




Proposition of a Parallel and Distributed Algorithm for the Dimensionality Reduction with Apache Spark

Abdelali Zbakh¹ , Zoubida Alaoui Mdaghri¹,
Mourad El Yadari², Abdelillah Benyoussef¹, and Abdellah El Kenz¹

¹ Faculty of Sciences, University Mohammed V, Rabat, Morocco
zbakhabdou@gmail.com, zoubidaalaouimdaghri@gmail.com,
benyous.a@gmail.com, akenzele@yahoo.com
² University Moulay Ismail, Meknes, Morocco
mouradelyadari@gmail.com

Abstract. In recent years, the field of storage and data processing has known a radical evolution, because of the large mass of data generated every minute. As a result, traditional tools and algorithms have become incapable of following this exponential evolution and yielding results within a reasonable time. Among the solutions that can be adopted to solve this problem, is the use of distributed data storage and parallel processing. In our work we used the distributed platform Spark, and a massive data set called hyperspectral image. Indeed, a hyperspectral image processing, such as visualization and feature extraction, has to deal with the large dimensionality of the image. Several dimensions reduction techniques exist in the literature. In this paper, we proposed a distributed and parallel version of Principal Component Analysis (PCA).

Keywords: Distributed PCA · BIG DATA · Spark platform · Map-Reduce Dimension reduction · Hyperspectral data

1 Introduction

The data collected today by the sensors increases rapidly and especially the hyperspectral data, which allow to give more physical information on the observed area.

The hyperspectral image is an image that represents the same scene following the hundreds of contiguous spectral bands in various wavelength ranges. The data of a hyperspectral image are organized in the form of a cube of three dimensions: Two dimensions denoted x and y represent the spatial dimensions and a spectral dimension denoted z (see Fig. 1) [1].

It will be noted that there are multispectral images composed of a dozen bands, while the hyperspectral image exceeds a hundred bands, which implies a significant requirement in terms of data processing and storage.

Unlike the classic color image, the hyperspectral image gives more physical information about each observed object of the scene. Thus the technique of

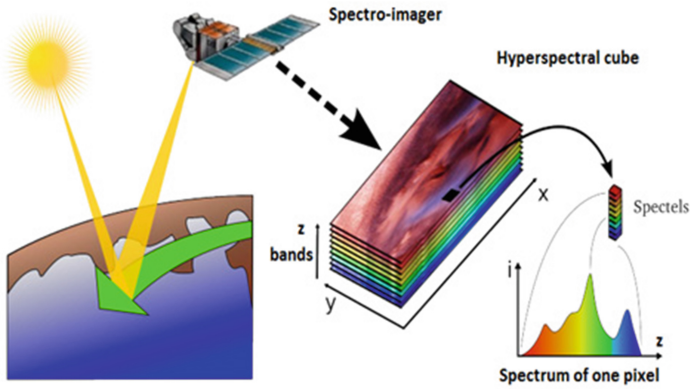


Fig. 1. Acquisition and decomposition of hyperspectral image

hyperspectral imaging is used in several fields, for example: geology, agriculture, town planning, forestry, in the military field.

To prepare hyperspectral image for visualization or further analysis such as classification, it is necessary to reduce the dimensions of the image to dimensions that can be analyzed by humans. Several dimensions reduction techniques exist. We find iterative versions and also parallel ones [2].

In this paper, we will propose a distributed and parallel version of the PCA dimension reduction algorithm that will be tested on the Spark platform using the MapReduce paradigm.

The rest of this paper is organized as follows: In Sect. 2, we will make an overview of the distributed parallel platforms most known in the field of BIG DATA processing. Thereafter, in Sect. 3, we will see the classic PCA dimension reduction technique and our proposed parallel distributed PCA. The tests of the proposed algorithm are in Sect. 4. Finally, we finish this paper with a conclusion and the future work.

2 Parallel and Distributed Platforms

In order to deal with BIG DATA such as our case hyperspectral images, we will use parallelized and distributed calculations in order to obtain results in a reasonable time.

