2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Scalable Kernelized Deep Fuzzy Clustering Algorithms for Big Data

Preeti Jha
Computer Science and Engineering
Indian Institute of Technology Indore, India
now at Koneru Lakshmaiah Education Foundation
Bowrampet, Hyderabad, India
preetijha@klh.edu.in

Aruna Tiwari
Computer Science and Engineering
Indian Institute of Technology Indore
Indore, India
artiwari@iiti.ac.in

Neha Bharill
Computer Science and Engineering
Mahindra University
Hyderabad, India
neha.bharill@mahindrauniversity.edu.in

Milind Ratnaparkhe
Biotechnology
ICAR-IISR
Indore, India
milind.ratnaparkhe@gmail.com

Om Prakash Patel
Computer Science and Engineering
Mahindra University
Hyderabad, India
omprakash.patel@mahindrauniversity.edu.in

Anjali Gupta
Computer Science and Engineering
Indian Institute of Technology Indore
Indore, India
guptaanjali7786@gmail.com

Deepali Sukhija
Computer Science and Engineering
Indian Institute of Technology Indore
Indore, India
sukhijadeepali2001@gmail.com

Deepika Sukhija
Computer Science and Engineering
Indian Institute of Technology Indore
Indore, India
sukhijadeepika2001@gmail.com

Rajesh Dwivedi
Computer Science and Engineering
Indian Institute of Technology Indore
Indore, India
rajeshdwivedi@iiti.ac.in

Abstract—**Conventional scalable clustering-based Deep Neural Network (DNN) algorithms cluster linearly separable data, however non-linearly separable data in the feature space is harder to cluster. This paper proposes a novel Scalable Deep Neural Network Kernelized Literal Fuzzy C-Means (SDnnKLFCM) and Scalable Deep Neural Network Kernelized Random Sampling Iterative Optimization Fuzzy C-Means for Big Data (SDnnKRSIO-FCM). These kernelized clustering methods solve non-linear separable issues by non-linearly transforming the input data space into a high-dimensional feature space using a Cauchy Kernel Function (CKF). We create kernelized deep neural network fuzzy clustering methods using Apache Spark in-memory cluster computing technique to efficiently cluster Big Data on High-Performance Computing (HPC) machine. To demonstrate the effectiveness of the proposed (SDnnKLFCM) and (SDnnKRSIO-FCM) in comparison to previous scalable deep neural network clustering methods, extensive tests are carried out on a variety of large datasets. The reported experimental results show that the kernelized non-linear deep clustering algorithms in comparison with linear fuzzy clustering algorithms achieve significant improvement in terms of Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and F-score, respectively.**

Index Terms—**Kernelized Algorithms, Big Data, Non-linear, Fuzzy Clustering, Deep Neural Network**

1. Introduction

Clustering is an approach that aims to organize datasets that are comparable into a single cluster on the basis of several measures of similarity (e.g., Euclidean distance). On the other hand, numerous approaches to the clustering of data have been proposed [1], [2]. The performance of conventional clustering is not good, and it has a very high level of computing complexity, due to the ineffectiveness of the similarity measures that are utilized in these algorithms on high-dimensional data. Due to these factors, dimensionality reduction and feature transformation have received a significant amount of research attention for the purpose of mapping the data into a new feature space. This will make it possible for the existing classifiers to separate the newly created data in a more straightforward manner. Linear transformations, like principal component analysis (PCA) [3], and non-linear transformations, including kernel methods [4] and spectral methods [5], make up the bulk of the currently available approaches to transforming data. Unfortunately, the highly complex latent structure of the data continues to be a barrier to the efficacy of existing approaches to clustering, despite their best efforts. The development of deep learning has led to the emergence of a property known as highly non-linear transformation, which enables DNNs to convert data into representations that are more conducive to clustering. This property enables DNNs to be used for the purpose of data transformation [6].

