Lebanese University
Faculty of Sciences
Dean

Master Thesis

In Order to Obtain the

Professional Master in

GIS and Data Science

Presented and Defended by:

Mohammad Ali Fneich

On November 15, 2022

Title:

Agricultural Land Assessment

Supervisor:

Dr. Kifah Raafat Tout

Reviewers:

Dr. Ghassan Karam Elnemr

Dr. Hassan Mohamad Tout

Lebanese University-Faculty of sciences Academic Year 2021-2022

ACKNOWLEDGMENTS

First and above all, I praise God, the almighty for providing me with this opportunity and granting me the capability to proceed successfully. This thesis appears in its current form, due to the assistance and guidance of several people. I would therefore like to offer my sincere thanks to all of them.

I would like to express my special thanks of gratitude to my teacher Ms. Gretta Kelzi as well as our supervisor of this master's program Dr.Kifah Tout who gave me the golden opportunity to do this wonderful project on the topic (Agricultural land assessment), which also helped me in doing a lot of Research and I came to know about so many new things.

I give many thanks to the faculty of Sciences at Lebanese university, the department of computer science, administration, staff, and lecturers for their kindness and cooperation with me during my studies at the university. Not forgetting Esri Lebanon team at Khatib and Alami for the hard training they offered during this master's program.

Finally, my deep and sincere gratitude to my family for their continuous and unparalleled love, help, and support. I am grateful to my sisters and my brother for always being there for me.

ABSTRACT

Agriculture has great economic importance in each country, especially in Lebanon, which suffers from a severe economic crisis. In Addition, climate change has a significant impact on food security. Providing reliable, real-time information on crop development is essential to support agriculture. It should look towards emerging technologies to find solutions to overcome some of the challenges it faces.

One of the most recent developments is the increase in the use of small-unmanned aerial vehicles (UAVs) and drones that provide very high-resolution multispectral and hyper-spectral aerial and satellite imagery. Due to the technical, financial, and security obstacles, we decided to use remote sensing and satellite images instead of drones.

Satellite remote sensing technologies have high potential in the evaluation of applications in land conditions and can facilitate the optimal planning of agricultural sectors. However, misleading land selection decisions reduce crop yields and increase production-related costs for farmers. Therefore, the purpose of the project was to develop a land suitability assessment system using satellite images and spatial analysis models.

This project uses satellite remote sensing methods, deep learning algorithms, and spatial analysis tools in GIS platforms for agricultural land assessment to select areas of arable land with the potential to increase agricultural production.

Table of Contents

Table of Figures

Table of Equations

CHAPTER 1

INTRODUCTION

In current and future climate scenarios, the resilience and productivity of agricultural systems will be increasingly compromised. In this changing context, the agricultural sector faces many hurdles.

In most cases, land assessment by soil sampling and analysis is expensive and challenging. In addition, there is a lack of datasets in some areas of developing regions where assessments of land suitability are changed. In addition, recent datasets of those geographic information systems have limitations, especially regarding land uses, drainage, and lack of soil sampling information.

Remote sensing technologies can make critical contributions to agriculture monitoring and improving our understanding of the effects of environmental changes. It provides coverage of large areas with high accuracy and can be a very effective tool for agriculture improvement. In this sense, remote sensing using multispectral images is an alternative for intensive manual monitoring of crops in the field and potential savings in time and resources.

In addition, many conventional techniques for earth monitoring applications require specific spectral features that are defined only for multispectral data such as deep learning, exploiting both temporal and cross-sensor dependencies and deep networks achieve much better performance than traditional methods.

Many of the limitations in the agricultural application of remote sensing techniques in previous years were overcome with the launch of Sentinel-2 A + B. The Sentinel-1 and Sentinel-2 constellation, with an improved spatial, spectral, and temporal resolution that is tailored to the needs of the agricultural community, both farmers and academic researchers with a focus on international agricultural development.

Remote sensing indices are useful for tracing the development of crops, their interrelatedness, and the consequences of the variables of interest for crop development. Following this concern, the application of smart agriculture and satellite remote sensing-based soil-vegetation index evaluations for agricultural land condition assessments is the key target of this project. Therefore, land suitability assessments can be performed using the multi-criteria decision method. The Multi-criteria Decision Method (MCDM) becomes more suitable when incorporating geospatial references. In recent years, computing technologies combined with GIS have enabled geospatial references using MCDM-land suitability evaluations.

Therefore, the purpose of this project is to develop a soil-vegetation intent land suitability assessment model based on multi-criteria decision-making analysis to determine optimal land distributions according to soil-vegetation indices to ensure elevated productivity.

Spatial information is key to improving the management of agricultural land. Up-to-date mapping of these lands provides opportunities for government and research organizations to monitor agricultural activities and for growers to understand crop status, and predict yields.

This project aims to build the infrastructure of a highly scalable system to monitor and develop the agricultural sector in Lebanon. The spatial information needed is not limited to imagery. This assessment cannot be completed without additional information related to agriculture like soils, weather, elevation, slope, and other data.

Our project aims to create an artificial intelligence model using deep learning algorithms to extract lands from images and classify them (Cultivated, Uncultivated ...). Then to build a spatial analysis model to use the result of the first one to assess lands in an agricultural and economical aspect.

The implementation of a system allows the government to survey the existing farms and farmers in order to monitor the health of individual crops in the fields and support vulnerable farm communities and support agriculture practices and development.
Finally create maps, apps, and dashboards to provide reports and statistics to help officials to make decisions and take action.

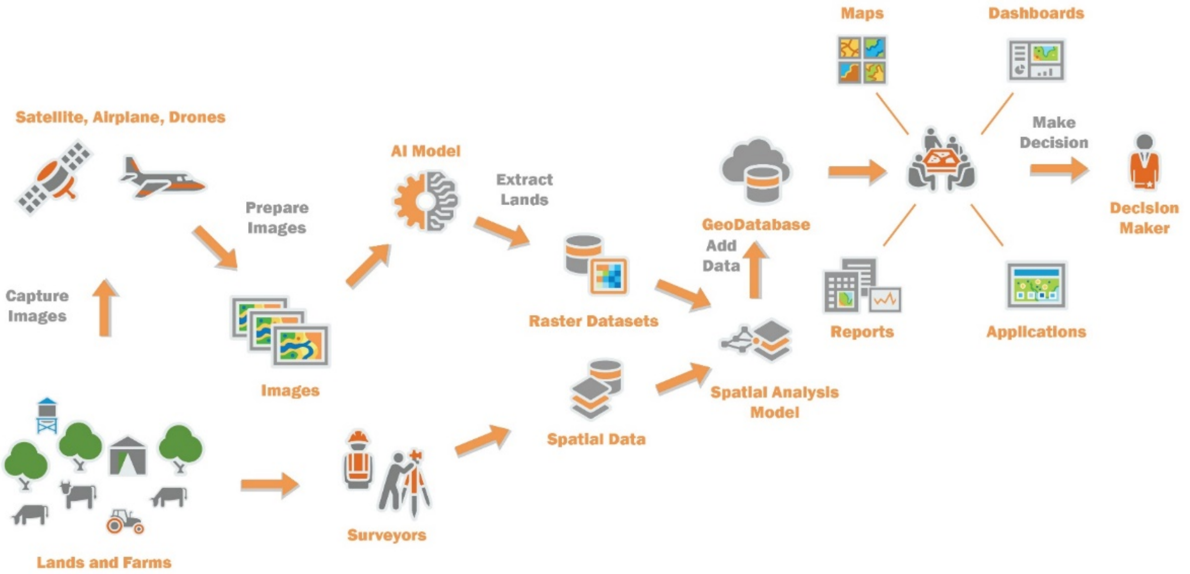


FIGURE 1 - PROJECT WORKFLOW

The Project has four main features:

- **Lands Extracting:** use the deep learning model to extract lands, update, and retrain the model.
- **Data Collection:** the process to survey farms and farmers, their status, and their needs permanently.
- **Data Analysis:** It is a toolbox used to assess the existing lands every time depending on changes in climate, season, agricultural indices, and other factors.
- **Monitoring & Reporting:** represent and show farm and lands data to support the decision-maker.

These four features are related to each other and depend on each other they create our solution life cycle.

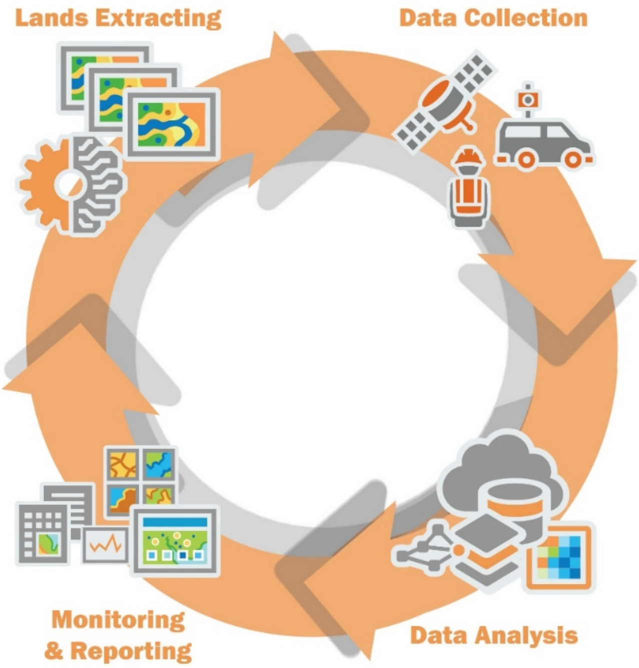


FIGURE 2 - SYSTEM LIFECYCLE

The Outcome of the project is a system that users depending on its role can use to collect, import, and export data, visualize and print maps and reports, apply spatial analysis on the data, and share the result with others or with the communities.

1. Study Zone

According to the complexity to cover all areas of Lebanon, from a time perspective to the availability of the data, and the resources needed to store and process this data. Due to these obstacles, we take a portion of the Zahle district, where we have all the data needed and represent a feasible study zone to apply our techniques in order to build a system that can achieve the agricultural land assessment.

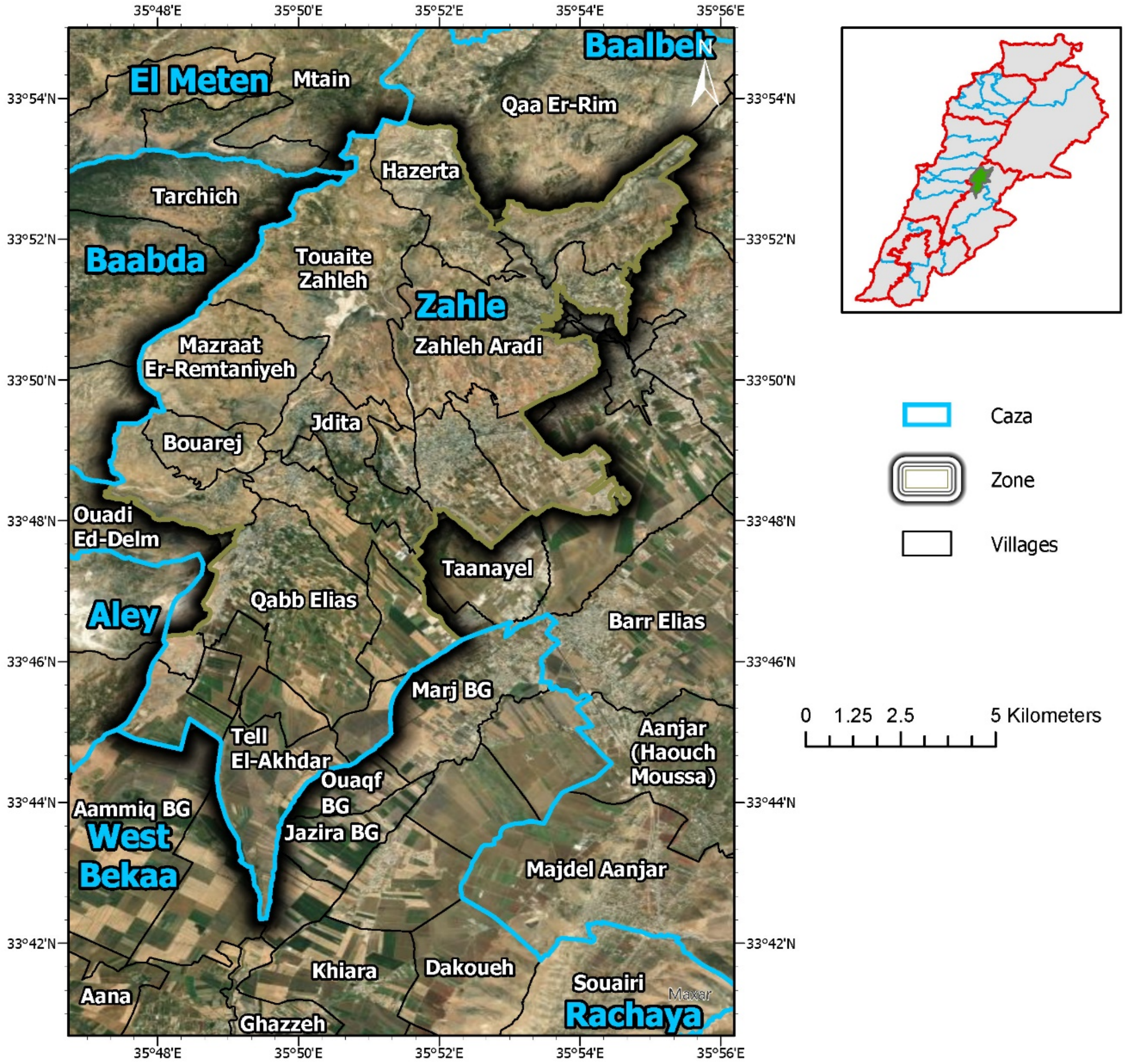


FIGURE 3 - STUDY ZONE MAP

Our zone covers 22 cadastral zones in the Zahle district located in the Bekaa governorate with a total area of 120 Km^2 and more than three small cities with a medium population like Zahle city.

This zone is considered a very diverse area, which is well populated in some cadastral zone and has approximately an equal area of exploited and unexploited lands. Therefore, it is a very good prototype example to test the success of the project techniques if we will apply them to the whole country in the future.

CHAPTER 2

Lands Extracting: Deep Learning Model

This chapter provides a region-based semantic segmentation model to extract exploited and unexploited lands from satellite images using the deep convolutional neural network algorithm.

1. Overview

The goal of this model is to convert the raster satellite images into a thematic raster that represents the exploited areas (cultivated land, farms, crops...) and the unexploited areas (Barred lands, grassland, …) in order to use it in the assessment phase.

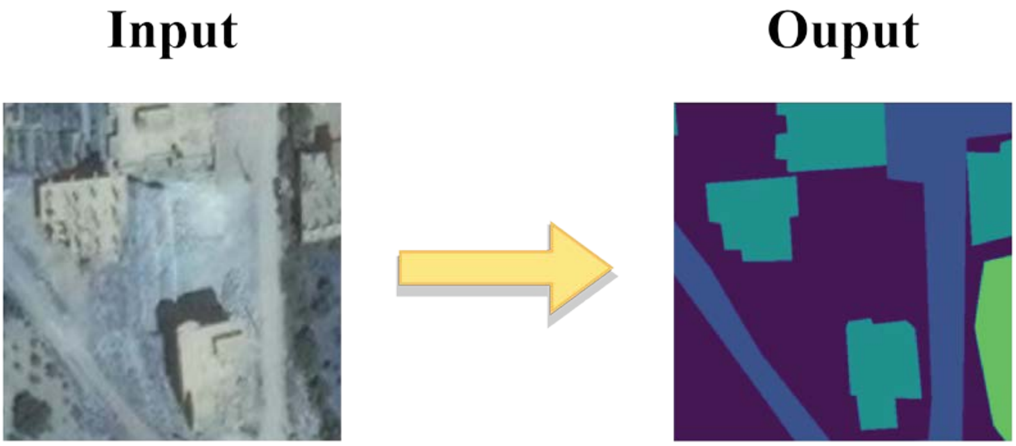


FIGURE 4 - DEEP LEARNING MODEL INPUT AND OUTPUT

To simplify the model we assume that the image has four classes:

- **Class 0:** represent the area that doesn’t classified.
- **Class 1:** represent the urban areas that contain roads, buildings squares, and lands that cannot be exploited anymore.
- **Class 2:** stands for exploited and cultivated areas.
- **Class 3:** stand for the unexploited areas.

2. Data

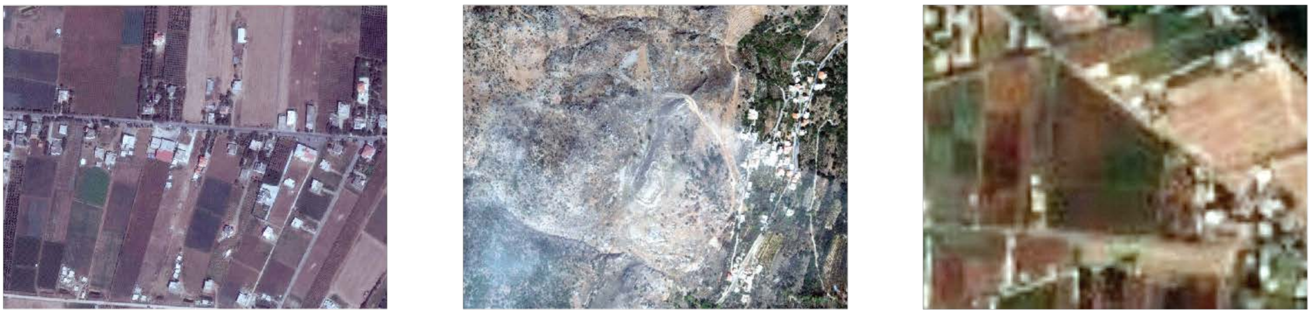
Due to the complexity of using Drones, we replace them with satellite images with high resolution that can be efficient to train this model.
Anyone can access the latest free satellite imagery of Earth; it only takes to know where you can find them. There are many platforms where you can find free satellite imagery. Some of them are streaming free current satellite images, while some provide licensed data.

Using The Global Mapper Software, you can download these images for any place on the earth. The original spatial resolution is 15 cm.
The downloaded images have two uses, the first one is to train and validate the model, and the second one is to predict and extract lands using the trained model.

2.1. Data Preprocessing

2.1.1. Data Cleaning

The downloaded satellite images are not cleaned. Although all images have the same spatial resolution, some images contain noise and can be less clear than others.
Many images contained clouds that block the feature contained in them. Therefore, cleaning images and choosing the net images is an important task before using any image.



Net Image **Image with cloud** **Image with noise**

FIGURE 5 - IMAGE TYPE

2.1.2. Data Labeling

With ArcGIS Pro, we can make our training dataset by digitizing masks on the images and labeling them. Then export it to create the dataset.
The first step is to create a feature class and add a label field then digitize areas on the images and determine for each feature its class.

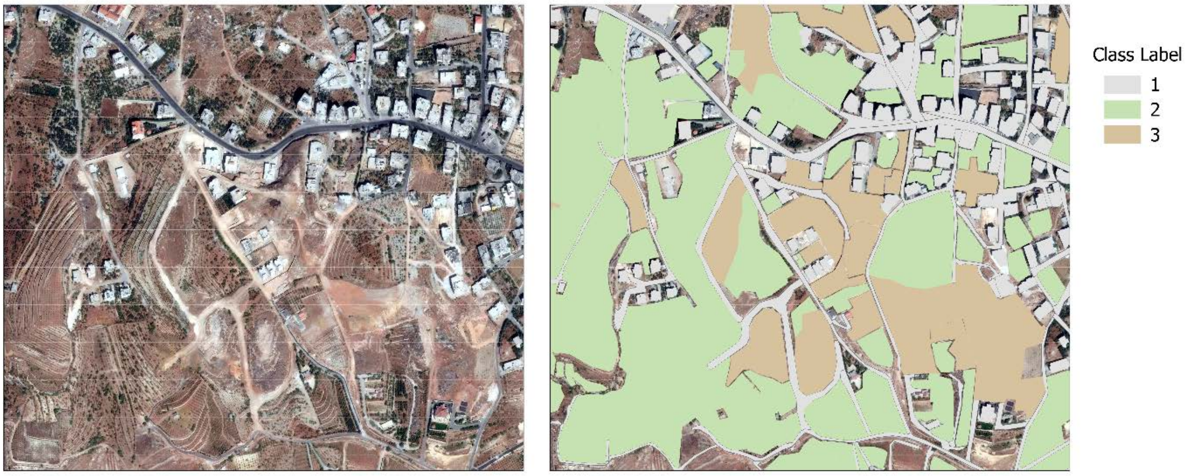


FIGURE 6 - IMAGE LABELING RESULT

2.1.3. Data Exporting

Using the “Export Training Data For Deep Learning” tool to convert labeled vector data into a deep learning training dataset using the satellite image. By choosing RCNN Masks for the Metadata Format, the output will be image chips that have a mask on the areas where the sample exists. The model generates bounding boxes and segmentation masks for each instance of an object in the image. This format is based on Feature Pyramid Network (FPN) and a ResNet101 backbone in the deep learning framework model.

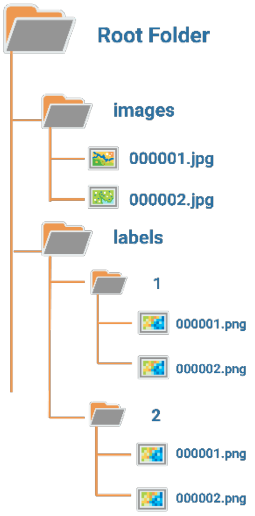


FIGURE 7 - TRAINING DATA SCHEMA

With this tool, we converted the satellite images and the mask feature class to a set of small images (512×512) pixels, and for every class set of mask images (512×512) pixels.

TABLE 1 - TRAINING DATA COUNT

	Count	Format	Pixels	Size	Channel
Images	3136	jpg	512 × 512	40 KB	3 (RGB)
Class1 Mask	3136	png	512 × 512	5 KB	1
Class2 Mask	3130	png	512 × 512	5 KB	1
Class3 Mask	3129	png	512 × 512	5 KB	1

2.1.4. Data Preparing

Preparing the data to be ready for start training we need two sets one is for X, which is the images set, and one for Y representing the labels mask. Therefore, we must merge for each image all masks into one image to represent Y.

The objective of the function **CreateSet()** is to read images and masks and convert them to Numpy arrays after merging masks.

