Analysis of Remotely Sensed Images Through Social Media

Alejandro Redondo, Juan M. Haut, *Senior Member, IEEE*, Mercedes E. Paoletti, *Senior Member, IEEE*, Xuanwen Tao, *Student Member, IEEE*, Javier Plaza, *Senior Member, IEEE*, and Antonio Plaza, *Fellow, IEEE*

***Abstract*—During the past decades, the volume of big data available in remote sensing (RS) applications has grown significantly. In addition, a number of applications related to monitoring human activity are being developed based on this kind of data. This has considerably increased the demand for RS processing methods. In this sense, the scientific community is facing the challenge of how to maximize the potential of the data that are produced in a fast and efficient way. In particular, the provision of processing algorithms that can be developed in an easy way is a fundamental problem for the RS community, due to the large volume of data offered by different portals and agencies, and the need for algorithms developed on different platforms and using a variety of programming frameworks. To address these challenges, this article takes advantage of social media tools to bring images and algorithms closer to users. In particular, a new system based on the *Telegram* messaging application *Bot* (called @*ThuleRSbot*) has been developed to provide a wide variety of RS image processing methods to users under the same interface. The system has been developed using the *Python* language and the *Telegram Bot* application program interface (API). The most remarkable characteristics of the system are: first, a completely open architecture that facilitates the incorporation of new algorithms without effort; and second, an easy way to automatically gather satellite images from the Sentinel Hub platform.**

***Index Terms*—Big data, image processing, remote sensing (RS), social media tools.**

I. Introduction

IN RECENT years, many developments have been conducted in earth observation (EO) and remote sensing (RS) fields [1]–[3]. Technological advances have enabled the deployment of powerful sensing instruments on both airborne and spaceborne platforms [4]–[8], which are employed in a large number of new EO missions to capture rich information from the surface of the earth. This information is normally arranged as a set of RS scenes or images that provide details about different properties of the observed materials along the electromagnetic spectrum, depending on the sensor characteristics. As a result, a wide variety of RS data is now being collected by multiple EO instruments. For instance, active sensors are able to collect synthetic-aperture radar (SAR) [9] and light detection and ranging (LiDAR) [10] data in the microwave and visible-near infrared spectral ranges, respectively. Passive sensors measure the solar radiation that is reflected, absorbed, and transmitted by the elements located in the observed area, usually, gathering the spectral information in the *visible-to-shortwave infrared* (VSWIR) region. Thus, depending on the number of spectral bands, several types of RS scenes can be identified, such as panchromatic (PAN) [11], [12], standard RGB [13], [13], multispectral (MSI) [14], [15], and hyperspectral (HSI) [16], [17] images. Fig. 1 shows different types of RS images according to the number of bands, while Table I provides a summary of the main characteristics of these images. Also, depending on the technical characteristics of the instrument, the spectral bandwidth, and the revisit time, these images exhibit different spatial and temporal resolutions [18]–[20].

TABLE I
Summary of Different Types of RS Images

	RGB	**Multispectral**	**Hyperspectral**
Spatial resolution	Very High	High	Low
Spectral resolution	Low	Moderate	Very High
Number of bands	3	$3 \sim 10$	>100
Image size	Low	High	Very High

The wide variety of RS data, coupled with the almost constant data acquisition flow [21]–[23] and the implementation of new and large data repositories and libraries [17], [24]–[26], have provided a powerful tool for the development of a wide variety of human activities [27], such as precision agriculture [28], crop monitoring [29], [30], and collection of vegetation indices [31], [32], natural resources exploitation such as forestry [33], [34], mineralogy [35], and water resources [36], [37], along with environmental and land degradation modeling [38]–[40], pest and invasive species control [41]–[43], and risk prevention [44]–[46], among many other activities.

However, in order to take full advantage of RS data, these images should be processed in an efficient and effective way, with the aim of providing a final data representation that satisfies the final user/application. In this context, the demand for RS image

Manuscript received December 17, 2020; revised February 5, 2021; accepted February 21, 2021. Date of publication February 26, 2021; date of current version March 19, 2021. This work was supported in part by the Junta de Extremadura, Ref. GR18060, in part by the European Union's Horizon 2020 Research and Innovation Program under Grant 734541 (EOXPOSURE), and in part by the MINECO Project PID2019-110315RB-100 (APRISA). *(Corresponding author: Juan M. Haut.)*

Alejandro Redondo, Mercedes E. Paoletti, Xuanwen Tao, 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: aredondosy@alumnos.unex.es; mpaoletti@unex.es; taoxuanwenupc@gmail.com; jplaza@unex.es; aplaza@unex.es).

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).

Digital Object Identifier 10.1109/JSTARS.2021.3062116

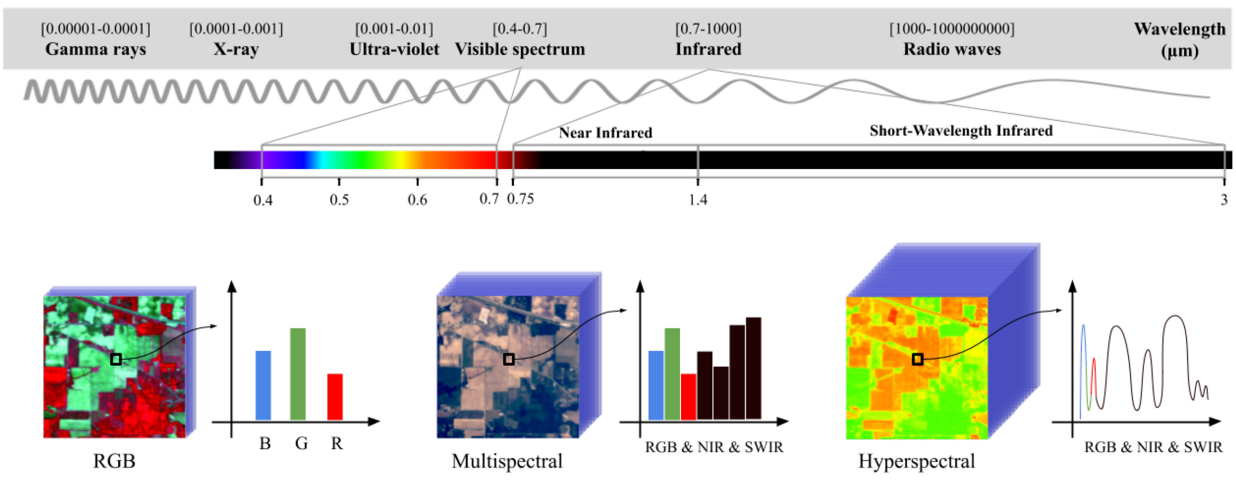


Fig. 1. Different types of RS images according to the number of spectral bands.

processing methods has increased over the past decades [47]–[52], while the scientific community has made great efforts to develop a large number of methodologies to address very specific problems, such as spatial/spectral resolution enhancement [53]–[57], data restoration/denoising [58]–[61], data compressing, band selection and dimensionality reduction [62]–[68], image segmentation [69]–[72], spectral unmixing [73]–[77], target, object, change, and anomaly detection [20], [78]–[84], and data classification [85]–[90]. The original codes of many of these methods are available in different repositories. In addition, there are many programming frameworks that significantly facilitate the development of new processing methods.

As it has been pointed out previously, both RS data and processing codes are generally scattered in different repositories, with very diverse formats and adopting different programming frameworks. This may hinder the exploitation and maximization of the RS data processing potential, leading to the adoption of a small set of RS scenes for validation purposes and the use of benchmark algorithms, which may not always be appropriate.

To overcome these limitations, this article presents a new RS image processing tool, called *ThuleRSBoot*, that offers a unified processing environment that can accommodate different types of RS datasets and processing methods in a simple way. In this sense, the application takes advantage of the great success of social media technologies to make available a complete testbed of images and processing algorithms. In addition, it provides a completely new approach to easily download new RS scenes, allowing also for the incorporation of newly developed algorithms in a simple and intuitive way by following a "plug & play" approach. Finally, the developed application interprets software and datasets in different languages and formats. The main goal is to break down the format-related barriers that have hindered the wide exploitation of RS data within the scientific community.

A. *Rise of Social Networks*

The popularity of the Internet sparked the concept of social networks as a powerful information sharing system based on establishing relationships between people that create and consume content regardless of their location [91]. Such interactions have attracted a significant number of users, which is increasing steadily, and resulted in the appearance of many different types of social networks [92], [93]. In fact, social networking is playing a fundamental role in the modern society, even modifying the way most daily human activities are conducted by leveraging the conducted relationships through the collection of user profiles that join and establish social links with each other [94]; for instance, information sharing and specific shared resources search, events organization, job searching, or commercial advertising, among many others [95]. As a result, the number of users registered in social networks increases exponentially every day. Furthermore, with the widespread use of social networks and the evolution of the Internet, multimedia has become one of the main vehicles for communication, improving the attention, and understanding of the involved contents [96]. In particular, digital images have a high visibility in social networks and websites, due to their great potential to represent information in a graphic and visual way [97]. As a result, every day thousands of millions of images are shared through social networks, either to report events or just for entertainment purposes.

At the same time, and also supported by the impressive growth of Internet technologies and social networks, Bot-type functionalities have emerged as an interesting tool to avoid repetitive and tedious tasks, such as reviewing the content of social media, monitoring the performance of a website, grammar correction, or speech recognition, among others [98], [99]. In this context, the difficulties faced by social networks fostered the development of many social applications such as Telegram. It is well known that Telegram [100]–[102] is an interesting social application that allows bots to be developed in a simple way, and they can also be implemented in the application itself without installing any plug-ins. These bots stand out for their flexibility and versatility of use, which can be defined by each user/developer, allowing to conduct more complex tasks such as the automatic processing of digital images.

In this regard, the RS community can seize these potentialities to promote the study of an event located in a certain region, or to substantiate the impact that a certain phenomenon has caused over time [103], [104]. As a result, diverse techniques have been developed to combine human activities on social networks with RS images in order to obtain more complete spatial-temporal characterization, monitoring the impact of events, or observing the influence of different human behaviors over a geographic area. For instance, Jendryke *et al.* [105] combined RS scenes with online social media messages (paying particular attention to their geographical coordinates and timestamps) to analyze urban dynamics, such as the variations between built-up areas and human activity. Moreover, new techniques attempt to combine RS optical scenes (such as PAN, MSI, and HSI images) with geotagged images shared on social networks (usually RGB images) to provide greater details to the information acquired about a target area [106], [107]. In this sense, social networks serve not only as a fast and efficient mechanism for sharing large amounts of information, but also as a complementary resource that can be combined with existing image data to provide a richer and more complex interpretation [103], [104], [108].

