UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA EN INFORMÁTICA
EN INGENIERÍA DE COMPUTADORES

TRABAJO FIN DE GRADO

Un bot social para el procesamiento de imágenes de la Tierra

UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA EN INFORMÁTICA
EN INGENIERÍA DE COMPUTADORES

TRABAJO FIN DE GRADO

Un bot social para el procesamiento de imágenes de la Tierra

Autor: Miguel Blanco Bazaga

Tutor: Antonio Plaza Miguel

Co-Tutor: Mercedes Eugenia Paoletti Ávila

Este trabajo se lo dedico
a todas las personas que me apoyaron
y me hicieron seguir adelante.
Gracias a todos.

Agradecimientos

Quiero expresar mis agradecimientos a todas las personas que me prestaron su apoyo para conseguir llevar a cabo este trabajo: profesores, compañeros de clase, amigos y familiares. A los profesores, ya que me aportaron las nociones necesarias para conseguir realizar este trabajo. A los compañeros de clase, por intercambiar opiniones sobre cómo desarrollar de forma correcta las actividades que se nos requerían. A mis amigos, por ayudarme en los momentos más complejos, animándome a no rendirme. Y a mi familia, por respetar mi espacio y tiempo de aislamiento.

En particular, me gustaría agradecer a mi amigo y compañero de clase, Fernando Mancha Sánchez, que me ayudó en la tarea de encontrar las mejores ideas sobre cómo desarrollar los objetivos marcados y, sobre todo, en la fase de pruebas, haciendo las labores de *beta tester* intentando llevar al límite a la aplicación para conseguir encontrar fallos con el fin de solventarlos.

También me gustaría poder darles las gracias a Antonio Plaza Miguel y a Mercedes Eugenia Paoletti Ávila, que me han dado la oportunidad de desarrollar este trabajo y me han orientado con una gran disposición ante las dudas que me han ido surgiendo.

Resumen

El uso de las redes sociales y las funcionalidades que aportan las aplicaciones de mensajería instantánea crecen cada día. Si sumamos esto a las continuas mejoras en las tecnologías de observación remota de la Tierra, tenemos como resultado un escenario en el que los usuarios pueden obtener rápidamente información sobre eventos críticos que suceden a lo largo de nuestro planeta, participando de forma activa (aunque se encuentren en ubicaciones geográficas dispersas). Por ejemplo, en un terremoto o inundación es difícil planificar una respuesta utilizando únicamente datos espaciales obtenidos por sensores remotos de observación de la Tierra. No obstante, si dichos datos se complementan con información en tiempo real ofrecida por usuarios en redes sociales, es posible complementar ambas fuentes de información para responder de forma más efectiva ante dicho evento crítico, permitiendo así incrementar la (limitada) resolución temporal de los datos de satélite obtenidos de forma remota.

En este contexto, el paradigma es convertir a los usuarios en sensores (tanto activos como pasivos) para recolectar información útil en este tipo de eventos. Dicha información puede complementarse con datos obtenidos a partir de satélites de observación remota de la Tierra. Esto desemboca en una nueva forma de interacción entre los usuarios con el planeta, y nos permite responder de forma más precisa ante eventos críticos. La integración de las redes sociales con las aplicaciones de mensajería (junto con las mejoras en la tecnología dentro del mundo de la teledetección) ofrecen, por tanto, nuevas perspectivas y (sobre todo) un gran volumen de información que precisa ser adecuadamente integrada y procesada. En la actualidad, no existen herramientas avanzadas para procesar este tipo datos, que puede aportar información de gran utilidad en determinados contextos.

En la actualidad, existen numerosas investigaciones abiertas sobre cómo recoger, almacenar, procesar y fusionar datos provenientes de redes sociales y de sensores de observación remota de la Tierra. Debido al gran volumen de información de estos datos integrados, es necesario un gran poder de cómputo e infraestructuras informáticas cada vez más potentes, puesto que procesar este tipo de información consume muchos recursos tanto de computación como de almacenamiento. En este trabajo, desarrollamos una nueva aplicación que pretende combinar nuevas técnicas de procesamiento de datos obtenidos de redes sociales con datos de satélites de observación remota de la Tierra, con el objetivo de apoyar en la toma de decisiones ante eventos críticos, reduciendo el tiempo que transcurre desde que se produce dicho evento hasta que se decide la mejor forma de afrontarlo.

Palabras clave: redes sociales, espacio, sensores remotos, teledetección.

Abstract

The use of social networks and instant messaging applications has exponentially increased in recent years. In addition to the continuous improvements on remote sensing technologies for Earth observation, the incorporation of social media data provides a scenario where users can quickly obtain information about critical events happening in our planet (such as earthquakes, floods, or forest fires, and interact with each other generating critical information about these events (regardless of their geographical location). For example, in the aforementioned disasters, it is difficult to plan a response by only using remotely sensed data obtained by Earth observation sensors. However, if these data are combined with real-time information provided by users on social networks, it is possible to complement both sources of information to respond more effectively to a critical event by increasing the (often limited) temporal resolution of satellite data.

In this context, the main paradigm consists of transforming users in both active and passive sensors able to collect information about critical events happening in the Earth. The information generated in social media can be fused and integrated with data obtained from Earth observation satellites satellites. This leads to a new form of interaction between users and our planet, which allows us to explore new applications with important societal impact. In fact, the integration of social networks with messaging applications (together with improvements in remote sensing technology) offers new perspectives and, specifically, a tremendous volume of information that needs to be efficiently processed, managed and interpreted. To the best of our knowledge, there are no advanced tools to process this kind of integrated data, which can provide very useful information in different contexts.

The collection, storage, manage and integration of data coming from social networks and Earth observation instruments is a challenging topic, due to the extremely high requirements imposed by the generated data volume, and the need for high-power and high-performance computing infrastructures. Processing this type of integrated information is expected to consume many computational resources, as well as a great amount of disk space.

In this work, we developed a new application to combine social media data with Earth observation data coming from remote sensing instruments. Our goal is to reduce the latency since a critical event (e.g. forest fire, earthquake, flood) takes place until an adequate response plan can be properly elaborated by emergency response teams.

Keywords: social networks, spatial data, remote sensing.

Índice

Índice de tablas

Índice de figuras

1. INTRODUCCIÓN

El número de usuarios que utilizan redes sociales crece exponencialmente día a día. Se estima que, en 2017, las redes sociales tuvieron 2.796 millones de usuarios. En 2018, el número de usuarios creció un 14 % (ampliando la cifra hasta llegar a 3.196 millones de usuarios). En 2019, se alcanzaron los 3.484 millones de usuarios [1]. Este notable crecimiento se ilustra en la Figura 1. Los usuarios de redes sociales pueden aportar información o consumirla. En el presente trabajo, exploramos la posibilidad de combinar esta información con datos recogidos por sensores de observación remota de la Tierra, complementando así estos datos con información *in-situ* que puede ser de gran utilidad en el caso de eventos críticos como terremotos, inundaciones, incendios forestales, etc. en los que la resolución temporal de los datos de satélites de observación remota de la Tierra es demasiado limitada para realizar un seguimiento en tiempo real.

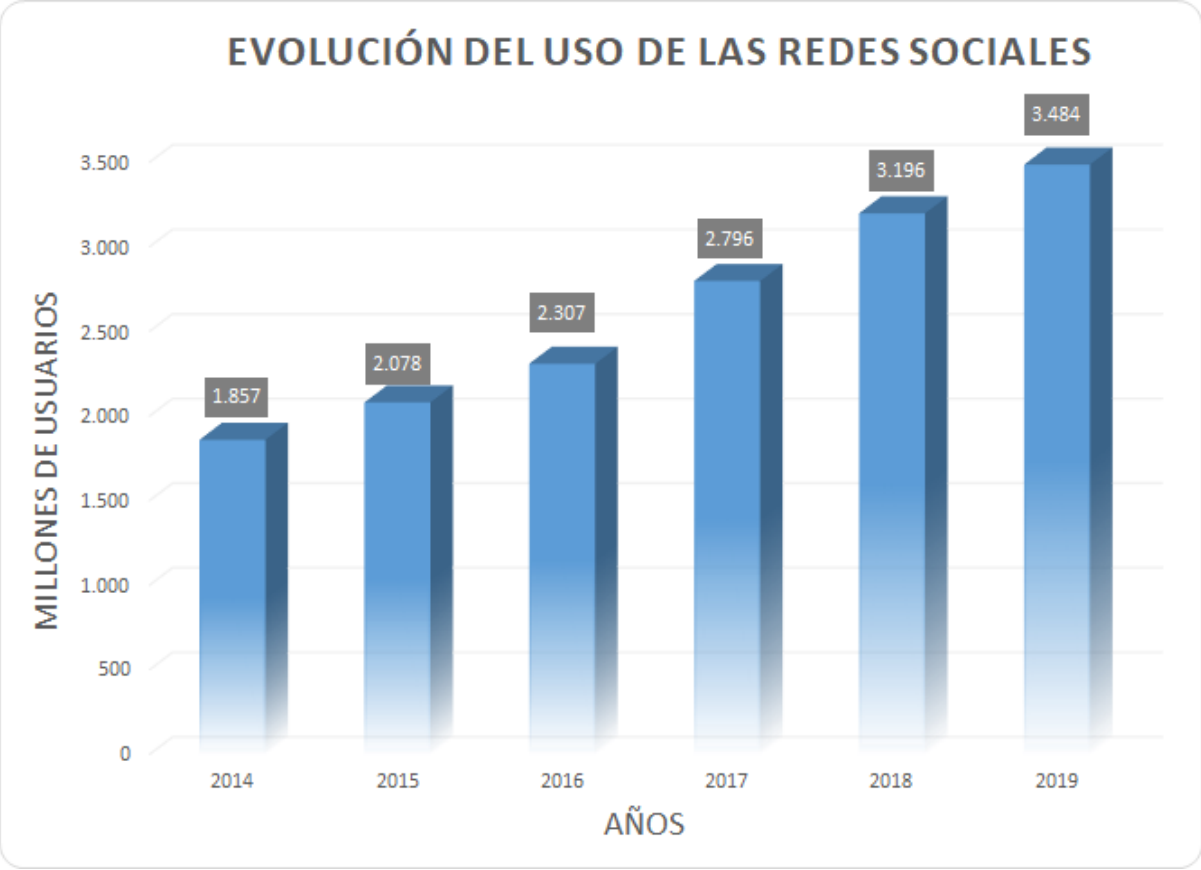


Figura 1: Crecimiento del número de usuarios de las redes sociales durante los seis últimos años. [1].

Los datos de observación remota de la Tierra, normalmente provenientes de satélites

dotados con potentes sensores remotos, pero también de sistemas de información geográfica (SIG), o incluso de los propios usuarios, abren nuevas posibilidades para entender cómo un conjunto de usuarios de redes sociales pueden convertirse en sensores activos y pasivos, lo que se conoce como el concepto de "*citizen sensor*" [2]. Además, la posible integración de dichos datos con la información proveniente de sistemas de observación remota de la Tierra sienta un precedente para los nuevos sistemas sociales de uso popular. Esto se traduce en que, para cada usuario, podemos conocer de manera rápida y sencilla su ubicación y recoger información acerca de su entorno que puede complementar a la información obtenida por dichos sistemas de observación remota de la Tierra. Dichos sensores pueden ser aerotrasportados o encontrarse ubicados en plataformas espaciales [3]. En este sentido, la integración de datos existentes en redes sociales con datos de observación remota de la Tierra permite explotar las posibilidades que ofrece la colaboración y la externalización de tareas completamente abiertas al público (*crowdsourcing*) [4] de las redes sociales. Todos estos datos que se obtienen, que pueden entenderse como una forma de *big data* [5], precisan técnicas avanzadas de aprendizaje automático y, en particular, técnicas de *machine learning* [6] para su interpretación, además de un sistema informático de alto rendimiento para su almacenamiento y procesamiento eficiente, debido al gran volumen de información generada, a la variedad de dichos datos, y a la velocidad con la que estos datos se generan (sobre todo, en el contexto de las redes sociales).

Las imágenes obtenidas por sensores de observación remota de la Tierra pueden dividirse en dos grandes grupos, según su número de bandas espectrales [7]: imágenes multiespectrales, cuando tienen decenas de bandas, e imágenes hiperespectrales, cuando tienen cientos de ellas. En una imagen hiperespectral, las bandas espectrales cubren gran parte del espectro electromagnético. Se recogen desde sensores montados a bordo de satélites y en plataformas aerotransportadas [8]. Dichos sensores pueden ser activos o pasivos (al igual que los usuarios de las redes sociales). Un sensor activo genera su propia radiación de forma artificial y la recibe reflejada [9]. Los sensores pasivos son aquellos que reciben la radiación electromagnética que emite o refleja la Tierra [9].

Uno de los grandes problemas que presenta el uso de sensores de observación remota de la Tierra es que no siempre se encuentran disponibles cuando se produce algún evento crítico, como pueden ser terremotos, inundaciones o incendios, lo que produce gran lentitud en la toma de decisiones debido a la falta de disponibilidad de datos con la suficiente frecuencia temporal que permita hacer un seguimiento adecuado del evento, preferiblemente en tiempo real. En este

sentido, una de las soluciones que se han explorado en la literatura consiste en convertir a los usuarios que se encuentren en la zona afectada en sensores ("*citizen sensors*", tanto activos como pasivos), utilizando los datos adquiridos por dichos usuarios (eventualmente disponibles en redes sociales) para mitigar las carencias temporales en la adquisición de datos por parte de las plataformas de observación remota de la Tierra (por ejemplo, el tiempo empleado por un satélite en volver a obtener una imagen sobre una misma zona puede ser de varios días, lo cual no aporta la resolución temporal necesaria para efectuar el seguimiento de un evento crítico como un incendio, un terremoto o una inundación).

Uno de los propósitos principales del presente trabajo es analizar nuevas estrategias para reducir el tiempo en ofrecer una respuesta a un evento crítico gracias a la integración de datos de redes sociales y de sensores de observación remota de la Tierra. En estudios recientes, se han analizado algunas alternativas [10, 11]. El uso de procesadores más veloces, como las *tarjetas gráficas programables* (*GPUs*) puede ayudar a lograr este aumento de velocidad desde el punto de vista del procesamiento de los datos, pero no desde el punto de vista de la disponibilidad necesaria de información en el caso de eventos críticos. Por este motivo, es necesario establecer nuevas líneas de investigación que permitan integrar y procesar eficientemente los datos obtenidos por redes sociales y sensores de observación remota de la Tierra, así como su interpretación conjunta mediante técnicas avanzadas de *machine learning* que además puedan ofrecer una respuesta en tiempo cercano al real.

En los últimos años, se han conseguido diversas mejoras en cuanto a la recogida, tratamiento y almacenamiento de información que nos ofrecen las redes sociales y datos de teledetección [12]. Sin embargo, para conseguir una toma de decisiones más rápida, es necesario integrar y explotar de forma conjunta toda esta información a la hora de afrontar un evento crítico. Con idea de avanzar en esta novedosa línea de investigación, el principal objetivo del presente trabajo es aportar un nuevo enfoque para la integración de la información que se recoge en sensores de observación remota de la Tierra y en redes sociales, basado en un marco de procesamiento de datos amplio centrado en una aplicación novedosa capaz de procesar imágenes en el contexto de la red social *Telegram*. Este nuevo enfoque proporciona una visión única sobre la integración de datos espaciales y sociales, que nunca antes había sido explorado en la literatura.

La principal contribución novedosa del presente trabajo es, por tanto, proponer una nueva solución para la combinación de datos de teledetección e información proveniente de

redes sociales para poder realizar el seguimiento de eventos críticos en tiempo muy rápido, satisfaciendo así la necesidad de recortar el tiempo en la toma de decisiones ante dichos eventos críticos [13]. El trabajo se encuentra estructurado en una serie de capítulos que se han organizado de la siguiente manera:

- En el capítulo 2 se describen los problemas existentes y se enumeran los objetivos detallados del presente trabajo.
- En el capítulo 3 se profundiza en el estado del arte actual, analizando tres aspectos fundamentales sobre los que el presente trabajo se asienta:
  - I) Qué son las imágenes hiperespectrales, en qué condiciones se adquieren y cuáles son sus principales aplicaciones (sección 3.1).
  - II) Qué métodos y algoritmos han sido los más ampliamente utilizados para el procesamiento de este tipo de imágenes (sección 3.2).
  - III) Qué herramientas se han empleado para desarrollar el presente trabajo, enfatizando las herramientas utilizadas para extracción de información a partir de redes sociales (sección 3.3).
- El capítulo 4 desarrolla la metodología empleada, describiendo la arquitectura del sistema que se ha implementado, las partes en las que se divide, y las funcionalidades y contenido de cada una de ellas.
- El capítulo 5 describe la implementación de la aplicación desarrollada, ofreciendo ejemplos de uso y detallando las funciones de que dispone.
- El capítulo 6 describe los resultados experimentales obtenidos en la validación de cada una de las operaciones disponibles en la herramienta, utilizando los diferentes algoritmos implementados en la aplicación. También se muestran datos relativos al tiempo de ejecución desde que se envía una orden hasta que el usuario recibe el resultado.
- Por último, el capítulo 7 recoge las conclusiones finales a las que se ha llegado una vez completado el desarrollo del trabajo, apuntando también posibles mejoras de la aplicación y líneas de investigación futuras.

2. OBJETIVOS

Los sensores ubicados en satélites y plataformas aerotransportadas para observación remota de la Tierra no presentan una gran resolución temporal. Es decir, no pueden tomar imágenes sobre la misma zona de la superficie terrestre en intervalos muy cortos de tiempo. Esto supone un notable inconveniente, ya que puede provocar retrasos a la hora de reaccionar ante eventos críticos en los que se requiere una respuesta rápida (por ejemplo, incendios, inundaciones o terremotos). Existen numerosos estudios sobre las diferentes posibilidades existentes para solucionar esta carencia. Hoy en día existe la posibilidad de predecir algunos eventos. Por ejemplo, es posible prever ciertos desastres naturales para minimizar los daños que éstos puedan ocasionar. Las tecnologías actuales de teledetección también brindan la posibilidad de aportar nueva información sobre el estado de una zona afectada con anterioridad al evento, como por ejemplo en un incendio se pueden generar mapas de riesgo anticipando cómo se va a propagar el mismo en función del estado fenológico de la vegetación en la zona.

Sin embargo, el aumento del uso de las redes sociales durante los últimos años, unido a las nuevas funcionalidades que van incorporando las aplicaciones que explotan la información contenida en dichas redes, ha abierto la posibilidad de utilizar estos datos en diferentes contextos, por ejemplo, para realizar el seguimiento de un evento crítico en tiempo real. Otra cuestión importante es la forma de procesar los datos que nos ofrecen estas tecnologías, ya que se necesitan nuevos métodos que proporcionen un tratamiento integrado de estos datos en conjunción con otras fuentes de información, como pueden ser los datos de satélite, de manera más rápida y precisa.

