Image Processing for Earth Observation

Project Report

Glacier retreat in the Himalayas and implications for local populations

Loïc Brouet
Joshua Cayetano-Emond

Instructor
Frank de Morsier

Lausanne - January 8, 2021

1 Introduction

1.1 Context

Global warming is an accelerating crisis with worldwide implications. The rapid warming of the Earth well above pre-industrial levels is wreaking all sorts of havoc on the various ecosystems and biomes of the globe. One particularly hit region are the Himalayas, where massive glacial melt is present. This rapid melting is obviously cause for alarm and threatens local populations. Though, the way it threatens the local populations may seem surprising. Indeed, it is not directly the glacial melt that is threatening local populations but instead the glacial lakes that are being formed in the process. In fact, glacier-fed lakes are often dammed by unstable moraines which can collapse and lead devastating glacial lake outbursts floods (GLOFs) [1]. These lakes have been increasing in size and quantity due to global warming and are the main cause of concern for local populations.

1.2 Literature Review

Before starting to process remote sensing imagery, some literature review had to be performed in order to understand the features that could be used to properly extract information from the Himalayan scenes. Initially, articles that were read mainly involved glaciers and not glacial lakes. The first articles show how remote sensing can be used to estimate glacier retreating [2, 3]. The next article was very similar and used remote sensing to notice that the total glacier area was decreasing, while the number of glaciers has increased due to fragmentation [4]. This article also uses an interesting method to estimate glacial depth and ultimately volume.

After reading some articles on glacial retreat in the Himalayas, it was understood that while it is alarming, the main cause for concern is instead the expansion of glacial lakes. In fact, as mentioned earlier, these are often dammed by unstable moraines which can collapse and cause devastating GLOFs. The project's focus was therefore reframed and instead changed to glacial lakes. For this, a comprehensive article on an inventory of the glacial lakes in the Third Pole region (Himalaya) was read [1]. This article studied in depth the evolution of glacial lakes between 1990 and 2010 by manually labelling images in the Third Pole region, which mostly coincides with the Tibetan Plateau. The article provided a good understand for the current situation, showing that lakes have grown by about 20% between 1990 and 2010, with their number increasing in similar proportions. Another article was read providing further on a different region, this time the central Himalayas [5]. The situation in that region was found to be similar, though with a much more significant 122% expansion between 1976 and 2010.

While the above articles provided a good qualitative understanding for the situation, the lakes had to be manually labelled. There was a desire to determine if this process could be automated in some way. Some articles were read regarding the bands of interest that could be used for glacial lake identification [6]. Other articles were then read regarding possible indices that could be used to better identify these lakes. Unsurprisingly, these articles recommended using water indices, namely the NDWI [7, 8]. However, some other articles also suggested using snow indices (NDSI) to identify glaciers [9].

A very big challenge in identifying glacial lakes was also put forward in some of these articles. Indeed, mountain shadows have a spectral signature that is very close to that of glacial lakes. It can therefore be very difficult to properly separate glacial lakes from mountain shadows using an automatic classifier. The articles read proposed methods to counter this which mainly involved using DEMs. Indeed, glacial lakes should in theory be perfectly flat and have zero slope. As such, a DEM can be used to help filter out lakes from mountain shadows using the slope calculated from it [7, 8].

1.3 Objective

The objective of this report and project was therefore to try to come up with a mainly automatic way of characterizing glacial lakes from remote sensing imagery. The methods presented above were merged into a workflow that combined the strongest aspects from each method in an attempt to obtain a better model. This workflow is better explained in Section 3, but essentially involves using a combination of bands to calculate relevant indices as well as the slope computed from the DEM to classify images using a supervised machine learning algorithm.

In terms of a study region, a reference area was chosen from [1], as an image was available against which the obtained results could be compared. This image is shown in Figure 1.1. As this article had labelled their entire data set manually, an interesting side objective was the comparison between manual and automatic methods for identifying glacial lakes.

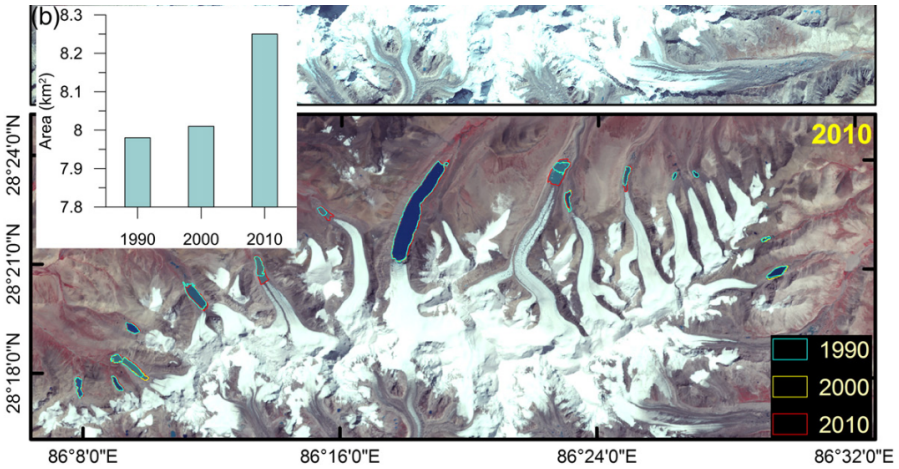

Figure 1.1: Evolution of the lakes in the reference region provided by [1]

1.4 Challenges

As outlined in Section 1.2, the main challenge in automatically classifying glacial lakes lies in mountain shadows. Indeed, these have a spectral signature that is very close to glacial lake bodies and this was confirmed during this project. To assist in properly identifying these shadowy zones, the terrain slope computed from the DEM was be used, as suggested by [8, 7]. Additionally, to aid the supervised machine learning classifier in properly distinguishing between mountain shadows and lakes, various examples of both classes were be provided to the classifier to give it more data points with which it can properly learn the features that separate both.

2 Data description

A first zone located on Tibetan Plateau in South China (North Himalaya) was defined as the reference zone. It is around 28°/ Lng. 86°, between Yarleb and Ronxar in China. Its area is about 1'400 km^2. Glacial lakes over there have been studied by [1]. Therefore, for the purpose of this project, images are acquired in this zone for different epochs to see the evolution of glacial lakes. The first epoch would be ideally around 2010 in order to compare results directly with [1] who have studied this zone between 1990 and 2010; however, the oldest data found were in 2013 with Landsat 8. Data were then collected from 2013 to 2020, once per year. Moreover, a second zone was considered to test the model in a different terrain. It is located also on Tibetan Plateau but at the border of China, Nepal and Bhutan, it is around Lat. 28°/ Lng. 82° and has the same area as the first zone.

The first source of data used comes from Landsat 8 satellite [10], an American Earth observation satellite launched in 2013. It belongs to the Landsat satellites whose first satellite was launched in 1972, being up to now the longest-running enterprise for acquisition of satellite imagery of Earth [11].

Landsat 8 has two sensors: the Operational Land Imager (OLI) and the Thermal InfraRed Sensor (TIRS). OLI catch data from nine spectral bands (band 1 to 9), 2 more bands are captured compared to the previous Landsat 7 (a deep blue coastal/aerosol band (1) and a shortwave-infrared cirrus band (7)). All bands have a spatial resolution of 30 m except for a Panchromatic band (8) which has a resolution of 15 m. TIRS realizes thermal imaging, collecting band 10 and 11, which is one more band than Landsat 7. Spatial resolution is 100 m. Data from OLI and TIRS are then registered together and corrected.

The data acquired from Landsat 8 were the ones from 2013 to 2016, because from 2017 another satellite offers a better spatial resolution. The data described in Figure 2.1, i.e band 2 to band 6 of Landsat, were acquired for the reference zone on 10.04.2014, 10.07.2015, 10.18.2016, and for the second zone on 10.09.2015. Data were download from the Sentinel Hub EO Browser [12]. The acquisition date was chosen according to the weather conditions. The goal was to have the less snow possible, as well as less clouds and shadows as possible. The favorable period was therefore around September or October, i.e after the monsoon and before the first snowfalls [1].

			Resolution					
Band	**Sensor**	**Spatial reference system**	**Spatial**	**Spectral**	**Radiometric**	**Temporal**	**Accuracy**	**Type of product**
Band 2 - Blue	Operational Land Imager (OLI) - Landsat 8	WGS 84 (EPSG:4326)	30m	0.450 - 0.515 µm	12 bit (0 - 4'095)	16 days	12 metres circular error, 90% confidence	raw
Band 3 - Green				0.525 - 0.600 µm				
Band 4 - Red				0.630 - 0.680 µm				
Band 5 - NIR				0.845 - 0.885 µm				
Band 6 - SWIR				1.560 - 1.660 µm				

Figure 2.1: Landsat 8 data description

The second source of data used came from Sentinel-2 [13], two satellites from European Space Agency launched in 2015 and 2017. It belongs to the Sentinel-2 Mission including two satellites: Sentinel 2A and Sentinel 2B. Both are identical and they are phased 180 degrees from each other on the same orbit to cover earth observations well. The revisit frequency (temporal resolution) of each single SENTINEL-2 satellite is 10 days but the combined constellation revisit is 5 days. They carry a single multi-spectral instrument (MSI) imager with 13 spectral channels in the visible/near infrared (VNIR) and short wave infrared spectral range (SWIR).

From 2017 when images were available over the Tibetan Plateau, the data were acquired from Sentinel-2 instead of Landsat 8 since it offers a better the spatial resolution. The data described in Figure 2.2 below were acquired for zone 1 on 10.10.2017, 10.05.2018, 10.10.2019 and 10.29.2020, and for the second zone on 10.11.2020. Data were also download from the Sentinel Hub EO Browser [12].

			Resolution					
Band	Sensor	Spatial reference system	Spatial	Spectral	Radiometric	Temporal	Accuracy	Type of product
Band 2 - Blue	Multi-Spectral Instrument (MSI) imager - Sentinel 2	WGS 84 (EPSG:4326)	10m	0.427 - 0.558 µm	12 bit (0 - 4'095)	5 days of the constellation of Sentinel 2-A and 2-B	3 metres circular error, 95% confidence	raw
Band 3 - Green				0.524 - 0.596 µm				
Band 4 - Red				0.634 - 0.696 µm				
Band 8A - Narrow NIR			20m	0.844 - 0.886 µm				
Band 11 - SWIR				1.523 - 1.705 µm				

Figure 2.2: Sentinel-2 data description

In order to compute indices, the bands described in Figure 2.1 and Figure 2.2 were used as stated in Section 1.2. However, given that the sensors are not the same, spectral resolution of the two platforms is not strictly identical but the bands were chosen to be very similar and comparable.

