Cloud Implementation of Extreme Learning Machine for Hyperspectral Image Classification

Juan M. Haut, *Senior Member, IEEE*, Sergio Moreno-Álvarez, *Graduate Student Member, IEEE*, Enrique Moreno-Ávila, Victor A. Ayma, *Member, IEEE*, R. Pastor-Vargas, *Senior Member, IEEE*, and Mercedes E. Paoletti, *Senior Member, IEEE*

***Abstract*— Classifying remotely sensed hyperspectral images (HSIs) became a computationally demanding task given the extensive information contained throughout the spectral dimension. Furthermore, burgeoning data volumes compound inherent computational and storage challenges for data processing and classification purposes. Given their distributed processing capabilities, cloud environments have emerged as feasible solutions to handle these hurdles. This encourages the development of innovative distributed classification algorithms that take full advantage of the processing capabilities of such environments. Recently, computational-efficient methods have been implemented to boost network convergence by reducing the required training calculations. This letter develops a novel cloud-based distributed implementation of the extreme learning machine (`CC-ELM`) algorithm for efficient HSI classification. The proposal implements a fault-tolerant and scalable computing design while avoiding traditional batch-based backpropagation. `CC-ELM` has been evaluated over state-of-the-art HSI classification benchmarks, yielding promising results and proving the feasibility of cloud environments for large remote sensing and HSI data volumes processing. The code available at https://github.com/mhaut/scalable-ELM-HSI.**

***Index Terms*— Cloud computing, distributed computing, hyperspectral imaging, machine learning.**

I. Introduction

TECHNOLOGICAL advances in remote sensing (RS) have enabled a burgeoning collection of high-quality hyperspectral images (HSIs). These are captured by imaging spectrometers and radiometers, which record an assemblage of light-reflection and electromagnetic-radiation measurements from the Earth's surface [1]. Collected scenes are represented as 3-D data cubes, generally with hundreds of spectral channels (continuous correlated bands), where each pixel defines a spectral signature that identifies the materials inside the observed region [2]. This spectral information based on visible, near-infrared, and short-wave infrared spectral range is used for a wide range of applications, including agriculture [3], urban planning, and forest management [4].

RS data are a sophisticated data form given its high variability and dimensionality. Consequently, HSI is characterized by complex data structures and features, compounding computing hurdles and posing new challenges to analyze and interpret such massive collections. These challenges exacerbate the data requirements caused by the significant scarcity of available training data when compared to natural scenes. In this regard, factors, such as limited sensor imaging and the obstacle in labeling, play a crucial role. For instance, in machine learning (ML) algorithms, a sufficient number of labeled samples are required for supervised training. To tackle this issue, the concept of few-shot learning has been proposed in [5]. On the other hand, scalability needs to be ensured when facing with such large spectral features, substantial volumes of ever-expanding data, and computational requirements [6], [7]. In this context, parallel and distributed approaches are considered the most interesting solutions, highlighting high-performance computing (HPC) [8] and cloud computing (CC) technologies [9]. CC, in particular, provides services via an online platform consisting of processing and storage equipment situated in remote clusters. Its main advantages are: 1) flexibility to scale resources without the need to deploy physical computing infrastructures; 2) robust backup and protection procedures to ensure data security and preservation; 3) efficient economic investment to prevent computing overbuilding and overprovisioning; and 4) capacity for better collaboration and availability of better access to the resources needed for a computation. Among others, Amazon Web Services (AWS) and Microsoft Azure have lowered the cost of accessing and using such technologies [10].

