INVITED PAPER

Distributed Deep Learning for Remote Sensing Data Interpretation

By Juan M. Haut, *Senior Member IEEE*, Mercedes E. Paoletti, *Senior Member IEEE*, Sergio Moreno-Álvarez, Javier Plaza, *Senior Member IEEE*, Juan-Antonio Rico-Gallego, and Antonio Plaza, *Fellow IEEE*

ABSTRACT | As a newly emerging technology, deep learning (DL) is a very promising field in big data applications. Remote sensing often involves huge data volumes obtained daily by numerous in-orbit satellites. This makes it a perfect target area for data-driven applications. Nowadays, technological advances in terms of software and hardware have a noticeable impact on Earth observation applications, more specifically in remote sensing techniques and procedures, allowing for the acquisition of data sets with greater quality at higher acquisition ratios. This results in the collection of huge amounts of remotely sensed data, characterized by their large spatial resolution (in terms of the number of pixels per scene), and very high spectral dimensionality, with hundreds or even thousands of spectral bands. As a result, remote sensing instruments on spaceborne and airborne platforms are now generating data cubes with extremely high dimensionality, imposing several restrictions in terms of both processing runtimes and storage capacity. In this article, we provide a comprehensive review of the state of the art in DL for remote sensing data interpretation, analyzing the strengths and weaknesses of the most widely used techniques in the literature, as well as an exhaustive description of their parallel and distributed implementations (with a particular focus on those conducted using cloud computing systems). We also provide quantitative results, offering an assessment of a DL technique in a specific case study (source code available: https://github.com/mhaut/cloud-dnn-HSI). This article concludes with some remarks and hints about future challenges in the application of DL techniques to distributed remote sensing data interpretation problems. We emphasize the role of the cloud in providing a powerful architecture that is now able to manage vast amounts of remotely sensed data due to its implementation simplicity, low cost, and high efficiency compared to other parallel and distributed architectures, such as grid computing or dedicated clusters.

KEYWORDS | Big data; cloud computing; deep learning (DL); parallel and distributed architectures; remote sensing.

NOMENCLATURE

EO	Earth observation.
GPUs	Graphics processing units.
CPUs	Central processing units.
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer.
AVIRIS-NG	Airborne Visible/Infrared Imaging Spectrometer New Generation.

Manuscript received January 22, 2021; revised February 23, 2021; accepted February 28, 2021. This work was supported in part by the Junta de Extremadura under Grant GR18060, in part by the Spanish Ministerio de Ciencia e Innovación under Project PID2019-110315RB-I00 (APRISA), in part by the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement 734541 (EOXPOSURE), and in part by the Computing Facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT) funded by the European Regional Development Fund (ERDF); CETA-CIEMAT belongs to CIEMAT and the Government of Spain. *(Corresponding author: Antonio Plaza.)*

Juan M. Haut is with the Department of Communication and Control Systems, Higher School of Computer Engineering, National Distance Education University, 28015 Madrid, Spain (e-mail: juanmariohaut@unex.es).

Mercedes E. Paoletti is with the Department of Computer Architecture, School of Computer Science and Engineering, University of Málaga, 29071 Málaga, Spain (e-mail: mpaoletti@unex.es).

Sergio Moreno-Álvarez and **Juan-Antonio Rico-Gallego** are with the Department of Computer Systems Engineering and Telematics, University of Extremadura, 10003 Cáceres, Spain (e-mail: smoreno@unex.es; jarico@unex.es).

Javier Plaza and **Antonio Plaza** are with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, 10003 Cáceres, Spain (e-mail: mpaoletti@unex.es; jplaza@unex.es; aplaza@unex.es).

Digital Object Identifier 10.1109/JPROC.2021.3063258

NASA	National Aeronautics and Space Administration.
TB	Terabyte.
GB	Gigabyte.
PB	Petabyte.
mpp	Meters per pixel.
EnMAP	Environmental Mapping and Analysis Program.
EOSDIS	Earth Observing System Data and Information System.
NSMC	China National Satellite Meteorological Center.
CCRSDA	China Center for Resources Satellite Data and Application.
ESA	European Space Agency.
GEO	Group on Earth Observations.
HPC	High-performance computing.
SAR	Synthetic aperture radar.
LIDAR	Light Detection and Ranging.
OLI	Operational land imager.
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer.
VNIR	Visible and near-infrared.
TIR	Thermal infrared.
SWIR	Shortwave infrared.
GSD	Ground sample distance.
PAN	Panchromatic.
MSIs	Multispectral images.
HSI	Hyper-spectral images.
PCA	Principal component analysis.
TCT	Tasseled cap transformation.
WT	Wavelet transform.
KNN	K-nearest neighbor.
DTs	Decision trees.
RFs	Random forests.
MLR	Multinomial logistic regression.
GMMs	Gaussian mixture models.
NB	Naive Bayes-based approaches.
HMMs	Hidden Markov models.
SVMs	Support vector machines.
ANNs	Artificial neural networks.
SAEs	Stacked autoencoders.
DBNs	Deep belief networks.
RNNs	Recurrent neural networks.
CNNs	Convolutional neural networks.
DNNs	Deep neural networks.
RBMs	Restricted Boltzmann machines.
LSTM	Long short-term memory.
RESFlow	Remote Sensing data Flow.
InSPIRE	Integrated Sustainable Pan-European Infrastructure for Researchers in Europe.
XSEDE	Extreme Science and Engineering Discovery Environment.
MTFC	Multi-GPU training framework.
PNPE	Parallel neighbor pixel extractor.
G-POD	Grid processing on demand.
GENESI-DR	Ground European Network for Earth Science Interoperations and Digital Repositories.
GiSHEO	Grid Services for training and High Education in Earth Observation.
CEOS	Committee on Earth Observation Satellites.
DGGS	Discrete global grid systems.
CBIR	Content-based image retrieval.
ICP	InterIMAGE cloud Platform.
HDFS	Hadoop file system.
YARN	Yet Another Resource Negotiator.
RDDs	Resilient distributed data sets.
VMs	Virtual machine.
SaaS	Software as a Service.
PaaS	Platform as a Service.
IaaS	Infrastructure as a Service.
API	Application programming interface.
SDNs	Software-defined networks.
DNs	Domain name servers.
GFS	Google File System.
DAG	Directed acyclic graph.
I/O	Input–output.
HPCC	High-performance computing challenger.
AWS	Amazon Web Services.
HPL	High-performance linpack.
EC2	Elastic computing cloud.
GCE	Google Compute Engine.
IBM SL	IBM SoftLayer.
MRAP	MapReduce with Access Patterns.
MPI	Message passing interface.
BLAS	Basic linear algebra subprogram.
ML	Machine learning.
DL	Deep learning.
OS	Operating system.
MAP	Maximum *a posteriori*.
BS	Block size.
MLP	Multilayer perceptron.
ReLU	Rectified linear unit.
BFGS	Broyden–Fletcher–Goldfarb–Shanno.
DGEMM	Double-precision matrix-matrix multiplication.
TF	TensorFlow.
PS	Parameter server.
IP	Indian Pines.
BIP	Big Indian Pines.
HDD	Hard disk drive.
RAM	Random access memory.
FLOPs	Floating-point operations.
PRACE	Partnership for Advanced Computing in Europe.
UCI	Unified cloud interface.

I. INTRODUCTION

A. Big Remote Sensing Data

Remotely sensed images provide very detailed information about the surface of the Earth, which often results

in very significant computational requirements. Present and future missions for EO have significantly increased the spatial, spectral, and temporal resolutions of existing imaging instruments, resulting in higher data volumes. Meanwhile, the variety of multisensor and multiresolution acquisitions inevitably leads to the gradual acceptance of the generated data sets as "big remote sensing data" [1], not merely due to their high volume but also due to the inherent complexity of the data, which calls for advanced processing techniques that often use external sources of information (e.g., social media data) for adequate interpretation [2].

For instance, remotely sensed hyperspectral images [3] record information using hundreds of spectral bands collected at nearly contiguous wavelengths in the electromagnetic spectrum. This significantly increases their volume with regards to other kinds of image data used in remote sensing applications, such as MSIs (tens of bands), radar, or microwave data sets, creating important requirements in terms of storage and processing. In fact, there has been an exponential growth in these requirements due to recent technological advances in both the quality and number of available imaging instruments. This results from the ever-increasing number of EO missions that are now generating a nearly continuous flow of remotely sensed data, which fostered the creation of large remote sensing data repositories that can only be exploited using adequate parallel and distributed processing techniques [4]. Only in the hyperspectral domain, the AVIRIS [5] and its new generation (AVIRIS-NG)—operated by NASA's Jet Propulsion Laboratory—acquires almost 9 GB of data per hour. Similarly, the EO-1 Hyperion sensor acquires about 71.9 GBs of data per hour, which means over 1.6 TB of data per day. Most of the EO missions that will be operational soon, e.g., the German EnMAP,[1] exhibit equal or even higher data acquisition rates. This confirms that remote sensing data have entered the "big data era" [6]. To be more specific, we summarize, in the following, the properties that qualify remote sensing data as a kind of big data [7].

1) *Data Volume:* As mentioned before, remote sensing data in the optical, radar, and microwave domains are now characterized by their huge dimensionality, which results in the acquisition of several TBs of data per day. The total amount of data archived by the EOSDIS[2] now exceeds 30 PB of data. The amount of archived data at the NSMC[3] exceeds 5 PBs, and the CCRSDA[4] has more than 20 PBs of remote sensing data in its archive.
2) *Data Variety:* According to the state of the satellite industry report,[5] there are more than 300 EO satellites currently in orbit. All of them carry at least one EO instrument, and they are able to collect and transmit data continuously to Earth stations. This means that hundreds of different kinds of remotely sensed data are transmitted (in parallel) to the respective ground receiving stations every day. In addition, since Landsat-1 was first operational in 1972, there have been more than 500 EO satellites launched into space, collecting and archiving more than 1000 different types of remote sensing data.
3) *Data Velocity:* With the development of satellite constellations, the satellite revisit times have gradually transitioned from months to days, hours, or even minutes. As a result, the multitemporal resolution of remote sensing data has increased exponentially, allowing for advanced environmental monitoring and climate change applications. As mentioned before, data centers now receive an almost continuous flow of remote sensing data at ever-increasing speeds, which creates important requirements in terms of storage and analysis (particularly in the context of real-time applications).

In addition to the three aforementioned characteristics, commonly known as the "three V's" of remote sensing big data [6], there are also other important aspects inherent to the analysis and interpretation of such data. One of the most important ones is data *heterogeneity*. Specifically, due to the existence of various satellite orbits and specifications for different sensors (with different storage formats, data projections, spatial resolutions, and revisit times), there are vast differences in the formats of the archived data, and these differences create difficulties when developing general data interpretation techniques. Currently, big remote sensing data analysis is attracting significant attention from governmental and commercial partners, as well as from academic institutions. This is because the high-level products that can be obtained from the data are useful in many relevant applications, including agriculture, disaster prevention and reduction, environmental monitoring, public safety, and urban planning, among many others [8].

One of the most important remote sensing big data projects has been the EOSDIS [9], which provides end-to-end capabilities for managing NASA's Earth science data from various sources. In Europe, the ESA has been organizing the "Big Data from Space" conference, with the ultimate goal of stimulating interactions and bringing together partners and service providers willing to exploit and interpret remotely sensed big data collected from space. The GEO [10], a large-scale cooperation organization, also promotes the development of big data in remote sensing applications. As for commercial applications, Google Earth[6] is a successful case of bringing remotely sensed big data to a large number of users around the world. Many remote sensing applications, such as target detection, land-cover classification, spectral unmixing, and pansharpening, can now be developed easily by resorting to Google Earth and

[1] http://www.enmap.org/
[2] https://earthdata.nasa.gov/eosdis
[3] https://www.nsmc.org.cn/en/
[4] https://data.globalchange.gov/organization/china-center-resources-satellite-data-application
[5] https://sia.org/news-resources/state-of-the-satellite-industry-report/
[6] https://www.google.com/intl/es/earth/

advanced processing algorithms. At the academic level, we have also seen important efforts in top journals launching multiple special issues devoted to the processing and analysis of remotely sensed big data [1], [6]. Despite these significant advances, the processing of remotely sensed big data still faces significant challenges that we summarize in the following.

1) *Data Integration Challenges:* A unified data standard is needed for heterogeneous remote sensing data integration. This includes uniform data standards, metadata standards, and image standards. However, due to the massive, multimodal, and heterogeneous nature of big remote sensing data, this is a challenging and yet unaccomplished goal.
2) *Data Processing Challenges:* How to design computationally efficient and application-specific data processing and storage techniques (while providing unified interfaces to simplify the access to distributed collections of big remote sensing data) is also a pressing challenge. In a computing system, data transmission is generally the bottleneck due to the limited network bandwidth [8]. Also, the dependence between tasks may introduce ordering constraints, and the optimized scheduling of these tasks may be critical to achieve satisfactory processing performance.

B. Cloud Computing in Remote Sensing

Currently, cloud computing[7] platforms are increasingly being used to process and store remotely sensed big data in distributed architectures [7]. The explosive growth of remote sensing big data has revolutionized the way these data are managed and processed. Still, important challenges remain due to the complex management of multimodal, multispectral, multiresolution, and multitemporal remote sensing data sets, which appears in various formats and/or distributed across data centers. In this regard, cloud computing (based on virtualization technology) offers the potential to integrate computing, storage, network, and other physical resources to build a virtual resource pool where advanced techniques for remote sensing data processing can be developed and deployed. In other words, the cloud provides users with services to integrate data, processing, production, computing platforms, storage, and integrated spatial analysis, so as to provide solutions in different application domains, such as environmental problems, land-use/land-cover, and urban planning. Cloud computing technology also offers advanced capabilities for service-oriented and HPC. The use of cloud computing for the analysis of large repositories of remote sensing images is now considered a natural solution, resulting from the evolution of techniques previously developed for other types of computing platforms, such as commodity clusters or grid environments [12].

Quite surprisingly, there are not too many works yet describing the use of cloud computing infrastructures for the implementation of remote sensing data processing techniques. This is partially due to the lack of open repositories containing labeled remote sensing images for public use, a situation that is now changing due to the initiatives, such as BigEarthNet.[8] Also, NASA and ESA are now providing large distributed repositories of remote sensing data for open use by the scientific community (e.g., the Sentinel program).[9] Due to the availability of such repositories, the development of techniques based on cloud computing for distributed processing of remote sensing images has become a very timely research line.

A relevant question at this point is whether cloud computing can become a *de facto* architecture for remotely sensed data interpretation in future years. Our belief is that, by virtue of its elasticity and high transparency levels, cloud computing offers a truly unique paradigm for big remote sensing data processing, in which computational resources can be accommodated in the form of ubiquitous services on-demand, on a pay-per-use basis. In addition, cloud-enabled remote sensing data processing infrastructures and services can now be delivered for large-scale remote sensing data across geographically distributed data centers, which was impossible even with the most powerful compute clusters. As a result, the incorporation of the cloud to big remote sensing data processing initiatives reveals its capacity to deal with the increased computational and storage challenges introduced by modern remote sensing applications, especially when coupled with the powerful new DL algorithms [13] that have been shown to provide an excellent tool for information extraction from scientific data, in general, and from remotely sensed data sets, in particular [14].

C. Article Organization and Contributions

In this article, we provide a review of the most important initiatives that have been developed so far in the use of cloud computing architectures (compared with other HPC solutions, such as commodity clusters or grid computing platforms) for remote sensing data interpretation, with particular focus on DL techniques and their implementations in the cloud. We believe that this review is quite unique in the sense that it provides a completely new flavor with regards to other published works that focus on DL in remote sensing [15]–[17], big remote sensing data [1], [6], or HPC in remote sensing [4], [18], [19]. None of these review papers considered cloud computing as a *de facto* architecture for big remote sensing data processing, which is now a widespread implementation option (in particular, when computationally demanding DL algorithms are involved in the information extraction process). As a result, we believe that this review is necessary given the recent advances in processing strategies, which inevitably

[7]A formal widely accepted definition of cloud computing can be found in [11].

[8]http://bigearth.net/

[9]https://sentinel.esa.int/

led to using DL algorithms in the cloud for the successful processing and interpretation of big remote sensing data sets. The remainder of this article is structured as follows.

1) Section II provides a general overview of techniques for DL in remote sensing data processing and taxonomy of DL architectures that have been widely used in this context.
2) Section III provides a comprehensive review of available approaches for the efficient implementation of remote sensing data processing techniques based on DL algorithms in HPC architectures, including clusters, grids, and cloud computing systems. This section also includes a discussion and some critical observations resulting from the in-depth analysis of the works published so far in these areas.
3) Section IV focuses on cloud computing as the current *de facto* architecture for big remote sensing data processing using DL algorithms. This section first provides some basic concepts about cloud computing. Then, it details some of the most popular frameworks and programming models that have been used for DL-based processing of remotely sensed data and other scientific applications in the cloud.
4) Section V describes the most popular ML and DL libraries and frameworks in cloud computing environments, with a particular emphasis on those that have been already used in remote sensing applications.
5) Section VI provides a case study with a processing example that illustrates a representative technique for DL-based distributed processing of remotely sensed hyperspectral data in the cloud, providing also some suggestions for practical use and exploitation.
6) Finally, Section VII concludes this article with some remarks and hints at plausible future research avenues.

II. DEEP LEARNING IN REMOTE SENSING

A. Remote Sensing Data Processing

Remote sensing technology now provides high-quality data from the surface of the Earth (in terms of detailed resolution, good signal-to-noise ratio, robustness to perturbations, and accurate error corrections). Advanced analysis and interpretation methods are required to extract the most useful information contained in the data. As a result, the remote sensing discipline involves many Earth science disciplines, such as meteorology, geology, or ecology, as well as a variety of engineering skills to properly interpret the huge amount of remotely sensed data provided by a constellation of air/space EO instruments.

Furthermore, a wide variety of remotely sensed data can be obtained from these instruments, where optical imaging systems capturing reflected sunlight are pretty popular due to the rich information that they contain, with different formats and resolutions (spatial, spectral, and temporal), enabling a very detailed and comprehensive assessment of surface properties [20], [21]. For illustrative purposes, Table 1 lists some popular remote sensing instruments that have been or are currently operational.