B. Machine Learning to Process RS Images

The images collected by EO instruments need to be processed and interpreted to obtain the most discriminative features, translating them into useful information for the final users. Usually, a branch of artificial intelligence, known as *machine learning* (ML) [109], is used to process this kind of images. In fact, ML provides a wide variety of processing algorithms that automatically learn from the data and extract the most useful information [86]. In general, these algorithms comprise several stages.

1) *Model creation:* Any ML method begins by defining its purpose and characteristics. In this sense, the objective of the model must be in line to successfully conduct the given task, such as classification, grouping, segmentation, optimization, etc.
2) During the *data preparation*, the ML algorithm normalizes and formats the data so that the relevant features can be extracted.
3) *Model adjustment:* Once the data are ready to be used, the ML model is trained and its parameters are adjusted to fit the data. Accordingly, the ML algorithm learns how to extract the patterns and internal regularities of the data.
4) During the *inference* step, new (or unseen) data are presented to the ML algorithm, in order to obtain the desired outputs and evaluate the performance of the model.
5) Finally, the *evaluation of the model* is conducted by checking its performance when unseen samples are processed, measuring its ability to generalize the information learned during the training and adjustment phase.

In this work, depending on the knowledge about the data employed during the third stage, we consider three main kinds of algorithms.

1) *Unsupervised learning* [110], [111] in which the classes are not known in advance; namely: only the characteristics of the data are known, but not the existing classes, and therefore, it is the algorithm itself that has to recognize the patterns and group them according to their characteristics. This means that the algorithm creates the classes.
2) *Supervised learning* [112], [113] trained with tagged data, that is, the class to which the data belongs is known *a priori*. The algorithm establishes a function between the input data and the output data during the training phase, while in the prediction phase an unlabeled piece of data will be labeled. In this type of learning, the characteristics of each element are known. In an RS image, the inputs are the spectral bands and the output is the class to which each pixel belongs, also known as the ground-truth.
3) *Semisupervised learning* [114], [115] that consists of a combination of the two aforementioned approaches, that is, classified and unsorted data are available. Generally, in this type of algorithm, there are more unclassified data than classified data.

Although other model types and taxonomies can be found [116], [117], this article takes into account those that are most relevant within the field of RS image processing. One of the main issues that our newly developed *ThundeRSbot* intends to solve is the diversity of processing methods and their implementations. In fact, specialized users within the RS field often need to conduct a vast and expensive search within the available literature, comparing several algorithms and their obtained results over different RS datasets, in order to select those methods that best suit their scientific purpose. This initial search often requires significant time and resources. Moreover, the applications that are currently available in the market for the processing of RS data, such as ENVI [118] or LEOworks [119], may not contain all the algorithms that the user requires, while the execution of the processing methods is usually conducted over the final user resources, which may not be too powerful (computationally speaking), resulting in high response times.

C. Contribution of This Article: Bringing ML Closer to RS Users Through Social Tools

To engage RS users and provide them with the tools required to solve the aforementioned problems, our work introduces a new system called @*ThuleRSbot*, which exploits social media tools for advanced RS data processing. The main goal of our proposed system is to provide RS users with a simple, intuitive, and accessible tool. To fulfill this objective, a new *Telegram Bot*[1] has been developed, in which the execution of algorithms is conducted externally by a server, regardless of the resources which the user has available. The major contributions of our work are twofold.

1) Our newly proposed system provides an easy way to incorporate new algorithms and operations in a simple plug-and-play mode, acting as a container for new algorithms and datasets.
2) Our system allows downloading new satellite images from a consolidated platform, such as the Sentinel-Hub, that can be then easily processed by the algorithms included in the tool (or those incorporated by the user).

[1]Code can be downloaded and installed from https://github.com/mhaut/ThuleRSbot

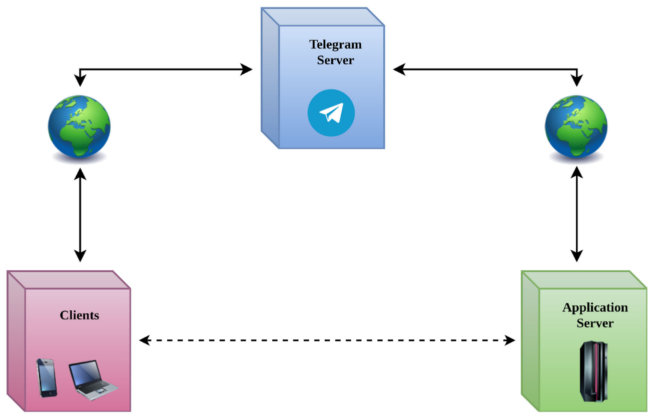


Fig. 2. Architecture of the system, which comprises three main agents: the client, the Telegram server, and the compute server. These agents exchange messages between them through the internet.

These features conform a completely innovative application that can be run in smartphones, to quickly process vast amounts of RS data in a computationally efficient and effective way, that is also available to many users due to its app format.

The rest of the article is structured as follows. Section II describes the development of the app and defines the user requirements. Section III-B describes the experiments carried out in order to illustrate the performance and scalability of the application. Section IV concludes this article with some remarks and hints at plausible future research lines.

II. Methodology

This section provides the design of the proposed system, which has been developed and implemented under the principles of high-quality software development [120].

A. System Architecture

This section delves into the system architecture details, which has been designed to be completely scalable. In other words, the proposed @*ThuleRSbot* takes advantage of all existing resources on the compute server, and includes new operations and algorithms in a way that is completely transparent to the user. In this sense, the architecture of @*ThuleRSbot* comprises three main blocks, as shown in Fig. 2.

1) The *client* should install the *Telegram* application in the device and uses the *Bot* called @*ThuleRSbot*. This agent initiates communication with the *Bot*, which has previously been executed on the computing server device.
2) The *Telegram* server is located in the middle of the communication, and acts as a *middleware* [121]. Its main functionality is to manage communications and provides security between the two final agents: the client and the computing server.
3) The *server* is responsible for performing all the tasks related to RS image processing. It should be noted that this server is also responsible for receiving and sending messages from/to the *Telegram* server. The server comprises two main parts: a server which is specifically devoted to the communication with the *Telegram* server, and another server that takes care of computing purposes. However, we have integrated both of them in order to reduce latency.

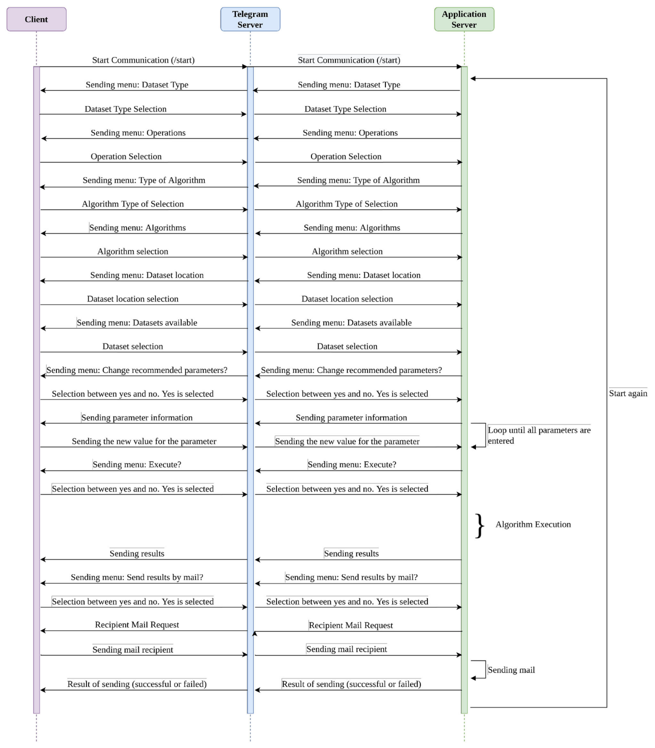

Client
Telegram Server
Application Server
Start Communication (/start)
Start Communication (/start)
Sending menu: Dataset Type
Sending menu: Dataset Type
Dataset Type Selection
Dataset Type Selection
Sending menu: Operations
Sending menu: Operations
Operation Selection
Operation Selection
Sending menu: Type of Algorithm
Sending menu: Type of Algorithm
Algorithm Type of Selection
Algorithm Type of Selection
Sending menu: Algorithms
Sending menu: Algorithms
Algorithm selection
Algorithm selection
Sending menu: Dataset location
Sending menu: Dataset location
Dataset location selection
Dataset location selection
Sending menu: Datasets available
Sending menu: Datasets available
Dataset selection
Dataset selection
Sending menu: Change recommended parameters?
Sending menu: Change recommended parameters?
Selection between yes and no. Yes is selected
Selection between yes and no. Yes is selected
Start again
Sending parameter information
Sending parameter information
Loop until all parameters are entered
Sending the new value for the parameter
Sending the new value for the parameter
Sending menu: Execute?
Sending menu: Execute?
Selection between yes and no. Yes is selected
Selection between yes and no. Yes is selected
Algorithm Execution
Sending results
Sending results
Sending menu: Send results by mail?
Sending menu: Send results by mail?
Selection between yes and no. Yes is selected
Selection between yes and no. Yes is selected
Recipient Mail Request
Recipient Mail Request
Sending mail recipient
Sending mail recipient
Sending mail
Result of sending (successful or failed)
Result of sending (successful or failed)

Fig. 3. Sequential diagram with the messages exchanged between the agents of the architecture.

In the following, we briefly comment on some of the main features of the model architecture given in Fig. 2.

1) Our architecture allows to modify and/or change the server in a transparent way to the user, tolerates heterogeneity of computing resources, reduces the computation of the processing server, supports scalability, and provides important services (e.g., security services). The distribution of the end devices is transparent to the user, and the system balances the workload of the compute server as it supports multiple users.
2) Internet connection is required, although the availability of connection is very flexible, that is: the server always saves the status of each user, so if the connection is interrupted before the user executes a task, the user resumes the process from the last checkpoint. When the same user reconnects, the user receives the results by means of the middleware. In this regard, failures in the *Telegram* server (as compared to other messaging applications, such as *Whatsapp*) are significantly less frequent[2]

As shown in Fig. 2, the system architecture is based on message passing through the communication network. The flow of message exchange is represented graphically in Fig. 3, and summarized below.