Cloud approaches rely on efficient distributed programming implementations and are designed on popular frameworks such as Apache Hadoop [11] or Spark [12], where the MapReduce model [13] is widely used to handle data management. Its operation is straightforward and intuitive. Two functions are

Manuscript received 31 January 2023; revised 20 June 2023; accepted 6 July 2023. Date of publication 20 July 2023; date of current version 27 July 2023. This work was supported in part by the European Regional Development Fund "A way to do Europe" (FEDER), Junta de Extremadura, under Grant IB20040; in part by the Consejería de Economía, Ciencia y Agencia Digital (Junta de Extremadura); in part by the Open access funding by the Universidad Nacional de Educacion a Distancia (UNED); in part by the Ministry of Science and Innovation (APRISA) under Project PID2019-110315RB-I00/AEI/10.13039/501100011033; and in part by the Junta de Extremadura FEDER under Grant GR21040 and Grant GR21099. *(Corresponding author: R. Pastor-Vargas.)*

Juan M. Haut and Mercedes E. Paoletti are with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, 10003 Cáceres, Spain.

Sergio Moreno-Álvarez is with the Department of Computer Systems Engineering and Telematics, Escuela Politécnica, University of Extremadura, 10003 Cáceres, Spain.

Enrique Moreno-Ávila is with the Department of Technology of Computers and Communications, University of Extremadura, 10003 Cáceres, Spain.

Victor A. Ayma is with the Department of Engineering, Universidad del Pacífico, Lima 15072, Peru.

R. Pastor-Vargas is with the Departamento de Sistemas de Comunicación y Control, Escuela Técnica Superior de Ingenieria Informática, Universidad Nacional de Educación a Distancia (UNED), 28040 Madrid, Spain (e-mail: rpastor@scc.uned.es).

Digital Object Identifier 10.1109/LGRS.2023.3295742

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

performed using the datasets organized as key–value pairs. First, the map function generates a set of key–value pairs for a particular task and splits it into parallel subtasks. The reduce function then collects all of the results from the parallel subtasks for a specified key. Furthermore, and related to Apache Hadoop framework, Apache Pig [14] has emerged as a compelling platform for abstracting the operation of MapReduce at a high level. Based on data flow instructions, it uses the high-level, easy-to-use Pig Latin language, which incorporates multiquery support for user-defined functions (UDFs).

Progress in CC has enabled the deployment of ML methods for big data processing, reaping the benefits of the data parallelization and distribution capabilities provided by Apache Hadoop, Spark, and Apache Pig. ML methods identify trends and patterns in large volumes of data to solve complex problems, yielding predictions that support better decision-making [15]. ML approaches are widely used in various applications for the analysis of remotely sensed HSI data. Due to its fast convergence and easy design, the extreme learning machine (ELM) [16] is a very popular solution, as it was designed for training single hidden layer feedforward neural networks (SLFNs) to avoid the iterative process of traditional artificial neural networks (ANNs). Several alternatives have been developed based on the fundamental principles of ELM for applications in image processing [17]. Notwithstanding the promising results, improving predictions requires large amounts of data, which compounds the mathematical complexity of their operation (in relation to the inverse matrix calculation) and imposes considerable challenges. In this respect, CC approaches offer viable solutions to overcome such problems. Nevertheless, these technologies have not been adequately exploited for HSI processing purposes.

A. Fundamentals of ELM

ELM is designed to train an SLFN classifier by avoiding the traditional iterative procedure. Thus, ELM provides a fast learning speed. Its operation is similar to ANN-based approaches, where parameters (weights and biases) are initialized randomly and adjusted by training. Nevertheless, parameters do not require fine-tuning by backpropagation iterating.

Considering the data $\mathbf{X} \in \mathbb{R}^{M\times S}$, with $M = H \cdot W$ and S spatial and spectral dimensions, respectively (i.e., M training samples), the ELM takes the spectral features of the mth sample $\mathbf{x}_m = [x_{m,1}, \dots, x_{m,S}] \in \mathbb{R}^S$ as inputs nodes and produces the corresponding output $\mathbf{o}_m = [o_{m,1}, \dots, o_{m,Z}] \in \mathbb{R}^Z$, by applying the SLFNN transformation given in (1), where $\delta(\cdot)$ denotes the activation function, $\boldsymbol{\omega} \in \mathbb{R}^{S\times N}$ and $\boldsymbol{\beta} \in \mathbb{R}^{N\times Z}$ are the hidden and output weights, respectively, and $\mathbf{b} \in \mathbb{R}^N$ is the hidden vector bias

$$o_{m,t} = \sum_{i=1}^{n} \beta_{i,t}\delta\left(b_i + \sum_{j=1}^{s} \omega_{j,i} x_{m,j}\right) \quad \forall t \in [1, z]. \tag{1}$$