En este sentido, uno de los objetivos primordiales de este trabajo es la combinación de los datos obtenidos por sensores remotos de observación de la Tierra junto con datos procedentes de redes sociales, mediante una herramienta intuitiva y sencilla que sea capaz de unificar ambas fuentes de información y explotarla de forma computacionalmente eficiente. De esta forma, se aprovecha la información proporcionada por las aplicaciones basadas en redes sociales y se reducen los problemas asociados a la toma de decisiones tardía en aplicaciones de observación remota de la Tierra (en particular, en el caso de eventos críticos). Para conseguir este objetivo general, se persiguen varios objetivos específicos, los cuales se enumeran a continuación:

1. Minimizar, e incluso eliminar, las carencias que pueden suponer las caídas de los servicios de alguno de los satélites o plataformas aerotransportadas de observación

remota de la Tierra, o su limitada resolución temporal, a la hora de monitorizar eventos críticos, como desastres naturales.

2. Combinar los avances realizados en los últimos años en el campo de la teledetección con los realizados en las tecnologías de redes sociales, permitiendo a cualquier usuario funcionar como sensor activo/pasivo en la generación de datos, y poniendo a su disposición la posibilidad de utilizar algoritmos ampliamente utilizados en el mundo de la teledetección para el procesamiento de imágenes y datos disponibles en redes sociales.

3. Obtener información visual y datos de las zonas afectadas en un evento crítico mediante la simple toma de una fotografía (geo-localizada) del lugar, lo cual puede suponer un importante complemento a los sistemas actuales basados en observación remota de la Tierra a la hora de tomar decisiones en situaciones de emergencia.

4. Convertir a los usuarios de las redes sociales en sensores tanto activos como pasivos, para que puedan aportar nueva información relevante sobre una determinada zona de interés, en particular, en el caso de eventos críticos.

5. Almacenar un histórico de todas las consultas y ejecuciones realizadas de forma que se puedan realizar estudios temporales y acelerar nuevas consultas en el supuesto de que un usuario se encuentre ante una situación similar a una ya acontecida en el pasado.

3. ESTADO DEL ARTE

En este capítulo se presenta el estado del arte en cuanto a imágenes hiperespectrales, las operaciones y algoritmos considerados, y las herramientas de las que se parte para el desarrollo de un *bot* (programa informático que está preparado para realizar tareas que se repiten constantemente a través de Internet como si de un humano se tratase) de la aplicación *Telegram* en el que se basa nuestra nueva herramienta.

3.1. Imágenes hiperespectrales: concepto y aplicaciones

Una imagen hiperespectral es una imagen que aporta información mucho más precisa que la que puede proveer una imagen tomada por una cámara fotográfica normal, la cual solo obtiene información en tres canales visibles (RGB). Las imágenes hiperespectrales contienen cientos de bandas espectrales de información a través de todo el espectro electromagnético [14].

La longitud de onda del espectro de la luz que se puede ver se encuentra entre 0.4 y 0.7 μm, lo que en colores se puede traducir como desde el violeta hasta el rojo. La radiación ultravioleta es la porción del espectro que tiene longitudes de onda por debajo de 0.4 μm. La región que supera el valor de 0.7 μm es conocida como región del infrarrojo, y puede llegar a tomar un valor aproximado de 100 μm. La banda infrarroja puede dividirse en dos regiones: región de infrarrojo reflejada y región de infrarrojo emitida [15].

Conocer el espectro de un cuerpo permite apreciar propiedades que son invisibles al ojo humano. La forma de obtener una imagen hiperespectral consiste en emplear un sensor de barrido (*whiskbroom*) o empuje (*pushbroom*) para recolectar, línea a línea y píxel a píxel la información espectral que lo caracteriza. La finalidad es recoger, para cada pixel, su firma espectral característica. En otras palabras, para adquirir estas imágenes se utilizan sensores hiperespectrales, los cuales obtienen imágenes digitales con una gran cantidad de bandas espectrales en las que, para cada píxel, se adquiere una firma espectral o *huella* característica de cada material observado [16].

Para representar estas imágenes se utiliza un cubo de datos tridimensional, donde las dos primeras dimensiones representan las dimensiones espaciales de la escena, y la tercera representa la dimensión espectral. Gracias a esta representación se puede acceder a cualquier firma espectral capturada por el sensor accediendo a la información registrada en cada uno de los píxeles de la imagen.

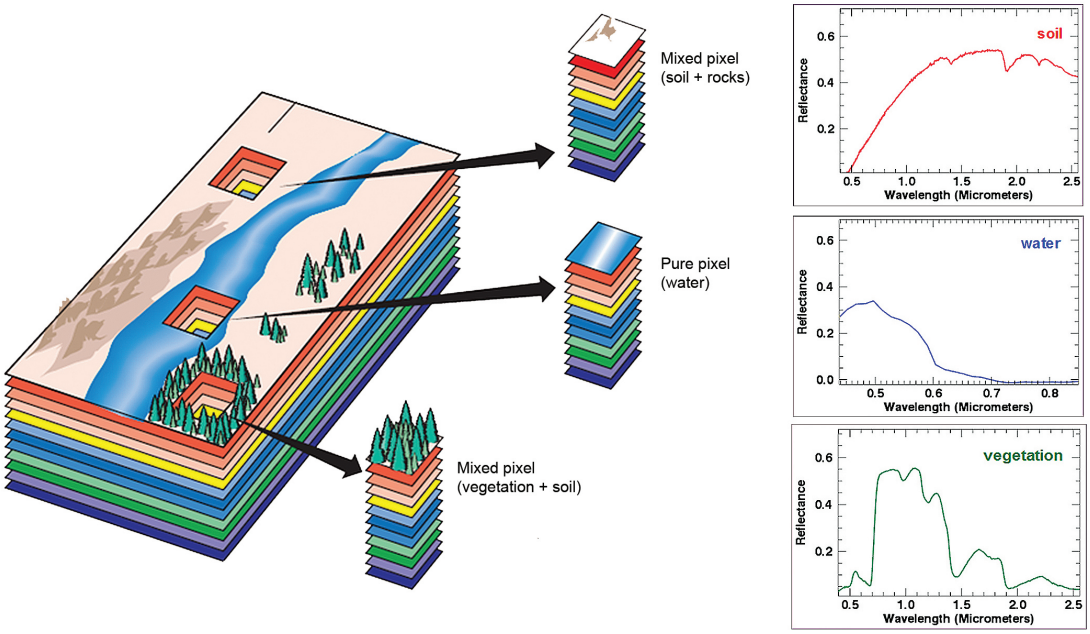


Figura 2: Ejemplo de imagen hiperespectral con píxeles puros y píxeles mezcla [17].

En este tipo de imágenes, los píxeles pueden ser de dos tipos: puros y mezcla. Se puede definir un pixel mezcla como aquel en el que coexisten diferentes materiales puros [18]. Estos píxeles son frecuentes en las imágenes hiperespectrales, como se observa en la Figura 2.

Los dispositivos de adquisición de imágenes remotas están diseñados para medir la radiación reflejada desde varias áreas adyacentes de la superficie terrestre. Hasta hace poco, estos dispositivos estaban restringidos a unas pocas longitudes de onda, debido a las limitaciones en el diseño del sensor y a la forma de almacenar, procesar y transmitir esta información. Los sensores que recogen decenas de bandas espectrales son conocidos como sensores multiespectrales, y los sensores que pueden captar cientos de bandas se conocen como sensores hiperespectrales [7]. La diferencia entre las imágenes multiespectrales e hiperespectrales se puede apreciar en la Figura 3.

Los sensores que captan estas imágenes se denominan espectrómetros [19], y se encuentran en plataformas aerotransportadas o en satélites. Estos instrumentos recogen la reflectancia de los materiales que componen la escena. El desarrollo de estos sensores integra dos tecnologías: la espectroscopia y la teledetección. La espectroscopia estudia la luz que emiten o reflejan los materiales y su variación de energía con la longitud de onda [20], y se basa en el espectro de la luz solar reflejado de forma dispersa por los materiales terrestres. La teledetección se refiere precisamente a la capacidad de observación remota de dichos sensores.

Dependiendo de su funcionamiento, los sensores pueden clasificarse en sensores activos o

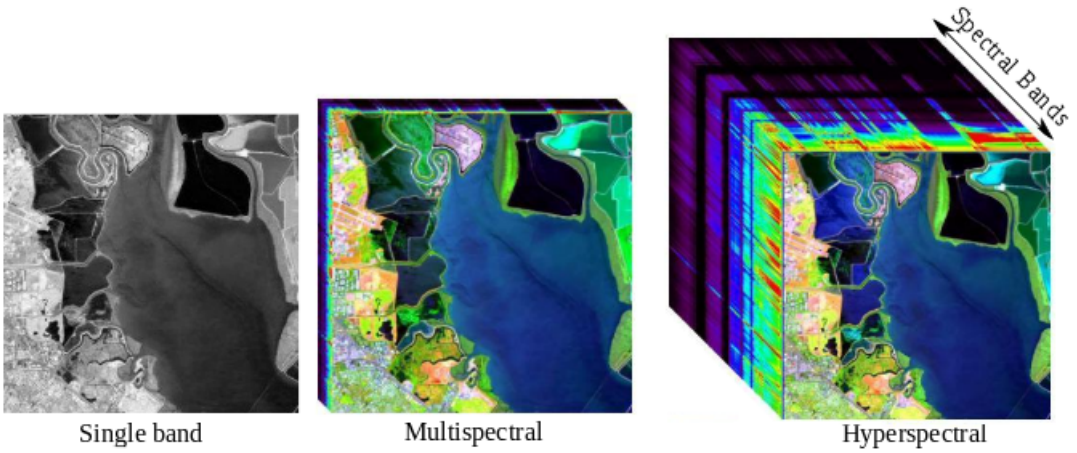


Figura 3: Diferencia entre imagen en niveles de gris (una banda), imagen multiespectral (decenas de bandas) e imagen hiperespectral (cientos de bandas). De Wikimedia Commons, el repositorio multimedia libre. autor: Ant Beck (CC BY-SA 3.0).

pasivos [9]. Los sensores pasivos son los que trabajan con la radiación que emiten o reflejan los materiales de la superficie de la Tierra. Los sensores activos son los que reciben la radiación que ellos mismos emiten de forma artificial. La diferencia entre estos dos tipos de sensores se ilustra en la Figura 4.

Uno de los sensores hiperespectrales más conocidos es el denominado *Airbone Visible Infra-Red Imaging Spectrometer (AVIRIS)* [21], un sensor aerotransportado que captura información en las zonas visible e infrarrojo del espectro. Fue el primero en obtener información en numerosas bandas espectrales estrechas y casi contiguas, y almacena la información espectral en 224 canales, cubriendo un rango de longitudes de onda entre 0.4 μm y 2.5 μm. Por su parte, el sensor *Hyperion* [22] se encuentra instalado en el satélite *Earth Observing-1 (EO-1)* [23] y adquiere 220 bandas que cubren el rango espectral de 0.4 μm a 2.5 μm. La resolución espacial de los píxeles obtenidos por ambos sensores es de 30 metros, lo cual hace muy probable que se produzcan efectos de mezcla a nivel sub-píxel en las imágenes obtenidas, debido a la reducida resolución espacial de las mismas.

Como ya se ha comentado, la resolución temporal de dichas imágenes es también reducida, al ser muy difícil planificar campañas de adquisición repetidas sobre una misma zona, en el caso de sensores aerotransportados y al hecho de que el tiempo de re-visitación de una misma zona en adquisiciones de datos vía satélite puede llevar varios días, lo cual limita la explotación adecuada de estos datos en el caso de eventos críticos que requieren una respuesta en tiempo cercano al real. Es por esto por lo que, en el presente trabajo, exploramos la posibilidad de

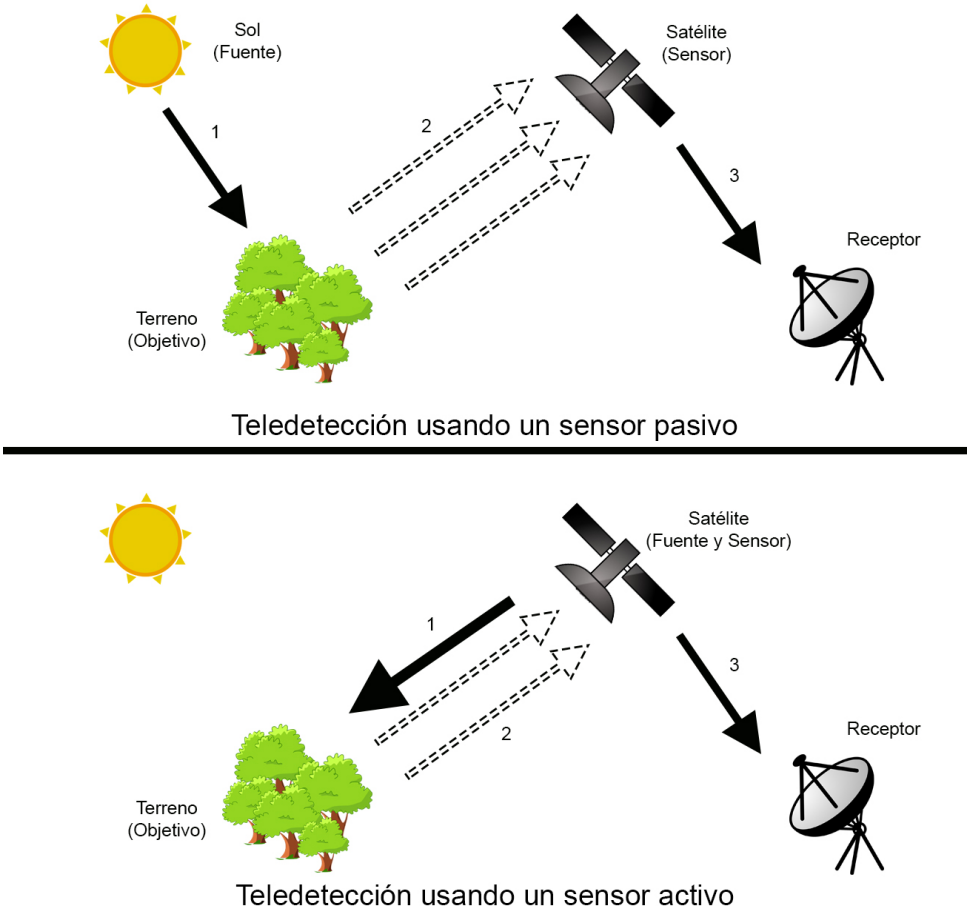


Figura 4: Ejemplo de funcionamiento de los sensores pasivos y activos.

utilizar datos de redes sociales para complementar la información proporcionada por este tipo de sensores. Antes de plantear dicha posibilidad, comentamos algunos de los algoritmos más utilizados para interpretar imágenes hiperespectrales.

3.2. Operaciones y algoritmos implementados

En este apartado se describen las operaciones y algoritmos de procesamiento de imágenes que se implementan en la aplicación desarrollada.

- Clasificación: La operación de *clasificación* [24] [25] se refiere a la tarea de agrupar un conjunto de datos de tal manera que los datos que pertenecen a un mismo grupo (clúster) sean más parecidos entre ellos que a los de otros grupos [26]. Se trata de una técnica común para el análisis de datos estadísticos. Dentro de esta operación, hemos considerado los siguientes algoritmos dada su popularidad en el ámbito de procesamiento de imágenes en general y análisis de imágenes hiperespectrales en particular:

1. *K-means:* Se trata de un algoritmo de clasificación no supervisado que agrupa los datos al tratar de separar muestras en N grupos de igual varianza, minimizando un criterio conocido como la inercia o la suma de cuadrados dentro del clúster. Requiere que se especifique el número de grupos y se adapta de forma correcta a un gran número de muestras. Este algoritmo divide un conjunto de muestras $\mathbf{X}$ en k grupos, cada una descrita por la media c_j. Las medias son comúnmente llamadas los centroides del clúster. El algoritmo *K-means* tiene como objetivo elegir los centroides que minimizan la inercia, o el criterio de suma de cuadrados dentro de cada clúster [27].

$$\sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \tag{1}$$

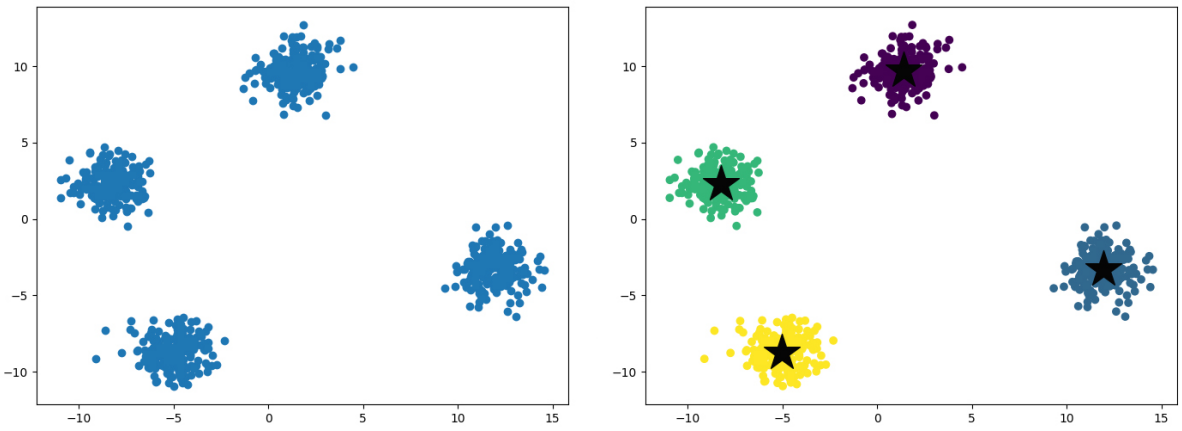


Figura 5: Ejemplo de ejecución del algoritmo *K-means* sobre un conjunto de datos aleatorios. Las estrellas representan los centroides

2. *K-nearest neighbors (KNN):* La clasificación basada en vecinos es un tipo de aprendizaje supervisado basado en instancias. No intenta construir un modelo interno general, sino que almacena instancias de los datos de entrenamiento, de forma que la clasificación se obtiene a partir de una mayoría simple de votos de los vecinos más cercanos a cada punto. A cada punto se le asigna la clase de datos con más representantes dentro de los vecinos más cercanos al punto. En este sentido, el algoritmo KNN implementa el aprendizaje basado en los K vecinos más cercanos a cada punto, donde K es un valor especificado por el usuario. La elección óptima de este valor depende en gran medida de la dispersión de los datos. En general, un valor mayor suprime los efectos del ruido, pero hace que los límites de clasificación

estén menos definidos [28].

Para determinar la distancia entre los elementos se utiliza comúnmente la distancia euclidiana:

$$d(x,x') = \sqrt{\sum_{i=1}^{p} \left(x_i - x_i'\right)^2} \tag{2}$$

El entrenamiento consiste en almacenar los elementos característicos y las etiquetas de las clases de dichos elementos de entrenamiento. En la fase de clasificación se calcula la distancia entre los valores almacenados y el nuevo y se seleccionan los K elementos más cercanos. La clase que más se repita será a la que se clasifique el nuevo elemento.