1) The user initiates the communication by using the /start command in the initial *Bot* screen, and the *Bot* sends a first menu known as the dataset types menu, which provides

[2][Online]. Available: https://downdetector.es/

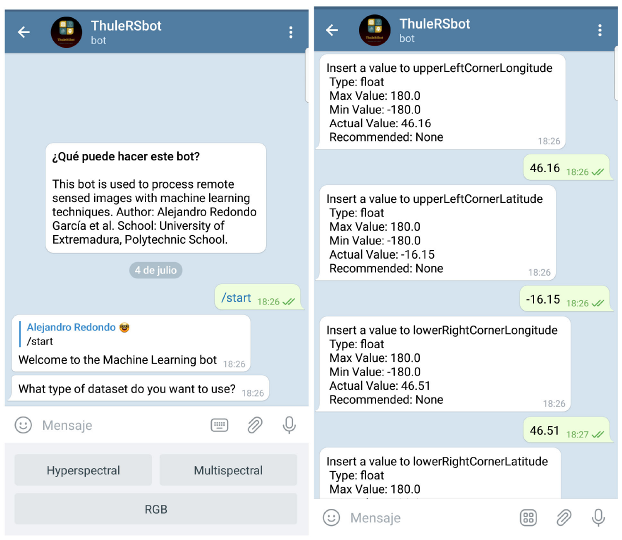
three different types of datasets that can be processed: HSI, MSI, and RGB. The user should select an option from the menu of available datasets and choose the type of data on which he/she wants to perform an analysis.

2) Depending on the data type, the bot sends a list with the available tasks that are conducted on that data. This is known as the operations menu, which contains a maximum of five possible operations, i.e., the bot filters the menu according to the response that the user gives. The operations are: Classification, Dimensional Reduction, Restoration, Unmixing, and Change Detection.
3) The user selects an operation (from those available in the menu) and the *Bot* sends the list of available algorithms, displaying the different options as the user selects the operations, which can be supervised, semisupervised, and unsupervised.
4) The user chooses a type of algorithm and the *Bot* sends the menu with the algorithms available for that operation and type of algorithm.
5) The user chooses an algorithm from those that are available, and the *Bot* sends the menu for the user to choose the location of the dataset. The location of the dataset can be: public, private, uploading a dataset, default, or obtaining a dataset using Sentinel Hub (only available if MSI dataset type is selected).
6) The user chooses the final dataset and the *Bot* asks the user if the user wants to modify the algorithm's recommended parameters.
7) At this point, the user chooses (yes or no). If the answer is no, the user continues in the next step and otherwise the *Bot* will ask the values of the parameters.
8) The *Bot* sends a summary of the subsequent execution and asks if the user would like to execute the summary. If this is not the case, we go back to the first step and, otherwise, the algorithm is executed on the computing server (no computational resources of the user are consumed).
9) When the server has finished executing the algorithm, it sends the results back to the user and asks if the user would like to receive those results also by e-mail.
10) If the user decides not to receive the results by e-mail, the system returns to step one. Otherwise, the user is asked about his/her email and the results are sent. When the email is delivered to the user, the *Bot* acknowledges the sending of the email.
11) At the end of the process, we return to step one.

The user stops the *Bot* using the /stop command. Also, he/she requests help from the application by executing the /help command. In all the cases, the client has at his/her disposal the available options through a complete menu, where the user selects the desired option. Fig. 4(a) provides a snapshot of one of the available menus. Moreover, the application itself provides the necessary information to the user in order to enter the parameters requested by the bot, as it can be observed in Fig. 4(b).

B. *Plug-and-Play*

One of the main characteristics of the newly developed application is its flexible architecture, which provides an easy way for the introduction of new algorithms for the processing of RS images without effort. To accomplish this goal, a *Plug-And-Play* device line has been adopted. The main inspiration lies in hardware devices such as the *Pen Drive*, whose goal is "plug, connect, and use," that is, they are connected to a computer, being used directly without the user worrying about their configuration. In this sense, a similar strategy has been developed to enable the easy integration of new RS data processing algorithms.



¿Qué puede hacer este bot?

This bot is used to process remote sensed images with machine learning techniques. Author: Alejandro Redondo García et al. School: University of Extremadura, Polytechnic School.

/start 18:26

Alejandro Redondo
/start
Welcome to the Machine Learning bot 18:26

What type of dataset do you want to use? 18:26

Insert a value to upperLeftCornerLongitude
Type: float
Max Value: 180.0
Min Value: -180.0
Actual Value: 46.16
Recommended: None 18:26

46.16 18:26

Insert a value to upperLeftCornerLatitude
Type: float
Max Value: 180.0
Min Value: -180.0
Actual Value: -16.15
Recommended: None 18:26

-16.15 18:26

Insert a value to lowerRightCornerLongitude
Type: float
Max Value: 180.0
Min Value: -180.0
Actual Value: 46.51
Recommended: None 18:26

46.51 18:27

Insert a value to lowerRightCornerLatitude
Type: float
Max Value: 180.0

(a) (b)

Fig. 4. Two possible options for entering the data requested by the client in our tool.

As a first step, the user should create a new file with the extension *. py*. The file can be named with the processing algorithm name. This file will inherit from the *Base* class of the operation to which the algorithm belongs. The content of this file consists of a fixed part and an optional one. The fixed part is a class whose name is the same as the one established in the *. py* file. This class contains a constructor, which creates the algorithm instance by inserting the default parameters (since the user has an option to change the most important parameters) and then the previously created algorithm instance (the model), the types of devices and algorithms are introduced. The optional part is also a class, within the same file. It contains the implementation of the algorithm. It is optional as the algorithm is implemented in some *python* library. However, all the algorithms that are introduced have to follow the same structure and format of *scikit-learn* library algorithms, such as the following.

1) Classification algorithms must contain the following parts: *fit*, to train the model; *predict*, to perform the inference stage; and *score,* to get the final performance.
2) Dimensionality Reduction, Restoration, and Unmixing algorithms should have also three methods: *fit* (to train), *transform* (to perform the inference), and *score* (to obtain performance).
3) Change detection must have a method called *execution*, where the change detection operation is implemented.

In addition to these methods, the tool allows the insertion of new methods, such as target and anomaly detection. Finally, this file will be inserted in the corresponding folder, according to its operation and the type of algorithm.

The second step is to include the name of the algorithm file in the file *ParametersDefautl.yaml* with the most important parameters that the user can modify. If there are no parameters, only the name of the algorithm will be inserted.

In order to simplify the proposed application, a template called *templateAlgorithm.py* is provided within the bot code, which can be found at the link provided in the abstract. Moreover, this link provides all the code that is needed to include new processing methods. A simple step-by-step example can be found in the *KNN.py* file.

In addition to new algorithms, the @*ThuleRSbot* tool provides an easy way to include new RS images, such as optical data (RGB, MSI, HSI, and PAN scenes), DEM, SAR, and LiDaR data within the *datasets* folder. The user has the possibility to include his/her own data to be processed with the available algorithms, taking into account that different types of files are distinguished according to their content and the operation to be performed, such as the following.

1) For *classification*, *dimensionality reduction,* and *data restoration* operations, the supported formats are . *mat* and . *tif*. Moreover, there are three file types: First, the *data file*, also called *dataset*, which contains the remotely sensed image information; Second, the *label file*, this file is only needed when supervised classification algorithms are executed; Third, the *RGB file*, which contains an RGB representation of the RS image for display purposes. If this file is not available, three bands are obtained from the *data file* and the RGB representation is displayed in false color.
2) For *unmixing* operations, the supported format are also . *mat* and . *tif*. Again, there are three file types: First, the *data file*, containing the RS image information; Second, the *file of spectral signatures* of endmembers, which is required only by the supervised unmixing algorithms, and Third, the *RGB file*.

Moreover, the directories can be easily modified to adapt the server to the needs of the researchers.

C. Security Between Clients and Server

Security is a very important feature of our application, particularly for the passing of messages between the two final agents. The protocol that *Telegram* implements for sending messages is *MTProto* [122] (*Mobile Transport Protocol*). This protocol is focused on cross-platform multisession and file transport, regardless of format or capacity. It is based on the TCP/IP stack, belonging to the application level. The encryption adopted is 256-bit [123] *Advanced Encryption Standard (AES*) along with 2048-bit [123] *Rivest, Shamir, and Adleman* (*RSA*). *AES* is a symmetric encryption algorithm, which works as follows: both the sender and the receiver use the same key to encrypt and decrypt the message; therefore, it is necessary that both agree on the key at the beginning of the communication, while *RSA* is a public key cryptographic algorithm or asymmetric encryption that works as follows: the sender encrypts with the public key of the receiver and the receiver decrypts with its private key. The complexity of this algorithm lies in the factorization of integers. In order for both to have the same key in a secure way, the *Diffie-Hellman* [124] key establishment protocol is used; its security lies in the extreme difficulty of calculating discrete logarithms in a finite body. Finally, the message is signed using the [125] *Secure Hash Algorithm* (SHA) algorithm of 256 bits.

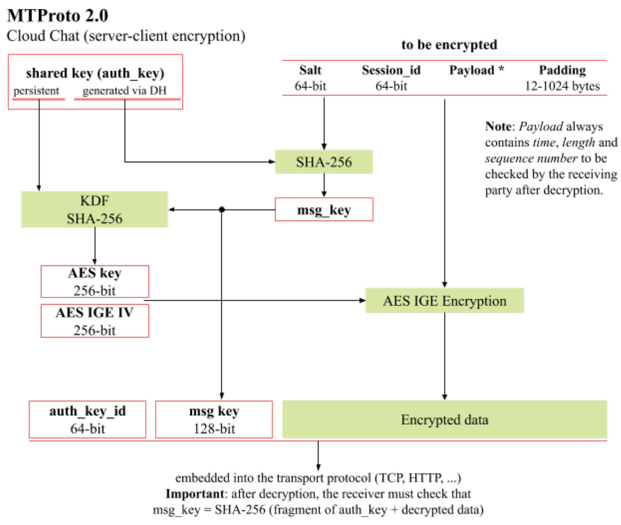

MTProto 2.0
Cloud Chat (server-client encryption)
to be encrypted
shared key (auth_key)
persistent
generated via DH
Salt 64-bit
Session_id 64-bit
Payload *
Padding 12-1024 bytes
Note: Payload always contains time, length and sequence number to be checked by the receiving party after decryption.
SHA-256
KDF SHA-256
msg_key
AES key 256-bit
AES IGE IV 256-bit
AES IGE Encryption
auth_key_id 64-bit
msg key 128-bit
Encrypted data
embedded into the transport protocol (TCP, HTTP, ...)
Important: after decryption, the receiver must check that msg_key = SHA-256 (fragment of auth_key + decrypted data)

Fig. 5. Normal chat operation of the *Telegram* application.

On the other hand, *Telegram* provides two types of chats.

1) Normal chat, called*cloud chat*, which uses client-server-client encryption.
2) Secret Chat, called*Secret chat*, encrypted in end-to-end manner.

In our system, the normal chat type was used for the communication with the Bot. The formation of a message in this type of chat is shown in Fig. 5.