Platforms that perform parallel distributed processing are multiplying in recent years. The two most recognized tools are Apache Hadoop and Apache Spark.

2.1 Apache Hadoop

Hadoop is among the most widely used, distributed platforms in the BIG DATA domain for storing data with his file manager named HDFS, and processing data with MapReduce on thousands of nodes [3]. (see Fig. 2).

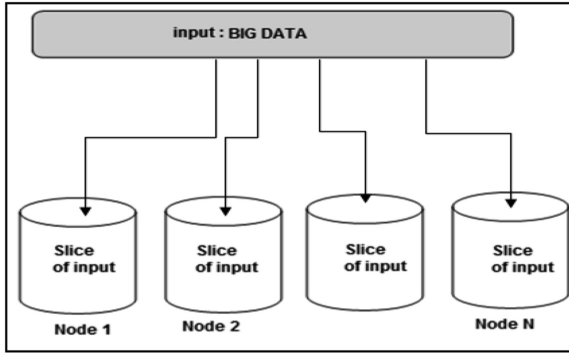


Fig. 2. HDFS abstraction of physical storage

2.2 Apache Spark

Apache Spark is an open source distributed platform for faster and sophisticated processing of BIG DATA, developed by AMPLab, from Berkeley University in 2009 and became an open source Apache project in 2010 [4].

Compared to other BIG DATA platforms, Apache Spark has many advantages:

- At storage levels: Spark allows to store the data in memory in the form of Resilient Distributed Data Set (RDD)
- At processing level: Spark extend Hadoop’s Map-Reduce programming that works on disk to process RDDs in memory, allowing it to run programs in memory, up to 100 times faster than Hadoop MapReduce and in disk 10 Times faster
- In addition to the operations that exist in Hadoop (MapReduce), Apache Spark offers the possibility to work with SQL queries, streaming, graph processing, Learning machine... (see Fig. 3)

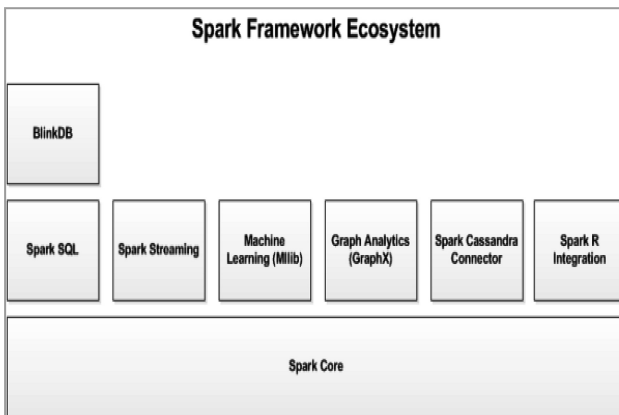


Fig. 3. Spark framework libraries

3 Dimensionality Reduction

Now to understand the information hidden in the hyperspectral image cube from human, or extract a useful part of the image, we often resort to visualization.

However, narrow human perception cannot visualize more than 3 hyperspectral bands. So before starting the visualization of our hyperspectral image, we must start by reducing the spectral bands to 3 without losing the quality of the information.

In the last few years, several techniques of dimensionality reduction have been made to reduce the hyperspectral data to a space of lower dimension, important examples include: ISOMAP, LLE, Laplacian eigenmap embedding, Hessian eigenmap embedding, conformal maps, diffusion maps and Principal Components Analysis (PCA) [5].

In the following, we will use PCA, the most popular technique in several domains: reduction of dimensionality, image processing, visualization of data and discovery of hidden models in the data [6].

3.1 Classic PCA Algorithm

Principal Component Analysis is a technique of reducing dimensions of a matrix of quantitative data. This method allows the dominant profiles to be extracted from the matrix [7]. The description of the classical PCA algorithm is as follows:

We assume that our hyperspectral image is a matrix of size $(m = L \times C, N)$ where L is the number of lines in the image, C is the number of columns and N is the number of bands with $m \gg N$.

$$X \begin{bmatrix} X_{11} & \cdots & X_{1N} \\ \vdots & \ddots & \vdots \\ X_{m1} & \cdots & X_{mN} \end{bmatrix}$$

Each line of the matrix X represents the pixel vector. For example the first pixel is represented by the vector: $[X_{11}, X_{12}, \dots, X_{1N}]$, with X_{1j} is the value of the pixel 1 taken by the spectrum of number j .