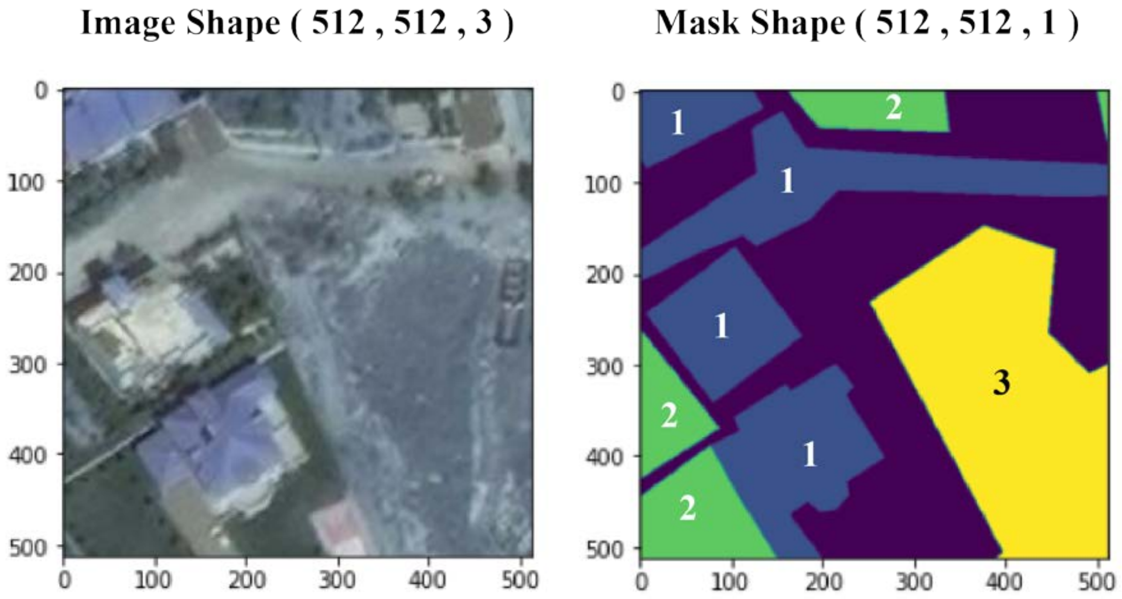


FIGURE 8 - IMAGE DATA AND MASK AS NUMPY ARRAY

The final dataset contains the Numpy array X which contains 3136 images with 3 channels and the Y array which contains also 3136 masks with 1 channel.

2.1.5. Data Splitting

One of the key aspects of supervised machine learning is model evaluation and validation. When you evaluate the predictive performance of your model, it is essential that the process be unbiased. Using **train_test_split()** from the data science library scikit-learn, you can split your dataset into subsets that minimize the potential for bias in your evaluation and validation process.

We have two levels of splitting, the first level is to have a training set and test set. The second level is to have the validation test.

TABLE 2 - TRAINING DATA SPLITTING RESULT

Training	Testing	Validation	Total
2509	564	63	3136

2.1.6. Data Binarize

Keras provides Numpy utility library, which provides functions to perform actions on Numpy arrays. Using the method **to_categorical()**, a Numpy array that has integers that represent different categories can be converted into a Numpy array (or) a matrix that has binary values and has columns equal to the number of categories in the data. This function returns a matrix of binary values (either '1' or '0'). It has the number of rows equal to the length of the input vector and a number of columns equal to the number of classes. This task separates between mask classes that can simplify the calculation of some metrics.

3. Model Architecture

Due to the low number of parameters in our deep learning model and the need of extracting the high-level features from our images, the "Convolutional Neural Network" is the best type of neural network that can be used for our task.

A Convolutional Neural Network is a Deep Learning algorithm that can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other. [1]

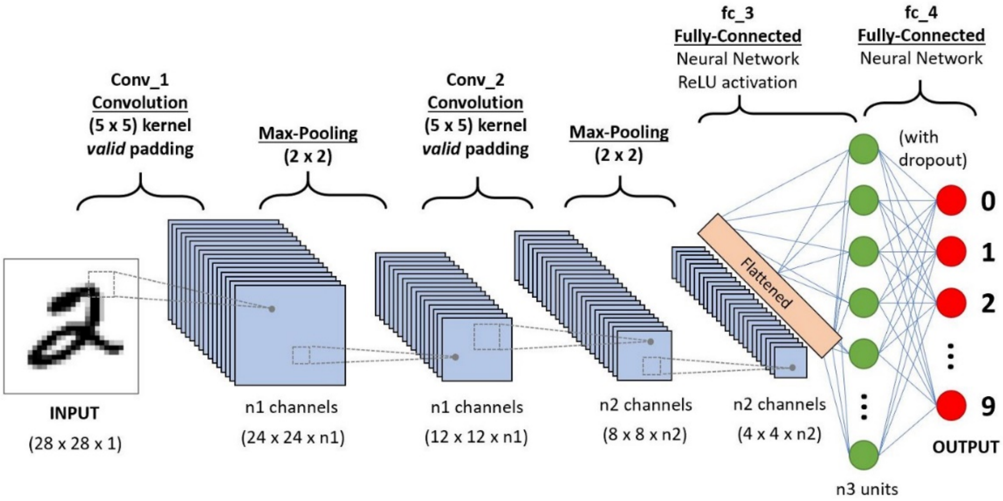


FIGURE 9 - CONOLUTIONAL NEURAL NETWORK

3.1. Overview

Semantic segmentation of remote sensing imagery aims at identifying the land-cover or land-use category of each pixel in an image. As one of the fundamental visual tasks, semantic segmentation has been attracting wide attention in the remote sensing community and has proven to be beneficial to a variety of applications, such as land cover mapping, traffic monitoring, and urban management. Recently, many studies have resorted to learning deep convolutional neural networks (CNNs) with full supervision for semantic segmentation and have obtained enormous achievements. [2]

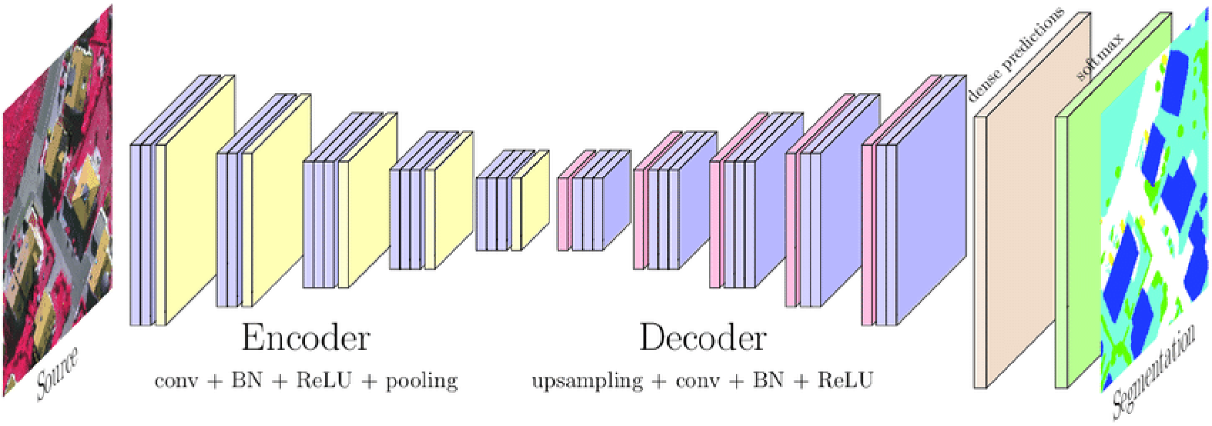


FIGURE 10 - SEMANTIC SEGMENTATION MODEL

Our model uses the semantic segmentation algorithm to infer pixel-level class masks using a fully convolutional network.

Using convolutions, we down sample the image to capture only the required information reducing the dimensions of the input image. The converse is up sampling with Deconvolution. As we use Deconvolutions, we up sample the reconstructed image from a lower dimension to higher dimensions. [3]

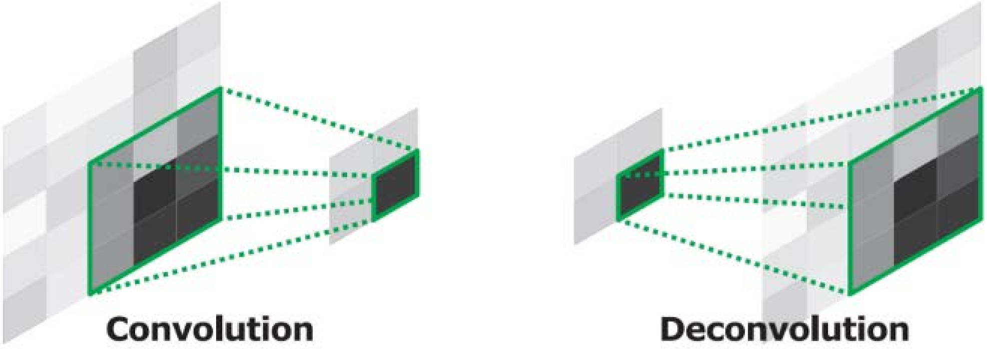
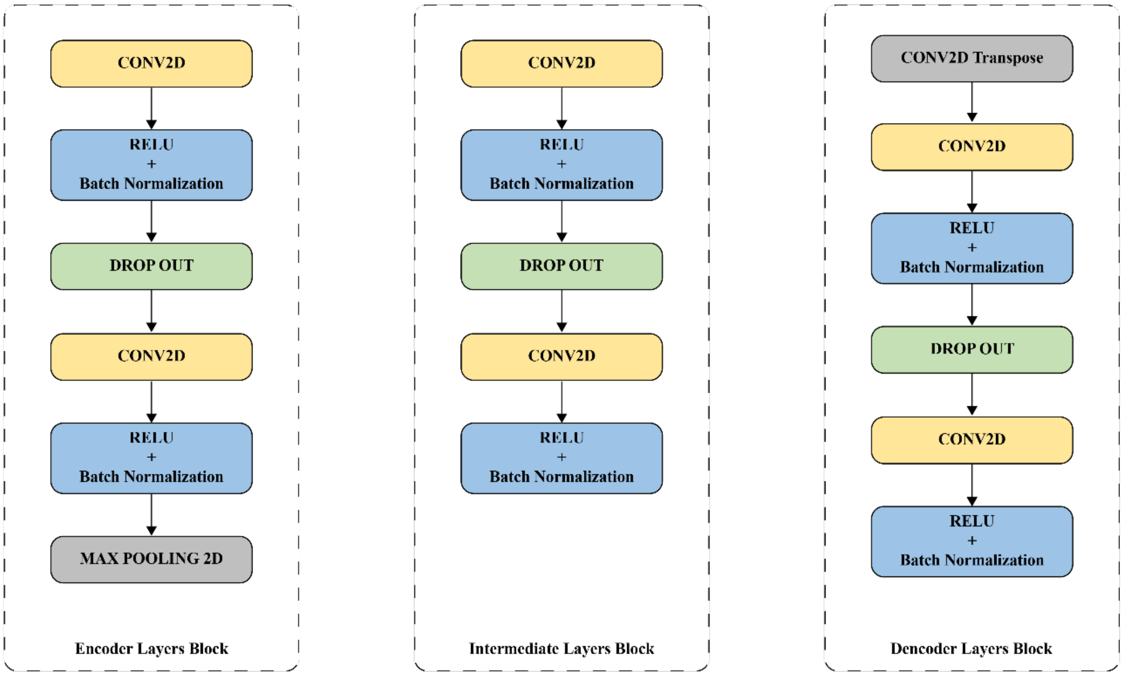


FIGURE 11 - CONVOLUTION AND DECONVOLUTION TECHNICS

3.2. Layers

An encoder-decoder cascaded dense semantic segmentation framework. Our model has eight blocks of layers, four encoder blocks and 4 decoder blocks, and one intermediate block.



CONV2D
RELU + Batch Normalization
DROP OUT
CONV2D
RELU + Batch Normalization
MAX POOLING 2D
Encoder Layers Block
CONV2D
RELU + Batch Normalization
DROP OUT
CONV2D
RELU + Batch Normalization
Intermediate Layers Block
CONV2D Transpose
CONV2D
RELU + Batch Normalization
DROP OUT
CONV2D
RELU + Batch Normalization
Dencoder Layers Block

FIGURE 12 – MODEL LAYERS DESIGN

Each encoder block has four main layers. Two convolutional each one followed by ReLU activation and batch normalization, separated by a dropout layer. Finally, a max-pooling layer ends the block.
Each decoder block starts with an up-sampling layer and four main layers. Two convolutional each one followed by ReLU activation and batch normalization, separated by a dropout layer. Finally, a max-pooling layer ends the block.

We start with an encoder block of 16 pixels size, and then each block has double the size of the previous block. The intermediate block has a 256-pixel size and contains layers as same as the encoder layers. For the decoder block, we start with 128 pixels then each block has a half-size of the previous block.

The first concatenation layer allows for any number of input channels. The output depicts the probability of four classes using the softmax function.

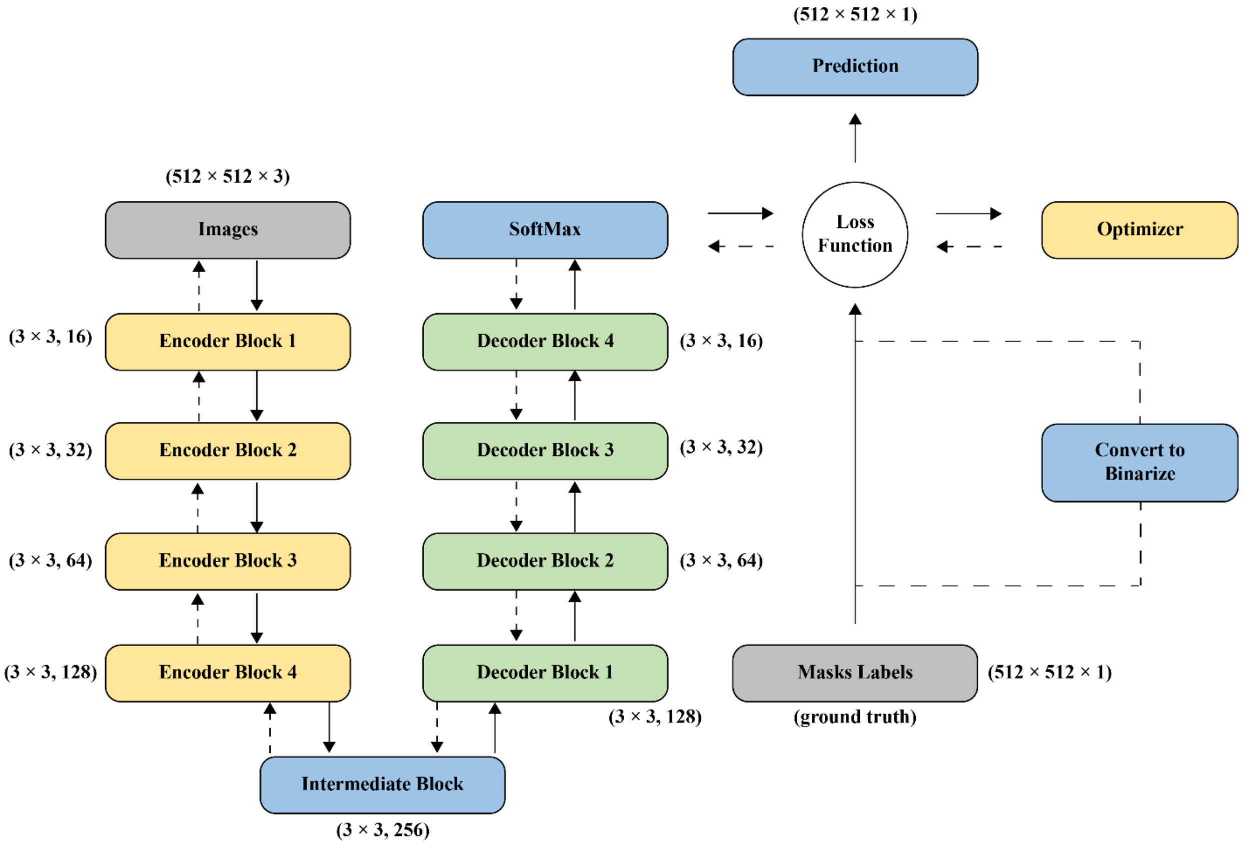


FIGURE 13 - MODEL LAYERS PATH

3.3. Optimizer

Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. [4]

We choose the Adam optimizer for our model because:

- Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. [4]

- Adam is relatively easy to configure where the default configuration parameters do well on most problems. [4]

3.4. Loss Function

Categorical cross-entropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.
Formally, it is designed to quantify the difference between two probability distributions. [5]
We use Categorical cross-entropy as a loss function for our model.

3.5. Hyper parameters

3.5.1. Learning Rate

The learning rate is the rate at which the gradient updates to the parameters occur in the gradient direction. When this rate, ε is too small, model convergence takes a long time. On the other hand, if ε is too large, the model diverges, and the loss might fluctuate indefinitely. To ensure optimal learning, an initial learning rate ε_0 is first defined (0.01 is a generally accepted standard here), following which the rate is updated by reducing its value using **ReduceLROnPlateau()** function that reduces the learning rate ε (multiply by 0.1) when the validation loss can’t decrease for N successively epoch (N=4).

EQUATION 1 - LEARNING RATE FORMULA

$$\begin{cases} \varepsilon_0 = 0.01 \\ \varepsilon_t = 0.1 \times \varepsilon_{t-1} \\ \end{cases}$$

3.5.2. Batch size

Batch size refers to the number of training examples utilized in one iteration. We use a mini-batch mode, where the model works fine for batch size equal to 16 for the size of the images, RAM Memory of the server used.

3.6. Implementation

There are many ready implementations on GitHub or Kaggle that we can use it. However, as it is very robust and complex, it can be hard to thoroughly understand every bit of it. In addition, the even bigger problem is that it does not run without error. Therefore, in our project, we build the model from scratch.

3.6.1. Language

Favored for applications ranging from web development to scripting and process automation, Python is quickly becoming the top choice among developers for artificial intelligence (AI), machine learning, and deep learning projects. [6]
Therefore, we use Python as a programming language to build our model.

3.6.2. Library

TensorFlow is the premier open-source deep learning framework developed and maintained by Google. Although using TensorFlow directly can be challenging, the modern tf.keras API beings the simplicity and ease of use of Keras to the TensorFlow project.

Using tf.keras allows us to design, fit, evaluate, and use deep learning models to make predictions in just a few lines of code.

3.6.3. Device

"Google Colab", is a product from Google Research. Colab allows us to write and execute our python code through the browser and is especially well suited to our deep learning model.

More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs. We use Colab since the lack of resources (GPU, RAM) using the available hosted computer.

4. Result and Evaluation

4.1. Metrics

Confusion Matrix is used to know the performance of a Machine learning classification. It is represented in a matrix form. Confusion Matrix gives a comparison between Actual and predicted values.

The confusion matrix is a 4 x 4 matrix, where 4 is the number of classes plus the background.

FN: The False-negative value for a class will be the sum of values of corresponding rows except for the TP value.

FP: The False-positive value for a class will be the sum of values of the corresponding column except for the TP value.

TN: The True Negative value for a class will be the sum of values of all columns and rows except the values of that class that we are calculating the values for.

TP: The True positive value is where the actual value and predicted value are the same.

Confusion Matrix allows us to measure Recall, Precision, Accuracy, IOU coefficient, and AUC-ROC curve are the metrics to measure the performance of the model.

EQUATION 2 - ACCURACY FORMULA

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

EQUATION 3 - PRECISION FORMULA

$$Precision = \frac{TP}{TP + FP}$$

EQUATION 4 - RECALL FORMULA

$$Recall = \frac{TP}{TP + FN}$$

The performance of a classifier that produces decision values of a pixel belonging to either class can be interpreted more with the Precision-Recall (Pre-Rec) and Receiver Operating Characteristic (ROC) curves.

The ROC curve plots the True Positive Rate TPR with respect to the False Positive Rate FPR.

EQUATION 5 - FALSE POSITIVE RATE FORMULA

$$FPR = \frac{FP}{FP + TN}$$

EQUATION 6 - TRUE POSITIVE RATE FORMULA

$$TPR = \frac{TP}{TP + FN} = Recall$$

The area under a ROC curve provides a means of measuring the classifier’s ability to discriminate between classes in the dataset. By this definition, maximizing the area (ROC AUC) leads to better classification accuracy.

The best performing models on each dataset are assessed with the F1-score and Intersection over Union (IOU) that consolidate the above results into fewer metrics.

The F1 score is the harmonic mean of Precision and Recall, while IOU is interpreted as the name suggests the intersection of the actual and predicted values, divided by the union of this set for a specific class.

IOU:

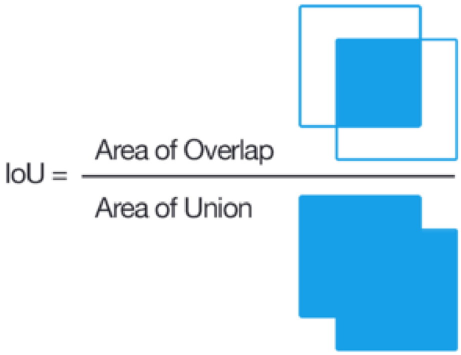


FIGURE 14 - IOU FORMULA

The Intersection-Over-Union (IoU), also known as the Jaccard Index, is one of the most commonly used metrics in semantic segmentation. The IoU is a very straightforward metric that's extremely effective.

EQUATION 7 - INTERSECTION OVER UNION FORMULA

$$IOU = \frac{TP}{TP + FP \ + FN}$$

F1-score:

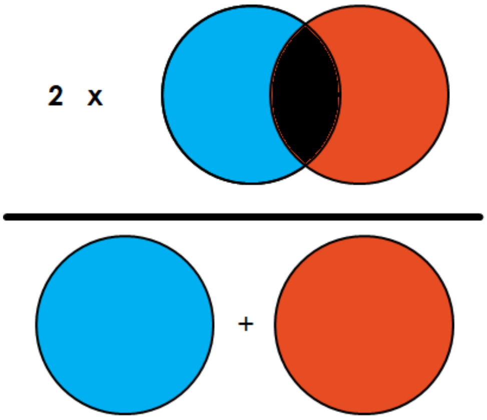


FIGURE 15 - F1 SCORE FORMULA

The Dice Coefficient Known as F1 score is 2 × the Area of Overlap divided by the total number of pixels in both images.

EQUATION 8 - F1 SCORE FORMULA

$$F1 = \frac{2 \times TP}{P + P'} = \frac{2 \times TP}{TP + FP + TP + FN} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

4.2. Model Evaluation

4.2.1. Training and testing

After running the model.fit() function which trains the model using the training set and validates with the test set, a historic result for each epoch is displayed containing loss and metrics like accuracy on training and validation. Every epoch takes about 3 minutes to finish, our model trains with 50 epochs, and the all-training function took about 3 hours to finish.

TABLE 3 - MODEL EPOCHS METRIC RESULT

	Metrics	Epoch 1	Epoch 10	Epoch 20	Epoch 30	**Epoch 34**	Epoch 40	Epoch 50
Train	Loss	1.0123	0.7336	0.6805	0.6449	0.6200	0.6119	0.6112
	Accuracy	0.7035	0.8193	0.8412	0.8595	0.8710	0.8751	0.8752
	Precision	0.5946	0.7032	0.7310	0.7483	0.7604	0.7559	0.7562
	Recall	0.6894	0.7653	0.7852	0.7983	0.8047	0.8081	0.8082
	F1 Score	0.4383	0.5570	0.5999	0.6207	0.6338	0.6296	0.6299
	IOU	0.6267	0.7211	0.7391	0.7578	0.7727	0.7741	0.7732
Test	Loss	2.2523	0.6401	0.6155	0.5966	0.5888	0.5946	0.5959
	Accuracy	0.5697	0.8701	0.8784	0.8876	0.8907	0.8889	0.8883
	Precision	0.5810	0.7469	0.7737	0.7918	0.7874	0.7853	0.7873
	Recall	0.5797	0.7645	0.7703	0.7754	0.7800	0.78	0.7798
	F1 Score	0.3071	0.6125	0.6473	0.6717	0.6697	0.6713	0.6743
	IOU	0.6519	0.7439	0.7498	0.7530	0.7555	0.7559	0.7559

The model has reached the optimal level at epoch 34 which provides a less test-loss value. After the epoch 34 the test-loss value did not improve despite the reduction of the learning rate from 10^{-5} to 10^{-9}.

