Multispectral Image Processing using Principal Component Analysis (PCA)

Group 3

Muhammad Farae
Faculty of Computer Science and Engineering
Ghulam Ishaq Khan Institute of Engineering Scienes and Technology
Topi, Pakistan
u2020292@giki.edu.pk

Muhammad Umair
Faculty of Computer Science and Engineering
Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
Topi, Pakistan
u2020369@giki.edu.pk

Syed Zaeem Shakir
Faculty of Computer Science and Engineering
Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
Topi, Pakistan
u2020487@giki.edu.pk

***Abstract*—This manuscript discusses the project that was undertaken in the ES304-Linear Algebra 2 Semester Project. The process of analyzing a multispectral image necessitated the need to download the Landsat image from the United States Geological Survey (USGS) website. The project entailed basic operations to be performed on the multispectral image bands such as band concatenation, clipping, and visualization. The Python programming language was chosen for its efficiency and familiarity the group members already posed. Principal Component Analysis (PCA) was implemented from scratch in Python and appropriate Error Analysis was conducted afterwards. The following sections describe each phase of the project in extensive detail.**

Keywords—Principal Component Analysis (PCA), image reconstruction, Landsat, satellite image, image processing

I. Introduction

Principal Component Analysis, from here onward referred to simply as PCA for legibility, is used extensively in modern applications for data compression. The data could be of many formats including simple scatterplot data or an image. It is a versatile statistical technique that allows us to analyze and understand high-dimensional data. By identifying the most important patterns and relationships, PCA empowers us to make informed decisions, improve computational efficiency, and gain valuable insights from complex data sets.

In many real-world scenarios, data sets often contain a large number of variables or features. Analyzing such high-dimensional data can be challenging and computationally expensive. PCA provides a way to transform this high-dimensional data into a lower-dimensional representation while retaining as much information as possible. At its core, PCA aims to find a new set of variables, called principal components, that capture the maximum amount of variation present in the original data. These principal components are linear combinations of the original variables, arranged in decreasing order of their importance. The first principal component accounts for the most significant variation, followed by the second component, and so on. By selecting a subset of these principal components, we can effectively reduce the dimensionality of the data.

PCA also helps in identifying the relationships and correlations among variables. It accomplishes this by ensuring that the principal components are orthogonal to each other, meaning they are uncorrelated. This property allows us to understand the underlying structure of the data by examining how different variables contribute to each principal component.

One of the key benefits of PCA is its ability to simplify complex data sets while preserving the most essential information. This reduction in dimensionality not only facilitates data visualization but also aids in overcoming the curse of dimensionality, where the performance of machine learning algorithms can deteriorate as the number of variables increases. Furthermore, PCA has applications beyond dimensionality reduction. It is commonly used in exploratory data analysis, outlier detection, feature extraction, and data preprocessing. By transforming the data using PCA, we can enhance the performance of various machine learning algorithms and improve our understanding of the data.

II. Image Reconstruction

A. Acquiring the Data Set

First, the satellite multispectral image of the intersection between Punjab and Balochistan Province were downloaded from the USGS website [1]. It must be noted that the dataset was generated by the Landsat 7 Enhanced Thematic Mapper Plus (ETM+) satellite, which captures images across eight bands, as specified in the image below.

Landsat 7	Wavelength (micrometers)	Resolution (meters)
Band 1 - Blue	0.45-0.52	30
Band 2 - Green	0.52-0.60	30
Band 3 - Red	0.63-0.69	30
Band 4 - Near Infrared (NIR)	0.77-0.90	30
Band 5 - Shortwave Infrared (SWIR) 1	1.55-1.75	30
Band 6 - Thermal	10.40-12.50	60 (30)
Band 7 - Shortwave Infrared (SWIR) 2	2.09-2.35	30
Band 8 - Panchromatic	.52-.90	15

The sensors aboard each Landsat satellite are designed to acquire data in different ranges of frequencies along the electromagnetic spectrum. By combining these different bands together, the information represented by the unified image is manifold those of standard pictures with only three channels (red, green, and blue). In a practical scenario, if a

deep learning technique such as a Convolutional Neural Network (CNN) were to be implemented for Computer Vision applications, processing images with such high dimensionality would pose a resource constraint even on the post powerful Graphical Processing Units (GPUs).