Furthermore, a digital elevation model (DEM) was used. For each zone the DEM comes from the Mapzen DEM available in the Sentinel Hub EO Browser [12]. It is based on the Shuttle Radar Topography Mission (SRTM) led in February 2000. As the DEM is mainly independent of the date (static), one DEM was used per zone. The horizontal resolution indicated by Sentinel Hub is 19 pixels/m, and the absolute vertical accuracy is less than 16m [14].

3 Methods

A methodology for extracting glacial lakes from satellite imagery was established and mainly inspired by the one suggested in [8]. The flowchart presented in the article was modified and slightly simplified to fit within the timeframe of this project. The method involves various steps of image processing that are detailed below. The entire process described below is also described visually as a flowchart in Figure 3.1.

The first step in the extraction process was simply to load the data of a given zone from the relevant bands for the analysis. These were, as stated in Section 2, the blue, green, red, NIR and SWIR bands. A digital elevation model (DEM) was also loaded for the zone in question. As the bands did not always have the same resolution as the DEM, they were resized to have the finer resolution of 19m present in the DEM scenes. This was performed using a simple resizing using interpolation. Pansharpening would have been a preferable option in this case, although it was not implement due to time constraints.

After the data was properly loaded, it needed to be normalized to have better contrast. To do so, histogram normalization was performed. The images therefore had their bands normalized, although the DEM band was left unchanged. One of the images was used as a reference on which data was manually labelled to provide to a supervised machine learning algorithm. For other images to properly work in this algorithm, a similar histogram to the reference image was required. As such, other images had their histograms matched to the reference image, instead of simply normalized as was the case of the reference image.

Once the image had its histogram normalized or matched to the reference image, the relevant metrics could be computed from the band information. The NDWI, NDSI and MNDWI indices were therefore calculated using the green, red, NIR and SWIR bands. While it would not help identify glacial lakes, the NDSI was included as it would help identify glaciers and the hope was that this would help in the classification of glacial lakes. The DEM was then filtered with a small low-pass filter to smooth it out and the slope of the terrain was calculated using it. After the indices and slope were computed, they were combined together in a single matrix of dimensions equal to that of the original image.

After the indices and slope of the pixels were computed, they could be used in a supervised machine learning classifier. This first involved manually labelling data on the reference image. Lakes, mountainous regions, snowed regions and regions of mountain shadows were included to give the classifier as much information as possible for classification. By providing the classifier with more classes to help discern between pixels, it was hoped that it would perform better. The known classified pixels were then used in combination with the unlabelled image in a simple GLCM classifier. This resulted in a classification map with the pixels of the image being classified in the above-mentioned classes.

Once the classes were extracted, the lakes were isolated in a separate binary map. Some simple morphology filters were applied to clean up the map. This involved a closing operation to fix gaps in lakes (with the adverse effect of potentially removing some islands) and eliminating lakes smaller than a certain size. This removed a lot of stray pixels in shadowy regions that were being picked up as lakes. Although, as a consequence, lakes smaller than the threshold were no longer detected by the algorithm. The final result was a binary map containing the pixels that were identified as being lakes.

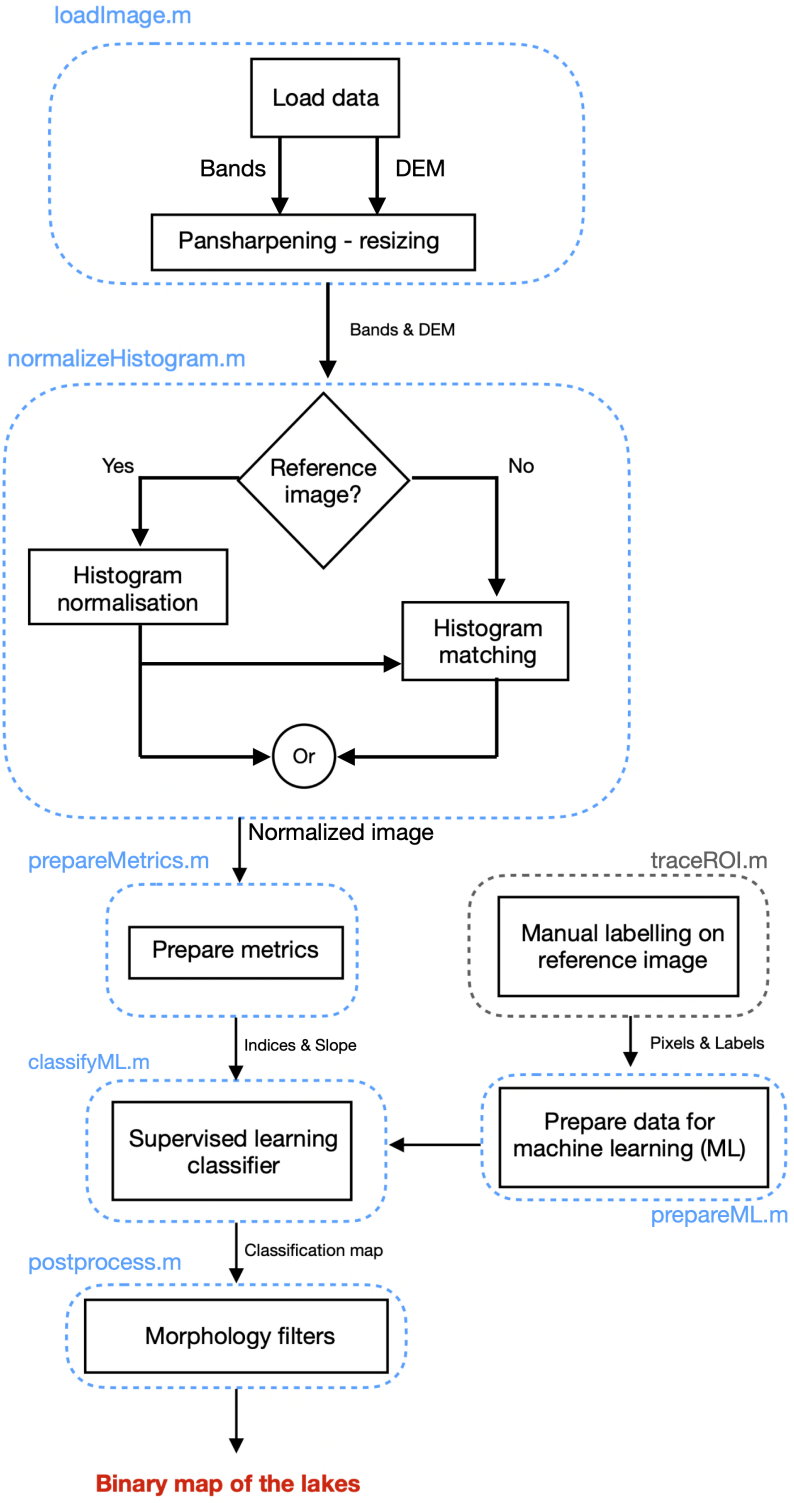

Figure 3.1: Flowchart of processing routine

4 Results

The method described in Section 3 was first applied to classify the reference image. Figure 4.1 shows both the original image with enhanced contrast on the left as well as the lake overlay on the right.

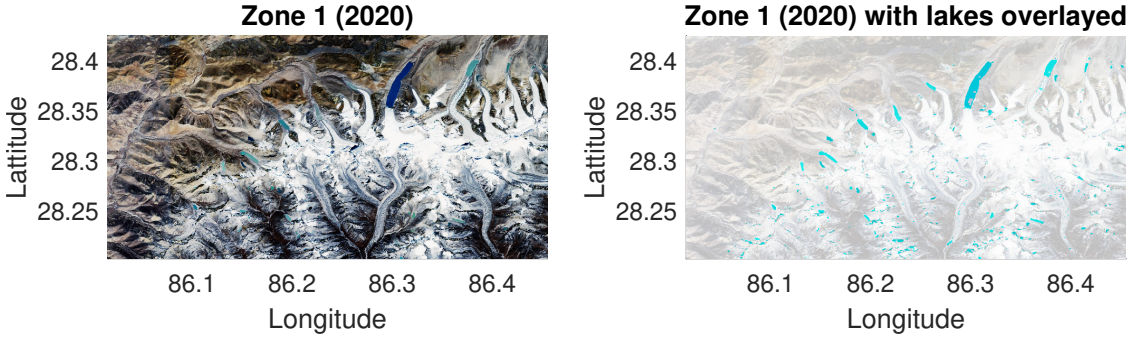

Figure 4.1: Satellite imagery of the reference image in 2020. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

As can be seen on the image, most lakes were successfully identified with the process used. It is quite remarkable how clearly the lakes are identified in the upper region of the image. Even smaller lakes that are hard to notice and properly label for humans on the snow range in the bottom right are properly identified. Nonetheless, the labelling is far from perfect. Several regions, especially mountain shadows, are wrongly classified as lakes. Additionally, some artefacts are present on some of the lakes, such as a hole present in the largest of the lakes.

As the image was partially labelled, some quantitative metrics could be computed from the labelling. The classification process is stochastic and therefore the values varied when the model was ran multiple times, but an overall accuracy of around 96% was obtained. This seems to indicate a very good model; however, accuracy alone is rarely a good indicator in these cases. Indeed, since the overwhelming majority of pixels of the image are not lakes, a naive classifier stating there are no lakes on the image would perform fairly well even if it would not be a very useful classifier! As such, the precision and recall are usually much better metrics to assess classification. These were of around 83% and 99%, respectively. These values coincide with the qualitative results. Indeed, a lot of regions were falsely classified as lakes (false positives) and lead to a lower precision. Though, the model almost always properly identified the lakes, with very few false negatives and an excellent recall to testify.

The method was then applied to the same region over multiple years. Most of the images are found in the appendix, but one of the scenes from 2014 is shown in Figure 4.2.

