PA026 Report - LIBS hyperspectral map segmentation

Pavel Nedělník, 485564
Petr Kadlec, 485208

June 20, 2023

1 Project Description

The goal of the project was to design and implement an interactive graphical application for the analysis of hyperspectral images obtained by Laser-Induced Breakdown Spectroscopy. The application provides the user with tools to explore the images and assign labels to individual spectra with machine learning techniques. Due to the time limitations, the project mainly focuses on the necessary graphical application rather than fully developing the machine learning models.

The main focus point of the application is the Image Panel, it displays the hyperspectral image as a heatmap with intensities summed over all measured wavelengths and provides the user with a brush tool that allows them to quickly assign labels to large areas of the image by drawing over them.

A line plot allows the user to inspect any spectrum by hovering over the corresponding point in the heatmap. To change which wavelengths are used to calculate the user can select a region on a line plot of the mean spectrum. The user can control the machine learning models supporting the visualization through the Model Panel. The manually assigned labels can be downloaded and uploaded. The application also allows the user to download the labels suggested by any of the machine learning models.

2 Related Works

There are several ways to tackle the problem of image segmentation in hyperspectral maps. These approaches could be split into these categories: [1, 2]

1. Threshold-based segmentation

   Intensity-based method where pixels belonging to a range of intensities are in a class and the rest belongs to some others.

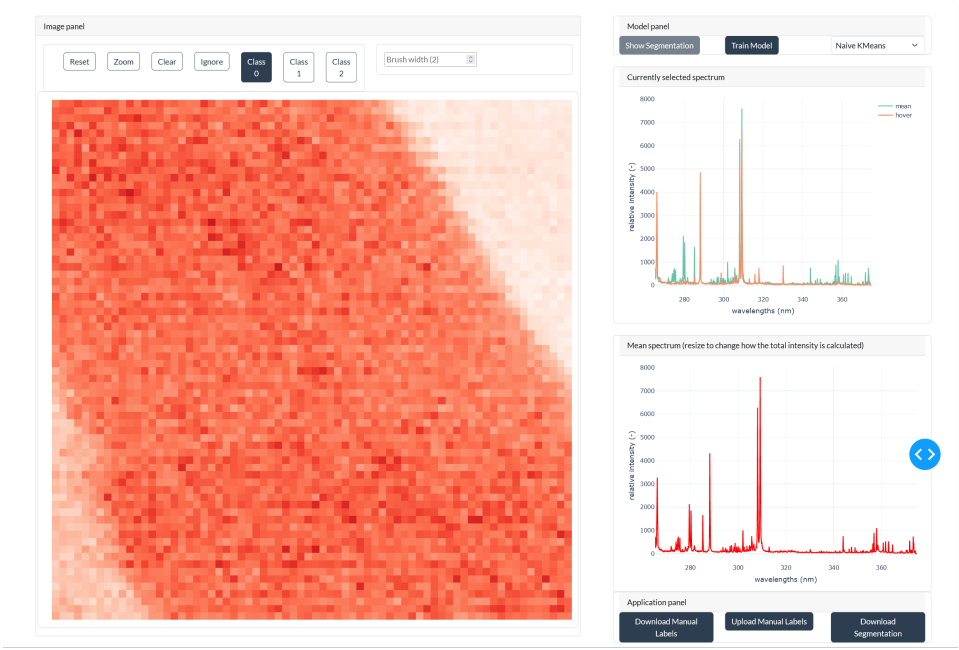

Figure 1: Layout of the application.

2. Clustering-based segmentation

   Standard clustering task that is well known where we try to segment the hyperspectral image into multiple classes based on pixel distances.

3. Watershed segmentation

   Watershed segmentation involves treating the image gradient as a topographic surface. Bright pixels are viewed as elevated areas or watershed lines, while dark pixels represent lower regions or basins. To initiate the segmentation process, a marker or seed point is intuitively selected within each object and expanded using the morphological watershed technique. By flooding the basins, the areas where floodwater from multiple basins intersects are identified, and borders are established using a segmentation approach.

4. Morphological segmentation

   Morphological segmentation involves segmenting images by manipulating their shapes and structure using structural elements. Morphological operations include opening, closing, dilatation, and erosion.

5. Edge detection-based segmentation

   Segmentation based on edge detection relies on identifying the discontinuities in pixel intensity values and generating binary images. The

edges are detected by comparing the first-order derivative of the pixels to a specific threshold value or by identifying zero crossings in the second-order derivative. These detected edges are then connected to form the boundaries of objects.

6. Superpixel segmentation

   Superpixel segmentation refers to the process of grouping pixels based on their similar characteristics, such as intensity values. These superpixels contain more information than individual pixels and are perceptually relevant as they capture visual qualities shared by comparable pixels. The most commonly used methods for generating superpixels include simple linear iterative clustering and entropy rate segmentation.

7. Region segmentation

   This approach involves dividing the regions with similar properties into groups. It can be categorized into three main techniques: region growth, region splitting, and region merging.

8. Deep Learning

   Approaches using deep neural networks, convolutional neural networks, long short-term memory, and transfer learning.