Considering the M training samples, this is reformulated as

$$f(\mathbf{X}) = \mathbf{O} = \mathcal{H}(\mathbf{X}, \boldsymbol{\omega}, \mathbf{b})\boldsymbol{\beta} \tag{2}$$

Algorithm 1 ELM-UDF Structure

```
 1: Require: Initialization settings and $\mathbf{X}^k$.
 2:  Configure ELM.
 3:  Buffer $\mathbf{X}^k$ in local memory (S3 $\longrightarrow$ EC2).
 4: ▷ Training step:
 5:      Compute $\mathcal{H}^k$ and $\mathcal{H}^{\dagger k}$.
 6:      Calculate output weights $\boldsymbol{\beta}^k$.
 7:      Average global output weights $\boldsymbol{\beta}$.
 8:      return $\boldsymbol{\beta}$.
 9: ▷ Validation step:
10:      Process validation $\mathbf{X}^k$.
11:      return ELM classification accuracy (%).
```

where $\mathcal{H} \in \mathbb{R}^{M\times N} = \delta(\mathbf{b} + \mathbf{X}\cdot\boldsymbol{\omega})$ is the invertible hidden layer output matrix. The ELM adjusts the set of $\boldsymbol{\beta}$ to minimize the cost function $\min_{\boldsymbol{\beta}} ||\mathcal{H}\boldsymbol{\beta} - \tilde{\mathbf{O}}||^2$, with $\tilde{\mathbf{O}}$ the target output. The optimal solution is determined by $\boldsymbol{\beta} = \mathcal{H}^\dagger\tilde{\mathbf{O}}$, where $\mathcal{H}^\dagger$ is the Moore–Penrose pseudoinverse. This prevents iterative parameter tuning, reducing computational time. As a result, with enough hidden nodes N, the ELM can approximate any function $f : \mathbb{R}^S \to \mathbb{R}^Z$.

Focusing on $\mathcal{H}^\dagger$, singular value decomposition (SVD) is used to conduct the pseudoinverse of nonsquare matrices by orthogonal transformations

$$U, \Sigma, V^T = \mathrm{SVD}(\mathcal{H}) \tag{3a}$$

$$\mathcal{H}^\dagger = D^T \cdot \left(DD^T\right)^{-1} \cdot \left(C^T C\right)^{-1} \cdot C^T \tag{3b}$$

where $U = \mathcal{H}\mathcal{H}^T$ and $V = \mathcal{H}^T\mathcal{H}$ are the orthogonal matrices of eigenvectors, Σ contains the diagonal squared eigenvalues, and $C = U(\Sigma)^{1/2}$ and $D = (\Sigma)^{1/2}V^T$ are the respective factorization.

Although this method avoids costly optimization procedures, it becomes memory-intensive when dealing with large data volumes. Nevertheless, the fast convergence of ELM during training provides remarkable performance for classification, clustering, or regression tasks, encouraging further work on its stability, performance, and accuracy. Recent research [18] has identified fluctuation issues in classification, providing alternative methods such as automated modification of the number of nodes in the hidden layer, set learning, or voting procedures, among others. Also, imbalanced data scenarios negatively affect the performance, particularly due to the influence of minority classes. To address this challenge, the use of weighting averaging methods has been employed as a mitigation strategy [19]. Finally, pseudoinverse optimization has also been investigated, and efforts to improve computational performance have been highlighted. However, the issue of scalability has not been addressed.