Table 1 List of Popular Remote Sensing Instruments. GSD: Ground Sample Distance, PAN: Panchromatic, MSI: Multispectral, and HSI: Hyperspectral

Name	Type	Spatial resolution	Spectral bands	Spectral Range
RADARSAT-2 [22]	SAR	1-100 m	1 (C-band)	5.405 GHz
ALOS PALSAR [23]	SAR	6.25-12.5 m	1 (L-band)	1.27 GHz
AirMOSS [24]	SAR	100 m	1 (P-band)	0.43 GHz
TerraSAR-X [25]	SAR	0.24-40 m	1 (X-band)	19.65 GHz
Sentinel-1 [26]	SAR	5-20 m	1 (C-band)	5.404 GHz
LVIS [27]	Laser	20 m	1	1064 nm
EROS-B [28]	PAN	0.7m	1	500-900nm
GAOFEN-1 [29]	MSI	2.0-8 m	5: PAN Blue Green Red NIR	450-900 nm 450-520 nm 520-590 nm 630-690 nm 770-890 nm
GAOFEN-2 [29]	MSI	0.81-3.24 m	5: PAN Blue Green Red NIR	450-900 nm 450-520 nm 520-590 nm 630-690 nm 770-890 nm
IKONOS [30]	MSI	0.82-3.2 m	5: PAN Blue Green Red NIR	526-929 nm 445-516 nm 506-595 nm 632-698 nm 757-853 nm
WorldView-4[31]	MSI	0.3-1.2 m	5: PAN Blue Green Red NIR	450-800 nm 450-510 nm 510-580 nm 655-690 nm 780-920 nm
Sentinel-2 [32]	MSI	10-60 m	13	443-2190 nm
Sentinel-3 OLCI [33]	MSI	300-1200 m	21	400-1200 nm
Landsat-8 [34]	MSI	15-30 m	11	300-12510 nm
MODIS [35]	MSI	250 - 1000 m	36	405-14385 nm
CHRIS [36]	HSI	18-36 m	62	415-1050 nm
AVIRIS [5]	HSI	20 m	224	360-2450 nm
AVIRIS-NG [37]	HSI	0.3-4.0 m	600	380-2510 nm
ROSIS [38]	HSI	1.3 m	115	430-860 nm
CASI [39]	HSI	2.5 m	114	360-1050 nm
HYDICE [40]	HSI	1-7 m	210	400-2500 nm
HYMAP [41]	HSI	5 m	126	450-2500 nm
PRISM [42]	HSI	2.5 m	248	350-1050 nm
EnMAP [43]	HSI	30 m	228	420-2400 nm
HISUI [44]	HSI	30 m	185	400-2500 nm
DESIS [45]	HSI	30 m	180	400-1000 nm
HYPERION [46]	HSI	30 m	220	400-2500 nm
PRISMA [47]	HSI	30 m	237	400-2500 nm
SHALOM [48]	HSI	10 m	241	400-2500 nm

In fact, optical remote sensing data play an important role in many different activities [49]. On the one hand, PAN, standard RGB, and multispectral products often exhibit impressive spatial resolution, which strongly facilitates the detection of contours, textures, and structures within the scene. On the other hand, although the spatial resolution is partially sacrificed, hyperspectral instruments collect hundreds of narrow and near-continuous bands ranging from the VNIR to the SWIR wavelength regions, providing very detailed spectral signatures of the materials covered by each pixel, thus allowing more accurate and detailed analysis of the spectral features available in the data [50].

The advanced products resulting from the analysis and interpretation of remotely sensed data sets allow for the implementation of long-term development strategies in several fields, such as precision agriculture [51], urban planning [52], or desertification monitoring [53], among others.

The rich and detailed information contained in remote sensing data needs to be adequately extracted and processed to be consequently exploited at the user level. In this regard, there is a strong demand for accurate and

Fig. 1. ***Remote sensing classification can be tackled at three different levels: pixel-level (left), object-level (center), and scene-level (right).***

computationally efficient analysis techniques, which can be categorized considering different criteria. In particular, according to their purpose [54], we can classify available techniques into the following groups.

1) *Restoration and denoising methods* manage data corruption and anomalies introduced during the acquisition process that may significantly degrade the quality of the collected data [55], involving radiometric and geometric corrections, or diffuse solar radiation, and management of atmospheric effects.
2) *Enhancement methods* transform the captured data to increase the quality of certain features, making them more suitable to human vision skills to conduct visual analysis. This involves contrast enhancement, super-resolution, and pan-sharpening, for instance.
3) *Transformation methods* modify the scene content in either the spectral or spatial domain for feature extraction, image compression, or filtering purposes, such as the PCA, TCT, vegetation indices, or the WT.
4) *Classification methods* interpret the content of remotely sensed scenes. Three classification levels can be distinguished (see Fig. 1) [56], where pixel-level classification labels each pixel in a scene with a semantic class [57]; object-level classification seeks to recognize the elements present in the scene (by usually combining spectral–spatial features) [58]; and scene-level classification provides a global meaning (i.e., a semantic class) to the entire scene by understanding and interpreting its features [59]. Moreover, subpixel analyses can also be conducted by analyzing the spectral mixtures at each pixel (termed subpixel classification or spectral unmixing) [18], [60], [61].

Focusing on data transformation and classification approaches, ML and DL methods have provided a wide range of processing algorithms for both regression and classification of complex nonlinear systems [14], [16], [62], [63], implementing promising learning paradigms to derive information from the data. These methods range from purely unsupervised strategies to supervised ones, with a vast collection of semisupervised and hybrid-based methods in between [64]–[67].

For instance, k-means clustering is a popular unsupervised method that groups similar samples together by exploring similarity measures and is able to discover underlying patterns [68]. On the contrary, the kNNs usually explores the similarity between samples in a supervised way [69]. Also, DTs [70] and RFs [71] are supervised methods, where RFs develop multiple trees from the randomly sampled subspace of the input sample and then combine the output through a voting/maximum rule. In addition, the MLR [72], GMMs [73], and naive Bayes-based (NB) approaches [74] are probabilistic models that analyze the data distribution to conduct their assumptions. HMMs [75] and SVMs [76] are accurate statistical classifiers. Particularly, the SVM is considered an efficient and stable algorithm for high-dimensional data classification. This method learns the decision hyperplane that can best separate training samples in a kernel-included high-dimensional feature space. Finally, ANNs [77] are versatile empirical-modeling algorithms composed of hierarchical layers of neurons that process input stimuli using synaptic weights and transmit their responses through activation functions. As a result, each layer refines the neural responses to the input data, obtaining increasingly abstract representations by adjusting the model weights, which are automatically learned from the data through the forward–backward propagation mechanism to extract the most relevant information. Moreover, ANNs offer a very flexible architecture in which both the number of layers and neurons (and even the shape and direction of the connections) can be established by the programmer. In this sense, the great flexibility and automatic adjustment of neural models (without any prior knowledge about the statistical features of the data) are major advantages that have positioned ANNs as a very attractive approach, creating a sharp contrast to traditional ML techniques, which usually requires careful engineering to extract complex handcrafted features, requiring specific knowledge to recognize the specific regularities present in the data.

Indeed, the study and implementation of ANNs are so extensive that, within ML, the subfield of DL has emerged as a hot topic for signal data processing [13]. Particularly, DL algorithms have gained significant popularity in remote sensing data analysis over the past few years. For illustrative purposes, Fig. 2 shows the total number of papers per year published on this topic and the number of citations received, revealing an exponential increase in recent years. The figure was generated using the Web of Knowledge engine,[10] and the exact search string used (and the date of the query) is specified in the figure caption—where "AB" indicates that the search was conducted in the abstracts of the papers—for reproducibility purposes.

[10]http://www.webofknowledge.com/

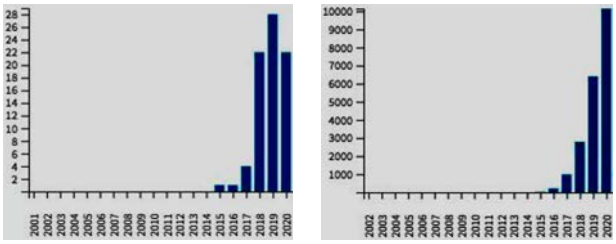


Fig. 2. *Total number of (a) papers per year and (b) citations received by papers in the area "DL in remote sensing." Source: Web of Knowledge. Search string: AB = ("deep learning" AND "remote sensing"). Total number of results: 1621. Date of the query: February 20, 2021.*

A detailed review of available approaches in this area was given in [15], which discusses how DL has been applied for remote sensing data analysis tasks, such as image fusion, registration, scene classification, object detection, land-use and land-cover classification, and segmentation. Different application fields are also covered, including open challenges and directions for future research. Also, the paper [17] focuses on remotely sensed hyperspectral data, providing a comprehensive review of methods for DL-based classification in this field and discussing the strengths and weaknesses of these methods. Accordingly, Section II-B includes a brief taxonomy of the main DL models developed for remote sensing data analysis.

B. Taxonomy of DL Architectures

There is a great variety of deep models due to the great flexibility of existing architectures in terms of topology, data-path connections, and types of layers. In general terms, the scientific community recognizes five models, i.e., SAEs, DBNs, RNNs, and CNNs, as the main architectures, from which a great variety of modified networks have been implemented [17], such as generative adversarial networks (GANs), which have gradually become a mainstream architecture in the field of remote sensing. In the following, we review these DL models.

1) SAEs: The autoencoder (AE) implements an encoder–decoder structure to learn a code representation from the input data in an unsupervised way. It defines an optimization problem that attempts to minimize the reconstruction error $|| \left(\mathbf{W}_d \sigma \left(\mathbf{W}_e \mathbf{X}^T\right)\right)^T - \mathbf{X}||_2^2$ by learning the matrices of bases $\mathbf{W}_d$ and $\mathbf{W}_e$, where $\mathbf{X} \in \mathbb{R}^{N \times B}$ defines the remote sensing data as an input matrix of N samples with B channels (feature space), $\mathbf{W}_e \in \mathbb{R}^{B' \times B}$ comprises the recognition weights of the encoder, which maps the input data to a code/latent representation with $B' \neq B$ features (code/latent space), and $\mathbf{W}_d \in \mathbb{R}^{B \times B'}$ comprises the generative weights of the decoder, which recovers the original feature space by reconstructing the input data. σ acts as an activation function. The SAE deepens the model by stacking several AEs [see Fig. 3(a)], where the AEs of the stacked-encoder (bottom-half) find a series of lower dimensional features, and the stacked-decoder (top-half) performs the opposite function [78].

2) DBNs: The DBN is a multilayer generative model inspired by RBMs. An RBM is a two-layer stochastic network trained to minimize the input reconstruction error in a similar way as AEs (*Gibbs sampling*), where the visible layer deals with the input data and the hidden layer conducts feature extraction, capturing higher order data correlations observed in the visible units while learning a probability distribution over the input data [see Fig. 3(d)]. DBNs take advantage of RBMs, concatenating several pretrained RBMs and refining the full-model parameters through labeled data.

3) RNNs: The RNN [see Fig. 3(b)] retains memory and learns data sequences by introducing loops in its connections. As a result, the neural responses in each step depend on those of the previous step by means of an internal state, which creates a sequential dependence that provides an association between the current and the previous data sample. According to which hidden states are created and how they are managed, three types of RNNs can be distinguished [79]: the simple vanilla RNN, the gate-based LSTM, and the simplified gated recurrent unit (GRU).

4) CNNs: In contrast to other deep models (which were originally implemented with fully connected layers, e.g., the AE), the CNN introduces the convolution layer as a set of locally connected weights that are rearranged in an n-dimensional grid. As a result, the convolution kernels act as feature detection-extraction filters, where neuronal responses are arranged in a feature map that not only indicates the presence of a particular stimulus detected by the kernel (edges, borders, and shapes) but also the location of these stimuli in the spatial domain. This enables abstract and refined spatial relationships to be maintained and extracted within the data through a hierarchical stack of convolution layers [see Fig. 3(c)], which are combined with other layers (activation, normalization, and pooling functions, for instance) to extract elaborate patterns from the raw inputs.

The flexibility of convolution kernels has demonstrated a great potential to extract any kind of feature from the raw data, without applying complex preprocessing mechanisms. Furthermore, the great architectural plasticity in terms of kernel size and grid organization (which produces 1-D, 2-D, and 3-D models), layer connections (direct, residuals, skip, and short-connections), data paths settings, and wide/depth configuration, along with the impressive generalization power and the ability to make strong assumptions about the input data, have established convolution-based models as the most successful and popular deep networks. In fact, these networks represent the state of the art in image processing through

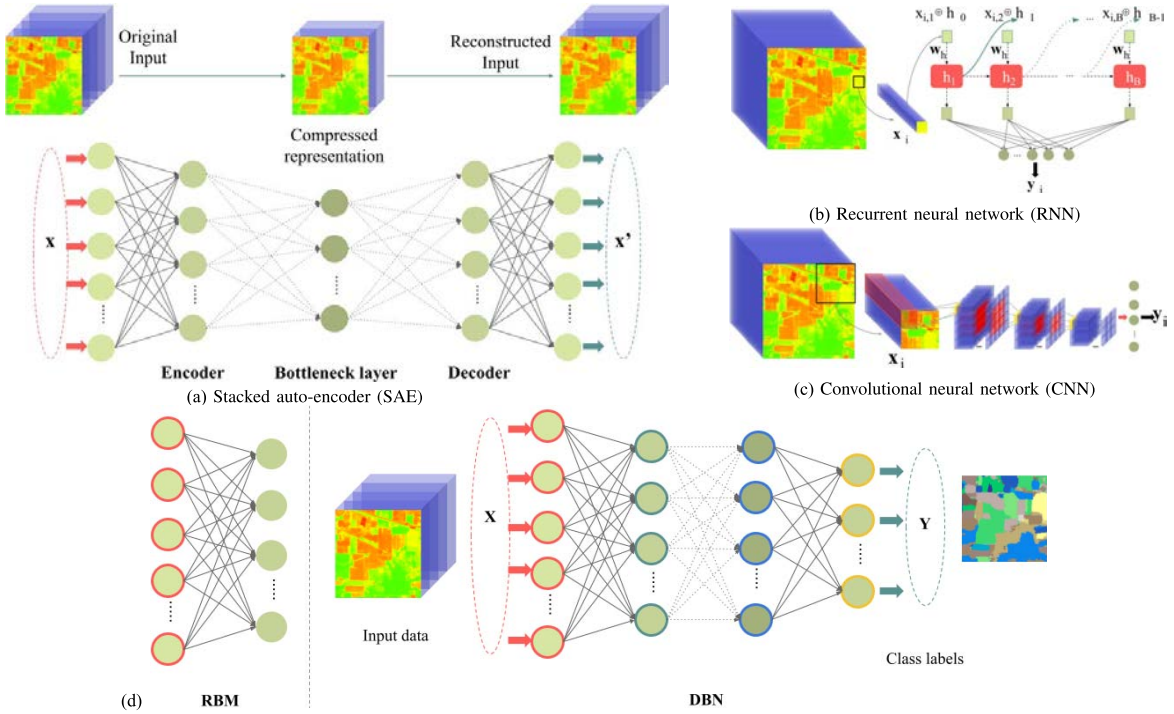


Fig. 3. ***Graphical illustration of traditional deep network architectures applied for processing remote sensing data cubes. (a) SAE optimizes the reconstruction error between its vector input X and its output X′, where the bottleneck layer contains the latent data representation. (b) In the traditional RNN model, the data cube is processed in band-by-band fashion, where the spectral signature of each pixel x_i is processed as a temporal sequence, obtaining, as a result, a hidden state h that works as the model memory. (c) CNN processes 3-D inputs (i.e., the pixel x_i and its surrounding area) by applying multidimensional kernels that act as filters for particular features (borders, shapes, and so on), obtaining a set of feature maps with the neuronal responses to that stimuli. (d) Finally, the DBN is composed of several RBMs that process and reconstruct the input data, mimicking the SAE behavior to learn the probability distribution of the input in an unsupervised fashion (DBN based on RBM, where the visible layer is highlighted with a colored border).***

derived network models, such as residual (ResNets), dense (DenseNets), and capsule-based (CapsNets), among many others [80], [81].

5) GANs: The aforementioned networks work as discriminative models, which maps original inputs to some desired outputs (by learning conditional distributions between them) to minimize a loss function. In contrast, generative approaches (such as GANs) learn the joint probability between inputs and outputs, modeling the data distribution to generate new samples rather than just evaluating the available ones. Thus, GANs (see Fig. 4) model a data distribution from a random noise vector through an adversarial process, where two neural models, i.e., *generative* and *discriminative* networks, are simultaneously trained in the competition (the former to deceive the latter, and the latter to avoid being deceived by the samples generated by the former).

III. IMPLEMENTATIONS

In typical DL models, such as those illustrated in Fig. 3, there are millions of parameters (which defines the model), and large amounts of data are required to learn these parameters. This leads to a computationally intensive process in which the learning step consumes a lot of time. Therefore, it is important to come up with parallel and distributed algorithms that can run much faster and drastically reduce the training times in the context of remote sensing applications. In the following, we provide a description of different parallelization strategies for accelerating DL algorithms in remote sensing applications by resorting to three types of HPC architectures: clusters, grids, and clouds. A description of the main challenges faced by these different architectures, along with a brief comparison among them, is then presented. The section

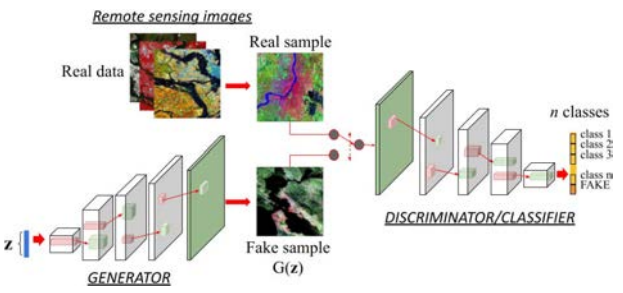
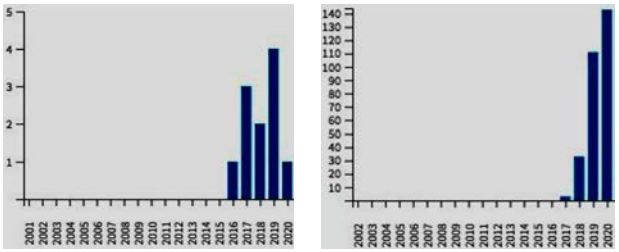


Fig. 4. ***GAN for remote sensing data processing.***



(a) (b)

Fig. 5. *Total number of (a) papers per year and (b) citations received by papers in the area "DL in remote sensing using cluster computing." Source: Web of Knowledge. Search string: AB = ("deep learning" AND "remote sensing" AND "cluster" AND "comput*"). Total number of results: 11. Date of the query: February 20, 2021.*

concludes with a discussion on their potential role in solving remote sensing problems via DL algorithms.

A. Cluster Computing

In this section, we describe some of the most relevant approaches in the recent literature to exploit cluster computer architectures (including GPU clusters) for the efficient interpretation of remote sensing data. Fig. 5 shows the total number of papers published in this area, along with the number of citations received (according to Web of Knowledge). In the following, we discuss some of the most relevant contributions in this field.

As one of the most notable recent developments, Lunga *et al.* [82] implement a RESFlow for improving DL algorithms and allowing them to perform computations on large-scale remotely sensed images. The RESFlow works by dividing the data into homogeneous partitions that can fit simple models in homogeneous (i.e., commodity cluster-based) machines. Despite its cluster-oriented nature, RESFlow uses Apache Spark (a tool that has been widely used in cloud computing, as described in Section IV) to accelerate DL inference. The RESFlow incorporates a strategy to optimize resource utilization across multiple executors assigned to a single worker. The framework invokes DL inference at three stages: during deep feature extraction, deep metric mapping, and deep semantic segmentation. Resource sharing in GPUs is adopted to achieve a fully parallelized pipeline for all execution steps.