Each column of the matrix X represents the values of all the pixels of the image taken by a spectrum. For example $X_{i1} = [X_{11}, X_{21}, \dots, X_{m1}]$ represents the data of the image taken by the spectrum 1.

To apply the PCA algorithm to the hyperspectral image X , the following steps are followed:

- Step 1: Calculate the reduced centered matrix of X denoted: XRC

$$XRC_{ij} = \frac{X_{ij} - \overline{X_j}}{\sigma_j} \text{ for each } i = 1 \dots m \text{ and for each } j = 1 \dots N \tag{1}$$

$$\text{With } \overline{X_j} = \frac{1}{m} \sum_{i=1}^m X_{ij} \text{ And } \sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (X_{ij} - \overline{X_j})^2$$

In the formula 1, $\overline{X_j}$ denoted the average of column j and σ_j denoted the Standard deviation of column j .

- Step 2: Calculate the correlation matrix of size (N, N) denoted: Xcorr.

$$X_{\text{corr}} = \frac{1}{m} (XRC^T \cdot XRC) \quad (2)$$

In the formula 2, $XRC^T \cdot XRC$ denoted the matrix product between the transpose of the matrix XRC and the matrix XRC

- Step 3: Calculate the eigenvalues and eigenvector of the Xcorr matrix denoted: $[\lambda, V]$
- Step 4: Sort the eigenvector in descending order of the eigenvalues and take the first k columns of V ($k < N$)
- Step 5: Project the matrix X on the vector V : $U = X \cdot V$
- Step 6: use the new matrix U of size (m, k) for the visualization of the hyperspectral image

3.2 Distributed and Parallel PCA Algorithm

Related works:

There are currently two popular libraries that provide a parallel distributed implementation for the PCA algorithm: MLLib [8] on spark and the Mahout based on MapReduce [9]. In the Mllib library of Spark, we find an implementation for the parallel distributed PCA, but this implementation is done with the two languages: Sclala and Java. No implementation is made for the Python language.

In [6], Tarek et al. have shown that these two libraries do not allow a perfect analysis of a large mass of data and proposed a new PCA implementation, called sPCA. This proposed algorithm has a better scaling and accuracy than these competitors.

In [2], Zebin et al. proposed a new distributed parallel implementation for the PCA algorithm. The implementation is done using the Spark platform and the results obtained are compared with a serial implementation on Matlab and a parallel implementation on Hadoop. The comparison shows the efficiency of the proposed implementation in terms of precision and computation time.

In the following, we will propose a new implementation for the parallel distributed PCA algorithm based on the Apache Spark platform using the Python programming language and which uses the distributed Mllib matrices.

Proposed implementation:

Since the hyperspectral image is a representation of the same scene with several spectral bands, we can decompose the hyperspectral image into several images, each image for a given spectrum (see Fig. 4).

The classic PCA algorithm requires intensive computation because of large hyperspectral image. We will present in this part a method of parallel distributed implementation of the algorithm using the Spark platform.

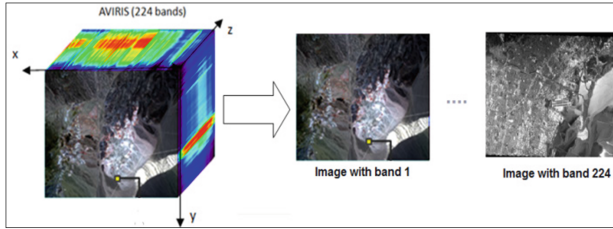


Fig. 4. Representation of the hyperspectral image by several images

First, we began by transforming the X matrix (see Fig. 5a) used to represent the hyperspectral image in the classic PCA to a vector of images denoted M , where each column of X is represented by an image in M (see Fig. 5b).