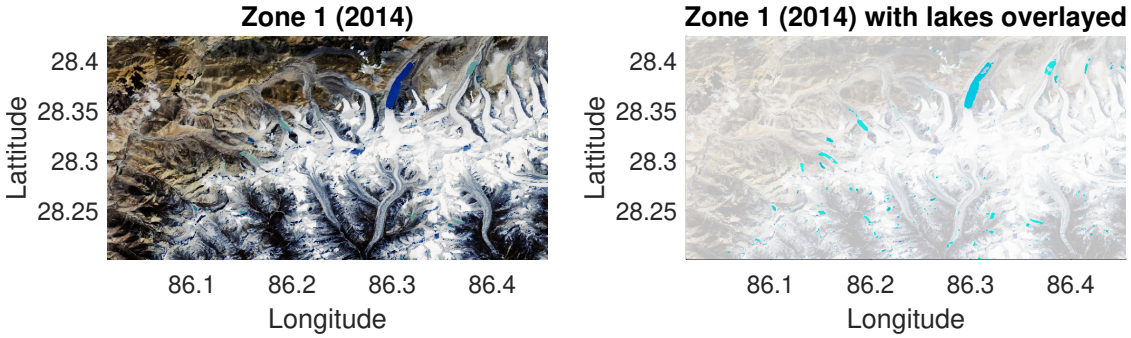

Figure 4.2: Satellite imagery of the same zone in 2014. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

Figure 4.2 shows that the classifier and method could easily be transferred to another scene. Here, the results were qualitatively very similar and perhaps even better, although there are no quantitative results to validate this claim as the image was not manually labelled. The classification does seem less sensitive than the reference image, with perhaps less false positives but also more false negatives in response. Indeed, two small lakes in the upper central region of the region are not properly detected in this scene.

Using the lake classification algorithm on the same region over multiple years allowed to evaluate the evolution of lakes in the region. The number of lakes, the total area and the average area per lake were computed for each scene and are shown in Figure 4.3.

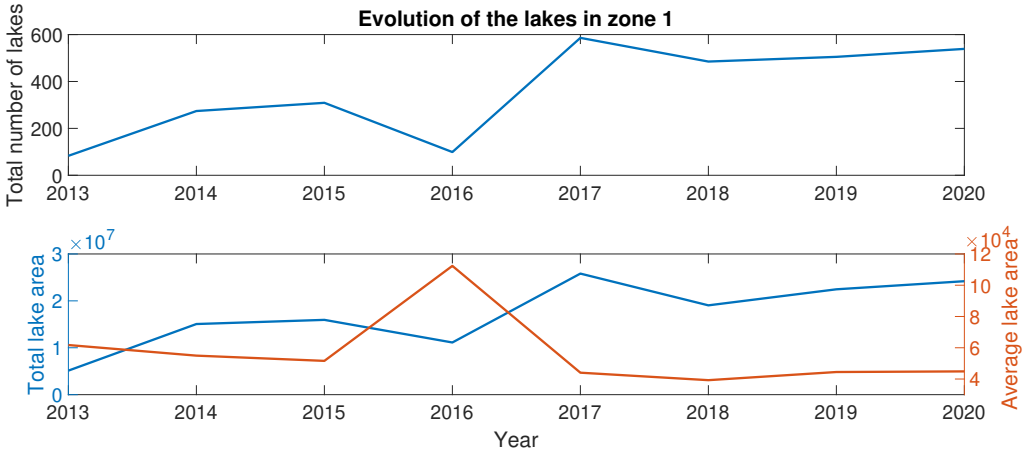

Figure 4.3: Evolution of the lakes in the reference region

Figure 4.3 seems to indicate a slight upward trend in the number of lakes and the total lake area in the region, in line with [1]. Proper care must however be taken to avoid making bold claims with the data collected. The lake classification is far from perfect and does not properly identify all of the lakes. As such, any quantitative claims about the increase should not be made. On the other hand, the classification was equally as imperfect for all of the scenes and therefore each data point has a comparable error. Therefore, general trends should still be maintained and is seen with the increase in total lake area.

Finally, the method was applied to a different zone to ensure that the machine learning algorithm and model was transferable to a different region in the Himalayas. The scene is shown in Figure 4.4. A scene from the same region in a different year can also be seen in the appendix.

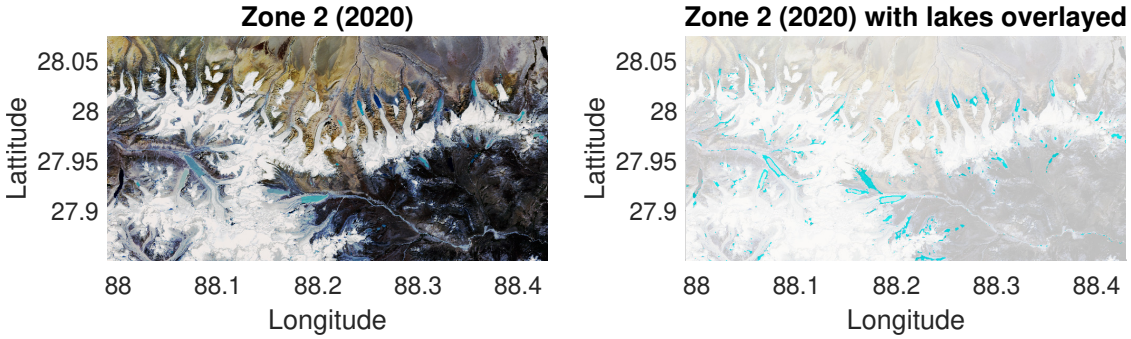

Figure 4.4: Satellite imagery of a different in 2014. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

The qualitative results are less good this time around. While most of the lakes are indeed detected, some have artefacts that are once again present in the form of holes. Additionally, many regions of mountain shadow are once again detected as lakes.

5 Discussion

The results seen in Section 4 show that while the method used allowed in a decent identification of the lakes, there is still some work to do before it can be as accurate as a human operator. Most of the improvements that can be made reside in improving the classification between mountain shadows and lakes. This problem affects most methods that automatically classify water bodies. Indeed, the classification map provided by Sentinel 2 imagery, as seen in Figure 5.1 is also plagued by mountain shadow zones being wrongly classified as lakes. Nonetheless, with proper and intelligent optimization, there are multiple aspects of the model used that can be changed in order to further improve the classification.

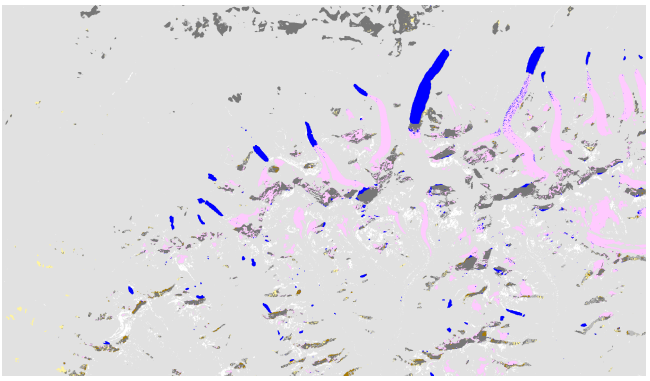

Figure 5.1: Classification map provided by the Sentinel 2 data. Notice the regions of mountain shadows in the center that are being wrongly classified as lakes.

The first method to improve the classification would be to improve the contrast between glacial lakes and mountain shadows. While they have a similar spectral signature to each other, they do not have an identical one. Indeed, this can be exploited to potentially increase the difference between the signatures by playing with the histograms or other methods or adjusting the bands. It is also possible that with a combination of other bands, or with other indices, a better contrast can be achieved which would ultimately lead in a better classification. For example, it would be possible to compute an alternative NDWI index using the blue band and the NIR band [7]. While this alternative NDWI index might not be a silver bullet, it is possible that with a clever combination of other bands and indices such as the one proposed, a better contrast can be achieved between shadows and lakes.

The next thing that can be done to improve classification would be to use the DEM information in a better way. In this report, the DEM was simply used to calculate the slope of the terrain and this was fed into the machine learning classifier without any other kind of processing. However, it is known that lakes should in theory have a slope of zero and therefore any region with a slope above a threshold (some studies suggest values of 5 or 10 degrees) could be excluded. This was attempted in this project, although it results in too many artefacts on lakes and was removed. It is likely that the slope value that was obtained was not properly calculated and resulted in lakes having a non-zero slope. An additional clever way to use the DEM information is to combine it with the sun elevation and azimuth angle at the acquisition time to create a shading map [8]. Pixels that are calculated to be in the mountain shadows can therefore be excluded using this technique.

As mountain shadows make lake detection more difficult because of similar spectral properties, another option is to try to negate them as much as possible. To avoid them, the ideal case would be to have no mountain shadows. This would be possible if the sun and satellite were both perfectly above the zone observed, though this does not happen in reality. In fact, at those latitudes, the zone is never perfectly overhead, even at the summer solstice (June 21th), since it is past the Tropic of Cancer. On top of this, June is generally in the monsoon period in the Himalayas and it is not possible to get clear satellite images without clouds. During this project, it was a sort of multi criteria optimization in this project to find clear images with less shadows as

possible. To deal with theses constraints related to mountains shadows, an idea is to do a multi day composition with images, eventually by using different satellites. The goal is to have images from different points of view to have a resulting image without mountain shadows thanks to a photogrametry process for example. It supposes that satellites can capture a zone at different time and angles.

To further help the classification process, more information can be used to help train the machine learning classifier. The method outlined in this report only used a single labelled zone. It is very likely that by classifying other zones that maybe have slightly different spectral characteristics the classification could be improved. This is further supported by the results in the second zone which are not as good as the ones in the first zone. Had this zone also been used to classify, it is likely that the classification process would have been better. The classifier can further be helped by being provided various other classes of data in the scenes, such as separating vegetation, rock, or other classes seen in the scenes. With more information for the classifier to work, a better segmentation can likely be achieved.

After improving the data used for the classification, the classifier itself can be optimized. Indeed, no significant optimization was done on the classifier side; however, using more advance machine learning models such as SVM or even deep learning can likely help with the image classification task. The metric used to train the model can also be tuned. As stated in Section 4, accuracy is a very poor indicator of the classification performance when lakes represent a small percentage of the total image area. The machine learning classifier could therefore be made to optimize precision or recall, depending on what is more preferable. A very good recall might be more desirable than a perfect precision. It is always easy to go and manually correct shadows that were wrongly classified as lakes, but finding lakes which were missed would essentially require manually going through the entire Himalayas once more. Therefore, a model with good recall such as the one presented in this report would be desirable and could do with some additional manual post-processing, as is often done in literature [7].