The following graphs show the evolution of the train and the test metrics in the function of epochs.

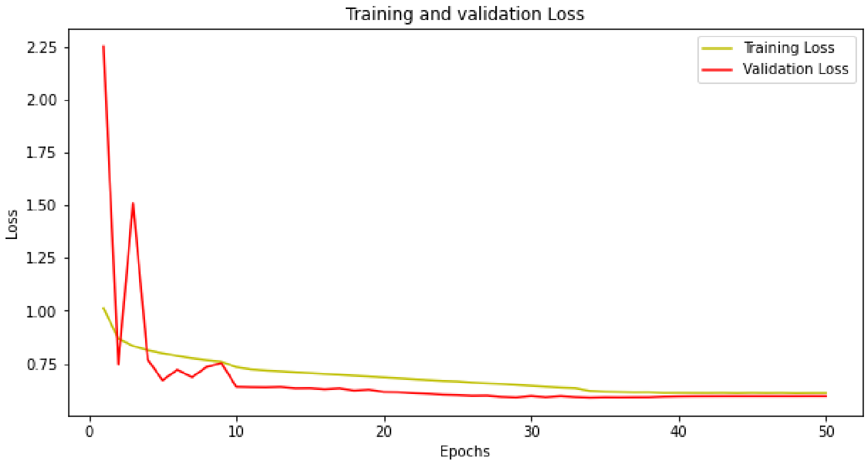


FIGURE 16 - GRAPH REPRESENTS THE VARIATION OF TRAINING AND VALIDATION LOSS IN FUNCTION OF EPOCHS

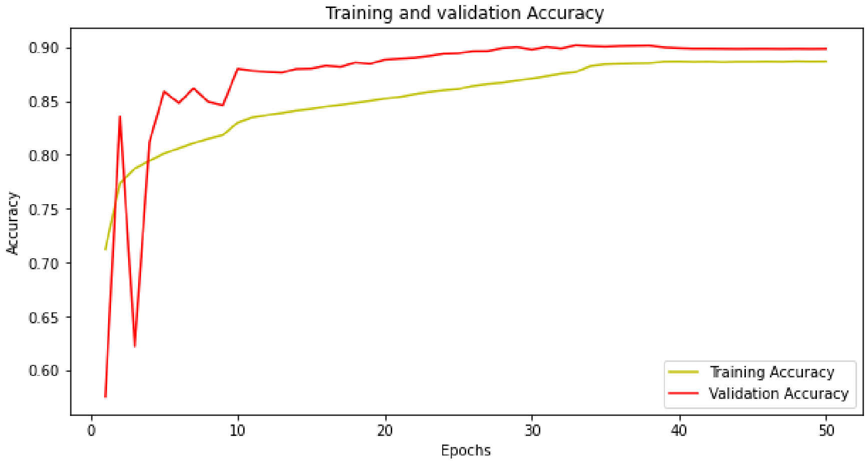


FIGURE 17 - GRAPH REPRESENTS THE VARIATION OF TRAINING AND VALIDATION ACCURACY IN FUNCTION OF EPOCHS

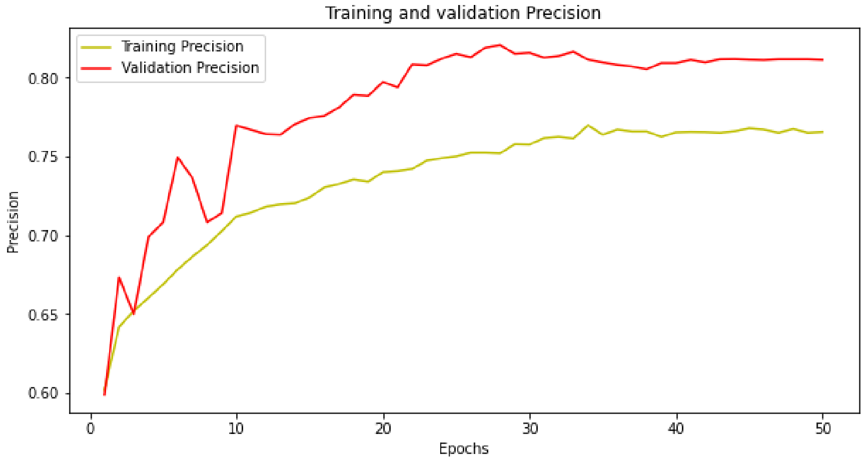


FIGURE 18 - GRAPH REPRESENTS THE VARIATION OF TRAINING AND VALIDATION PRECISION IN FUNCTION OF EPOCHS

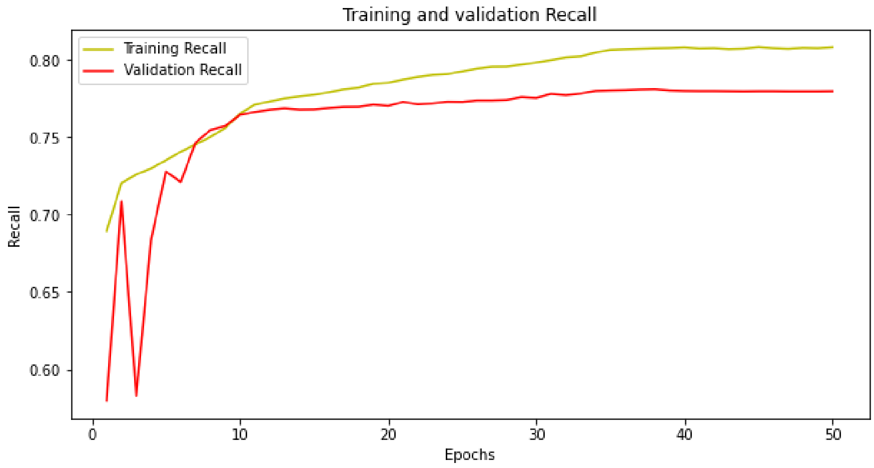


FIGURE 19 - GRAPH REPRESENTS THE VARIATION OF TRAINING AND VALIDATION RECALL IN FUNCTION OF EPOCHS

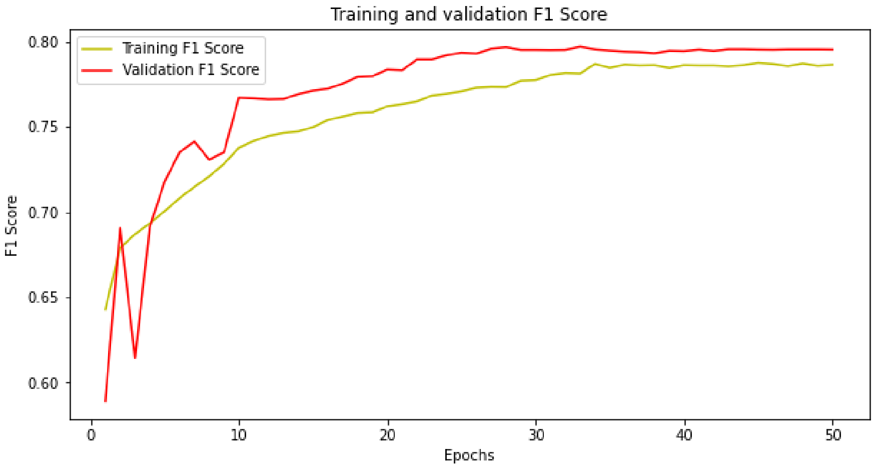


FIGURE 20 - GRAPH REPRESENTS THE VARIATION OF TRAINING AND VALIDATION F1 SCORE IN FUNCTION OF EPOCHS

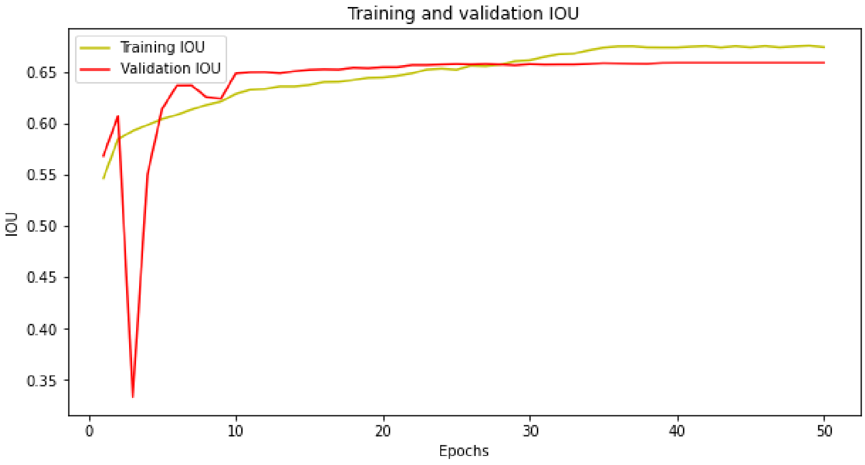


FIGURE 21 - GRAPH REPRESENTS THE VARIATION OF TRAINING AND VALIDATION IOU IN FUNCTION OF EPOCHS

4.2.2. Validation

Now we use the validation set to validate the optimal model after saving the weights at the optimal epoch (epoch 34).

Therefore, we load the model and predict the mask of the validation images. The calculation of the confusion matrix between the true masks and the predicted mask results in the following table.

TABLE 4 - CONFUSION MATRIX OF THE MODEL

	Predicted Values (Pixels)				
Actual Values (Pixels)		**Class 0**	**Class 1**	**Class 2**	**Class 3**
	Class 0	344692	1371732	2086852	3156971
	Class 1	326697	4704999	1246380	1283421
	Class 2	209614	402416	22774952	5281134
	Class 3	121883	897234	1925468	38800211

The above table represents the multi-label confusion matrix that shows for each class the distribution of predicted pixels in each class.

Class 0 represents the unlabeled pixels, which have zero value.

For each class, we represent a detailed confusion matrix that represents the true positive, false negative, false positive, and true negative.

TABLE 5 - CONFUSION MATRIX OF CLASS 1

		Class 1		
		Prediction outcome		
		+	-	Total
Actual value	+	TP 4,704,999	FN 2,856,498	P’ 7,561,497
	-	2,671,382 FP	2,671,382 TN	5,342,764 N’
Total		P 7,376,381	5,527,880 N	

TABLE 6 - CONFUSION MATRIX OF CLASS 2

		Class 2 (exploited Area)		
		Prediction outcome		
		+	-	Total
Actual value	+	TP 22,774,952	FN 5,893,164	P’ 28,668,116
	-	5,258,700 FP	5,258,700 TN	10,517,400 N’
Total		P 28,033,652	11,151,864 N	

TABLE 7 - CONFUSION MATRIX OF CLASS 3

		Class 3 (Unexploited Area)		
		Prediction outcome		
		+	-	Total
Actual value	+	TP 38,800,211	FN 2,944,585	P’ 41,744,796
	-	9,721,526 FP	9,721,526 TN	19,443,052 N’
Total		P 48,521,737	12,666,111 N	

We can now calculate the metrics for each class and then calculate them for our model to get the final evaluation values. Since the classes do not have the same number of pixels, we should calculate a weighted average of metrics for the model depending on the number of pixels for each class.

EQUATION 9 - OVERALL WEIGHED METRIC MODEL FORMULA

$$Model_{metric} = \frac{\sum_{i=1}^{n} Class_{metric} \times Nb\ of\ pixels}{Total\ Nb\ of\ pixels}$$

The below table represents the value of metrics for each class in addition to the overall model metrics.

TABLE 8 - METRICS VALUES OF THE MODELS

	Class 0	Class 1	Class 2	Class 3	Model
Pixels Number	6,960,247	7,561,497	28,668,116	41,744,796	169,869,312
Accuracy	91.44	93.49	86.87	85.09	87.16
Precision	34.37	63.78	81.24	79.96	75.22
Recall	4.95	62.22	79.44	92.95	78.44
IOU	4.52	45.98	67.13	75.39	64.18
F1 Score	8.66	62.99	80.33	85.97	76.79

The Pre-Rec curves and ROC curves for the model are shown below figures.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

Roc Values is the AUC in our case 0.79 for class 1 and 0.85 for class 2 and 3 that are good.

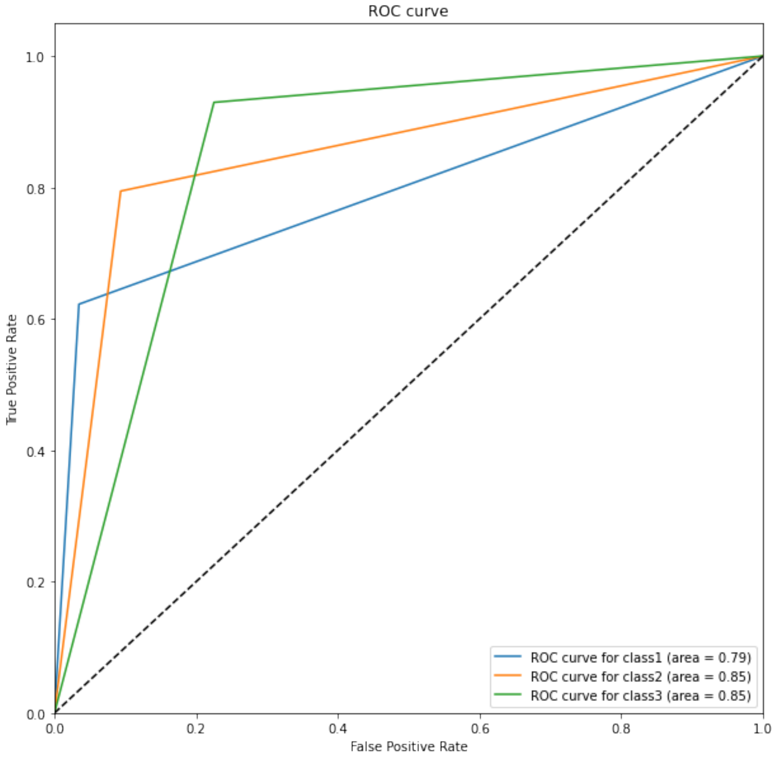


FIGURE 22 - ROC CURVE OF THE MODEL BY CLASSES

4.3. Result

The Figure below shows the result of example images on the validation set, its mask, and the model prediction.

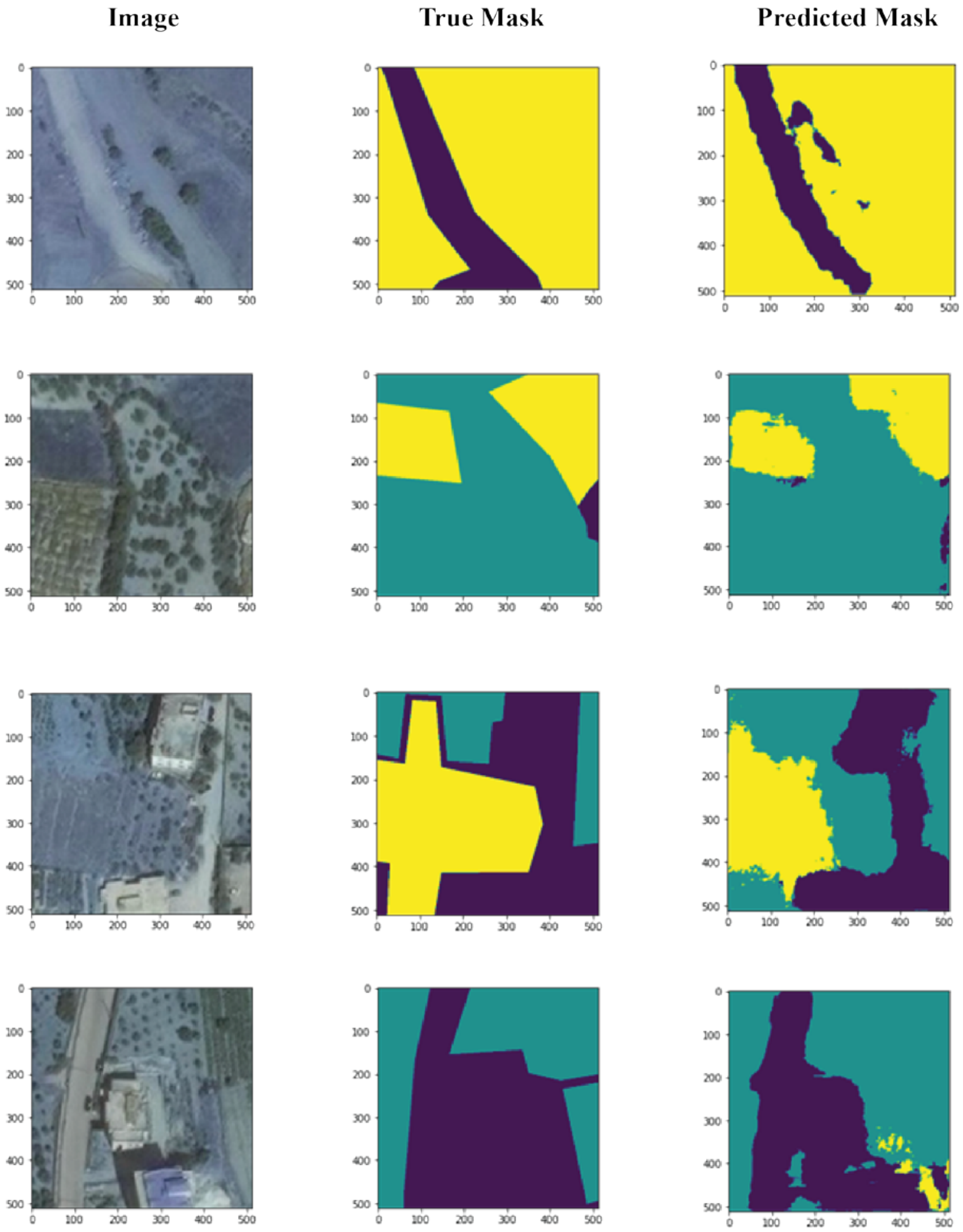


FIGURE 23 - SAMPLES IMAGES WITH THEIR TRUE MASKS AND PREDICTED MASKS

After merging all the predicted masks, we got a final raster for the entire study zone that contains three values: Urban area, exploited area, and unexploited area.

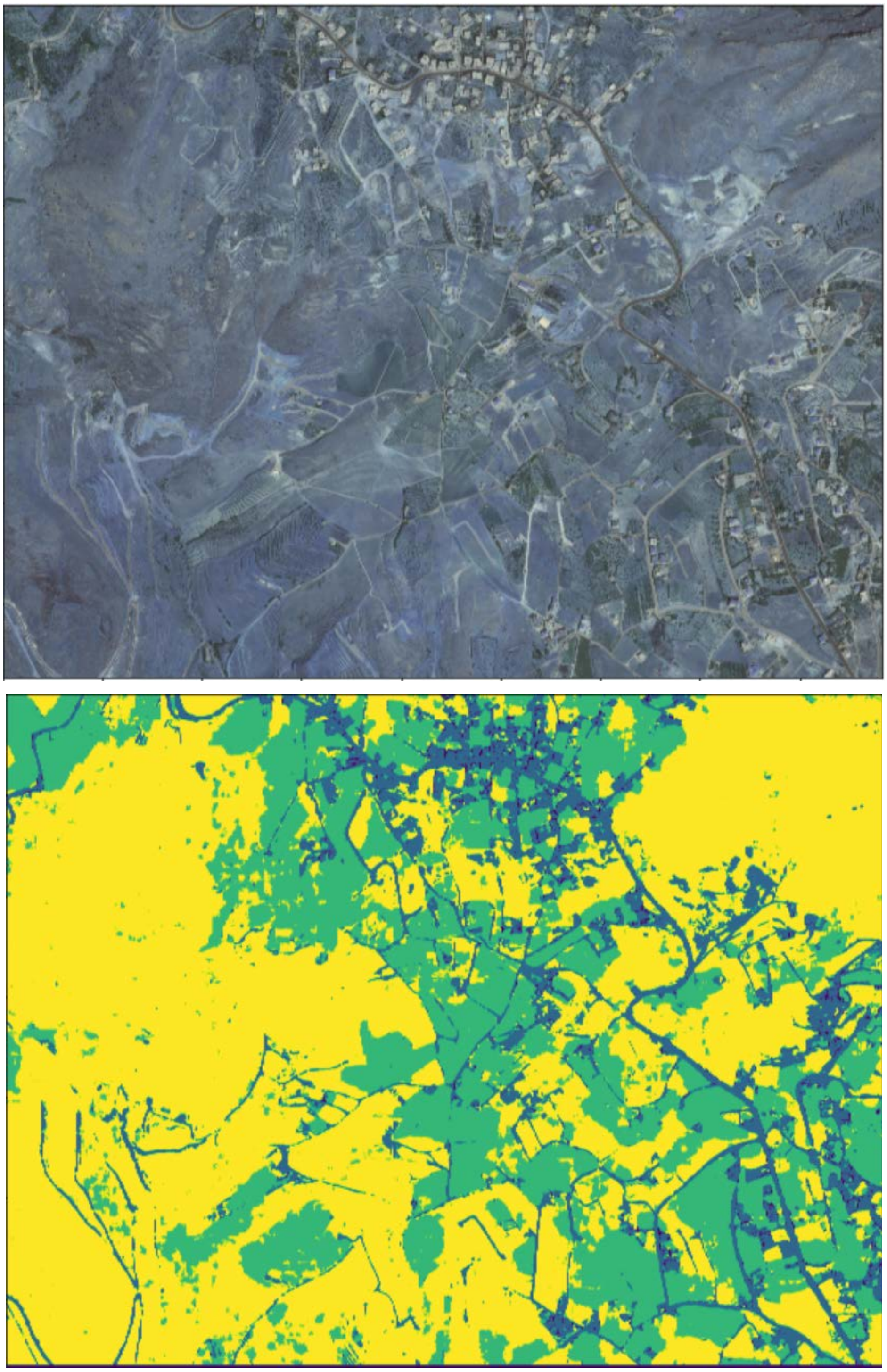


FIGURE 24 - SAMPLE IMAGE WITH ITS PREDICTED MASK

CHAPTER 3

Data Management

This chapter describes what kind of data has been used in this project and how it was collected and stored, and what type of new data is created and extracted from the original set.
In addition, it shows the management of the centralized geodatabase that forms the data source of all implemented and future planned features in this project.

1. Overview

All data used in this project is stored in the centralized geodatabase. It represents our data store that contains vector, raster, and tabular data.