D. Privacy Between Clients and Server

Apart from security, privacy is also very important in our context. In the developed application, the data provided by the user is differentiated from the sample data provided in the application examples. As for the sample datasets, they are available to all users. However, the datasets generated by a user are private by default, which means that only the owner can work with them and the rest of the users cannot access them (unless explicitly specified by the owner). On the other hand, the server saves all the datasets (public and private), and therefore, it can access and/or modify them. Other than this, the server is also in charge of recording all the operations that each user performs individually and also all the result files sent by mail (if the user decides to send them).

E. Multiple User Support

The application is capable of supporting multiple simultaneous users in order to make the most of the resources of the computing server. The mechanism used to allow this feature is the one described below.

Within the configuration file, there is a parameter called “num_Threads”; this parameter controls the *WorkerThread* threads that are created at the beginning of the *Bot* execution. These threads are responsible for solving the queries made by users, and therefore, they are not assigned a specific task within the application, or a specific active user. In the application, the *WorkerThread* threads primarily handle three tasks: messages received from users, execution of intermediate steps before the final algorithm execution, and final algorithm execution. To avoid the need to create a thread for each active user, the following mechanisms have been adopted.

1) *Callback* functions have been implemented as far as possible, avoiding active waiting for user responses.
2) For the active waiting stretches, a*timeout* has been modified and the number of users in these waiting periods has been restricted to the number of threads*WorkerThread* minus 1, to reserve a thread to execute the*listener* and, thus, avoid server crashes and deadlocks.
3) Finally, to reduce the number of threads, a *listener* has been implemented that is capable of handling all the messages produced at the same time by different users. This *listener* is capable of introducing the content of these messages into the global variable that registers the state of all the active users in an atomic way, avoiding that it is permanently inconsistent.

The aforementioned characteristics provide the system with the following advantages.

1) The application is capable of performing the processing tasks in the server according to its processing capabilities, that is, it just allocates the number of simultaneous executions that the server supports in the parameter “num_Threads.” When the number of executions exceeds the number of threads, executions are performed on a first-come, first-served basis.
2) The application supports more users with fewer threads. An important clarification in this regard is that the application requires at least two threads: One is for listeners and the other is for serving users.
3) Collapse and system crashes are avoided mainly by reserving a thread to act as an application *listener*.

F. Multiple Workflow Support

The general configuration of the bot is summarized in the file config.json, where we can find the Telegram, Email, and Sentinel-Hub main configuration. An important additional parameter is “parallelOrQueueExecutionAlg” which represents the execution mode of the bot. There are two modes of execution: “parallel” and “queue.” In the following, both modes are explained in detail.

The “parallel” mode allows the simultaneous operation of N executions on the available hardware resources, where N represents the number of threads (set in the Telegram configuration parameter called “num_Threads”). The concept is illustrated in Fig. 6. The “queue” mode creates an execution priority queue in which the user’s execution requests are queued up. A priority between 1.0 and 0.0 is assigned to each user, where 1.0 is the

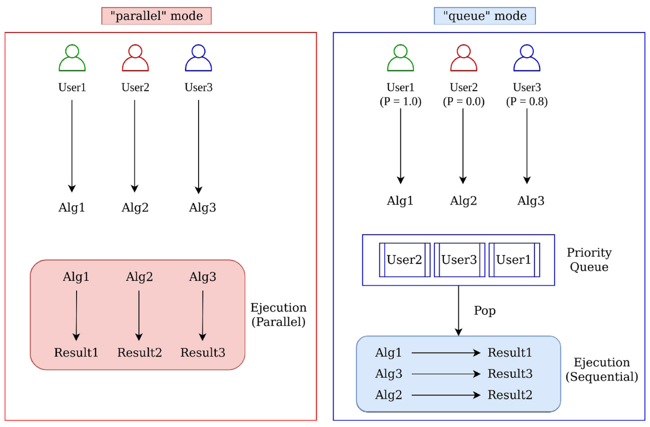


Fig. 6. Graphical representation of the two execution modes available in the bot.

maximum priority and 0.0 is the minimum. At first, an user that has not launched any execution will have a priority of 1.0, which will be reduced by 0.1 each time the user requires a new execution, until reaching the minimum level of priority. This strategy has been implemented in order to prevent the system from being monopolized by a single user, thus, consuming all hardware resources. In this sense, the priority queue attends the requests one by one, sending the obtained results to the corresponding user. The maximum number of requests is N, which represents the maximum number of possible threads (set in the Telegram configuration parameter called “num_Threads”). The user’s threads are waiting until their execution requests are processed. Fig. 6 provides a graphical overview of this process.

G. Error Control

The lack of experience with the application (and even with the data and/or algorithms that are implemented) may cause the occurrence of internal errors that, if not controlled and reported to the users in a clear way, may cause frustration and subsequent abandonment of the application. This is the main reason why the application should be highly fault-tolerant. Throughout all the interactions with the user, many mechanisms have been established to control errors and allow the continued integration of the application, maintaining all its functionalities active. The following are some of the errors that are actually controlled by our application.

1) Errors related to the response provided by the user have been limited through the use of the keyboard offered by the bot. If the user enters a nonavailable option, the application will inform the user of the error and, later, of the possibility to choose one out of several possible options.
2) Regarding the modification of the default parameters, the user is guided to enter suitable parameters in terms of type (i.e., integer, string, etc.) and ranges (in the case of numeric parameters).
3) In the execution step, the user may introduce internal errors related to the data and parameters previously entered when executing the algorithm, for example, when the internal

TABLE II
REPRESENTATION OF THE STRUCTURE OF THE EOPATH OBTAINED WHEN THE DATA ARE REQUESTED FROM THE SENTINEL HUB SERVER, WHERE *T* IS THE TEMPORAL COMPONENT, *N* AND *M* ARE THE SPATIAL COMPONENTS (HEIGHT AND WIDTH), AND *D* IS THE NUMBER OF BANDS

FeatureType	Type of data	Time component	Spatial component	Type of values	Python object	Shape
DATA	raster	yes	yes	float	numpy.ndarray	t x n x m x d
MASK	raster	yes	yes	integer	numpy.ndarray	t x n x m x d
SCALAR	raster	yes	no	float	numpy.ndarray	t x d
LABEL	raster	yes	no	integer	numpy.ndarray	t x d
DATA_TIMELESS	raster	no	yes	float	numpy.ndarray	n x m x d
MASK_TIMELESS	raster	no	yes	integer	numpy.ndarray	n x m x d
SCALAR_TIMELESS	raster	no	no	float	numpy.ndarray	d
LABEL_TIMELESS	raster	no	no	integer	numpy.ndarray	d
VECTOR	vector	yes	yes	/	geopandas.GeoDataFrame	Required columns geometry and TIMESTAMP
VECTOR_TIMELESS	vector	no	yes	/	geopandas.GeoDataFrame	Required column geometry
META_INFO	anything	no	no	anything	anything	anything
TIMESTAMP	timestamps	yes	no	datetime	list(datetime.datetime)	t
BBOX	bounding box and CRS	no	yes	coordinates	sentinelhub.BBox	/

Source: https://github.com/sentinel-hub/eo-learn-workshop/blob/master/notebooks/eolearn_basics.ipynb

format of the data is not correct. In these cases, if the application detects such type of errors, it sends an alert to the user informing that an algorithm execution failure has occurred. At this point, the application returns to the initial status.

H. Algorithms

The algorithms that have been implemented in the application are the following ones (they are categorized according to their aim).

1) *Classification:* We considered both supervised and unsupervised techniques. The supervised algorithms implemented are*K*-nearest neighbors (KNN) [126], linear discriminant analysis (LDA) [127], multinomial logistic regression (MLR) [128], random forest (RF) [129], support vector machines (SVM) [130], multilayer perceptron (MLP) [131], convolutional neural network 1D (CNN1) [131], and convolutional neural network 2D (CNN2) [131]. The unsupervised algorithms implemented are KMeans [132] and spectral clustering (SC) [133].
2) *Dimensionality Reduction:* The algorithms (all unsupervised) implemented are fast independent component analysis (FastICA) [134], nonnegative matrix factorization (NMF) [135], and principal component analysis (PCA) [136].
3) *Restoration:* The (unsupervised) algorithm implemented is Pansharpening [137].
4) *Unmixing:* The supervised algorithms implemented are linear spectral unmixing (LSU) [138] and fully constrained linear spectral unmixing (FCLSU) [138]. The unsupervised algorithms implemented are latent Dirichlet allocation (LDA) [139], probabilistic latent semantic analysis (pLSA) [140], and vertex component analysis (VCA) [75].
5) *Change detection:* This operation is performed with the data provided by the SentinelHub platform. This platform, in collaboration with the European Space Agency (ESA) through the Copernicus Program, allows to download RS images from the most widely used satellites, such as Sentinel-2-L1C and Sentinel-2-L2A. By selecting this operation and the algorithm to use, the user obtains the Sentinel-Hub datasets or uses one of the sample datasets that are available.[3] If the user decides to obtain the datasets from the Sentinel-Hub platform, the bot will require the necessary parameters to perform a request to the platform server. These parameters are the upper-left and lower-right corners of the area to be analyzed, the time interval, the Sentinel satellite, and finally the spatial resolution of the pixel in the image. Then, the request is sent to the platform and a dictionary called "EOPath" is obtained. Table II provides the content of dictionary, where the fields "DATA" and "MASK" contain the RS images and the corresponding labels of the requested area within the indicated time frame. In addition, the true color of the area to be analyzed is provided and shown to the user at the end of the procedure. From the obtained satellite images, the first and last scenes and the corresponding ground-truth are selected. Once the data are obtained/selected, the change that has occurred in that area is analyzed. The bot has three change detection algorithms available: changes in vegetation (obtained by fixing the vegetation class labels and removing the rest); changes in humidity (the operation is performed with the bands); and changes in chlorophyll (the operation is performed with the bands of the images), shown, respectively, in the following equations:

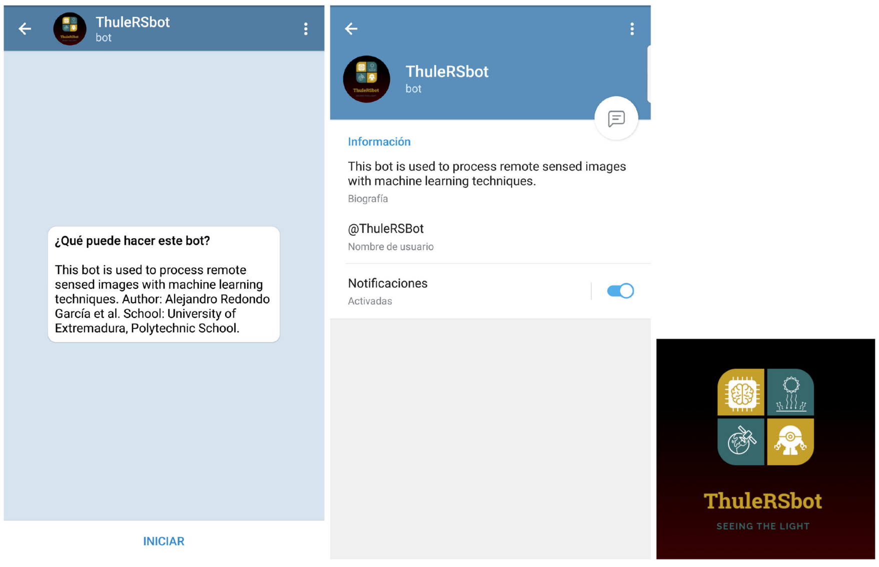
$$\text{Humidity} = \frac{B8A - B11}{B8A + B11} \tag{1}$$

$$\text{Chlorophyll} = \frac{B8A - B4}{B8A + B4}. \tag{2}$$

I. Graphical User Interface (GUI)

The general appearance of the *Bot* is shown in Fig. 7. The user can enter the information requested by the *Bot*, use the keyboard provided by the *Bot*, or type the option to execute by following the steps specified in Section II-A. An example of how the *Bot* works is shown in Fig. 8. In the example, we run the KNN algorithm with an HSI dataset (AVIRIS Indian Pines) that will be described in the following section.