B. Contributions and Limitations of This Work

To afford the scalability problem, this work presents a new CC-based implementation of the ELM for massive HSI classification. This approach, called cloud-based distributed implementation of the ELM (`CC-ELM`), provides a solution for processing the increasing amounts of existing HSI data, as well as for the inherent complexity of HSI. The advantages of using CC as a distributed platform are combined with the versatility

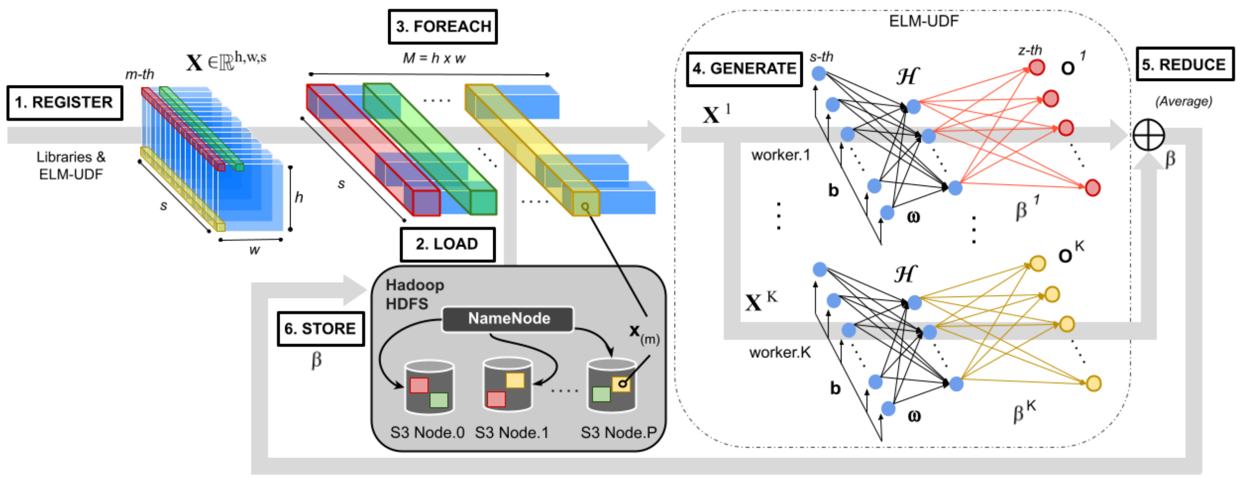


Fig. 1. Apache Pig execution pipeline for the training step. Different colors represent the K machines. The implementation of the ELM in Java is provided through UDF. Each pixel is represented as $m \in [1, M]$. The number of S3 nodes is represented as P. The kth worker processes its local data subset $\mathbf{X}^k$.

offered by Apache Pig to create new UDF functions, enabling the development of new functions in an accessible and cost-effective deployment. Accordingly, high levels of scalability are ensured while maintaining classification accuracy. Furthermore, this work demonstrates that, given small amounts of data, classification is affected by the randomness introduced in the selection of the training samples. This randomness has a large impact as the training is performed in a single iteration. In this regard, the proposal is intended to handle large amounts of data.

II. CC-ELM: Efficient Distributed Approach

This section describes the novel distributed procedure of the ELM (CC-ELM), implemented in a primary/secondary approach to work in a multinode environment. In this approach, the Apache Pig platform (high-level programming language for processing large data sets) has been adopted as the processing model and the Apache Hadoop framework (open-source software used for efficient and scalable distributed computing) as the distributed programming model.

A. Distributed-Parallel Architecture

Apache Pig is built on top of Apache Hadoop and translates MapReduce instructions into efficient execution tasks within a distributed environment. In this regard, Pig Latin extends language capabilities by creating UDFs [20], while Apache Hadoop enables the parallel processing of large volumes of data and provides an optimized storage system, the Hadoop Distributed File System (HDFS). HFDS includes fault-tolerant capabilities and high availability through data replication over deployed machines, enhancing the processing speed when using parallelization techniques.

B. Overview of the Proposal Operation

The ELM obtains the Moore–Penrose pseudoinverse through SVD. The proposed implementation manages the high-computing requirements of this calculation for HSI classification in four steps.