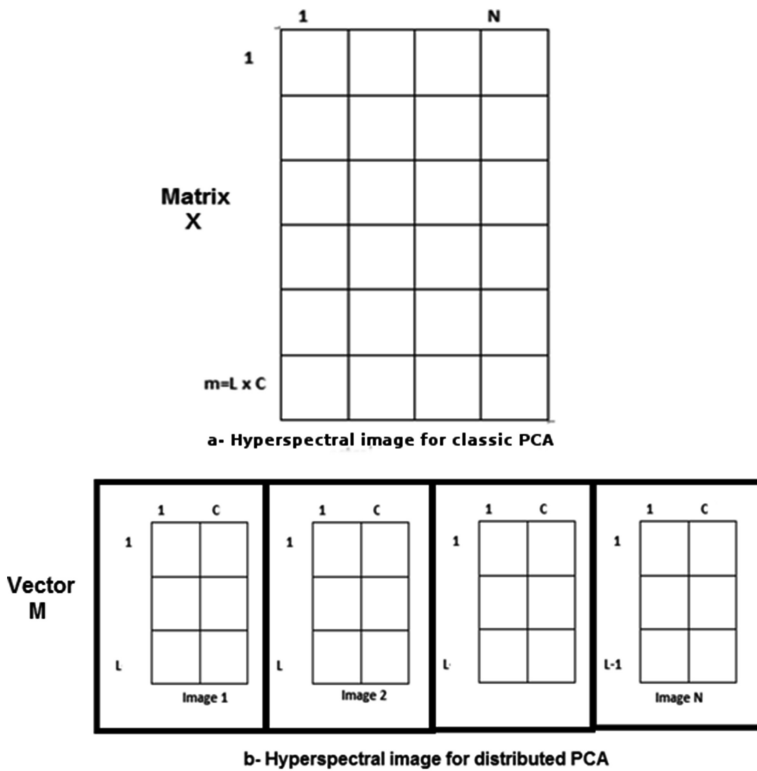


Fig. 5. Representation of hyperspectral image for classic PCA and distributed PCA

Now, each image t of M denoted M_t is a matrix in our implementation (Represents an RDD in parallel distributed spark programming):

To make a parallel distributed implementation of PCA we will use the map reduce paradigm of spark. The proposed algorithm proceeds as follows

- Step 1: Calculate the reduced centered matrix of M:

As has been seen before, the matrix M contains several images and each image M_t is represented by a matrix (an RDD in Spark notation) of size (L, C) where L is the number of lines in image and C is the number of columns. Therefore, to calculate the reduced centered matrix of M denoted MCR, a parallel distributed computation is carried out of each image M_t (See graphical description of the algorithm in Fig. 6).

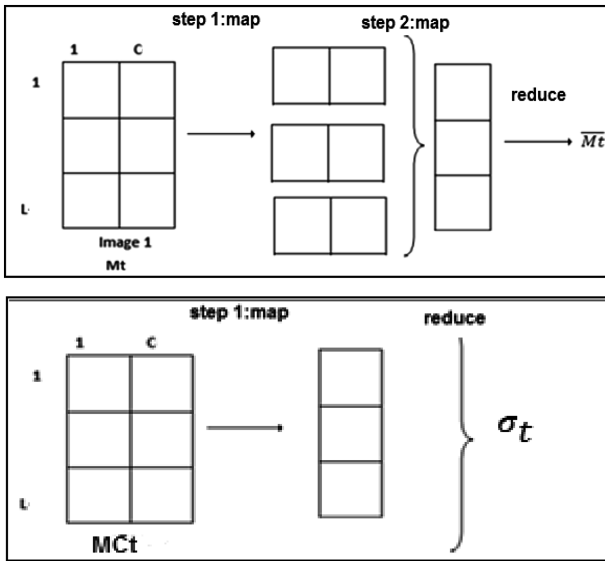


Fig. 6. Calculating the average of M_t and σ_t of MC_t with Spark

- Calculate the reduced matrix of M denoted MC:

$$MC_{t,ij} = M_{t,ij} - \bar{M}_t \text{ for each } i = 1 \dots L \text{ and for each } j = 1 \dots C \tag{3}$$

$$\text{with } \bar{M}_t = \sum_{i=1}^L \sum_{j=1}^C \left(\frac{1}{L \times C} \times M_{t,ij} \right)$$