[3]Several example RS images are provided by the bot, in particular an urban scene of the city of Cáceres (in Extremadura, Spain), the Melero Meander (Extremadura, Spain), or the Betsiboka river (Madagascar).



¿Qué puede hacer este bot?

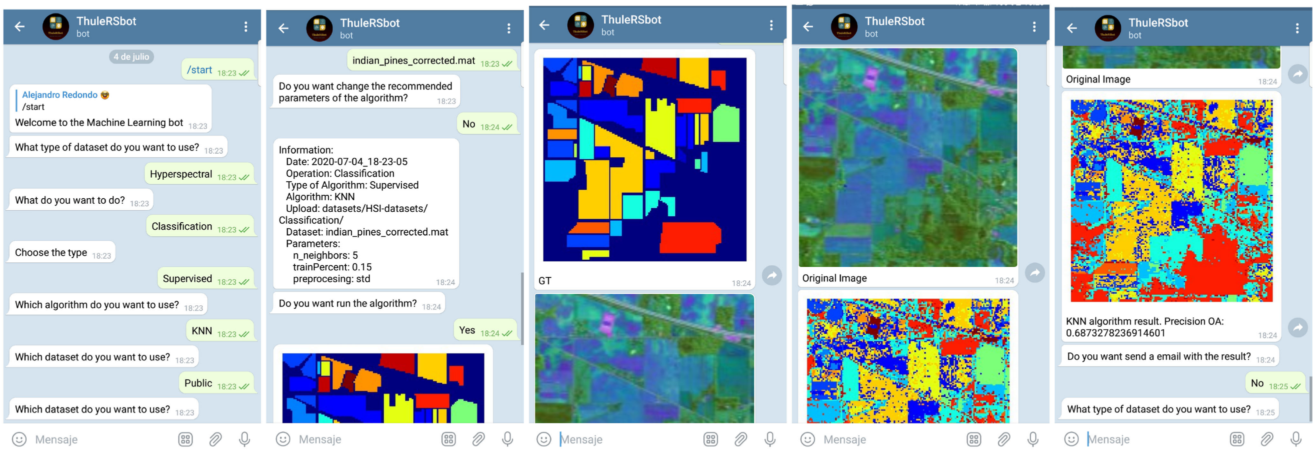
This bot is used to process remote sensed images with machine learning techniques. Author: Alejandro Redondo García et al. School: University of Extremadura, Polytechnic School.

This bot is used to process remote sensed images with machine learning techniques.
Biografía

@ThuleRSBot
Nombre de usuario

(a) (b) (c)

Fig. 7. General aspect of the application. (a) Initial screen of the *Bot* of *Telegram*. (b) General description of the *Bot*. (c) *Bot* logo.



4 de julio
/start 18:23
Alejandro Redondo
/start
Welcome to the Machine Learning bot 18:23
What type of dataset do you want to use? 18:23
Hyperspectral 18:23
What do you want to do? 18:23
Classification 18:23
Choose the type 18:23
Supervised 18:23
Which algorithm do you want to use? 18:23
KNN 18:23
Which dataset do you want to use? 18:23
Public 18:23
Which dataset do you want to use? 18:23

indian_pines_corrected.mat 18:23
Do you want change the recommended parameters of the algorithm? 18:23
No 18:24

```
Information:
  Date: 2020-07-04_18-23-05
  Operation: Classification
  Type of Algorithm: Supervised
  Algorithm: KNN
  Upload: datasets/HSI-datasets/
Classification/
  Dataset: indian_pines_corrected.mat
  Parameters:
    n_neighbors: 5
    trainPercent: 0.15
    preprocesing: std
```
18:24

Do you want run the algorithm? 18:24
Yes 18:24

KNN algorithm result. Precision OA: 0.6873278236914601 18:24
Do you want send a email with the result? 18:24
No 18:25
What type of dataset do you want to use? 18:25

(a) (b) (c) (d) (e)

Fig. 8. Example execution of the KNN algorithm in the *Bot.* (a) Algorithm selection. (b) Algorithm parameters. (c) Image and ground-truth. (d) Classification result. (e) Classification metrics.

III. Experiments

A. Runtime Environment

Our tests have been performed on Intel i9-9940X processor, 128 GB of DDR4 RAM, and a clock frequency of 2100 MHz and the GPU is an NVidia Titan RTX.

B. Datasets Description

This section explains the datasets supported by the developed application. In particular, three types of optical RS datasets have been employed: 1) standard RGB datasets, such as UCMERCED [141] and NWPU-RESISC45 [142] datasets; 2) MSI datasets, such as Berlin and Munich datasets [143], and 3) HSI datasets, such as AVIRIS Indian Pines and ROSIS University of Pavia scenes [86]. Also, the final user can obtain his/her own datasets through the SentinelHub platform. In this sense, the developed bot provides the available options in the corresponding menu, where the user should select between available options to obtain the desired dataset. The proposed application support dataset files with. mat,. tif,. png, and. jpg extensions.

C. Case Study

In this case study, we focus on a change detection application to demonstrate the usefulness of our application to observe the changes that have occurred in the environment.

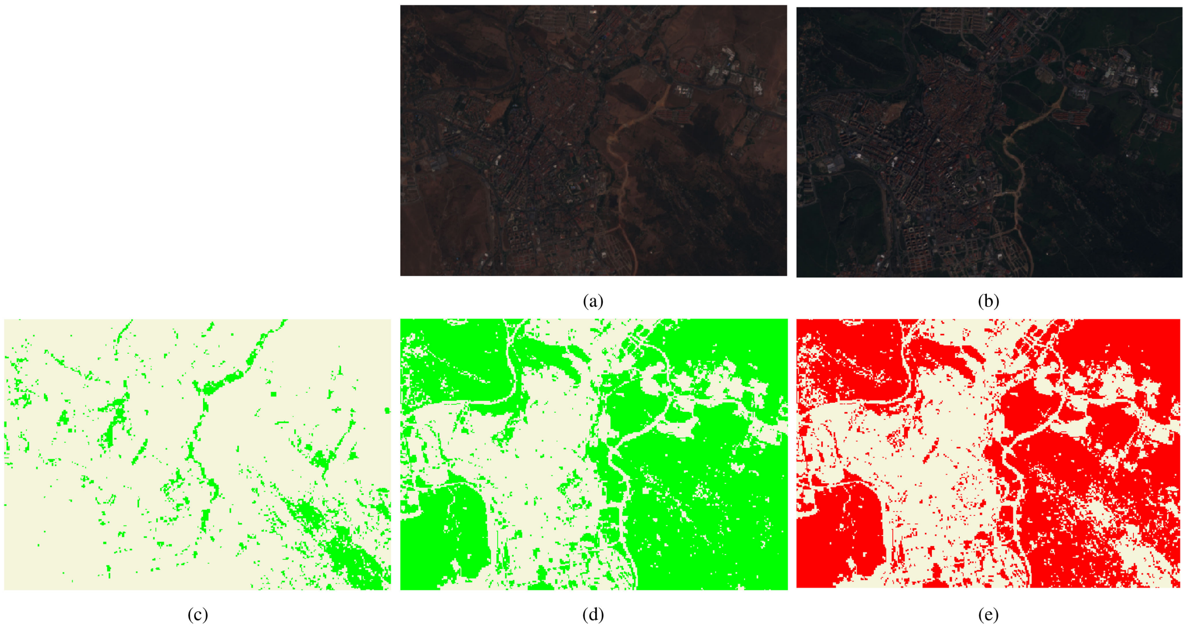


Fig. 9. (a) RGB image obtained in the summer over the city of Cáceres, Spain. (b) RGB image obtained in the winter. (c) Vegetation present in the area of Cáceres on Jul. 19, 2019. (d) Vegetation present in the area of Cáceres on Feb. 26, 2020. (e) Subtraction of both figures representing the change of vegetation. Color legend: White is the area without vegetation and green represents vegetation.