B. Image Concatenation, Cropping, & Visualization

This part of the project required the use of Python frameworks such as PIL, xarray, rioxarray, earthpy in order to accomplish the basic image manipulation functionality for viewing and analyzing images in the Integrated Development Environment (IDE), which was of crucial importance during the results analysis stage.

After loading the images using OpenCV, the bands below were retrieved. It must be noted that although all eight bands are displayed in Appendix A, the dimensions of the Panchromatic Band (i.e., Band 8) was double the dimension of the rest of the bands, creating compatibility issues during the coding process; thus, band 8 was truncated from the overall concatenated image that is displayed in Appendix B.

It must be noted that the band 8 image was easily resized using the built-in resize command of OpenCV; however, the datatype of the resultant image subsequently was altered and as a result it created datatype mismatch errors when the band was attempted to be stacked alongside the other bands.

The code file provided with the project also demonstrates how the cropping functionality was implemented using the PIL library in Python.

III. Preprocessing for PCA

For this phase, the true color image was downloaded as shown in Appendix C. Using the "split" function of OpenCV, the color image was segregated into its red, green, and blue channels and plotted as shown in Appendix D. This step is significant due to the fact that the PCA function will operate on a single channel of image at a time and stack the results at the end. It must also be specified that in order to minimize the error produced during PCA, the data that was to be passed into the PCA function was standardized in advance by dividing all pixel values by 255 to scale all the values from 0-1 so that no component is dominant over the other when we apply PCA. Standardization ensures that the data is linearly distributed so that the original data can be recovered from the transformed data.

Aside from normalization, the sample image was also reduced from its original dimension of 8081 by 7021 to a 500 by 500 image to ensure the PCA function executes more efficiently.

IV. PCA Implementation

This section outlines the actual implementation of the statistical technique that is PCA[2]. First, the code in Python is presented, followed by the step-by-step explanation of the theoretical background and basis of each line.

A. PCA Code

The implementation of the PCA [3] is shown below this paragraph. As can be seen, the function accepts two arguments: the original image to be operated upon and the number of principal components to be specified from it.

```
def pca_art(img, k):
    mean_data = img - np.mean(img, axis = 0)
    cov = np.cov(mean_data.T)
    #cov = np.round(cov, 2)
    eig_val, eig_vec = np.linalg.eig(cov)
    indices = np.arange(0, len(eig_val), 1)
    indices = ([x for _,x in sorted(zip(eig_val, indices))])[::-1]
    eig_val = eig_val[indices]
    eig_vec = eig_vec[:,indices]

    sorted_eig_vec = eig_vec[:,:k]

    pca_data = np.dot(mean_data, sorted_eig_vec)
    recon_img = np.dot(pca_data, sorted_eig_vec.T)+np.mean(img, axis=0)

    loss = np.mean(np.square(recon_img - img))
    return recon_img, loss
✓ 0.0s
```

The steps of PCA are outlined as below [4]:

1. Standardization is required due to the sensitive nature of PCA regarding variances of the initial variables. Transforming data to comparable scales of zero to one circumvents this problem.
2. Covariance matrix computation is the subsequent step. The covariance matrix is a p × p symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. In other words, the covariance matrix summarizes the correlation between all the possible pairs of variables.
3. Computation of eigenvectors and eigenvalues of the covariance matrix to identify the principal components is the following step, implemented in the above code by linalg.eig.
4. Sorting the eigenvectors and eigenvalues in descending order enable us to find the principal components in order of significance, extracting a feature vector map specified above as sorted_eig_vec.
5. Recasting of the data along the principal component axes is accomplished by multiplying the eigenvector matrix with the standardized dataset.
6. Lastly, the original image can be reconstructed by multiplying the resultant image with the eigenvector matrix and adding the mean to it to negate the standardization step. By doing so, the average loss of information can be computed and the reconstructed image can be plotted with the same dimensions as the original image for comparison purposes.