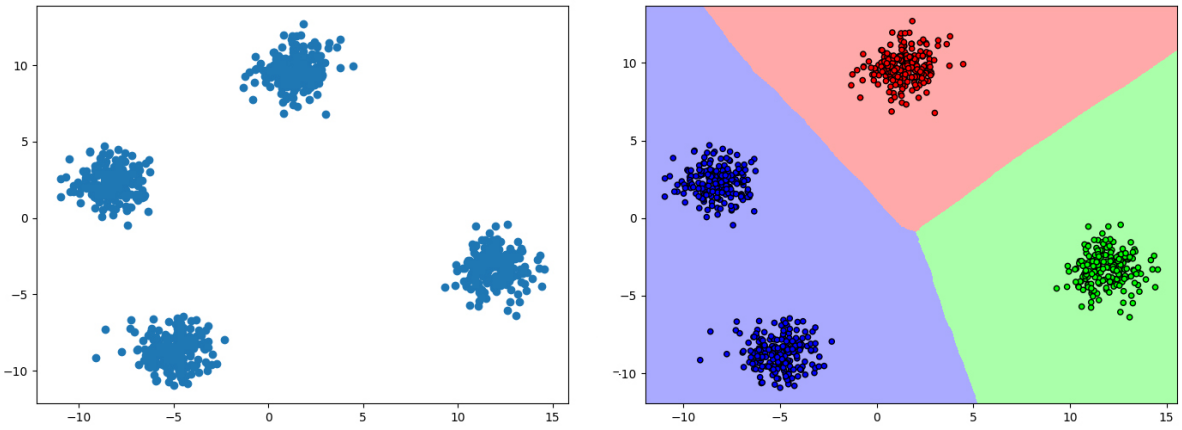


Figura 6: Ejemplo de ejecución del algoritmo *KNN* sobre un conjunto de datos aleatorios.

- Reducción de la dimensionalidad: Esta operación [29] consiste en el proceso de reducir la dimensionalidad del conjunto de los datos. Existen varios métodos para conseguir este objetivo que han sido ampliamente utilizados para reducir la dimensionalidad de imágenes hiperespectrales.

  1. *Principal Component Analysis (PCA):* Se utiliza para identificar patrones en el conjunto de los datos encontrando las direcciones de variación máxima en datos de alta dimensionalidad y proyectándolos en un subespacio dimensional más pequeño conservando la mayor parte de la información. Un componente es cada una de las variables o dimensiones que tienen los datos y cada uno de los componentes principales generados se corresponde a un *eigenvector* (vectores no nulos que cuando son transformados por el operador dan lugar a un múltiplo escalar de

sí mismos, con lo que no cambian su dirección) [30] [31]. El objetivo es identificar las combinaciones lineales que mejor representan las variables X_1,..., X_p.

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j \tag{3}$$

Donde ϕ_{1m},...,ϕ_{pm} son las constantes de los componentes principales.

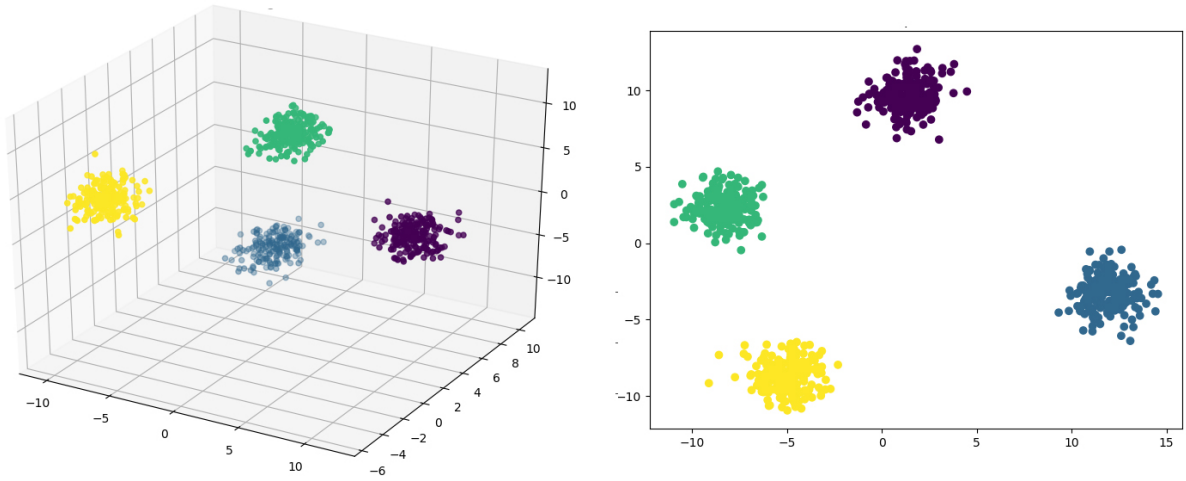


Figura 7: Ejemplo de aplicación del método PCA a un *dataset* aleatorio. Se aprecia cómo PCA es capaz de identificar las direcciones principales en las que se distribuyen los datos en el espacio, proyectando los datos sobre dichas direcciones o componentes principales.

2. *Non-Negative Matrix Factorization (NMF):* Este algoritmo presenta una alternativa que asume que los datos y los componentes no son negativos. El algoritmo encuentra una descomposición de las muestras **X** en dos matrices **W** y **H** de elementos no negativos, optimizando la distancia d entre **X** y el producto de la matriz **WH**. La función de distancia que se utiliza es la norma de *Frobenius* al cuadrado, que es una extensión de la norma euclídea a un contexto matricial [32].

$$d(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} ||\mathbf{X} - \mathbf{Y}||^2_{\text{Fro}} = \frac{1}{2} \sum_{i,j} (x_{ij} - y_{ij})^2 \tag{4}$$

Siendo $Y = WH$.

- Superresolución: La operación de *superresolución* [33] es el proceso de aumentar la resolución de las imágenes consiguiendo proporcionar detalles espaciales más precisos que los que se han capturado originalmente aumentando la cantidad de píxeles de la imagen. Esta operación es muy importante para integrar datos provenientes de

diferentes sensores y/o fuentes de información, incluyendo imágenes obtenidas por satélites, plataformas aerotransportadas e imágenes capturadas por los usuarios en su aplicación móvil. Para redimensionar las imágenes hemos considerado estos tres tipos de interpolación:

1. *Bicúbica:* Esta es una implementación de alto rendimiento que generalmente proporciona excelentes resultados considerando un entorno de 4x4 píxeles. A los píxeles más cercanos se les da una mayor ponderación en el cálculo. La interpolación *bicúbica* produce imágenes notablemente más nítidas que otros tipos de interpolaciones, y es quizás la mejor solución en términos de calidad y rendimiento [34].

$$p(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{5}$$

2. *Lanczos:* La función de interpolación de *Lanczos* es una fórmula matemática utilizada para interpolar suavemente el valor de una imagen digital entre sus muestras. Asigna cada muestra de la imagen a una copia traducida y escalada del kernel de *Lanczos*. La suma de estos núcleos traducidos y escalados se evalúa luego en el píxel deseado. La interpolación de *Lanczos* tiene las mejores propiedades en términos de preservación de detalles y generación mínima de ruido y *aliasing* para transformaciones geométricas que no impliquen un fuerte muestreo descendente [35]. El kernel de Lanczos funciona como sigue:

$$S(x) = \sum_{i=\lfloor x \rfloor - a + 1}^{\lfloor x \rfloor + a} s_i L(x - i) \tag{6}$$

3. *Nearest Neighbor:* Este es el tipo de interpolación computacionalmente más simple. La interpolación del vecino más próximo selecciona el valor del píxel más cercano redondeando las coordenadas del punto de interpolación deseado. Este método crea orificios de pixelación que dividen las curvas en pasos o bordes irregulares. Esta forma de interpolación tiene efectos normalmente inaceptables, tanto en el caso de la ampliación como en el de la reducción de tamaño de las imágenes [34]. Su ecuación es la siguiente:

$$G(x) = \sum_{i=1}^{n} w_i(x) f(x_i) \tag{7}$$

Donde $G(x)$ es la estimación de x, w_i son los pesos y $f(x_i)$ son los datos que se conocen en x_i. Los pesos x_i se calculan buscando qué parte de cada una de las áreas cercanas se pierde cuando se inserta x.

- Unmixing La operación de *unmixing* se refiere al proceso de descomponer la firma espectral de un píxel en un conjunto de firmas espectrales puras (*endmembers*) y sus correspondientes abundancias [36] [37]. Es fundamental para solventar la problemática asociada a la limitada resolución espacial de las imágenes hiperespectrales, que hace que se produzcan efectos de mezcla a nivel sub-píxel y que puede impedir una adecuada integración de datos provenientes de otras fuentes de información como, por ejemplo, datos de otros sensores, o bien imágenes geo-localizadas tomadas por usuarios desde el móvil.

  1. *Fully Constrained Linear Spectral Unmixing (FCLSU):* Es un algoritmo usado para calcular las abundancias de cada *endmember*, minimizando el error en la aproximación lineal de la imagen hiperespectral. Este método impone dos restricciones en el cálculo de abundancias: no negatividad y suma unitaria [38]. Su ecuación es la siguiente:

$$E' = \begin{bmatrix} \delta M \\ 1^T \end{bmatrix}, \Phi'(i,j) = \begin{bmatrix} \delta \Phi(i,j) \\ 1 \end{bmatrix} \tag{8}$$

  Donde E' es la nueva matriz de *endmembers*, $\Phi'(i,j)$ las fracciones de abundancias y δM controla el impacto de la restricción *abundance sum-to-one constraint (ASC)*, $\sum_{z=1}^{p} \Phi_z(i,j) = 1$, siendo p el número total de *endmembers*.

  2. *Linear Discriminant Analisys (LDA)*: Es un modelo generativo probabilístico usado para colecciones de conjuntos de datos discretos. También es un modelo que se utiliza para descubrir temas abstractos de una colección de documentos [39]. El modelo *LDA* para extracción de *endmembers* en imágenes hiperespectrales utiliza tres niveles, como se muestra en la Figura 8. Para el análisis linear se modela como una distribución gaussiana multivariable con densidad:

$$P(X|y=k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t \Sigma_k^{-1}(X-\mu_k)\right) \quad (9)$$

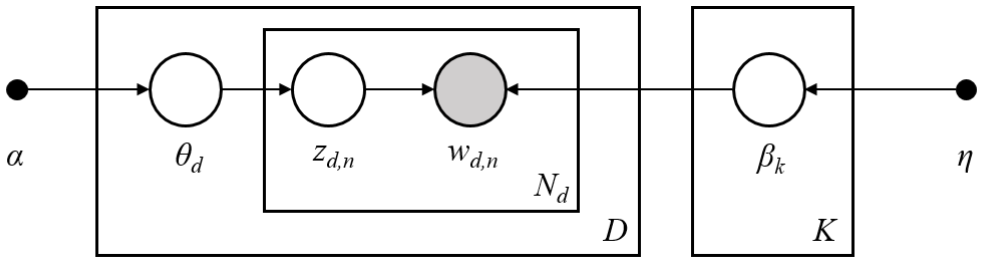


Figura 8: Ejemplo de *LDA* para *unmixing*. El cuerpo es una colección de *endmembers* en D. Un *endmember* se trata como una secuencia de *N* valores espectrales. Hay *K endmembers* en el cuerpo. Las cajas representan muestreo repetido. Figura obtenida de la web "scikit-learn".

3. *Vertex Component Analysis (VCA):* Este algoritmo es utilizado para identificar automáticamente *endmembers* en imágenes hiperespectrales. El algoritmo es un método no supervisado en el que los *endmembers* son los vértices del simplex dado por los píxeles (vectores) de la imagen hiperespectral. Funciona tanto con la imagen original o bien con una versión dimensionalmente reducida de la misma, y proyecta iterativamente los datos en una dirección ortogonal al subespacio que abarcan los *endmembers* ya determinados, iterando hasta encontrar las firmas espectrales de todos los *endmembers* [40].

$$r = x + n = M\gamma\alpha + n \quad (10)$$

Donde r es un *L*-vector (*L* es el número de bandas), $M \equiv [m1, m2, ..., m_p]$ es la matriz, γ es un factor de escala y α es el vector de abundancias de cada *endmember*.

3.3. Entorno utilizado

Los instrumentos que se han empleado para el desarrollo de este trabajo han sido: el sistema operativo *Ubuntu* 16.04, el lenguaje de programación *Python* 3.5.2, utilizando el editor de texto avanzado *Kate* y la aplicación de mensajería *Telegram*, que permite el desarrollo de *bots* que emulan el comportamiento de las redes sociales actuales.

- *Ubuntu* es una distribución del sistema operativo Linux, basado en la arquitectura de Debian de código abierto y está enfocado tanto a ordenadores de escritorio como

servidores. Utiliza una estructura de árbol para organizar el almacenamiento, y el sistema de archivos que emplea es `ext4`. Se ha optado por el uso de este sistema operativo debido a la gran cantidad de aplicaciones que vienen preinstaladas, el conocimiento del trabajo a través de consola, y las facilidades que aporta en términos de programación [41].

- *Python* es un lenguaje de programación que no utiliza compilador, ya que realiza la traducción a lenguaje máquina a medida que es necesaria. Su punto fuerte es conseguir una sintaxis con código legible y poseer una licencia de código abierto. Se optó por el uso de este lenguaje de programación debido a la existencia de módulos para trabajar tanto con *bots* de *Telegram* como con algoritmos para procesamiento de imágenes, además de funciones para interactuar con el sistema operativo [42].

- *Kate* es un editor de texto avanzado de KDE de software libre, código abierto y gratuito publicado bajo la licencia LGPL, que cuenta con resaltado de sintaxis para la mayoría de lenguajes de programación, además de un emulador de terminal integrado basado en Konsole que permite mantener abiertos varios documentos en una misma ventana.

- *Telegram* es una aplicación de mensajería y voz sobre IP que utiliza el protocolo `MTProto` para las comunicaciones y que permite el envío de mensajes y archivos entre usuarios. Los mensajes que se mandan son almacenados de forma privada y están cifrados por el propio servidor. Tiene implementados *bots* integrados que interactúan con archivos multimedia sin necesidad de depender de una aplicación externa. Y, lo más importante, permite el desarrollo de *bots* que puedan ofrecer nuevos servicios creados por los propios usuarios, emulando el comportamiento de las redes sociales actuales. Para su desarrollo es necesario obtener un *token* de validación de su funcionamiento [43].

Para poder ejecutar correctamente la aplicación hay que ejecutar los siguientes comandos en la consola del sistema operativo que instalarán las librerias necesarias:

1. `apt install python3-pip`

2. `pip3 install pyTelegramBotAPI numpy matplotlib scipy sklearn tweepy opencv-python`

3. `pip3 install google-api-python-client`

Una vez que ha finalizado el proceso de instalación de las librerías, en la consola, accedemos a la carpeta donde se encuentra ubicada la aplicación y ejecutamos el comando:

4. `python3 main.py`

Y ya podremos iniciar la conversación con el *bot* de *Telegram*.

4. METODOLOGÍA

En este capítulo se presenta la arquitectura del sistema que se ha implementado. La metodología es el conjunto de pasos y procedimientos que se han seguido para desarrollar el trabajo [44]. Una metodología correcta nos ayuda a crear software de calidad y una buena arquitectura del sistema nos ayuda a conseguir una herramienta robusta y que siempre se encuentre disponible.

4.1. Arquitectura del sistema

El sistema implementado está compuesto por tres bloques claramente diferenciados, como se puede apreciar en la Figura 9. Cada uno de estos bloques zonas cumple una serie de funciones y propósitos específicos. En concreto, la arquitectura propuesta está basada en capas completamente independientes entre ellas desde un punto de vista de alto nivel:

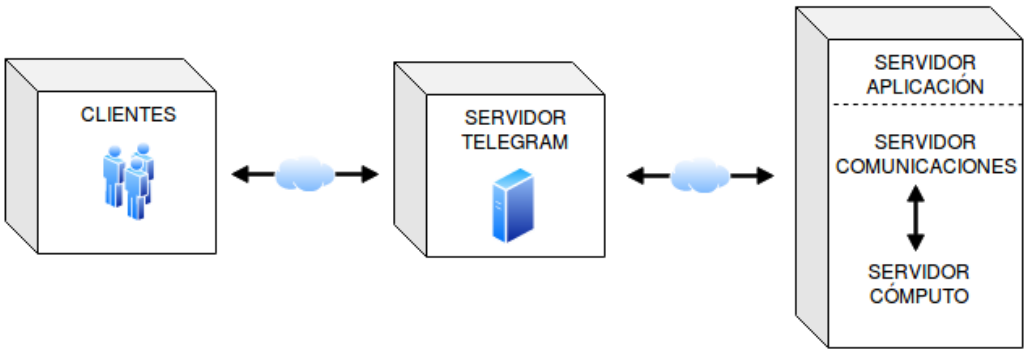


Figura 9: Diseño de la arquitectura del sistema.

Uno de los beneficios obtenidos al utilizar este tipo de sistemas es que cualquier parte puede ser reemplazada de una manera transparente al usuario. Para comunicar un bloque con otro, solamente es necesario que tengan acceso a Internet. Dicha comunicación se realiza de forma cifrada a través del protocolo `MTProto` [43], que permite el intercambio de una clave entre dos dispositivos utilizando el método intercambio de claves *Diffie-Hellman* [45].

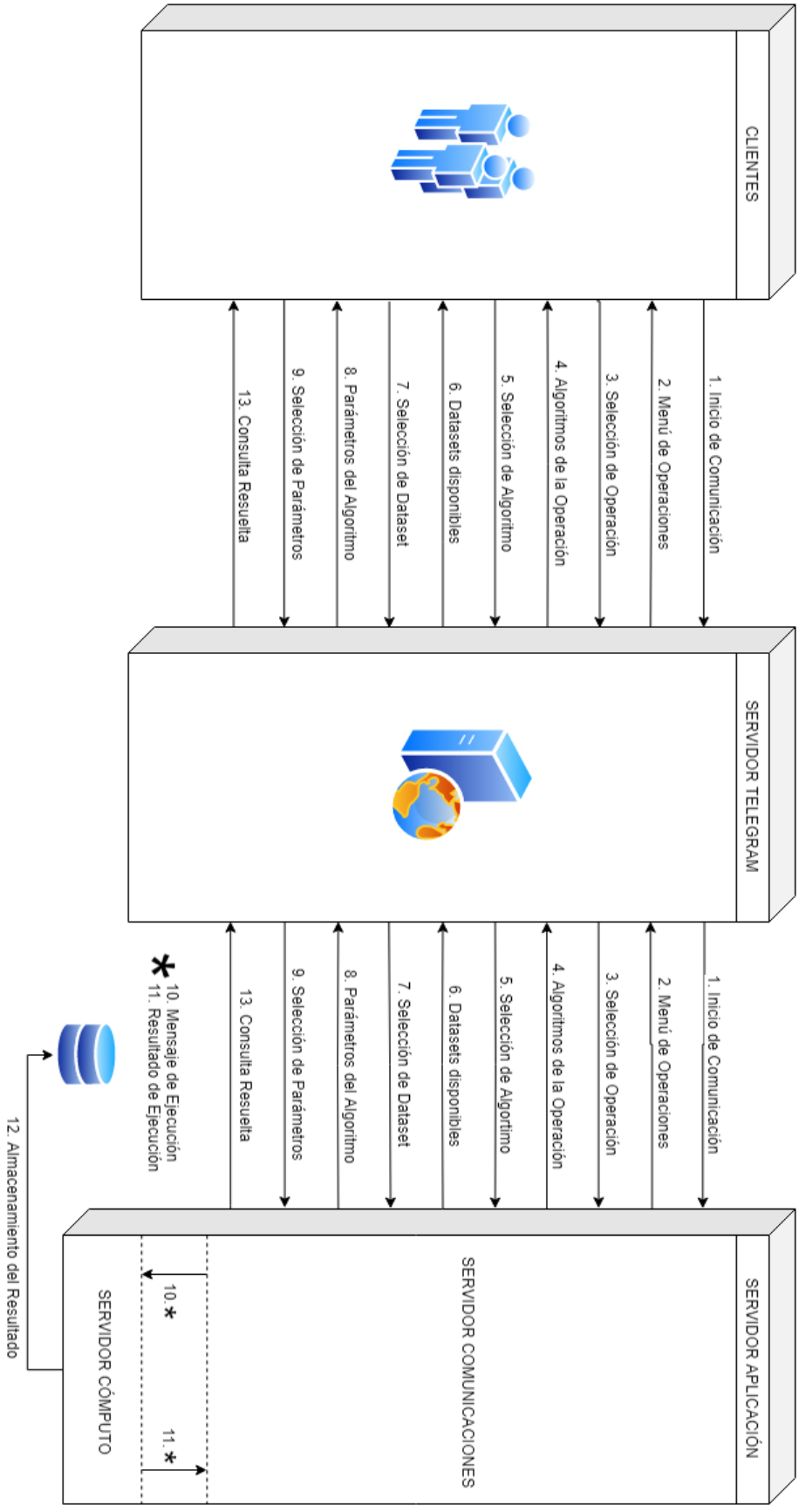


Figura 10: Especificación detallada del funcionamiento de la arquitectura.