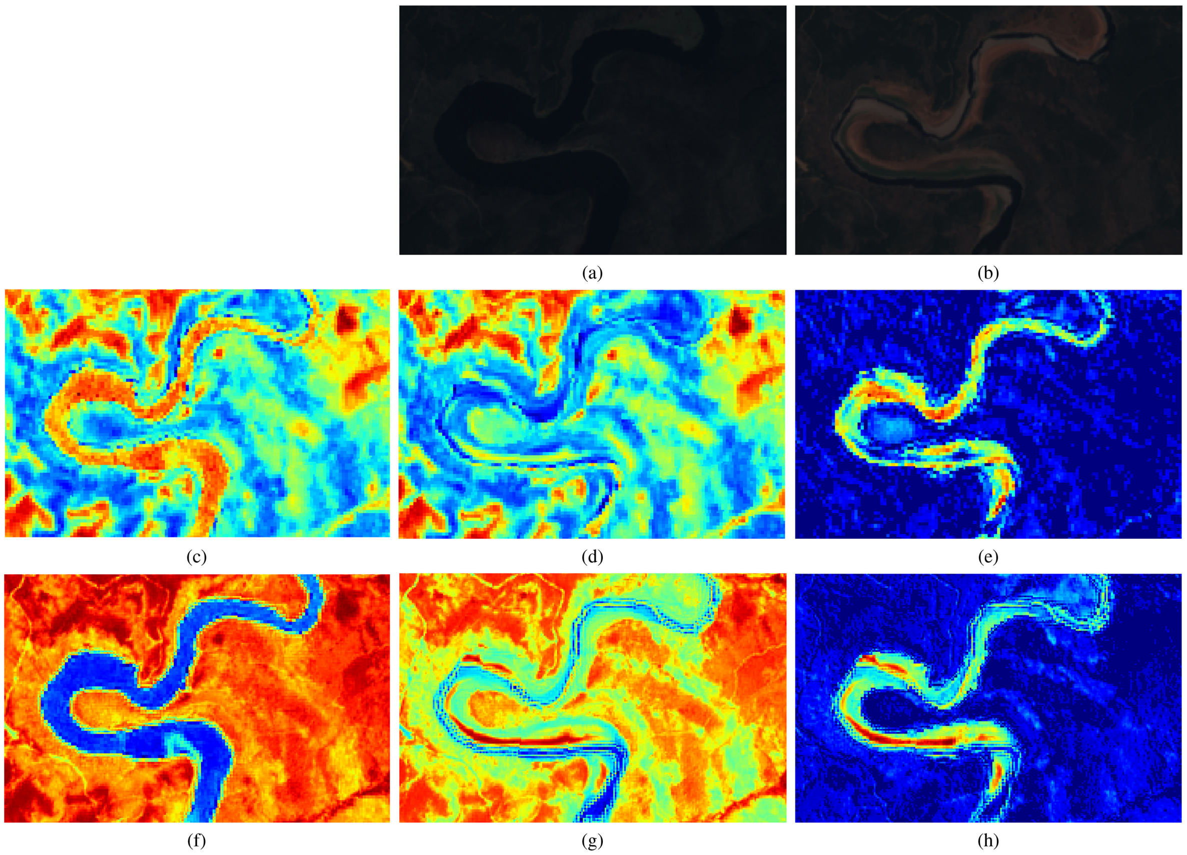


Fig. 10. (a) Winter RGB image obtained over *Melero Meander* in Spain. (b) Summer RGB image obtained over *Melero Meander*. (c) Humidity present in *Melero Meander* in Apr. 17, 2018. (d) Vegetation present in *Melero Meander* in Jul. 16, 2019. (e) Restoration of the images representing the change in humidity that has occurred in the area. (f) Chlorophyll present in *Melero Meander* in Apr. 17, 2018. (g) Chlorophyll present in *Melero Meander* in Jul. 16, 2019. (h) Restoration of both figures representing the chlorophyll change that has occurred in the area. The colors represent the degree of humidity/chlorophyll: dark blue, light blue, orange, and red (from no humidity/chlorophyll to high humidity/chlorophyll, respectively).

Green spaces in cities are becoming more important, as they help to reduce environmental pollution and improve the quality of life. In our first experiment, the Sentinel-Hub platform is used to obtain two labeled satellite images of the same area at different times. To visualize the change, only the labels belonging to the vegetation class are obtained. The considered city is Cáceres, Spain, which belongs to the region of Extremadura and was declared a World Heritage Site by Unesco in 1986 for being one of the most complete urban complexes of the Middle Ages and the Renaissance.

The studied area is bounded by the coordinates (−6.40185, 39.48914) and (−6.33327, 39.45197), these being the upper left and lower right point of the imaginary rectangle. The images were obtained on Jul. 19, (summer) and Feb. 26, 2020(winter), the area is represented in RGB in Fig. 9(b) and (c). In the results obtained and presented in Fig. 9, the changes of vegetation in the city can be observed. In Fig. 9(e), the vegetation that has grown at that time can be also appreciated.

The second experiment focuses on studying the variation of humidity and chlorophyll of a natural landscape. The considered landscape is *Melero Meander* that forms the Alagón River in Extremadura. This natural gem is defined by many tourism magazines as one of the most beautiful and impressive places in Spanish geography.

The studied area is delimited by the coordinates (−6.0875, 40.3970) and (−6.059, 40.3831), these being the upper left and lower right point of the imaginary rectangle. The images were obtained from the day Apr. 17, 2018 (spring) and Jul. 16, 2019 (summer), the area is represented in RGB in Fig. 10(b) and (c).

As we can observe in Fig. 10(e), the greatest change in humidity in this time interval has occurred in the river by rising temperatures. On the other hand, with the results obtained from chlorophyll in Fig. 10(h), an increase in the river area can be observed, and this leads to the presence of algae.

IV. Conclusion

In this article, we have introduced a new *Telegram* application for the automatic processing of RS images. The application has been developed in a *Bot* of *Telegram* as a software tool that is easy to use, safe, and widely accessible. The app provides a simple mechanism to include new algorithms and operations for the processing of RS images in a simple and organized way. Specially, if research works want to use the app, they first need to download our code shared in github and install it, and then use telegram bot as an interface to input the corresponding datasets and algorithms to get the final results.

As future work, we plan to carry out the implementation of new algorithms, both on CPUs and graphical processing units (GPUs), and new operations related to image processing. The current *Bot* allows to collect information from the Sentinel Hub platform. A possible extension is to collect satellite images from other platforms and/or other satellites. Finally, another possible way to expand this application is to port it to *WeChat* [144], the most famous instant messaging application in Asia, since *Telegram* is mostly used in Europe and America.

References

[1] M. Herold, D. A. Roberts, M. E. Gardner, and P. E. Dennison, "Spectrometry for urban area remote sensing-development and analysis of a spectral library from 350 to 2400nm," *Remote Sens. Environ.*, vol. 91, no. 3/4, pp. 304–319, 2004.

[2] 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.

[3] E. Chuvieco *et al.*, "Historical background and current developments for mapping burned area from satellite earth observation," *Remote Sens. Environ.*, vol. 225, pp. 45–64, 2019.

[4] J. C. Demro, R. Hartshorne, L. M. Woody, P. A. Levine, and J. R. Tower, "Design of a multispectral, wedge filter, remote-sensing instrument incorporating a multiport, thinned, CCD area array," *Imag. Spectrometry*, vol. 2480, pp. 280–286, 1995.

[5] J. Everaerts *et al.*, "The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping," *Int. Archives Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 37, no. 2008, pp. 1187–1192, 2008.

[6] M. Younis, F. Bordoni, N. Gebert, and G. Krieger, "Smart multi-aperture radar techniques for spaceborne remote sensing, in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2008, vol. 3, pp. III- 278–III-281.

[7] D. F. Heath, "Large-aperture spectral radiance calibration source for ultraviolet remote sensing instruments," *Proc. SPIE*, vol. 4891, pp. 335–342, 2003.

[8] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS J. Photogrammetry Remote Sens.*, vol. 92, pp. 79–97, 2014.

[9] C. Livingstone, A. Gray, R. Hawkins, P. Vachon, T. Lukowski, and M. Lalonde, "The CCRS airborne SAR systems: Radar for remote sensing research," *Can. J. Remote Sens.*, vol. 21, no. 4, pp. 468–491, 1995.

[10] M. A. Lefsky, W. Cohen, S. Acker, G. G. Parker, T. Spies, and D. Harding, "Lidar remote sensing of the canopy structure and biophysical properties of Douglas-fir western Hemlock forests," *Remote Sens. Environ.*, vol. 70, no. 3, pp. 339–361, 1999.

[11] L. Zhang, K. Yang, and H. Li, "Regions of interest detection in panchromatic remote sensing images based on multiscale feature fusion," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 12, pp. 4704–4716, Dec. 2014.

[12] M. Choi, R. Y. Kim, M.-R. Nam, and H. O. Kim, "Fusion of multispectral and panchromatic satellite images using the curvelet transform," *IEEE Geosci. Remote Sens. Lett.*, vol. 2, no. 2, pp. 136–140, Apr. 2005.

[13] R. Kestur, S. Farooq, R. Abdal, E. Mehraj, O. S. Narasipura, and M. Mudigere, "UFCN: A fully convolutional neural network for road extraction in RGB imagery acquired by remote sensing from an unmanned aerial vehicle," *J. Appl. Remote Sens.*, vol. 12, no. 1, 2018, Art. no. 016020.

[14] P. D. Heermann and N. Khazenie, "Classification of multispectral remote sensing data using a back-propagation neural network," *IEEE Trans. Geosci. Remote Sens.*, vol. 30, no. 1, pp. 81–88, Jan. 1992.

[15] P. Mitra, B. U. Shankar, and S. K. Pal, "Segmentation of multispectral remote sensing images using active support vector machines," *Pattern Recognit. Lett.*, vol. 25, no. 9, pp. 1067–1074, 2004.

[16] M. Borengasser, W. S. Hungate, and R. Watkins, *Hyperspectral Remote Sensing: Principles and Applications*. Boca Raton, FL, USA: CRC Press, 2007.

[17] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geosci. Remote Sens. Mag.*, vol. 1, no. 2, pp. 6–36, Jun. 2013.

[18] L. Bruzzone and D. F. Prieto, "An adaptive semiparametric and context-based approach to unsupervised change detection in multitemporal remote-sensing images," *IEEE Trans. Image Process.*, vol. 11, no. 4, pp. 452–466, Apr. 2002.

[19] Q. Zhu, Y. Zhong, B. Zhao, G.-S. Xia, and L. Zhang, "Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 6, pp. 747–751, Jun. 2016.

[20] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS J. Photogrammetry Remote Sens.*, vol. 117, pp. 11–28, 2016.

[21] G. Pajares, "Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs)," *Photogrammetry Eng. Remote Sens.*, vol. 81, no. 4, pp. 281–330, 2015.

[22] B. Qi, H. Shi, Y. Zhuang, H. Chen, and L. Chen, "On-board, real-time preprocessing system for optical remote-sensing imagery," *Sensors*, vol. 18, no. 5, pp. 1328–1346, 2018.

[23] M. Dalla Mura, S. Prasad, F. Pacifici, P. Gamba, J. Chanussot, and J. A. Benediktsson, "Challenges and opportunities of multimodality and data fusion in remote sensing," *Proc. IEEE*, vol. 103, no. 9, pp. 1585–1601, Sep. 2015.

[24] G. B. Marchisio and A. Q. Li, "Intelligent system technologies for remote sensing repositories," in *Inf. Proc. Remote Sensing*. Singapore: World Scientific, 1999, pp. 541–562.

[25] J. Sevilla and A. Plaza, "A new digital repository for hyperspectral imagery with unmixing-based retrieval functionality implemented on GPUs," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2267–2280, Jun. 2014.

[26] M. Gashnikov and N. Glumov, "Hyperspectral images repository using a hierarchical compression," in *Proc. Conf. Comput. Graph., Visual. Comput. Vis.*, 2015, pp. 1–4.