V. Results & Error Analysis

In this section, the overall quality of the results and whether they matched expectations are discussed insofar as to conclude whether the project was a success.

Appendix E displays the PCA that was iterated over multiples of 25 principal components until 200. The initial iterations proved to be highly lossy compressions with the reconstructed image being blurred to the point of hardly resembling the original image sans its color tone. However, the exponential improvement in the PCA procedure

implemented is evidenced by the increase in resemblance within just a few iterations and higher principal components. It is clear the project achieved desirable and expected results inline with theory.

To quantify the gradual improvement of the PCA, the mean square formula was applied (1) for each iteration in order to discern at which number of components could the technique be halted once suitable accuracy was reached.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \quad (1)$$

```
For PCA Value of 0
Error for Red Channel: 0.011173673373748557
Error for Blue Channel: 0.011173673373748557
Error for Green Channel: 0.011173673373748557



For PCA Value of 25
Error for Red Channel: 0.0008158496513114649
Error for Blue Channel: 0.0008158496513114649
Error for Green Channel: 0.0008158496513114649



For PCA Value of 50
Error for Red Channel: 0.00045961763940376296
Error for Blue Channel: 0.00045961763940376296
Error for Green Channel: 0.00045961763940376296



For PCA Value of 75
Error for Red Channel: 0.0003037442958598006
Error for Blue Channel: 0.0003037442958598006
Error for Green Channel: 0.0003037442958598006



For PCA Value of 100
Error for Red Channel: 0.00021160378915043148
Error for Blue Channel: 0.00021160378915043148
Error for Green Channel: 0.00021160378915043148



For PCA Value of 125
Error for Red Channel: 0.0001503202051019376
Error for Blue Channel: 0.0001503202051019376
Error for Green Channel: 0.0001503202051019376



For PCA Value of 150
Error for Red Channel: 0.00010745750785436094
Error for Blue Channel: 0.00010745750785436094
Error for Green Channel: 0.00010745750785436094



For PCA Value of 175
Error for Red Channel: 7.662547859684596e-05
Error for Blue Channel: 7.662547859684596e-05
Error for Green Channel: 7.662547859684596e-05



For PCA Value of 200
Error for Red Channel: 5.4227090302049524e-05
Error for Blue Channel: 5.4227090302049524e-05
Error for Green Channel: 5.4227090302049524e-05

```

Conclusion

In conclusion, the mean square error converged to a reasonable value by the time the iterations had reached one hundred principal components, though one could argue suitable accuracy was reached by seventy-five or even fifty principal components depending upon the desired degree of accuracy. All in all, the project was a success since the group members applied the PCA data compression technique manually without built-in shortcuts in Python and achieved results that matched theoretical expectations.

References

1. US Geological Survey (USGS). [Online]. Available: https://www.usgs.gov/. [Accessed: May 11, 2023].

2. "A Step-by-Step Explanation of Principal Component Analysis," Built In, [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Accessed: May 11, 2023].

3. Jolliffe, I. T. (2002). Principal Component Analysis. John Wiley & Sons.

4. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.