In the formula 3, \bar{M}_t denoted the average of image M_t

- Calculate the reduced centered matrix of M denoted MCR:

$$MCR_{t,ij} = \frac{MC_{t,ij}}{\sigma_t} \text{ for each } i = 1 \dots L \text{ and for each } j = 1 \dots C \tag{4}$$

$$\text{with } \sigma_t^2 = \frac{1}{L \times C} \sum_{i=1}^L \sum_{j=1}^C (M_{t,ij} - \bar{M}_t)^2$$

$$\text{or } \sigma_t^2 = \frac{1}{L \times C} \sum_{i=1}^L \sum_{j=1}^C (MC_{t,ij})^2$$

In the formula 4, σ_t denoted the standard deviation of image t

- Step 2: Calculate the MCR correlation matrix of size (N, N) denoted: M_{corr}

According to step 1, the MCR is an images vector of size N . Each image represents a reduced centered matrix.

The next step is to calculate the correlation matrix of size (N, N) , by making the matrix product, between the vector MCR^T and the vector MCR , using a distributed parallel computation MapReduce of Spark (See graphical description of the algorithm in Fig. 7).

$$M_{corr} = \frac{1}{L \times C} (MCR^T \cdot MCR) \tag{5}$$

$$M_{corr_{t,k}} = \frac{1}{L \times C} (MCR_t \cdot MCR_k) \tag{6}$$

for each $t = 1 \dots N$ and for each $k = 1 \dots N$

$$\text{with } MCR_t \cdot MCR_k = \sum_{i=1}^L \sum_{j=1}^C MCR_{t,ij} \cdot MCR_{k,ij}$$

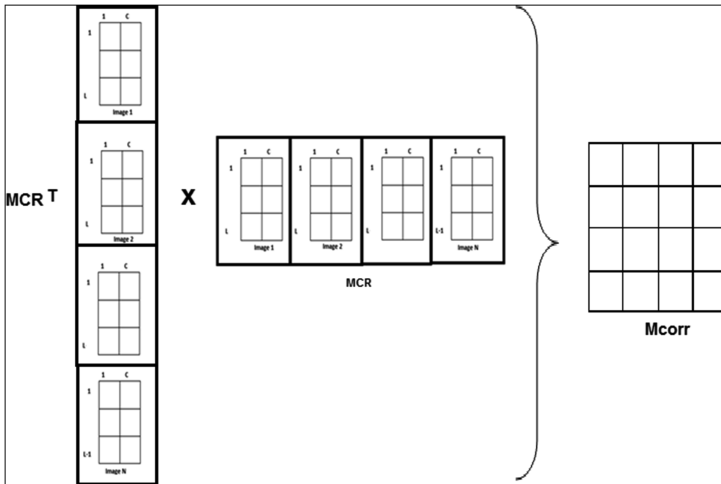


Fig. 7. Calculation of the correlation matrix with Spark

To calculate the value of each $Mcorr_{t,k}$ (Formula 6), the image MCR_t is multiplied by the image MCR_k pixel by pixel. Then we calculate the mean of result (See graphical description of the algorithm in Fig. 8).

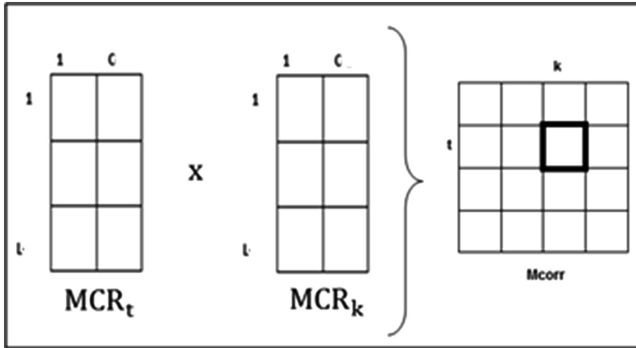


Fig. 8. Multiplication of two images

- Step 3: Calculate the eigenvalues and eigenvector of the Mcorr matrix: $[\lambda, V]$
- Step 4: Sort the eigenvector in descending order of the eigenvalues and take the first k columns of V ($k < N$)
- Step 5: Project the matrix X on the vector V : $U = X.V$
- Step 6: use the new matrix U of size (m, k) for the visualization of the hyperspectral image