RESFlow uses Apache Spark, but there are multiple options to distribute training over cluster computing implementations. Other options include TF,[11] PyTorch,[12] or Horovod.[13] These frameworks provide multiple benefits. For example, Pytorch includes multiple extensions (as NVIDIA Apex[14]) to enable streamline mixed precision with distributed training, and Horovod employs efficient inter-GPU communications. Similar to Apache Spark, some cluster computing approaches take advantage of Kubernetes[15] (k8s) architecture workflow, which allows for the deployment automation, scaling, and management of ML/DL applications, as described in [83].

The work in [84] exemplifies the unique advantages provided by parallel computing environments and programming techniques to solve large-scale problems, such as the training of classification algorithms for remote sensing data. Specifically, the authors demonstrate that the training of deep CNNs can be efficiently implemented using cluster computers containing a large number of GPUs. The obtained results confirm that parallel training can dramatically reduce the amount of time needed to perform the full training process, obtaining almost linear scalability in a cluster of GPUs without losing any test accuracy.

At this point, we emphasize that some works do not need to exploit clusters of GPUs to conduct the desired calculations. For instance, in the work [85], the authors resort to a shared memory system with only four GPUs to develop an MTFC of a CNN for remotely sensed hyperspectral image classification. The authors first develop a PNPE that generates 3-D cube samples from the input data automatically. Then, they perform a series of optimizations in the MTFC, such as task division, the fine-grained mapping between tasks and GPU thread blocks, and shared memory usage reduction. To further improve the training speed, the authors exploit CUDA streams and multiple GPUs to train minibatches of data samples simultaneously. The MTFC is shown to outperform popular ML frameworks, such as Caffe[16] and Theano,[17] while offering the same level of classification accuracy.

The paper [86] provides several parallelization approaches for DNNs, taking into account network overheads and optimal resource allocation, since network communication is often slower than inter-machine communication (while some layers are more computationally expensive than others). Specifically, the authors consider a multimodal DNN architecture and identify several strategies to optimize performance when training is accomplished on Apache Spark (this framework will be described in detail in Section IV). The authors compare their newly developed architecture with an equivalent DNN architecture modeled after a data parallelization approach. The experiments in the paper reveal that the way in which the model is parallelized has a very significant impact on resource allocation and that hyperparameter tuning can significantly reduce network overheads.

Other relevant developments in this area include the paper [87], which presents parallel versions of DL techniques for dimensionality reduction of remotely sensed images, also implemented in an Apache Spark cluster. The

[11] https://www.tensorflow.org/
[12] https://pytorch.org/
[13] https://horovod.ai/
[14] https://nvidia.github.io/apex/
[15] https://kubernetes.io/es/
[16] https://caffe.berkeleyvision.org/
[17] https://github.com/Theano/

paper [88] presents an improved version of the aforementioned development that scales even better in clusters of GPUs. The paper [89] presents a parallel and distributed DL-based spectral unmixing algorithm for remotely sensed hyperspectral data, again using Apache Spark for the implementation on a cluster computer.

B. Grid Computing

Although many parallel systems are inherently homogeneous, a most recent trend in HPC systems is to use highly heterogeneous computing resources, where the heterogeneity is generally the result of technological advancement in the progress of time. With increasing heterogeneity, grid computing emerged as a premier technology that could facilitate the processing of remote sensing data in heterogeneous and distributed computing platforms.

Although the grid has recently evolved into architectures with more quality of service, such as the cloud, there were several reasons for using grid computing for remote sensing image processing when the first grid-oriented architectures appeared. First and foremost, the required computing performance may not be available locally. Also, the required performance may not be available in just one location, with a possible solution being cooperative computing. Last but not least, the required computing services may be only available in specialized centers, and in this case, the solution is application-specific computing. This led to the development of some grid-based approaches that are now mostly transitioning into cloud computing implementations, as will be described in Section III-C.

For illustrative purposes, Fig. 6 shows the total number of papers published in this area, along with the number of citations received. In the following, we discuss some of the most relevant contributions focused on using grid computing platforms for solving remote sensing problems via DL algorithms.

The GEOGrid project was one of the first DL-oriented initiatives aimed at providing an e-Science infrastructure to the remote sensing community. It is specifically developed to integrate a wide variety of remote sensing data sets and is accessible online as a set of services.[18]

The platform called G-POD[19] was the first one to provide a grid-based environment for processing satellite images provided by ESA, offering several image processing services and DL algorithms mainly intended for environmental studies. G-POD has been successfully applied in real applications, such as flood area detection.

As an add-on to this tool, the platform for satellite imagery search and retrieval, called GENESI-DR [90], offers an advanced interface for digital data discovery and retrieval, where the original images are processed using G-POD facilities (comprising some DL algorithms). The ultimate goal of GENESI-DR was to build an open-access

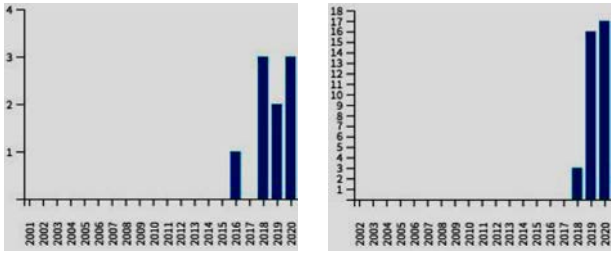


Fig. 6. *Total number of (a) papers per year and (b) citations received by papers in the area "DL in remote sensing using grid computing." Source: Web of Knowledge. Search string: AB = ("deep learning" AND "remote sensing" AND "comput*" AND "grid"). Total number of results: 9. Date of the query: February 20, 2021.*

service to digital repositories focusing on fast search, discovery, and access to remotely sensed imagery in the context of postdisaster damage assessment. Once a disaster alert has been issued, response time is critical to providing relevant damage information to analysts and/or stakeholders. In this regard, GENESI-DR provides rapid area mapping and near real-time orthorectification web processing services to support postdisaster damage needs.

Also, the GiSHEO[20] platform (on-demand Grid services for training and high education in EO) addresses an important need for specialized training services in EO. Solutions were developed for data management, image processing service deployment, workflow-based service composition, and user interaction, with particular attention to services for image processing (able to exploit free image processing tools, along with some DL techniques). A special feature of the platform is the connection with the GENESI-DR catalog, which provides plenty of remote sensing data sets for free.

To conclude this section, it is important to emphasize that the CEOS,[21] an international coordinating body for spaceborne missions focused on the study of the Earth, maintains a working group on information systems and services, with the ultimate goal of promoting the development of interoperable systems for managing EO data internationally. In this regard, several grid platforms have greatly benefited from CEOS standards when developing grid-based tools for accurate interpretation of remotely sensed data [7].

C. Cloud Computing

Cloud computing solutions represent an evolution of grid-based approaches and exhibit the potential to manage and process vast amounts of remotely sensed data in fault-tolerant environments by interconnecting distributed and specialized nodes. This strategy can significantly reduce the processing costs often involved in grid computing,

[18] https://www.geogrid.com/
[19] https://gpod.eo.esa.int/
[20] http://cgis.utcluj.ro/projects/gisheo
[21] https://ceos.org/

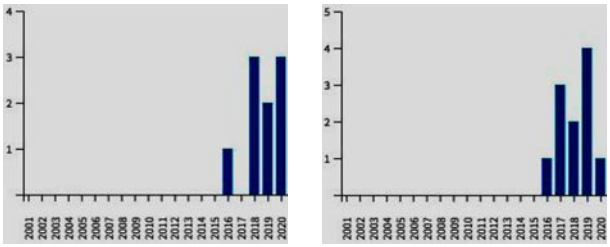


Fig. 7. *Total number of (a) papers per year and (b) citations received by papers in the area "DL in remote sensing using cloud computing." Source: Web of Knowledge. Search string: AB = ("deep learning" AND "remote sensing" AND "comput*" AND "cloud"). Total number of results: 78. Date of the query: February 20, 2021.*

leading to natural and cheap solutions for remotely sensed data processing. For illustrative purposes, Fig. 7 shows the total number of papers published in this area, along with the number of citations. In the following, we review some of the most significant contributions based on cloud computing for solving remote sensing problems via DL techniques.

One of the most relevant works addressing the implementation of DL algorithms for remote sensing data analysis in the cloud was presented in [91], in which the authors introduced a new cloud-based technique for spectral analysis and compression of hyperspectral images. Specifically, the authors provide a cloud implementation of the AE, a popular deep network for non-linear data compression. Apache Spark (described in detail in section IV) serves as the backbone of the cloud computing environment by connecting the available processing nodes using a master–slave architecture. The obtained results indicate that cloud computing architectures offer an adequate solution for compressing and interpreting big remotely sensed data sets.

The paper [92] proposes an acceleration method for hyperspectral image classification that exploits scheduling metaheuristics to automatically and optimally distribute the workload across multiple computing resources on a cloud platform. A representative DL-based classification processing chain is first distributed and implemented in parallel based on the MapReduce mechanism (described in detail in Section IV) on Apache Spark. The optimal execution on Spark is further formulated as a divisible scheduling framework that takes into account both task execution precedences and task divisibility when allocating the divisible and indivisible subtasks onto computing nodes. The scheduling results provide an optimized solution to the automatic processing of big hyperspectral data on cloud environments. The experimental results demonstrate that this approach can achieve significant speedups in the classification of hyperspectral imagery on Spark, obtaining also significant scalability with regards to increasing data volumes.

The paper [93] exploited the idea that state-of-the-art DL-based algorithms and cloud computing infrastructure have become available with a great potential to revolutionize the processing of remotely sensed images. Specifically, their study evaluated, using thousands of images obtained over a 12-month period, the performance of three ML and DL approaches (RFs, LSTMs, and U-Nets). The DL algorithms (LSTMs and U-Nets) were implemented using the TF framework (described in detail in Section IV), while the ML-based RF utilized the Google Earth Engine platform. The study concluded that, although the use of ML/DL algorithms depends highly on the availability of labeled samples and the generalization of these methods still presents some challenges, algorithms based on ANNs can still be used in the cloud to map large geographic regions that consider a wide variety of satellite data formats.

The paper [94] uses cloud computing to make global-oriented spatiotemporal data simulations using the OpenStack management framework (described in detail in Section IV). This is accomplished by resorting to DGGSs, designed to portray real-world phenomena by providing a spatiotemporal unified framework on discrete geospatial data structures, along with a DL-based algorithm to address the challenges resulting from big remote sensing data storage, processing, and analysis.

The paper [95] presents a new parallel CBIR system from remotely sensed hyperspectral image repositories, implemented on a cloud computing platform. The method exploits information from spectral unmixing [96] and DL to accurately retrieve hyperspectral scenes. To this end, the authors implement a distributed and DL-based unmixing method that operates on a cloud computing environment. In addition, they implement a global standard distributed repository of hyperspectral images equipped with a large spectral library in a SaaS mode (this concept will be described in detail in Section IV), providing users with big hyperspectral data storage, management, and retrieval capabilities through a powerful web interface. The parallel unmixing process is then incorporated into the CBIR system to achieve a highly efficient unmixing-based content retrieval system.

Other important contributions in this area include the paper [97], which proposed a model that facilitates the utilization and performance of Apache Spark algorithms in cloud environments. Also, the paper [98] presents the architecture of the ICP data mining package, a distributed tool for the analysis of remotely sensed data. The paper [99] proposes an extension of Apache Hadoop (described in detail in Section IV) that executes operations for processing remotely sensed images (including DL methods) in a highly distributed and efficient way. The paper [100] proposes a highly scalable and efficient segmentation model for remotely sensed images, capable of segmenting very high-resolution imagery with DL algorithms. The paper [101] proposes a method for DL-based cloud computing based on image sampling, which

Table 2 Comparison Between Cluster, Grid, and Cloud Architectures

	Cluster computing	Grid computing	Cloud computing
Scalability	Low	Low	High
Elasticity	No	No	Yes
Heterogeneity	No	Yes	Yes
Compute capability	Cluster dependant	Grid dependant	On-demand
Business services	No	No	Yes
Private service cost	Medium	High	On-demand
Public service available	Yes	Yes	Yes
Resource handling/allocation	Centralized	Distributed	Both
Job queuing	Yes	Yes	No

models the remotely sensed data set to be processed as a streaming service and divides it with a Voronoi diagram. The paper [102] describes a Java software based on the MapReduce model for handling and processing remotely sensed images using DL methods. The paper [103] presents a deep method for storing images using MapReduce. The paper [104] also uses a MapReduce framework for DL-based parallel processing of remotely sensed data through Apache Hadoop. The work [105] describes a set of requirements to achieve a generalized and integrated EO information system and the associated (real-time and off-line) data processing techniques based on DL. The paper [106] presents a new DL-based approach for distributed processing of large-scale satellite images in the cloud. The paper [107] presents a distributed spatiotemporal indexing architecture implemented on the cloud and a distributed DL-based algorithm for improved spatial–temporal queries. The paper [108] discusses the requirements of overlapping data organization and proposes two extensions of the HDFS—described in Section IV—and the MapReduce programming model for dealing with remotely sensed data. The paper [109] describes a DL framework for the efficient analysis of large image volumes, which processes, daily, the data obtained by NASA's EO-1 satellite.

D. Challenges and Comparison

The different distribution perspectives described in the three previous subsections exhibit numerous differences. As a summary, Table 2 provides an overlook of the main similarities and differences between the discussed strategies.

Regarding the cost, there are initiatives that provide free computing platform services for the scientific and research communities. An example is PRACE,[22] aimed at high-impact scientific research and engineering development across different disciplines. Also, there are specific projects for different distributed computing approaches. Condor[23] was created for research and education purposes. EGI-InSPIRE[24] was created by the European Commission for the benefit of the scientific communities within the European Research Area to exploit grid infrastructures. This project is also available for cloud computing.

Usually, cloud computing has been identified as a distributed service, which is not entirely true. Alternatives are UCI[25] or OpenNebula,[26] which brings flexibility, scalability, and simplicity for cloud computing management. In addition to these projects, the XSEDE[27] ecosystem and the EGI[28] (European Grid Infrastructure) provide different cost-free alternatives.

One of the most important features of the cloud (and one of the main reasons for its popularity) is the elasticity and scalability of its architecture. Since cluster/grid architectures are limited to available hardware resources, cloud computing offers the possibility to increase such resources by resorting to the elasticity property, i.e., using resources from different infrastructures. This leads to an increase in the heterogeneity of computing and communication resources, and to the use of both centralized and distributed resource handling and allocation.

Finally, another relevant feature of cloud computing is that the infrastructure does not need to use a job queue for managing executions from different users. This significantly reduces the waiting times for the execution of jobs that are needed in other architectures, such as clusters and grids.

E. Discussion

In this section, we discuss some important aspects identified after the systematic review conducted in the previous subsections, in an attempt to answer relevant questions, such as the role of parallel and distributed computing as an efficient tool for solving remote sensing problems via DL algorithms.

In our systematic review, we have found that distributed computing technologies are highly demanded for DL-based data processing when large volumes of remotely sensed data need to be processed. Commodity cluster computers (possibly including hardware accelerators, such as GPUs) [110], grid environments [111], [112], and cloud computing systems [113] have been the most demanded types of HPC platforms for big remote sensing data processing. Recently, cloud computing has become a standard for distributed processing due to its scalability, low cost, service-oriented, and high-performance properties [7]. Therefore, this technology offers the potential to deal with tasks that must be accomplished over large data repositories. As result, cloud computing can be seen as the most natural solution for the analysis of large volumes of remotely sensed data, as well as an evolution of other HPC techniques (such as cluster and grid computing) that correct their limitations and expands their possibilities.

We have also observed that there are comparatively few efforts in the literature aimed at the exploitation of cluster and grid computing infrastructure for the processing

[22] https://prace-ri.eu/
[23] http://www.cs.wisc.edu/condor/condorg
[24] http://www.egi.eu/projects/egi-inspire/
[25] https://code.google.com/archive/p/unifiedcloud/
[26] https://opennebula.io/
[27] https://www.xsede.org/
[28] https://www.egi.eu/services/cloud-compute/

of remote sensing images compared to cloud computing implementations. In fact, we have noticed that the cloud is now in clear expansion and routinely used to solve remote sensing problems (particularly those involving DL algorithms). This is because of the popularity of the programming languages and frameworks available for implementing DL-based algorithms in the cloud (some of these tools will be described in detail in Section IV).

Another important observation arising from our literature review is that the most widely used tool for remote sensing data processing in cloud computing environments is Apache Hadoop [114], described in mode details in Section IV) [115] although recently Apache Spark [116] has also become a reference tool. The main difference between Spark and Hadoop is the fact that the former distributes the data in RDDs [117] that can be managed more efficiently. As a result, the speedup that Spark can achieve with respect to Hadoop is very high. In addition, Spark provides an ML/DL library called *MLlib* [118]—described in Section IV—that operates in a distributed and parallel manner, so that it provides very good performance with big remote sensing data. Although Hadoop has been widely used in the past, Spark is now a standard due to its speed and better memory usage. Our study also shows that most researchers take advantage of existing cloud computing frameworks when dealing with remote sensing data, rather than developing new ones.

In Section IV, we focus on cloud computing as a *de facto* paradigm for distributed processing of remotely sensed data sets and describe the most widely used frameworks and programming languages that have been adopted in this context (some of them already mentioned in this section), with a particular emphasis on the availability of DL-oriented tools.

IV. FOCUS ON CLOUD COMPUTING

This section first introduces some basic concepts about cloud computing and then glances at the delivery models, available execution frameworks, and programming models supporting this technology (with a particular emphasis on the tools that have been specifically used in remote sensing applications).

A. Cloud Computing Basics

Advances in distributed computing technologies, together with the high amount of data generated and consumed by a growing number of devices (including remote sensing instruments), have leveraged the adoption of emergent cloud computing technology. Nowadays, cloud computing has become a suitable model to cover a wide range of users' needs, including data analytics, data mining, remote sensing, social media, and other computational and data-intensive applications. Cloud computing is a model that provides users with ubiquitous, on-demand access to remote hardware and software resources in the form of services. This model has become a cost-effective solution that usually reduces initial investment, management, and maintenance costs with respect to previous clusters and grid technologies. Cloud computing provides elastic computing, storage, and networking payment services. The term elastic refers to the ability of the model to dynamically adapt to user scalability and variable workload necessities [119], [120].

At the core of cloud computing is the *virtualization technology* [121], [122]. Virtualization abstracts the underlying physical resources, such as computing, storage, and networking, as a set of virtual resources, typically enclosed in the form of isolated instances called *VMs*. Such modular design enables some advantageous features, as replication of data instances, fault tolerance, security (by limiting the interaction between modules), and migration of instances [123].