La interacción del usuario con la aplicación se realiza a través de *Telegram*, que deberá ir seleccionando y enviando las diferentes opciones que le van apareciendo, hasta recibir el resultado con la información procesada. *Telegram* cifra las comunicaciones y almacena todos los mensajes que pasan a través de él, de manera que el bloque de aplicación se encarga de controlar las opciones del menú, de ejecutar el algoritmo con una imagen concreta, de establecer parámetros correctos, y de almacenar el resultado en la base de datos. Todos estos pasos aparecen ilustrados en la Figura 10.

Los pasos que se van realizando entre los bloques de *Telegram* y la aplicación, incluso los que se realizan dentro de ésta, son completamente transparentes al usuario. A continuación, se describe cada uno de los bloques de nuestra arquitectura en detalle.

A) *Cliente:*

En esta parte se encuentra instalada la aplicación de mensajería *Telegram* que es la herramienta con la que interactúa el usuario. Es una aplicación utilizada en múltiples plataformas, por lo que su adquisición y posterior instalación se puede realizar a través de sus repositorios oficiales, o bien a través de las plataformas de distribución digital que contenga el dispositivo:

- El cliente inicia una comunicación con el servidor de *Telegram*, enviando una petición de solicitud de comunicación con la aplicación desarrollada. El usuario recibe el menú con las diferentes operaciones y algoritmos disponibles, y simplemente debe seleccionar y enviar sus preferencias. Todas las acciones que realiza se ejecutan en el servidor de la aplicación, por lo que el servidor de *Telegram* sólo actúa como intermediario entre ambos.
- Después debe seleccionar la operación que desea ejecutar, y recibe los algoritmos asociados a dicha operación. Después, la aplicación le preguntará por la imagen que desea procesar, y puede seleccionarla desde la biblioteca pública, la biblioteca privada, buscar una imagen en *Google*, o incluso aportar su propia imagen capturada. La aplicación de *Telegram* incorpora una funcionalidad para buscar imágenes en los motores de búsqueda de *Bing* o de *Yandex*, que también puede ser usada por el usuario. Posteriormente se le requerirá que aporte información extra relativa a los parámetros predeterminados. Llegados a este punto, el usuario deberá esperar a recibir la imagen resultante del proceso de ejecución del algoritmo con el *dataset* escogido.

B) *Servidor de Telegram:*

Esta parte es la encargada de gestionar las comunicaciones y realiza las labores de intermediario entre el cliente y el servidor de la aplicación. Su principal función es que los mensajes viajen de forma cifrada a través del protocolo `MTProto` [43], para que éstos no puedan verse comprometidos por un agente externo. Para comprobar que esto no se produzca, hace uso de sus propios métodos y conecta al usuario con la aplicación de manera transparente. Para lograr identificar al servidor y a los clientes de forma anónima, envía un identificador que será único para cada uno de ellos. Todos los mensajes y consultas que pasan por él serán almacenados en su base de datos privada. En el caso de que el cliente quiera aportar su propia imagen para la ejecución de algún algoritmo, éste debe pasar por el bloque de *Telegram* primero. Esto no supone problema alguno, ya que dicho bloque no tiene restricciones a la hora de enviar archivos, ni en lo referente a la gestión de ficheros de gran tamaño y/o extensiones no conocidas. Además, si el usuario quiere enviar una imagen RGB, *Telegram* le ofrece dos posibilidades: enviar la imagen original o enviar una imagen de tamaño más reducido sin perder demasiada calidad. Para la correcta ejecución de la aplicación se recomienda que se envíen las imágenes originales. En caso de que no se disponga de una buena calidad de acceso a Internet, se recomienda el envío de una imagen de menor tamaño.

C) *Servidor de la aplicación:*

Es la parte más importante del sistema desarrollado. Está formado por dos bloques claramente diferenciados. El primero de ellos, el servidor de comunicaciones, se encarga de recoger los mensajes provenientes de *Telegram* y de enviar lo que corresponda en cada momento. El segundo, el servidor de cómputo, se encarga de la ejecución eficiente de los diferentes algoritmos de procesamiento a partir del mensaje de ejecución recibido.

- El servidor de comunicaciones es el encargado de manejar y procesar el tráfico entrante que proviene de *Telegram*, y aloja las imágenes de la biblioteca pública y privada, además de posibilitar la realización de las siguientes operaciones: *clasificación* [24] [25], *reducción de la dimensionalidad* [29], *superresolución* [33] y *unmixing* [36] [37], recibiendo la operación que el usuario selecciona en un menú de opciones. Los diferentes usuarios son reconocidos a través del identificador que *Telegram* les ha impuesto. Si el cliente decide trabajar con una imagen aportada por

él, deberá enviarla a *Telegram* y éste a su vez la envía la aplicación, con lo que se almacena en la biblioteca privada, adjudicándosele automáticamente un nombre que además incluye la fecha y hora actual. Una vez que este bloque termina de recibir todos los parámetros necesarios, crea el mensaje que enviará al servidor de cómputo la ejecución del algoritmo correspondiente. Dicho mensaje estará compuesto por el nombre del algoritmo y de la imagen seleccionada por el usuario, además de los parámetros que el usuario ha escogido y el identificador del usuario. A partir de este momento, espera a recibir los datos resultantes del servidor de cómputo y, seguidamente, envía a *Telegram* el resultado de procesar la imagen para que lo reciba el usuario. Este bloque fue desarrollado bajo el sistema operativo *Ubuntu* con el lenguaje de programación *Python*.

- El servidor de cómputo se encarga de administrar y procesar las diferentes operaciones, así como de alojar los algoritmos implementados. Este bloque recibe el mensaje de ejecución con la imagen y los parámetros que el usuario ha seleccionado, y ejecuta el algoritmo que corresponda. Una vez que ha terminado la ejecución del algoritmo, almacena la consulta del cliente y su resultado en la base de datos y devuelve la dirección de la información procesada, además del identificador del cliente que envió la consulta. Este segundo bloque ha sido desarrollado utilizando el lenguaje de programación *Python*, haciendo uso de la librería *Sklearn* [46], que es una biblioteca de aprendizaje automático de software gratuito que cuenta con varios algoritmos ya implementados y que está preparada para trabajar con las librerías numéricas y científicas *Numpy* [47] y *SciPy* [48]. Para el trabajo con imágenes normales se ha optado por el uso de la librería *OpenCV* [49], la cual es una biblioteca libre de visión artificial para el procesamiento de imágenes.
- Conviene destacar que los dos bloques anteriormente descritos se encuentran dentro de la misma máquina física, para conseguir reducir el tiempo de comunicación entre ellos, reduciendo considerablemente el tiempo de ejecución total desde que el cliente inicia una operación hasta que recibe el resultado final.

5. IMPLEMENTACIÓN Y DESARROLLO

5.1. Funciones de la aplicación

El modelo seguido para el desarrollo es un modelo iterativo-incremental en el que, después de cada iteración, se obtiene un prototipo de la versión que cumple una serie de características. A continuación, se comentan en detalle todas las funcionalidades implementadas y para qué se utilizan, así como resultados de cada una a modo de ejemplo.

Para poder utilizar la herramienta desarrollada, el usuario simplemente debe abrir la aplicación de *Telegram* e iniciar una nueva conversación con el *bot* llamado "@TFG_MBB_bot". Llegados a este punto, aparece un botón en la parte inferior de la conversación que muestra la palabra "INICIAR" (ver Figura 11 a)). Una vez pulsado ese botón, se puede ver el menú de la herramienta con las diferentes operaciones que se realizan: *Classification*, *Dimensionality Reduction*, *Superresolution* y *Unmixing*. Según la operación que el usuario seleccione, le aparecerán los algoritmos disponibles para esa operación y debe escoger uno de ellos, como se muestra en la Figura 11 c).

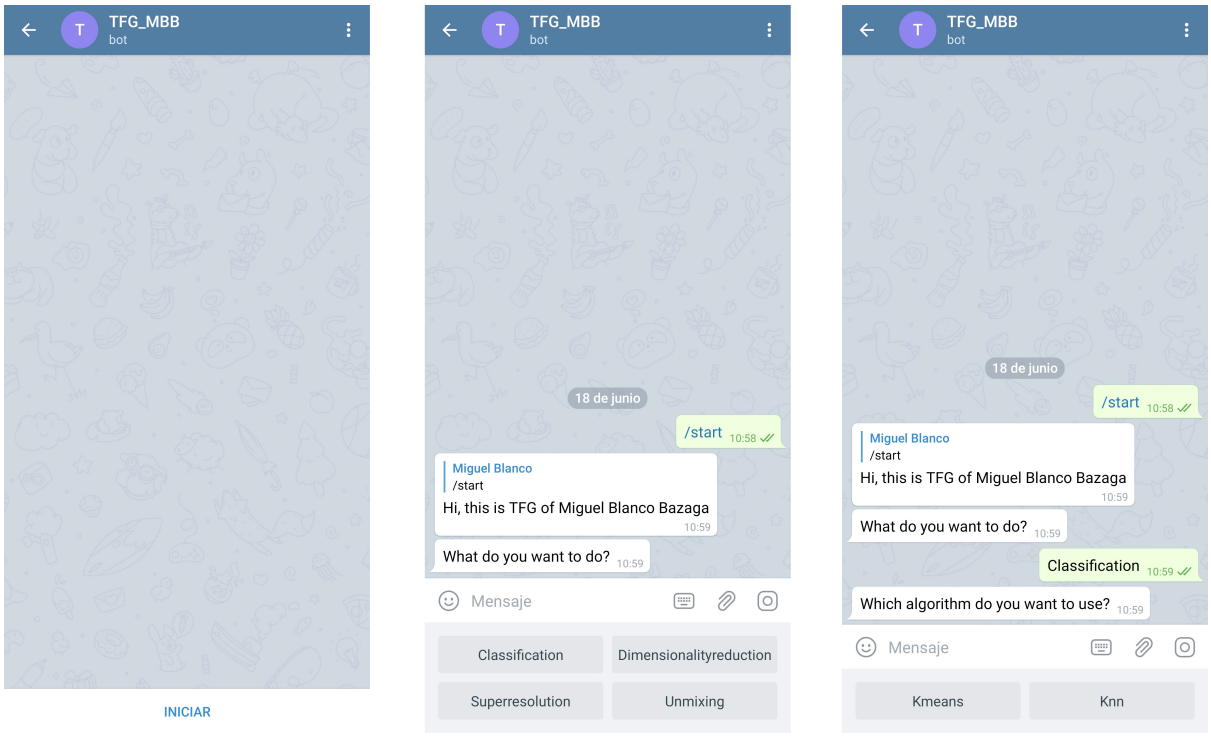


Figura 11: Manejo del *bot* de *Telegram*. (A) Inicio. (B) Algoritmos disponibles. (C) Selección de un algoritmo.

La forma de implementar el menú de *Telegram* con las diferentes opciones y algoritmos se realiza utilizando el método *KeyboardButton* de la librería *telebot* de *Python*, al que se le envía como parámetro la lista de los botones que se quieren mostrar. Este método intercambia el teclado del usuario por uno virtual, e inserta los botones de los datos recibidos en la zona del teclado, no permitiendo escribir al usuario opciones erróneas.

Una vez que el usuario ha seleccionado la operación y algoritmo deseado, le aparecerá en medio de la conversación una imagen, a modo de mosaico, de todas las imágenes que se encuentran disponibles en la aplicación para el algoritmo elegido, como se puede ver en las imágenes de la Figura 12. Para realizar este mosaico, se ha desarrollado un método que utiliza las librerías *Matplotlib* y *Pillow*. El método recibe la lista de las imágenes que inserta en un mosaico, calcula el número de filas y columnas que debe tener para que dicho mosaico sea lo más cuadrado posible, y coloca el nombre asociado encima de cada imagen.

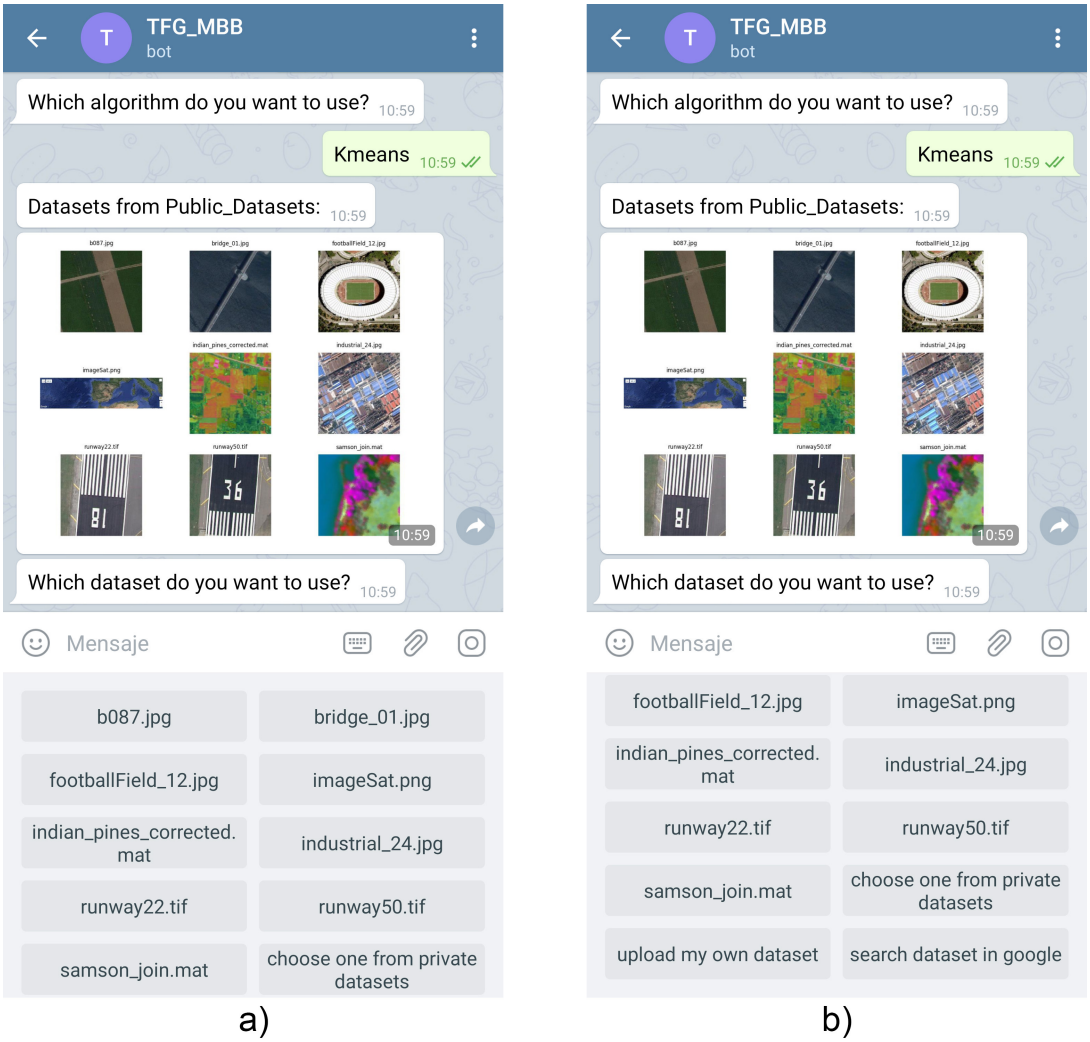


Figura 12: Capturas de pantalla mostrando el mosaico de las imágenes disponibles y las opciones que se pueden seleccionar.

En la parte inferior de las imágenes de la Figura 12 se pueden apreciar las diferentes opciones que el usuario puede seleccionar como imagen. Las primeras opciones son las imágenes que se encuentran disponibles para el algoritmo elegido. Además, se encuentran disponibles las siguientes opciones implementadas: *"choose one from private datasets"*, *"upload my own dataset"* y *"search dataset in Google"*. Estas opciones permiten al usuario utilizar como imagen alguna que otro usuario haya subido anteriormente, o bien imágenes almacenadas en su dispositivo o que pueda buscar en *Google Imágenes*, respectivamente.

Si el usuario selecciona la opción *"choose one from private datasets"*, le aparecerá un nuevo mosaico y nuevos botones correspondientes a las imágenes que se encuentran dentro de la carpeta *"private_datasets"*, como se puede ver en la Figura 13 a). El usuario deberá seleccionar una de estas imágenes mediante las diferentes posibilidades que se le muestran en forma de botones en la parte del teclado correspondiente a cada imagen. Para generar este mosaico se ha utilizado el método comentado previamente para la creación de mosaicos.

Si el usuario prefiere aportar una imagen propia, deberá seleccionar la opción de *"upload my own dataset"* y la aplicación esperará a que el usuario envíe una imagen con una de las extensiones que permite el algoritmo que haya escogido. Para esto, la herramienta le enviará al usuario un mensaje con las diferentes extensiones que el algoritmo admite, como se puede ver en la Figura 13 b). Si envía un tipo de archivo con el que el algoritmo no es capaz de trabajar, la aplicación mostrará un mensaje de error y requerirá que el usuario aporte una imagen con la extensión correcta. Si la imagen enviada es correcta, se guardará en la carpeta de *"private_datasets"* y continuará la ejecución.

Como última opción, mencionamos *"search dataset in Google"*. Si se selecciona esta opción, la aplicación muestra un mensaje pidiendo al usuario que escriba la cadena de texto que desee buscar en *Google Imágenes* y, una vez escrita y enviada dicha cadena, la herramienta realizará la búsqueda correspondiente en el servidor de *Google*, recogiendo las diez primeras imágenes del resultado y creando un nuevo mosaico que se mostrará al usuario. En dicho mosaico se coloca, encima de cada imagen, el número de la opción correspondiente a los botones del teclado en la parte inferior de la pantalla, como se puede ver en la Figura 13 c). Para desarrollar esta opción, se ha utilizado la librería *googleapiclient* a la que hay que proporcionarle una serie de credenciales para poder realizar las búsquedas asociadas.