Fuzzy c-Means (FCM) clusters well if feature space data is linear [7]. Kernel-based Fuzzy c-Means (KFCM) are designed to deal with non-linear shape clusters by substituting the Euclidean distance metric with a kernel metric [8]. Scalable fuzzy clustering algorithms were devised by modeling data in a neural network feature space to handle high-dimensional data with a complex latent distribution [9]. Scalable DNN based on fuzzy clustering algorithms: Scalable Deep-Neural Network Random Sampling Iterative Optimization-FCM (SDnnRSIO-FCM) and Scalable Deep-Neural Network Literal Fuzzy c-Means (SDnnLFCM), developed to handle Big data [10]. Here, decoding the encoded representation with an Autoencoder (AE) neural network recovers the original data using the Apache Spark cluster. Autoencoder maps data into lower-dimensional feature distribution in latent space in this work [10]. Although the SDnnLFCM and SDnnRSIO-FCM algorithms handle DNN using fuzzy clustering approaches [10]. The SDnnLFCM and SDnnRSIO-FCM were unable to process data that cannot be linearly separated. By making modifications to the cluster centers, membership matrix, and vector norms the proposed SDnnKLFCM and SDnnKRSIO-FCM algorithms have been effective in resolving this issue. Instead of using the Euclidean distance, a Cauchy Kernel Function (CKF) is used to define the vector norm [11].

The following is an outline of the subsequent section of this paper: Both the scalable fuzzy clustering algorithm and the deep neural network architecture are explained in detail in Section 2. The proposed SDnnKRSIO-FCM and SDnnKLFCM method is discussed in 3. The experimental results are reported on the extremely large benchmark datasets in Section 4. Section 5 concludes our work.

2. Preliminaries

Deep neural network (DNN) optimization and the combination of unsupervised clustering algorithms are currently active research areas, although these techniques are not scalable. The Scalable Deep-Neural Network Random Sampling Iterative Optimization-FCM (SDnnRSIO-FCM) and Scalable Deep-Neural Network Literal Fuzzy c-Means (SDnnLFCM) are based on DNN techniques [10]. These algorithms employ the Normal Autoencoder (NAE). A detailed discussion of DNN techniques, SDnnLFCM, and SDnnRSIO-FCM, is presented next.

2.1. Deep Autoencoder

AE learns unlabeled input embedding for a specified network architecture. Encoding and decoding are the two parts that comprise AE. The encoder takes data from a higher-dimensional space and transforms it into a lower-dimensional space, while the decoder takes data from a lower-dimensional space and changes it to a higher-dimensional space. The data that the decoder has received is used as output so that the latent space can acquire the vast majority of the information about the data space. Given a dataset X with N data points. The k^{th} neuron in the l^{th} layer of the AE for each x_i (i^{th} data point) is denoted as z_{ki}^l and defined as follows:

$$z_{ki}^l = \mathrm{f}\left(\sum_{j=1}^{n_{l-1}} z_{ji}^{l-1} w_{jk}^l + b_k^l\right). \tag{1}$$

In this case, L represents the overall number of layers in the AE, and $l \in \{1, 2, \dots L\}$ represents the layer index. The weights, bias, and the number of the neurons in the l^{th} layer are denoted as w_{jk}^l, b_k^l, and n_l, respectively. In Equation (1) activation function is denoted by f. We utilized the sigmoid activation function represented as $\mathrm{f}(a)=\frac{1}{1+\exp(-a)}$. The output vector ($z_{1i}^l, z_{2i}^l), \dots, z_{n_l i}^l$ in the l^{th} layer. An initial input layer of the AE is defined as $z_i^1 = x_i$ in Equation (1). The encoder consists of the $L/2$ layers and $z_i^{l=L/2}$ hidden features vector.

The decoder comprises the subsequent $L/2$ layers of the neural network. It aims to reconstruct the x_i from the encoded output, $z_i^{l=L/2}$. Let the output of the decoder be denoted as $z_i^L = h_{W,B}(x_i)$, with all the weights (W) and biases (B) in the AE. Identical to the output z_{ki}^l of the l^{th} layer, the output of the k^{th} neuron in the final layer of the AE is given by $z_{ki}^L = h_{W,B}(x_i)_k = \mathrm{f}(\sum_{j=1}^{n_{L-1}} z_{ji}^{L-1} w_{jk}^L + b_k^L)$. Since the decoded data should be close to the original data, so AE aims to minimize the reconstruction loss given by:

$$J(x,\theta) = \frac{1}{N}\sum_{i=1}^{N} \parallel h_{W,B}(x_i) - x_i \parallel^2 \tag{2}$$