Cloud computing determines how the virtualized resources are allocated, deployed, and delivered to users. In this sense, delivery models are structured in three basic categories that describe the form in which users access the resources [124]:

1) At the higher level of abstraction is *SaaS*, in which the user accesses to end applications usually developed, managed, and maintained by the cloud provider in its cloud infrastructures. Users access these applications via web interfaces. A common example is a CBIR system for remote sensing data repositories.
2) *PaaS* refers to the provision of development full stacks, including OSs, libraries, management, and monitoring tools. The users accessing those services are usually software developers with limited control over the underlying infrastructure, which is managed by cloud providers.
3) The *IaaS* model lies at the lower level of abstraction and allows users to manage an elastic infrastructure composed of a set of computational, storage, and networking virtual resources. The users develop and deploy their applications on such virtual resources and manage the infrastructure.

Without neglecting its benefits, cloud computing faces an important set of challenges. The most significant is *security*, in the sense of both ensuring the access to the data exclusively by authorized users and systems (privacy), and maintaining the integrity and availability of the data distribution (even geographically) and replication across different locations (e.g., as in different remote sensing data centers). *Performance* is an additional key factor impacting the adoption of cloud computing, especially by scientific applications that usually execute on tightly coupled high-performance clusters.

Clusters and clouds have different design goals and features [125]. While the main goal of clusters is performance (supported by dedicated parallel computing resources connected with minimal latency and stable high bandwidth networks), the goal of clouds is to make available on-demand virtual resources in an elastic platform at a

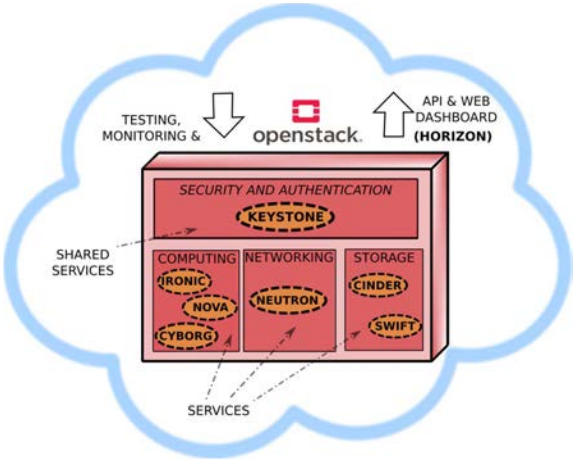


Fig. 8. ***OpenStack main computing, networking, and storage components.***

reasonable cost. Hence, technical management issues—derived from the shared use of physical resources by VMs and multiple users, dynamic on-demand scalability, data movement, virtualization overhead, and workload balance in the presence of heterogeneity—impact the performance of applications deployed on the cloud infrastructure.

B. Deployment Frameworks

Henceforth, we adopt the perspective of users of an IaaS model, allocating a pool of virtual resources connected by networking services to deploy and run scientific applications. As an effective management framework to deploy and run the applications, we highlight OpenStack[29] although there are other open-source frameworks, such as Apache Cloudstack[30] and OpenNebula that offer similar features. In the following, we review the functionality of several of the multiple services available in OpenStack.

Openstack is oriented to manage the complete life cycle of an IaaS cloud system composed of a large number of virtual resources for computing, storage, and networking. The Openstack design architecture is highly modular. Each independent module implements a specific service and exposes a well-defined API, making the system extensible and able to support the integration of third-party services.

Some of the main software components of the OpenStack framework are outlined next and shown in Fig. 8. We structure them into three categories.

1) *Computing service components* for deploying and managing VM and Linux containers. These facilities are supported fundamentally by the *Nova* compute engine. Furthermore, OpenStack offers a general-purpose management framework component for hardware accelerators (such as GPUs) called *Cyborg*.
2) *Networking service components* with support for different network technologies and equipment. *Neutron* is the main component, and it allows managing SDNs and attaching virtual devices to ports on these networks.
3) *Storage service components*, including *Cinder* component for block storage and *Swift* component that delivers services for securely storing unstructured data as a pool of objects and files.

In addition to the aforementioned computing, networking, and storage services, it is worth noting that OpenStack includes multiple software services for monitoring, development, recovery, databases, orchestration of virtualized resources, workload balance, and more. Among them, for instance, *Keystone* provides services for security and authentication of users and applications. Finally, *Horizon* presents a web interface in the form of a dashboard to manage the virtual resources composing the cloud infrastructure.

C. Programming Models

The term, remote sensing big data, was coined to refer to the increasing need of storing, managing, and processing vast amounts of remotely sensed data, produced at a high rate and in a wide range of formats. Such overwhelm flow of data requires flexible parallel platforms with a large computational capacity and specific programming facilities.

In contrast to cluster computer-centric paradigms, as message passing, big data applications require a data-centric approach, in which a computational task should be deployed in a computational resource as close as possible to the data location. As a result, the movement of data through the network can be minimized. The limited capability of the network bandwidth is a factor that, together with the high volume and heterogeneity of remotely sensed data, highly affects the performance. In the following, we review some relevant programming models and application ecosystems that have been widely used in big remote sensing applications.

1) MapReduce: It is a programming model aimed at developing scalable and robust applications working with large data sets [126], [127]. It was originally developed by Google and, together with the distributed GFS [128], has been successfully used in solving numerous big remote sensing data problems (as described in Section III). This model is particularly suitable for data-centric environments, such as big remote sensing data processing on cloud computing platforms [129]. MapReduce provides the developer with a simple interface based on the *map* and *reduce* functions. An application takes as an input a structured set of key-value data. The *map(k1,v1)* function transforms the input to an intermediate set of key-value pairs in the target domain (k2,v2). The MapReduce library combines the intermediate values (v2) on a per-key basis. Finally, the *reduce(k2, list(v2))* function operates on the

[29]https://www.openstack.org
[30]https://cloudstack.apache.org

list of values corresponding to each target key to generate the output. Usually, both *map* and *reduce* are executed by parallel tasks deployed across a set of computational resources. The model interface abstracts the complexities of the execution of the remote sensing application in the specific platform and leaves the underlying details to the implementation.

MapReduce implementations are ultimately based on the master–worker approach, in which a *master* process manages the automatic parallelization and scheduling of the tasks on the computational nodes and coordinates their execution. It partitions the remotely sensed data to be processed by each *map* and *reduce* tasks and schedules those tasks on computational resources as close as possible to the data to be processed. To achieve this, it relies on GFS that provides the locations of the blocks of data to be processed. This design improves data locality and minimizes the data transfers through the network, hence improving the performance.

Besides, MapReduce combines the output intermediate files of the *map* tasks and combines the data according to the output keys. This intermediate stage tackles the complexity of a high amount of communication and coordination tasks, which are hidden to the developer. Finally, the implementation delivers processed data chunks to the *reduce* tasks. Furthermore, MapReduce implementation promotes fault tolerance mechanisms to detect and reexecute tasks when necessary. In this sense, the communication between the deployed tasks is achieved using intermediate files.

2) Hadoop Ecosystem: Hadoop[31] has been widely used to parallelize remote sensing data processing tasks (see Section III). It is a popular open-source and scalable big data software framework based on the MapReduce paradigm, the HDFS, and YARN [115] resource manager.

HDFS basic functionality is similar to that of GFS. Data are split up into blocks of fixed size and replicated across several nodes to ensure fault tolerance and availability. Its implementation follows a master–worker approach. A *Namenode* process manages metadata (such as block mapping information) and delivers that information to the MapReduce library when requested. *DataNode* processes execute in each virtual resource and effectively store data blocks and provide data reading and writing services to applications.

MapReduce functionality is implemented by a *JobTracker* process, which receives job requests and schedules job tasks to different nodes. Each node is controlled by a *TaskTracker* process, which monitors the execution and reports to the *JobTracker* if a problem appears. In such a case, *JobTracker* resubmits the involved tasks to the same or a different *TaskTracker*. In this sense, Hadoop decouples the MapReduce programming model and the associated resource management. YARN delivers the former services. Its internal architecture is based on three main components.

1) The first component is the global per-cluster *ResourceManager* process, which accepts job submissions and allocates resources for the application. It is responsible for scheduling the application in the available resources.
2) The second component is the *NodeManager*, a per-node process. It is responsible for the execution of the tasks assigned to its node.
3) Finally, the *ApplicationMaster* is a per-application process that monitors the application necessities and their status along their lifecycle, negotiating resources with the *ResourceManager*.

The Hadoop framework has been enhanced with multiple tools and services forming the so-called *Hadoop Ecosystem*, which has been exploited in a variety of remote sensing applications (see Section III). It includes relational databases managers (as *Hive*), NoSQL databases (as *HBase*, a column-oriented distributed database running on top of HDFS), distributed ML/DL and linear algebra solvers (as *Mahout*), real-time facilities (*Storm*), efficient alternatives to the MapReduce programming model, and utilities for the orchestration of the services and components (*ZooKeeper*).

3) Apache Spark: Originally developed at UC Berkeley, Apache Spark [116], [130] was designed to gain velocity in the processing of big data. Although the MapReduce model adequately adapts to a large number of applications as highly parallel batch jobs, it incurs significant latency in both interactive applications and in those with an iterative pattern of execution, in which the same data sets need to be continuously reloaded from the file system.

In this respect, one of the most relevant features of Spark is its ability to support persistent data in memory, which greatly benefits performance. This feature is implemented in the run-time system of Spark, known as *Spark Core Engine*. In addition to this module, the Spark ecosystem includes a set of utilities, as *Spark SQL*, which allows managing structured and semistructured data organized in columns, known as DataFrames, the *Spark Streaming* module for performing real-time processing on data streams (ideal for remote sensing applications with real-time constraints, as described in Section III), and the *MLlib* library that includes distributed ML and DL algorithms.

Spark follows a master–worker execution model, with a *driver* node acting as the master and a set of worker nodes. The execution model is shown in Fig. 9. An application submitted to Spark starts its execution in the context of the driver node. The application creates a *SparkContext* object that transforms the sequence of operations described in the main program into an execution plan. The execution plan is represented as a DAG, in which nodes are data elements and edges are operations on such data. SparkContext splits up the DAG in stages to be executed by tasks. Then, SparkContext negotiates the acquisition of resources with

[31] https://hadoop.apache.org

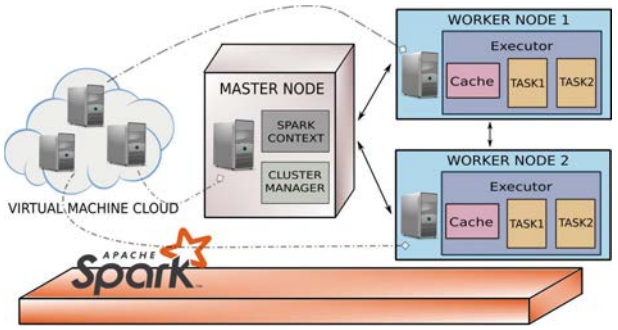


Fig. 9. ***Apache Spark execution model in a cloud computing platform.***

the *Cluster Manager*. While the normal form of executing a Spark application is to use its own native cluster manager, Spark is able to run on Hadoop clusters on top of the YARN resource manager. In any case, after the resources have been obtained, SparkContext launches an *Executor* process in each worker node. Based on the execution plan, SparkContext schedules tasks to worker nodes and coordinates its execution. Conversely, Executors are in charge of effectively executing tasks assigned to its worker nodes and providing access to data.

The *RDD* [117] is the main abstraction supporting the Spark model. An RDD is a read-only collection of objects partitioned and distributed across the worker nodes. The assignment of tasks to worker nodes considers data-locality, that is, the availability (or closeness) of the data to be processed by the task in the RDD partition assigned to the work node. An important feature of RDDs is that the data can be cached in memory and, hence, reused in recurrent parallel operations with minimum overheads. Furthermore, the RDD is a fault-tolerant data structure. In this sense, any operation on an RDD object is logged in such a way that, in the case of a node failure, the RDD can be reconstructed using the operation *lineage*. Because lineage dependencies become large and their management is time-consuming, users may decide to establish checkpoints in the execution. In addition, the immutable nature of the RDD objects benefits another fault-tolerance mechanism, i.e., the execution of backup copies (duplicates) of running tasks if failed or straggler tasks are detected.

RDDs can be created from data structures in memory or in the file system and also using data obtained from any Hadoop service, including the HDFS or databases as HBase. In addition to the mechanisms for creating RDDs, Spark includes a set of coarse-grained operations to process RDD data sets. These operations are structured in two types.

1) The first type is given by *transformation* operations that apply a function to an RDD and generate a transformed RDD data set as a result. In turn, transformations are classified into *narrow*, which involves data located in the worker node where the task executes the operation, and *wide*, which involves data across multiple worker nodes, and therefore, the required data are copied from other partitions. The movement of data is coordinated by the driver. Examples of transformation operations defined in the API are *map*, *filter*, *groupbykey*, and *reducebykey*.
2) The second type of operations is called *Actions*. Actions are operations that retrieve non-RDD values (as statistical or processed values) from RDDs, and their value is returned to the driver program. Examples of actions are *count*, *collect*, *reduce*, and *foreach*. Some of them take as an input parameter a function to be applied to the data.

Transformations are *lazily* executed, in the sense that a sequence of transformations is effectively executed when an action is performed on the transformed RDD. This mechanism improves performance in cases in which only final results (and not intermediate results of the sequence of transformations) are transferred to the driver program. Nevertheless, by default, each transformed RDD is recomputed each time an action is executed on it. The persistence mechanism of Spark allows keeping in memory the data, improving the performance of recurrent operations on the RDD (this is particularly beneficial for image processing operations involving sliding windows or kernels, which are very popular in many remote sensing applications).

Moreover, wide transformations are inefficient operations in the Spark model, consisting of independent tasks operating on their own assigned RDD partitions, because such operations require data movement. Every time a task executes an operation on remote data, SparkContext coordinates the disk I/O and network transmissions between the involved nodes. This costly procedure is called *shuffle*. The persistence capability highly improves the performance of the shuffle operations by caching in memory the RDD data that are going to be reused in wide transformations, which is also popular in multiscale image processing operations adopted in many remote sensing applications (see Section III).

Finally, Spark includes two additional mechanisms to avoid recurrent copies of shared data between the worker nodes, called *shared variables*. The first one is called *broadcast*, and it allows to diffuse a set of values to worker nodes, which will hold a read-only copy of the data. The second one allows to maintain simple (associative) *accumulators* shared across worker nodes.

D. Cloud Computing for Scientific Applications

Scientific applications (including remote sensing ones) exploit the low latency and dedicated resources of clusters to obtain high performance. On the contrary, cloud computing offers elastic multitenancy (resource time sharing) and nonpredictable and unstable network facilities. Nevertheless, there is a great interest in evaluating if cost-efficient and flexible cloud platforms can execute scientific HPC applications at a reasonable level of efficiency.

The paper [131] proposes AzureMapReduce, a decentralized MapReduce implementation for Microsoft Azure cloud infrastructures, and evaluates its (weak-scale) scalability and performance with a remote sensing application. The authors claim that MapReduce applications in cloud infrastructures exhibit comparable performance to MapReduce applications executed on traditional clusters. On the contrary, the work [132] analyzes the performance of the HPC Challenge (HPCC) benchmark [133] on the Amazon EC2 cloud platform and concludes that the performance of general scientific applications on cloud infrastructures is at least one order of magnitude lower than that on clusters and supercomputers. Moreover, the work [134] analyzes the performance of loosely coupled many-task scientific computing applications on four commercial cloud computing provider platforms with the same aforementioned conclusion.

The thorough study in [135] presents a run-time performance comparison of the characteristics of the Amazon EC2 cluster computing instances and a supercomputer. The paper evaluates latency and bandwidth microbenchmarks, HPCC matrix multiplication kernels, NAS Parallel Benchmarks (NPB [136]), and four full-scale remote sensing applications used at NASA. The results show that, in one node, performances are equivalent, while, in several nodes, the network overheads of the cloud computing infrastructure have a huge impact on performance.

The paper [125] evaluates Amazon IaaS services at different levels. The authors execute microbenchmarks to extract raw performance of latency, bandwidth, memory, and processing services. Furthermore, they execute the parallel HPL [137] benchmark to compare cluster and cloud environments. The goal was to identify the advantages and limitations of cloud platforms. They conclude that I/O and network performance differences are the main factors impacting the applications' performance. One of the main drawbacks detected in the cloud is the network infrastructure based on Ethernet, which is often not suitable for the necessities of HPC applications (including remote sensing ones). The paper [123] proposes the HPC2 model that bridges the gap between cluster and cloud platforms with a set of proposals, including using Infiniband as network technology (as it is commonly the case in current remote sensing applications implemented in cloud environments).

There is a consensus in which the performance differences between platforms come from the inherent overheads in virtualization, memory, storage and I/O, and latency of the network infrastructure [138], [139]. Furthermore, the work [140] offers an extensive study of the performance of several cloud providers of public IaaS services: Amazon EC2, Microsoft Azure, GCE, and IBM SL, and it concludes that, indeed, there are substantial differences between the performance of infrastructures of different cloud providers.

To overcome the differences between cluster and cloud platforms, several works maintain that it is not enough to straightforwardly run cluster applications on cloud platforms. This is in contrast with many cloud implementations of remote sensing algorithms described in Section III, which simply run available cluster-based codes in cloud environments. For instance, the paper [141] proposes to slightly transform HPC applications as representative HPC kernel solvers by optimizing computational granularity, which has a high impact on scheduling and communication/computation overlapping. In addition, they propose to transform cloud facilities to use thin VMs and CPU affinity mechanisms. They conclude that, by transforming HPC applications (such as remote sensing ones) to be run in a cloud and making clouds *HPC-aware*, the impact of the latency and multitenancy is significantly reduced. In this sense, the paper [142] proposes to use the MRAP model that extends MapReduce with usual HPC application data access pattern semantics (noncontiguous and fine-grained) while taking advantage of the inherent scalability and fault-tolerance features of MapReduce. This is a promising solution to increase the performance of the MapReduce-based remote sensing implementations described in Section III.

V. MACHINE AND DEEP LEARNING LIBRARIES AND FRAMEWORKS IN CLOUD COMPUTING ENVIRONMENTS

Numerous efforts have been devoted to the development and efficient execution of ML and DL applications on cloud computing infrastructures, not only as optimized libraries and services but also as applications on top of the Spark and MapReduce programming models. In this respect, cloud providers offer several facilities for this challenging task. Among them, Amazon AWS promotes an ML platform called SageMaker[32] to build and train different models, with support for TF and Spark. IBM provides tools for different frameworks, including TF and Keras.[33] GCE also supports the use of TF and provides an infrastructure based on GPU computational devices. Microsoft Azure, on the other hand, bases its services on Kubernetes and allows using accelerators for its ML/DL resources.

Nevertheless, the iterative execution pattern of ML/DL learning applications (in which the same data are recurrently operated, as in many remote sensing data processing algorithms) does not naturally adapt to established cloud computing programming models. Virtualization and network overheads are key factors impacting the efficiency of ML/DL learning applications, which are usually supported on highly optimized computing linear algebra libraries, such as BLASs [143] and high-performance networks and communications based on MPI [144] to achieve high performance.