4 Experimental and Computational Details

To test the validity of the proposed algorithm on hyperspectral images using the Apache Spark platform, we chose a set of open hyperspectral images of different sizes (see Table 1) and we tested our serial PCA algorithm, serial PCA of the Sklearn library of Python and parallel distributed PCA proposed on these datasets.

Table 1. Datasets

	Name	Spatial dimensions	Hyperspectral bands	Size
Dataset1	Moffett Field	500 × 500	3	5.3 MB
Dataset2	Moffett Field	500 × 500	10	17.5 MB
Dataset3	Moffett Field	1924 × 753	224	2.3 GB

The hyperspectral image used in our experiments for the classic PCA algorithm or the proposed PCA distributed algorithm is the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) Moffett Field image with 224 spectral bands in the 2.5 nm to 400 nm, which was acquired on August 20, 1992 [10].

The three algorithms are implemented in Python 3 and are executed on several configurations (see Table 2) and the results of the experiments of PCA are given in Table 3.

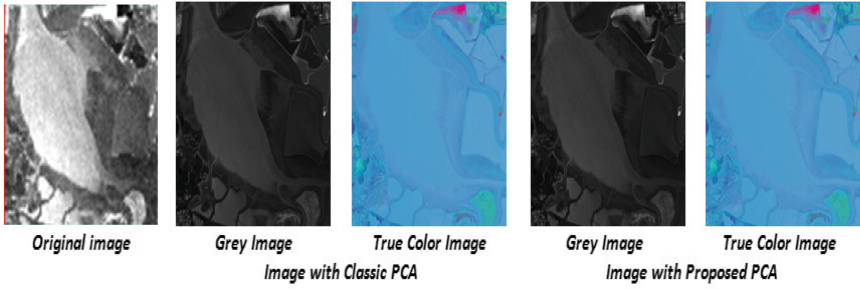
Table 2. Configuration parameters

Classic PCA	Distributed and parallel PCA
CPU: Intel Core I5, 3.3 GHZ RAM: 4G OS: Ubuntu 16.04 LTS	Cluster Spark: Master Node: 1 Slave Nodes: 4 CPU of each node: Intel Core I5, 3.3 GHZ RAM of each node : 4G Network speed : 100 MB/s OS: Ubuntu 16.04 LTS

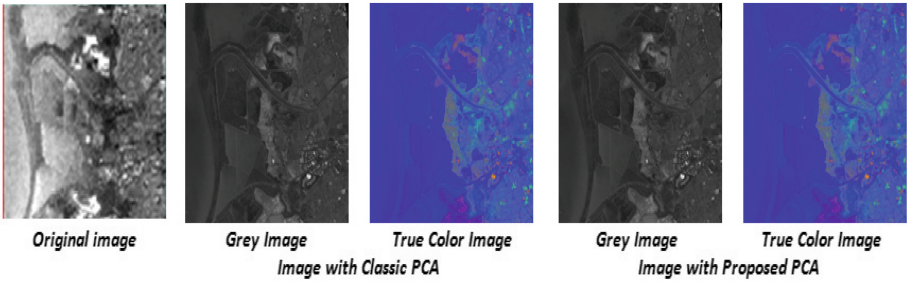
Table 3. The three most significant eigenvalues of PCA

