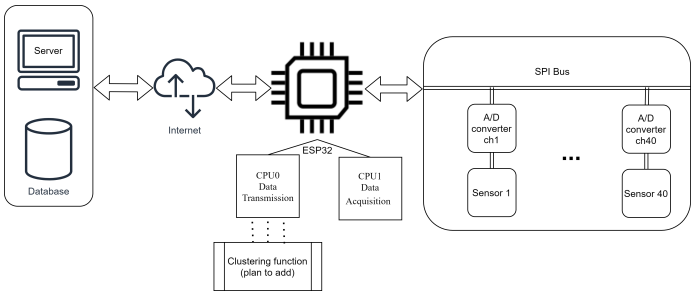

Fig. 3. The sensor platform with ESP32 in the study case.

To be concrete, the embedded platform we used for the evaluation is the ESP32 microcontroller unit (MCU), which has a dual-core microprocessor with a frequency of up to 240 MHz [2]. The experiment models are implemented and run on a single core of it. We choose this microprocessor since our target sensor system utilizes the same MCU to acquire and transmit data from pressure sensors. If we want to insert the clustering function into this system, it should be scheduled in the time gap of the recurrent transmission

tasks on the same core of the ESP32, meanwhile, the sensor reading process will occupy the other core. The structure of the system is shown in Figure 3. In our sensor platform setup, the ESP32 collects pressure data from 40 pressure sensors at a designed acquisition rate of 50 Hz. A packet of 40 frames, each frame with a length of 50 data points is generated in one second. Under the lab situation, the data transmission task costs an average of 77.1 ms to handle the packet round trip between the server and the sensor system, while a peak of 163.8 ms is recorded. Consider that the data are already in the memory when we apply the clustering algorithm to them before the packet is sent, we have an upper bound of around $1000 - 163.8 = 836.2$ ms for the inference process. This setup provides a time baseline as a study case.

B. Implementation of Clustering Models

For the clustering algorithms selected, the similarity measurement kernel appears in the expectation step in K-means and the competition step in SOM. The common Euclidean distance measurement can be replaced by the DTW algorithm mentioned in the previous section. The K-means algorithm takes the number of clusters K, the data field D and the number of iterations M as the input. It also needs two extra parameters, the window size w and the bound size b, for the DTW as the distance measurement method. Algorithm 1 shows the example pseudo code on how we implement the distance measurement methods in the K-means algorithm, and we will do the same to the SOM algorithm.

First, we initialize the cluster centroids by randomly selecting K samples from the original data field D (line 1). After the initialization, we repeat the following two steps during each iteration until it reaches the maximum iteration number M.

- Step 1: Measure the similarity of each sample in D to the cluster centroids, and find the best matching centroid C_i with the shortest distance as its cluster (line 3-19).
- Step 2: Update the mean position of each cluster. The mean position equals the mean value sequence along each column of the data matrix in that cluster (line 20-22).

Finally, we assign each sample with clustering label and export the cluster centroids. Notice that we have two blocks (line 8-12 and line 13-18) about the distance measurement method in this pseudo code, which explain the usage of the Euclidean metric and DTW in K-means. The DTW part includes the lower bound trick [15], which takes one more comparison statement than the Euclidean metric. Either of them is selected to build up one model according to the evaluation process.

The SOM implementation also contains the block for distance measurement. We choose the rectangular grid for the competitive layer to simplify the network and the cooperation process. The weights of the neurons on the competitive layer are randomly initialized with small values before the learning iterations. Each iteration is divided into three parts: the competitive process, the cooperation process, and the adaptation process. The updates for one cycle only happen in the neurons around the BMU, while other neurons keep unchanged. The closer neurons to the BMU will have a larger weight update.