On another hand, it is important to remember that glacial lakes are part of an hydrological system and that their volumes and areas vary throughout the year. This has to be taken into account if the area variation is strictly compared. However in this project data were collected quite consistently in the hydrologic year (in October), and the goal is to observe an indicative trend between year, so the lakes were assumed to be comparable. Nonetheless, this is an important thing to consider when doing this kind of study.

While the results obtained in this report were far from perfect, they show the potential for what can be achieved with remote sensing methods and free data. As technology progresses, more and more methods can be used and machine learning can be used to make smart classifications of the data obtained, perhaps eventually better than humans can. Machine learning and automatic methods also have the advantage of being much more scalable than manual methods, as these are very time consuming and require trained experts to properly identify the regions. The methods outlined in this report could obviously be expanded upon to have better results, but once fine-tuned, would allow for a rapid identification of glacial lakes in the Himalayas and perhaps eventually other regions. The work done here could also be broadened to consider glaciers as well. Indeed, some proposed methods allow for an estimation of the glacier volumes. It would be very interesting to see if there is a strong correlation between the glacier volume decrease and the glacial lakes volume increase. It is likely that there is a strong inverse relationship between them, as most of the glacier melt will likely end up in glacial lakes.

Through methods using remote sensing, the reality of the regions in far and hard to access parts of the world such as the Himalayas can be studied. The growing number of glacial lakes in the Himalayas is cause for concern as it can cause devastating glacial lake outburst floods which can harm the local population. While the study of these regions might not necessarily directly help the people affected by them, it can perhaps raise awareness about the disastrous effects of global warming and provide yet another example as to why status quo must be challenged.

6 Appendix

The appendix contains the additional images not shown in the body of the report as well as the code that was used to generate the images. The entirety of the code as well as the scenes used for this report can be found at https://github.com/joshdboss/IPEO.

6.1 Additional images

This section contains the additional image labelling for the regions that were not shown in the body of the report.

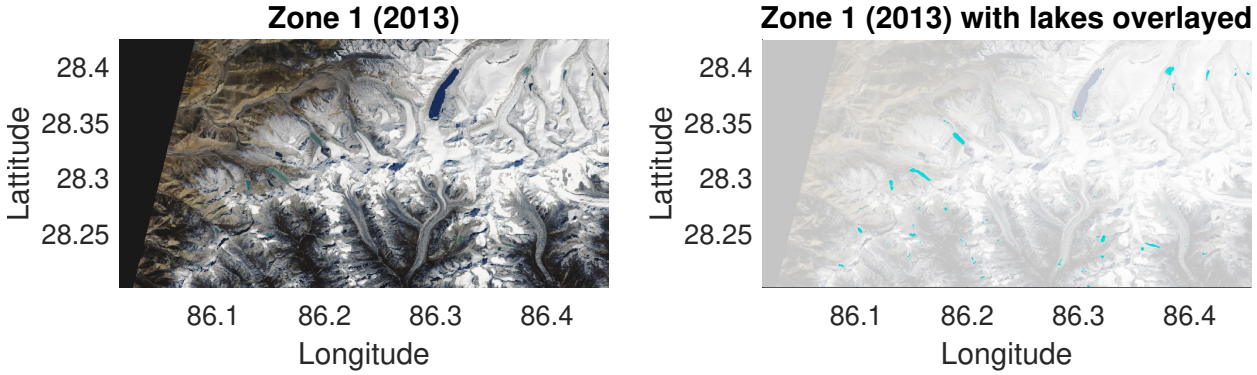

Figure 6.1: Satellite imagery of the reference zone in 2013. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

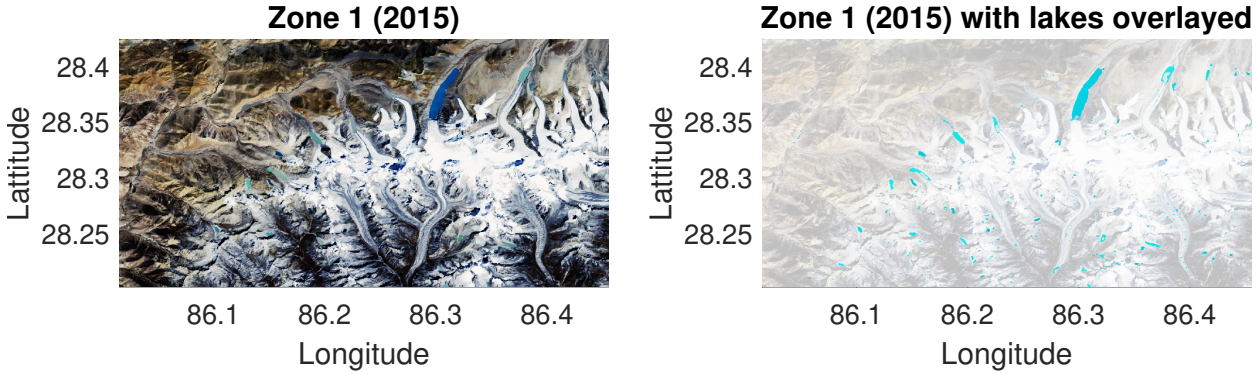

Figure 6.2: Satellite imagery of the reference zone in 2015. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

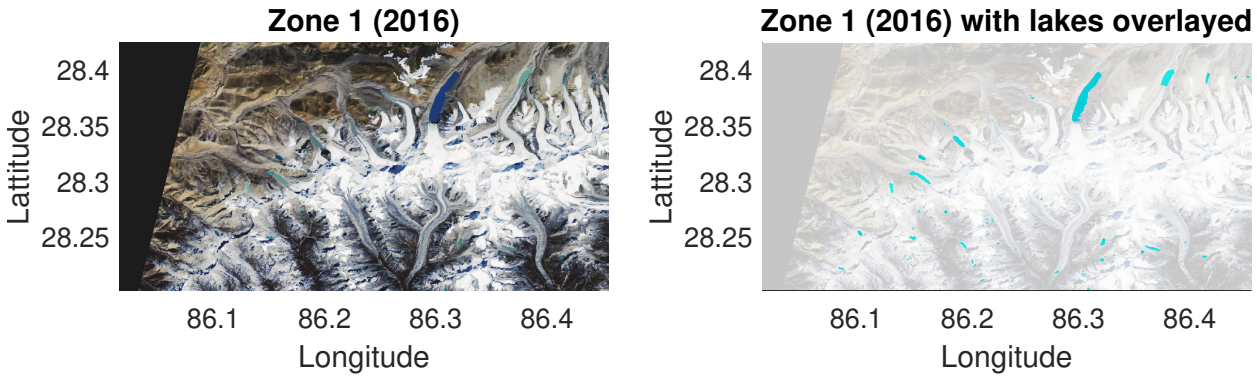

Figure 6.3: Satellite imagery of the reference zone in 2016. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

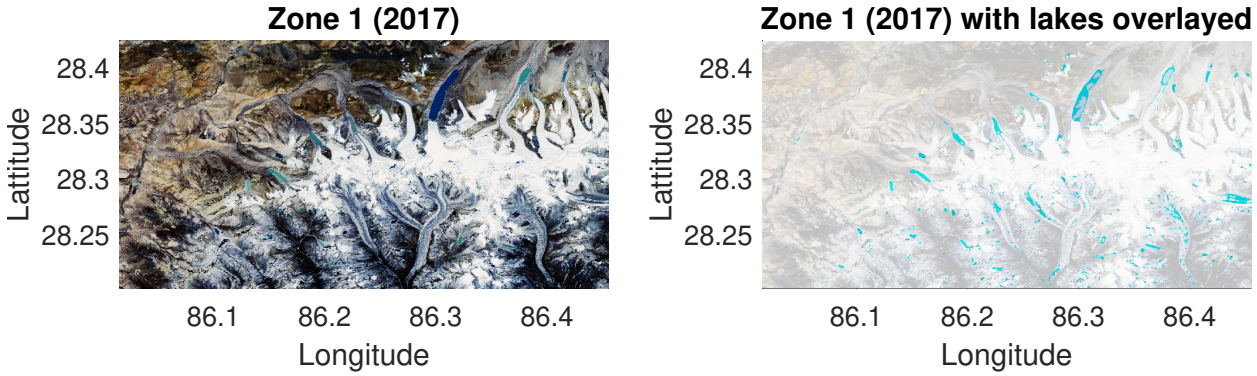

Figure 6.4: Satellite imagery of the reference zone in 2017. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

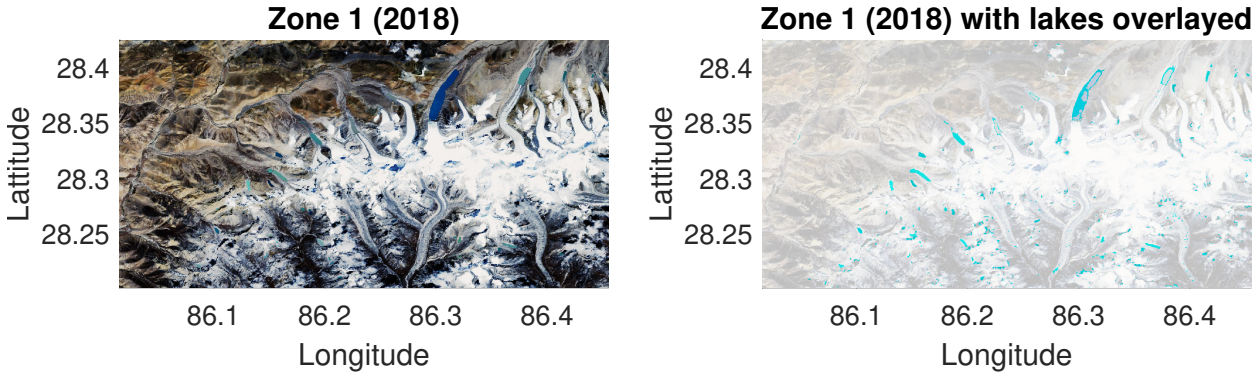

Figure 6.5: Satellite imagery of the reference zone in 2018. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