[27] Y. Zhong *et al.*, "Mini-UAV-borne hyperspectral remote sensing: From observation and processing to applications," *IEEE Geosci. Remote Sens. Mag.*, vol. 6, no. 4, pp. 46–62, Dec. 2018.

[28] D. J. Mulla, "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps," *Biosyst. Eng.*, vol. 114, no. 4, pp. 358–371, 2013.

[29] M. S. Moran, Y. Inoue, and E. Barnes, "Opportunities and limitations for image-based remote sensing in precision crop management," *Remote Sens. Environ.*, vol. 61, no. 3, pp. 319–346, 1997.

[30] C. Delenne *et al.*, "Textural approaches for vineyard detection and characterization using very high spatial resolution remote sensing data," *Information Processing for Remote Sensing*, vol. 29, no. 4, pp. 1153–1167, 2008.

[31] F. Gao *et al.*, "Fusing Landsat and Modis data for vegetation monitoring," *IEEE Geosci. Remote Sens. Mag.*, vol. 3, no. 3, pp. 47–60, Sep. 2015.

[32] J. G. Clevers and L. Kooistra, "Using hyperspectral remote sensing data for retrieving canopy chlorophyll and nitrogen content," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 574–583, Apr. 2012.

[33] P. R. Coppin and M. E. Bauer, "Digital change detection in forest ecosystems with remote sensing imagery," *Remote Sens. Rev.*, vol. 13, no. 3/4, pp. 207–234, 1996.

[34] R. E. McRoberts and E. O. Tomppo, "Remote sensing support for national forest inventories," *Remote Sens. Environ.*, vol. 110, no. 4, pp. 412–419, 2007.

[35] C. A. Bishop, J. G. Liu, and P. J. Mason, "Hyperspectral remote sensing for mineral exploration in Pulang, Yunnan province, China," *Int. J. Remote Sens.*, vol. 32, no. 9, pp. 2409–2426, 2011.

[36] Y. J. Kaufman and B.-C. Gao, "Remote sensing of water vapor in the near IR from EOS/MODIS," *IEEE Trans. Geosci. Remote Sens.*, vol. 30, no. 5, pp. 871–884, Sep. 1992.

[37] V. E. Brando and A. G. Dekker, "Satellite hyperspectral remote sensing for estimating estuarine and coastal water quality," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 6, pp. 1378–1387, Jun. 2003.

[38] C. King, N. Baghdadi, V. Lecomte, and O. Cerdan, "The application of remote-sensing data to monitoring and modelling of soil erosion," *Catena*, vol. 62, no. 2/3, pp. 79–93, 2005.

[39] S. Malec, D. Rogge, U. Heiden, A. Sanchez-Azofeifa, M. Bachmann, and M. Wegmann, "Capability of spaceborne hyperspectral EnMAP mission for mapping fractional cover for soil erosion modeling," *Remote Sens.*, vol. 7, no. 9, pp. 11 776–11 800, 2015.

[40] D. Žížala, T. Zádorová, and J. Kapička, "Assessment of soil degradation by erosion based on analysis of soil properties using aerial hyperspectral images and ancillary data, Czech Republic," *Remote Sens.*, vol. 9, no. 1, pp. 28–52, 2017.

[41] B. W. Pengra, C. A. Johnston, and T. R. Loveland, "Mapping an invasive plant, phragmites Australis, in coastal wetlands using the EO-1 hyperion hyperspectral sensor," *Remote Sens. Environ.*, vol. 108, no. 1, pp. 74–81, 2007.

[42] S. L. Ustin, D. DiPietro, K. Olmstead, E. Underwood, and G. J. Scheer, "Hyperspectral remote sensing for invasive species detection and mapping," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2002, vol. 3, pp. 1658–1660.

[43] M. Prabhakar, Y. Prasad, M. Thirupathi, G. Sreedevi, B. Dharajothi, and B. Venkateswarlu, "Use of ground based hyperspectral remote sensing for detection of stress in cotton caused by leafhopper (hemiptera: Cicadellidae)," *Comput. Electron. Agriculture*, vol. 79, no. 2, pp. 189–198, 2011.

[44] E. Chuvieco *et al.*, "Development of a framework for fire risk assessment using remote sensing and geographic information system technologies," *Ecological Modelling*, vol. 221, no. 1, pp. 46–58, 2010.

[45] H. Zhang, H.-J. Song, and B.-C. Yu, "Application of hyper spectral remote sensing for urban forestry monitoring in natural disaster zones," in *Proc. Int. Conf. Comput. Manage.*, 2011, pp. 1–4.

[46] S. Veraverbeke, "Hyperspectral remote sensing of fire: State-of-the-art and future perspectives," *Remote Sens. Environ.*, vol. 216, pp. 105–121, 2018.

[47] R. A. Schowengerdt, *Remote Sensing: Models and Methods for Image Processing*. New York, NY, USA: Academic, 1997.

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

[49] S. Ahmad, A. Kalra, and H. Stephen, "Estimating soil moisture using remote sensing data: A machine learning approach," *Adv. Water Resour.*, vol. 33, no. 1, pp. 69–80, 2010.

[50] 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.

[51] X. Ceamanos and S. Valero, "Processing hyperspectral images," *Opt. Remote Sens. Land Sur.*, pp. 163–200, 2016, doi: 10.1016/B978-1-78548-102-4.50004-1.

[52] 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.

[53] M. R. Farrar and E. A. Smith, "Spatial resolution enhancement of terrestrial features using deconvolved SSM/I microwave brightness temperatures," *IEEE Trans. Geosci. Remote Sens.*, vol. 30, no. 2, pp. 349–355, Mar. 1992.

[54] D. G. Long and D. L. Daum, "Spatial resolution enhancement of SSM/I data," *IEEE Trans. Geosci. Remote Sens.*, vol. 36, no. 2, pp. 407–417, Mar. 1998.

[55] J. B. Lee, A. S. Woodyatt, and M. Berman, "Enhancement of high spectral resolution remote-sensing data by a noise-adjusted principal components transform," *IEEE Trans. Geosci. Remote Sens.*, vol. 28, no. 3, pp. 295–304, May 1990.

[56] M. T. Eismann and R. C. Hardie, "Application of the stochastic mixing model to hyperspectral resolution enhancement," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 9, pp. 1924–1933, Sep. 2004.

[57] X. Sun, L. Zhang, H. Yang, T. Wu, Y. Cen, and Y. Guo, "Enhancement of spectral resolution for remotely sensed multispectral image," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 5, pp. 2198–2211, May 2015.

[58] P. Liang, W. Shi, and X. Zhang, "Remote sensing image classification based on stacked denoising autoencoder," *Remote Sens.*, vol. 10, no. 1, pp. 16–28, 2018.

[59] P. Liu, F. Huang, G. Li, and Z. Liu, "Remote-sensing image denoising using partial differential equations and auxiliary images as priors," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 3, pp. 358–362, May 2012.

[60] B.-C. Gao, M. J. Montes, C. O. Davis, and A. F. Goetz, "Atmospheric correction algorithms for hyperspectral remote sensing data of land and ocean," *Remote Sens. Environ.*, vol. 113, pp. S 17–S24, 2009.

[61] H. Zhang, W. He, L. Zhang, H. Shen, and Q. Yuan, "Hyperspectral image restoration using low-rank matrix recovery," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 8, pp. 4729–4743, Aug. 2014.

[62] J. Serra-Sagristà and F. Aulí-Llinàs, "Remote sensing data compression," in *Computational Intelligence for Remote Sensing*. New York, NY, USA: Springer, 2008, pp. 27–61.

[63] F. Garcia-Vilchez and J. Serra-Sagrista, "Extending the CCSDS recommendation for image data compression for remote sensing scenarios," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 10, pp. 3431–3445, Oct. 2009.

[64] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, no. 4, pp. 779–785, Jul. 1994.

[65] H. Shirmard, E. Farahbakhsh, A. B. Pour, A. M. Muslim, R. D. Müller, and R. Chandra, "Integration of selective dimensionality reduction techniques for mineral exploration using aster satellite data," *Remote Sens.*, vol. 12, no. 8, pp. 1261–1290, 2020.

[66] D. Fernandez, C. Gonzalez, D. Mozos, and S. Lopez, "FPGA implementation of the principal component analysis algorithm for dimensionality reduction of hyperspectral images," *J. Real-Time Image Process.*, vol. 16, no. 5, pp. 1395–1406, 2019.

[67] W. Sun and Q. Du, "Hyperspectral band selection: A review," *IEEE Geosci. Remote Sens. Mag.*, vol. 7, no. 2, pp. 118–139, 2019.

[68] Q. Wang, Q. Li, and X. Li, “Hyperspectral band selection via adaptive subspace partition strategy,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 12, pp. 4940–4950, Dec. 2019.

[69] T. Su, “Scale-variable region-merging for high resolution remote sensing image segmentation,” *ISPRS J. Photogrammetry Remote Sens.*, vol. 147, pp. 319–334, 2019.

[70] L. Mi and Z. Chen, “Superpixel-enhanced deep neural forest for remote sensing image semantic segmentation,” *ISPRS J. Photogrammetry Remote Sens.*, vol. 159, pp. 140–152, 2020.

[71] A. A. Farag, R. M. Mohamed, and A. El-Baz, “A unified framework for map estimation in remote sensing image segmentation,” *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 7, pp. 1617–1634, Jul. 2005.

[72] J. Yuan, D. Wang, and R. Li, “Remote sensing image segmentation by combining spectral and texture features,” *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 16–24, Jan. 2014.

[73] Y. Zhong, R. Feng, and L. Zhang, “Non-local sparse unmixing for hyperspectral remote sensing imagery,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 1889–1909, Jun. 2014.

[74] W.-K. Ma *et al.*, “A signal processing perspective on hyperspectral unmixing: Insights from remote sensing,” *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 67–81, Jan. 2014.

[75] J. M. Nascimento and J. M. Dias, “Vertex component analysis: A fast algorithm to unmix hyperspectral data,” *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 4, pp. 898–910, Apr. 2005.

[76] M.-D. Iordache, J. M. Bioucas-Dias, and A. Plaza, “Sparse unmixing of hyperspectral data,” *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 6, pp. 2014–2039, Jun. 2011.

[77] X. Tao, T. Cui, A. Plaza, and P. Ren, “Simultaneously counting and extracting endmembers in a hyperspectral image based on divergent subsets,” *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8952–8966, Dec. 2020.

[78] G. Cheng *et al.*, “Object detection in remote sensing imagery using a discriminatively trained mixture model,” *ISPRS J. Photogrammetry Remote Sens.*, vol. 85, pp. 32–43, 2013.