1) Configuration Step: The input data are split into random partitions (subsets) of $\tilde{M} < M$ samples, $\mathbf{X}^k \in \mathbb{R}^{\tilde{M}\times S} = [\mathbf{x}_1, \ldots, \mathbf{x}_{\tilde{M}}]^T$, and distributed among the K workers.

2) Moore–Penrose Step: Each worker implements a copy of the SLFN architecture. Random initialization of input weights and bias is conducted using the same seed among the secondary, ensuring that the CC-ELM initial trained model is the same for all workers. Then, each worker performs the SLFN transformation to obtain its $\mathcal{H}^k$ and computes its Moore–Penrose pseudoinverse $\mathcal{H}^{\dagger k}$ locally.

3) Beta Step: This step represents the most computationally intensive process of the approach, involving workers computing its output weights through $\boldsymbol{\beta}^k = \mathcal{H}^{\dagger k}\mathbf{O}^k$. Obtained weights are collected and reduced to obtain the final output weights $\boldsymbol{\beta}$, which are stored in an Amazon Simple Storage Service (S3) repository.

4) Validation Step: Once the final $\boldsymbol{\beta}$ is available, each worker validates the model during the test.

The implementation of the distributed CC-ELM algorithm is built as a UDF to be processed through Apache Pig. In this context, Algorithm 1 describes the ELM-UDF structure. Its design follows the previous steps in an orderly manner. As a requirement, the initialization settings for the ELM model and the HSI subset are provided as a set of tuples. Then, each data subset $\mathbf{X}^k$ is loaded in the corresponding worker to be used in the training of the ELM. Also, the ELM is configured with the provided settings, such as the number of nodes of the different layers and seeds for the parameters initialization, among others. For the training step, no additional configuration is required, following the procedure of the algorithm in Section I-A. Finally, during the validation step, each worker retrieves back $\boldsymbol{\beta}$ to provide the corresponding classification on its $\mathbf{X}^k$ local subset. Fig. 1 shows the training procedure of the proposal. The ELM-UDF is defined and implemented via a Pig Latin script that abstracts data distribution and oversees the entire classification process of the proposed method. Fig. 1 shows the training steps performed by the Pig Latin script. The execution encompasses six steps. The REGISTER step ensures the inclusion of all requisite libraries. Subsequently, the LOAD steps retrieve the global HSI $\mathbf{X}$ dataset from S3. The FOREACH step facilitates the distributed execution of the ELM-UDF at each worker node and GENERATE the outputs $\boldsymbol{\beta}^k$. Finally, the REDUCE and STORE instructions combine the outputs $\boldsymbol{\beta}^k$ and store in S3 the resulting $\boldsymbol{\beta}$. This procedure simplifies the complexity of MapReduce programming into intuitive lines of code.

III. Experiments

The experiments have been conducted by the AWS resources provided by Elastic Compute Cloud (EC2). Amazon S3 is used to store the data, while Amazon Elastic MapReduce (EMR) service is used as the managed cluster platform for data processing with Hadoop (2.10.1) + Pig (0.17.0). Several experiment configurations have been launched in the distributed environment, concretely, for two, four, eight, 16, 32, and 64 nodes. The computing nodes used are the *m5.xlarge*[1] instances, comprising an Intel Xeon Platinum 8175 3.1-GHz CPU and 16-GB RAM each one. In addition, the ELM baseline method (named L) is executed locally.

A. Dataset Description

The classification is performed for the Big Indian Pines (BIP) dataset. This complex dataset was collected by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor, capturing the land surface of the Northern Indiana Pines and gathering the solar reflected spectrum from 400 to 2500 nm. The image size is 2678 $\times$ 614 pixels with 224 spectral bands that contain the information about 58 classes. For scaling purposes and addressing the scarcity of training samples, experiments have been launched with different sizes: 1) 5%; 2) 10%; 3) 20%; 4) 40%; 5) 60%; and 6) 80% of training samples.