The first dataset represents the administrative division (Lebanon boundary, Provinces, Caza, and the cadastral zones) and the study zone boundary. They are stored as vector data (feature classes).

In addition, the parcels inside the study zone are represented as a feature class. Topography data like DEM and Slope are stored as raster.

Sentinel 1 and sentinel 2 images are stored in a mosaic dataset that allows us to store, manage, view, and query collections of raster and image data. It is a data model within the geodatabase used to manage a collection of raster datasets (images) stored as a catalog and viewed as a mosaicked image.

Predicted lands classes are also stored in the geodatabase as a mosaic dataset.

Land use, Soil, and geological data are stored as feature classes.

Tabular data represents tables stored in general the calculated indices and statistical summary related to all stored data.

2. Source of data

There is no unique source of data, each layer or each dataset was found on a different site.

The base map imagery used in the deep learning model is extracted from Global Mapper software. Parcels get from Cad files delivered by the official survey departments in Zahle district and then processed to be ready to be stored inside the geodatabase.

Sentinel-1 and sentinel-2 images were downloaded from the Copernicus Open Access Hub (previously known as Sentinels Scientific Data Hub), it provides complete, free, and open access to Sentinel-1, Sentinel-2, Sentinel-3, and Sentinel-5P user products, starting from the In-Orbit Commissioning Review (IOCR).

The national center for remote sensing (CNRS) was contributed to getting the soil data needed.

Administrative boundaries, digital elevation models, roads data downloaded from open sites on the internet like open street map and other references.

TABLE 9 - EXTERNAL DATA USED IN THE PROJECT

Name	Source	Native Format	Resolution	Size
Base map imagery	Global Mapper software	Raster	15 cm	96 GB
Sentinel-1	The European Space Agency	Raster	10 m	890 MB per image
Sentinel-2	The European Space Agency	Raster	10 m	1.1 GB per image
DEM	Alaska Satellite Facility	Raster	12.5 m	525 MB
Parcels	official survey departments	Vector		66 MB
Road Network	Open Street Map	Vector		
Soil Data	The national center for remote sensing	Vector		1 MB
Geology Data	Directorate of Geographic Affairs - Lebanese Army	Vector		1 MB
Administrative Boundary Level 0 (Lebanon)	The Humanitarian Data Exchange	Vector		360 KB
Administrative Boundary Level 1 (Provinces)	The Humanitarian Data Exchange	Vector		800 KB
Administrative Boundary Level 2 (Caza)	The Humanitarian Data Exchange	Vector		2 MB
Administrative Boundary Level 3 (Cadaster)	The Humanitarian Data Exchange	Vector		9 MB

3. Import and preprocessing Data

Importing data to the geodatabase is not always easy. Some data are ready to be imported directly using ArcGIS pro tools, but some of them need cleaning and processing before importing. Writing python code to create custom tools to automate the preprocessing operation is essential, especially for the imagery dataset that needs to be updated with the new satellite images.

3.1. Import directly

Vector data like soil data, geology, Administrative boundaries, and road network are imported using the ArcGIS pro tool “Feature Class To Feature Class (Conversion)”. [7]

Data cleaning at the level of the attribute table to clear some unused fields or some wrong values to have a clean feature class for each layer.

For the raster data, DEM and Satellite images are imported using always ArcGIS pro by creating a raster dataset “Create Raster Dataset (Data Management)”. [8]

3.2. Import Parcels

Parcels cannot be imported straightly into the geodatabase, since they came as multiple AutoCAD files as polylines and annotation that each cadaster zone has its cad sheet.

Converting those files to a single feature class needs first importing them to the ArcGIS pro and then creating a polygon feature class after cleaning the cad files and ensuring that all parcel polylines are closed and have their number annotation inside them. Then use the spatial analysis join to catch each parcel its number and write it in the attribute table. All this process needs to be applied to each cadaster zone, so the best practice to do that is to use automation and geo-processing, create a python tool, and run it for each zone. After this task, merging the multiple feature class into one after giving each zone an id to separate parcels based on the cadaster zone (VillageId). The final output will be one feature class named parcels as polygon contains in its attribute tables the parcel number and the cadaster zone id.

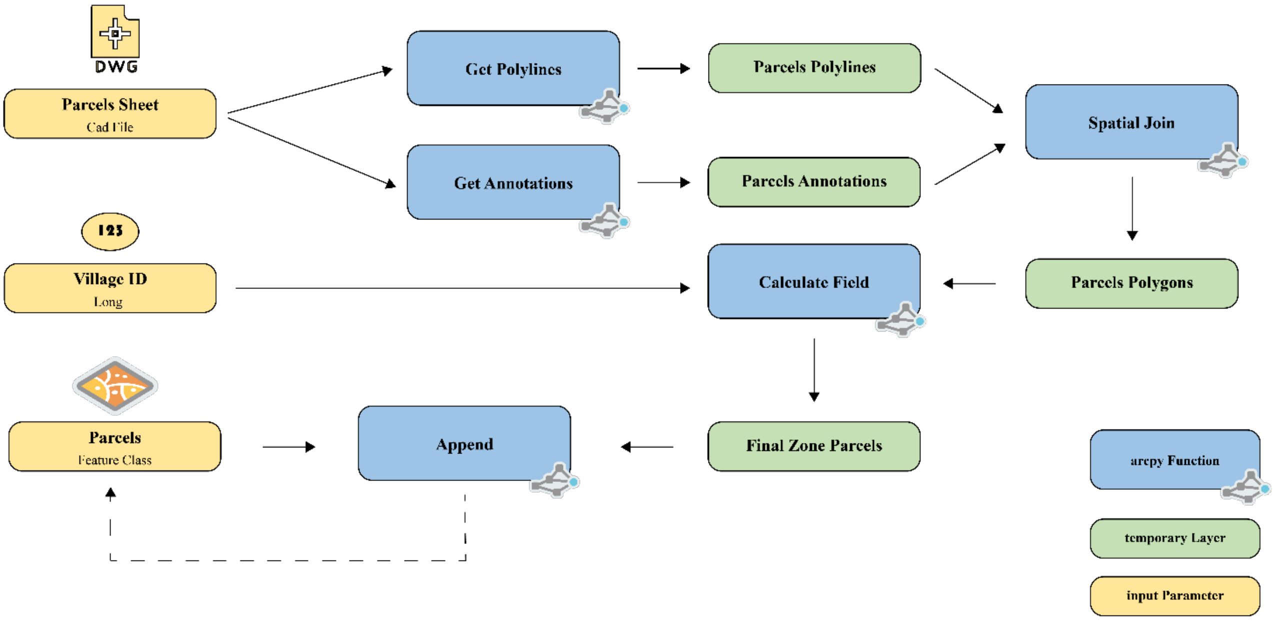


FIGURE 25 - IMPORT PARCELS SCRIPT ORGANIGRAM

3.3. Import Sentinel-1 Images

SENTINEL-1 is an imaging radar mission providing continuous all-weather, day-and-night imagery at C-band. The SENTINEL-1 constellation provides high reliability, improved revisit time, geographical coverage, and rapid data dissemination to support operational applications in the priority areas of marine monitoring, land monitoring, and emergency services.

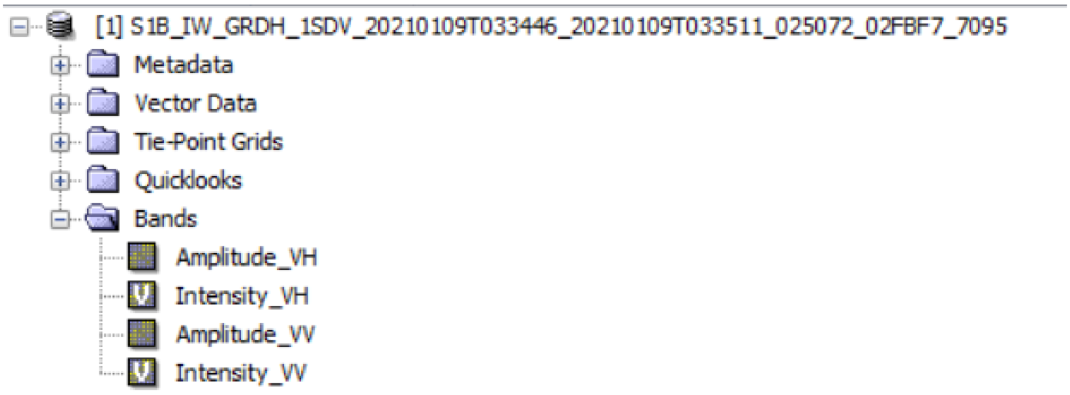
The SENTINEL-1 Synthetic Aperture Radar (SAR) instrument acquire data in four exclusive modes, we use in this project the Interferometric Wide swath (IW) - Data is acquired in three swaths using the Terrain Observation with Progressive Scanning SAR (TOPSAR) imaging technique. In IW mode, bursts are synchronized from pass to pass to ensure the alignment of interferometric pairs. IW is SENTINEL-1's primary operational mode over land. [9]

The observation mode over (non-polar) land is the Interferometric Wide (IW) mode providing dual-polarization (VV and VH) imagery over a 250 km swath at a 5 × 20 m spatial resolution.

SENTINEL-1 data products distributed by ESA include multi-levels, but we use the Ground Range Detected (GRD) Level-1 data with a multi-looked intensity only (systematically distributed).

Each downloaded file in .zip format stores the data in four bands:

- Amplitude_VH
- Intensity_VH
- Amplitude_VV
- Intensity_VV



```
[1] S1B_IW_GRDH_1SDV_20210109T033446_20210109T033511_025072_02FBF7_7095
    Metadata
    Vector Data
    Tie-Point Grids
    Quicklooks
    Bands
        Amplitude_VH
        Intensity_VH
        Amplitude_VV
        Intensity_VV
```

FIGURE 26 - SENTINEL1 IMAGE CATALOG VIEW IN THE SNAP SOFTWARE

Before using the sentinel-1 images, the pre-processing steps including radiometric calibration, removal of thermal noise offset, and ortho-rectification with radiometric correction should be applied using SNAP (Sentinel Application Platform) is a remote sensing toolbox architecture developed by the European Space Agency. It includes tools for all common remote sensing satellites.

Using SNAP we can build a preprocessing model and use it in batch processing to convert all images and get the final images.

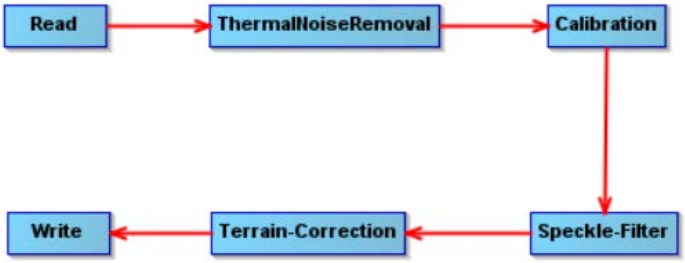


FIGURE 27 - SENTINEL1 PREPROCESSING WORKFLOW

Each image results in two image files (sigma0_VH and sigma0_VV). These images need to be in one image with two bands.

Importing sentinel-1 images to the geodatabase in a mosaic dataset, and saving each image in its date with two bands after clipping it to the study zone will be done using a custom geoprocessing tool. This tool can run each time we need to import a new sentinel-1 image to the geodatabase.

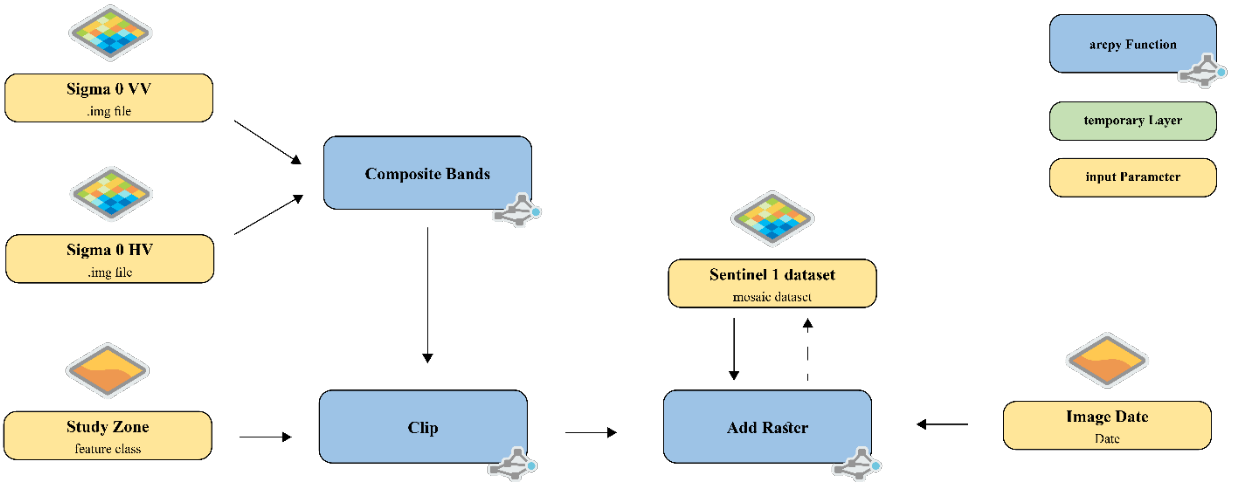


FIGURE 28 - IMPORT SENTINEL1 IMAGES SCRIPT ORGANIGRAM

3.4. Import Sentinel-2 Images

SENTINEL-2 is a European wide-swath, high-resolution, multi-spectral imaging mission. The full mission specification of the twin satellites flying in the same orbit but phased at 180°, is designed to give a high revisit frequency of 5 days at the Equator.

SENTINEL-2 carries an optical instrument payload that samples 13 spectral bands: four bands at 10 m, six bands at 20 m, and three bands at 60 m spatial resolution. The orbital swath width is 290 km.

The Sentinel-2 product used is **Level-2A,** which is the Bottom-Of-Atmosphere reflectance in cartographic geometry.

There are 13 Sentinel 2 bands in total. We use four bands with 10 meters in pixel size that came from Sentinel 2B, to form a multispectral image. Blue (B2), green (B3), red (B4), and near-infrared (B8) channels with a 10-meter resolution.

Each band is downloaded in a single .jp2 file, so a pre-processing phase is mandatory to merge these bands into one multispectral image and added it to a mosaic dataset.

Another custom tool was created to automate the importing task of each group of bands into the mosaic dataset. This tool can be run each time we need to import a new sentinel-2 image to the geodatabase.

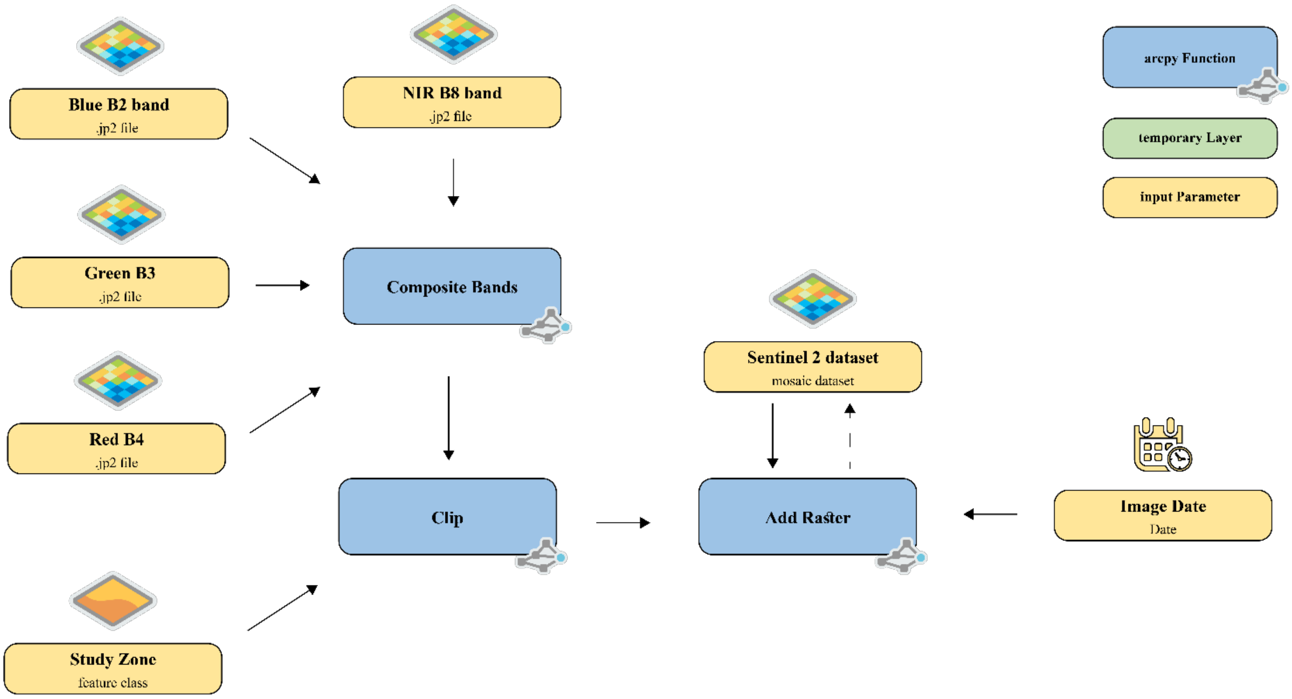


FIGURE 29 - IMPORT SENTINEL2 IMAGE SCRIPT ORGANIGRAM

4. Data extraction

The set of data listed in the last section represents layers and tables used as native sets. There are more tables, layers, and raster images inside the geodatabase. Those datasets are extracted from the first dataset. Using existing geoprocessing tools in the ArcGIS pro or custom tools developed with python and Arcpy.

4.1. Slope layer

The Slope layer represents the steepness at each cell of a raster surface. The lower the slope value, the flatter the terrain; the higher the slope value, the steeper the terrain.

It is calculated usually from the DEM raster; in our geodatabase, we use the preexisted geoprocessing tool “Slope (Spatial Analyst)”.

4.2. Roads Closeness

It is a raster layer that represents the average minimum distance in each pixel from the road network, this raster indicates where we are close to the road network and where we are far from it.

To create this layer we use the vector road network layer (Polylines) and we create a raster using the existing geoprocessing tool **Calculate Distance (Raster Analysis).**

It is used to calculate the Euclidean distance from a single source or set of sources.

4.3. Create Indices

Using the sentinel-2 images added recently to the geodatabase with its four bands to create multiple single-band raster layers, each raster represents a vegetation index calculated based on its proper formula using the four bands of sentinel-2 and the two bands of sentinel-1. These calculations are repeated every month and the result will be appended to the indices mosaic dataset.

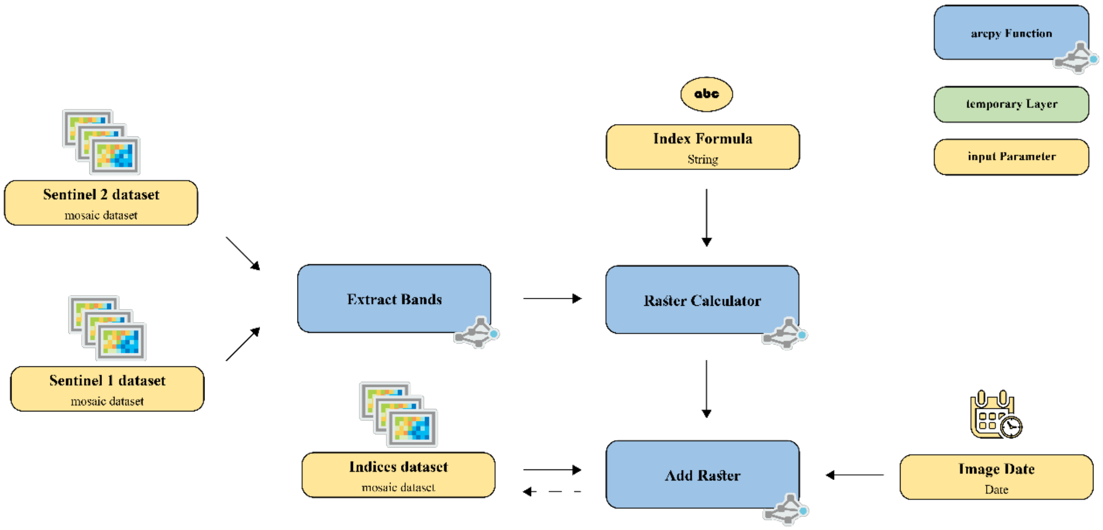


FIGURE 30 - GENERATE INDICES SCRIPT ORGANIGRAM

This Tool generates eight indices each month, RVI using sentinel one bands (VV and HV) and NDVI, ARVI, MSAVI, AVI, CIG, GNDVI, and SR using sentinel two bands (Red, Green, Blue, and Infrared).

4.3.1. Radar Vegetation Index (RVI)

EQUATION 10 - RADAR VEGETATION INDEX FORMULA

$$RVI = \frac{4VH_{\Upsilon 0}}{VH_{\Upsilon 0} + VV_{\Upsilon 0}}$$

γ0 (gamma-nought) represents the radiometrically and geometrically corrected SAR backscattering coefficient for each polarization combination in linear units (m2/m2).

RVI is a ratio of cross-polarization to ~total power from all polarization channels. It generally ranges between zero and one, and it is a measure of scattering randomness. As a ratio, RVI has less sensitivity to radar measurement geometry and topography and remains insensitive to absolute calibration error in radar data.

RVI is near zero for a smooth bare surface and increases with vegetation growth. It has an enhanced sensitivity to vegetation cover and biomass.

4.3.2. Radar Vegetation Index (NDVI)

Normalized Difference Vegetation Index (NDVI) quantifies vegetation by measuring the difference between near-infrared (which vegetation strongly reflects) and red light (which vegetation absorbs).

NDVI always ranges from -1 to +1. But there isn't a distinct boundary for each type of land cover.

As shown below, **Normalized Difference Vegetation Index (NDVI)** uses the NIR and red channels in its formula.

EQUATION 11 - NORMALIZED DIFFERENCE VEGETATION INDEX FORMULA

$$NDVI = \frac{IR - R}{IR + R}$$

Healthy vegetation (chlorophyll) reflects more near-infrared (NIR) and green light compared to other wavelengths. But it absorbs more red and blue light.

This is why our eyes see vegetation as the **color green**. If you could see near-infrared, then it would be strong for vegetation too. Satellite sensors like Landsat and Sentinel-2 both have the necessary bands with NIR and red. [10]

4.3.3. Atmospherically Resistant Vegetation Index (ARVI)

ARVI is most useful in regions of high atmospheric aerosol content. It uses blue light reflectance measurements to correct for the atmospheric scattering effects that also influence the reflectance of red light.

EQUATION 12 - ATMOSPHERICALLY RESISTANT VEGETATION INDEX FORMULA

$$ARVI = \frac{IR - 2R + B}{IR + 2R - B}$$

The range for an ARVI is -1 to 1 where green vegetation generally falls between values of 0.20 to 0.80.

4.3.4. Modified Soil-Adjusted Vegetation Index (MSAVI)

The modified soil-adjusted vegetation index (MSAVI) is an index designed to substitute NDVI and NDRE where they fail to provide accurate data due to low vegetation or a lack of chlorophyll in the plants.