[79] C.-I. Chang and D. C. Heinz, “Constrained subpixel target detection for remotely sensed imagery,” *IEEE Trans. Geosci. Remote Sens.*, vol. 38, no. 3, pp. 1144–1159, May 2000.

[80] H. Sun, X. Sun, H. Wang, Y. Li, and X. Li, “Automatic target detection in high-resolution remote sensing images using spatial sparse coding bag-of-words model,” *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 1, pp. 109–113, Jan. 2011.

[81] A. Y.-M. Lin, A. Novo, S. Har-Noy, N. D. Ricklin, and K. Stamatiou, “Combining GeoEye-1 satellite remote sensing, UAV aerial imaging, and geophysical surveys in anomaly detection applied to archaeology,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 4, pp. 870–876, Dec. 2011.

[82] S. Khazai, S. Homayouni, A. Safari, and B. Mojaradi, “Anomaly detection in hyperspectral images based on an adaptive support vector method,” *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 646–650, Jul. 2011.

[83] R. E. Kennedy *et al.*, “Remote sensing change detection tools for natural resource managers: Understanding concepts and tradeoffs in the design of landscape monitoring projects,” *Remote Sens. Environ.*, vol. 113, no. 7, pp. 1382–1396, 2009.

[84] J. R. Jensen and J. Im, “Remote sensing change detection in urban environments,” in *Geo-Spatial Technologies in Urban Environments*. New York, NY, USA: Springer, 2007, pp. 7–31.

[85] F. Wang, “Fuzzy supervised classification of remote sensing images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 28, no. 2, pp. 194–201, Mar. 1990.

[86] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, “Deep learning classifiers for hyperspectral imaging: A review,” *ISPRS J. Photogrammetry Remote Sens.*, vol. 158, pp. 279–317, 2019.

[87] F. Melgani and L. Bruzzone, “Classification of hyperspectral remote sensing images with support vector machines,” *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[88] W. Zhang, P. Tang, and L. Zhao, “Remote sensing image scene classification using CNN-CapsNet,” *Remote Sens.*, vol. 11, no. 5, pp. 494–516, 2019.

[89] M. Han, X. Zhu, and W. Yao, “Remote sensing image classification based on neural network ensemble algorithm,” *Neurocomputing*, vol. 78, no. 1, pp. 133–138, 2012.

[90] S. Ji, C. Zhang, A. Xu, Y. Shi, and Y. Duan, “3D convolutional neural networks for crop classification with multi-temporal remote sensing images,” *Remote Sens.*, vol. 10, no. 1, pp. 75–92, 2018.

[91] Y. Dong, Z. Ding, F. Chiclana, and E. Herrera-Viedma, “Dynamics of public opinions in an online and offline social network,” *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2017.2676810.

[92] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, “Measurement and analysis of online social networks,” in *Proc. 7th ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 29–42.

[93] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, “User interactions in social networks and their implications,” in *Proc. ACM EuroSys*, 2009, pp. 205–218.

[94] M. Trusov, A. V. Bodapati, and R. E. Bucklin, “Determining influential users in internet social networks,” *J. Marketing Res.*, vol. 47, no. 4, pp. 643–658, 2010.

[95] L. Nguyen, Z. Yang, J. Li, Z. Pan, G. Cao, and F. Jin, “Forecasting people’s needs in hurricane events from social network,” *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2019.2941887.

[96] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for smart cities,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[97] A. Kumar, S. R. Sangwan, and A. Nayyar, “Multimedia social big data: Mining, in *Multimedia Big Data Computing for IoT Applications*. New York, NY, USA: Springer, 2020, pp. 289–321.

[98] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, “Online human-bot interactions: Detection, estimation, and characterization,” in *Proc. 11th Int. AAAI Conf. Web Social Media*, vol. 11, no. 1, May 2017.

[99] J. Seering, J. P. Flores, S. Savage, and J. Hammer, “The social roles of bots: Evaluating impact of bots on discussions in online communities,” *Proc. ACM Human-Comput. Interact.*, vol. 2, pp. 1–29, 2018.

[100] “About Telegram,” Accessed: Nov. 1, 2020. [Online]. Available: https://telegram.org/

[101] “ Telegram bot,” Accessed: Nov. 1, 2020. [Online]. Available: https://core.telegram.org/bots

[102] “Bot API,” Accessed: Nov. 1, 2020. [Online]. Available: https://core.telegram.org/bots/api

[103] J. Li, J. A. Benediktson, 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.

[104] J. Li *et al.*, “Social media: New perspectives to improve remote sensing for emergency response,” *Proc. IEEE*, vol. 105, no. 10, pp. 1900–1912, Oct. 2017.

[105] M. Jendryke, T. Balz, S. C. McClure, and M. Liao, “Putting people in the picture: Combining big location-based social media data and remote sensing imagery for enhanced contextual urban information in shanghai,” *Comput., Environ. Urban Syst.*, vol. 62, pp. 99–112, 2017.

[106] L. Qi, J. Li, Y. Wang, and X. Gao, “Urban observation: Integration of remote sensing and social media data,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 11, pp. 4252–4264, Nov. 2019.

[107] Y. Yao, J. Zhang, Y. Hong, H. Liang, and J. He, “Mapping fine-scale urban housing prices by fusing remotely sensed imagery and social media data,” *Trans. GIS*, vol. 22, no. 2, pp. 561–581, 2018.

[108] J. F. Rosser, D. Leibovici, and M. Jackson, “Rapid flood inundation mapping using social media, remote sensing and topographic data,” *Natural Hazards*, vol. 87, no. 1, pp. 103–120, 2017.

[109] P. Flach, *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge, U.K: Cambridge Univ. Press, 2012.

[110] M. Zeng, Y. Cai, Z. Cai, X. Liu, P. Hu, and J. Ku, “Unsupervised hyperspectral image band selection based on deep subspace clustering,” *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 12, pp. 1889–1893, Dec. 2019.

[111] Y. Su, J. Li, A. Plaza, A. Marinoni, P. Gamba, and S. Chakravortty, “DAEN: Deep autoencoder networks for hyperspectral unmixing,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 4309–4321, Jul. 2019.

[112] 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.

[113] L. Mou and X. X. Zhu, “Learning to pay attention on spectral domain: A spectral attention module-based convolutional network for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 1, pp. 110–122, Jan. 2020.

[114] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, “Active learning with convolutional neural networks for hyperspectral image classification using a new Bayesian approach,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6440–6461, Nov. 2018.

[115] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.

[116] X. Zhu, F. Cai, J. Tian, and T. K.-A. Williams, “Spatiotemporal fusion of multisource remote sensing data: Literature survey, taxonomy, principles, applications, and future directions,” *Remote Sens.*, vol. 10, no. 4, pp. 527–550, 2018.
[117] J. Kukačka, V. Golkov, and D. Cremers, “Regularization for deep learning: A taxonomy,” 2017, *arXiv:1710.10686*.
[118] “ENVI,” Accessed: Nov. 1, 2020. [Online]. Available: https://www.sigsa.info/productos/envi
[119] “Leoworks,” Accessed: Nov. 1, 2020. [Online]. Available: http://www.esa.int/SPECIALS/Eduspace_ES/SEMNAWQVEAG_0.html
[120] A. M. L. Echeverry, C. Cabrera, and L. E. V. Ayala, “Introducción a la calidad de software,” *Scientia et technica*, vol. 2, no. 39, pp. 326–331, 2008.
[121] D. Bakken, “Middleware,” in *Encyclopedia of Distributed Learning*, vol. 11, Mar. 2004, doi: 10.4135/9781412950596.
[122] J. Jakobsen and C. Orlandi, “On the CCA (in) security of mtproto,” in *Proc. 6th Workshop Secur. Privacy Smartphones Mobile Devices*, 2016, pp. 113–116.
[123] P. Mahajan and A. Sachdeva, “A study of encryption algorithms AES, DES and RSA for security,” *Global J. Comput. Sci. Technol.*, vol. 13, no. 15-E, p. 15, 2013.
[124] M. Steiner, G. Tsudik, and M. Waidner, “Diffie-Hellman key distribution extended to group communication,” in *Proc. 3rd ACM Conf. Comput. Commun. Secur.*, 1996, pp. 31–37.
[125] P. Gallagher and A. Director, “Secure hash standard (SHS),” *FIPS PUB*, vol. 180, p. 183, 1995. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/180/4/final
[126] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.
[127] S. Balakrishnama and A. Ganapathiraju, “Linear discriminant analysis–A brief tutorial,” *Inst. Signal Inf. Process.*, vol. 18, pp. 1–8, 1998.
[128] D. Bóhning, “Multinomial logistic regression algorithm,” *Ann. Inst. Statist. Math.*, vol. 44, no. 1, pp. 197–200, 1992.
[129] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[130] B. Schólkopf *et al.*, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.
[131] S. S. Haykin *et al.*, *Neural Networks and Learning machines*. Hamilton. ON, Canada: McMaster Univ., 2009.
[132] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, 2003.
[133] U. V. Luxburg, “A tutorial on spectral clustering,” *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
[134] E. Oja and Z. Yuan, “The fastica algorithm revisited: Convergence analysis,” *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1370–1381, Nov. 2006.
[135] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 556–562.
[136] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrrics Intell. Lab. Syst.*, vol. 2, no. 1-3, pp. 37–52, 1987.
[137] G. Vivone *et al.*, “A critical comparison among pansharpening algorithms,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2565–2586, May 2015.
[138] J. L. Silván-Cárdenas and L. Wang, “Fully constrained linear spectral unmixing: Analytic solution using fuzzy sets,” *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 3992–4002, Nov. 2010.
[139] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” *J. Mach. Learn. Res.*, vol. 3, no. Jan, pp. 993–1022, 2003.
[140] T. Hofmann, “Probabilistic latent semantic analysis,” 2013, in *Proc. 22nd Ann. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, New York, NY, USA, 1999, pp. 50–57, doi: 10.1145/312624.312649.
[141] Y. Yang and S. Newsam, “Bag-of-visual-words and spatial extensions for land-use classification,” in *Proc. ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. ACM*, 2010, pp. 270–279.
[142] G. Cheng, J. Han, and X. Lu, “Remote sensing image scene classification: Benchmark and state-of-the-art,” *Proc. IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct. 2017.
[143] R. Fernandez-Beltran, J. M. Haut, M. E. Paoletti, J. Plaza, A. Plaza, and F. Pla, “Multimodal probabilistic latent semantic analysis for Sentinel-1 and Sentinel-2 image fusion,” *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 9, pp. 1347–1351, Sep. 2018.
[144] E. Harwit, “WeChat: Social and political development of China’s dominant messaging app,” Chin. *J. Commun.*, vol. 10, no. 3, pp. 312–327, 2017.