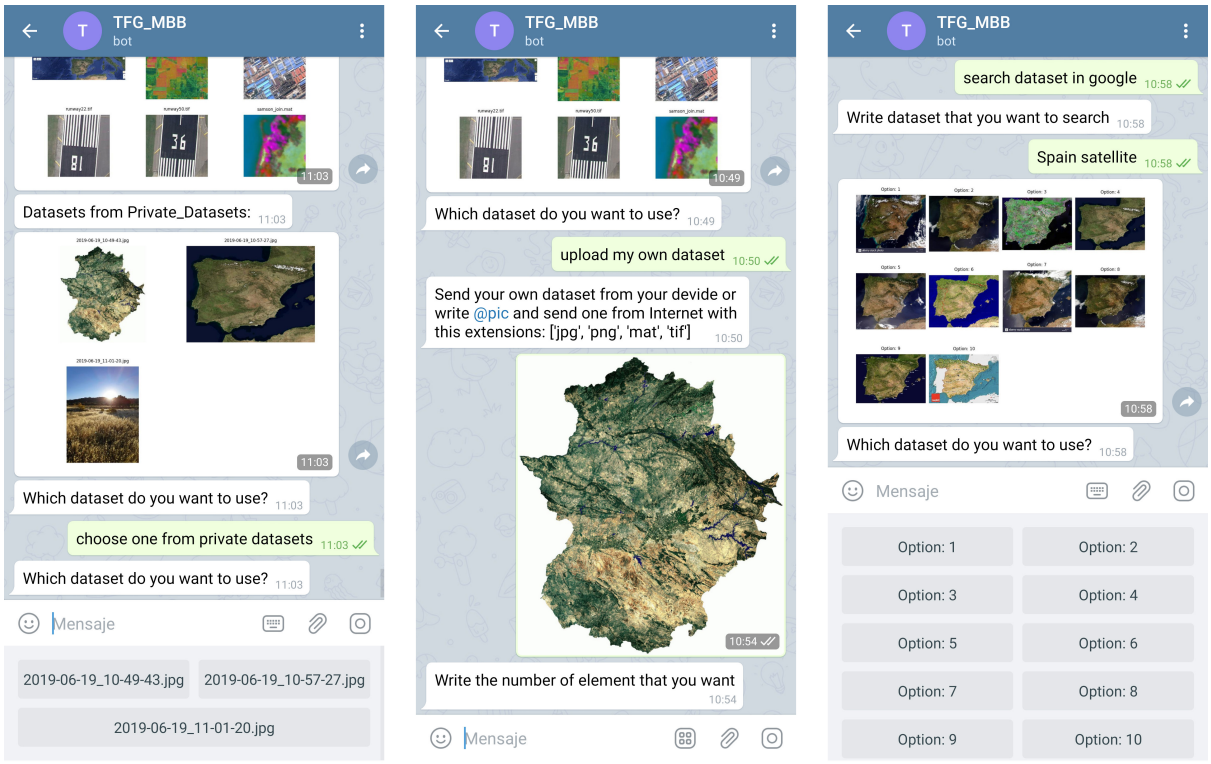
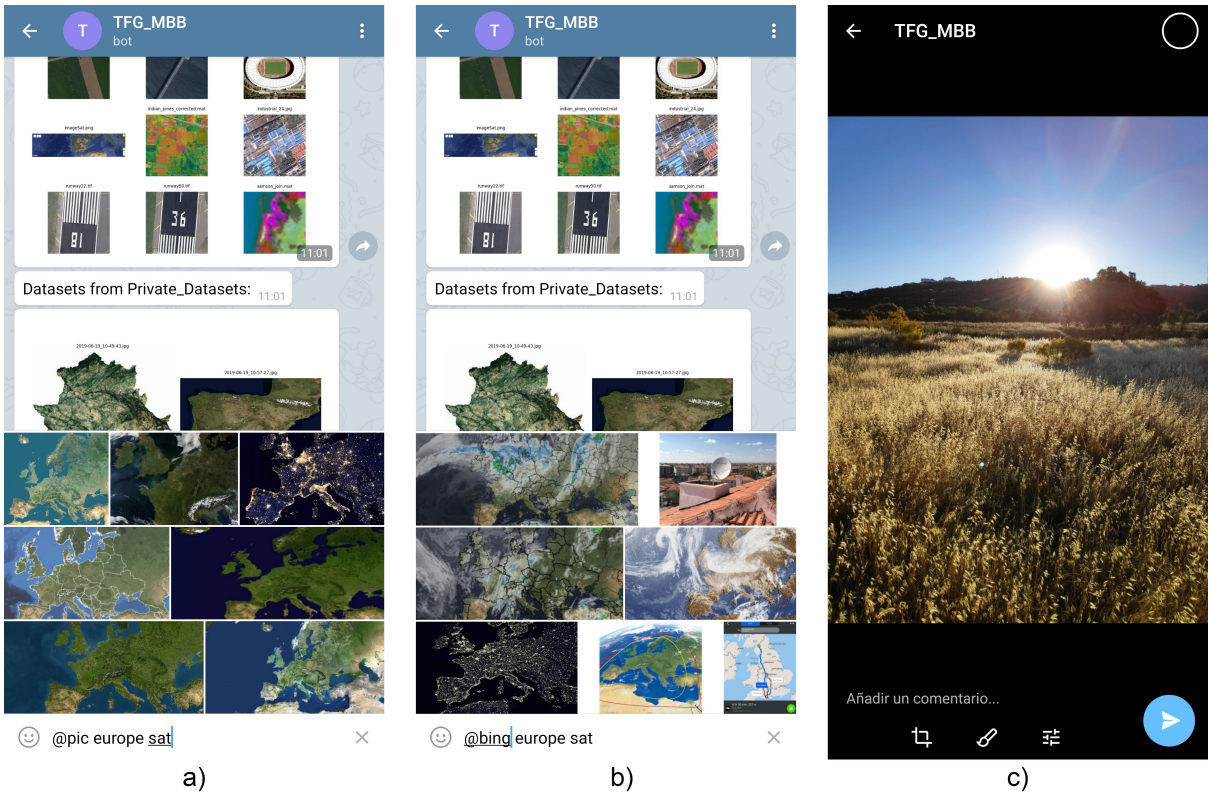


Figura 13: Capturas de pantalla mostrando un mosaico de imágenes privadas (A), la posibilidad de aportar una imagen propia y enviarla (B) y la posibilidad de buscar la imagen en el servidor de *Google Imágenes* (C).

Para utilizar una imagen propia, el usuario debe seleccionar la opción *"upload my own dataset"* y se le facilitan diversas posibilidades. La primera de ellas es hacer uso de otros *bots* de *Telegram* como *"@bing"* y *"@pic"*. Para ello, basta con escribir el nombre de cualquiera de los dos seguido del texto que se desea buscar, y en este momento aparece una ventana flotante con diferentes imágenes resultantes de la búsqueda, como se puede ver en las imágenes correspondientes en la Figura 14 a) y la Figura 14 b). El *bot* de *"@bing"* utiliza el servidor de búsqueda de imágenes de *Bing*, y el de *"@pic"* utiliza el servidor de *Yandex*. Otra posibilidad para aportar una imagen propia es abrir la cámara del *smartphone* y tomar una fotografía del lugar que se quiere procesar, como también se muestra en la Figura 14 c). Por último, también es posible enviar un archivo o imagen que se encuentre en el dispositivo como se aprecia en la Figura 13 b). Si se escoge esta opción de enviar una imagen, la aplicación de *Telegram* ofrece dos opciones: enviar la imagen con su resolución y tamaño original o bien enviar una copia de la misma aplicando previamente un filtro de reducción de tamaño. Esto implica una pérdida de resolución leve en la imagen, disminuyendo al mismo tiempo gran parte de su tamaño. Si se envía un archivo, éste debe tener como extensión una de las disponibles para el algoritmo

elegido, por lo que la aplicación muestra previamente un mensaje con las extensiones con las que el algoritmo puede trabajar.

Una vez que el usuario ha aportado una imagen diferente a las que se encuentran en la herramienta, dicha imagen se guardará dentro de la carpeta *"private_datasets"* con nombre del archivo la fecha y hora en la que se recibe, y con extensión la que tuviera originalmente el archivo aportado.



a) b) c)

Figura 14: Capturas de pantalla mostrando la búsqueda de imágenes tanto en *Bing* (A) como en *Yandex* (B), así como la posibilidad de realizar una fotografía del lugar con el *smartphone* y enviarla (C).

Llegados a este punto, puede ser que al usuario se le requiera un último aporte dependiendo del algoritmo que haya seleccionado. Por ejemplo, para el algoritmo *K-means*, es necesario el número de clústers en los que se dividirá la imagen. Para los algoritmos de *reducción de la dimensionalidad* es necesario el número de componentes a los que se quiere reducir la imagen ya que sólo se podrá reducir a un número de bandas inferior al original. Para los algoritmos de la operación de *superresolución*, se necesita el valor de la escala con la que trabajará la aplicación. Finalmente, para los algoritmos *LDA* y *VCA* de la operación de *unmixing*, es necesario el número de *endmembers* que se quiere identificar. Estas opciones se pueden apreciar en las imágenes de la Figura 15.

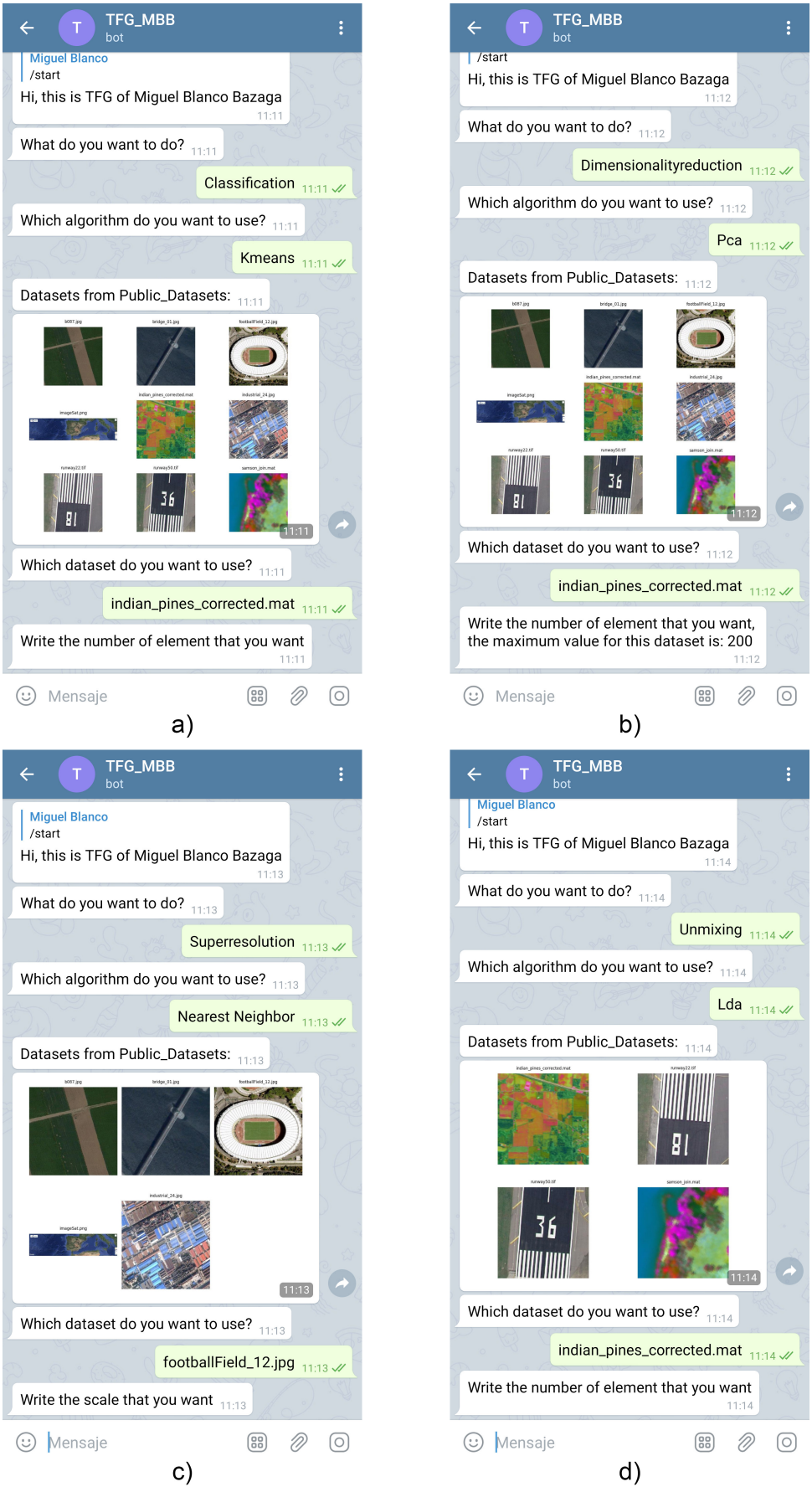


Figura 15: Capturas de pantalla en las que se muestra el último aporte que debe realizar el usuario para las diferentes operaciones disponibles en la herramienta.

Una vez que el usuario ha aportado todas las opciones que se le han requerido, se muestra un mensaje a modo de resumen de los parámetros que la herramienta ha obtenido y se le pregunta si desea continuar la ejecución con estos datos, o bien si desea modificarlos. En caso de querer modificarlos, el usuario puede añadir un *"pathlabel"* o modificar el valor de *"train percent"*. Primero se le pregunta por el *"pathlabel"* y en caso de no querer utilizar ninguno, el usuario debe escribir *"None"*. Después se le pregunta por el valor de *"train percent"* con el que quiere trabajar. Este valor se refiere al porcentaje de entrenamiento empleado por el algoritmo considerado, y deberá ser un número comprendido entre 0.01 y 0.99. Estas opciones se pueden ver con detalle en la Figura 16.

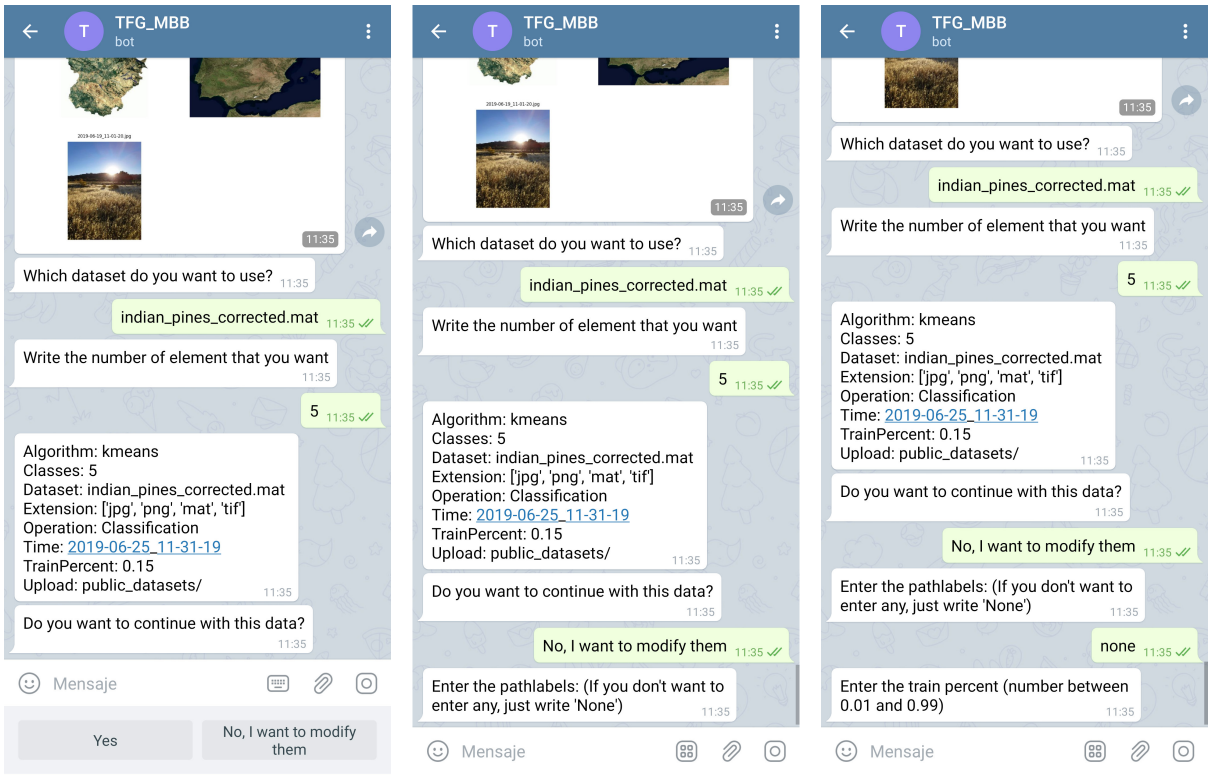


Figura 16: Capturas de pantalla en las que se muestra un mensaje a modo de resumen de la ejecución que se va a realizar y la posibilidad de modificar algunos valores.

Cuando se pulsa la opción *"Yes"* comienza la ejecución del algoritmo con los parámetros que el usuario ha seleccionado. Al cabo de unos segundos se recibe la imagen procesada, o bien un valor numérico. Dependiendo del algoritmo, dicho valor puede ser un *"Score"* (clustering), *"Overall Accuracy"* (clasificación), o bien un archivo con los mapas de abundancia (resultado de la operación de *unmixing*) si el usuario ha seleccionado el algoritmo *FCLSU*, como se puede observar en la Figura 17.

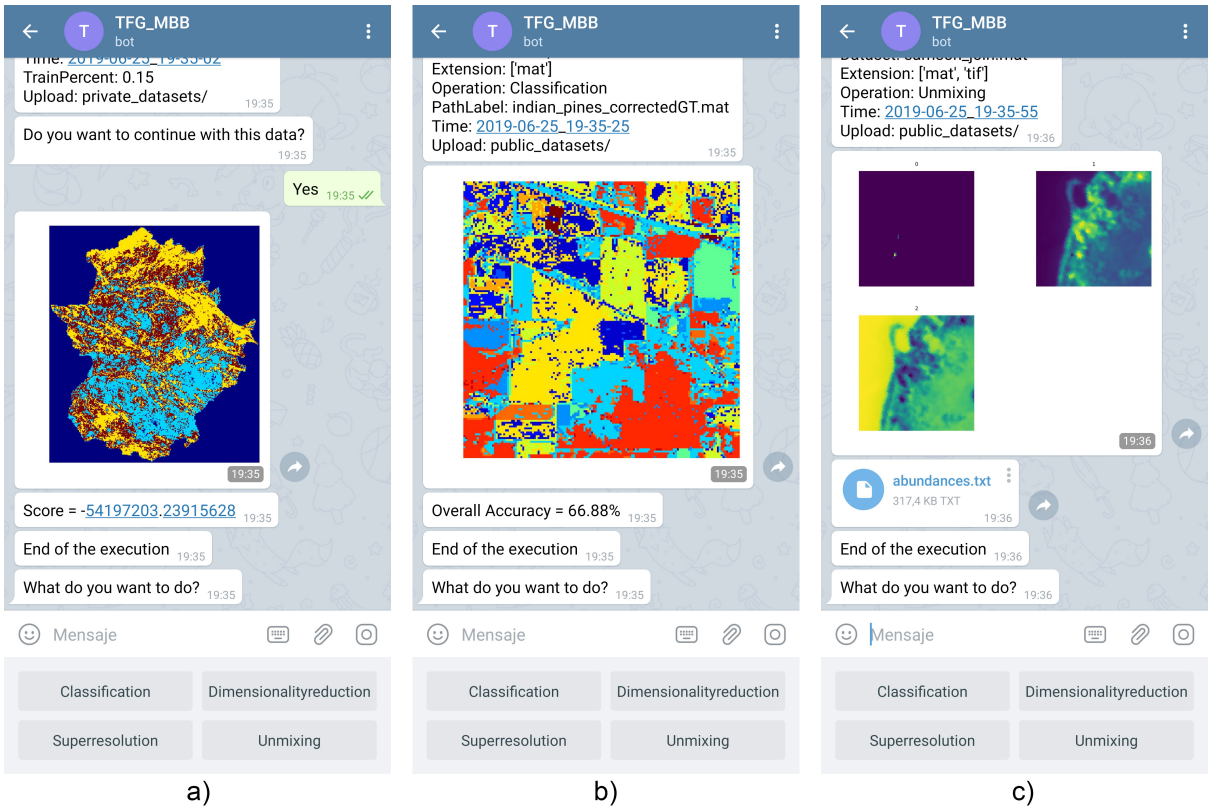


Figura 17: Capturas de pantalla en las que se muestran los resultados obtenidos tras la etapa de procesado, en los que además se recoge un valor de *"Score"* (clustering, A), *"Overall Accuracy"* (clasificación, B) o bien un archivo con los mapas de abundancia (resultado de la operación de *unmixing*) si el usuario ha seleccionado el algoritmo *FCLSU* (C).