3 Methodology

Out of all of the presented categories in the previous chapter, we put our focus on the underlying machine learning algorithms used.

3.1 The goal of the ML task

The goal of the machine-learning task was to segment the image into multiple zones, with the idea being that the sample from which we got the hyperspectral map is composed of several base elements. Each of the elements outputs a different wavelength when hit with a high-powered laser, thus from that, we can theoretically deduce the combination of elements that produced a given spectrum.

3.2 Models

We implemented a selection of promising and interesting machine-learning models from a hyperspectral image segmentation overview study [1]. The selection process was based on the novelty of the approach and the ease of implementation.

Based on a literature review [2, 3, 1], we decided to incorporate the following approaches: Principal Component Analysis (PCA) preprocessing followed by machine-learning models, namely Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP), K-Nearest Neighbors classifier (KNN), Convolutional Neural Network (CNN), Gradient Boosting classifier (GB), Gaussian Naive Bayes (GNB), and Random Forest (RF). As the baseline for the experiment, we chose an unsupervised K-Means clustering model.

3.3 Training process

The main problem for the training process was providing labels to the data points. As a remedy for this problem, we have created a solution to it by implementing the interactive map where it is possible to input class/segment labels for the machine learning models to learn. From this, we were able to have a manual training dataset. We have also created a synthetic dataset, this dataset is explained later in the evaluation section. One data point in the map represents roughly a vector of the size 3800 - the strength of response at different wavelengths.

After a user inputs these labels either through the included interactive map or upload option the machine-learning model is started on this map with the annotated data points being the training set and the rest of the data points are then to be predicted.

4 Implementation Details

The project is fully realized in Python, utilizing the Plotly Dash framework for defining a browser-based interactive user interface. The application incorporates a wide range of machine learning models implemented with the use of the scikit-learn package, as well as Keras. Data manipulation was done using NumPy.

The model hyperparameters are too extensive to be listed as a part of this report but can be found in the main.ipynb notebook alongside model definitions.

5 Evaluation

In the current phase of the project, we focused on validating the benefits of the user input and the effectiveness of the user interface. To this end, we designed two experiments.

In the first experiment, we directly simulated a possible use case of the application. A smaller dataset was first fully labeled by one member of the team before allowing the other to do the same with the use of the application, with as little human effort as possible. After a short window

for user interaction, results from all the considered models were collected and analyzed, see below. More information about the dataset used can be found in the aforementioned article, see Section 6.

The second experiment was done analogously to the first, the major change being that a synthetic dataset was used, rather than a real one. The synthetic data mimicked a steel sample - the considered elements were Iron, Carbon, Chromium, Nickel, and Manganese, with Iron and Carbon being varied randomly, and a gradient of Nickel and Manganese was used to define the target classes. The theoretical spectra were obtained with the use of a spectral database and were smoothed and Gaussian noise was added to them. The results of the experiments can be seen in the table below.

	K-Means	KNN	GNB	RF	GB
real	0.83	0.85	0.89	0.89	0.89
synthetic	0.63	0.94	0.78	0.95	0.95
	PCA+SVM	PCA+MLP	Large MLP	CNN	MLP
real	0.90	0.90	0.89	0.91	0.89
synthetic	0.95	0.96	0.97	0.97	0.95

Table 1: Accuracy of selected models for the experiments on real and synthetic data.

In both experiments, only approximately 1% of the dataset (single stroke per visible inclusion) had to be manually labeled for most models to achieve very high accuracy, speaking volumes in terms of the effectiveness of the approach, as the necessary steps can be done in terms of low minutes. Additionally, in all the experiments the supervised methods outperformed the baseline unsupervised K-Means model, affirming the benefits of the application.

6 Installation

The project code, along with a toy dataset, is available via a Github repository (https://github.com/PavelNedelnik/libs-segmentation). The code to generate the synthetic data used for evaluation, see Section 5, is also available. The dataset used to simulate a real application is available through Figshare (https://dx.doi.org/10.6084/m9.figshare.20713504).

After cloning the repository and installing all the necessary requirements, the application can be launched by running the main.ipynb notebook and can be accessed by most web browsers on localhost on port 8050. To replicate the experiments mentioned in Section 5, change the mode in the second cell of the main.ipynb notebook. The SimulatedLIBS package might need to be installed manually.

References

[1] Reaya Grewal, Singara Singh Kasana, and Geeta Kasana. "Hyperspectral image segmentation: a comprehensive survey". In: *Multimedia Tools and Applications* 82 (Oct. 2022), pp. 1–54. DOI: 10.1007/s11042-022-13959-w.

[2] Shraddha Tripathi et al. "Image segmentation: a review". In: *International Journal of Computer Science and Management Research* 1.4 (2012), pp. 838–843.

[3] Antonio Plaza et al. "Recent advances in techniques for hyperspectral image processing". In: *Remote Sensing of Environment* 113 (2009). Imaging Spectroscopy Special Issue, S110–S122. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2007.07.028. URL: https://www.sciencedirect.com/science/article/pii/S0034425709000807.