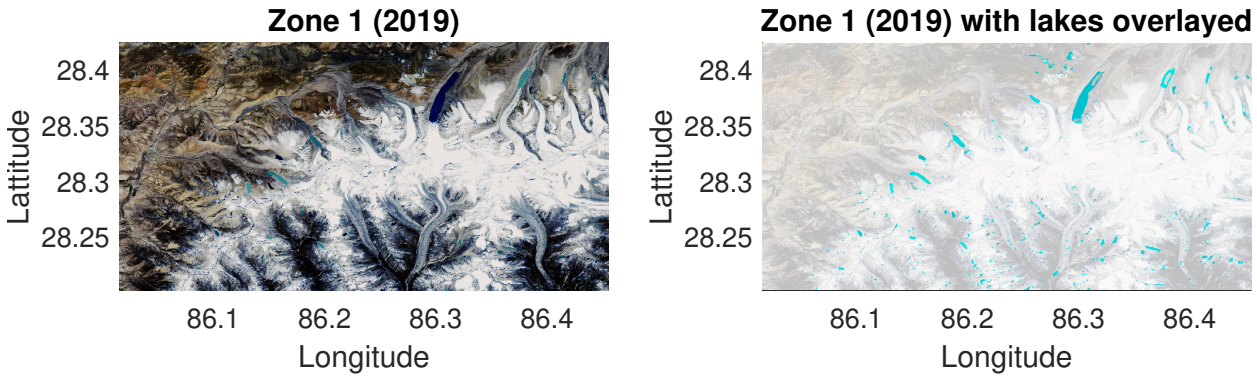

Figure 6.6: Satellite imagery of the reference zone in 2019. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

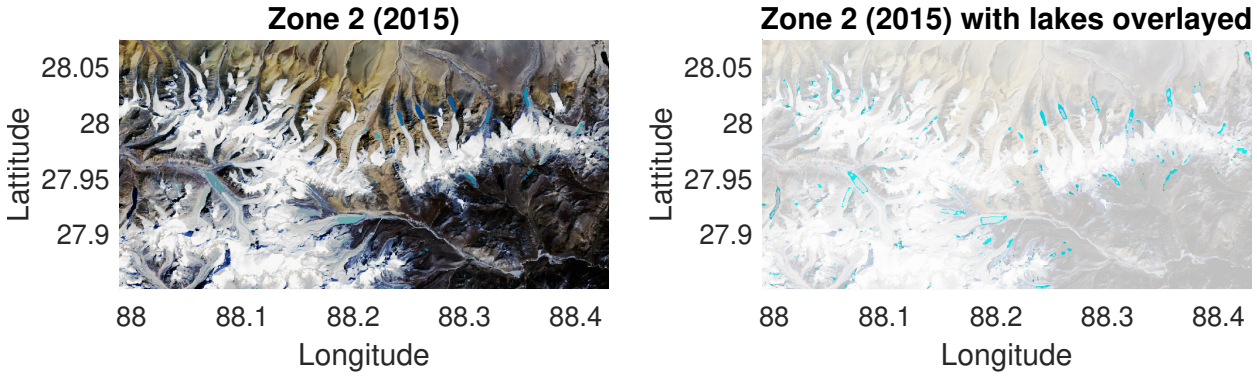

Figure 6.7: Satellite imagery of the other zone in 2015. True color enhanced contrast view (left) and a true color with the lakes overlayed (right).

6.2 Matlab code

This section contains the Matlab code that was used to generate the data in this report. As a reminder, the entirety of the code and data used is available at https://github.com/joshdboss/IPEO.

6.2.1 main.m

```
% MAIN Script for IPEO project
%% HEADER
% MAIN — main script for IPEO project
% Project title:
% Glacier retreat in the Himalayas and implications for local populations
% Authors: L o c Brouet, Joshua Cayetano—Emond
% SCIPER NUMBER: 266712, 309043
% Date: 2020—12—15


%% General settings ========================================================
clc;
clear;
%close all;

% plotting settings to be more visible
set(groot,'DefaultAxesFontSize',14)
set(groot,'DefaultLineLineWidth',1.5)
set(groot,'defaultfigureposition',[400 100 1000 400])


%% Define variables ========================================================
% pixel size in meters. obtained from the reference matrix of the DEM
pixelSize = 19.57;

% Location of the labelled data
indices_file = 'Data/zone1/index_better.mat';
labels_file = 'Data/zone1/labels_better.mat';

% File names for the reference image.
% The reference image is ALWAYS the first element in the imagData variable
imgData(1).blue_file = 'Data/zone1/2020/S2_WGS84/2020—10—29—S2_B02.tiff';
imgData(1).green_file = 'Data/zone1/2020/S2_WGS84/2020—10—29—S2_B03.tiff';
imgData(1).red_file = 'Data/zone1/2020/S2_WGS84/2020—10—29—S2_B04.tiff';
imgData(1).nir_file = 'Data/zone1/2020/S2_WGS84/2020—10—29—S2_B08A.tiff';
imgData(1).swir_file = 'Data/zone1/2020/S2_WGS84/2020—10—29—S2_B11.tiff';
imgData(1).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(1).zone = 1;
imgData(1).year = 2020;

% Zone 1 2014. Landsat
```

```
imgData(2).blue_file = 'Data/zone1/2014/L8_WGS84/2014_10_04_L8_B02.tiff';
imgData(2).green_file = 'Data/zone1/2014/L8_WGS84/2014_10_04_L8_B03.tiff';
imgData(2).red_file = 'Data/zone1/2014/L8_WGS84/2014_10_04_L8_B04.tiff';
imgData(2).nir_file = 'Data/zone1/2014/L8_WGS84/2014_10_04_L8_B05.tiff';
imgData(2).swir_file = 'Data/zone1/2014/L8_WGS84/2014_10_04_L8_B06.tiff';
imgData(2).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(2).zone = 1;
imgData(2).year = 2014;

% Zone 1 2015. Landsat
imgData(3).blue_file = 'Data/zone1/2015/L8_WGS84/2015_10_07_L8_B02.tiff';
imgData(3).green_file = 'Data/zone1/2015/L8_WGS84/2015_10_07_L8_B03.tiff';
imgData(3).red_file = 'Data/zone1/2015/L8_WGS84/2015_10_07_L8_B04.tiff';
imgData(3).nir_file = 'Data/zone1/2015/L8_WGS84/2015_10_07_L8_B05.tiff';
imgData(3).swir_file = 'Data/zone1/2015/L8_WGS84/2015_10_07_L8_B06.tiff';
imgData(3).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(3).zone = 1;
imgData(3).year = 2015;

% Zone 1 2016. Landsat
imgData(4).blue_file = 'Data/zone1/2016/L8_WGS84/2016_10_18_L8_B02.tiff';
imgData(4).green_file = 'Data/zone1/2016/L8_WGS84/2016_10_18_L8_B03.tiff';
imgData(4).red_file = 'Data/zone1/2016/L8_WGS84/2016_10_18_L8_B04.tiff';
imgData(4).nir_file = 'Data/zone1/2016/L8_WGS84/2016_10_18_L8_B05.tiff';
imgData(4).swir_file = 'Data/zone1/2016/L8_WGS84/2016_10_18_L8_B06.tiff';
imgData(4).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(4).zone = 1;
imgData(4).year = 2016;

% Zone 1 2017. Sentinel
imgData(5).blue_file = 'Data/zone1/2017/S2_WGS84/2017_10_10_S2_B02.tiff';
imgData(5).green_file = 'Data/zone1/2017/S2_WGS84/2017_10_10_S2_B03.tiff';
imgData(5).red_file = 'Data/zone1/2017/S2_WGS84/2017_10_10_S2_B04.tiff';
imgData(5).nir_file = 'Data/zone1/2017/S2_WGS84/2017_10_10_S2_B08A.tiff';
imgData(5).swir_file = 'Data/zone1/2017/S2_WGS84/2017_10_10_S2_B11.tiff';
imgData(5).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(5).zone = 1;
imgData(5).year = 2017;

% Zone 1 2018. Sentinel
imgData(6).blue_file = 'Data/zone1/2018/S2_WGS84/2018_10_05_S2_B02.tiff';
imgData(6).green_file = 'Data/zone1/2018/S2_WGS84/2018_10_05_S2_B03.tiff';
imgData(6).red_file = 'Data/zone1/2018/S2_WGS84/2018_10_05_S2_B04.tiff';
imgData(6).nir_file = 'Data/zone1/2018/S2_WGS84/2018_10_05_S2_B08A.tiff';
imgData(6).swir_file = 'Data/zone1/2018/S2_WGS84/2018_10_05_S2_B11.tiff';
imgData(6).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(6).zone = 1;
```

```
imgData(6).year = 2018;

% Zone 1 2019. Sentinel
imgData(7).blue_file = 'Data/zone1/2019/S2_WGS84/2019_10_10_S2_B02.tiff';
imgData(7).green_file = 'Data/zone1/2019/S2_WGS84/2019_10_10_S2_B03.tiff';
imgData(7).red_file = 'Data/zone1/2019/S2_WGS84/2019_10_10_S2_B04.tiff';
imgData(7).nir_file = 'Data/zone1/2019/S2_WGS84/2019_10_10_S2_B08A.tiff';
imgData(7).swir_file = 'Data/zone1/2019/S2_WGS84/2019_10_10_S2_B11.tiff';
imgData(7).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(7).name = 'Zone 1 (2019)';
imgData(7).zone = 1;
imgData(7).year = 2019;

% Zone 2 2015
imgData(8).blue_file = 'Data/zone2/2015/2015_10_09_L8_B02.tiff';
imgData(8).green_file = 'Data/zone2/2015/2015_10_09_L8_B03.tiff';
imgData(8).red_file = 'Data/zone2/2015/2015_10_09_L8_B04.tiff';
imgData(8).nir_file = 'Data/zone2/2015/2015_10_09_L8_B05.tiff';
imgData(8).swir_file = 'Data/zone2/2015/2015_10_09_L8_B06.tiff';
imgData(8).DEM_file = 'Data/zone2/DEM/DEM_raw.tiff';
imgData(8).zone = 2;
imgData(8).year = 2015;

% Zone 2 2020
imgData(9).blue_file = 'Data/zone2/2020/2020_10_11_S2_B02.tiff';
imgData(9).green_file = 'Data/zone2/2020/2020_10_11_S2_B03.tiff';
imgData(9).red_file = 'Data/zone2/2020/2020_10_11_S2_B04.tiff';
imgData(9).nir_file = 'Data/zone2/2020/2020_10_11_S2_B08A.tiff';
imgData(9).swir_file = 'Data/zone2/2020/2020_10_11_S2_B11.tiff';
imgData(9).DEM_file = 'Data/zone2/DEM/DEM_raw.tiff';
imgData(9).zone = 2;
imgData(9).year = 2020;

% Zone 1 2013
imgData(10).blue_file = 'Data/zone1/2013/L8_WGS84/2013_10_10_L8_B02.tiff';
imgData(10).green_file = 'Data/zone1/2013/L8_WGS84/2013_10_10_L8_B03.tiff';
imgData(10).red_file = 'Data/zone1/2013/L8_WGS84/2013_10_10_L8_B04.tiff';
imgData(10).nir_file = 'Data/zone1/2013/L8_WGS84/2013_10_10_L8_B05.tiff';
imgData(10).swir_file = 'Data/zone1/2013/L8_WGS84/2013_10_10_L8_B06.tiff';
imgData(10).DEM_file = 'Data/zone1/DEM_2015/DEM_raw.tiff';
imgData(10).zone = 1;
imgData(10).year = 2013;


%% Define functions ====================================================
imnorm = @(x) (x - min(x(:))) ./ (max(x(:)) - min(x(:)));