In addition, ML and DL models have dramatically grown in terms of structural complexity and depth. Training models on huge data sets (such as those involved in remote

[32] https://aws.amazon.com/sagemaker/

[33] https://keras.io/

Table 3 Summary of Parallelization Schemes for Distributed Computing Approaches

Scheme	Cloud computing	Cluster/Grid computing
Data-parallelism	• Master node holds the data. • Data/computation are split. • Model is replicated. • Workers sends results to the master.	• Data are partitioned over executors. • Model is replicated over executors. • Executors share intermediate results.
Model-parallelism	• Master node holds the data. • Data are not partitioned. • Model/computation is split. • Workers sends results to the master.	• Data are replicated over executors. • Model/computation is partitioned. • Executors share layer outputs and results.
Hybrid-parallelism	• Master node holds the data. • Data are partitioned. • Model is split over workers. • Computation depends on data and model workload. • Workers sends results to the master.	• Data are split over executors. • Model is partitioned over executors. • Computation depends on data and model workload. • Executors share intermediate results and outputs.

sensing applications) has become a computationally very intensive (as well as a memory-consuming) task that usually requires several days even using specialized hardware, such as GPUs. To overcome this limitation, several methods for parallel training of ML and DL models have been developed. The parallelization schemes can be structured in three main groups [145], [146].

1) The first type is the *data-parallelism* scheme, in which several replicas of a model are simultaneously trained in different computational devices on disjoints partitions of the remote sensing data set.
2) The second type is called *model-parallelism*, and it is used when a model overcomes the memory capacity of one computational device; then, it has to be partitioned and deployed on several devices.
3) Finally, the last type is the *hybrid-parallelism* scheme that merges data and model-based approaches.

These parallelization schemes can be implemented as multiple distributed computing approaches. Table 3 summarizes the different characteristics of each scheme for cloud and cluster/grid computing approaches.

The rest of the section outlines the main frameworks and libraries offered by cloud providers to face the challenge of efficient training of ML and DL models when processing remotely sensed data on cloud computing infrastructures.

A. Libraries

This section provides an overview of some well-known ML and DL libraries that are used in cloud computing environments to build models efficiently. These libraries have been used in the past to process and accelerate remote sensing applications.

1) Weka: The *Weka*[34] library was developed at the University of Waikato [147]. It is a Java-based open-source library that allows building ML and DL models for several types of algorithms, including classification, clustering, and data mining. A relevant feature of the library is that it is multiplatform and even runs on lightweight devices on top of the Android OS [148]. It supports multiple programming languages with different packages and plugins, such as the DeepLearning4J,[35] the RPlugin,[36] or several Python[37] DL libraries. *Weka* was initially developed to offer a simple and easy-to-use interface. Currently, it provides a distributed version [149] implemented on top of Spark and RDDs.

2) MLlib: The *MLlib*[38] is a distributed ML/DL library that provides model training based on the data parallelism scheme [118]. It was developed in the Scala[39] programming language and supports Java, Scala, and Python programming languages. Its main features are scalability and fast implementation of numerous ML/DL algorithms, as well as linear algebra, statistics, and optimization primitives. It is built on Spark and implements efficient communication primitives for data transmissions performed by a large number of processes and training large models using the data-parallelism scheme. Some communication primitives of special interest are the *broadcast*, which efficiently distributes data over processes training the model, and the tree-structured *aggregation* primitive, which collects processed data avoiding possible bottlenecks. *MLlib* was initially designed to efficiently operate on fully distributed environments, which entails a significant performance advantage with respect to Weka [150].

The execution model of *MLlib* is based on the master–worker paradigm, where the master process acts as a *PS* and maintains a centralized copy of the global parameters of the model. It combines values received from the worker tasks after each training iteration. Tasks deployed on the computational resources of the platform process their assigned data set partitions and communicate results to the PS. The data partitions assigned to each task are processed in *batches*, being the size of each batch a key optimization parameter, which directly affects the resulting accuracy of the model and the efficiency of the training [151], [152].

B. Frameworks

This section discusses different frameworks to develop ML/DL applications in cloud environments.

1) TensorFlowOnSpark: This framework combines salient features of the TF DL library with Spark and Hadoop to provide a scalable ML/DL development and training platform [153]. It supports all TF functionalities, including asynchronous and synchronous training, data, and model-based parallelism schemes, and monitoring with TensorBoard.[40] It also enables distributed TF-based training on cloud computing clusters, with the additional goal of minimizing the amount of code refactoring required to run existing TF applications. In summary, *TensorFlowOnSpark* deploys a Spark cluster on a cloud

[34] https://weka.sourceforge.io/

[35] https://deeplearning4j.org/
[36] https://weka.sourceforge.io/packageMetaData/RPlugin/
[37] https://www.python.org/
[38] https://spark.apache.org/mllib/
[39] https://www.scala-lang.org/
[40] https://www.tensorflow.org/tensorboard

infrastructure and provides facilities for injecting both RDD and HDFS data in the TF models executed by the tasks scheduled in the worker nodes.

2) SparkTorch: This framework is intended to execute code based on the PyTorch library [154] across nodes in a Spark cluster. The distributed training works under a data-parallel paradigm and uses both tree reductions and PS mechanisms to combine partial results from tasks deployed on the cloud platform. There are two main training modes available in *SparkTorch*.

1) The first one is the *asynchronous* training mode, which ensures that the replicated models deployed in nodes are synchronized through each training iteration.
2) The second training mode, called the *Hogwild* approach [155], allows lock-free task accessing to shared memory in order to update parameter values. This mode eliminates the overheads associated with locking. However, in this mode, a task could overwrite the progress of other tasks a risk that the developers claim that could be assumed when the data to be accessed is sparse.

3) BigDL: This framework is also implemented on top of Apache Spark to run DL applications as standard Spark programs [156]. It offers support for large-scale distributed applications and provides efficient processing for data analysis, data injection to neural network models, and distributed training or inference, using a unified pipeline. Before training, the model and RDDs are partitioned and cached in memory across the cloud resources. *BigDL* supports two-parameter synchronization mechanisms. The first one maintains a centralized PS, and the second one uses collective operations as *AllReduce* to combine the parameters computed by tasks. Despite the fact that collective message passing primitives are not particularly suitable for the execution model of a Spark cluster, *BigDL* implements an efficient *AllReduce* algorithm using Spark primitives, allowing for the integration of DL algorithms in cloud computing environments.

It is important to note that the three aforementioned frameworks can use two different communication approaches.

1) The *PS approach*, as illustrated in Fig. 10(a) [157], consists of a centralized architecture where the computational nodes are partitioned into masters and workers. The workers maintain a workload and data partition, while the master maintains the global shared parameters. The workers communicate with the master to share the weights generated at each iteration of the model. The master is responsible for the aggregation of the global weights. In cloud computing environments, additional workers may be added or removed from the execution. This must be handled by the system, so as to switch on any new workers and send to them the appropriate computations and data partitions.

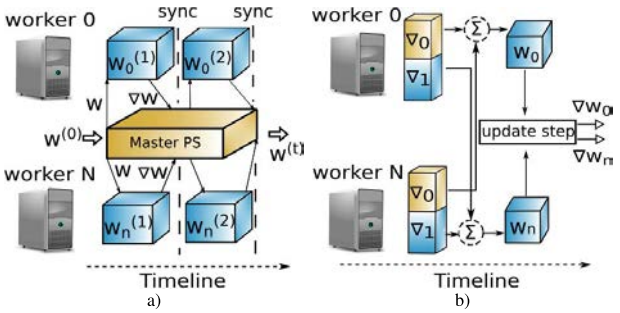


Fig. 10. ***Cloud computing training pipeline in the three discussed DL frameworks (TensorFlowOnSpark, SparkTorch, and BigDL). (a) Parameter server approach. (b) AllReduce-Ring BigDL approach.***

2) The *AllReduce-Ring BigDL approach*, as shown in Fig. 10(b), consists of a decentralized architecture where each Spark task computes its local gradients, dividing the local gradients into N partitions. Each task manages its corresponding parameter partition, which is shuffled to the corresponding task to aggregate gradients and then update the corresponding weights. Then, each task launches a Spark broadcast operation with the updated weights so that these are read before the next step.

A discussion between the aforementioned frameworks and their pros and cons in cloud environments is needed at this point.

1) In terms of applications, all of them provide full integration with Spark ML/DL algorithms. While *TensorFlowOnSpark* provides a large amount of algorithms and applications, *BigDL* includes extensive DL functionalities.
2) The ease of use differs among different frameworks. While the *TensorFlowOnSpark* interface is clear and easy to use, the documentation for *BigDL* is intuitive and provides a comprehensive support for ML/DL algorithms.
3) Attending to the distributed training, *SparkTorch* provides asynchronous and synchronous schemes since *TensorFlowOnSpark* asynchronous PS is highly efficient. We also note that *SparkTorch* is in a premature developing phase compared to the *TensorFlowOnSpark* and *BigDL* implementations. Thus, the latter frameworks still offer notable advantages. An important aspect of *TensorFlowOnSpark* is the creation of checkpoints to recover from failures. These checkpoints are stored in the HDFS by TF. A similar point between these two last frameworks can be found in the monitoring.
4) In terms of scaling and performance, *BigDL* takes a step forward from the other frameworks. This is due to multiple factors. First, it provides extensive documentation to deploy ML/DL algorithms in different providers as EC2. Also, attending to the execution, it provides a synchronous SGD and an optimized

AllReduce in the communication step. Another interesting point is that it can be used only for prediction, and hence, it can load models from different ML/DL frameworks. Finally, the aforementioned frameworks execute powerful long-running tasks, while *BigDL* uses short-running, nonblocking tasks for the model computation.

VI. CASE STUDY

This section presents a case study in which a DNN (implemented on the cloud) is used to process a large hyperspectral remote sensing image. As noted before, hyperspectral data cubes comprise a significant amount of information, which allows us to model the physical characteristics of the observed materials by analyzing the detailed spectral signatures (collected on a pixel-by-pixel basis), which provides rich information for land-cover analysis. When applied to hyperspectral images, classification methods suffer from important processing time and computing/storage constraints, resulting from the extremely high dimensionality of the data. Therefore, the implementation of such classifiers in cloud computing architectures is an effective solution. Here, we use a deep MLP [17] as the distributed classifier and perform classification experiments on a benchmark classification data set widely used in the hyperspectral imaging community.

The remainder of this section is organized as follows. First, we describe our cloud implementation of the MLP classifier. Then, we describe the hyperspectral image used for validation purposes. Then, we provide the characteristics and configuration of the cloud computing platform used for experiments. The section ends with a detailed discussion of our conducted experiments and with some remarks on the practical utility of distributed DL algorithms in remote sensing applications.

A. Distributed Multilayer Perceptron Classifier

Let us denote a hyperspectral data cube as $\mathbf{X} \in \mathbb{R}^{h\times w\times n_{\text{bands}}}$, where each data sample $\mathbf{x}_i$ is of size $h \times w$ (h being the height and w the width in pixels of the image), and each pixel can be denoted by $\mathbf{x}_i \in \mathbb{R}^{n_{\text{bands}}} = [x_{i,1}, x_{i,2}, \ldots, x_{i,n_{\text{bands}}}]$. In an MLP classifier, each layer l performs a data transformation of the weights and the input data ($\mathbf{W}^l$ and $\mathbf{x}_i$) as follows:

$$\mathbf{x}_i^{l+1} = \mathcal{H}(\mathbf{x}_i^l \cdot \mathbf{W}^l + b^l) \tag{1}$$

where $\mathbf{x}_i^{l+1}$ is a feature representation of the input data obtained by the neurons of that layer (l). Neurons are obtained as the dot product between the output from the previous layer neurons plus the bias b (shift parameter) through an activation function, such as the ReLU or the sigmoid function, among others [17]. Hence, the kth feature of the $\mathbf{x}_i^{(l+1)}$ sample can be obtained as

$$x_{i,k}^{l+1} = \mathcal{H}\left(\sum_{j=1}^{n^{l-1}} (x_{i,j}^l \cdot w_{k,j}^l) + b^l\right). \tag{2}$$

Attending to the MLP optimization step, the optimizer tries to obtain the set of parameters $\mathbf{W}$ and $bias$ that minimize the loss error. Hence, a backpropagation step calculates the gradient of the error in order to minimize the final error. At each step, the updating process is defined as follows:

$$W_{t+1} = W_t + \nabla W \tag{3}$$

where $\Delta W = \mu_t \cdot \mathbf{p}_t$, with μ being the learning rate and $\mathbf{p}$ the descent direction of the gradient at time step t. To optimize this operation, traditional methods use the information obtained from the Hessian matrix

$$\mathbf{H}_t \cdot p_t = -\nabla E(\mathbf{X}, \mathcal{W}_t) \tag{4a}$$

$$\mathbf{p}_t = -\mathbf{H}_t^{-1} \cdot \nabla E(\mathbf{X}, \mathcal{W}_t) \tag{4b}$$

$$\mathcal{W}_{t+1} = \mathcal{W}_t - \mu_t \cdot \mathbf{H}_t^{-1} \cdot \nabla E(\mathbf{X}, \mathcal{W}_t) \tag{4c}$$

where W_t is the network weight, H is the Hessian matrix, and $\nabla E(\mathbf{X}, \mathcal{W}_t)$ is the gradient of the error at step t. Regarding this, as the computation requirements are high, the optimization provided by the BFGS algorithm [158] is used, providing an estimation of the Hessian matrix changes

$$\mathcal{W}_{t+1} = \mathcal{W}_t - \mu_t \cdot \mathbf{G}_t \nabla E(\mathbf{X}, \mathcal{W}_t) \tag{5}$$

with $\mathbf{G}_t$ being the inverse Hessian approximation matrix. Since $\mathbf{G}$ is the inverse of the Hessian matrix $\mathbf{H}^{-1}$, and the approximation matrix $\mathbf{G}$ needs to be updated in each step, the BFGS method updates it using the following equation, assuming that $\mathbf{q}_t = \mathbf{H} \cdot \mathbf{p}_t$ and $\mathbf{H}^{-1} \cdot \mathbf{q}_t = \mathbf{p}_t$:

$$\mathbf{G_{t+1}} = \mathbf{G_t} + \frac{\mathbf{p_t} \cdot \mathbf{p_t^T}}{\mathbf{p_t^T} \cdot \mathbf{q_t}} - \mathbf{G_t} \cdot \frac{\mathbf{q_t} \cdot \mathbf{q_t^T}}{\mathbf{q_t^T} \cdot \mathbf{G_t} \cdot \mathbf{q_t}} \cdot \mathbf{G_t}. \tag{6}$$

This strategy is quite appropriate for hyperspectral data sets because the computation is high and the processing requires repetitively reading the original data set. In this way, a distributed cloud computing implementation can make the MLP faster and highly scalable.

Specifically, our distributed implementation reshapes the hyperspectral data from $\mathbf{X} \in \mathbb{R}^{h\times w\times n_{\text{bands}}}$ to $\mathbf{X} \in \mathbb{R}^{n_{pixels}\times n_{\text{bands}}}$. This way, each row or column collects the full representation of a pixel. The master node reads and divides the original data set over P partitions, which are assigned to the corresponding workers in the cluster. The data are stored in each worker as an RDD.

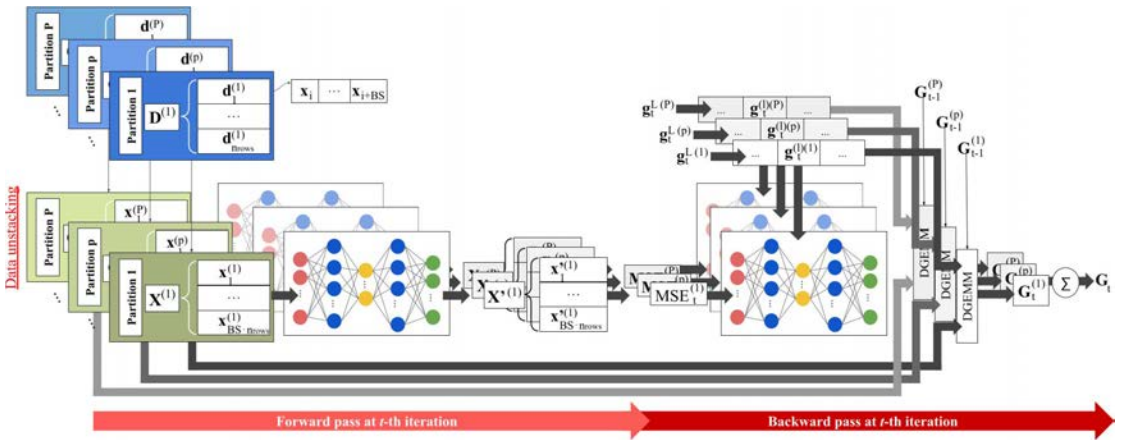


Fig. 11. ***Graphical overview of the forward and backward pipelines of our distributed MLP classifier.***

To take advantage of this, we can improve the computation time of the distributed algorithm approaches using a BS implementation. In our implementation, each data partition (row) $\mathbf{r}_j^{(p)}$ is transformed based on the minimum and maximum features of a sample with the minimum and maximum from the column of its data partition as follows:

$$\mathbf{r}_j^{(p)} = \frac{\mathbf{r}_j^{(p)} - \mathbf{r}_{\min}^{(p)}}{\mathbf{r}_{\max}^{(p)} - \mathbf{r}_{\min}^{(p)}} \cdot (\mathbf{x}_{\max} - \mathbf{x}_{\min}) + \mathbf{x}_{\min}. \tag{7}$$

Now, every training iteration can be performed using a forward–backward procedure. An aggregation is done after each step to compute and process the gradients and losses from workers and, hence, return a single gradient and loss. In each forward propagation, each worker forward its corresponding data partition $\mathbf{X}^{(p)}$ through the layers. Then, gradients are computed at the backpropagation step, obtaining, for each partition (p), the $\mathbf{G}_t^{(p)}$ matrix at time step t. Gradients are sent to the master node, and then, the computation of ΔW_t takes place. Assuming that $\mathbf{X}^{(p)} \in (\mathbb{R}^{BS \cdot n_{\text{rows}}) \times n_{\text{bands}}}$ for (p) partitions, (1) is distributed as

$$\mathbf{X}_i^{(l+1),(p)} = \mathcal{H}(\mathbf{X}^{(l),(p)} \cdot \mathbf{W}^{(l)} + b^{(l)}) \tag{8}$$

where $x_i^{(l+1),(p)}$ represents the output matrix neurons of size $(\mathbb{R}^{BS \cdot n_{\text{rows}}) \times n_{\text{neurons}}^l}$ from layer l, $x^{(l),(p)}$ is the input pixel matrix of size $(BS \cdot n_{\text{rows}})$ from the previous layer, $\mathbf{W}^{(l)}$ is the matrix of weights, which connects neurons from previous layers with the actual one, and $\mathcal{H}$ is the activation function (e.g., the ReLU).