	Our serial PCA	Sklearn serial PCA	Proposed distributed PCA
Dataset1	1.9371026343 0.913755533084 0.149141832615	1.9371026343 0.913755533084 0.149141832615	1.9371026343 0.913755533084 0.149141832615
Dataset2	8.12709201824 0.880534232525 0.797956006441	8.12709201825 0.880534232525 0.797956006442	8.12709201825 0.880534232525 0.797956006442
Dataset3	160.638762642 28.0031335174 14.5639033111	160.638762642 28.0031335174 14.5639033111	160.638762642 28.0031335174 14.5639033111

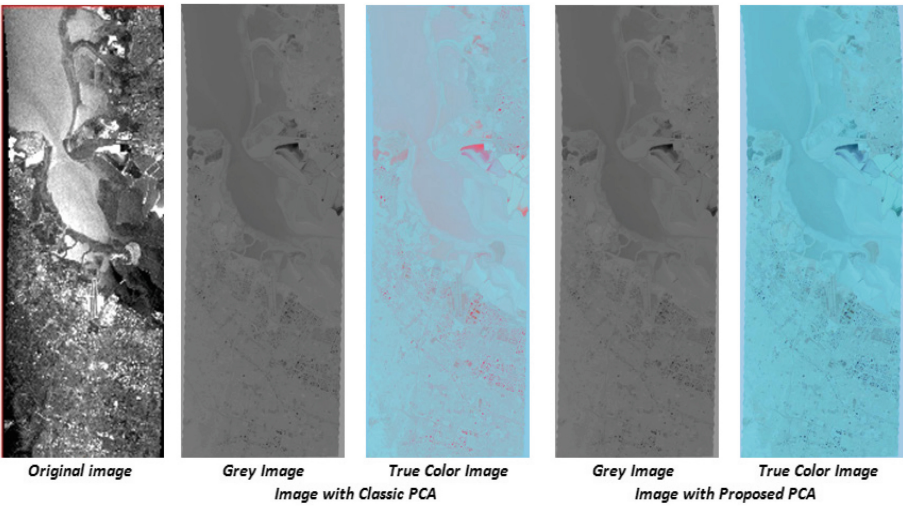
The visualization of the hyperspectral image after the application of the classic PCA algorithm or the proposed PCA distributed algorithm is given in Fig. 9.



(a) DataSet1



(b) DataSet2



(c) DataSet3

Fig. 9. Dataset Visualization, before and after application of classic PCA (our serial PCA or Sklearn serial PCA) and the proposed Distributed PCA

5 Conclusions

In this work, we proposed a parallel and distributed algorithm for dimensionality reduction called PCA. The algorithm is developed in Python 3 and tested on hyperspectral images using the Spark platform. The results coincide with the results of classic PCA and the visualization of the images after the application of our reduction algorithm confirms the validity of our algorithm.

In the next work we try to validate our algorithm based on the execution time of each method.

References

1. Mercier, L.: Système d'analyse et de visualisation d'images hyperspectrales appliqué aux sciences planétaires (2011)
2. Zebin, W., et al.: Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures. *IEEE J. Sel. Top. Appl. Earth Observations Remote Sens.* **9** (6), 2270–2278 (2016)
3. Apache Software Foundation. Official apache hadoop. <http://hadoop.apache.org/>. Accessed 10 July 2017
4. Apache Spark - Lightning-Fast Cluster Computing. <http://spark.apache.org/>. Accessed 10 July 2017
5. Van Der Maaten, L., Postma, E., Van den Herik, J.: Dimensionality reduction: a comparative. *J. Mach. Learn. Res.* **10**, 66–71 (2009)
6. Elgamal, T., et al.: sPCA: scalable principal component analysis for big data on distributed platforms. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM (2015)
7. Shlens, J.: A tutorial on principal component analysis. arXiv preprint [arXiv:1404.1100](https://arxiv.org/abs/1404.1100) (2014)
8. MLlib machine learning library. <https://spark.apache.org/mllib/>. Accessed 10 July 2017
9. Mahout machine learning library. <http://mahout.apache.org/>. 10 July 2017
10. AVIRIS - Airborne Visible/Infrared Imaging Spectrometer - Data. http://aviris.jpl.nasa.gov/data/image_cube.html. 10 July 2017