5.2. Control de errores

Uno de los principales objetivos durante todo el desarrollo de la herramienta ha sido controlar los errores que el usuario podía cometer al seleccionar los diferentes parámetros para la ejecución. Incluso con la implementación de cambio del teclado al usuario, *Telegram* brinda la posibilidad de poder visualizar el teclado clásico. En este caso, si no recibe un mensaje correcto, envía un mensaje de error como se puede ver en la Figura 18 a). En algunas situaciones, la aplicación requiere que el usuario envíe algún mensaje, como puede ser para decidir el número de clústers. El mensaje del usuario debe ser un número entero. Si por error envía un texto o un archivo, la aplicación le enviará un mensaje solicitándole que vuelva a enviar un nuevo mensaje en el formato correcto (ver Figura 18 b)). También se controla, en el caso de que el usuario elija uno de los algoritmos de la operación de *reducción de la dimensionalidad*, que el número que envíe sea menor que el número de bandas de la imagen original, mostrando el número máximo que puede especificar como se aprecia en la Figura 18 c). Otra posibilidad

que también está contemplada en el control de errores es el caso de que el usuario quiera aportar su propia imagen. En este caso, se le muestra un mensaje con las extensiones disponibles y, en caso de no enviar un fichero con una de esas extensiones, recibirá un nuevo mensaje de error y se le pedirá que mande un archivo en el formato correcto, como se muestra en la Figura 18 d).

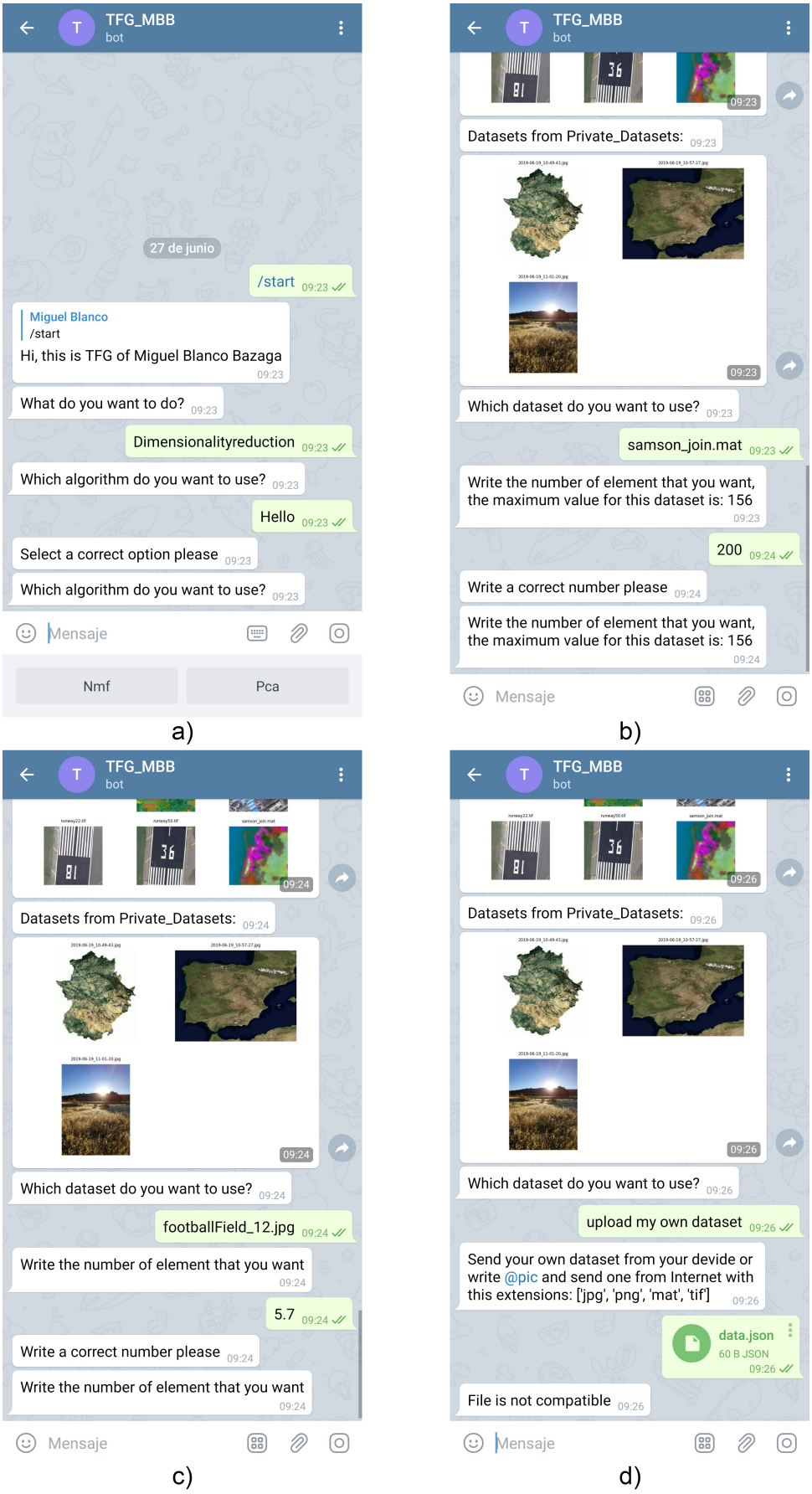


Figura 18: Manejo de diferentes tipos de errores.

5.3. Organización de los archivos

Para contemplar las diferentes posibilidades la Figura 19 muestra todos los archivos y carpetas que se encuentran en la aplicación y pasamos a detallar los más importantes:

- La carpeta *"algorithms"* que contiene los siguientes directorios:
  - *"bases"*: Tiene dos archivos: *"BaseClassifiers.py"* y *"BaseCompressors.py"* que apoyarán las implementaciones de los algoritmos de las operaciones de clasificación y reducción de la dimensionalidad.
  - *"CLASSIFICATION"*: Alberga los archivos correspondientes a los algoritmos de la operación de clasificación disponibles: *"kmeans.py"* y *"knn.py"*.
  - *"DIMENSIONALITYREDUCTION"*: Se encuentran los ficheros de *"nmf.py"* y *"pca.py"* pertenecientes a la operación de reducción de la dimensionalidad.
  - *"SUPERRESOLUTION"*: Aloja un único archivo, *"sr.py"* que alberga los tres tipos de interpolación disponibles: *bicubic*, *lanczos* y *nearest neighbor*.
  - *"UNMIXING"*: Dentro de esta carpeta están los ficheros de los algoritmos de la operación de *unmixing*: *"fclsu.py"*, *"lda.py"* y *"vca.py"*.
- Después, se puede ver la carpeta *"consults"*. A modo de ejemplo se comenta su contenido:
  - Directorio *"104386"*: *Id* de *Telegram* de un usuario que realizó una consulta. Contiene el resultado de dicha consulta dentro de una carpeta con la fecha y hora a la que la realizó.
- Del directorio *"etc"* se puede destacar el archivo *"config.json"* que contiene las claves privadas de manejo del *bot*.
- El fichero *"main_alg.py"* es el encargado de procesar los diferentes algoritmos. Para su ejecución es necesario una serie de parámetros adicionales.
- *"main.py"*: Archivo principal que hay que ejecutar para arrancar la aplicación.
- La carpeta *"private_datasets"* almacena los *datasets* que aporten los usuarios.

- La siguiente, llamada *"public_datasets"*, aloja los *datasets* que se encuentran disponibles en la herramienta desarrollada.
- *"telegram_manager.py"*: es el encargado de manejar todas las peticiones que se realizan al *bot* de *Telegram* y enviar el resultado al usuario.
- El directorio *"utils"* contiene dos archivos: *"tools.py"* que implementa algunos métodos importantes como son el de buscar en *Google* y el de listar los ficheros de una ruta y *"workimages.py"* que alberga diferentes funciones para el procesamiento de imágenes.

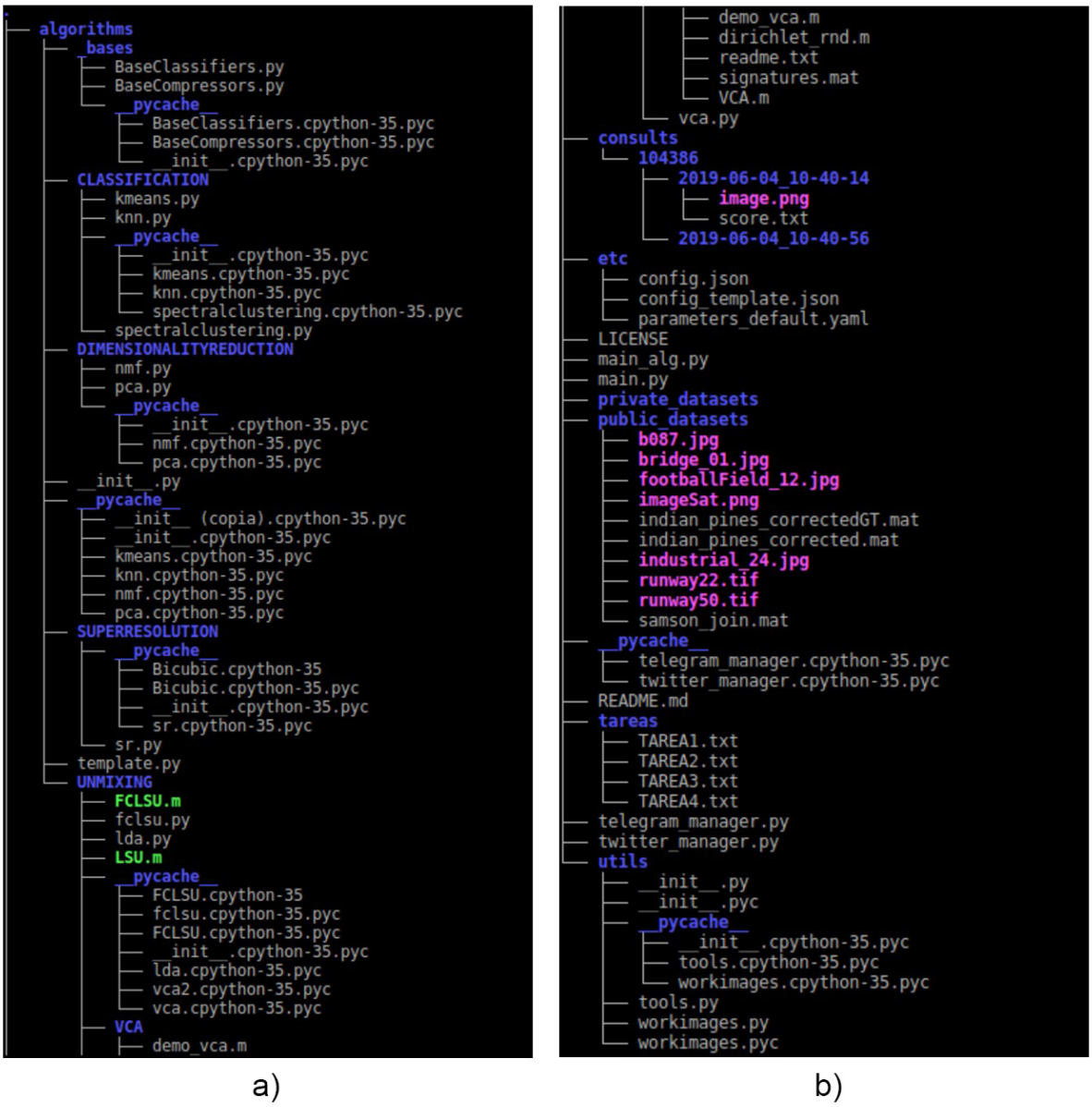
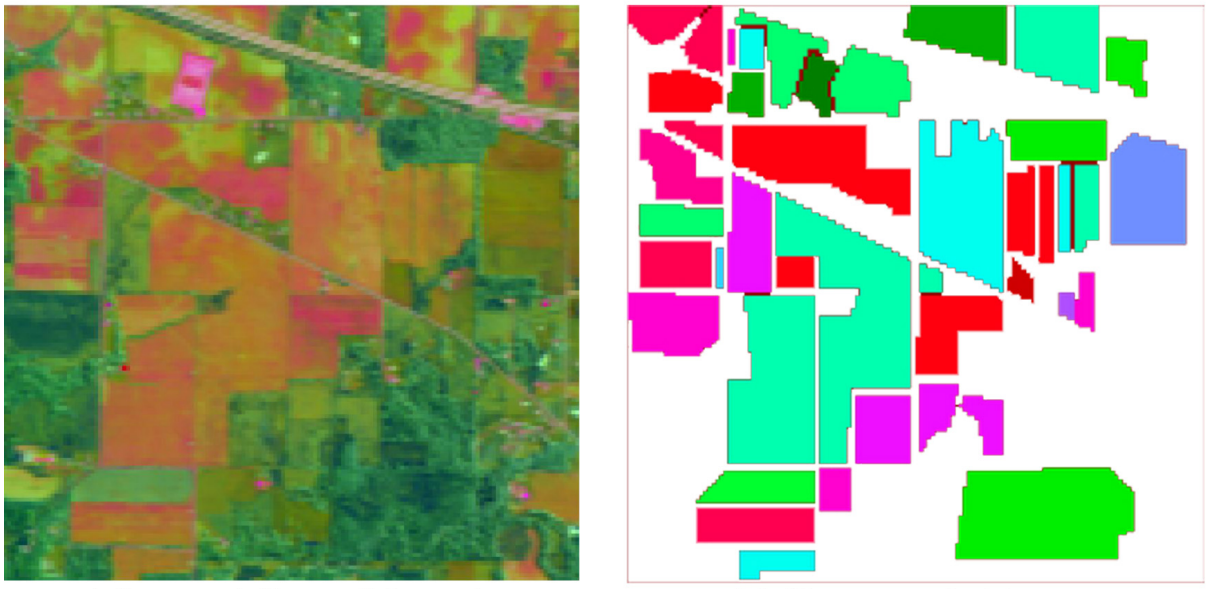


Figura 19: Organización de los ficheros que componen la herramienta desarrollada.

6. RESULTADOS

Las imágenes utilizadas durante la fase de pruebas y durante los experimentos han sido obtenidas mediante sensores hiperespectrales, descritas en la sección 3.1. Además, se han utilizado también imágenes RGB obtenidas a través de *Google Earth*. En este capítulo se exponen los resultados obtenidos durante la validación de la aplicación, que se ha llevado a cabo realizando numerosas pruebas ejecutando varios algoritmos utilizando imágenes de diferentes tipos. A continuación se describen las imágenes utilizadas.

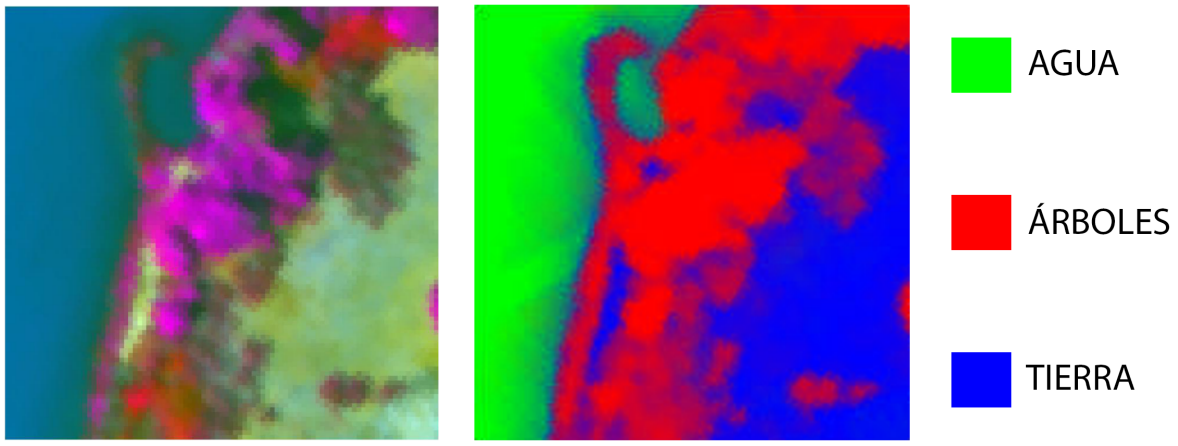
- La principal imagen hiperespectral utilizada en el estudio es la conocida imagen *Indian Pines*, que fue obtenida mediante el sensor AVIRIS [50]. La imagen cubre una zona agrícola situada al noroeste del estado de Indiana, Estados Unidos. Consta de 145x145 píxeles y 224 bandas espectrales en el rango de longitudes de onda entre 0.4 y 2.5 nm. El número de bandas ha sido reducido a 200 tras la eliminación de algunas bandas ruidosas. La imagen cubre una zona agrícola compuesta por dos tercios de cultivos, siendo el resto bosques y vegetación. Esta imagen se utiliza como un *benchmark* para validar algoritmos de clasificación, debido a la disponibilidad de una serie de clases verdad terreno (*ground truth*) que aparecen ilustradas en la Figura 20.



a) Composición en falso color (b) Clases verdad-terreno

Figura 20: Imagen AVIRIS *Indian Pines* junto con sus correspondientes clases verdad terreno (*ground truth*).

- Otra imagen hiperespectral que se utiliza en la fase de validación es la conocida como *Samson* [51]. La imagen original consta de 952x952 píxeles. Cada píxel contiene 156 bandas espectrales, que cubren un rango de longitudes de onda de 401 a 889 nm. La resolución espectral de la imagen es de 3.13 nm. La imagen procesada en este trabajo es un subconjunto de la imagen original de 95x95 píxeles. El contenido de la imagen se puede resumir en tres materiales fundamentales: tierra, árboles y agua. Esta imagen se utiliza como un *benchmark* en aplicaciones de *unmixing*, debido a la disponiblidad de la abundancia de sus tres materiales (*endmembers*) fundamentales, que aparecen representados en la Figura 21.



a) Composición en falso color b) Composición en color de los mapas de abundancia

Figura 21: Imagen *Samson* junto a las abundancias de sus materiales principales: agua, árboles y tierra.