Algorithm 1: K-means algorithm

Input: data field D, max iteration M, cluster number K, window size w, bound size b
Output: cluster centroids C, labels $assignment$

```
1  C = randomly picked K samples from D;
2  for n in each iteration M do
3      ▷ Step 1: Do clustering based on centroids;
4      for i in enumerate D do
5          min_dist = inf ;
6          closest_cluster = None;
7          for j in enumerate C do
8              ▷ Distance measurement 1: Euclidean;
9              dist = EUC(D[i],C[j],b);
10             if dist < min_dist then
11                 min_dist = dist;
12                 closest_cluster = j;
13             ▷ Distance measurement 2: DTW;
14             if LB_Keogh(D[i], C[j], b) < min_dist
                then
15                 dist = DTW(D[i],C[j],b);
16                 if dist < min_dist then
17                     min_dist = dist;
18                     closest_cluster = j;
19             Add i in assignment[closest_cluster];
20         ▷ Step 2: Update centroids;
21         for p in assignment do
22             C[p] = average value of samples in
                assignment[p];
23 return C and assignment;
```

The DTW algorithm is programmed and inserted into the K-means and SOM clustering methods. The basic idea of the DTW algorithm is to get a warping curve in the distance matrix, formed with each point pairs of two input time series, with a minimum accumulated distance. The distance matrix is initialized and then updated according to the dynamic programming approach with the three restrictions on the monotonicity, the continuity and the search boundary to find this curve. After that, the DTW distance is derived from this path. Besides, the Keogh's lower bound algorithm [15] is also used to provide a mathematical lower bound of the DTW distance with faster speed. So we can skip some of the DTW calculations during the distance comparison in K-means and SOM in order to accelerate the clustering process.

The implementation of these clustering models includes the training process and the inference process. For the convenience of the experiments, the K-means and SOM algorithms are programmed in two formats, the Python version and the C version. The Python version is used to run on the computer for training and accuracy evaluation. The C version is implemented on the embedded processor to evaluate the real-time feasibility of the inference process. The trained parameters are copied from the Python version to the C version for inference.

V. Experimental Results

A. Dataset and evaluation metrics

To evaluate the clustering algorithms before applying them on our sensor platform, we use well-organized and widely used public data sets. In our experiment, several entries are picked up from the UCR time series classification archive [16]. They are labeled and formatted with a short description of the data source. In the following evaluation, the BME dataset (3 classes, 180 samples and 128 points per sample) and the three Cricket datasets (12 classes, 780 samples and 300 points per sample) are selected from the UCR archive. All of the datasets are under the motion category, as one of the essential targets for clustering on an embedded system is to deal with sensor data related to motions.

We can use Rand index [17] to evaluate the accuracy of clustering models, which compares the number of the data pairs co-assigned or separated in true clusterings according to their mutual similarity with the total number of pairs. For a small rand index close to 0, the trained clusters are totally different from the desired results. While for a large rand index close to 1, the trained clusters are substantially identical to the ground truth. Besides, we also want to find out the inference time of the models on the microprocessor. The trained models on the selected UCR datasets are used as a reference and the time requirement of the described sensor platform is discussed as a study case. The software timer is set up to record the calculation part of the inference process on the ESP32, and the whole dataset is shuffled for the evaluation. The average time can then represent the inference time for one sample.

B. Performance Comparison on accuracy

Four models are trained using K-means and SOM, for the selected BME and cricket datasets. Table I records the accuracy results of the Python version of the K-means and SOM models run on a normal computer as well as the training and inference time. For all of the experiments in K-means, the parameters in training are set as follows: $w = 100$, $b = 5$, $k = 3$ and 12 respectively for BME and cricket datasets, and the iteration number is 10. While for the experiments in SOM, the parameters are $w = 15$, $b = 5$, $k = 1 \times 3$ and 3×4 respectively for BME and cricket datasets, the learning rate is 0.5, the decay of neighbour range is 1.0, and the iteration number is set to 10. Regarding the accuracy performance of the four models, we can observe that the DTW performs a little better than Euclidean distance in most of the test cases, but the advantage is limited to an average of 3.75%. The possible reason is that the DTW has strength in dealing with data distortion and phase movement, while the original data are well defined and cleaned and no such defect exists in them.

Figure 4 plots two example results as the mean value of the BME samples located in the corresponding clusters using the K-means with DTW model and the SOM with DTW model. The curves in the three clusters meet the description of the dataset, where each time series contains one peak in the beginning, middle or end. Hence both of them generate a relatively good clustering as we can judge from the shape.