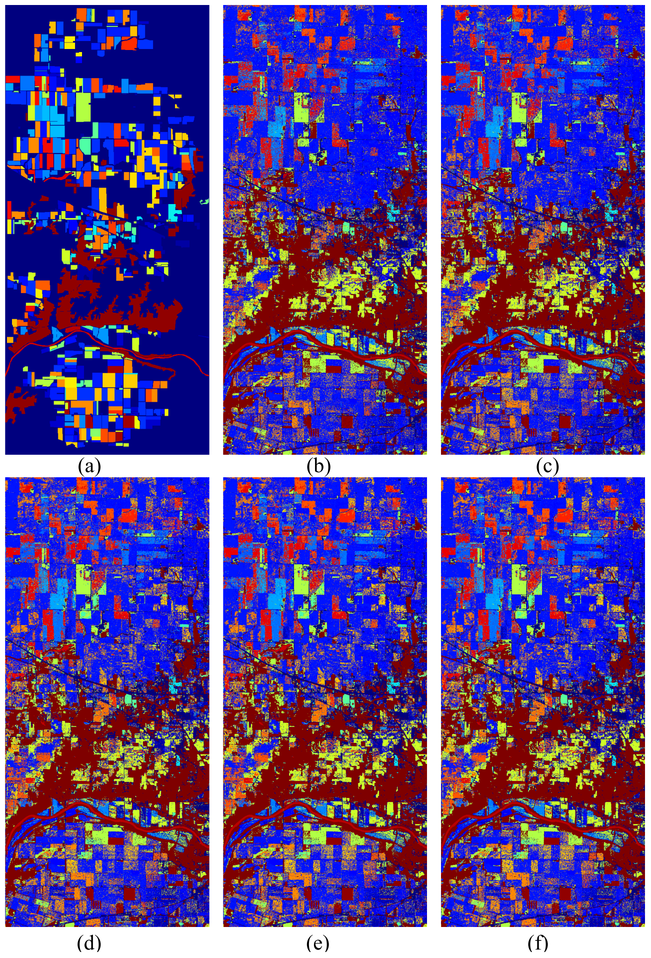


Fig. 2. Classification maps using $K = 64$ workers for different data sizes. The accuracy is displayed within parentheses. (a) GT. (b) 10% (44.43%). (c) 20% (47.56%). (d) 40% (51.58%). (e) 60% (51.56%). (f) 80% (51.46%).

B. Experimental Results

The experiments are divided into two different branches. The first one is the analysis of the HSI classification accuracy. The second one analyzes the scaling properties of the distributed approach. The results of the former are detailed in Table I. It should be noted that the more nodes, the less memory limitations, so from 16 nodes upward, larger training sets can be tested. This also reveals two important facts. The first one is determined by the limited training samples, where the randomness introduced in the selection of data in the smaller proportions of the BIP dataset (10%) significantly impairs the accuracy obtained by the classifier. The utilization of such small portions of data, primarily composed of background pixels, leads to a constrained representation of other land-cover classes. Nevertheless, this shortcoming is overcome for larger amounts of data, where the accuracy reaches similar values to the baseline algorithm, i.e., the local implementation. In this regard, the proposed `CC-ELM` classification is successful when dealing with large amounts of data. In addition, the second observable point is the positive trend toward increasing accuracy as the training size increases. As a matter of fact, increasing the number of workers K has no negative influence on the accuracy attained compared with the local L counterpart results. This is a significant strength of the proposal, as the larger the data distribution, the lower the accuracy attained, which is hampered by the limited number of training data available in each worker node. As regards the study of accuracy, the experiments with 5% have been considered irrelevant due to the low amount of training data. Achieving high classifier performance is particularly complex due to the spectral mixing of the BIP scene. Also, this dataset causes overfitting given the large amount of background samples and pixel similarity. This is clearly shown in Fig. 2, which depicts a clear trend: as the data size increases, a significant reduction in the misclassification of background pixels is achieved. The classification algorithm shows improved accuracy in correctly identifying and assigning landscape classes to these pixels.