2.2. Deep Clustering Based on Student's-t Distribution

AEs are optimized by deep clustering methods DEC [12] and IDEC [13]. DEC and IDEC cluster the data using a Student's-t distribution-based clustering layer. The data X is input for the network, where, $x_i \in R^d$. And $Z = z_1, \dots, z_i, \dots, z_N$, where, a hidden feature from the AE is $z_i \in R^c$. The d and c are the dimensions of the input layer and hidden feature layer, respectively. The clustering layers will continue to receive z_i for Student's-t distribution clustering [12], [13]. The output obtained from the clustering layer is given as follows:

$$\mu_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k=1}^{c}(1 + \|z_i - \mu_k\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \sum_{j=1}^{c}\mu_{ij} = 1 \quad \forall i \tag{3}$$

The clustering layer training weights defined as follows: $\mu = [\mu_1, \dots, \mu_j, \dots, \mu_c]$, class depicted by c, and clustering layer j with j^{th} neuron. μ_{ij} is the degree of membership for a given cluster. Whereas, μ_{ij} depicts the cluster center or Student's-t distribution. The target p_{ij} of the Student's-t based membership μ_{ij} is determined by Equation (4).

$$p_{ij} = \frac{\mu_{ij}^2 / \sum_i \mu_{ij}}{\sum_{k=1}^{c} (\mu_{ik}^2 / \sum_i \mu_{ik})}, \sum_{j=1}^{c} p_{ij} = 1 \quad \forall i. \tag{4}$$

2.3. KL-Divergence-Based Objective Term

The deep fuzzy clustering model limitation improves FCM clustering. Several clustering methods now assess memberships using within-cluster compactness. The Student's t-distribution-based algorithms are given in Equation (3). The μ_{ij} is defined as p_{ij} in Equation (4). Because the output μ_{ij} of the clustering layer is connected to the input z_i, which is obtained by minimizing distributions by the KL divergence. As a result, the clustering layers output is displayed in Equation (4), where the membership μ_{ij} approaches p_{ij}.

The mean squared error (MSE) term from Equation (2) is motivated by AE design and the concept of data recovery. The optimization of membership from Equation (5) is related to the KL-divergence component. The objective function of the deep fuzzy clustering is represented in Equation (6).

$$\min \mathrm{KL}(P||Q) = \min \sum_{i=1}^{N} \sum_{j=1}^{c} p_{ij} \log \frac{p_{ij}}{\mu_{ij}}. \tag{5}$$

$$L = \sum_{i=1}^{N} \parallel h_{W,B}(x_i) - x_i \parallel_2^2 + \alpha \sum_{i=1}^{N} \sum_{j=1}^{c} p_{ij} \log \frac{p_{ij}}{\mu_{ij}}. \tag{6}$$

2.4. Scalable Deep-Neural Network Random Sampling Iterative Optimization-FCM (SDnnRSIO-FCM) and Scalable Deep Neural Network Literal Fuzzy c-Means (SDnnLFCM)

The SDnnLFCM and SDnnRSIO-FCM algorithms are implemented on the Apache Spark cluster under the HPC environment [10]. The dataset is batch-divided. In each iteration, the batch is encoded by the encoder layers. Clustering is done after passing encoded data to decoder layers. Equation (2) calculates the NN and MSE loss after decoding the reconstruction. Scalable Literal FCM (SLFCM) processes encoded data to determine cluster centers and membership values during clustering. SLFCM is the scalable version of the FCM algorithm [14]. Equation (4) calculates the target distribution from membership values. As described in section 2.3, we calculate the KL-divergence loss for clustering using this target distribution and membership values. Equation (6) calculates the final loss value from the MSE loss and clustering KL loss. The stochastic algorithm adjusts the NN weights using this loss. SLFCM inputs cluster centers as follows: The first batch pre-initialized the cluster centers. The second batch clustering uses output cluster centers from the first batch. Membership values from the first and second batches are blended to discover updated third-batch cluster centers. All Batches follow the same technique. In SDnnLFCM, SLFCM runs iteratively. Whereas, SDnnRSIO-FCM works as follows: It calls SLFCM for the initial two batches and then merges the output of the first two batches to obtain membership values that can be used to calculate updated cluster centers. The formulation for membership values and cluster centers is given in Equations. (7) and (8), respectively.