Task Distribution

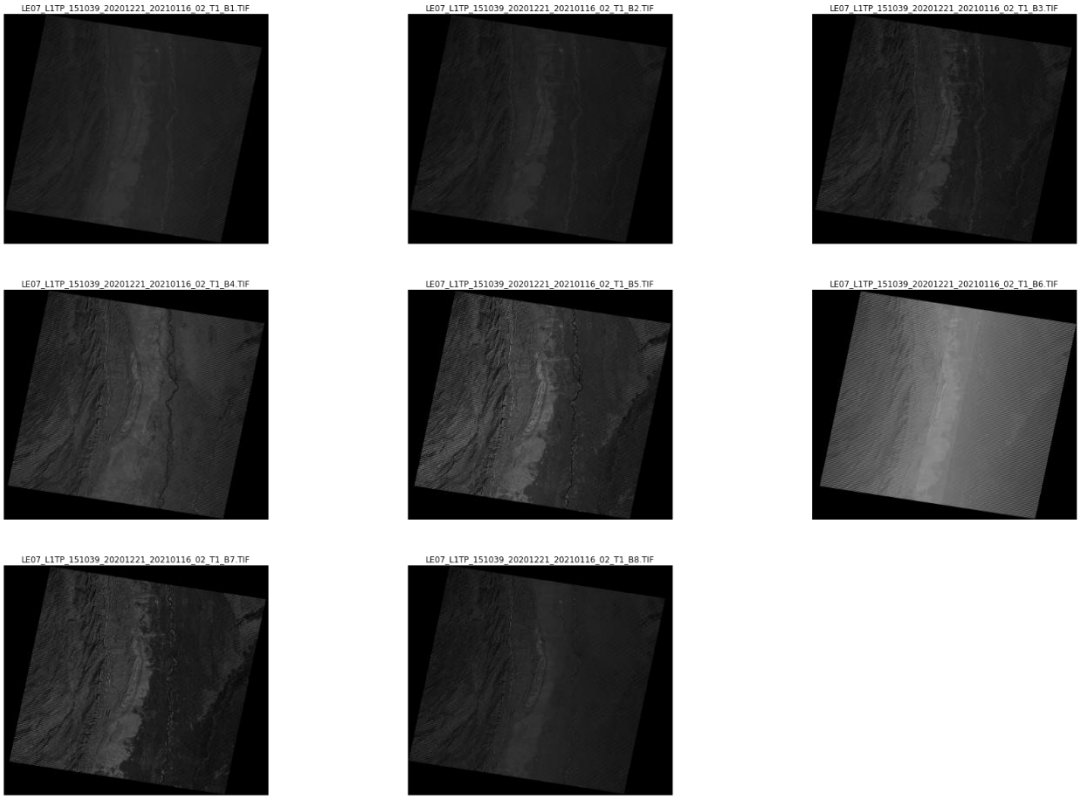
Throughout the project, Zaeem, Farae, and Umair maintained effective communication and shared their progress regularly. They provided feedback, exchanged ideas, and resolved any challenges they encountered. This collaborative effort ensured the successful completion of the PCA project, with each student making valuable contributions in their respective areas of expertise.

Zaeem took the lead in implementing the image reconstruction component of the project. This involved understanding the theoretical concepts behind image reconstruction and implementing the necessary Python code and aided in the report writing and results analysis.

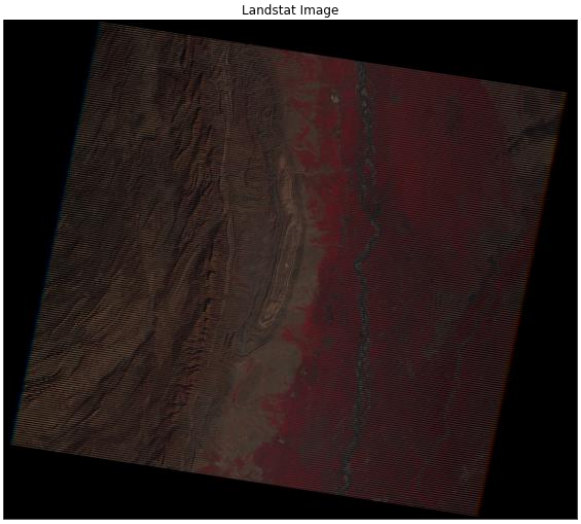
Farae focused on the implementation of PCA itself. He delved into the mathematical foundations of PCA, familiarized themselves with the required Python libraries, and developed the code for performing PCA on the given dataset. Farae ensured the extraction of principal components, calculated eigenvalues, and eigenvectors, and carried out the necessary data transformations.