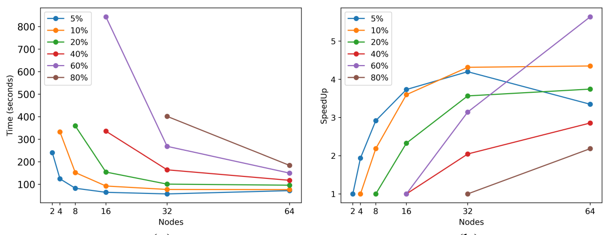
The second experiment focuses on the evaluation of the scaling of the proposal. It should be noted that the number of training samples has been adjusted to the number of nodes due to memory constraints. Thus, training with 5% of the data is feasible in environments with two, four, eight, 16, 32, and 64 nodes, whilst training with 80% of the data is only feasible in environments with 32 and 64 nodes. Fig. 3(a) shows that the

[1]Details for EC2 in https://aws.amazon.com/es/ec2/instance-types/m5/

TABLE I

SUMMARY OF DIFFERENT ACCURACY RESULTS OBTAINED FOR THE FIRST EXPERIMENT. COLUMNS CORRESPONDING TO THE AMOUNT OF DATA. K IS THE NUMBER OF WORKERS AND L IS THE LOCAL EXECUTION

K	**10%**	**20%**	**40%**	**60%**	**80%**
4	50.52 ±0.01	-	-	-	-
8	50.47 ±0.05	50.89 ±0.06	-	-	-
16	50.06 ±0.05	51.06 ±0.06	51.22 ±0.04	51.02 ±0.06	-
32	44.90 ±0.01	51.05 ±0.01	51.46 ±0.01	51.35 ±0.01	51.21 ±0.01
64	44.43 ±0.10	47.56 ±0.11	51.58 ±0.04	51.56 ±0.11	51.46 ±0.09
L	48.79 ±0.48	50.4 ±0.51	51.4 ±0.42	51.53 ±0.05	51.65 ±0.03



(a) (b)

Fig. 3. Scaling experiments are represented in time at the left and speedup at the right. The baseline for each data distribution is given by the first execution. (a) Runtime. (b) SpeedUp.

execution time decreases as the number of workers increases. In this context, the workers train disjoint data portions and successfully share the computational load. Therefore, the run time increases with data size and reduces with the number of workers, as expected. The speedup in Fig. 3(b) has been calculated for each training percentage experiment with a different number of nodes (considering memory limitations): for 5% of data, the slowest time (i.e., the base case) is provided with two nodes; for 10%, the base case is four nodes; for 20%, the base case is eight nodes; for 40% and 16%, the base case is 16 nodes; and for 80%, the slowest time is provided by 32. Fig. 3(b) shows a remarkable improvement in the obtained speedup, in particular with 40%, 60%, and 80% of training samples. The speedup assessment employs a baseline speedup, determined by the initial node configuration, for each dataset size. The findings reveal a consistent upward trend in speedup across all experiments until the maximum scaling is attained.

IV. CONCLUSION

In this letter, a novel CC approach is presented to improve the computational efficiency of a popular ML algorithm, known as ELM. The key point of the proposed methodology is the distribution and parallel processing of large data volumes for classification purposes. In this regard, the pixel classification of remotely sensed imagery, i.e., HSI, becomes especially challenging given the complexity of the spatial and spectral information. This fact is accentuated in ELM, where the entire data are processed in one single training step. Consequently, the memory of the hardware devices suffers overloads. To this end, this letter proposes a methodology to address this problem. Our distributed ELM implementation offers good performance for both classification and scaling purposes. The proposal offers an optimal solution using available and easily manageable commercial hardware through AWS. The methodology is versatile and flexible to develop future functions to address computationally expensive operations or high memory requirements. We found that CC offers good computational performance for scaling purposes. As future work, we intend to apply this procedure to complex ML algorithms.