$$Membership\ Degree\ (\mu_r) = \frac{(1 - K(x_i, v_j))^{1/(p-1)}}{\sum_{j=1}^{c} (1 - K(x_i, v_j))^{1/(p-1)}} \tag{7}$$

$$Cluster\ Center\ (c') = \frac{\sum_{i=1}^{s} m_{ij}^{p} K(x_i, v_j) x_i}{\sum_{i=1}^{s} m_{ij}^{p} K(x_i, v_j)} \tag{8}$$

Despite the fact that the SDnnLFCM and SDnnRSIO-FCM algorithms manage DNN by employing fuzzy clustering methods, data that cannot be linearly separated could not be processed by either the SDnnLFCM or the SDnnRSIO-FCM. Hence, both the proposed SDnnKLFCM and SDnnKRSIO-FCM algorithms have been successful addressing this issue. A detailed description of the SDnnKLFCM and SDnnKRSIO-FCM algorithms is given in the next section.

3. PROPOSED WORK

The SDnnKRSIO-FCM and SDnnKLFCM kernelized versions of the SDnnRSIO-FCM and SDnnLFCM, respectively, are proposed in this study for handling Big Data analysis. The SDnnLFCM and SDnnRSIO-FCM use fuzzy clustering to handle DNN [10]. Hence, SDnnLFCM and SDnnRSIO-FCM could not process non-linear data. The proposed SDnnKLFCM and SDnnKRSIO-FCM algorithms solved this problem by modifying the cluster centers, membership matrix, and vector norms. Here, Cauchy Kernel Functions (CKFs) define vector norms instead of Euclidean distances. The proposed algorithms are cutting-edge methods that can be used to cluster non-linear data. Each data block is processed in parallel using Apache Spark. To acquire more expressive characteristics that aid in handling non-linear relations, kernel functions CKF [11] utilized are defined as follows:

$$K(x,\ y) = \frac{\sigma^2}{\|x - y\|^2 + \sigma^2} \tag{9}$$

Where σ denotes the smoothing parameter and $\|.\|$ denotes the Euclidean distance. The Cauchy kernel is a unit-norm kernel, i.e., $0 \leq K(x,\ y) \leq 1 \ \forall \ x, y \in X$. If σ in Equation (9) tends to zero then $K(x,\ y) \approx 0$, and if it tends to infinite then $K(x,\ y) \approx 1$.

3.1. Scalable Kernelized Deep Neural Network Literal FCM (SDnnKLFCM)

As mentioned before, linear relations are supportable by SDnnLFCM. To expand SDnnLFCM, the kernel method concept is presented in order to handle non-linear relationships. By using the CKF, the SDnnKLFCM algorithm is a kernelized variant of the SDnnLFCM method. Algorithm 1 provides a summary of the proposed SDnnKLFCM. The cluster nodes are initialized once the NAE has been pre-trained on the data using a learning rate (0.001) for 100 iterations. With a learning rate of 0.001, the initial training now starts. For each iteration and batch, layers from the network encoder section are used to create the hidden representation Z_r. The MSE loss is then computed using this format, and data clustering is performed in Line 5. Line 6 and 7 employs the KSLFCM method for clustering. The KSLFCM is a scalable kernelized version of the FCM method [7]. To determine the desired distribution for membership values, the Student's t-distribution is employed at Line 8. The values of the goal distribution and current distribution are used to calculate the clustering loss for the particular batch in the line that follows. The NN weights are finally changed using Stochastic Gradient Descent on Line 10. It contains the KL Divergence loss from Line 9 and the MSE loss identified in Line 5.

Algorithm 1: Proposed SDnnKLFCM

```
   Input: $X = x_1, x_2, \ldots, x_N$, $N_c$, $m$, $\epsilon$;
   Output: $\mu, c$
 1 Sample: $X$ into $N/N_b$ batches, and train batch of
     training data using $X_r, r = 1, 2, \ldots, N/N_b$ ;
 2 Initialize weights $W$ and biases $B$ to pretrain the
     AE. Initialize cluster center $c$;
 3 while (epoch $\leq$ MaxIter) do
 4     for (batch $\in$[1,N/$N_b$]) do
 5         Calculate the hidden features $Z_r$, the
             reconstruction $h_{W,B}(X_r)$ and the MSE
             loss with Equation (2).
 6         Calculate $\mu_r$ ( membership degrees) and $c'$
             (cluster centers) using Equation (7) and
             (8), respectively.
 7         $\mu_r, c = KSLFCM(Z_r, c, m, \epsilon)$ [7], here,
             $K$ is the Cauchy kernel.
 8         Calculate the target $P_r$ by Equation (4).
 9         Calculate the $KL$ loss by Equation (5).
10         Calculate total loss by Equation (6) and
             backpropagate.
```