Once the forward step is completed in every partition $\mathbf{X}^{(p)}$, the error loss is computed by every worker. The final error is calculated by the master as the mean of all the errors provided by the slaves. Then, the partition error is backpropagated to calculate the $\mathbf{G}_t^{(p)}$ gradient at each time step t. The gradients of each partition are computed using a parallel DGEMM, implemented in BLAS

$$C = \alpha * \mathbf{X}^{(p)} * g_t^{L(p)} + \beta * \mathbf{G}_{t-1}^{(p)} \tag{9}$$

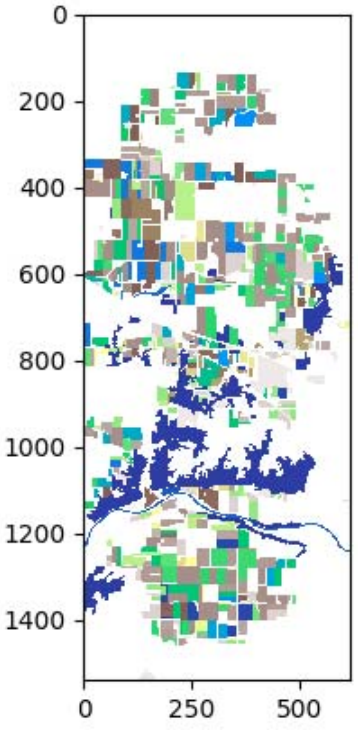
where α and β are regularization parameters set to $1/n_{\text{bands}}$ and 1, respectively, $g_t^{L(p)}$ is a matrix representing the neuron impact per layer $L = [l_1, l_2, \ldots, l_n]$, and $\mathbf{G}_{t-1}^{(p)}$ denotes the previous gradient matrix values. The variable p, as indicated previously, represents the partition of the data. In the end, all partition gradients $\mathbf{G}_t^{(p)}$ are summed to obtain the global gradient matrix $\mathbf{G}_t$ at the time step t.

Fig. 11 provides a graphical description of the distributed forward and backward pipelines of our distributed MLP during the training stage (considering iteration t). This is conducted after unstacking the hyperspectral samples in each distributed data partition, where each one is allocated to a different worker node.

B. Hyperspectral Data Set

To evaluate the performance of our cloud implementation of the MLP classifier, several experiments have been conducted over the BIP data set.[41] It was gathered by the AVIRIS sensor [2] during a flight campaign over the agricultural IP test site in Northwestern Indiana. The scene was collected at the beginning of the 1992 growing season and comprises several regular patches of different crops coupled with irregular forest and grass zones. The data cube comprises 1417 × 617 pixels, with a ground sampling distance of 20 mpp. Furthermore, each pixel comprises 220 channels recorded over a spectral range of 400–500 nm, with a nominal spectral resolution of 10 nm. However, 20 bands [0–9, 210–219] were removed in order to avoid null, noisy, and water absorption bands, keeping the remaining 200 bands for experimental purposes.

[41] https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html



Color	Land cover type	Samples	Color	Land cover type	Samples
	Background	1310047		BareSoil	57
	Buildings	17195		Concrete/Asphalt	69
	Corn	17783		Corn?	158
	Corn-EW	514		Corn-NS	2356
	Corn-CleanTill	12404		Corn-CleanTill-EW	26486
	Corn-CleanTill-NS	39678		Corn-CleanTill-NS-Irrigated	800
	Corn-CleanTilled-NS?	1728		Corn-MinTill	1049
	Corn-MinTill-EW	5629		Corn-MinTill-NS	8862
	Corn-NoTill	4381		Corn-NoTill-EW	1206
	Corn-NoTill-NS	5685		Fescue	114
	Grass	1147		Grass/Trees	2331
	Grass/Pasture-mowed	19		Grass/Pasture	73
	Grass-runway	37		Hay	1128
	Hay?	2185		Hay-Alfalfa	2258
	Lake	224		NotCropped	1940
	Oats	1742		Oats?	335
	Orchard	39		Pasture	10386
	pond	102		Soybeans	9391
	Soybeans?	894		Soybeans-NS	1110
	Soybeans-CleanTill	5074		Soybeans-CleanTill?	2726
	Soybeans-CleanTill-EW	11802		Soybeans-CleanTill-NS	10387
	Soybeans-CleanTill-Drilled	2242		Soybeans-CleanTill-Weedy	543
	Soybeans-Drilled	15118		Soybeans-MinTill	2667
	Soybeans-MinTill-EW	1832		Soybeans-MinTill-Drilled	8098
	Soybeans-MinTill-NS	4953		Soybeans-NoTill	2157
	Soybeans-NoTill-EW	2533		Soybeans-NoTill-NS	929
	Soybeans-NoTill-Drilled	8731		Swampy Area	583
	River	3110		Trees?	580
	Wheat	4979		Woods	63562
	Woods?	144			
		Total samples 1644292			

Fig. 12. ***Available labeled samples (and their distribution) in the AVIRIS BIP data set considered in experiments.***

The complexity of this challenging image is quite remarkable, as the pixels are very mixed due to the low spatial resolution, while the available ground truth is composed of 58 different and highly unbalanced land-cover classes, covering only 20.33% of the samples. The size of the data set exceeds 1 GB. Fig. 12 shows the available labeled samples per class and their distribution in the scene.

C. Platform Configuration

The designed experiments have been conducted over an OpenStack-based cloud infrastructure, which has been implemented onto a hardware platform composed of two × Intel Xeon CPUs E5-2650v2 @2.60 GHz with eight cores (16 way multitask processing), 16-GB RAM, and 600 GB of HDD SAS 10k. In this sense, within the cloud environment, nine VMs (one master instance and eight slave instances) have been launched. Each VM runs Ubuntu 20.04 as OS, with Spark 3.0.1 and Java 9.0.4 serving as running platforms. Furthermore, the Spark framework provides the distributed *MLlib* library, which is used to support the implementation of our cloud-based MLP classifier.[42]

[42]The source code used in this experimentation is currently available on https://github.com/mhaut/cloud-dnn-HSI

D. Experimental Discussion

During the experimentation, the computational load of a fully connected deep network has been distributed in order to evaluate the performance of the cloud infrastructure when dealing with hyperspectral remote sensing image classification. In particular, a deep MLP has been designed to explore the impact of its computational burden over the cloud environment by involving all model parameters with all input elements in the computation of the matrix operations described by (1).

Table 4 describes the architecture of the implemented MLP, specifying the number of layers and the number of neurons comprised by each layer. It is noteworthy that a fully connected neural model entails $\sum_i N_i N_{i+1}$ trainable parameters (where N_i represents the number of nodes at the ith layer with $i = [1, L-1]$, and L is the number of layers), which, in turn, involves approximately $\sum_i 2N_i N_{i+1}$ FLOPs. In particular, the implemented MLP comprises 78 624 trainable parameters, which implies at least 157 248 FLOPs. Furthermore, in the calculation of

Table 4 Configuration of the Implemented MLP Classifier (Number of Neurons per Layer)

Input	Hidden 1	Hidden 2	Hidden 3	Output
200	144	144	144	58

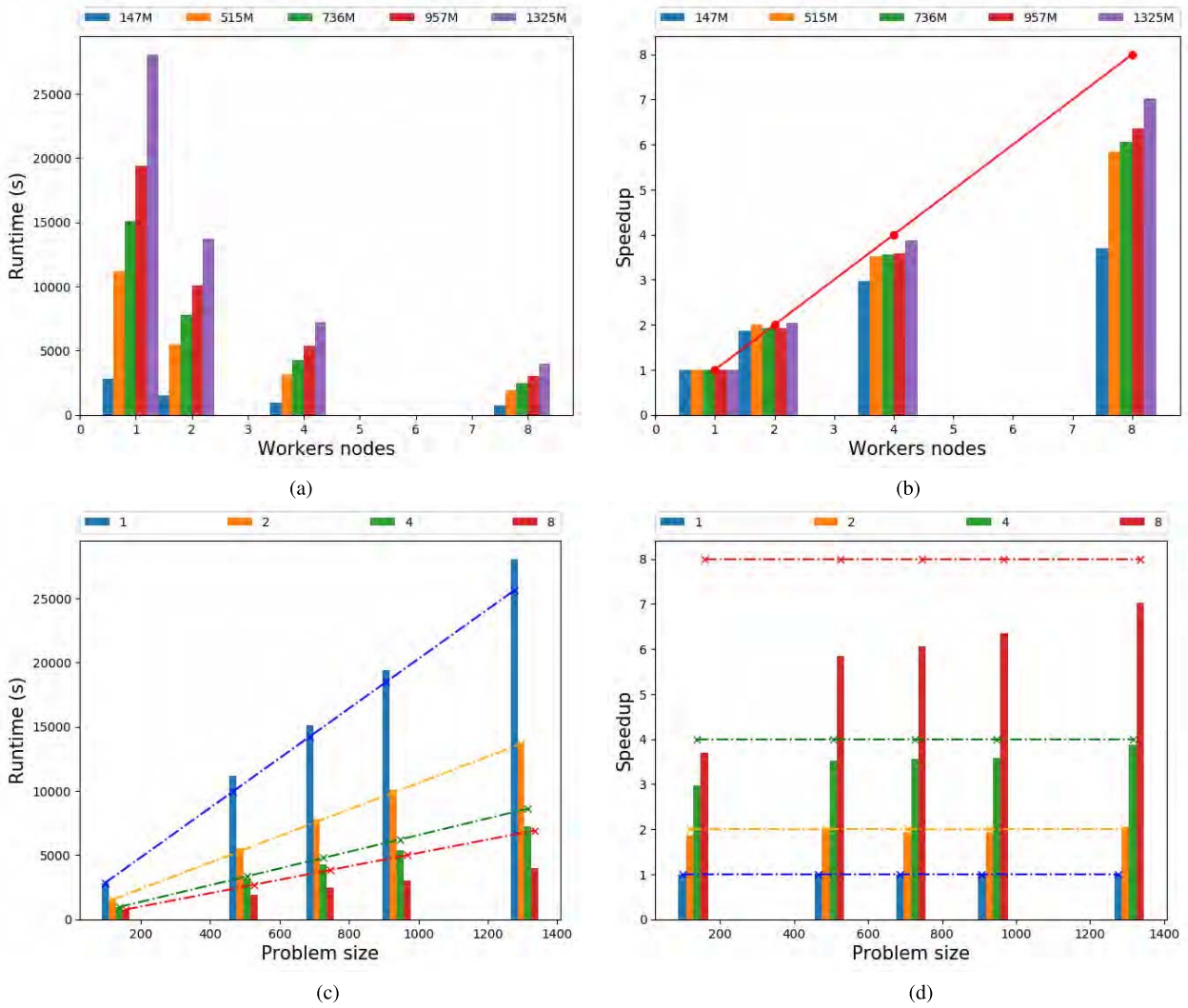


Fig. 13. ***Distributed MLP performance. (a) and (b) Runtimes and speedups as the number of workers increases, considering different amounts of training samples for each one. (c) and (d) Obtained runtimes and speedups as the problem size increases, evaluating the theoretical and obtained behavior for a different number of workers. The different problem sizes result from the selection of different amounts of training data. Depicted lines provide theoretical speedups and expected runtime measurements, while bars provide the actual values obtained.***

FLOPs, the number of samples processed by the model must be taken into account.

In addition, two approaches have been followed to assess the obtained performance in terms of runtimes and speedup. Indeed, experiments have been conducted considering both different numbers of workers and different amounts of training data, where the first approach analyses the obtained runtimes and speedup by focusing on the number of workers, while the second approach evaluates the scalability of the cloud environment by focusing on the amount of data. For each experiment, we conduct five Monte Carlo runs and report the average results.

1) Approach Based on the Cloud Environment Size: The first approach distributes the MLP over the cloud infrastructure to measure the overall performance improvement of the system by evaluating its potential with a different number of workers. In this regard, a cloud environment with one, two, four, and eight working nodes has been launched. Moreover, for each configuration, different amounts of training data have been considered. Particularly, 10%, 35%, 50%, 65%, and 90% of the available labeled data have been randomly selected from the BIP data set to learn the 78 624 trainable parameters comprised by the MLP.

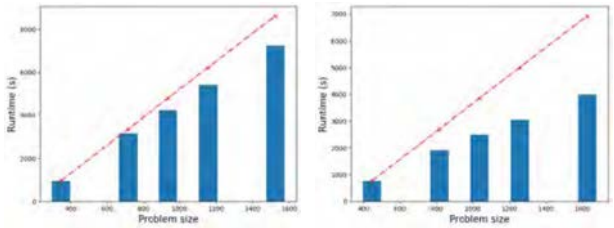
Fig. 13(a) shows a graphical representation of the obtained results in terms of runtime. As we can observe, for each configuration of the cloud infrastructure, five measurements have been collected, which corresponds to the number of training samples. In this regard, focusing on each configuration, the obtained runtimes increase as more data are included during the training stage. This is

a reasonable and expected behavior since, with the same resources, an increase in the problem size brings a corresponding increase in the runtime. However, comparing the runtimes among the different configurations, with only one worker, the execution time soars, easily exceeding 20 000 s with 90% of training samples (purple bar). On the contrary, as the number of workers increases, the observed runtimes decrease significantly. In particular, with eight workers, the highest runtime with 90% of the training samples never exceeds 4000 s.

These results have a clear impact on the speedup of the MLP model, which is increased as more workers are launched into the cloud environment. Fig. 13(b) shows a graphical description of the obtained speedups. Once again, for each configuration, five measurements are depicted corresponding with different training percentages. In this regard, the cloud infrastructure with one worker node has been considered as the baseline configuration, which exhibits a speedup of 1 for all training sizes. Therefore, the speedups of the following configurations have been consequently obtained. It should be noted that the improvement in speedup is not exactly the same as the theoretical speedup marked by the red line in Fig. 13(b), as communication and scheduling times inevitably affect the system performance. However, the data volume is high enough to take full advantage of the cloud environment, without the communication bottleneck preventing a good performance result in terms of runtimes. This is clearly evident in every configuration of the cloud environment. For instance, focusing on configurations with two and four workers, the speedups obtained with different training percentages are quite similar, suggesting that, already, with 10%–50% of training data, computational resources are being optimally exploited. However, with eight working nodes, the speedup is significantly higher when more data are processed, as computational resources are much larger, thus providing more room to exploit a bigger amount of data (i.e., to process larger data sets).

2) Approach Based on Problem Size: As mentioned above, the second approach evaluates the behavior of the implemented cloud solution by placing the focus on the problem size, i.e., by considering different training set sizes. In this regard, the experiment attempts to measure the scalability of each of the cloud environment configurations adopted to solve the problem (with one, two, four, and eight nodes) when different amounts of labeled samples are considered at the training stage. As in the previous experiment, 10%, 35%, 50%, 65%, and 90% of the available labeled samples have been randomly selected to comprise the training sets, resulting in different data sizes (in MBs), which are indicated on the x-axis of the plots reported in Fig. 13(c) and (d).

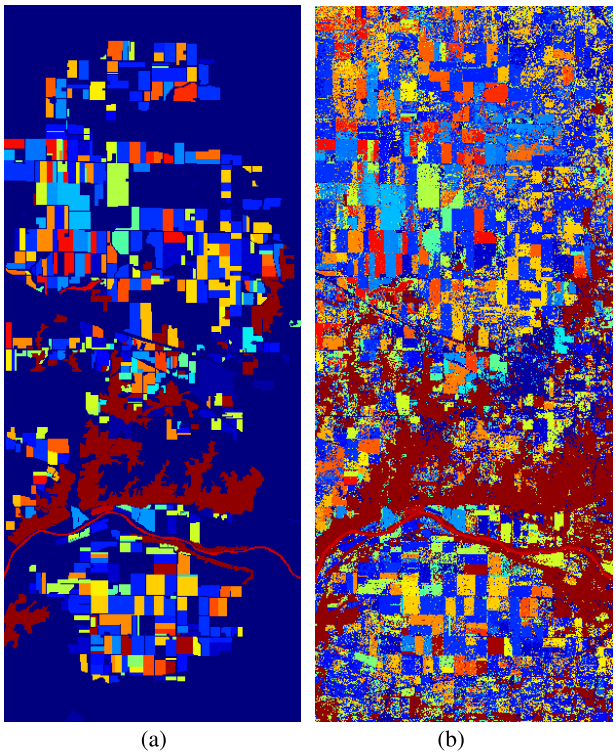
Fig. 13(c) provides the obtained results in terms of runtimes. Following the previous results, for each amount of training data, runtimes decrease as more workers are launched into the cloud environment. In this sense,



(a) (b)

Fig. 14. ***Runtime details of the distributed MLP as the problem size grows, with (a) four and (b) eight working nodes. The different problem sizes result from the selection of different amounts of training data. Red lines provide expected measurements, while blue bars provide the actual measured values.***

the one-worker configuration is the slowest one, with runtimes that are more than seven times longer than the ones obtained by the eight-worker configuration for the case with the most training data. In this sense, if we connect the top of each bar, it is particularly interesting to observe the slope of the line, where the one corresponding to the one-worker configuration is the steepest. It even exceeds the theoretical line, which makes it the worst configuration. On the contrary, the four- and eight-worker configurations scale remarkably well. Moreover, they even give better times than theoretically expected. Fig. 14(a) and (b)



(a) (b)

Fig. 15. ***Classification map obtained by the cloud-based MLP for the BIP data set using 50% of the available labeled data for training. (a) Ground truth. (b) Predicted labels.***

Table 5 OA Comparison

Nodes	10%	35%	50%	65%	90%
1	63.65±0.36	77.57±0.62	81.27±0.29	83.04±0.21	85.16±0.34
2	63.50±0.28	77.67±0.63	80.99±0.14	83.25±0.26	85.39±0.08
4	63.62±0.48	77.56±0.12	81.15±0.38	83.22±0.33	85.80±0.09
8	63.24±0.51	77.76±0.34	81.25±0.18	83.44±0.11	85.40±0.36
Parallel (PyTorch)	61.05 ±1.81	77.45 ±0.49	80.50 ±0.34	82.24 ±0.38	84.37 ±0.33

provides the runtime details for these configurations. As it can be observed in Fig. 14(a), although the runtime increases (as expected), the curve is not entirely linear, which implies that the distribution model is more optimal when there are more data to distribute. Moreover, the distance between the obtained runtimes and the theoretical ones (which are highlighted as a red line) increases as more data are processed. Particularly, the theoretical runtimes times are 1.18 times higher than the obtained ones. This proves that the model scales appropriately with the size of the problem. This is clearly visible in Fig. 14(b), where theoretical runtimes are 1.59 times higher than those currently obtained.

Finally, Fig. 13(d) provides the obtained speedups regarding the problem size. Once more, the one-worker configuration has been considered as the baseline where, for each size of the training set, its speedup is set to 1. Therefore, the speedups of the two-, four-, and eight-worker configurations have been consequently obtained. As in the previous plots, the theoretical speedups have been marked as dotted lines for each configuration. In this sense, the speedup exhibited by the two-worker configuration is quite close to the theoretical one, while, for the four- and eight-worker configurations, their speedups improve as the size of the processed data grows. This is particularly evident when eight working nodes are launched into the cloud environment. These results indicate that the computational resources provided by the eight-worker configuration exhibit great scalability, with a great potential to process bigger remote sensing data sets in order to optimize the use of the cloud environment capacities.