```

```
136
137 % %% Label data ========================================================
138 % % Label the data on the given region
139 % % This section is commented since the labelROI function is quite buggy
140 % % and labelling should be done beforehand.
141 % % Uncomment and run this section after having run the Define variables
142 % % section to manual label the data. Do not forget to save the index and
143 % % labels variables!
144 % [refImage, ~] = loadImage(imgData(1).blue_file, ...
145 %                   imgData(1).green_file, ...
146 %                   imgData(1).red_file, ...
147 %                   imgData(1).nir_file, ...
148 %                   imgData(1).swir_file, ...
149 %                   imgData(1).DEM_file);
150 % traceROI(refImage(:,:,[3,2,1]));
151
152
153 %% Load labelled data ========================================================
154 % Data must have already been labelled. See above section to manually label
155 % the data.
156 fprintf('Loading the labelled data.\n');
157 load(indices_file);
158 load(labels_file);
159
160
161 %% Loop through all images and find the lakes! ============================
162
163 for i = 1:length(imgData)
164     fprintf('========== STARTING TO WORK ON ZONE %d (%d)\n', ...
165         imgData(i).zone, imgData(i).year);
166
167     % Load the image
168     fprintf('---- Zone %d (%d): Loading image bands and DEM.\n', ...
169         imgData(i).zone, imgData(i).year);
170     [imgData(i).rawImage, imgData(i).refInfo, imgData(i).refMatrix] = ...
171         loadImage(imgData(i).blue_file, ...
172                   imgData(i).green_file, ...
173                   imgData(i).red_file, ...
174                   imgData(i).nir_file, ...
175                   imgData(i).swir_file, ...
176                   imgData(i).DEM_file);
177     fprintf('---- Zone %d (%d): Bands and DEM loaded.\n', ...
178         imgData(i).zone, imgData(i).year);
179
180     % for the reference image
181     if i == 1
182         % normalize histograms
```

```
183         fprintf('--- Zone %d (%d): Normalizing the histograms.\n', ...
184             imgData(i).zone, imgData(i).year);
185         imgData(i).normImage = normalizeHistogram(imgData(i).rawImage, NaN);
186         fprintf('--- Zone %d (%d): Histograms normalized.\n', ...
187             imgData(i).zone, imgData(i).year);
188     else
189         % for the others, perform histogram matching with reference image
190         fprintf('--- Zone %d (%d): Matching the histograms.\n', ...
191             imgData(i).zone, imgData(i).year);
192         imgData(i).normImage = normalizeHistogram(imgData(i).rawImage, ...
193             imgData(1).normImage);
194         fprintf('--- Zone %d (%d): Histograms matched.\n', ...
195             imgData(i).zone, imgData(i).year);
196     end
197
198     % Extract the metrics
199     fprintf('--- Zone %d (%d): Extracting the metrics.\n', ...
200         imgData(i).zone, imgData(i).year);
201     imgData(i).imageMetrics = prepareMetrics(imgData(i).normImage, ...
202         imgData(i).refInfo);
203     fprintf('--- Zone %d (%d): Metrics extracted.\n', ...
204         imgData(i).zone, imgData(i).year);
205
206     % prepare the machine learning data with the reference image
207     if i == 1
208         fprintf('--- Zone %d (%d): Preparing machine learning data on reference image.\n', ...
209             imgData(i).zone, imgData(i).year);
210         [data_train_sc, label_train, data_valid_sc, label_valid, ...
211             dataMax, dataMin] = prepareML(imgData(i).imageMetrics, ...
212             index, labels);
213         fprintf('--- Zone %d (%d): Data prepared.\n', ...
214             imgData(i).zone, imgData(i).year);
215
216         % compute the accuracy on training data
217         fprintf('--- Zone %d (%d): Computing training accuracy.\n', ...
218             imgData(i).zone, imgData(i).year);
219         [~, train_acc, lake_train_precision, lake_train_recall] = ...
220             classifyML(data_train_sc, label_train, false, data_train_sc, ...
221             label_train, dataMax, dataMin);
222         fprintf('--- Zone %d (%d): Training accuracy is of %.3f %%.\n', ...
223             imgData(i).zone, imgData(i).year, train_acc * 100);
224         fprintf('--- Zone %d (%d): Training lake precision is of %.3f %%.\n', ...
225             imgData(i).zone, imgData(i).year, lake_train_precision * 100);
226         fprintf('--- Zone %d (%d): Training lake recall is of %.3f %%.\n', ...
227             imgData(i).zone, imgData(i).year, lake_train_recall * 100);
228
229         % compute the accuracy and statistics on validation data
```

```
        fprintf('--- Zone %d (%d): Computing validation accuracy.\n', ...
            imgData(i).zone, imgData(i).year);
        [~, val_acc, lake_val_precision, lake_val_recall] = ...
            classifyML(data_valid_sc, label_valid, false, data_train_sc, ...
            label_train, dataMax, dataMin);
        fprintf('--- Zone %d (%d): Validation accuracy is of %.3f %%.\n', ...
            imgData(i).zone, imgData(i).year, val_acc * 100);
        fprintf('--- Zone %d (%d): Validation lake precision is of %.3f %%.\n', ...
            imgData(i).zone, imgData(i).year, lake_val_precision * 100);
        fprintf('--- Zone %d (%d): Validation lake recall is of %.3f %%.\n', ...
            imgData(i).zone, imgData(i).year, lake_val_recall * 100);
    end

    % Classify the images
    fprintf('--- Zone %d (%d): Classifying the image.\n', ...
        imgData(i).zone, imgData(i).year);
    [imgData(i).classMap, ~] = classifyML(imgData(i).imageMetrics, ...
        NaN, true, data_train_sc, label_train, dataMax, dataMin);
    fprintf('--- Zone %d (%d): Image classified.\n', ...
        imgData(i).zone, imgData(i).year);

    % There are multiple classes in the labelled data to help the model
    % work better. However, only the lakes (label 1) are of interest
    % Therefore, discard the other classes.
    imgData(i).lakes = imgData(i).classMap == 1;

    % Post process the data
    fprintf('--- Zone %d (%d): Post-processing the image classification.\n', ...
        imgData(i).zone, imgData(i).year);
    imgData(i).processedLakes = postprocess(imgData(i).lakes);
    fprintf('--- Zone %d (%d): Image classified.\n', ...
        imgData(i).zone, imgData(i).year);

    % Get interesting data on the lakes
    imgData(i).numLakes = length(regionprops(imgData(i).processedLakes));
    imgData(i).lakeArea = ...
        sum(cat(1, regionprops(imgData(i).processedLakes).Area)) ...
        * pixelSize ^ 2;
    imgData(i).averageLakeArea = imgData(i).lakeArea / imgData(i).numLakes;
    fprintf('--- Zone %d (%d): Number of lakes: %d.\n', ...
        imgData(i).zone, imgData(i).year, imgData(i).numLakes);
    fprintf('--- Zone %d (%d): Total lake area: %.1f m^2.\n', ...
        imgData(i).zone, imgData(i).year, imgData(i).lakeArea);
    fprintf('--- Zone %d (%d): Average lake area: %.3f m^2.\n', ...
        imgData(i).zone, imgData(i).year, imgData(i).averageLakeArea);

    % Plot everything
```

```
    plotOverlay(imgData(i).normImage(:,:,[3,2,1]), ...
        imgData(i).processedLakes, imgData(i).refMatrix, ...
        0.75, sprintf('Zone %d (%d)', imgData(i).zone, imgData(i).year))


    fprintf('========== DONE WITH ZONE %d (%d)\n', ...
        imgData(i).zone, imgData(i).year)

end


%% Generate area comparison graphs
uniqueZones = unique(cat(1,imgData.zone))';

for zoneName = uniqueZones
    data = imgData(cat(1,imgData.zone) == zoneName); % scenes in right zone
    years = cat(1,data.year); % years for each of the scenes

    % lake information
    numLakes = cat(1,data.numLakes);
    lakeAreas = cat(1,data.lakeArea);
    averageLakeArea = cat(1,data.averageLakeArea);

    % sort everything according to chronological order
    [years, idx] = sort(years);
    numLakes = numLakes(idx);
    lakeAreas = lakeAreas(idx);
    averageLakeArea = averageLakeArea(idx);

    figure
    subplot(2,1,1)
    plot(years,numLakes);
    title(sprintf('Evolution of the lakes in zone %d', zoneName))
    ylabel('Total number of lakes');
    subplot(2,1,2)
    yyaxis left
    plot(years,lakeAreas);
    ylabel('Total lake area');
    yyaxis right
    plot(years,averageLakeArea);
    ylabel('Average lake area');
    xlabel('Year');

end
```