3.2. Scalable Kernelized Deep Neural Network Random Sampling Iterative Optimization FCM (SDnnKRSIO-FCM)

The SDnnKRSIO-FCM is the kernelized version of SDnnRSIO-FCM [10]. The SDnnKRSIO-FCM is implemented on the Apache Spark cluster in the HPC environment. The proposed algorithm SDnnKRSIO-FCM is summarized in Algorithm 2. The number of batches that NN will divide the dataset into is determined in Line 1. After that, cluster nodes are initialized and the Normal Autoencoder is pre-trained on the data using a learning rate (= 0.001) for 100 iterations. With a learning rate of 0.001, the initial training now begins. For each iteration and batch, layers from the network encoder section are used to create the hidden representation Z_r. The MSE loss is then computed using this format, and data clustering is performed. One of the parameters given to KSLFCM for the initial batch was pre-initialized cluster centers. The KSLFCM is a kernelized version of scalable FCM algorithms [7]. The input cluster centers for the second batch are taken from the first batch output cluster centers. Updated output cluster centers were made by merging all previous membership values used as input cluster centers for the remaining batches. To determine the desired distribution for membership values, the Student's-t distribution is employed. The values of the goal distribution and current distribution are used to calculate the clustering loss for the particular batch in the line that follows. The NN weights are finally changed using Stochastic Gradient Descent on Line 16. It contains the KL Divergence loss and the MSE loss identified in Line 7.

4. EXPERIMENTAL RESULTS

In the experiments, we evaluate the performance of SDnnKLFCM and SDnnKRSIO-FCM in comparison with SDnnLFCM and SDnnRSIO-FCM by utilizing various measures such as Normalized Mutual Information (NMI), Adjusted Rand index (ARI), and F-score, respectively.

4.1. Experimental Environment and dataset description

The experimentation employed two supercomputers: PARAM Shakti and PARAM Siddhi-AI[1] provided by National Supercomputing Mission (NSM) from IIT Kharagpur and CDAC-Pune India. The Apache Spark cluster framework is utilized to compute proposed deep fuzzy clustering algorithms. We computed results on three Benchmark datasets[2]: MNIST, Fashion MNIST, and QMNIST for our experiments by testing our proposed SDnnKLFCM and SDnnKRSIO-FCM in comparison with SDnnLFCM and SDnnRSIO-FCM algorithms. We have replicated all three datasets to work with Big Data. Table 1 demonstrates features of these datasets used for experimentation.

4.2. Evaluation Criteria

The NMI, ARI, and F-score measures are used to evaluate the proposed scalable kernel-based deep fuzzy clustering

1. https://en.wikipedia.org/wiki/PARAM
2. https://www.kaggle.com/datasets

Algorithm 2: Proposed SDnnKRSIO-FCM

```
Input: $X = x_1, x_2, \ldots, x_N$, number of clusters
       $N_c$, $m, \epsilon$
Output: $\mu, c$
1  Sample: $X$ into $N/N_b$ batches.
2  Train each batch of training data and represent it
     as $X_r, r = 1, 2, \ldots, N/N_b$;
3  Initialize weights $W$ and biases $B$ to pretrain the
     AE.
4  Initialize cluster centers $c$.
5  while (epoch $\leq$ MaxIter) do
6      for (batch $\in$[1,N/$N_b$]) do
7          Calculate the hidden features $Z_r$, the
             reconstruction $h_{W,B}(X_r)$ and the MSE
             loss with Equation (2).
8          if batch is 1 then
9              Calculate the memberships $\mu_r$ and
                 cluster centers $c'$ by using Equation (7)
                 and (8), respectively.
10             $\mu_r, c' = KSLFCM(Z_r, c, m, \epsilon)$ [7],
                 here, $K$ is the Cauchy kernel.
11         else
12             $\mu_r, c' = KSLFCM(Z_r, c', m, \epsilon)$
13             Aggregate membership degrees
                 obtained from previous batches.
14             Calculate new cluster centers.
15             $c' = \frac{\sum_{i=1}^{N} \mu'_{ij} x_i}{\sum_{i=1}^{N} \mu'_{ij}}, \forall j.$
16         Calculate the target $P_r$ by Equation (4),
             $KL$ loss by Equation (5), total loss by
             Equation (6) and backpropagate;
```

TABLE 1. Datasets descriptions.