3) Accuracy Evaluation: Finally, the reliability of the classification results has been measured in terms of overall accuracy (OA). In this sense, Table 5 provides the OAs that have been obtained after training the deep MLP with 10%, 35%, 50%, 65%, and 90% of randomly selected samples. Furthermore, for the cloud-distributed MLP, configurations with one, two, four, and eight working nodes have been considered. These results have been compared with a parallel implementation based on the PyTorch framework. As we can observe, the obtained OAs improve as the MLP network is trained with more samples. Moreover, after comparing the different implementations, we can see that the obtained results are quite similar. Fig. 15 provides a classification map that has been obtained by training the cloud-based MLP with 50% of the available labeled samples. In this regard, the cloud solution not only provides an efficient way to distribute the storage and computation load but also reaches good performance in terms of accuracy.

To conclude this section, we emphasize that the results that have been obtained with the MLP are perfectly extrapolable to other deep networks, such as CNNs. Indeed, the cloud environment can run both architectures in a distributed manner (as it is not specialized hardware), reaching impressive performance in image analysis with CNNs as well [159].

VII. CONCLUSION AND FUTURE LINES

In this article, we have presented a comprehensive review of recent efforts in parallel and distributed processing of remotely sensed images, with a particular emphasis on DL-based approaches and their cloud implementation. Our review reflects the growing importance of using cloud computing techniques for distributed processing of remote sensing images, which is of great importance due to the current availability of open big remote sensing data repositories. Our review also summarized the processing tools and techniques that have been used in different remote sensing applications, which is believed to provide a useful guideline for new users that wish to develop computationally efficient techniques in this field.

We provide a case study illustrating the results obtained by a DL algorithm (implemented in the cloud) when processing a big hyperspectral image. Since hyperspectral data are characterized by their large size and complex processing and storage requirements, we believe that the results provided in our case study offer a good perspective on the possibilities of implementing DL algorithms in the cloud for addressing the processing challenges involved in the extraction of information from remotely sensed images. In the future, we will expand our study by considering the inclusion and optimization of specific accelerators (such as GPUs) in the cloud environment for DL-based remote sensing data processing and interpretation. ■

REFERENCES

[1] B. Zhang *et al.*, "Remotely sensed big data: Evolution in model development for information extraction [point of view]," *Proc. IEEE*, vol. 107, no. 12, pp. 2294–2301, Dec. 2019.

[2] J. Li, J. A. Benediktsson, B. Zhang, T. Yang, and A. Plaza, "Spatial technology and social media in remote sensing: A survey," *Proc. IEEE*, vol. 105, no. 10, pp. 1855–1864, Oct. 2017.

[3] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York, NY, USA: Kluwer, 2003.

[4] A. Plaza, J. Plaza, A. Paz, and S. Sanchez, "Parallel

hyperspectral image and signal processing [applications corner]," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 119–126, May 2011.
[5] R. O. Green *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, no. 3, pp. 227–248, Sep. 1998.
[6] M. Chi, A. Plaza, J. A. Benediktsson, Z. Sun, J. Shen, and Y. Zhu, "Big data for remote sensing: Challenges and opportunities," *Proc. IEEE*, vol. 104, no. 11, pp. 2207–2219, Nov. 2016.
[7] Y. M. L. Wang and J. Yang, *Cloud Computing in Remote Sensing*. Boca Raton, FL, USA: CRC Press, 2019.
[8] C.-I. C. A. Plaza, *High Performance Computing in Remote Sensing*. Boca Raton, FL, USA: CRC Press, 2006.
[9] J. Behnke, T. H. Watts, B. Kobler, D. Lowe, S. Fox, and R. Meyer, "EOSDIS petabyte archives: Tenth anniversary," in *Proc. 22nd IEEE/13th NASA Goddard Conf. Mass Storage Syst. Technol. (MSST)*, Apr. 2005, pp. 81–93.
[10] B. Ryan and D. Cripe, "The Group on Earth Observations (GEO) through 2025," in *Proc. 40th COSPAR Sci. Assembly*, vol. 40, Jan. 2014, pp. 1–14.
[11] P. M. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. Sp 800-145, 2011.
[12] J. Plaza, R. Pérez, A. Plaza, P. Martínez, and D. Valencia, "Parallel morphological/neural processing of hyperspectral images using heterogeneous and homogeneous platforms," *Cluster Comput.*, vol. 11, no. 1, pp. 17–32, Mar. 2008.
[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[14] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016.
[15] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS J. Photogramm. Remote Sens.*, vol. 152, pp. 166–177, Jun. 2019.
[16] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.
[17] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review," *ISPRS J. Photogramm. Remote Sens.*, vol. 158, pp. 279–317, Dec. 2019.
[18] A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 528–544, Sep. 2011.
[19] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 508–527, Sep. 2011.
[20] A. F. H. Goetz, "Three decades of hyperspectral remote sensing of the Earth: A personal view," *Remote Sens. Environ.*, vol. 113, pp. S5–S16, Sep. 2009.
[21] N. Yokoya, C. Grohnfeldt, and J. Chanussot, "Hyperspectral and multispectral data fusion: A comparative review of the recent literature," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 2, pp. 29–56, Jun. 2017.
[22] C. Liu, J. Shang, P. W. Vachon, and H. McNairn, "Multiyear crop monitoring using polarimetric RADARSAT-2 data," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 4, pp. 2227–2240, Apr. 2013.
[23] A. Rosenqvist, M. Shimada, N. Ito, and M. Watanabe, "ALOS PALSAR: A pathfinder mission for global-scale monitoring of the environment," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 11, pp. 3307–3316, Nov. 2007.
[24] M. Moghaddam *et al.*, "Airborne microwave observatory of subcanopy and subsurface (AirMOSS) Earth venture suborbital mission overview," in *Proc. AGUFM*, 2015, pp. B53A–0537.
[25] W. Pitz and D. Miller, "The TerraSAR-X satellite," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 2, pp. 615–622, Feb. 2010.
[26] D. Geudtner, R. Torres, P. Snoeij, M. Davidson, and B. Rommen, "Sentinel-1 system capabilities and applications," in *Proc. IEEE Geosci. Remote Sens. Symp.*, Jul. 2014, pp. 1457–1460.
[27] D. Yi, J. P. Harbeck, S. S. Manizade, N. T. Kurtz, M. Studinger, and M. Hofton, "Arctic sea ice freeboard retrieval with waveform characteristics for NASA's airborne topographic mapper (ATM) and land, vegetation, and ice sensor (LVIS)," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1403–1410, Mar. 2015.
[28] N. Levin, K. Johansen, J. M. Hacker, and S. Phinn, "A new source for high spatial resolution night time images—The EROS-B commercial satellite," *Remote Sens. Environ.*, vol. 149, pp. 1–12, Jun. 2014.
[29] Y. Chen, R. Fan, M. Bilal, X. Yang, J. Wang, and W. Li, "Multilevel cloud detection for high-resolution remote sensing imagery using multiple convolutional neural networks," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 5, p. 181, May 2018.
[30] G. Dial, H. Bowen, F. Gerlach, J. Grodecki, and R. Oleszczuk, "IKONOS satellite, imagery, and products," *Remote Sens. Environ.*, vol. 88, nos. 1–2, pp. 23–36, Nov. 2003.
[31] M. Gašparović, L. Rumora, M. Miler, and D. Medak, "Effect of fusing Sentinel-2 and WorldView-4 imagery on the various vegetation indices," *Proc. SPIE*, vol. 13, no. 3, Jul. 2019, Art. no. 036503.
[32] M. Drusch *et al.*, "Sentinel-2: ESA's optical high-resolution mission for GMES operational services," *Remote Sens. Environ.*, vol. 120, pp. 25–36, May 2012.
[33] C. Donlon *et al.*, "The global monitoring for environment and security (GMES) Sentinel-3 mission," *Remote Sens. Environ.*, vol. 120, pp. 37–57, May 2012.
[34] D. P. Roy *et al.*, "Landsat-8: Science and product vision for terrestrial global change research," *Remote Sens. Environ.*, vol. 145, pp. 154–172, Apr. 2014.
[35] A. Savtchenko *et al.*, "Terra and Aqua MODIS products available from NASA GES DAAC," *Adv. Space Res.*, vol. 34, no. 4, pp. 710–714, Jan. 2004.
[36] M. A. Cutter, D. R. Lobb, and R. A. Cockshott, "Compact high resolution imaging spectrometer (CHRIS)," *Acta Astronautica*, vol. 46, nos. 2–6, pp. 263–268, Jan. 2000.
[37] J. W. Chapman *et al.*, "Spectral and radiometric calibration of the next generation airborne visible infrared spectrometer (AVIRIS-NG)," *Remote Sens.*, vol. 11, no. 18, p. 2129, Sep. 2019.
[38] B. Kunkel, F. Blechinger, R. Lutz, R. Doerffer, H. van der Piepen, and M. Schroder, "ROSIS (Reflective Optics System Imaging Spectrometer)—A candidate instrument for polar platform missions," in *Optoelectronic Technologies for Remote Sensing From Space*, vol. 0868, J. Seeley and S. Bowyer, Eds. Bellingham, WA, USA: SPIE, 1988, p. 8.
[39] S. K. Babey and C. D. Anger, "Compact airborne spectrographic imager (CASI): A progress review," in *Imaging Spectrometry of the Terrestrial Environment*, vol. 1937, G. Vane, Ed. Bellingham, WA, USA: SPIE, 1993, pp. 152–163, doi: 10.1117/12.157052.
[40] L. J. Rickard, R. W. Basedow, E. F. Zalewski, P. R. Silverglate, and M. Landers, "HYDICE: An airborne system for hyperspectral imaging," *Proc. SPIE*, vol. 1937, pp. 173–180, Sep. 1993.
[41] T. Cocks, R. Jenssen, A. Stewart, I. Wilson, and T. Shields, "The HyMap airborne hyperspectral sensor: The system, calibration and performance," in *Proc. 1st EARSeL Workshop Imag. Spectrosc.*, 1998, pp. 37–42.
[42] P. Mouroulis *et al.*, "Portable remote imaging spectrometer coastal ocean sensor: Design, characteristics, and first flight results," *Appl. Opt.*, vol. 53, no. 7, pp. 1363–1380, 2014.
[43] L. Guanter *et al.*, "The EnMAP spaceborne imaging spectroscopy mission for Earth observation," *Remote Sens.*, vol. 7, no. 7, pp. 8830–8857, Jul. 2015.
[44] N. Yokoya and A. Iwasaki, "Hyperspectral and multispectral data fusion mission on hyperspectral imager suite (HISUI)," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2013, pp. 4086–4089.
[45] A. Eckardt *et al.*, "DESIS (DLR Earth sensing imaging spectrometer for the ISS-MUSES platform)," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 1457–1459.
[46] J. S. Pearlman, P. S. Barry, C. C. Segal, J. Shepanski, D. Beiso, and S. L. Carman, "Hyperion, a space-based imaging spectrometer," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 6, pp. 1160–1173, Jun. 2003.
[47] C. Galeazzi, A. Sacchetti, A. Cisbani, and G. Babini, "The PRISMA program," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, vol. 4, Jul. 2008, pp. IV–105.
[48] T. Feingersh and E. Ben Dor, "SHALOM—A commercial hyperspectral space mission," in *Optical Payloads for Space Missions*. Hoboken, NJ, USA: Wiley, 2015, ch. 11, pp. 247–263. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118945179.ch11, doi: 10.1002/9781118945179.ch11.
[49] G. A. Swayze, "Effects of spectrometer band pass, sampling, and signal-to-noise ratio on spectral identification using the Tetracorder algorithm," *J. Geophys. Res.*, vol. 108, no. E9, pp. 5105–5135, 2003.
[50] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for Earth remote sensing," *Science*, vol. 228, no. 4704, pp. 1147–1153, 1985.
[51] S. K. Seelan, S. Laguette, G. M. Casady, and G. A. Seielstad, "Remote sensing applications for precision agriculture: A learning community approach," *Remote Sens. Environ.*, vol. 88, nos. 1–2, pp. 157–169, Nov. 2003.
[52] H. M. Pham, Y. Yamaguchi, and T. Q. Bui, "A case study on the relation between city planning and urban growth using remote sensing and spatial metrics," *Landscape Urban Planning*, vol. 100, no. 3, pp. 223–230, Apr. 2011.
[53] S. A. Azzouzi, A. Vidal-Pantaleoni, and H. A. Bentounes, "Desertification monitoring in Biskra, Algeria, with landsat imagery by means of supervised classification and change detection methods," *IEEE Access*, vol. 5, pp. 9065–9072, 2017.
[54] X. Ceamanos and S. Valero, "Processing hyperspectral images," in *Optical Remote Sensing of Land Surface*. Amsterdam, The Netherlands: Elsevier, 2016, pp. 163–200.
[55] A. Maffei, J. M. Haut, M. E. Paoletti, J. Plaza, L. Bruzzone, and A. Plaza, "A single model CNN for hyperspectral image denoising," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 4, pp. 2516–2529, Apr. 2020.
[56] G. Cheng, X. Xie, J. Han, L. Guo, and G.-S. Xia, "Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities," 2020, *arXiv:2005.01094*. [Online]. Available: http://arxiv.org/abs/2005.01094
[57] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 120–147, Nov. 2018.
[58] V. Walter, "Object-based classification of remote sensing data for change detection," *ISPRS J. Photogramm. Remote Sens.*, vol. 58, nos. 3–4, pp. 225–238, Jan. 2004.
[59] K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification," *Pattern Recognit.*, vol. 61, pp. 539–556, Jan. 2017.

[60] J. Plaza, A. Plaza, R. Perez, and P. Martinez, "On the use of small training sets for neural network-based characterization of mixed pixels in remotely sensed hyperspectral images," *Pattern Recognit.*, vol. 42, no. 11, pp. 3032–3045, Nov. 2009.

[61] J. A. G. Jaramago, M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, A. Plaza, and J. Plaza, "GPU parallel implementation of dual-depth sparse probabilistic latent semantic analysis for hyperspectral unmixing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 9, pp. 3156–3167, Sep. 2019.

[62] G. Camps-Valls, "Machine learning in remote sensing data processing," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, Sep. 2009, pp. 1–6.

[63] D. J. Lary, A. H. Alavi, A. H. Gandomi, and A. L. Walker, "Machine learning in geosciences and remote sensing," *Geosci. Frontiers*, vol. 7, no. 1, pp. 3–10, 2016.

[64] J. M. Haut, R. Fernandez-Beltran, M. E. Paoletti, J. Plaza, A. Plaza, and F. Pla, "A new deep generative network for unsupervised remote sensing single-image super-resolution," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6792–6810, Nov. 2018.

[65] D. Tuia and G. Camps-Valls, "Semisupervised remote sensing image classification with cluster kernels," *IEEE Geosci. Remote Sens. Lett.*, vol. 6, no. 2, pp. 224–228, Apr. 2009.

[66] D. Tuia, M. Volpi, L. Copa, M. Kanevski, and J. Munoz-Mari, "A survey of active learning algorithms for supervised remote sensing image classification," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 3, pp. 606–617, Jun. 2011.

[67] Y. Li, K. Fu, H. Sun, and X. Sun, "An aircraft detection framework based on reinforcement learning and convolutional neural networks in remote sensing images," *Remote Sens.*, vol. 10, no. 2, p. 243, Feb. 2018.

[68] J. M. Haut, M. Paoletti, J. Plaza, and A. Plaza, "Cloud implementation of the K-means algorithm for hyperspectral image analysis," *J. Supercomput.*, vol. 73, no. 1, pp. 514–529, Jan. 2017.

[69] E. Blanzieri and F. Melgani, "Nearest neighbor classification of remote sensing images with the maximal margin principle," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 6, pp. 1804–1811, Jun. 2008.

[70] S. Delalieux, B. Somers, B. Haest, T. Spanhove, J. Vanden Borre, and C. A. Mücher, "Heathland conservation status mapping through integration of hyperspectral mixture analysis and decision tree classifiers," *Remote Sens. Environ.*, vol. 126, pp. 222–231, Nov. 2012.

[71] K. Y. Peerbhay, O. Mutanga, and R. Ismail, "Random forests unsupervised classification: The detection and mapping of solanum mauritianum infestations in plantation forestry using hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 3107–3122, Jun. 2015.

[72] J. M. Haut and M. E. Paoletti, "Cloud implementation of multinomial logistic regression for UAV hyperspectral images," *IEEE J. Miniaturization Air Space Syst.*, vol. 1, no. 3, pp. 163–171, Dec. 2020.

[73] J. Ju, E. D. Kolaczyk, and S. Gopal, "Gaussian mixture discriminant analysis and sub-pixel land cover characterization in remote sensing," *Remote Sens. Environ.*, vol. 84, no. 4, pp. 550–560, Apr. 2003.

[74] J. Yang, Z. Ye, X. Zhang, W. Liu, and H. Jin, "Attribute weighted naive Bayes for remote sensing image classification based on cuckoo search algorithm," in *Proc. Int. Conf. Secur., Pattern Anal., Cybern. (SPAC)*, Dec. 2017, pp. 169–174.

[75] P. B. C. Leite, R. Q. Feitosa, A. R. Formaggio, G. A. O. P. da Costa, K. Pakzad, and I. D. Sanches, "Hidden Markov models for crop recognition in remote sensing image sequences," *Pattern Recognit. Lett.*, vol. 32, no. 1, pp. 19–26, Jan. 2011.

[76] M. E. Paoletti, J. M. Haut, X. Tao, J. P. Miguel, and A. Plaza, "A new GPU implementation of support vector machines for fast hyperspectral image classification," *Remote Sens.*, vol. 12, no. 8, p. 1257, Apr. 2020.

[77] S. Yang, Q. Feng, T. Liang, B. Liu, W. Zhang, and H. Xie, "Modeling grassland above-ground biomass based on artificial neural network and remote sensing in the three-river headwaters region," *Remote Sens. Environ.*, vol. 204, pp. 448–455, Jan. 2018.

[78] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[79] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Scalable recurrent neural network for hyperspectral image classification," *J. Supercomput.*, vol. 76, pp. 8866–8882, Feb. 2020.

[80] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, and F. Pla, "Deep pyramidal residual networks for spectral–spatial hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 740–754, Feb. 2019.

[81] M. E. Paoletti *et al.*, "Capsule networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, Apr. 2019.

[82] D. Lunga, J. Gerrand, L. Yang, C. Layton, and R. Stewart, "Apache spark accelerated deep learning inference for large scale satellite image analytics," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 271–283, 2020.

[83] M. Ma *et al.*, "Democratizing production-scale distributed deep learning," 2018, *arXiv:1811.00143*. [Online]. Available: https://arxiv.org/abs/1811.00143

[84] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, and J. Benediktsson, "Remote sensing big data classification with high performance distributed deep learning," *Remote Sens.*, vol. 11, no. 24, p. 3056, Dec. 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/24/3056

[85] Y. Lu, K. Xie, G. Xu, H. Dong, C. Li, and T. Li, "MTFC: A multi-GPU training framework for cube-CNN-based hyperspectral image classification," *IEEE Trans. Emerg. Topics Comput.*, early access, Aug. 17, 2020, doi: 10.1109/TETC.2020.3016978.