- El resto de las imágenes que se encuentran disponibles en la aplicación son imágenes RGB basadas en tres canales, correspondientes a longitudes de onda visibles: rojo, verde y azul. Estas imágenes están tomadas directamente a partir de *Google Earth*, sistema que obtiene información a partir de un conjunto de satélites de observación remota de la Tierra de gran resolución espacial. Las imágenes seleccionadas para su inclusión en nuestra aplicación corresponden a diferentes zonas urbanas, y se muestran en la Figura 22.

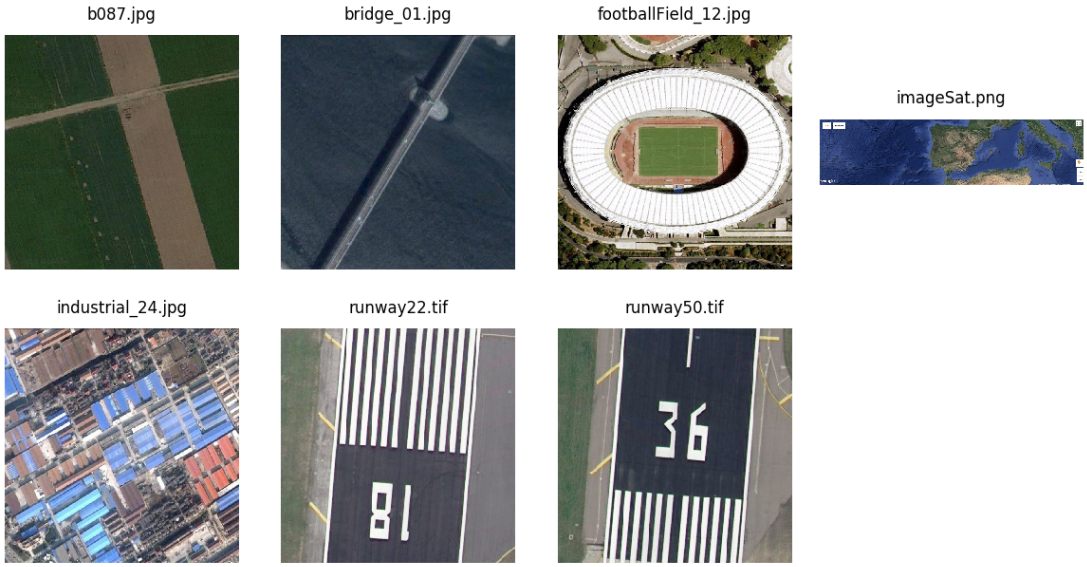


Figura 22: Imágenes de *Google Earth* que se encuentran disponibles en la aplicación.

En cuanto a la interacción del usuario con la aplicación, tal y como se ha descrito en la sección 5, se ha desarrollado una interfaz gráfica intuitiva y sencilla. El usuario, que quiere realizar una nueva consulta, abre una conversación con el *bot* de la aplicación de *Telegram* y escoge la operación y algoritmo deseado, permitiendo la posibilidad de interactuar con otros usuarios utilizando su agenda de contactos para enviar y comentar los resultados que ha obtenido. Para mostrar los resultados de los diferentes experimentos realizados para validar cada una de las operaciones disponibles, se presentan dichos resultados, así como los tiempos que tarda el usuario en recibir la información procesada, analizando la calidad de los resultados recibidos en cada operación concreta.

6.1. Evaluación de la operación de clasificación

La Tabla 1 muestra el tiempo (en segundos) que tarda el servidor de la aplicación desde que el usuario manda ejecutar una operación de clasificación hasta que recibe la imagen clasificada (se consideran diferentes imágenes). Para el algoritmo *K-means* se puede seleccionar el número de clústers en los que se agrupan los píxeles de la imagen. En nuestras pruebas, el algoritmo ha sido ejecutado con 4 y 8 clústers (ambas configuraciones ofrecen resultados satisfactorios para las imágenes consideradas) y con 3 clústers para el *dataset "samson_join.mat"* ya que se conoce el número de clases que contiene. Por otra parte, el algoritmo *KNN* se ha ejecutado con un

total de 16 vecinos. Este algoritmo solamente se ha podido aplicar a la imagen AVIRIS *Indian Pines*, ya que se trata de un algoritmo supervisado que necesita información verdad terreno (en este caso, se han utilizado las 16 clases verdad-terreno de la imagen *Indian Pines* para realizar la parametrización del algoritmo, mientras que el resto de imágenes RGB consideradas no disponen de información verdad-terreno).

Dataset	*K-means*			*KNN*
	3 clústers	4 clústers	8 clústers	16 vecinos
bridge_01.jpg	——	2.57355	2.70276	——
footballField_12.jpg	——	2.49803	2.67049	——
indian_pines.mat	——	2.15107	2.24758	12.11654
samson_join.mat	1.85213	——	——	——

Tabla 1: Tiempo (en segundos) que tarda el servidor de la aplicación desde que el usuario manda ejecutar una operación de clasificación hasta que el usuario recibe la imagen clasificada (se consideran diferentes imágenes).

La Figura 23 muestra los resultados obtenidos al procesar algunos de los *datasets* que se encuentran en la aplicación con diferentes configuraciones, como se ve en la Tabla 1. Las dos primeras filas corresponden a imágenes RGB (*"bridge_01.jpg"* y *"footballField_12.jpg"*) y las dos últimas a imágenes hiperespectrales (*"indian_pines.mat"* y *"samson_join.mat"*). En la figura se aprecia que los resultados de clasificación son particularmente precisos para la imagen hiperespectral *AVIRIS Indian Pines*, en la que el resultado de clasificación del algoritmo *KNN* es muy similar a la verdad terreno mostrada en la Figura 20. Incluso para el algoritmo *K-means* con un número de clústers reducido se obtiene un resultado que es visualmente consistente con las clases verdad terreno mostradas en la Figura 20. Para la imagen *Samson*, los resultados obtenidos son bastante precisos ya que el algoritmo tuvo que trabajar con un número de clústers igual a la cantidad de clases que se encuentran en la imagen (3). Finalmente, los resultados de clasificación de imágenes RGB parecen adecuados en función de los objetos identificados (carretera, estadio, etc.), aunque no se han podido validar de forma precisa debido a la falta de información verdad terreno para dichas imágenes.

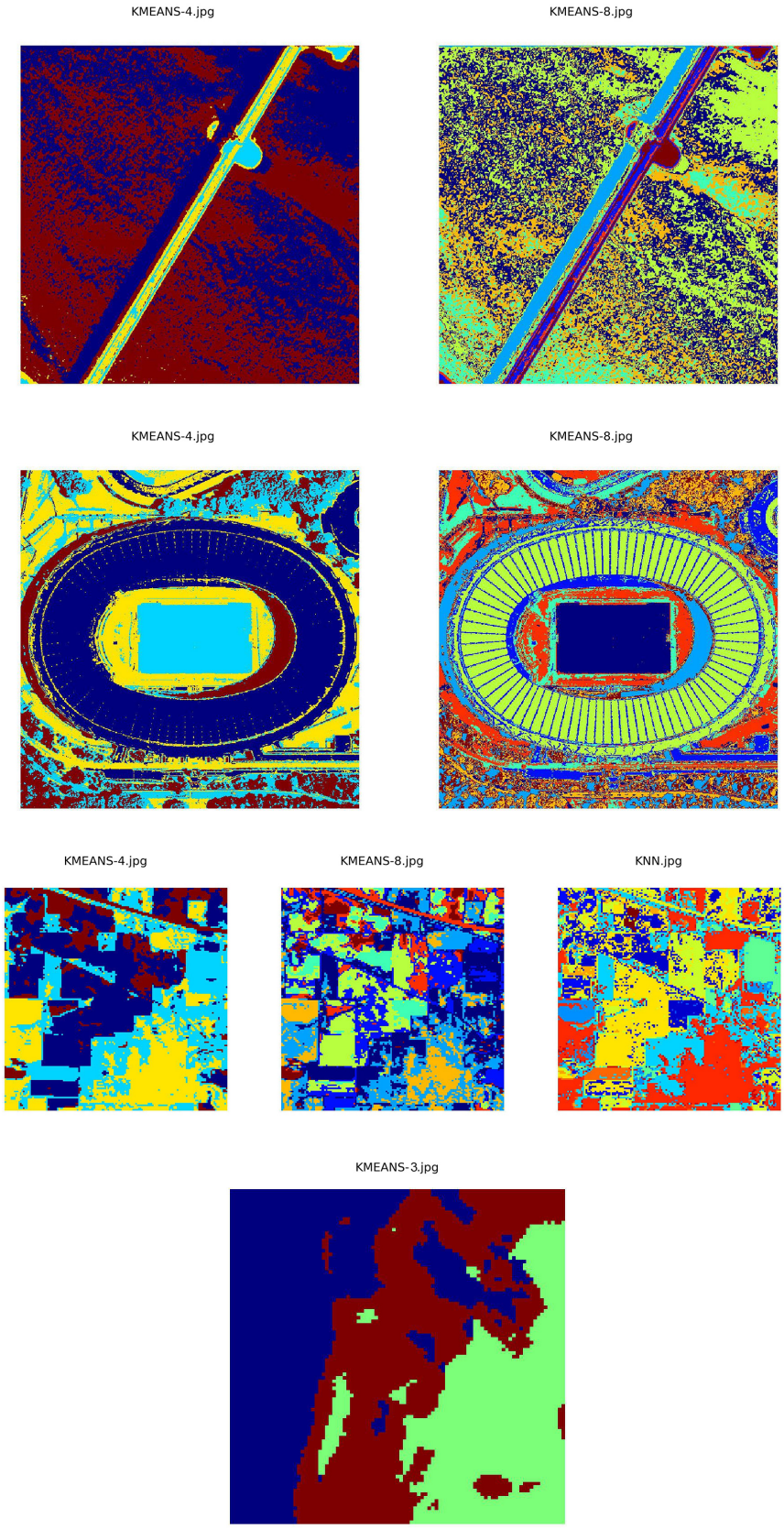


Figura 23: Representación de los resultados obtenidos con diferentes configuraciones e imágenes para la operación de *clasificación*.

6.2. Evaluación de la operación de reducción de la dimensionalidad

La Tabla 2 muestra el tiempo que tarda el servidor de la aplicación desde que el usuario manda ejecutar una operación de reducción de dimensionalidad hasta que el usuario recibe la imagen dimensionalmente reducida. En esta operación, el usuario puede seleccionar el número de componentes a los que reducir la imagen. Se han utilizado los algoritmos *NMF* y *PCA*, ejecutándolos para reducir varias imágenes a 2, 8 y 16 componentes. Para las imágenes RGB, al tener solamente 3 bandas, solamente se ha reducido la dimensionalidad a 2, de ahí que aparezcan algunos datos vacíos en la tabla para estas imágenes que solamente poseen sólo poseen tres canales. Las imágenes hiperespectrales, en cambio, han sido reducidas a 2, 8 y 16 componentes utilizando los dos métodos.

Dataset	NMF			PCA		
	2	8	16	2	8	16
bridge_01.jpg	4.19410	——	——	3.52428	——	——
footballField_12.jpg	3.99486	——	——	3.76978	——	——
indian_pines.mat	2.81313	3.84838	5.57568	2.49139	3.26099	4.18566
samson_join.mat	1.12778	3.00976	3.98327	1.10167	2.75171	3.06395

Tabla 2: Tiempo que tarda el servidor de la aplicación desde que el usuario manda ejecutar una operación de reducción de dimensionalidad hasta que el usuario recibe la imagen dimensionalmente reducida (se consideran diferentes imágenes).

Por su parte, la Figura 24 muestra las componentes resultantes tras el proceso de reducción dimensional de las imágenes consideradas, ofreciendo una representación visual del proceso de reducción dimensional mostrado en la Tabla 2. Las dos primeras filas corresponden a las imágenes RGB (*"bridge_01.jpg"* y *"footballField_12.jpg"*) y las dos últimas filas corresponden a las imágenes hiperespectrales (*"indian_pines.mat"* y *"samson_join.mat"*). En todos los casos, puede apreciarse que los componentes obtenidos tras el proceso de reducción dimensional son representativos de diferentes características presentes en los datos, siendo los resultados obtenidos por PCA similares a los obtenidos por NMF tanto para las imágenes RGB como para las imágenes hiperespectrales.

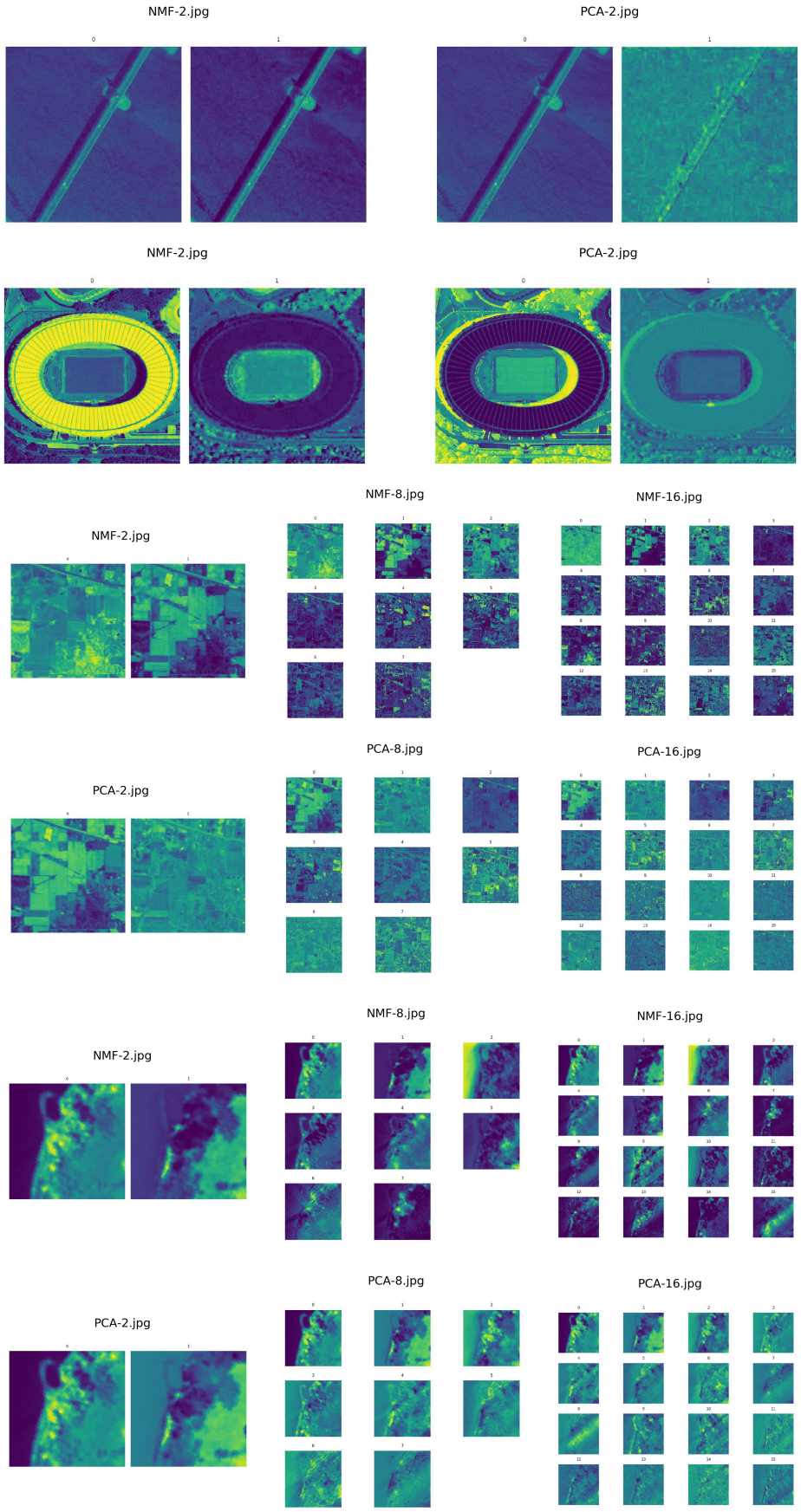


Figura 24: Representación de los resultados obtenidos con diferentes configuraciones e imágenes para la operación de *reducción de la dimensionalidad*.

6.3. Resultados de la operación de superresolución

La Tabla 3 muestra el tiempo que tarda el servidor de la aplicación desde que el usuario manda ejecutar una operación de superresolución hasta que el usuario recibe la imagen súper-resuelta. En esta operación, el usuario puede seleccionar la escala con la que trabajarán los algoritmos de interpolación. En nuestro experimento, todos los algoritmos de interpolación considerados (bicúbica, Lanczos y *Nearest Neighbor*) han sido ejecutados con escalas de 4 y 10.

Dataset	Bicúbica		Lanczos		*Nearest Neighbor*	
	4	10	4	10	4	10
brige_01.jpg	2.54922	2.76737	2.87967	2.62517	2.17528	1.88262
footballField_12.jpg	2.67371	2.73667	2.95494	2.74317	2.17389	1.92003
imageSat.png	2.62077	2.69256	2.46836	2.67896	1.96441	1.88118

Tabla 3: Tiempo que tarda el servidor de la aplicación desde que el usuario manda ejecutar una operación de superresolución hasta que el usuario recibe la imagen súper-resuelta (se consideran diferentes imágenes).

La Figura 25 muestra los resultados obtenidos al aplicar operaciones de superresolución sobre las imágenes RGB que se encuentran en la aplicación con diferentes configuraciones, ilustrando visualmente los resultados mostrados en la Tabla 2. En concreto, las imágenes con las que se ha trabajado para este resultado han sido imágenes RGB (*"bridge_01.jpg"*,*"footballField_12.jpg"* y *"imageSat.png"*). En este caso, no se han utilizado las imágenes hiperespectrales puesto que se ha observado que la operación de superresolución no contribuye a mejorar la resolución de dichas imágenes (para este tipo de operaciones con imágenes hiperespectrales, resulta más apropiado utilizar técnicas de *unmixing*, que serán evaluadas en el siguiente subapartado). Para las imágenes RGB consideradas, en general se observa que la escala de 4 resulta más apropiada que la escala de 10, ya que esta última escala introduce un efecto de difuminado en las imágenes que no resulta apropiado en el contexto de superresolución considerado.