Datasets	#Training instances	#Testing instances	#Features	#Classes
MNIST	60,000	10,000	784	10
Fashion-MNIST	60,000	10,000	784	10
QMNIST	60,000	60,000	784	10

methods on Big Data. The NMI computes the clustering results, and a comparison to the ground truth is performed using the results obtained from clustering algorithms. In this case, the class would stand for the absolute truth, and the cluster would be the result of the clustering method [15]. ARI is a measure of the degree to which two data clusters are comparable. To avoid erroneous positive results, ensure that the maximum value in each column is set to 1 and that all other values are set to 0 [15]. F-scores indicate clustering output correctness. Class cluster precision and recall are calculated. Higher F-scores indicate better grouping. Clustering findings match ground truth when F-score is approaching 1. The weighted total of class maximum F-scores determines the overall F-score [15].

4.3. Results and Discussion

In this section, we present the discussion on results of SDnnLFCM, SDnnRSIO-FCM, SDnnKLFCM, and SDnnKRSIO-FCM algorithms evaluated on three benchmark datasets, MNIST, Fashion-MNIST, and QMNIST, and do the comparative analysis on the performances. Comparison is done on the basis of NMI, ARI, and F-score measures. The algorithms are trained on the training dataset and then the results are calculated on the test dataset for every dataset.

Table 2 we have listed the results for the MNIST dataset on the respective performance measures: NMI, ARI, and F-score. All these measures represent the quality of the cluster obtained from the SDnnLFCM, SDnnRSIO-FCM, SDnnKLFCM, and SDnnKRSIO-FCM algorithms. The NMI measure of the proposed SDnnKRSIO-FCM is better than the SDnnRSIO-FCM. However, the proposed SDnnKLFCM and SDnnLFCM are significantly closer. The ARI measure of SDnnKRSIO-FCM is better than SDnnRSIO-FCM. The F-score measure for the proposed SDnnLFCM is better than SDnnLFCM and SDnnRSIO-FCM.

Table 3 we have listed the results for the Fashion-MNIST dataset on the respective performance measures. The NMI measure of the proposed SDnnKLFCM is better than the SDnnLFCM and SDnnRSIO-FCM. The ARI measure of the proposed algorithms is significantly closer to the earlier developed algorithms. The F-score measure for the proposed SDnnLFCM and SDnnKRSIO-FCM is far better than SDnnLFCM and SDnnRSIO-FCM. The proposed algorithms outperform in terms of F-score for the Fashion MNIST dataset.

Table 4 we have listed the results for the QMNIST dataset on the respective performance measures. The NMI measure of the proposed SDnnKLFCM and SDnnKRSIO-FCM are significantly closer. The ARI measure of the proposed algorithms is significantly closer to the earlier developed algorithms. The F-score measure for the proposed SDnnLFCM and SDnnKRSIO-FCM is significantly better than SDnnLFCM and SDnnRSIO-FCM.

TABLE 2. Results of SDnnLFCM, SDnnRSIO-FCM, SDnnKLFCM, and SDnnKRSIO-FCM on MNIST Dataset.

Algorithm	Measures		
	NMI	ARI	F-score
SDnnLFCM	0.1809	0.1071	0.0657
SDnnRSIO-FCM	0.0943	0.03902	0.0748
SDnnKLFCM	0.1535	0.0583	0.1438
SDnnKRSIO-FCM	0.1795	0.0908	0.0423

TABLE 3. Results of SDnnLFCM, SDnnRSIO-FCM, SDnnKLFCM, and SDnnKRSIO-FCM on Fashion-MNIST Dataset.