[86] M. Aspri, G. Tsagkatakis, and P. Tsakalides, "Distributed training and inference of deep learning models for multi-modal land cover classification," *Remote Sens.*, vol. 12, no. 17, p. 2670, Aug. 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/17/2670

[87] Y. Li, Z. Wu, J. Wei, A. Plaza, J. Li, and Z. Wei, "Fast principal component analysis for hyperspectral imaging based on cloud computing," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Milan, Italy, Jul. 2015, pp. 513–516.

[88] Z. Wu, Y. Li, A. Plaza, J. Li, F. Xiao, and Z. Wei, "Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 6, pp. 2270–2278, Jun. 2016.

[89] J. Gu, Z. Wu, Y. Li, Y. Chen, Z. Wei, and W. Wang, "Parallel optimization of pixel purity index algorithm for hyperspectral unmixing based on spark," in *Proc. 3rd Int. Conf. Adv. Cloud Big Data*, Oct. 2015, pp. 159–166. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7435468

[90] C. Bielski, S. Gentilini, and M. Pappalardo, "Post-disaster image processing for damage analysis using GENESI-DR, WPS and grid computing," *Remote Sens.*, vol. 3, no. 6, pp. 1234–1250, Jun. 2011. [Online]. Available: https://www.mdpi.com/2072-4292/3/6/1234

[91] 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.

[92] Z. Wu *et al.*, "Scheduling-guided automatic processing of massive hyperspectral image classification on cloud computing architectures," *IEEE Trans. Cybern.*, early access, Oct. 29, 2020, doi: 10.1109/TCYB.2020.3026673.

[93] L. Parente, E. Taquary, A. Silva, C. Souza, and L. Ferreira, "Next generation mapping: Combining deep learning, cloud computing, and big remote sensing data," *Remote Sens.*, vol. 11, no. 23, p. 2881, Dec. 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/23/2881

[94] X. Yao *et al.*, "Enabling the big Earth observation data via cloud computing and DGGS: Opportunities and challenges," *Remote Sens.*, vol. 12, no. 1, p. 62, Dec. 2019. [Online]. Available: https://www.mdpi.com/2072-4292/12/1/62

[95] P. Zheng *et al.*, "A parallel unmixing-based content retrieval system for distributed hyperspectral imagery repository on cloud computing platforms," *Remote Sens.*, vol. 13, no. 2, p. 176, Jan. 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/2/176

[96] J. M. Bioucas-Dias *et al.*, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 354–379, Apr. 2012.

[97] W. Huang, L. Meng, D. Zhang, and W. Zhang, "In-memory parallel processing of massive remotely sensed data using an Apache spark on Hadoop YARN model," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 1, pp. 3–19, Jan. 2017.

[98] V. A. Ayma *et al.*, "On the architecture of a big data classification tool based on a map reduce approach for hyperspectral image analysis," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 1508–1511.

[99] P. Bajcsy, P. Nguyen, A. Vandecreme, and M. Brady, "Spatial computations over terabyte-sized images on Hadoop platforms," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2014, pp. 816–824.

[100] P. N. Happ, R. S. Ferreira, G. A. O. P. Costa, R. Q. Feitosa, C. Bentes, and P. Gamba, "Towards distributed region growing image segmentation based on MapReduce," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 4352–4355. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7326790

[101] J. Xing and R. Sieber, "Sampling based image splitting in large scale distributed computing of Earth observation data," in *Proc. IEEE Geosci. Remote Sens. Symp.*, Jul. 2014, pp. 1409–1412. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84911366834&partnerID=tZOtx3y1

[102] X. Pan and S. Zhang, "A remote sensing image cloud processing system based on Hadoop," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Intell. Syst.*, vol. 1, Oct. 2012, pp. 492–494.

[103] Y. Liu, B. Chen, W. He, and Y. Fang, "Massive image data management using HBase and MapReduce," in *Proc. 21st Int. Conf. Geoinformatics*, Jun. 2013.

[104] R. Rajak, D. Raveendran, M. C. Bh, and S. S. Medasani, "High resolution satellite image processing using Hadoop framework," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets (CCEM)*, Nov. 2015, pp. 16–21. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7436925

[105] M. M. Rathore, A. Ahmad, A. Paul, and A. Daniel, "Hadoop based real-time big data architecture for remote sensing Earth observatory system," in *Proc. 6th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, vol. 1, Jul. 2015, pp. 1–7.

[106] W. Nina *et al.*, "A new approach to the massive processing of satellite images," in *Proc. 41st Latin Amer. Comput. Conf. (CLEI)*, Oct. 2015, pp. 1–6.

[107] Y. Zhong, J. Fang, and X. Zhao, "VegaIndexer: A distributed composite index scheme for big

spatio-temporal sensor data on cloud," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2013, pp. 1713–1716.

[108] R. Kune, P. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya, "XHAMI—Extended HDFS and MapReduce interface for image processing applications," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets (CCEM)*, Nov. 2015, pp. 43–51.

[109] M. T. Patterson *et al.*, "The Matsu wheel: A cloud-based framework for efficient analysis and reanalysis of Earth satellite imagery," in *Proc. IEEE 2nd Int. Conf. Big Data Comput. Service Appl. (BigDataService)*, Mar. 2016, pp. 156–165.

[110] A. Plaza, D. Valencia, J. Plaza, and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *J. Parallel Distrib. Comput.*, vol. 66, no. 3, pp. 345–358, 2006.

[111] G. Aloisio and M. Cafaro, "A dynamic Earth observation system," *Parallel Comput.*, vol. 29, no. 10, pp. 1357–1362, Oct. 2003.

[112] D. Gorgan, V. Bacu, T. Stefanut, D. Rodila, and D. Mihon, "Grid based satellite image processing platform for Earth observation application development," in *Proc. IEEE Int. Workshop Intell. Data Acquisition Adv. Comput. Syst., Technol. Appl.*, Sep. 2009, pp. 247–252.

[113] Z. Chen, N. Chen, C. Yang, and L. Di, "Cloud computing enabled Web processing service for Earth observation data processing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 6, pp. 1637–1649, Dec. 2012.

[114] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–10.

[115] V. K. Vavilapalli *et al.*, "Apache Hadoop YARN: Yet another resource negotiator," in *Proc. 4th Annu. Symp. Cloud Comput.*, Oct. 2013, pp. 1–3, doi: 10.1145/2523616.2523633.

[116] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*, 2010, p. 10.

[117] M. Zaharia, M. Chowdhury, T. Das, and A. Dave, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2012, p. 2. [Online]. Available: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

[118] X. Meng *et al.*, "MLlib: Machine learning in Apache spark," 2015, *arXiv:1505.06807*. [Online]. Available: http://arxiv.org/abs/1505.06807

[119] D. Marinescu, *Cloud Computing: Theory and Practice*. Amsterdam, The Netherlands: Elsevier, 2017. [Online]. Available: https://books.google.es/books?id=O9smDwAAQBAJ

[120] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010, doi: 10.1145/1721654.1721672.

[121] J. E. Smith and R. Nair, "The architecture of virtual machines," *Computer*, vol. 38, no. 5, pp. 32–38, May 2005.

[122] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, security threats, and solutions," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 1–39, Feb. 2013, doi: 10.1145/2431211.2431216.

[123] V. Mauch, M. Kunze, and M. Hillenbrand, "High performance cloud computing," *Future Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1408–1416, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X12000647

[124] (2009). *Security Guidance for Critical Areas of Focus in Cloud Computing V2.1*. [Online]. Available: https://cloudsecurityalliance.org/csaguide.pdf

[125] I. Sadooghi *et al.*, "Understanding the performance and potential of cloud computing for scientific applications," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 358–371, Apr. 2017.

[126] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. 6th Conf. Symp. Operating Syst. Design Implement.*, vol. 6. New York, NY, USA, 2004, p. 10.

[127] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.

[128] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Operating Syst. Rev.*, vol. 37, no. 5, pp. 29–43, Dec. 2003, doi: 10.1145/1165389.945450.

[129] J. Ekanayake, S. Pallickara, and G. Fox, "MapReduce for data intensive scientific analyses," in *Proc. IEEE 4th Int. Conf. eScience*, Dec. 2008, pp. 277–284.

[130] M. Zaharia, *An Architecture for Fast and General Data Processing on Large Clusters* (Association for Computing Machinery). San Rafael, CA, USA: Morgan & Claypool, 2016.

[131] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "MapReduce in the clouds for science," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci.*, Nov. 2010, pp. 565–572.

[132] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing," in *Cloud Computing*, D. R. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Berlin, Germany: Springer, 2010, pp. 115–131.

[133] J. Dongarra and P. Luszczek, *HPC Challenge Benchmark*. Boston, MA, USA: Springer, 2011, pp. 844–850, doi: 10.1007/978-0-387-09766-4_156.

[134] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011.

[135] P. Mehrotra *et al.*, "Performance evaluation of Amazon EC2 for NASA HPC applications," in *Proc. 3rd Workshop Sci. Cloud Comput.*, New York, NY, USA, 2012, pp. 41–50, doi: 10.1145/2287036.2287045.

[136] D. H. Bailey *et al.*, "The NAS parallel benchmarks," *Int. J. Supercomput. Appl.*, vol. 5, no. 3, pp. 63–73, Sep. 1991, doi: 10.1177/109434209100500306.

[137] A. Petitet, R. Whaley, J. Dongarra, and A. Cleary. (Dec. 2018). *HPL—A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*. [Online]. Available: http://www.netlib.org/benchmark/hpl/

[138] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of Amazon EC2 data center," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[139] J. R. Lange *et al.*, "Minimal-overhead virtualization of a large scale supercomputer," in *Proc. 7th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environments*, New York, NY, USA, 2011, pp. 169–180. [Online]. Available: https://doi.org/10.1145/1952682.1952705

[140] P. Leitner and J. Cito, "Patterns in the chaos—A study of performance variation and predictability in public iaas clouds," *ACM Trans. Internet Technol.*, vol. 16, no. 3, p. 15, Apr. 2016, doi: 10.1145/2885497.

[141] A. Gupta *et al.*, "Evaluating and improving the performance and scheduling of HPC applications in cloud," *IEEE Trans. Cloud Comput.*, vol. 4, no. 3, pp. 307–321, Jul. 2016.

[142] S. Sehrish, G. Mackey, P. Shang, J. Wang, and J. Bent, "Supporting HPC analytics applications with access patterns using data restructuring and data-centric scheduling techniques in MapReduce," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 158–169, Jan. 2013.

[143] L. S. Blackford *et al.*, "An updated set of basic linear algebra subprograms (BLAS)," *ACM Trans. Math. Softw.*, vol. 28, no. 2, pp. 135–151, 2002.

[144] M. P. Forum, "MPI: A message-passing interface standard," Univ. Tennessee, Knoxville, TN, USA, Tech. Rep. CS-94-230, 1994.

[145] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014, *arXiv:1404.5997*. [Online]. Available: https://arxiv.org/abs/1404.5997

[146] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," 2018, *arXiv:1802.09941*. [Online]. Available: https://arxiv.org/abs/1802.09941

[147] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten, *Weka—A Machine Learning Workbench for Data Mining*. Berlin, Germany: Springer, 2005, pp. 1305–1314. [Online]. Available: http://researchcommons.waikato.ac.nz/handle/10289/1497

[148] T. Guo, "Cloud-based or on-device: An empirical study of mobile deep inference," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2018, pp. 184–190.

[149] A.-K. Koliopoulos, P. Yiapanis, F. Tekiner, G. Nenadic, and J. Keane, "A parallel distributed Weka framework for big data mining using spark," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2015, pp. 9–16.

[150] M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, "Big data machine learning using Apache spark MLlib," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 3492–3498.

[151] Y. You, I. Gitman, and B. Ginsburg, "Large batch training of convolutional networks," 2017, *arXiv:1708.03888*. [Online]. Available: http://arxiv.org/abs/1708.03888

[152] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–11.

[153] J. Dowling, "Distributed deep learning with Apache spark and tensorflow," in *Proc. Eur. Spark+AI Summit*, 2018, pp. 1–48.

[154] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[155] F. Niu, B. Recht, C. Ré, and S. Wright, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. NIPS*, vol. 24, Jun. 2011, pp. 1–22.

[156] J. J. Dai *et al.*, "BigDL: A distributed deep learning framework for big data," in *Proc. ACM Symp. Cloud Comput.*, 2019, pp. 50–60. [Online]. Available: https://arxiv.org/pdf/1804.05839.pdf

[157] M. Li *et al.*, "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Symp. Operating Syst. Design Implement.*, New York, NY, USA, 2014, pp. 583–598.

[158] N. Nawi, M. Ransing, and R. Ransing, "An improved learning algorithm based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method for back propagation neural networks," in *Proc. 6th Int. Conf. Intell. Syst. Design Appl.*, vol. 1, Oct. 2006, pp. 152–157.

[159] S. Moreno-Álvarez, J. M. Haut, M. E. Paoletti, J. A. Rico-Gallego, J. C. Díaz-Martín, and J. Plaza, "Training deep neural networks: A static load balancing approach," *J. Supercomput.*, vol. 76, no. 12, pp. 9739–9754, Dec. 2020, doi: 10.1007/s11227-020-03200-6.

ABOUT THE AUTHORS

Juan M. Haut (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering and the Ph.D. degree in information technology from the University of Extremadura, Cáceres, Spain, in 2011, 2014, and 2019, respectively, supported by the University Teacher Training Programme from the Spanish Ministry of Education.

He was a member of the Hyperspectral Computing Laboratory (HyperComp), Department of Technology of Computers and Communications, University of Extremadura. He is also an Associate Professor with the Department of Communication and Control Systems, National Distance Education University, Madrid, Spain. His research interests include remote sensing data processing and high-dimensional data analysis, applying machine (deep) learning, and cloud computing approaches. In these areas, he has authored/coauthored more than 30 JCR journal articles (more than 20 in IEEE journals) and 20 peer-reviewed conference proceeding papers. Some of his contributions have been recognized as hot-topic publications for their impact on the scientific community.

Dr. Haut was a recipient of the Outstanding Ph.D. Award from the University of Extremadura in 2019. He was a recipient of the Outstanding Paper Award at the 2019 IEEE WHISPERS Conference. From his experience as a reviewer, it is worth mentioning his active collaboration in more than ten scientific journals, such as the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, and IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, being awarded with the Best Reviewer recognition of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS in 2018. Furthermore, he has guest-edited three special issues on hyperspectral remote sensing for different journals. He is also an Associate Editor of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and IEEE JOURNAL ON MINIATURIZATION FOR AIR AND SPACE SYSTEMS.

Mercedes E. Paoletti (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 2014 and 2016, respectively, and the Ph.D. degree, supported by the University Teacher Training Programme from the Spanish Ministry of Education, from the University of Extremadura in 2020.

She was a member of the Hyperspectral Computing Laboratory (HyperComp), Department of Technology of Computers and Communications, University of Extremadura. She is currently a Researcher with the Department of Computer Architecture, University of Málaga, Málaga, Spain. Her research interests include remote sensing and analysis of very high spectral resolution with the current focus on deep learning (DL) and high-performance computing.

Dr. Paoletti was a recipient of the 2019 Outstanding Paper Award Recognition at the IEEE WHISPERS 2019 Conference and the Outstanding Ph.D. Award at the University of Extremadura in 2020. She has served as a Reviewer for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING and IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, in which she was recognized as a best reviewer in 2019.

Sergio Moreno-Álvarez received the B.Sc. and M.Sc. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 2017 and 2019, respectively, where he is currently working toward the Ph.D. degree at the Department of Computer Systems Engineering and Telematics.

As research experience, he has participated in regional projects. He is currently a Researcher with the School of Technology, University of Extremadura. He has published four JCR articles in international journals and three presentations at international and national conferences. His main interests are high-performance computing, neural networks, and deep learning (DL).

Javier Plaza (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 2004 and 2008, respectively.

He is currently a member of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. He has authored more than 200 publications, including over 80 JCR journal articles, ten book chapters, and 100 peer-reviewed conference proceeding papers. His main research interests comprise hyperspectral data processing and parallel computing of remote sensing data.

Dr. Plaza was a recipient of the Outstanding Ph.D. Dissertation Award at the University of Extremadura in 2008. He was also a recipient of the Best Column Award of the *IEEE Signal Processing Magazine* in 2015 and the Most Highly Cited Paper (2005–2010) in the *Journal of Parallel and Distributed Computing*. He received the best paper awards at the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology. He has guest-edited four special issues on hyperspectral remote sensing for different journals. He is also an Associate Editor for IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and IEEE Remote Sensing Code Library. Additional information: http://www.umbc.edu/rssipl/people/jplaza.

Juan-Antonio Rico-Gallego received the Computer Science Engineering degree and the Ph.D. degree in computer science from the University of Extremadura, Cáceres, Spain, in 2002 and 2016, respectively.

He was a software consultant. He is currently an Associate Professor with the Department of Computer Systems Engineering, University of Extremadura. His research interests are in analytical performance models on heterogeneous platforms and the usage of deep and reinforcement learning techniques to high-performance computing (HPC) platform problems, including scheduling, process deployment and mapping, load balancing, and communication modeling. He codevelops AzequiaMPI, an efficient thread-based full MPI 1.3 standard implementation. His other research interests include current message passing interface (MPI) implementations and applications.

Antonio Plaza (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 1999 and 2002, respectively.

He is currently the Head of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. He has authored more than 600 publications, including over 300 JCR journal articles (over 220 in IEEE journals), 23 book chapters, and around 300 peer-reviewed conference proceeding papers. His main research interests comprise hyperspectral data processing and parallel computing of remote sensing data.

Prof. Plaza was a member of the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS). He is also a Fellow of IEEE "for contributions to hyperspectral data processing and parallel computing of Earth observation data." He was a recipient of the Best Column Award of the *IEEE Signal Processing Magazine* in 2015, the 2013 Best Paper Award of the JSTARS, and the Most Highly Cited Paper (2005–2010) in the *Journal of Parallel and Distributed Computing*. He received best paper awards at the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology. He was a recipient of the recognition of Best Reviewers of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS in 2009 and the recognition of Best Reviewers of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING in 2010, for which he served as Associate Editor from 2007 to 2012. He has guest-edited ten special issues on hyperspectral remote sensing for different journals. He is also an Associate Editor for IEEE ACCESS (receiving recognition as an Outstanding Associate Editor of the journal in 2017). He has served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) from 2011 to 2012 and the President of the Spanish Chapter of the IEEE GRSS from 2012 to 2016. He has reviewed more than 500 manuscripts for over 50 different journals. He has served as the Editor-in-Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING from 2013 to 2017. He is also the Editor-in-Chief of the IEEE JOURNAL ON MINIATURIZATION FOR AIR AND SPACE SYSTEMS (J-MASS). He has been distinguished as a Highly Cited Researcher by Clarivate Analytics from 2018 to 2020. Additional information: http://www.umbc.edu/rssipl/people/aplaza.