During the stages of germination and leaf development, there is a lot of bare soil between the seedlings.

EQUATION 13 - MODIFIED SOIL ADJUSTED VEGETATION INDEX FORMULA

$$MSAVI = 0.5 \times \left[(2IR + 1) - \sqrt{(2IR + 1)^2 - (8IR - R)} \right]$$

On EOS Crop Monitoring, MSAVI values range from -1 to 1, where:

-1 to 0.2 indicate bare soil, 0.2 to 0.4 is the seed germination stage, 0.4 to 0.6 is the leaf development stage.

When the values go over 0.6, it is now high time to apply NDVI instead. In other words, the vegetation is dense enough to cover the soil. [11]

4.3.5. Enhanced Vegetation Index (EVI)

EQUATION 14 - ENHANCED VEGETATION INDEX FORMULA

$$EVI = \frac{2.5 \times (IR - R)}{IR + 6R + 7.5B + 1}$$

Enhanced Vegetation Index (EVI) is similar to Normalized Difference Vegetation Index (NDVI) and can be used to quantify vegetation greenness. However, EVI corrects for some atmospheric conditions and canopy background noise and is more sensitive in areas with dense vegetation. [12]

4.3.6. Enhanced Vegetation Index (CIG)

EQUATION 15ENHANCED VEGETATION INDEX FORMULA

$$CIG = \frac{IR}{G} - 1$$

Calculates the Green Chlorophyll Index (CIg) from a multiband raster object and returns a raster object with the index values. [13]

4.3.7. Green Normalized Difference Vegetation Index (GNDVI)

EQUATION 16 - GREEN NORMALIZED DIFFERENCE VEGETATION INDEX FORMULA

$$GNDVI = \frac{IR - G}{IR + G}$$

The GNDVI (Green Normalized Difference Vegetation Index) is an index of the plant’s "greenness" or photosynthetic activity. It is one of the most widely used vegetation indices to determine water and nitrogen uptake in the crop canopy.

The values are given by this index also vary between -1 and 1. [14]

4.3.8. Enhanced Vegetation Index (GNDVI)

EQUATION 17 - ENHANCED VEGETATION INDEX FORMULA

$$SR = \frac{IR}{R}$$

This is the simplest VI which is a ratio between the reflectance recorded in the Near Infra-Red (NIR) and Red bands. This is a quick way to distinguish green leaves from other objects in the scene and estimate the relative biomass present in the image. Also, this value may be very useful in distinguishing stressed vegetation from non-stressed areas. [15]

CHAPTER 4

Spatial Analysis Models

In this chapter, we will create the spatial analysis models and geo-processing tools that perform essential operations on the geographic data already collected and stored in the geodatabase. In addition, create a new dataset that contains and combines the all characteristics of the lands data. Also, build the suitability models to weights locations relative to each other based on given criteria in order to find a favorable location.

1. Overview

There are three different types of spatial analysis in this project. The first type is to merge parcels with the vector data using the “Union Tool” in order to create a new dataset. Each row contains a unique combination of data. In other means, this model clip the parcel into small pieces, each piece represents a set of indices in addition to the parcel number and village Id.

The second type is to calculate the statistical value of raster indices on each parcel combined with the type of land already extracted (Unexploited, Exploited), the result will be stored in a new table of statistical indices.

The result of those analysis operations is two statistical tables one is permanent, which stores the static data of the parcels like (Elevation, Slope, Road closeness,…).

This table can store more data types, which is populated with a Geo-processing tool called “CalculatPermanentStatisticalIndicesTool”.

The second table will be dynamic and it stores the statistical indices of the indices raster created and stored in the central geodatabase. But it populated each time we import a new sentinel-1 and sentinel-2 image before creating the raster indices on a different date.

A custom Geoprocessing tool is used to populate this table.

The third type of analysis represents two models to create a suitability area and scores with criteria set by the user by merging multiple rasters using weighted suitability calculation where those weights are chosen by the user.

2. Merge the vector data

The first step before starting our analysis is to merge the parcels with the vector data. The geology layer, the soil type layer, and the parcels merged using the predefined analysis tool “Union”.
Computes a geometric union of the input features. All features and their attributes will be written to the output feature class.

3. Create permanent statistical indices

In order to calculate the topographical characteristics for the parcels, a python tool was developed tool to extract the statistical (minimum, average, maximum) values of the elevation, slope, and road closeness in each parcel. This calculation split the statistics based on the land exploitation class (Unexploited area, Exploited area) in each parcel.

Using the “ZonalStatisticsAsTable” predefined tool, our model summarizes the values of each raster within the zones of the parcels and reports the results as a table.

Those statistical indices are permanent and does not change in function of time.

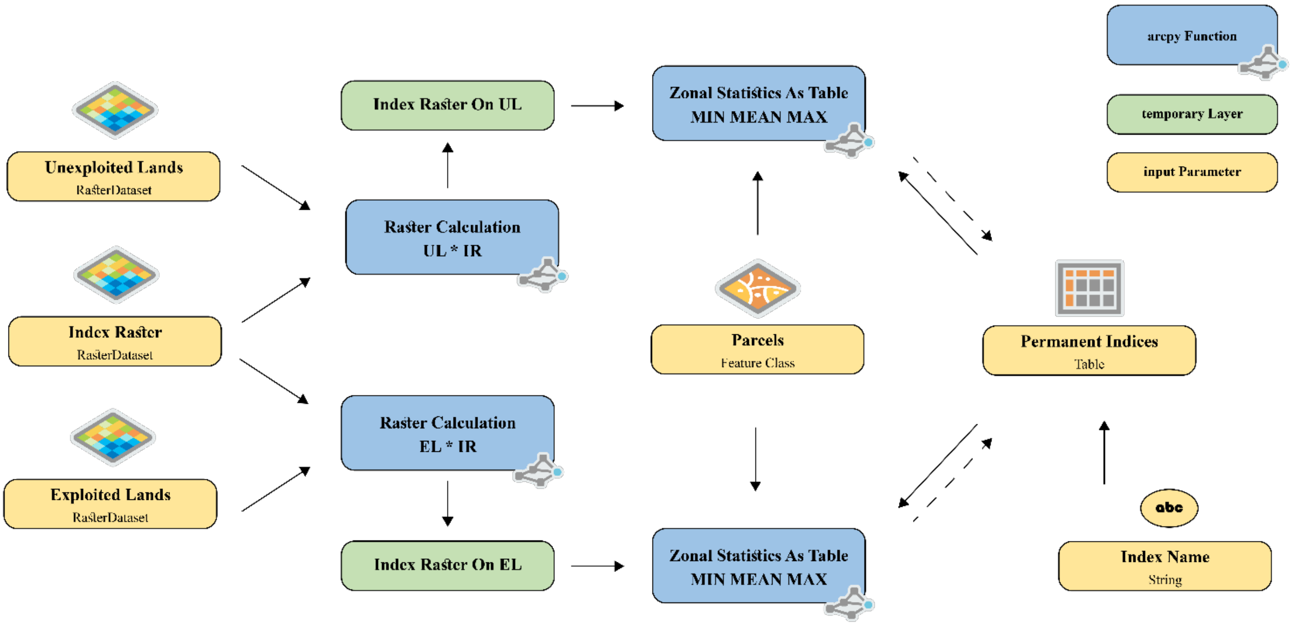
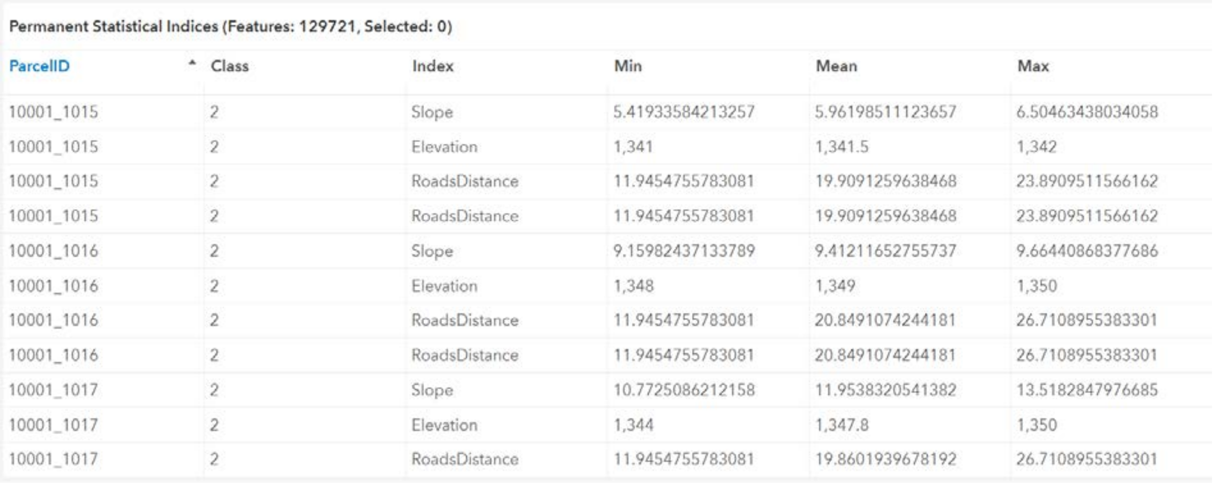


FIGURE 31 - CREATE PERMANENT STATISTICAL INDICES SCRIPT ORGANIGRAM

The result will be appended to the permanent indices dataset, which is store in each row the parcel id, land class, index name, and statistics (min, mean, and max).



Permanent Statistical Indices (Features: 129721, Selected: 0)

ParcelID	Class	Index	Min	Mean	Max
10001_1015	2	Slope	5.41933584213257	5.96198511123657	6.50463438034058
10001_1015	2	Elevation	1,341	1,341.5	1,342
10001_1015	2	RoadsDistance	11.9454755783081	19.9091259638468	23.8909511566162
10001_1015	2	RoadsDistance	11.9454755783081	19.9091259638468	23.8909511566162
10001_1016	2	Slope	9.15982437133789	9.41211652755737	9.66440868377686
10001_1016	2	Elevation	1,348	1,349	1,350
10001_1016	2	RoadsDistance	11.9454755783081	20.8491074244181	26.7108955383301
10001_1016	2	RoadsDistance	11.9454755783081	20.8491074244181	26.7108955383301
10001_1017	2	Slope	10.7725086212158	11.9538320541382	13.5182847976685
10001_1017	2	Elevation	1,344	1,347.8	1,350
10001_1017	2	RoadsDistance	11.9454755783081	19.8601939678192	26.7108955383301

FIGURE 32 - SAMPLE DATA OF THE PERMANENT INDICES

4. Create statistical indices

This model is created to calculate the vegetation indices of the exploited area in each parcel. It is a python tool that extracts the statistical (minimum, average, maximum) values of indices raster that is generated from sentinel-1 and sentinel-2 using the indices formulas on each overlay with the exploited and unexploited lands and the parcel.

Unlike the previous tool “permanent statistical indices” the result of this tool is appended to the statistical indices table where each index raster represents an image at a point in time, this model can be run each time we import new indices raster.

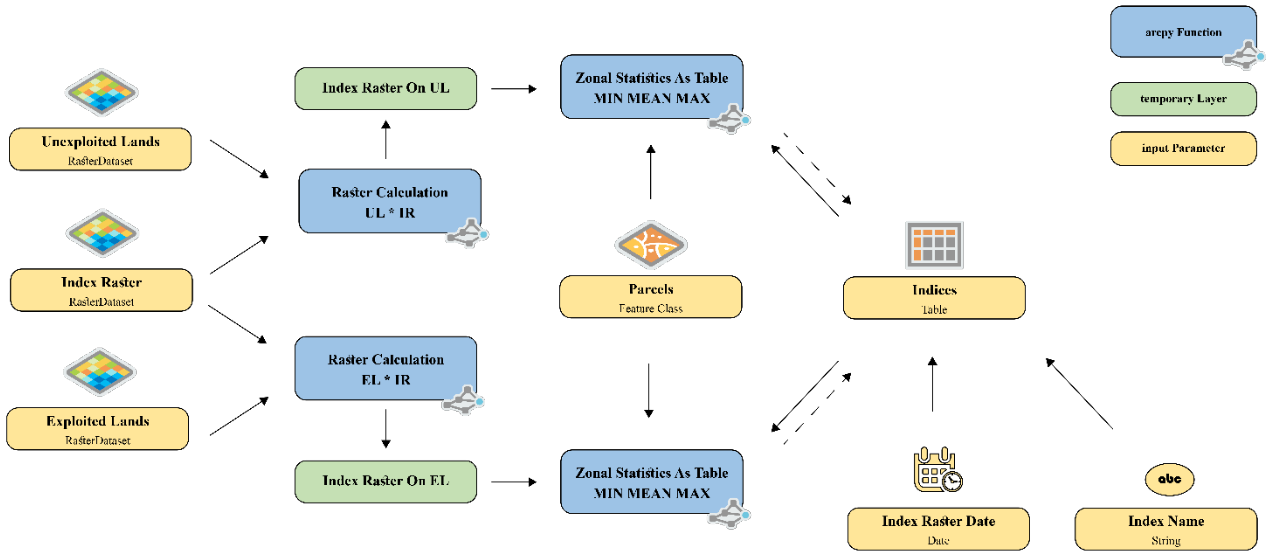
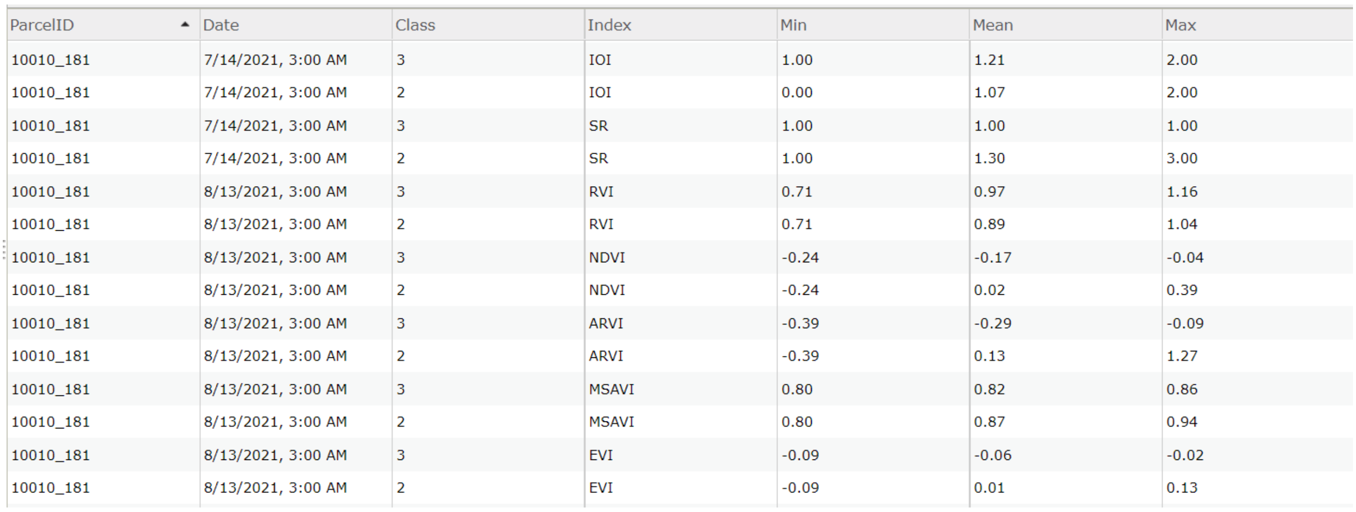


FIGURE 33 - CREATE STATISTICAL INDICES SCRIPT ORGANIGRAM

The result will be appended to the indices dataset, which is store in each row the parcel id, land class, index name, date, and statistics (min, mean, and max).



ParcelID	Date	Class	Index	Min	Mean	Max
10010_181	7/14/2021, 3:00 AM	3	IOI	1.00	1.21	2.00
10010_181	7/14/2021, 3:00 AM	2	IOI	0.00	1.07	2.00
10010_181	7/14/2021, 3:00 AM	3	SR	1.00	1.00	1.00
10010_181	7/14/2021, 3:00 AM	2	SR	1.00	1.30	3.00
10010_181	8/13/2021, 3:00 AM	3	RVI	0.71	0.97	1.16
10010_181	8/13/2021, 3:00 AM	2	RVI	0.71	0.89	1.04
10010_181	8/13/2021, 3:00 AM	3	NDVI	-0.24	-0.17	-0.04
10010_181	8/13/2021, 3:00 AM	2	NDVI	-0.24	0.02	0.39
10010_181	8/13/2021, 3:00 AM	3	ARVI	-0.39	-0.29	-0.09
10010_181	8/13/2021, 3:00 AM	2	ARVI	-0.39	0.13	1.27
10010_181	8/13/2021, 3:00 AM	3	MSAVI	0.80	0.82	0.86
10010_181	8/13/2021, 3:00 AM	2	MSAVI	0.80	0.87	0.94
10010_181	8/13/2021, 3:00 AM	3	EVI	-0.09	-0.06	-0.02
10010_181	8/13/2021, 3:00 AM	2	EVI	-0.09	0.01	0.13

FIGURE 34 - SAMPLE DATA OF THE STATISTICAL INDICES

5. Suitability models

Concerning completing the spatial analysis model, two suitability tools were created to help find the best place to cultivate any type of plantings based on dynamic criteria defined by the user.

The first tool is “Finding Suitability Zones” to get only the areas that coincide with the criteria defined by the user. The result is a thematic raster with zeros and one’s values represent where is suitable and where is not suitable.

The second tool is “Generate Suitability Score” to generate areas with a score based on normalization ranges and percentage weight for each raster to get suitability zones where the user can explore where the best areas and where the less suitable and where the worst places.

Those two tools are generic which means that the user can use rasters and define the best range of values in each raster to get the best areas that correspond to each condition.

5.1. Find Suitability Zones Tool

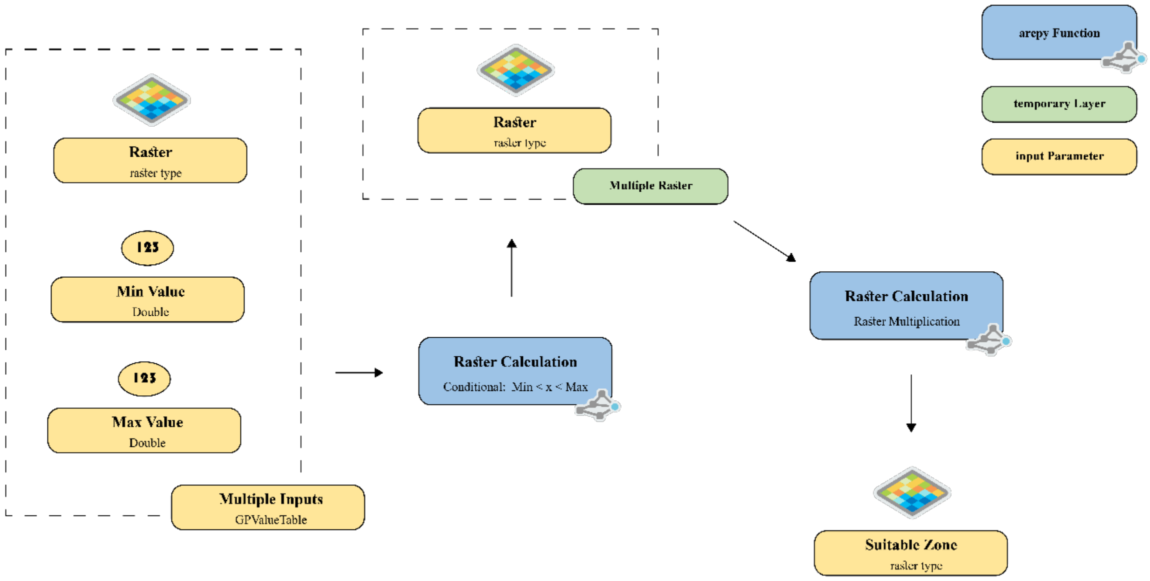


FIGURE 35 - FINDING SUITABILITY ZONES SCRIPT ORGANIGRAM

Users can use this model to find where the areas that hit off with their criteria are. By classifying the study zone into two classes (satisfied and not satisfied). A conditional and logical raster calculation is applied to produce the suitability area.

Users can use many rasters and define a range of accepted values for each raster.

5.2. Generate Suitability Scores

Users can use this model to classify the zone referring to their criteria and the defined range after giving the weight of each raster to produce a score raster to explore where the most suitable and where the unsuitable and get a general idea about the suitability scores in the study zone.

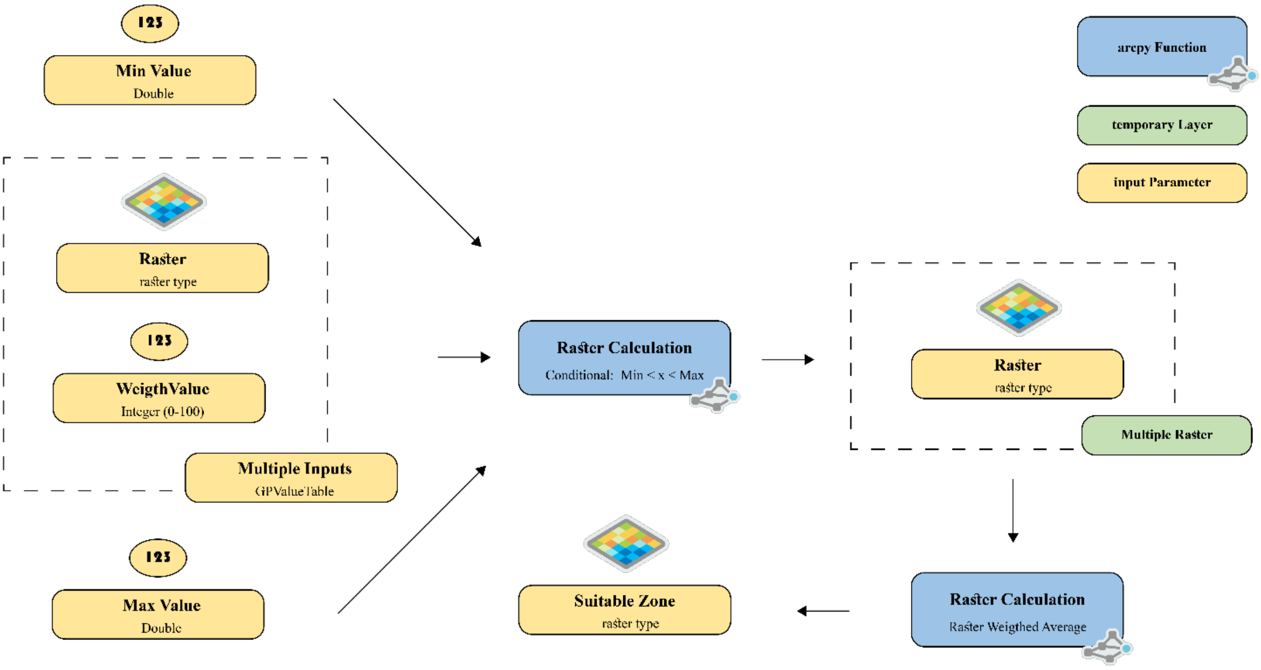


FIGURE 36 - GENERATE SUITABILITY SCORES SCRIPT ORGANIGRAM

CHAPTER 5

Monitoring and reporting

In this chapter, we created interactive web applications (dashboards) that contain indicators, charts, and maps to monitor and visualize land exploitation, vegetation health, soil characteristic, terrain properties, and other data to make the users identify the characteristics of agricultural lands from the quality of soil, monitor vegetation health and to determine the lands that cover his needs.