Algorithm	**Measures**		
	NMI	ARI	F-score
SDnnLFCM	0.1584	0.0378	0.0735
SDnnRSIO-FCM	0.1852	0.0594	0.0708
SDnnKLFCM	0.2124	0.0575	0.1526
SDnnKRSIO-FCM	0.1217	0.0343	0.1229

TABLE 4. Results of SDnnLFCM, SDnnRSIO-FCM, SDnnKLFCM, and SDnnKRSIO-FCM on QMNIST Dataset.

Algorithm	**Measures**		
	NMI	ARI	F-score
SDnnLFCM	0.0687	0.0324	0.0908
SDnnRSIO-FCM	0.1609	0.0705	0.0576
SDnnKLFCM	0.0012	-3.4638	0.0998
SDnnKRSIO-FCM	0.1414	0.0704	0.0932

5. Conclusion

This paper proposes SDnnKRSIO-FCM and SDnnLFCM clustering algorithms for handling Big Data efficiently. In the proposed algorithms, the Cauchy kernel is utilized as the kernel function for processing non-linear data, and the kernelized FCM algorithm is used as an integral part of the proposed algorithms. The data were reduced to lower dimensions using AE, and then clustering was applied to them. For Big Data fuzzy clustering issues, Apache Spark implemented the proposed algorithms on an HPC machine. SDnnKRSIO-FCM partitioned huge data into several batches and utilized AE to convert higher-dimensional data into lower-dimensional representations. Then the clustering of lower-dimensional data is performed in parallel. Here, the membership matrix is not stored during clustering, which speeds up processing and saves storage space. In order to calculate the next result, the SDnnKRSIO-FCM algorithm makes use of all previous iterations. Additionally, it updates the AE in order to get a better representation of the data clustered in the lower dimension. In the future, scalable deep fuzzy clustering algorithms can be applied to genome sequences for the identification of disease and drought in soybean plants.

Acknowledgments

This work is supported by the National Supercomputing Mission, HPC Applications Development Funded Research Project by DST in collaboration with the Ministry of Electronics and Information Technology (MeiTY), India.

References

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

[2] S. Agarwal, S. Yadav, and K. Singh, “Notice of violation of ieee publication principles: K-means versus k-means++ clustering technique,” in *2012 Students Conference on Engineering and Systems*. IEEE, 2012, pp. 1–6.

[3] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[4] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” *The annals of statistics*, pp. 1171–1220, 2008.

[5] A. Ng, M. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” *Advances in neural information processing systems*, vol. 14, pp. 849–856, 2001.

[6] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.

[7] P. Jha, A. Tiwari, N. Bharill, M. Ratnaparkhe, M. Mounika, and N. Nagendra, “A novel scalable kernelized fuzzy clustering algorithms based on in-memory computation for handling big data,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 6, pp. 908–919, 2020.

[8] Y.-P. Zhao, L. Chen, and C. P. Chen, “Multiple kernel shadowed clustering in approximated feature space,” in *International Conference on Data Mining and Big Data*. Springer, 2018, pp. 265–275.

[9] Q. Feng, L. Chen, C. P. Chen, and L. Guo, “Deep fuzzy clustering—a representation learning approach,” *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 7, pp. 1420–1433, 2020.

[10] P. Jha, A. Tiwari, N. Bharill, M. Ratnaparkhe, O. P. Patel, V. Anand, S. Arya, and T. Singh, “Hpc enabled a novel deep fuzzy scalable clustering algorithm and its application for protein data,” in *2022 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2022, pp. 1–8.

[11] A. dos SP Soares, W. D. Parreira, E. G. Souza, S. J. de Almeida, C. M. Diniz, C. D. Nascimento, and M. F. Stigger, “Energy-based voice activity detection algorithm using gaussian and cauchy kernels,” in *2018 IEEE 9th Latin American Symposium on Circuits & Systems (LASCAS)*. IEEE, 2018, pp. 1–4.

[12] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[13] J. Xie, R. Girshick, and a. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[14] N. Bharill, A. Tiwari, and A. Malviya, “Fuzzy based scalable clustering algorithms for handling big data using apache spark,” *IEEE Transactions on Big Data*, vol. 2, no. 4, pp. 339–352, 2016.

[15] M. Rezaei and P. Fränti, “Set matching measures for external cluster validity,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 2173–2186, 2016.