Umair took charge of the error analysis aspect of the project. They were responsible for evaluating the accuracy and effectiveness of the PCA results. Umair calculated various error metrics, such as reconstruction errors, to assess the quality of the reconstructed images and measure the performance of the PCA algorithm.

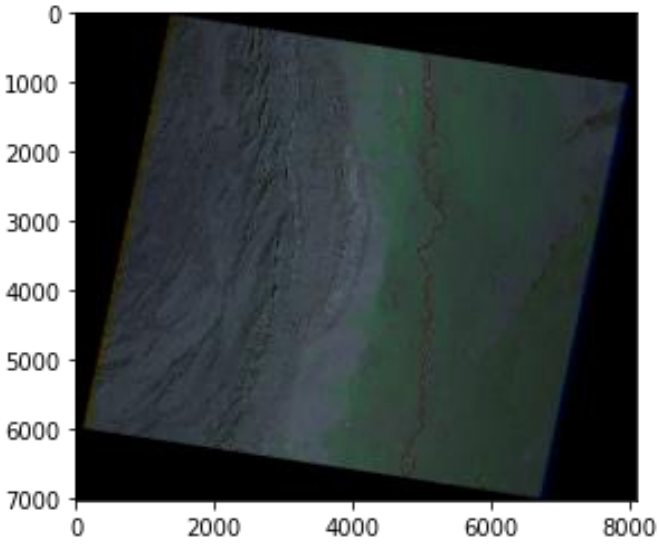
Appendix A



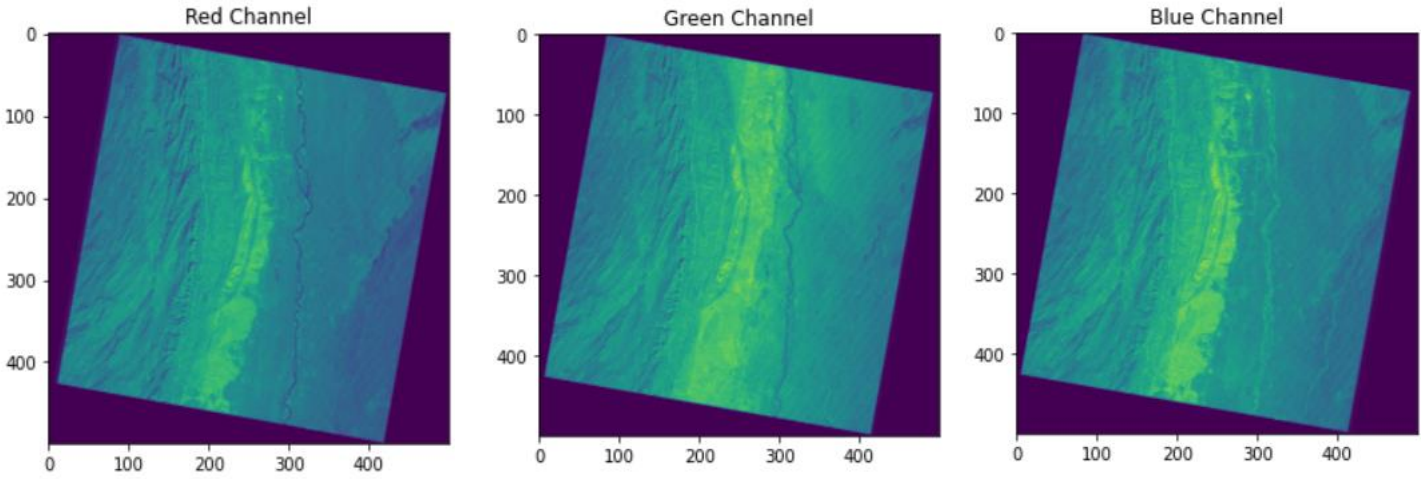
APPENDIX B



APPENDIX C



APPENDIX D



APPENDIX E