1. Overview

We created four monitoring and reporting applications. The first dashboard is a general reporting tool to give information about land exploitation at the level of the cadastral zones. The second one more specific is to visualize and query data at the level of lands and parcels to get the parcels' characteristics like average elevation and average slope and exploitation areas and other data. The third dashboard is created to monitor the exploited lands by visualizing the change of many vegetation and soil indicators in the function of time. The last application is soil data reporting which is to give an idea about the soil properties of each parcel.

All those applications are dynamic and interactive where the users can pan and zoom and select the target cadastral zone, parcel, or indicator to filter the data in such a way to interact with the application and help them to make decisions or take action and assess the lands from many points of view.

2. The Dashboards

Using ArcGIS Online, we create four web applications by publishing all the data stored in the centralized geodatabase to the portal to create web layers and web maps to be used in those dashboards.

All the dashboards are web applications and need only a browser and internet connection to be opened. Using Esri products, in this case, the ArcGIS online enables us to make those applications without writing code and without any development work.

2.1. Lands Exploitation Dashboard

This Application can be used to explore land exploitation and its percentages from the total area. It contains the map and its legend, four indicators of areas: total area, urban area, exploited area, and unexploited area. In addition, a pie chart for the distribution of land types.

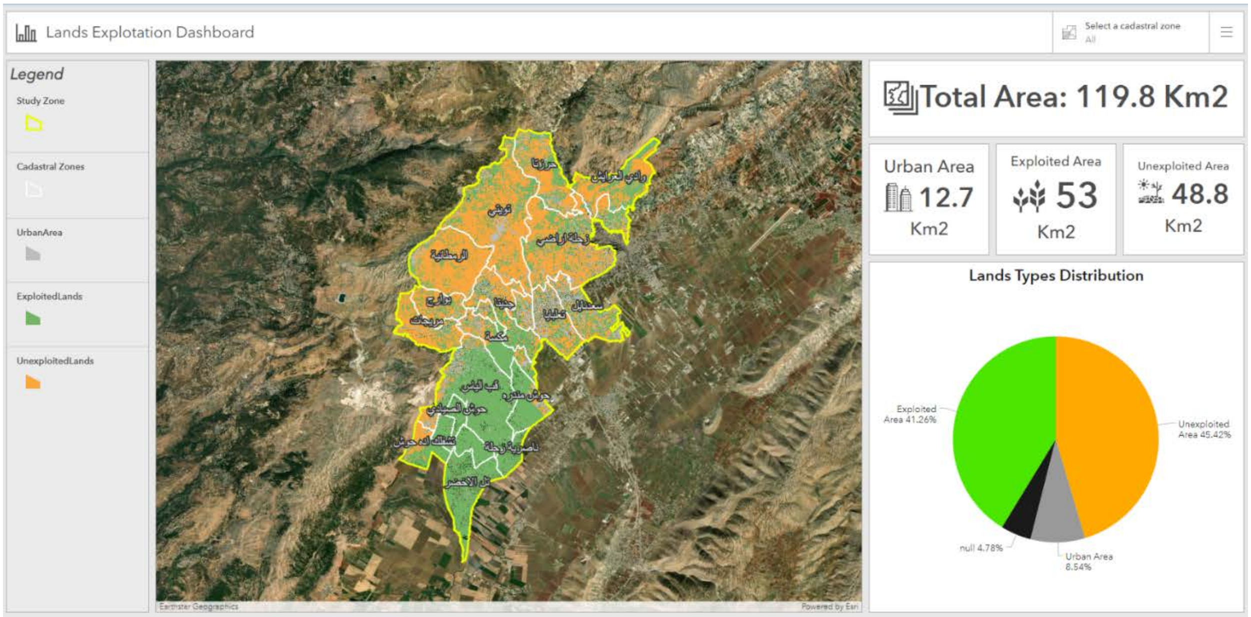


FIGURE 37 - SCREENSHOT OF THE LAND EXPLOITATION WEB APPLICATION

Users can use the drop-down list at the top to filter the indicator and the chart and zoom on the map into the target cadastral zone, to preview the same information in one cadastral zone.

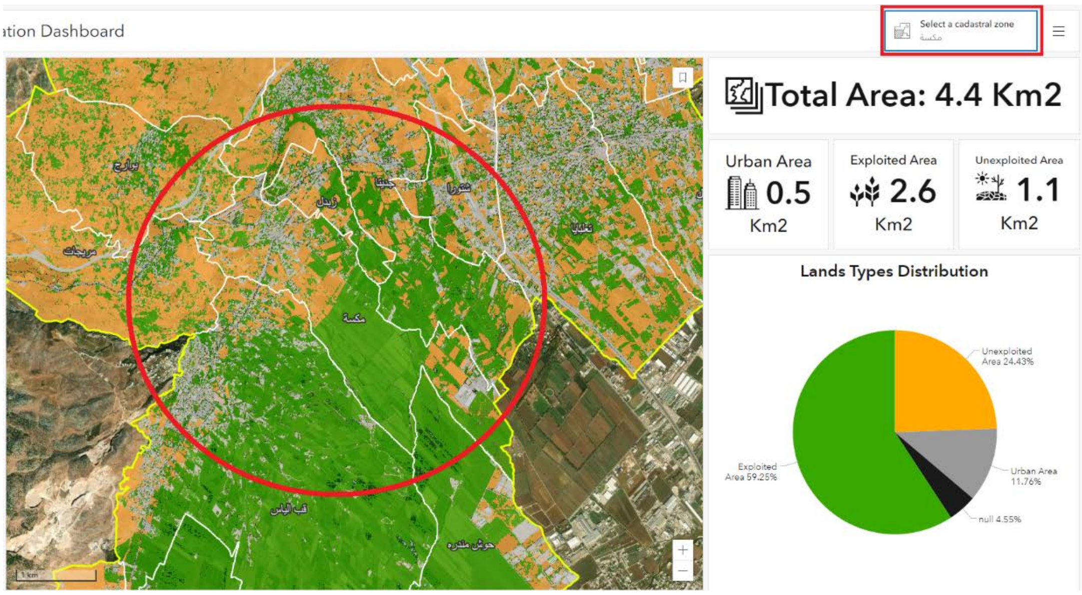


FIGURE 38 - THE FILTER CAPABILITY IN THE LAND EXPLOITATION DASHBOARD

When the user does not select any cadastral zone the application display the entire zone and clears the filter on the cadastral zone.

2.2. Lands Data Dashboard

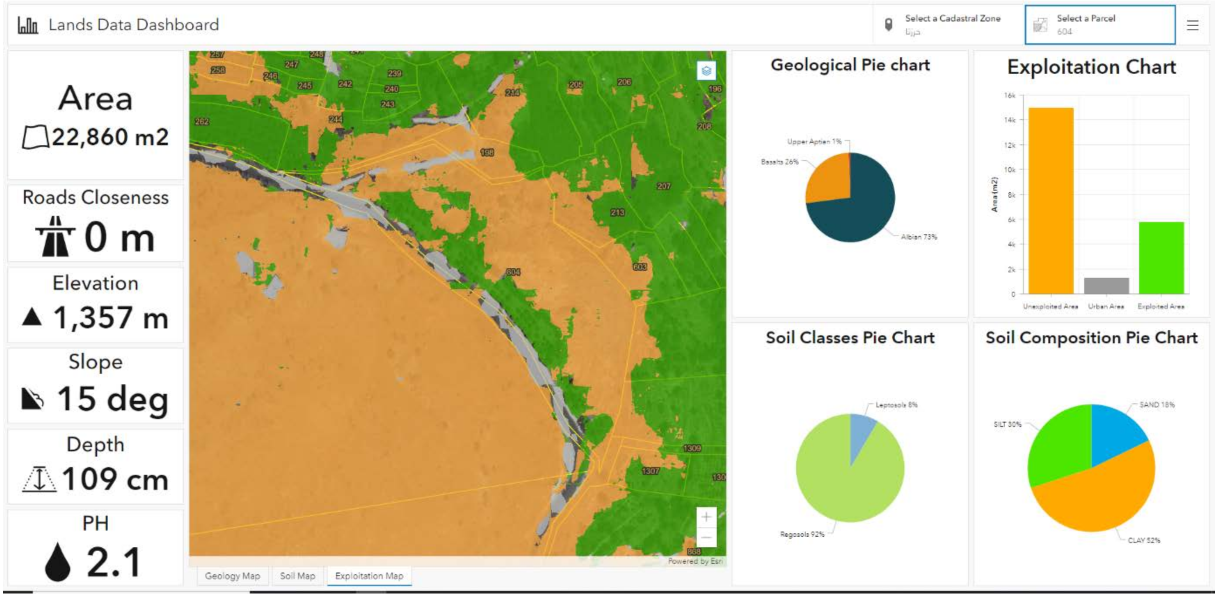
This Application is designed to report land data at the level of parcels. The user should select the cadastral zone and the parcel number to explore its data.

The dashboard contains six indicators at the left:

- The area of the selected parcel
- Its average distance from the closest road
- Average elevation
- Average slope
- Average soil depth
- Average soil PH

We have three maps in the middle containing the parcels layer:

- Geological map with the geological zones
- Soil map which has the soil classes
- The lands exploitation map.



Lands Data Dashboard

Select a Cadastral Zone
حزرتا

Select a Parcel
604

Area
22,860 m2

Roads Closeness
0 m

Elevation
1,357 m

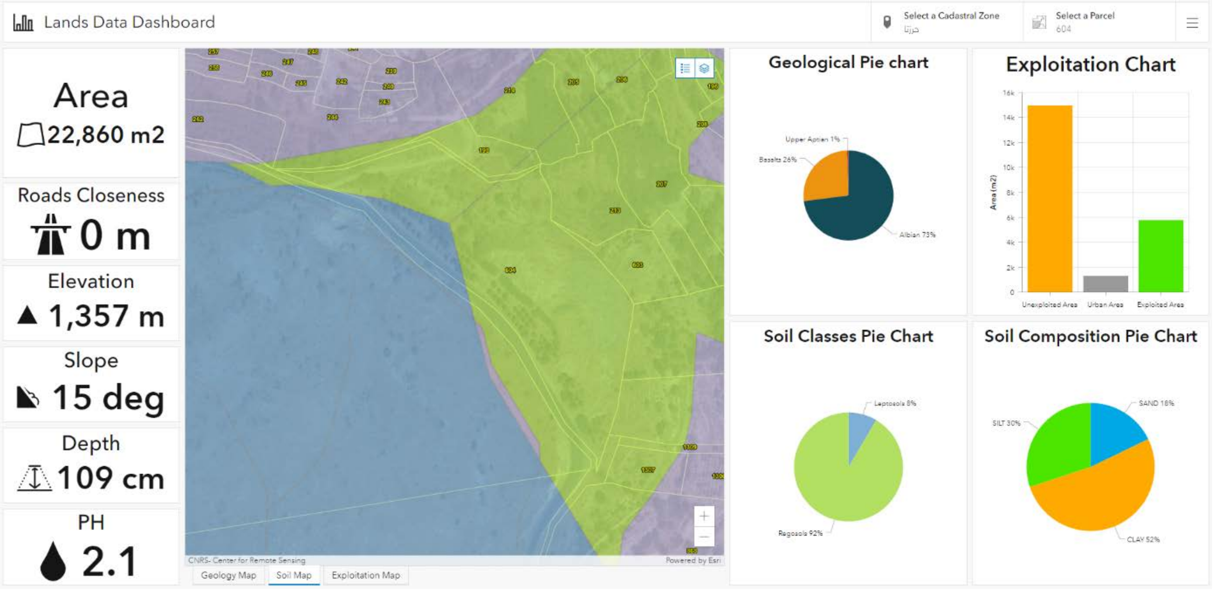
Slope
15 deg

Depth
109 cm

PH
2.1

FIGURE 39 – SCREENSHOT OF THE LAND DATA DASHBOARD

At the right, the dashboard preview the composition of the parcel in the geological and soil classes, also the soil composition using pie charts, in addition the exploitation zones using series chart.



Lands Data Dashboard

Select a Cadastral Zone: حرزتا

Select a Parcel: 604

Area: 22,860 m2

Roads Closeness: 0 m

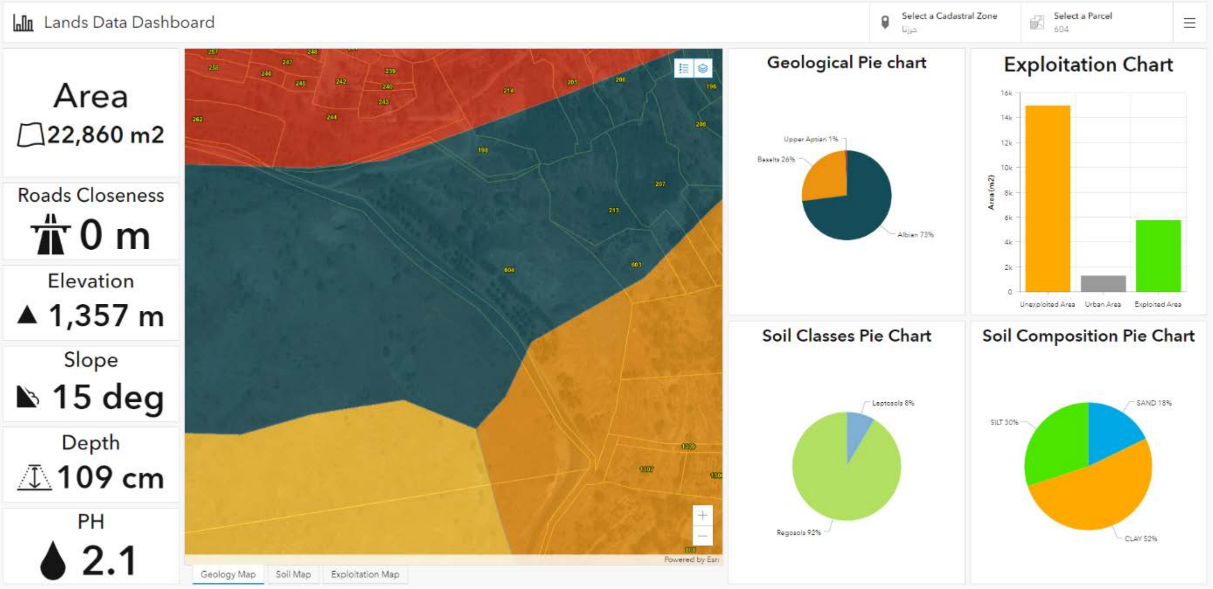
Elevation: 1,357 m

Slope: 15 deg

Depth: 109 cm

PH: 2.1

FIGURE 40 - SOIL MAP TAB IN THE LAND DATA DASHBOARD



Lands Data Dashboard

Select a Cadastral Zone: حرزتا

Select a Parcel: 604

Area: 22,860 m2

Roads Closeness: 0 m

Elevation: 1,357 m

Slope: 15 deg

Depth: 109 cm

PH: 2.1

FIGURE 41 - GEOLOGY TAB MAP IN THE LAND DATA DASHBOARD

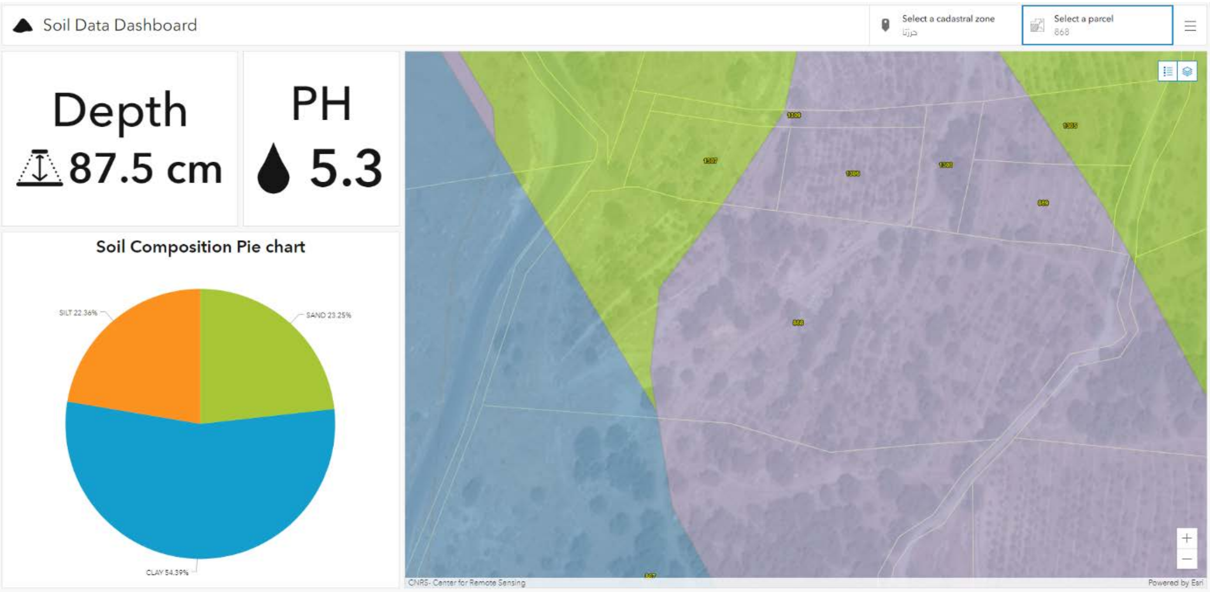
All the indicators and the charts will change according to the selected parcels. Zooming into this parcel in the maps is also performed.

2.3. Soil Data Dashboard

The aim of this dashboard is to explore the data related to the soil under the selected parcel.

Two indicators to the left show the average soil depth and the average PH. In addition, a pie chart represents the soil composition in the percentage of the parcel: % of silt, % sand, % clay.

The map to the right shows the overlay between the parcels and the soil class zones.



Soil Data Dashboard

Select a cadastral zone: حرزتا

Select a parcel: 868

Depth 87.5 cm

PH 5.3

Soil Composition Pie chart

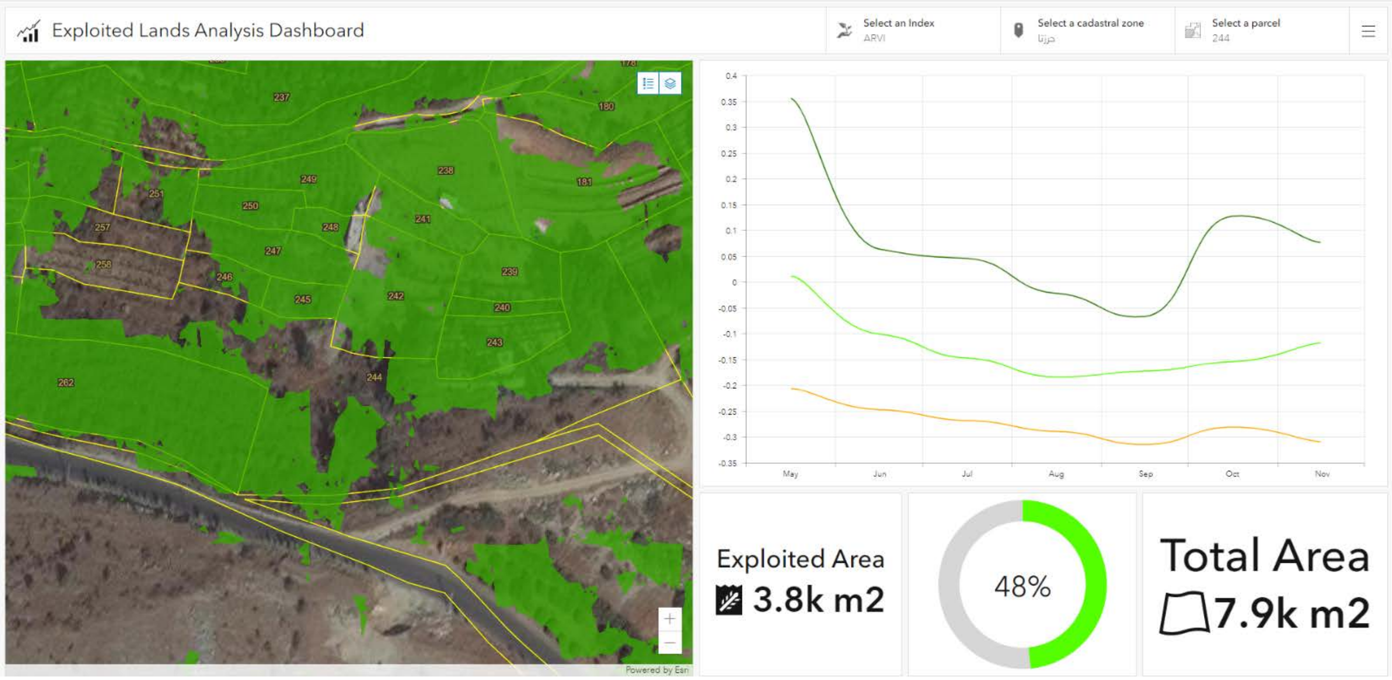
FIGURE 42 - SCREENSHOT OF THE SOIL DATA DASHBOARD

In addition, a selector dropdown user can use to filter the dashboard data to its target parcel.

2.4. Exploited Lands Analysis Dashboard

It is the crops monitoring application, this dashboard makes users to monitor and analyze the exploited lands in the function of time by providing the vegetation indices as a time series chart.

Users can filter the data by parcel and cadastral zone, and choose the target vegetation or soil index to get its max, min, and average chart over the year.



Exploited Lands Analysis Dashboard

Select an Index
ARVI

Select a cadastral zone
حرزتا

Select a parcel
244

Exploited Area
3.8k m2

48%

Total Area
7.9k m2

FIGURE 43 - SCREENSHOT OF THE EXPLOITED LANDS ANALYSIS DASHBOARD

In addition, you can preview the exploited area as a map and its percentage area from the entire parcel as an index and pie chart.

CHAPTER 6

Conclusion

As a conclusion, Data science and machine learning has played an important role to improve technology in wide industries. The agriculture sector has gained huge benefits from mixing GIS science and deep learning technology.

The deep learning model created at the beginning of this project has reduced a lot of efforts to find and extract lands and it can be an automated workflow and rerun every year to update the exploitation lands area.

The centralized geodatabase can be expended any time and populated with more data not at the quantity level only but also with a new type of data. A new ArcGIS product can be used here, which is the ArcGIS enterprise where we can power mapping and visualization, analytics, and data management to organize and share your work on any device, anywhere, at any time.

In addition, the enterprise geodatabase provides a good solution. It adds the ability to manage a shared, multiuser geodatabase as well as support for several critical version-based GIS workflows. The ability to leverage your organization's enterprise relational databases is a key advantage of the enterprise geodatabase.