REFERENCES

[1] J. Segarra, M. L. Buchaillot, J. L. Araus, and S. C. Kefauver, "Remote sensing for precision agriculture: Sentinel-2 improved features and applications," *Agronomy*, vol. 10, no. 5, p. 641, 2020.

[2] P. E. Clark and M. L. Rilee, *Remote Sensing Tools for Exploration: Observing and Interpreting the Electromagnetic Spectrum*. Springer, 2010. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4419-6830-2

[3] M. Wójtowicz, A. Wójtowicz, and J. Piekarczyk, "Application of remote sensing methods in agriculture," *Commun. Biometry Crop Sci.*, vol. 11, no. 1, pp. 31–50, 2016.

[4] A. M. Lechner, G. M. Foody, and D. S. Boyd, "Applications in remote sensing to forest ecology and management," *One Earth*, vol. 2, no. 5, pp. 405–412, 2020.

[5] X. Sun, B. Wang, Z. Wang, H. Li, H. Li, and K. Fu, "Research progress on few-shot learning for remote sensing image interpretation," *IEEE J. Sel. Topics Appl. Earth Observat. Remote Sens.*, vol. 14, pp. 2387–2402, 2021.

[6] P. Duan, X. Kang, S. Li, P. Ghamisi, and J. A. Benediktsson, "Fusion of multiple edge-preserving operations for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 10336–10349, Dec. 2019.

[7] P. Duan, P. Ghamisi, X. Kang, B. Rasti, S. Li, and R. Gloaguen, "Fusion of dual spatial information for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 9, pp. 7726–7738, Sep. 2021.

[8] A. Plaza, Q. Du, Y. Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE J. Sel. Topics Appl. Earth Observat. Remote Sens.*, vol. 4, no. 3, pp. 528–544, Sep. 2011.

[9] J. M. Haut et al., "Cloud deep networks for hyperspectral image analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 9832–9848, Dec. 2019.

[10] B. Gupta, P. Mittal, and T. Mufti, "A review on Amazon web service (AWS), Microsoft Azure & Google cloud platform (GCP) services," in *Proc. 2nd Int. Conf. ICT Digit., Smart, Sustain. Develop. (ICIDSSD)*, Jamia Hamdard, India, 2021, pp. 27–28.

[11] J. Nandimath, E. Banerjee, A. Patil, P. Kakade, S. Vaidya, and D. Chaturvedi, "Big data analysis using Apache Hadoop," in *Proc. IEEE 14th Int. Conf. Inf. Reuse Integr. (IRI)*, Aug. 2013, pp. 700–703.

[12] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," *Int. J. Data Sci. Anal.*, vol. 1, no. 3, pp. 145–164, 2016.

[13] K. Shim, "MapReduce algorithms for big data analysis," in *Proc. Int. Workshop Databases Netw. Inf. Syst.* Springer, 2013, pp. 44–48.

[14] C. Swarna and Z. Ansari, "Apache Pig—A data flow framework based on Hadoop MapReduce," *Int. J. Eng. Trends Technol.*, vol. 50, no. 5, pp. 271–275, 2017.

[15] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, pp. 1–16, 2016.

[16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.

[17] M. A. A. Albadra and S. Tiuna, "Extreme learning machine: A review," *Int. J. Appl. Eng. Res.*, vol. 12, no. 14, pp. 4610–4623, 2017.

[18] J. Wang, S. Lu, S.-H. Wang, and Y.-D. Zhang, "A review on extreme learning machine," *Multimedia Tools Appl.*, vol. 81, no. 29, pp. 41611–41660, 2022.

[19] P. R. Bal and S. Kumar, "WR-ELM: Weighted regularization extreme learning machine for imbalance learning in software fault prediction," *IEEE Trans. Rel.*, vol. 69, no. 4, pp. 1355–1375, Dec. 2020.

[20] A. Gates and D. Dai, *Programming Pig: Dataflow Scripting With hadoop*. Sebastopol, CA, USA: O'Reilly, 2016.