TABLE I
Performance of K-means and SOM on selected datasets.

Dataset	Measurement	K-means RI	SOM RI
BME	DTW	0.6978	0.7106
BME	Euclidean	0.6125	0.6233
CricketX	DTW	0.8596	0.8591
CricketX	Euclidean	0.8481	0.8487
CricketY	DTW	0.8505	0.8722
CricketY	Euclidean	0.8357	0.8521
CricketZ	DTW	0.8598	0.8715
CricketZ	Euclidean	0.8600	0.8609

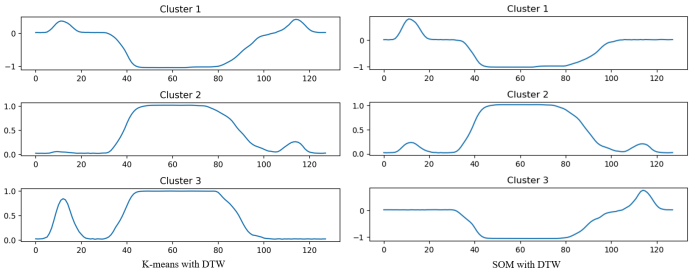

Fig. 4. The clustering results for the BME dataset.

Compared with the accuracy performance between K-means and SOM, the performance is quite close whereas the SOM has a small advantage of an average of 1.4%. One possible explanation is that we perfectly fit the number of clusters in K-means and SOM with the number of classes in the original datasets, so the accuracy results are quite similar. Since SOM supports empty clusters, the output layer can contain more neurons than reference, then the SOM model may increase its advantage on accuracy compared with the K-means model.

C. Performance Comparison on inference time

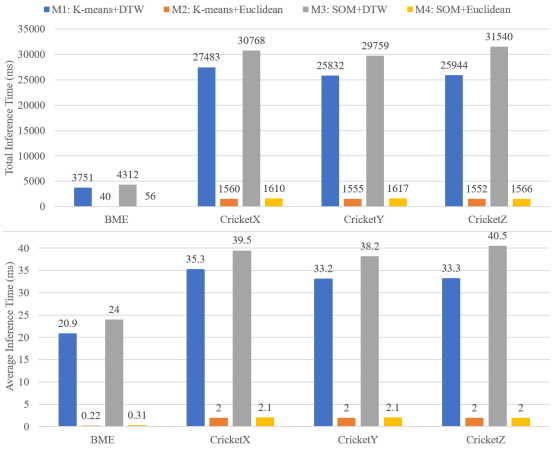

Fig. 5. Total and average inference time on ESP32.

The results of the experiments on the ESP32 are shown in Figure 5, in which model M1 is the K-means with DTW, model M2 is the K-means with Euclidean distance, model M3 is the SOM with DTW, and the model M4 is the SOM with Euclidean distance. From this figure, we can find that the time used in SOM is 12% to 40% longer than the K-means. Generally, the time used with DTW is 19% to 93% longer than with Euclidean distance. A sample in BME contains 128 data points, while a sample in Cricket dataset contains 300 samples.

Obviously that the sample with a shorter length will consume less time in the inference phase for the same clustering model. The longest total inference time here is 31.5s when using the model M3 on the CricketZ dataset. Generally, we can make a cluster model using the SOM with DTW on a sample of 128 points for three classes in 24 ms, or a time series sample of 300 points for twelve classes in 40 ms. Compared with the fastest M2 among four models, the inference time for a sample of 128 points for three classes in 0.22 ms, or a sample of 300 points for twelve classes in 2 ms.

D. Discussion on the case of an embedded sensor platform

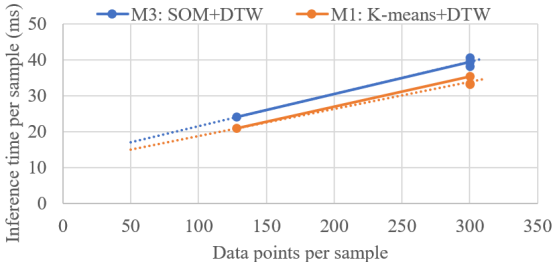

Fig. 6. Relationship between sample length and inference time.