In the end, the spatial analysis model was the most important factor in this project; it provides data extraction and data analysis to prepare data to be ready to visualize in the web maps and web applications where the decision makers can see what others cannot.

Integrating ESRI solutions can add many abilities and feasibility to the workflow. Using a product like ArcGIS insights can provide data exploring and perform advanced analytics such as spatial, statistical, predictive, and link analysis within an intuitive experience that works the way you do. Revolutionize decision-making with analysis that visually informs the organization of new, previously unexplored insights gained from the perspective of “where”.

Bibliography

[1] S. Saha, "Towardsdatascience," 15 10 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[2] Y. Hua, M. Diego, L. Mou, X. X. Zhu and D. Tuia, "Semantic Segmentation of Remote Sensing Images," p. 1, 2022.

[3] A. Kumar, "Image Segmentation: FCNet," 7 6 2020. [Online]. Available: https://medium.com/@abhishekkakiak/image-segmentation-fcnet-bff2d680e87a.

[4] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," Machine Learning Mastery, 3 7 2017. [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning.

[5] "Categorical crossentropy," Knowledge Center, [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy.

[6] J. Protasiewicz, "Python AI: Why Is Python So Good for Machine Learning?," netguru, 31 8 2018. [Online]. Available: https://www.netguru.com/blog/python-machine-learning.

[7] "Feature Class To Feature Class (Conversion)," esri, [Online]. Available: https://pro.arcgis.com/en/pro-app/2.8/tool-reference/conversion/feature-class-to-feature-class.htm.

[8] "Create Raster Dataset (Data Management)," esri, [Online]. Available: https://pro.arcgis.com/en/pro-app/2.8/tool-reference/data-management/create-raster-dataset.htm.

[9] "User Guides - Sentinel-1 SAR - Overview - Sentinel Online - Sentinel Online," European Space Agency Signature, [Online]. Available: https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/overview#:~:text=SENTINEL%2D1%20is%20an%20imaging,night%20imagery%20at%20C%2Dband..

[10] "What is NDVI (Normalized Difference Vegetation Index)?," GISGeography, 30 8 2022. [Online]. Available: https://gisgeography.com/ndvi-normalized-difference-vegetation-index/.

[11] "MSAVI: Monitor Crops At Earliest Growth Stages," EOS Data Analytics inc., [Online]. Available: https://eos.com/industries/agriculture/msavi/#:~:text=The%20modified%20soil%2Dadjusted%20vegetation,bare%20soil%20between%20the%20seedlings..

[12] "Landsat Enhanced Vegetation Index," U.S. Geological Survey, [Online]. Available: https://www.usgs.gov/landsat-missions/landsat-enhanced-vegetation-index.

[13] "CIg," esri, [Online]. Available: https://pro.arcgis.com/en/pro-app/2.8/arcpy/spatial-analyst/cig.htm.

[14] M. Agustín, "What is the GNDVI?," Auravant, 13 07 2022. [Online]. Available: https://help.auravant.com/en/articles/3636624-what-is-the-gndv.

[15] "Vegetation Indices Basics (SR – NDVI – PRI)," hiphen, [Online]. Available: https://www.hiphen-plant.com/vegetation-index/3582/.

[16] Saffer JD, "Introduction to biomedical literature text mining: context and objectives," [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/24788258.

[17] guru99.com. [Online]. Available: https://www.guru99.com/word-embedding-word2vec.html.

[18] M. Nayak. [Online]. Available: https://medium.com/towards-artificial-intelligence/an-intuitive-introduction-of-word2vec-by-building-a-word2vec-from-scratch-a1647e1c266c.

[19] T. &. O. H. Koiwa, "Extraction of disease-related genes from PubMed paper using word2vec," in *the 8th International Conference*, 2017.

[20] T. Peters, "Binary Classification on a Highly Imbalanced Dataset," in *Universiteit Utrecht*, Netherlands, 2018.

[21] S. Kurin, "A comparison of classification models for imbalanced datasets," Louvain School of Management, 2017.

[22] S. B. &. M. A. Chikh, "Medical imbalanced data classification," Tlemcen University, Algeria, 2017.

[23] A. Géron, Hands on Machine Learning with Scikit-Learn & TensorFlow., O'Reilly, 2017.

[24] J. Martinez, "Credit Fraud Detector," [Online]. Available: https://www.kaggle.com/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets.

[25] H. Su, "One class classification with Scikit Learn," 2015. [Online]. Available: http://www.hongyusu.com/imt/technology/one-class-classification-with-scikit.html.

[26] wikipedia, "One-class classification," [Online]. Available: https://en.wikipedia.org/wiki/One-class_classification.

Appendix

1. Deep Learning Model Code

```
import json
import numpy as np
import os
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from matplotlib.collections import PatchCollection
from matplotlib.patches import Rectangle

import gdal

import os
import shutil
import itertools
import random
import pandas as pd

import matplotlib.pyplot as plt
import tensorflow as tf

import keras

from keras import backend as K
from keras.applications import imagenet_utils
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.metrics import categorical_crossentropy

from keras.models import Sequential, Model, load_model


from keras.layers.core import Dense, Flatten
from keras.layers.convolutional import *
from sklearn.metrics import confusion_matrix
from scipy import misc, ndimage



from matplotlib import pyplot
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, MaxPool2D, Dense, Dropout
, Input, Flatten, Activation
```

```
from keras.layers import GlobalMaxPooling2D

from keras.layers.merge import Concatenate
from keras.models import Model
from keras import initializers
#from keras.optimizers import adam_v2

from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping

images_dir = 'Data/images/'
labels_dir = 'Data/labels/'
model_path = "Models/FinalModel2.h5"
randomstate=42

from sklearn.model_selection import train_test_split

from matplotlib.collections import PatchCollection
from matplotlib.patches import Rectangle

import gdal

import os
import shutil
import itertools
import random
import pandas as pd

import matplotlib.pyplot as plt
import tensorflow as tf

import keras

from keras import backend as K
from keras.applications import imagenet_utils
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.metrics import categorical_crossentropy

from keras.models import Sequential, Model, load_model


from keras.layers.core import Dense, Flatten
from keras.layers.convolutional import *
from sklearn.metrics import confusion_matrix
from scipy import misc, ndimage



from matplotlib import pyplot
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, MaxPool2D, Dense, Dropout
, Input, Flatten, Activation
from keras.layers import GlobalMaxPooling2D

from keras.layers.merge import Concatenate
from keras.models import Model
from keras import initializers
#from keras.optimizers import adam_v2

from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping

images_dir = 'Data/images/'
labels_dir = 'Data/labels/'
model_path = "Models/FinalModel2.h5"
randomstate=42
#tf.config.experimental_run_functions_eagerly(True)
tf.__version__
#tf.enable_eager_execution()
import warnings
warnings.filterwarnings('ignore')

import os
import cv2


import tensorflow as tf

from keras.models import Model, load_model
from skimage.morphology import label
import pickle
import tensorflow.keras.backend as K
from keras.models import Sequential
from matplotlib import pyplot as plt
from tqdm import tqdm_notebook
import random
from skimage.io import imread, imshow, imread_collection, concatenate_im
ages
from matplotlib import pyplot as plt
import h5py

seed = 56
from keras.models import Model, load_model
import tensorflow as tf
from keras.layers import Input
from keras.layers.core import Dropout, Lambda
from keras.layers.convolutional import Conv2D, Conv2DTranspose
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
from keras import optimizers
from keras.layers import BatchNormalization
from tensorflow.keras.metrics import MeanIoU
import keras
```

```
from matplotlib import pyplot
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, MaxPool2D, Dense,
Dropout, Input, Flatten, Activation
from keras.layers import GlobalMaxPooling2D

from keras.layers.merge import Concatenate
from keras.models import Model
from keras import initializers

from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping
#Resizing images, if needed
SIZE_X = 512
SIZE_Y = 512
n_classes=4
inputs = Input((512, 512, 3))


conv1 = Conv2D(16, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (inputs)
conv1 = BatchNormalization() (conv1)
conv1 = Dropout(0.1) (conv1)
conv1 = Conv2D(16, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv1)
conv1 = BatchNormalization() (conv1)
pooling1 = MaxPooling2D((2, 2)) (conv1)

conv2 = Conv2D(32, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (pooling1)
conv2 = BatchNormalization() (conv2)
conv2 = Dropout(0.1) (conv2)
conv2 = Conv2D(32, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv2)
conv2 = BatchNormalization() (conv2)
pooling2 = MaxPooling2D((2, 2)) (conv2)

conv3 = Conv2D(64, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (pooling2)
conv3 = BatchNormalization() (conv3)
conv3 = Dropout(0.2) (conv3)
conv3 = Conv2D(64, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv3)
conv3 = BatchNormalization() (conv3)
pooling3 = MaxPooling2D((2, 2)) (conv3)

conv4 = Conv2D(128, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (pooling3)
conv4 = BatchNormalization() (conv4)
conv4 = Dropout(0.2) (conv4)
conv4 = Conv2D(128, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv4)
conv4 = BatchNormalization() (conv4)
pooling4 = MaxPooling2D(pool_size=(2, 2)) (conv4)

conv5 = Conv2D(256, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (pooling4)
conv5 = BatchNormalization() (conv5)
conv5 = Dropout(0.3) (conv5)
conv5 = Conv2D(256, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv5)
conv5 = BatchNormalization() (conv5)


upsample6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')
 (conv5)
upsample6 = concatenate([upsample6, conv4])
conv6 = Conv2D(128, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (upsample6)
conv6 = BatchNormalization() (conv6)
conv6 = Dropout(0.2) (conv6)
conv6 = Conv2D(128, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv6)
conv6 = BatchNormalization() (conv6)

upsample7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')
(conv6)
upsample7 = concatenate([upsample7, conv3])
conv7 = Conv2D(64, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (upsample7)
conv7 = BatchNormalization() (conv7)
conv7 = Dropout(0.2) (conv7)
conv7 = Conv2D(64, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv7)
conv7 = BatchNormalization() (conv7)

upsample8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')
(conv7)
upsample8 = concatenate([upsample8, conv2])
conv8 = Conv2D(32, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (upsample8)
conv8 = BatchNormalization() (conv8)
conv8 = Dropout(0.1) (conv8)
conv8 = Conv2D(32, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv8)
conv8 = BatchNormalization() (conv8)

upsample9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')
(conv8)
upsample9 = concatenate([upsample9, conv1], axis=3)
conv9 = Conv2D(16, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (upsample9)
conv9 = BatchNormalization() (conv9)
```

```
conv9 = Dropout(0.1) (conv9)
conv9 = Conv2D(16, (3, 3), activation='elu',
kernel_initializer='he_normal', padding='same') (conv9)
conv9 = BatchNormalization() (conv9)



outputs = Conv2D(n_classes, (1, 1), activation='softmax') (conv9)



model = Model(inputs=[inputs], outputs=[outputs])
model.summary()
def createSet(images_dir,masks_dir,labels,count):
    images_list=[]
    masks_list=[]
    files = os.listdir(images_dir)
    n=0
    for f in sorted(files):
        mask=[]
        final_mask=None
        if n==count:
            break
        if f.endswith('.jpg'):
            images_list.append(cv2.imread(images_dir + '/' + f))
            print(f)
            n=n+1
            final_mask=np.zeros((512,512))
            for lf in labels:
                a=None
                maskfile=masks_dir + '/' + lf + '/' +
os.path.splitext(f)[0] + ".png"
                if os.path.exists(maskfile):
                    array=cv2.imread(maskfile,cv2.IMREAD_UNCHANGED)

                    if(len(array.shape)==2):
                        a=array
                    else:
                        a=np.sum(array,axis=2)
                    a[a>0]=lf
                    final_mask=final_mask+(final_mask==0)*a
            masks_list.append(final_mask)
    images_array =  np.asarray(images_list)
    del images_list
    masks_array = np.asarray(masks_list)
    del masks_list
    print(masks_array.dtype)
    return images_array.astype(np.uint8),masks_array.astype(np.uint8)
def plotImage(image):
    plt.imshow(image)
    plt.show()
labels=os.listdir(labels_dir)
```

```
images,masks=createSet(images_dir,labels_dir,labels,1800)
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
n, h, w = masks.shape
print("masks:",masks.shape)
masks_reshaped = masks.reshape(-1,1)
print("masks_reshaped:",masks_reshaped.shape)

print(np.unique(masks_reshaped))


del masks
print("start fiting")

labelencoder.fit(np.unique(masks_reshaped).reshape(-1,1))
masks_reshaped_encoded = labelencoder.transform(masks_reshaped)
#masks_reshaped_encoded = labelencoder.fit_transform(masks_reshaped)
masks_reshaped_encoded=masks_reshaped_encoded.astype(np.uint8)
print(masks_reshaped_encoded.dtype)

print("end fiting")
del masks_reshaped

masks_encoded_original_shape = masks_reshaped_encoded.reshape(n, h, w)

del masks_reshaped_encoded

print("masks_encoded_original_shape.unique:",
np.unique(masks_encoded_original_shape))
print("classes codes:",labelencoder.classes_)

from tensorflow.keras.utils import normalize
#images = normalize(images, axis=1)
print("images.shape:",images.shape)

masks_input = np.expand_dims(masks_encoded_original_shape, axis=3)

del masks_encoded_original_shape


#Create a subset of data for quick testing
#Picking 10% for testing and remaining for training
from sklearn.model_selection import train_test_split
X1, X_test, y1, y_test = train_test_split(images, masks_input,
test_size = 0.10, random_state = randomstate)
print("X1.shape:",X1.shape)

del images


#Further split training data t a smaller subset for quick testing of mod
els
```

```
X_train, X_val, y_train, y_val = train_test_split(X1, y1,
test_size = 0.2, random_state = randomstate)

del masks_input

#del X_do_not_use
#del y_do_not_use


print("Class values in the dataset are ... ", np.unique(y_train))  # 0 i
s the background/few unlabeled
y_train.reshape(-1,1)
from tensorflow.keras.utils import to_categorical

train_masks_cat = to_categorical(y_train.reshape(-1,1),
num_classes=n_classes)
y_train_cat = train_masks_cat.reshape((y_train.shape[0],
y_train.shape[1], y_train.shape[2], n_classes))

del y_train
del train_masks_cat

test_masks_cat = to_categorical(y_test, num_classes=n_classes)
y_test_cat = test_masks_cat.reshape((y_test.shape[0], y_test.shape[1],
y_test.shape[2], n_classes))


del test_masks_cat


val_masks_cat = to_categorical(y_val, num_classes=n_classes)
y_val_cat = val_masks_cat.reshape((y_val.shape[0], y_val.shape[1],
y_val.shape[2], n_classes))

#del y_val
del val_masks_cat
from keras import backend as K
def iou_coef(y_true, y_pred, smooth=1):
    y_true1=y_true.numpy()
    y_pred1=y_pred.numpy()
    y_true_argmax=np.argmax(y_true1, axis=3).astype(int)
    y_pred_argmax=np.argmax(y_pred1, axis=3).astype(int)
    y_true2=y_true_argmax.reshape(y_true_argmax.shape[0],
y_true_argmax.shape[1],y_true_argmax.shape[2],1)
    y_pred2=y_pred_argmax.reshape(y_pred_argmax.shape[0],
y_pred_argmax.shape[1],y_pred_argmax.shape[2],1)
    iou=0
    for i in range(1,n_classes):
      y_pred3=y_pred2*(y_pred2==i)
      y_true3=y_true2*(y_true2==i)
      y_pred3[y_pred3>0]=1
      y_true3[y_true3>0]=1
      intersection = K.sum(K.abs(y_true3* y_pred3), axis=[1,2,3])
      union = K.sum(y_true3,[1,2,3])+K.sum(y_pred3,[1,2,3])-intersection
      iou =iou+K.mean((intersection + smooth) / (union + smooth),
axis=0)
    return iou/(n_classes-1)

def iou_coef1(y_true, y_pred, smooth=1):
    y_true1=y_true.numpy()
    y_pred1=y_pred.numpy()
    y_true_argmax=np.argmax(y_true1, axis=3).astype(int)
    y_pred_argmax=np.argmax(y_pred1, axis=3).astype(int)
    y_true2=y_true_argmax.reshape(y_true_argmax.shape[0],
y_true_argmax.shape[1],y_true_argmax.shape[2],1)
    y_pred2=y_pred_argmax.reshape(y_pred_argmax.shape[0],
y_pred_argmax.shape[1],y_pred_argmax.shape[2],1)
    iou=0
    for i in range(1,2):
      y_pred3=y_pred2*(y_pred2==i)
      y_true3=y_true2*(y_true2==i)
      y_pred3[y_pred3>0]=1
      y_true3[y_true3>0]=1
      intersection = K.sum(K.abs(y_true3* y_pred3), axis=[1,2,3])
      union = K.sum(y_true3,[1,2,3])+K.sum(y_pred3,[1,2,3])-intersection
      iou =iou+K.mean((intersection + smooth) / (union + smooth),axis=0)
    return iou/(n_classes-1)

def iou_coef2(y_true, y_pred, smooth=1):
    y_true1=y_true.numpy()
    y_pred1=y_pred.numpy()
    y_true_argmax=np.argmax(y_true1, axis=3).astype(int)
    y_pred_argmax=np.argmax(y_pred1, axis=3).astype(int)
    y_true2=y_true_argmax.reshape(y_true_argmax.shape[0],
y_true_argmax.shape[1],y_true_argmax.shape[2],1)
    y_pred2=y_pred_argmax.reshape(y_pred_argmax.shape[0],
y_pred_argmax.shape[1],y_pred_argmax.shape[2],1)
    iou=0
    for i in range(2,3):
      y_pred3=y_pred2*(y_pred2==i)
      y_true3=y_true2*(y_true2==i)
      y_pred3[y_pred3>0]=1
      y_true3[y_true3>0]=1
      intersection = K.sum(K.abs(y_true3* y_pred3), axis=[1,2,3])
      union = K.sum(y_true3,[1,2,3])+K.sum(y_pred3,[1,2,3])-intersection
      iou =iou+K.mean((intersection + smooth) / (union + smooth),
axis=0)
    return iou


def iou_coef3(y_true, y_pred, smooth=1):
    y_true1=y_true.numpy()
    y_pred1=y_pred.numpy()
    y_true_argmax=np.argmax(y_true1, axis=3).astype(int)
    y_pred_argmax=np.argmax(y_pred1, axis=3).astype(int)
    y_true2=y_true_argmax.reshape(y_true_argmax.shape[0],
y_true_argmax.shape[1],y_true_argmax.shape[2],1)
    y_pred2=y_pred_argmax.reshape(y_pred_argmax.shape[0],
y_pred_argmax.shape[1],y_pred_argmax.shape[2],1)
    iou=0
    for i in range(3,4):
      y_pred3=y_pred2*(y_pred2==i)
      y_true3=y_true2*(y_true2==i)
      y_pred3[y_pred3>0]=1
      y_true3[y_true3>0]=1
      intersection = K.sum(K.abs(y_true3* y_pred3), axis=[1,2,3])
      union = K.sum(y_true3,[1,2,3])+K.sum(y_pred3,[1,2,3])-
intersection
      iou =iou+K.mean((intersection + smooth) / (union + smooth),
axis=0)
    return iou


def simple_iou_coef(y_true, y_pred, smooth=1):
    iou=0
    for i in range(1,n_classes):
      y_true1=y_true[:,:,:,i].reshape(y_true.shape[0],y_true.shape[1],
y_true.shape[2],1).astype(int)
      y_pred1=y_pred[:,:,:,i].reshape(y_pred.shape[0],y_pred.shape[1],
y_pred.shape[2],1).astype(int)
      intersection = K.sum(K.abs(y_true1* y_pred1), axis=[1,2,3])
      union = K.sum(y_true1,[1,2,3])+K.sum(y_pred1,[1,2,3])-
intersection
      iou =iou+K.mean((intersection + smooth) / (union + smooth),
axis=0)
    return iou/(n_classes-1)



def loss(y_true, y_pred, smooth=1):
    loss=0
    for i in range(1,y_true.shape[3]):
      intersection = K.sum(y_true[:,:,:,i]* y_pred[:,:,:,i])+ smooth
      union = K.sum(y_true[:,:,:,i]) + K.sum(y_pred[:,:,:,i])-
intersection + smooth
      loss =loss+ K.mean((intersection + smooth) / (union + smooth))
    loss = loss / (y_true.shape[3])
    return 1-loss


def F1_score(y_true, y_pred, smooth=1):
    y_true1=y_true.numpy()
    y_pred1=y_pred.numpy()
    y_true_argmax=np.argmax(y_true1, axis=3).astype(int)
    y_pred_argmax=np.argmax(y_pred1, axis=3).astype(int)
    y_true2=y_true_argmax.reshape(y_true_argmax.shape[0],
y_true_argmax.shape[1],y_true_argmax.shape[2],1)
    y_pred2=y_pred_argmax.reshape(y_pred_argmax.shape[0],
y_pred_argmax.shape[1],y_pred_argmax.shape[2],1)
    F1_score=0
    for i in range(1,n_classes):
      y_pred3=y_pred2*(y_pred2==i)
      y_true3=y_true2*(y_true2==i)
      y_pred3[y_pred3>0]=1
      y_true3[y_true3>0]=1
      intersection = K.sum(K.abs(y_true3* y_pred3), axis=[1,2,3])
      union = K.sum(y_true3,[1,2,3])+K.sum(y_pred3,[1,2,3])
      F1_score =F1_score+K.mean(2*(intersection + smooth) /
(union + smooth), axis=0)
    return F1_score/(n_classes-1)



def Precision(y_true, y_pred, smooth=1):
    y_true1=y_true.numpy()
    y_pred1=y_pred.numpy()
    y_true_argmax=np.argmax(y_true1, axis=3).astype(int)
    y_pred_argmax=np.argmax(y_pred1, axis=3).astype(int)
    y_true2=y_true_argmax.reshape(y_true_argmax.shape[0],
y_true_argmax.shape[1],y_true_argmax.shape[2],1)
    y_pred2=y_pred_argmax.reshape(y_pred_argmax.shape[0],
y_pred_argmax.shape[1],y_pred_argmax.shape[2],1)
    Precision=0
    for i in range(1,n_classes):
      y_pred3=y_pred2*(y_pred2==i)
      y_true3=y_true2*(y_true2==i)
      y_pred3[y_pred3>0]=1
      y_true3[y_true3>0]=1
      intersection = K.sum(K.abs(y_true3* y_pred3), axis=[1,2,3])
      union = K.sum(y_pred3,[1,2,3])
      Precision =Precision+K.mean((intersection + smooth) /
(union + smooth), axis=0)
    return Precision/(n_classes-1)

def Recall(y_true, y_pred, smooth=1):
    y_true1=y_true.numpy()
    y_pred1=y_pred.numpy()
    y_true_argmax=np.argmax(y_true1, axis=3).astype(int)
    y_pred_argmax=np.argmax(y_pred1, axis=3).astype(int)
    y_true2=y_true_argmax.reshape(y_true_argmax.shape[0],
y_true_argmax.shape[1],y_true_argmax.shape[2],1)
    y_pred2=y_pred_argmax.reshape(y_pred_argmax.shape[0],
y_pred_argmax.shape[1],y_pred_argmax.shape[2],1)
    Recall=0
    for i in range(1,n_classes):
      y_pred3=y_pred2*(y_pred2==i)
      y_true3=y_true2*(y_true2==i)
      y_pred3[y_pred3>0]=1
      y_true3[y_true3>0]=1
      intersection = K.sum(K.abs(y_true3* y_pred3), axis=[1,2,3])
      union = K.sum(y_true3,[1,2,3])
      Recall =Recall+K.mean((intersection + smooth) / (union + smooth),
axis=0)
    return Recall/(n_classes-1)
EPOCHS = 50
LEARNING_RATE = 0.001
BATCH_SIZE = 8
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPl
ateau
from datetime import datetime

checkpointer = ModelCheckpoint(model_path,save_best_only = True,verbose=
1)

lr_reducer = ReduceLROnPlateau(monitor='val_loss',
                               factor=0.1,
                               patience=4,
                               verbose=1,
                               epsilon=1e-4)
from sklearn.utils import class_weight



from tensorflow.keras.optimizers import Adam
opt = Adam(lr=LEARNING_RATE)

mIOU = tf.keras.metrics.MeanIoU(num_classes=n_classes)


#model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=[
'accuracy',tf.keras.metrics.MeanIoU(num_classes=n_classes)])
model.compile(run_eagerly=True,optimizer=opt, loss='categorical_crossent
ropy', metrics=['accuracy',iou_coef1,iou_coef2,iou_coef3,Precision,Recal
l,F1_score])
#model.summary()
history = model.fit(X_train, y_train_cat,
                    batch_size = BATCH_SIZE,
                    verbose=1,
                    epochs=EPOCHS,
                    validation_data=(X_test, y_test_cat),
                    shuffle=False,
                    callbacks = [checkpointer, lr_reducer])

del X_train
del y_train_cat
def GenerateMask(Sourceimage,model,sizex,sizey):
  height = Sourceimage.shape[0]
  width = Sourceimage.shape[1]
  mask = np.zeros((1,height,width))
  i=0
  while i+sizey < height :
        j=0
        while j+sizex < width:
          image=Sourceimage[i:i+sizey,j:j+sizex,:]
          print(image.shape)
          image=image.reshape(1,512,512,3)
          y_pred=model.predict(image)
          y_pred_argmax=np.argmax(y_pred, axis=3)
          print("i=",i," - ","j=",j)
          #print(y_pred_argmax.shape)
          mask[:,i:i+512,j:j+512]= y_pred_argmax
          j=j+sizex
        i=i+sizey
  return mask.astype(np.int8)

image_path ='/content/drive/MyDrive/Data_deeplearning/Multiple RCNN/test
.jpg'
image=cv2.imread(image_path)
print(image.shape)
mask= GenerateMask(image,model,SIZE_X,SIZE_Y)
```