6.2.2 classifyML.m

```
function [classMap, accuracy, lakePrecision, lakeRecall] = ...
    classifyML(classImage, classLabels, ...
    reshp, trainedData, trainedLabels, dataMax, dataMin)
%Classifies the given image using a known labelled dataset
%   Using a known labelled image, classify another image. First
%   starts by putting the unknown image in the right format, scales it
%   and then performs the machine learning classification.
%
%INPUTS
%   classImage (M x N): The image to classify
%   classLabels (M x N): If known, the labels of the image
%   reshp (boolean): Whether or not to reshape the matrix. This should be
%   false for the validation data which is already reshaped, and true for
%   any other case
%   trainedData : The data of the pixels that have already been trained.
%       This data should already be prepared in the right format.
%   trainedLabels : The labels for each of the pixels in trainedData
%   dataMax : max values of the trained data to rescale the unknown image
%   dataMin : mib values of the trained data to rescale the unknown image
%
%OUTPUTS
%   classMap (M x N): The labels of each of the pixels on the image
%   accuracy (float) : accuracy of the model, if labels given
%   lakePrecision (float): the classification precision of the lakes
%   lakeRecall (float): the classification recall of the lakes


% Reshape the image into a 2d matrix if desired
if reshp
    data = reshape(classImage, size(classImage,1)*size(classImage,2), ...
        size(classImage,3));
else
    data = classImage;
end

% Rescale the data according to how the labelled data was scaled
data_sc = classificationScaling(double(data), dataMax, dataMin, 'std');
% Classify the image
classMap = classify(data_sc, trainedData, trainedLabels, 'linear');

% Reshape the classMap back into the shape of the image if desired
if reshp
    classMap = reshape(classMap,size(classImage,1),size(classImage,2));
end

```

```
% if there are labels, compute the accuracy
if ~all(isnan(classLabels),'all')
    C = confusionmat(classLabels, classMap);
    accuracy = sum(diag(C)) / sum(C,'all');
    lakePrecision = C(2,2) / sum(C(2,:));
    lakeRecall = C(2,2) / sum(C(:,2));
else
    accuracy = NaN;
    lakePrecision = NaN;
    lakeRecall = NaN;
end

end
```

6.2.3 getIndices.m

```
function [NDWI, NDSI, MNDWI] = getIndices(G, R, NIR, SWIR)
%Gets the NDWI, NDSI and MNDWI indices given the proper bands
%   Finds the indices of interest given an image and its various bands.
%   Some bands can be omitted by being sent as NaN to the function and
%   the relevant indices will not be calculated.
%
%INPUTS
%   G (M x N) : The green band of the image
%   R (M x N) : The red band of the image
%   NIR (M x N) : The NIR band of the image
%   SWIR (M x N) : The SWIR band of the image
%
%OUTPUTS
%   NDWI (M x N): Matrix of the NDWI index of the image
%   NDSI (M x N): Matrix of the NDWI index of the image
%   MNDWI (M x N): Matrix of the NDWI index of the image

% get the indices
NDWI = (NIR - SWIR) ./ (NIR + SWIR);
NDSI = (SWIR - R) ./ (SWIR + R);
MNDWI = (G - SWIR) ./ (G + SWIR);

% remove NaNs from division by zero errors
NDWI(isnan(NDWI)) = 0;
NDSI(isnan(NDSI)) = 0;
MNDWI(isnan(MNDWI)) = 0;

end
```

6.2.4 loadImage.m

```
function [Image, ImageRef, refMat] = ...
    loadImage(fileName_blue, ...
              fileName_green, ...
              fileName_red, ...
              fileName_nir, ...
              fileName_swir, ...
              fileNameDEM)
%Loads an image from the given band files and DEM and pansharpens
%   Loads the given .tiff raster files and stacks the images properly into
%   an image variable for the 5 bands of interest as well as a DEM variable
%   Also outputs the reference information
%
%INPUTS
%   fileName_band2 (string) : Path to the image for band 2
%   fileName_band3 (string) : Path to the image for band 3
%   fileName_bands456 (string) : Path to the image for bands 4 to 6
%   fileNameDEM (string) : Path to the DEM image
%
%OUTPUTS
%   Image (Mimg x Nimg x 5) : Matrix containing the image of interest of
%   dimensions Mimg x Nimg pixels containing 4 bands + DEM
%   ImageRef : The image mapping reference information
%   refMat : reference matrix of the image

% load the 4 bands
% reference information is not saved, since we will be using the reference
% information for DEM which has the highest resolution
[bands(:,:,1), ~] = readgeoraster(fileName_blue);
[bands(:,:,2), ~] = readgeoraster(fileName_green);
[bands(:,:,3), ~] = readgeoraster(fileName_red);
[bands(:,:,4), ~] = readgeoraster(fileName_nir);
[bands(:,:,5), ~] = readgeoraster(fileName_swir);

% load the DEM
[DEM, ImageRef] = readgeoraster(fileNameDEM);
refMat = geotiffinfo(fileNameDEM).RefMatrix;

% Transform all images into a double precision number
bands = im2double(bands);
DEM = im2double(DEM);

% Performs the pansharpening. For now uses naive resizing
bands = imresize(bands, size(DEM));

% concatenates the bands and DEM to get the final image
```

```
46 Image = cat(3,bands,DEM);
47
48 end
```

6.2.5 normalizeHistogram.m

```
function [normImage] = normalizeHistogram(rawImage, refImage)
%Normalizes the histograms and optionally performs histogram matching
%   Normalizes the histograms of each band (except the last, which contains
%   the DEM) of the rawImage. If a refImage is provided, then the
%   histograms of the rawImage are adjusted to match those of the refImage.
%
%INPUTS
%   rawImage (M x N x B): raw image on which the histogram should be normed
%   refImage (M x N x B): normalized reference image that will be used to
%   match the histograms of the raw image. Optional, input NaN to simply
%   normalize rawImage.
%
%OUTPUTS
%   normImage (M x N x B): normalized and optionally matched image

% initialize the normalized image
normImage = zeros(size(rawImage));
normImage(:,:,end) = rawImage(:,:,end);

% go through all the bands (except the last, which contains the DEM)
for i = 1:size(rawImage,3)-1
    raw_uint8 = im2uint8(rawImage(:,:,i)); % get the raw image as a uint8

%     % adjust the image to remove top and bottom 1%
%     raw_uint8 = imadjust(raw_uint8, ...
%          stretchlim(raw_uint8(raw_uint8 ~= 0), ...
%                      [0.01 0.99]), [0,1], 1);

    % Compute the histogram, then CDF of the raw Image
    [counts_raw, ~] = imhist(raw_uint8,256);
    cdf_raw = cumsum(counts_raw,1) / sum(counts_raw,1);


    % check if there is a reference
    if all(isnan(refImage),'all')
        % if none, simply normalize the histogram
        raw_equ = cdf_raw(raw_uint8(:)+1);
    else % if there is one, perform histogram matching
        ref_uint8 = im2uint8(refImage(:,:,i)); % get the ref image as uint8

        % Compute the histogram, then CDF of the reference equalized image
        [counts_ref, ~] = imhist(ref_uint8, 256);
        cdf_ref = cumsum(counts_ref,1) / sum(counts_ref,1);

        % Build a look-up table between the two CDFs
```

```
        [cdf_ref_unique, id_unique] = unique(cdf_ref);
        LUT = interp1(cdf_ref_unique, id_unique, cdf_raw(id_unique));
        % remove duplicates in the CDF in interp1 function
        LUT2 = interp1(id_unique, LUT, 0:255,'linear','extrap');

        % Apply the LUT to the raw image
        raw_equ = uint8(LUT2(raw_uint8(:)+1));
        raw_equ = im2double(raw_equ);
    end

    % Put the mapped pixels back into the image
    normImage(:,:,i) = reshape(raw_equ,size(raw_uint8));
end

end
```

6.2.6 plotIndices.m

```
function plotIndices(NDWI, NDSI, MNDWI, refMat)
%Plots the indices and their normalized versions
%   Plots the NDWI, NDSI and MNDWI indices as well as their normalized
%   versions in a figure with subplots. The function automatically
%   determines which indices need to be plotted
%
%INPUTS
%   NDWI (M x N): Matrix of the NDWI index of the image
%   NDSI (M x N): Matrix of the NDWI index of the image
%   MNDWI (M x N): Matrix of the NDWI index of the image
%   refMat : reference matrix

% see which indices to plot
plotNDWI = not(all(isnan(NDWI),'all'));
plotNDSI = not(all(isnan(NDSI),'all'));
plotMNDWI = not(all(isnan(MNDWI),'all'));
nbIdx = plotNDWI + plotNDSI + plotMNDWI; % number indices to plot

% function to normalize an image between 0 and 1
imnorm = @(x) (x - min(x(:))) ./ (max(x(:)) - min(x(:)));

% plot only if indices present
if nbIdx ~= 0
    % create the empty figure
    figure()
    set(gcf,'Position',[100 -100 450*nbIdx 600])

    if plotNDWI
        % Unmodified NDWI index
        subplot(2,nbIdx,1);
        mapshow(NDWI, refMat)
        axis equal tight
        xlabel('Easting [m]')
        ylabel('Northing [m]')
        title('NDWI');

        % Normalized NDWI index
        subplot(2,nbIdx,nbIdx + 1);
        mapshow(imnorm(NDWI), refMat)
        axis equal tight
        xlabel('Easting [m]')
        ylabel('Northing [m]')
        title('Normalized NDWI');
    end

```

```
    if plotNDSI
        % Unmodified NDSI index
        subplot(2,nbIdx,1 + plotNDWI);
        mapshow(NDSI, refMat)
        axis equal tight
        xlabel('Easting [m]')
        ylabel('Northing [m]')
        title('NDSI');

        % Normalized NDSI index
        subplot(2,nbIdx,nbIdx + 1 + plotNDWI);
        mapshow(imnorm(NDSI), refMat)
        axis equal tight
        xlabel('Easting [m]')
        ylabel('Northing [m]')
        title('Normalized NDSI');
    end

    if plotMNDWI
        % Unmodified NDSI index
        subplot(2,nbIdx,1 + plotNDWI + plotNDSI);
        mapshow(MNDWI, refMat)
        axis equal tight
        xlabel('Easting [m]')
        ylabel('Northing [m]')
        title('MNDWI');

        % Normalized NDSI index
        subplot(2,nbIdx,nbIdx + 1 + plotNDWI + plotNDSI);
        mapshow(imnorm(MNDWI), refMat)
        axis equal tight
        xlabel('Easting [m]')
        ylabel('Northing [m]')
        title('Normalized MNDWI');
    end
end

end
```