Here we discuss the time requirement for deploying these time series clustering models in our embedded sensor platform study case. Using interpolation of the evaluation results for the two DTW enabled models, M1 and M3, with the longest inference time results among four models, the prediction lines are shown in Figure 6. Then we can estimate that the frame with length of 50 data points takes at most 17 ms to complete, which is 680 (17×40) ms for a whole 40-frame packet. Hence for all the implemented models, the inference time in our study case should be less than 680 ms. By comparing with the given deadline of 836.2 ms, the inference task is quite acceptable to be scheduled in our sensor platform.

Among our evaluation lists, the largest size of the SOM model weights is around 28.8 KB and the dataset is around 1872 KB, which can be fitted into the 4 MB flash memory on the ESP32 board. In our sensor platform study case, the data size is around 16 KB for one packet, with the expected weight matrix of 1.2 KB for a three-cluster model in double type, which can be completely loaded into the memory of the ESP32 microprocessor. Overall, the study case shows the potential to deploy the time series clustering function on our target sensor platform using the ESP32 microprocessor.

VI. Conclusion

The objective of this paper is to assess the feasibility of the time series clustering on an embedded sensor platform. Two typical clustering methods, the K-means and SOM algorithms, together with the Euclidean distance measurement and the DTW algorithm, are implemented and evaluated. The inference time bound is considered in the study case. All four algorithms can be scheduled based on the time requirement of the original data acquisition and transmission tasks. In our case on the ESP32 microprocessor, for a three-cluster model with 40 sensors collecting the length of 50 data points per frame, the SOM with DTW model requires 680 ms for inference, and the system requires 1.2 KB for loading the model weights and 16 KB for storing the data in memory in double precision.

In our evaluation, the clustering model parameters and input data are stored as double type floats. By reducing the precision, we expect to save storage and reduce the execution time. Such aspects are important to be investigated in future research.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825429.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] Espressif Systems. ESP32 series datasheet. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[3] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering: a decade review," *Information Systems*, vol. 53, pp. 16–38, 2015.

[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. [Online]. Available: https://projecteuclid.org/euclid.bsmsp/1200512992

[5] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[6] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[7] A. Singh, A. Yadav, and A. Rana, "K-means with three different distance metrics," *International Journal of Computer Applications*, vol. 67, no. 10, 2013.

[8] R. Kamimura, S. Aida-Hyugaji, and Y. Maruyama, "Information-theoretic self-organizing maps with minkowski distance." in *Proceedings of IASTED International Conference on Artificial Intelligence and Soft Computing*, 2003, pp. 15–20.

[9] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[10] N. A. Melo Riveros, B. A. Cardenas Espitia, and L. E. Aparicio Pico, "Comparison between k-means and self-organizing maps algorithms used for diagnosis spinal column patients," *Informatics in Medicine Unlocked*, vol. 16, p. 100206, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S235291481930098X

[11] M. A. de Abreu de Sousa and E. Del-Moral-Hernandez, "An fpga distributed implementation model for embedded som with on-line learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3930–3937.

[12] J. Canilho, M. Véstias, and H. Neto, "Multi-core for k-means clustering on fpga," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–4.

[13] T. Moon, "The expectation-maximization algorithm," *Signal Processing Magazine, IEEE*, vol. 13, pp. 47 – 60, 12 1996.

[14] P. Ranacher and K. Tzavella, *How to compare movement? A review of physical movement similarity measures in geographic information science and beyond*, 3 2014, vol. 41, pp. 286–307.

[15] S.-W. Kim, S. Park, and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," in *Proceedings of 17th International Conference on Data Engineering*. IEEE, 2001, pp. 607–614.

[16] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, P. Yan, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The UCR Time Series Classification Archive," October 2018, https://www.cs.ucr.edu/ eamonn/time_series_data_2018/.

[17] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.