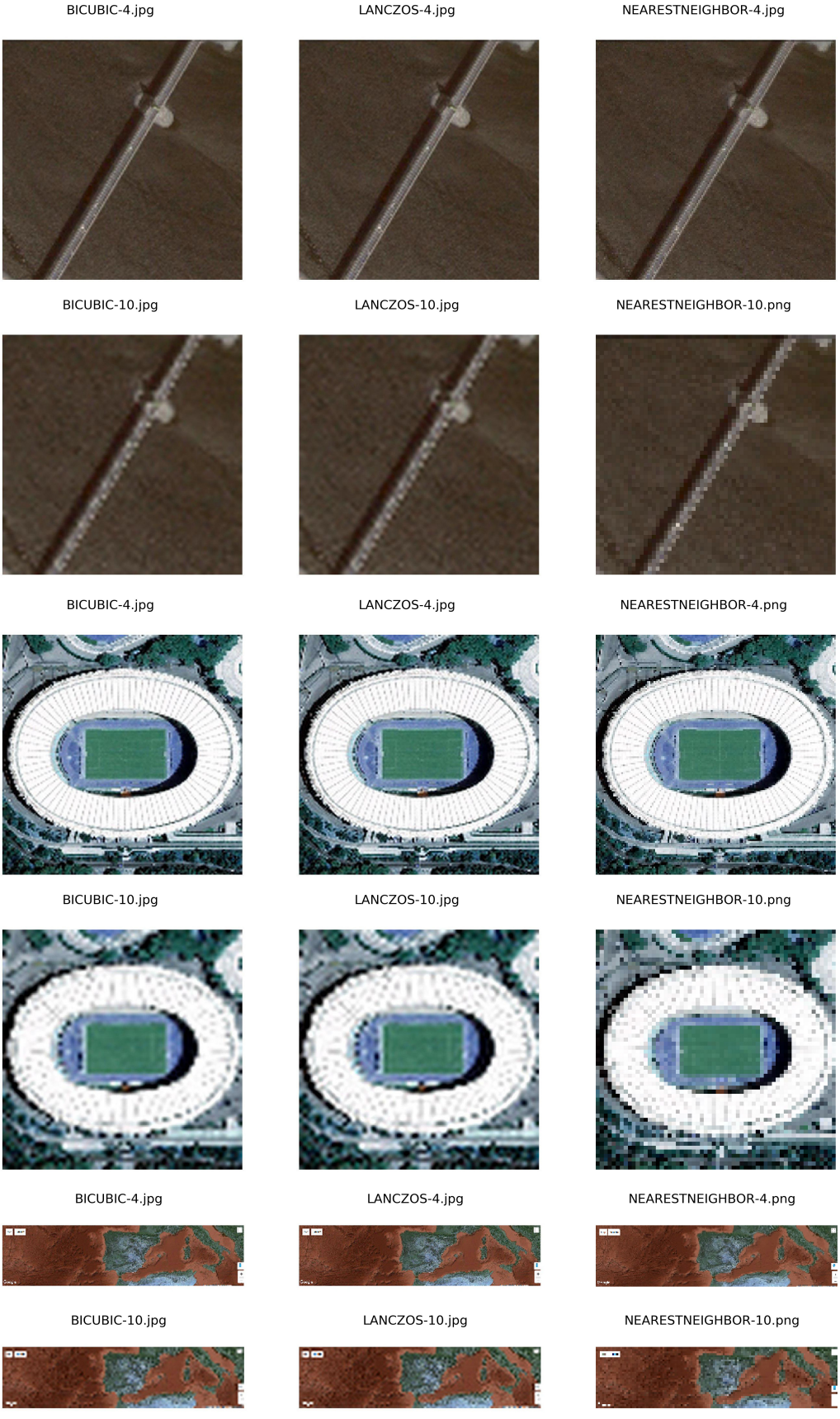


Figura 25: Representación de los resultados obtenidos con diferentes configuraciones e imágenes para la operación de *superresolución*.

6.4. Resultados de la operación de unmixing

La Tabla 4 muestra el tiempo que tarda el servidor de la aplicación desde que el usuario manda ejecutar una operación de *unmixing* hasta que el usuario recibe los mapas de abundancia resultantes de dicha operación. En esta operación el usuario puede seleccionar el número de *endmembers* a extraer. En concreto, se han utilizado los algoritmos *LDA* y *VCA* para extraer 4 y 8 *endmembers* a partir de las dos imágenes hiperespectrales consideradas en el estudio (*Indian Pines* y *Samson*). A partir de los *endmembers* extraídos, se ha utilizado el algoritmo FCLSU para obtener los mapas de abundancia.

Dataset	FCLSU	LDA		VCA	
		8	16	8	16
indian_pines.mat	——	20.66974	24.17927	1.10276	1.47612
samson_join.mat	3.02356	7.72152	8.16368	0.74007	1.03460

Tabla 4: Resultados con diferentes valores para la operación de *unmixing*. Tiempos en segundos.

La Figura 26 muestra los mapas de abundancia obtenidos al aplicar la operación de *unmixing* sobre las dos imágenes hiperespectrales consideradas, con la configuración mostrada en la Tabla 4. De nuevo, insistimos que las imágenes con las que se ha trabajado en este experimento han sido únicamente imágenes hiperespectrales de múltiples bandas (*"indian_pines.mat"*,*"samson_join.mat"*), ya que no tiene sentido realizar la operación de *unmixing* sobre imágenes RGB al tener que ser el número de bandas de la imagen superior al número de *endmembers* presentes en la misma. Los resultados mostrados en la Figura 26 muestran que los mapas de abundancia obtenidos en las diferentes configuraciones son representativos de los diferentes materiales presentes en las imágenes consideradas, por lo que dichos mapas se consideran cualitativamente adecuados en función de la información de referencia disponible para dichas imágenes.

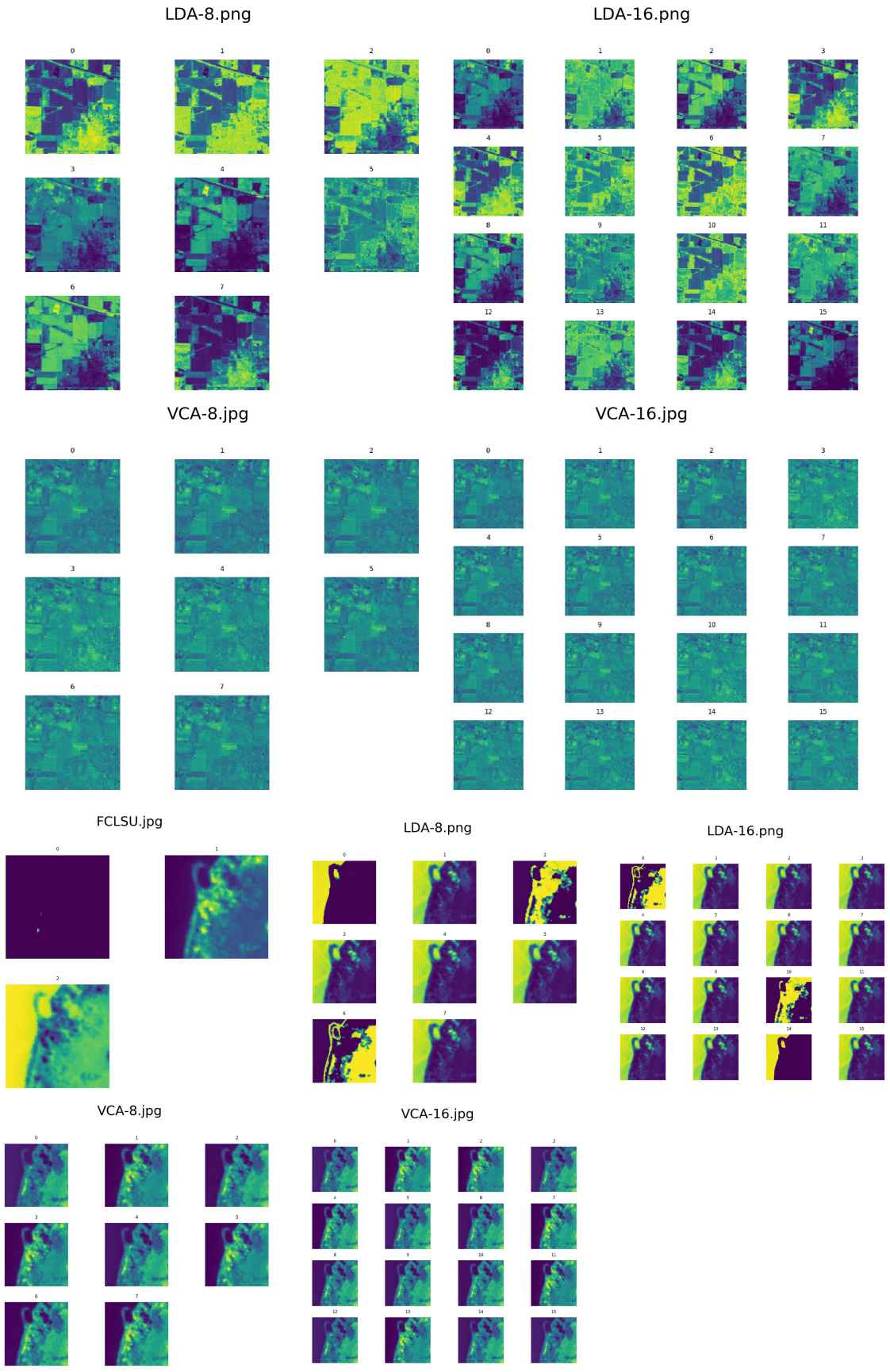


Figura 26: Representación de los resultados obtenidos con diferentes configuraciones e imágenes para la operación de *unmixing*.

7. CONCLUSIONES Y TRABAJO FUTURO

El objetivo principal de este trabajo ha sido explorar la posibilidad de desarrollado una nueva aplicación capaz de integrar técnicas de procesamiento de datos obtenidos de redes sociales con datos de satélites de observación remota de la Tierra, con el objetivo de apoyar en la toma de decisiones ante eventos críticos. Un aspecto primordial que se ha planteado como objetivo al desarrollar la aplicación ha sido reducir al máximo el tiempo que transcurre desde que se produce un evento hasta que la aplicación ofrece un resultado en forma de clasificación, reducción dimensional, *unmixing* o superresolución. En concreto, la aplicación desarrollada se basa en mensajería instantánea *Telegram* para procesar imágenes de diferentes tipos (RGB e hiperespectrales), realizando una primera aproximación hacia la integración de datos de redes sociales con datos de observación remota de la Tierra.

La aplicación desarrollada permite la extracción avanzada de información a partir de las imágenes consideradas, permitiendo aplicar diferentes algoritmos representativos disponibles en la literatura sobre imágenes que ya se encuentran disponibles en la aplicación, y también sobre imágenes capturadas por el *smartphone* o bien extraídas a partir de grandes repositorios de datos como *Google Earth*. Como resultado de ejecutar un algoritmo (de clasificación, reducción dimensional, *unmixing* o superresolución), la aplicación devuelve al usuario el resultado o imagen procesada, además de un valor numérico dependiendo de la operación seleccionada.

En este sentido, una contribución importante del presente trabajo es integrar la información procedente de dos fuentes de datos muy populares: redes sociales y teledetección, pero que nunca se habían explotado de forma conjunta mediante una aplicación de mensajería instantánea. A partir de ahora, cualquier usuario puede ejecutar operaciones avanzadas de procesamiento de imágenes mediante un sencillo *bot*, sin necesidad de instalar una aplicación específica en su teléfono móvil.

Como trabajo futuro se planea continuar mejorando este sistema añadiendo funcionalidades y nuevas características. Una de las posibles mejoras sería incluir los resultados obtenidos a través de una cuenta de la red social *Twitter* o a través de una funcionalidad de la propia aplicación *Telegram*, el uso de canales, lo que permitiría comprobar si la consulta que se quiere ejecutar ya habría sido antes ejecutada, dando incluso la posibilidad al usuario de que su consulta no se almacene en la base de datos que contiene la aplicación de consultas realizadas,

consiguiendo así mayor privacidad.

Otra mejora intuitiva sería añadir nuevas operaciones y algoritmos de procesamiento. También se pueden añadir nuevos motores de búsqueda de imágenes. Actualmente se encuentran implementados los de *Google*, *Yandex* y *Bing* consiguiendo así mayores facilidades para el usuario a la hora de extraer imágenes.

También destacamos que el uso de *Telegram* es habitual en Europa y en América. En el mercado asiático se suele utilizar otra aplicación de mensajería llamada *WeChat*, que alberga multitud de funcionalidades y que cuenta con cientos de millones de usuarios en China, donde la práctica totalidad de la población utiliza dicha aplicación de mensajería instantánea. Una funcionalidad muy interesante de *WeChat* es la de crear mini-programas dentro de la aplicación, por lo que consideramos que la herramienta desarrollada podría adaptarse de forma sencilla al mercado asiático consiguiendo así un gran volumen de usuarios [52].

Finalmente, es importante mencionar que la arquitectura del sistema permite emplear *hardware* de alto rendimiento (por ejemplo, la GPU incorporada en el *smartphone*) para procesar de forma eficiente imágenes de gran dimensionalidad (como las imágenes hiperespectrales), reduciendo así notablemente los tiempos de respuesta.

Referencias

[1] DataReportal, "Digital 2019: Global digital overview," *Recuperado de: https://datareportal.com/reports/digital-2019-global-digital-overview*, 2019.

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

[3] J. B. Campbell and R. H. Wynne, *Introduction to remote sensing*. Guilford Press, 2011.

[4] E. Estellés-Arolas and F. González-Ladrón-De-Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information science*, vol. 38, no. 2, pp. 189–200, 2012.

[5] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International journal of information management*, vol. 35, no. 2, pp. 137–144, 2015.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[7] E. Adam, O. Mutanga, and D. Rugege, "Multispectral and hyperspectral remote sensing for identification and mapping of wetland vegetation: a review," *Wetlands Ecology and Management*, vol. 18, no. 3, pp. 281–296, 2010.

[8] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, 2016.

[9] K. Erdle, B. Mistele, and U. Schmidhalter, "Comparison of active and passive spectral sensors in discriminating biomass parameters and nitrogen status in wheat cultivars," *Field Crops Research*, vol. 124, no. 1, pp. 74–84, 2011.

[10] A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 528–544, 2011.

[11] 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 Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 508–527, 2011.

[12] A. Hussain and E. Cambria, "Semi-supervised learning for big social data analysis," *Neurocomputing*, vol. 275, pp. 1662–1673, 2018.

[13] J. Li, J. A. Benediktsson, B. Zhang, T. Yang, and A. Plaza, "Spatial technology and social media in remote sensing: challenges and opportunities [point of view]," *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1583–1585, 2017.

[14] Q. Du and H. Yang, "Similarity-based unsupervised band selection for hyperspectral image analysis.," *IEEE Geosci. Remote Sensing Lett.*, vol. 5, no. 4, pp. 564–568, 2008.

[15] W. R. Johnson, D. W. Wilson, and G. Bearman, "All-reflective snapshot hyperspectral imager for ultraviolet and infrared applications," *Optics letters*, vol. 30, no. 12, pp. 1464–1466, 2005.

[16] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote sensing of environment*, vol. 113, pp. S110–S122, 2009.

[17] A. Remon, S. Sanchez, S. Bernabé, E. S. Quintana-Orti, and A. Plaza, "Performance versus energy consumption of hyperspectral unmixing algorithms on multi-core platforms," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, 12 2013.

[18] G. Foody and D. Cox, "Sub-pixel land cover composition estimation using a linear mixture model and fuzzy membership functions," *Remote sensing*, vol. 15, no. 3, pp. 619–631, 1994.

[19] M. T. Eismann, *Hyperspectral remote sensing*. SPIE Bellingham, 2012.

[20] R. N. Clark and T. L. Roush, "Reflectance spectroscopy: Quantitative analysis techniques for remote sensing applications," *Journal of Geophysical Research: Solid Earth*, vol. 89, no. B7, pp. 6329–6340, 1984.

[21] R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis, *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (aviris)," *Remote sensing of environment*, vol. 65, no. 3, pp. 227–248, 1998.

[22] J. S. Pearlman, P. S. Barry, C. C. Segal, J. Shepanski, D. Beiso, and S. L. Carman, "Hyperion, a space-based imaging spectrometer," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 6, pp. 1160–1173, 2003.

[23] S. G. Ungar, J. S. Pearlman, J. A. Mendenhall, and D. Reuter, "Overview of the earth observing one (eo-1) mission," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 6, pp. 1149–1159, 2003.

[24] C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification*, vol. 1. Springer Science & Business Media, 2003.

[25] C.-I. Chang, *Hyperspectral data exploitation: theory and applications*. John Wiley & Sons, 2007.

[26] B. Mirkin, *Mathematical classification and clustering*, vol. 11. Springer Science & Business Media, 2013.

[27] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[28] T. M. Cover, P. E. Hart, *et al.*, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[29] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Transactions on geoscience and remote sensing*, vol. 32, no. 4, pp. 779–785, 1994.

[30] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[31] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular value decomposition and principal component analysis," in *A practical approach to microarray data analysis*, pp. 91–109, Springer, 2003.

[32] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, pp. 556–562, 2001.

[33] 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 Transactions on Geoscience and Remote Sensing*, vol. 56, no. 11, pp. 6792–6810, 2018.

[34] K. Dawson-Howe, *A practical introduction to computer vision with opencv*. John Wiley & Sons, 2014.

[35] G. J. Grevera and J. K. Udupa, "Shape-based interpolation of multidimensional grey-level images," *IEEE transactions on medical imaging*, vol. 15, no. 6, pp. 881–892, 1996.

[36] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE journal of selected topics in applied earth observations and remote sensing*, vol. 5, no. 2, pp. 354–379, 2012.

[37] L. C. Parra, C. Spence, P. Sajda, A. Ziehe, and K.-R. Müller, "Unmixing hyperspectral data," in *Advances in neural information processing systems*, pp. 942–948, 2000.

[38] D. C. Heinz *et al.*, "Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery," *IEEE transactions on geoscience and remote sensing*, vol. 39, no. 3, pp. 529–545, 2001.

[39] Q. Du and C.-I. Chang, "A linear constrained distance-based discriminant analysis for hyperspectral image classification," *Pattern Recognition*, vol. 34, no. 2, pp. 361–373, 2001.

[40] J. M. Nascimento and J. M. Dias, "Vertex component analysis: A fast algorithm to unmix hyperspectral data," *IEEE transactions on Geoscience and Remote Sensing*, vol. 43, no. 4, pp. 898–910, 2005.

[41] C. Negus and F. Caen, *Ubuntu Linux toolbox: 1000+ commands for Ubuntu and Debian power users*. John Wiley & Sons, 2008.

[42] T. E. Oliphant, “Python for scientific computing,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.

[43] J. Jakobsen and C. Orlandi, “On the cca (in) security of mtproto.,” in *SPSM@ CCS*, pp. 113–116, 2016.

[44] Z. Cataldi, *Una metodología para el diseño, desarrollo y evaluación de software educativo*. PhD thesis, Facultad de Informática, 2000.

[45] M. Steiner, G. Tsudik, and M. Waidner, “Diffie-hellman key distribution extended to group communication,” in *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, CCS ’96, (New York, NY, USA), pp. 31–37, ACM, 1996.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[47] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, “The numpy array: a structure for efficient numerical computation,” *Computing in Science & Engineering*, vol. 13, no. 2, p. 22, 2011.

[48] E. Jones, T. Oliphant, P. Peterson, *et al.*, “SciPy: Open source scientific tools for Python,” 2001–. [Online; accessed ¡today¿].

[49] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. .ºReilly Media, Inc.”, 2008.

[50] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, “220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3,” Sep 2015.

[51] F. Zhu, Y. Wang, S. Xiang, B. Fan, and C. Pan, “Structured sparse method for hyperspectral unmixing,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 88, pp. 101–118, 2014.

[52] E. Harwit, “Wechat: Social and political development of china’s dominant messaging app,” *Chinese Journal of Communication*, vol. 10, no. 3, pp. 312–327, 2017.