6.2.7 plotOverlay.m

```
function plotOverlay(background, overlay, refmat, ...
    transparency, plot_title)
%Overlays a labelled image on the background image according to a colormap
%   Plots a binary image on a background image using provided colormap
%   Each label in the overlay is associated a different color
%
%INPUTS
%   background (M x N x 3): Background image/map. Should contain 3 bands
%   overlay (M x N): binary map to overlay on top
%   refmat : The reference information to be able to map everything
%   transparency : Transparency to use for the overlay. Between 0 (fully
%   transparent) and 1 (fully opaque).

% get an rgb representation of the overlay
RGB  = im2double(label2rgb(overlay));

% plot the map
figure()
set(gcf,'Position',[100 -100 1000 500])

% without overlay
subplot(1,2,1);
mapshow(background, refmat)
axis equal tight
xlabel('Longitude')
ylabel('Lattitude')
title(plot_title);

% with overlay
subplot(1,2,2);
mapshow((1 - transparency) * background + transparency * RGB, refmat)
axis equal tight
xlabel('Longitude')
ylabel('Lattitude')
title(strcat(plot_title,' with lakes overlayed'));

end
```

6.2.8 postprocess.m

```
function [processedLakeMap] = postprocess(rawLakeMap)
%Performs post-processing operations on the lake map obtained
%   Performs some post-processing on the binary lake map obtained.
%   As of now, includes morphology to fill gaps in lakes that are due to
%   artefacts in the ML process and removing lakes that are too small.
%
%INPUTS
%   rawLakeMap (M x N): Binary map of where the lakes are located
%
%OUTPUTS
%   processedLakeMap (M x N): New binary map of the lakes after processing
%   has been performed.

% Step 1. Perform morphology to fill gaps in lakes.
%   A non-reconstructive closing morpology filter is used. Reconstructive
%   might be better and preserve shape better, however it has been found to
%   not be as efficient/correct.

% Define the shape for the morphology. Disk as lakes do not have a prefered
% shape. Could be verified in a future project
SE = strel('disk',5); % 'diamond' 'square'
% Morphology works best with uint8
im_uint8 = uint8(255 * mat2gray(rawLakeMap));

% Performing Erosion. Not used
%Im_e = imerode(im_uint8,SE);
% Performing Dilation
Im_d = imdilate(im_uint8,SE);

% Reconstructive closing. Not used
% Im_cr = imcomplement(imreconstruct(imcomplement(Im_d), ...
%  imcomplement(im_uint8)));
% Non-reconstructive closing
Im_cl = imclose(im_uint8,SE);


% Step 2. Remove lakes that are too small.
% Articles were using 4 pixels with a 30m size.
% We have a 19m resolution, so will go a bit higher
minimum_lake_size = 7;
processedLakeMap = bwareaopen(Im_cl,minimum_lake_size);

end
```

6.2.9 prepareMetrics.m

```
function [imageMetrics] = prepareMetrics(rawImage, refMat)
%Takes a raw image (bands 3-6 and DEM) and extracts the desired metrics
%   Given a raw, pansharpened image with bands 3 to 6 and the DEM, extract
%   the relevant indices that will allow proper lake identification. These
%   are the NDWI, NDSI, MNDWI, bands 4 and 5 as well as the terrain slope.
%   These indices were found to have the best results for supervised ML.
%
%INPUTS
%   rawImage (M x N x 6): A pansharpened image with the histograms matched
%   containing blue, green, red, nir and swir bands as well as the DEM
%   refMat: Reference matrix of the image.
%
%OUTPUTS
%   imageMetrics (M x N x 6): The relevant indices to allow proper lake
%   identification. Contains the NDWI, NDSI, MNDWI of the image as well as
%   bands 4 and 5 and finally the terrain slope.


% % Step 1. Increase contrast in the image bands
% % Pre-allocate matrix of the new band values for speed
% % Excludes the last index, since we are not adjusting DEM values
% adjusted_bands = zeros(size(rawImage(:,:,1:end-1)));
% gamma = 1;
% for i = 1:size(rawImage,3)-1
%     current_band = rawImage(:,:,i);
%     % remove top and bottom 1%
%     adjusted_bands(:,:,i) = imadjust(current_band, ...
%         stretchlim(current_band(current_band ~= 0), ...
%                     [0.01 0.99]), [0,1], gamma);
% end

% Step 1. Isolate the bands for verbosity
green_band = rawImage(:,:,2);
red_band = rawImage(:,:,3);
nir_band = rawImage(:,:,4);
swir_band = rawImage(:,:,5);


% Step 2. Get the water and snow indices from the bands
[ndwi, ndsi, mndwi] = ...
    getIndices(green_band, red_band, nir_band, swir_band);


% Step 3. Filter the DEM
sigma = 2;
```

```
h3 = fspecial('gaussian', 3, sigma);
filtered_DEM = imfilter(rawImage(:,:,end),h3);


% Step 4. Get the slope
[aspect,slope,gradN,gradE] = gradientm(filtered_DEM, refMat);


% Step 5. Assemble the matrix of the relevant image data to return
%imageMetrics(:,:,1:5) = rawImage(:,:,1:end-1);
imageMetrics(:,:,1) = ndwi;
imageMetrics(:,:,2) = ndsi;
imageMetrics(:,:,3) = mndwi;
imageMetrics(:,:,4) = slope;

end
```

6.2.10 prepareML.m

```
function [data_train_sc, label_train, data_valid, label_valid,...
    dataMax, dataMin] = prepareML(refImage, indices, labels)
%Prepares the data for a supervised ML classification
%   Prepares the data from the refImage according to the indices and labels
%   obtained from the plotRoi function. Splits the data into train and
%   validation data. Also returns the training scale factor so it can be
%   applied on another dataset.
%
%INPUTS
%   refImageData (M x N x D): Relevant data to classify the M x N image.
%   Will usually contain the indices as well as slope
%   indices (Nb_poygons x 1): Structure containing the indices of
%   pixels in each of the labelled polygons/roi
%   labels (Nb_poygons x 1): Labels of the polygons
%
%OUTPUTS
%   data_train_sc : the training data, ready to use for classify()
%   label_train : the labels of the training data
%   data_valid : the validation data. NOT YET SCALED
%   label_valid : the labels of the validation data
%   dataMax : the maximum of the data_train range to scale another dataset
%   dataMin : the minimum of the data_train range to scale another dataset


% PREPARE DATA ============================================================

% Reshape the image into a 2d matrix
data = reshape(refImage, size(refImage,1)*size(refImage,2), ...
    size(refImage,3));

% Get pixels from each class mask (polygons)
data_roi = data(cell2mat(indices),:);

% concatenate the vector of labels
label_roi = [];

for c = 1:length(labels) % for each polygon
    % Create a vector with the label of the polygon class.
    label_roi = [label_roi; repmat(labels(c),size(indices{c},1),1)];
end

% Split into training and testing samples to evaluate performances
splt = 0.2;
trainID = randperm(length(label_roi), round(splt*length(label_roi)));
allID = [1:length(label_roi)];
```

```
testID = allID(~ismember(allID,trainID));

% Subsample the training and the validation (test) data + labels
data_train = data_roi(trainID,:);
label_train = label_roi(trainID);

data_valid = data_roi(testID,:);
label_valid = label_roi(testID);

[data_train_sc, dataMax, dataMin] = ...
    classificationScaling(double(data_train), [], [], 'std');
```

References

[1] G. Zhang, T. Yao, H. Xie, W. Wang, and W. Yang, "An inventory of glacial lakes in the Third pole region and their changes in response to global warming," *Global and Planetary Change*, vol. 131, pp. 148 – 157, 2015.

[2] S. Aher, "Remote Sensing Technique for Monitoring the Glacier Retreating Process and Climatic Changes Study," *Indian Streams Research Journal*, vol. II, pp. 1–5, 09 2012.

[3] A. Ashraf, M. Rustam, S. I. Khan, M. Adnan, and R. Naz, "Remote Sensing of the Glacial Environment Influenced by Climate Change," in *Environmental Applications of Remote Sensing* (M. Marghany, ed.), ch. 4, Rijeka: IntechOpen, 2016.

[4] A. V. Kulkarni, I. M. Bahuguna, B. P. Rathore, S. K. Singh, S. S. Randhawa, R. K. Sood, and S. Dhar, "Glacial retreat in himalaya using indian remote sensing satellite data," *Current Science*, vol. 92, no. 1, pp. 69–74, 2007.

[5] W. Weicai, X. Yang, G. Yang, L. Anxin, and Y. Tandong, "Rapid expansion of glacial lakes caused by climate and glacier retreat in the central himalayas," *Hydrological Processes*, vol. 29, no. 6, pp. 859–874, 2015.

[6] W. Xin, L. Shiyin, G. Wanqin, Y. Xiaojun, J. Zongli, and H. Yongshun, "Using Remote Sensing Data to Quantify Changes in Glacial Lakes in the Chinese Himalaya," *Mountain Research and Development*, vol. 32, no. 2, pp. 203 – 212, 2012.

[7] X. WANG, K. CHAI, S. LIU, J. WEI, Z. JIANG, and Q. LIU, "Changes of glaciers and glacial lakes implying corridor-barrier effects and climate change in the hengduan shan, southeastern tibetan plateau," *Journal of Glaciology*, vol. 63, no. 239, p. 535–542, 2017.

[8] N. Khadka, G. Zhang, and S. Thakuri, "Glacial Lakes in the Nepal Himalaya: Inventory and Decadal Dynamics (1977–2017)," *Remote Sens.*, 2018.

[9] H. Li, D. Mao, X. Li, Z. Wang, and C. Wang, "Monitoring 40-Year Lake Area Changes of the Qaidam Basin, Tibetan Plateau, Using Landsat Time Series," *Remote Sens.*, 2019.

[10] "Landsat 8." https://www.usgs.gov/core-science-systems/nli/landsat/landsat-8?qt-science_support_page_related_con=4#qt-science_support_page_related_con. Accessed: 2021-01-06.

[11] U. G. Survey, "Landsat—earth observation satellites (ver. 1.2, april 2020): U.s. geological survey fact sheet," no. 4, p. 2015–3081, 2016.

[12] "Sentinel Hub EO Browser." https://apps.sentinel-hub.com/eo-browser/. Accessed: 2021-01-06.

[13] "Sentinel 2." https://sentinel.esa.int/web/sentinel/missions/sentinel-2. Accessed: 2021-01-06.

[14] F. de Morsier, "Image processing for earth observation: 6 - digital elevation models," 2020.

[15] F. de Morsier, "Image processing for earth observation: 1b - sensors," 2020.