2. Spatial Analysis Python Toolbox Code

```
# -*- coding: utf-8 -*-

import arcpy

def ImportToMosaic(IndexName,Date,Raster,mosaicdataset):
    DateFormatted=Date.strftime("%Y_%m_%d")
    arcpy.management.AddRastersToMosaicDataset(mosaicdataset,"Raster
Dataset",Raster)
    arcpy.CalculateStatistics_management(mosaicdataset)
    desc = arcpy.Describe(mosaicdataset)
    with arcpy.da.Editor(desc.path) as edit:
        arcpy.MakeMosaicLayer_management(mosaicdataset, 'mosaic')
        ep = IndexName + "_" + DateFormatted
        Expression = "Name= '"+ ep+"'"
        rows=arcpy.UpdateCursor('mosaic\Footprint', Expression)
        for r in rows:
            r.Date=Date
            r.Code=IndexName
            rows.updateRow(r)
    return
```

```
def makeMosaic(MosaicDataSet,LayerMosaic,Month,Year):
        arcpy.MakeMosaicLayer_management(MosaicDataSet, LayerMosaic)
        Expression = 'EXTRACT(MONTH FROM "Date") = ' + str(Month) + ' AND
EXTRACT(Year FROM "Date") = ' + str(Year)
        rows=arcpy.SearchCursor(LayerMosaic + '\Footprint', Expression)
        OID=0
        for row in rows :
            OID=row.OBJECTID
            Date=row.Date
            break
        SelectedRaster=MosaicDataSet + r"\raster.objectid="+str(OID)
        Raster = arcpy.Raster(SelectedRaster,True)
        return Raster,Date

def SelectRastersFromMosaic(MosaicDataSet,LayerMosaic,Month,Year):
        arcpy.MakeMosaicLayer_management(MosaicDataSet, LayerMosaic)
        Expression = 'EXTRACT(MONTH FROM "Date") = ' + str(Month) + ' AND
EXTRACT(Year FROM "Date") = ' + str(Year)
        #arcpy.AddMessage(Expression)
        rows=arcpy.SearchCursor(LayerMosaic + '\Footprint', Expression)
        Rasters=[]
        RasterNames=[]
        for row in rows :
            Date=row.Date
            SelectedRaster=MosaicDataSet +
r"\raster.objectid="+str(row.OBJECTID)
            Rasters.append(arcpy.Raster(SelectedRaster,True))
            RasterNames.append(row.Code)
        return Rasters,RasterNames,Date



class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = "Toolbox"
        self.alias = "toolbox"

        # List of tool classes associated with this toolbox
        self.tools =
[ImportSentinelOneDataTool,ImportSentinelTwoDataTool,ImportParcelsTool,ImportE
xtractedLandsDataTool,CreateIndicesTool,CalculatStatisticalIndicesTool,Calcula
tPermanentStatisticalIndicesTool,GenerateSuitabilityScoreTool,FindSuitabilityZ
onesTool]
```

```
class ImportSentinelOneDataTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "import Sentinel1 Data"
        self.description = "import Sentinel 1 data and convert it to one
raster with multiple bands"
        self.canRunInBackground = False

    def getParameterInfo(self):
        param1 = arcpy.Parameter(displayName="Sigma 0 -
HV",name="SigmaHV",datatype="DERasterDataset",parameterType="Required",directi
on="Input")
        param2 = arcpy.Parameter(displayName="Sigma 0 -
VV",name="SigmaVV",datatype="DERasterDataset",parameterType="Required",directi
on="Input")
        param3 = arcpy.Parameter(displayName="Date of
Image",name="Date",datatype="GPDate",parameterType="Required",direction="Input
")
        param4 = arcpy.Parameter(displayName="Target Mosaic
dataset",name="mosaicdataset",datatype="GPMosaicLayer",parameterType="Required
",direction="Input")
        param5 = arcpy.Parameter(displayName="Clipping Feature
Class",name="ClippingFeature",datatype="DEFeatureClass",parameterType="Optiona
l",direction="Input")
        param6 = arcpy.Parameter(displayName="Images
Folder",name="OutputFolder",datatype="DEFolder",parameterType="Required",direc
tion="Input")
        params = [param1,param2,param3,param4,param5,param6]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        SigmaHV=parameters[0].valueAsText
        SigmaVV=parameters[1].valueAsText
        Date_str=parameters[2].valueAsText
        mosaicdataset=parameters[3].valueAsText
        ClippingFeature=parameters[4].valueAsText
        out_folder=parameters[5].valueAsText

        bands=SigmaHV+";"+SigmaVV+";"
        arcpy.CompositeBands_management(bands,arcpy.env.workspace + r"\temp1")
        arcpy.Clip_management(arcpy.env.workspace + r"\temp1", "#",
arcpy.env.workspace + r"\temp2", ClippingFeature, "#", "ClippingGeometry",
"NO_MAINTAIN_EXTENT")
        arcpy.Delete_management(arcpy.env.workspace + r"\temp1")
        from dateutil import parser
        Date = parser.parse(Date_str)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\SentinelOne_" + DateFormatted + ".tif"
        arcpy.CopyRaster_management(arcpy.env.workspace + r"\temp2",imagepath)
        arcpy.Delete_management(arcpy.env.workspace + r"\temp2")
        arcpy.management.AddRastersToMosaicDataset(mosaicdataset,"Raster
Dataset",imagepath)
        arcpy.CalculateStatistics_management(mosaicdataset)
        desc = arcpy.Describe(mosaicdataset)
        with arcpy.da.Editor(desc.path) as edit:
            arcpy.MakeMosaicLayer_management(mosaicdataset, 'mosaic')
            ep = "SentinelOne_" + DateFormatted
            Expression = "Name= '"+ ep+"'"
            rows=arcpy.UpdateCursor('mosaic\Footprint', Expression)
            for r in rows:
                r.Date=Date
                rows.updateRow(r)
        return




class ImportSentinelTwoDataTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "import Sentinel2 Data"
        self.description = "import Sentinel 2 data and convert it to one
raster with multiple bands"
        self.canRunInBackground = False

    def getParameterInfo(self):
        param1 = arcpy.Parameter(displayName="Band 2  Blue
Image",name="Band2",datatype="DERasterDataset",parameterType="Required",direct
ion="Input")
```

```
        param2 = arcpy.Parameter(displayName="Band 3  Green Image",name="Band3",datatype="DERasterDataset",parameterType="Required",direction="Input")
        param3 = arcpy.Parameter(displayName="Band 4  Red Image",name="Band4",datatype="DERasterDataset",parameterType="Required",direction="Input")
        param4 = arcpy.Parameter(displayName="Band 8  IRed Image",name="Band8",datatype="DERasterDataset",parameterType="Required",direction="Input")
        param8 = arcpy.Parameter(displayName="Date of Image",name="Date",datatype="GPDate",parameterType="Required",direction="Input")
        param5 = arcpy.Parameter(displayName="Target Mosaic dataset",name="mosaicdataset",datatype="GPMosaicLayer",parameterType="Required",direction="Input")
        param6 = arcpy.Parameter(displayName="Clipping Feature Class",name="ClippingFeature",datatype="DEFeatureClass",parameterType="Optional",direction="Input")
        param7 = arcpy.Parameter(displayName="Images Folder",name="OutputFolder",datatype="DEFolder",parameterType="Required",direction="Input")
        params = [param1,param2,param3,param4,param5,param6,param7,param8]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        Band2=parameters[0].valueAsText
        Band3=parameters[1].valueAsText
        Band4=parameters[2].valueAsText
        Band8=parameters[3].valueAsText
        mosaicdataset=parameters[4].valueAsText
        ClippingFeature=parameters[5].valueAsText
        out_folder=parameters[6].valueAsText
        Date_str=parameters[7].valueAsText
        bands=Band4+";"+Band3+";"+Band2+";"+Band8+";"
        arcpy.CompositeBands_management(bands,arcpy.env.workspace + r"\temp1")
        arcpy.Clip_management(arcpy.env.workspace + r"\temp1", "#",
arcpy.env.workspace + r"\temp2", ClippingFeature, "#", "ClippingGeometry",
"NO_MAINTAIN_EXTENT")
        arcpy.Delete_management(arcpy.env.workspace + r"\temp1")
        from dateutil import parser
        Date = parser.parse(Date_str)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\SentinelTwo_" + DateFormatted + ".tif"
        arcpy.CopyRaster_management(arcpy.env.workspace + r"\temp2",imagepath)
        arcpy.Delete_management(arcpy.env.workspace + r"\temp2")
        arcpy.management.AddRastersToMosaicDataset(mosaicdataset,"Raster
Dataset",imagepath)
        arcpy.CalculateStatistics_management(mosaicdataset)
        desc = arcpy.Describe(mosaicdataset)
        with arcpy.da.Editor(desc.path) as edit:
            arcpy.MakeMosaicLayer_management(mosaicdataset, 'mosaic')
            ep = "SentinelTwo_" + DateFormatted
            Expression = "Name= '"+ ep+"'"
            rows=arcpy.UpdateCursor('mosaic\Footprint', Expression)
            for r in rows:
                r.Date=Date
                rows.updateRow(r)
        return



class ImportParcelsTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Import Parcels"
        self.description = "Import Parcels from cad file"
        self.canRunInBackground = False

    def getParameterInfo(self):
        param1=arcpy.Parameter(displayName="Parcels Autocad
File",name="ParcelsCad",datatype="CAD Drawing
Dataset",parameterType="Required",direction="Input")
        param2=arcpy.Parameter(displayName="VillageId",name="VillageId",dataty
pe="Long",parameterType="Required",direction="Input")
        param3=arcpy.Parameter(displayName="Parcels Feature
Class",name="ParcelFeatureClass",datatype="DEFeatureClass",parameterType="Requ
ired",direction="Input")
        params = [param1,param2,param3]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        return

    def execute(self, parameters, messages):
        ParcelsCad=parameters[0].valueAsText
        VillageId=parameters[1].valueAsText
        ParcelFeatureClass=parameters[2].valueAsText
        Lines=ParcelsCad + "\\Polyline"
        Annotations=ParcelsCad + "\\Annotation"
        arcpy.CopyFeatures_management(Lines,arcpy.env.workspace+"\\parclesline
s")
        arcpy.CopyFeatures_management(Annotations,arcpy.env.workspace+"\\parce
lsanno")
        sr = arcpy.SpatialReference(4326)
        arcpy.DefineProjection_management(arcpy.env.workspace+"\\parcleslines"
, sr)
        arcpy.DefineProjection_management(arcpy.env.workspace+"\\parcelsanno",
sr)
        arcpy.FeatureToPolygon_management(arcpy.env.workspace+"\\parcleslines"
,arcpy.env.workspace+"\\parcelspolygon")
        arcpy.SpatialJoin_analysis(arcpy.env.workspace+"\\parcelspolygon",arcp
y.env.workspace+"\\parcelsanno",arcpy.env.workspace+"\\"
"completedparcels","JOIN_ONE_TO_ONE","#","#","COMPLETELY_CONTAINS")
        dropFields=arcpy.ListFields(arcpy.env.workspace+"\\completedparcels")
        for f in dropFields:
            if f.name != "Text" and f.name != "FID" and f.name != "Shape" and
f.name != "OBJECTID" and f.name != "Shape_Length" and f.name != "Shape_Area" :
                arcpy.DeleteField_management(arcpy.env.workspace+"\\completedp
arcels" , f.name)
        arcpy.AddField_management(arcpy.env.workspace+"\\completedparcels","Vi
llageId","LONG")
        arcpy.CalculateField_management(arcpy.env.workspace+"\\completedparcel
s" ,"VillageId",VillageId)
        arcpy.AddField_management(arcpy.env.workspace+"\\completedparcels","Nu
mber","Text")
        arcpy.CalculateField_management(arcpy.env.workspace+"\\completedparcel
s", "Number", "!Text!","PYTHON3")
        arcpy.DeleteField_management(arcpy.env.workspace+"\\completedparcels",
"Text")
```

```
        arcpy.Delete_management(arcpy.env.workspace+"\\parcleslines")
        arcpy.Delete_management(arcpy.env.workspace+"\\parcelspolygon")
        arcpy.Delete_management(arcpy.env.workspace+"\\parcelsanno")
        arcpy.management.Append(arcpy.env.workspace+"\\completedparcels",Parce
lFeatureClass)
        arcpy.Delete_management(arcpy.env.workspace+"\\completedparcels")
        return


class ImportExtractedLandsDataTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Import Extracted Lands"
        self.description = "import Extracted Lands"
        self.canRunInBackground = False

    def getParameterInfo(self):
        param1 = arcpy.Parameter(displayName="Extracted
Lands",name="Lands",datatype="DERasterDataset",parameterType="Required",direct
ion="Input")
        param5 = arcpy.Parameter(displayName="Date of
Image",name="Date",datatype="GPDate",parameterType="Required",direction="Input
")
        param2 = arcpy.Parameter(displayName="Target Mosaic
dataset",name="mosaicdataset",datatype="GPMosaicLayer",parameterType="Required
",direction="Input")
        param3 = arcpy.Parameter(displayName="Clipping Feature
Class",name="ClippingFeature",datatype="DEFeatureClass",parameterType="Optiona
l",direction="Input")
        param4 = arcpy.Parameter(displayName="Images
Folder",name="OutputFolder",datatype="DEFolder",parameterType="Required",direc
tion="Input")
        params = [param1,param2,param3,param4,param5]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        Lands=parameters[0].valueAsText
        mosaicdataset=parameters[1].valueAsText
        ClippingFeature=parameters[2].valueAsText
        out_folder=parameters[3].valueAsText
        Date_str=parameters[4].valueAsText
        from dateutil import parser
        Date = parser.parse(Date_str)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\ExtractedLands_" + DateFormatted + ".tif"
        arcpy.CopyRaster_management(Lands,imagepath)
        arcpy.management.AddRastersToMosaicDataset(mosaicdataset,"Raster 
Dataset",imagepath)
        desc = arcpy.Describe(mosaicdataset)
        with arcpy.da.Editor(desc.path) as edit:
            arcpy.MakeMosaicLayer_management(mosaicdataset, 'mosaic')
            ep = "ExtractedLands_" + DateFormatted
            Expression = "Name= '"+ ep+"'"
            rows=arcpy.UpdateCursor('mosaic\Footprint', Expression)
            for r in rows:
                r.Date=Date
                rows.updateRow(r)
        return


class CreateIndicesTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Create Indices"
        self.description = "Create Indices and import them to a mosaic 
dataset"
        self.canRunInBackground = False

    def getParameterInfo(self):
        param1 = arcpy.Parameter(displayName="Sentinel 1 
dataset",name="SentinelOne",datatype="GPMosaicLayer",parameterType="Required",
direction="Input")
        param2 = arcpy.Parameter(displayName="Sentinel 2 
dataset",name="SentinelTwo",datatype="GPMosaicLayer",parameterType="Required",
direction="Input")
        param3 = 
arcpy.Parameter(displayName="Month",name="Month",datatype="GPLong",parameterTy
pe="Required",direction="Input")
        param3.filter.type = "Range"
        param3.filter.list = [1, 12]
        param4 =
arcpy.Parameter(displayName="Year",name="Year",datatype="GPLong",parameterType
="Required",direction="Input")
        param4.filter.type = "Range"
        param4.filter.list = [2016, 2100]
        param5 = arcpy.Parameter(displayName="Target Indices Mosaic
dataset",name="mosaicdataset",datatype="GPMosaicLayer",parameterType="Required
",direction="Input")
        param6 = arcpy.Parameter(displayName="Images
Folder",name="OutputFolder",datatype="DEFolder",parameterType="Required",direc
tion="Input")
        #param3 = arcpy.Parameter(displayName="Elevation
DEM",name="DEM",datatype="DERasterDataset",parameterType="Required",direction=
"Input")
        #param4 =
arcpy.Parameter(displayName="Slope",name="Slope",datatype="DERasterDataset",pa
rameterType="Required",direction="Input")
        #param5 = arcpy.Parameter(displayName="Distance From
Roads",name="DistRoad",datatype="DERasterDataset",parameterType="Required",dir
ection="Input")

        params = [param1,param2,param3,param4,param5,param6]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return




    def execute(self, parameters, messages):
        SentinelOne=parameters[0].valueAsText
        SentinelTwo=parameters[1].valueAsText
        Month=parameters[2].valueAsText
        Year=parameters[3].valueAsText
```

```
mosaicdataset=parameters[4].valueAsText
out_folder=parameters[5].valueAsText



SentinelOneRaster,Date=makeMosaic(SentinelOne,'mosaic',Month,Year)

VH = arcpy.ia.ExtractBand(SentinelOneRaster, [1])
VV = arcpy.ia.ExtractBand(SentinelOneRaster, [2])
Index="RVI"
Expression='(4*VH)/(VH+VV)'
output = arcpy.ia.RasterCalculator([VH,VV],["VH","VV"],Expression)
DateFormatted=Date.strftime("%Y_%m_%d")
imagepath = out_folder + r"\\" + Index + "_" + DateFormatted + ".tif"
output.save(imagepath)
ImportToMosaic(Index,Date,imagepath,mosaicdataset)



SentinelTwoRaster,Date=makeMosaic(SentinelTwo,'mosaic',Month,Year)
R = arcpy.ia.ExtractBand(SentinelTwoRaster, [1])
G = arcpy.ia.ExtractBand(SentinelTwoRaster, [2])
B = arcpy.ia.ExtractBand(SentinelTwoRaster, [2])
IR = arcpy.ia.ExtractBand(SentinelTwoRaster, [2])

Index="NDVI"
Expression='(IR-R)/(IR+R)'
output = arcpy.ia.RasterCalculator([IR,R],["IR","R"],Expression)
DateFormatted=Date.strftime("%Y_%m_%d")
imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
output.save(imagepath)
ImportToMosaic(Index,Date,imagepath,mosaicdataset)

Index="ARVI"
Expression='(IR-2*R+B)/(IR+2*R-B)'
output = arcpy.ia.RasterCalculator([IR,R,B],["IR","R","B"],Expression)
DateFormatted=Date.strftime("%Y_%m_%d")
imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
output.save(imagepath)
ImportToMosaic(Index,Date,imagepath,mosaicdataset)

Index="MSAVI"
Expression='0.5*((2*IR+1)-SquareRoot(Square(2*IR+1)-(8*IR-R)))'
output = arcpy.ia.RasterCalculator([IR,R],["IR","R"],Expression)
DateFormatted=Date.strftime("%Y_%m_%d")
imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
output.save(imagepath)
ImportToMosaic(Index,Date,imagepath,mosaicdataset)
```

```
        Index="EVI"
        Expression='2.5*(IR-R)/(IR+6*R+7.5*B+1)'
        output = arcpy.ia.RasterCalculator([IR,R,B],["IR","R","B"],Expression)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
        output.save(imagepath)
        ImportToMosaic(Index,Date,imagepath,mosaicdataset)

        Index="CIG"
        Expression='(IR/G)-1'
        output = arcpy.ia.RasterCalculator([IR,G],["IR","G"],Expression)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
        output.save(imagepath)
        ImportToMosaic(Index,Date,imagepath,mosaicdataset)

        Index="GNDVI"
        Expression='(IR-G)/(IR+G)'
        output = arcpy.ia.RasterCalculator([IR,G],["IR","G"],Expression)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
        output.save(imagepath)
        ImportToMosaic(Index,Date,imagepath,mosaicdataset)

        Index="IOI"
        Expression='R/B'
        output = arcpy.ia.RasterCalculator([R,B],["R","B"],Expression)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
        output.save(imagepath)
        ImportToMosaic(Index,Date,imagepath,mosaicdataset)

        Index="SR"
        Expression='IR/R'
        output = arcpy.ia.RasterCalculator([IR,R],["IR","R"],Expression)
        DateFormatted=Date.strftime("%Y_%m_%d")
        imagepath = out_folder + r"\\" + Index +  "_" + DateFormatted + ".tif"
        output.save(imagepath)
        ImportToMosaic(Index,Date,imagepath,mosaicdataset)

        return


class CalculatStatisticalIndicesTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Calculate Statistical Indices"
        self.description = "Calculate Statistical Indices and import them to
the Statistical Indices table"
        self.canRunInBackground = False

    def getParameterInfo(self):
        param1 = arcpy.Parameter(displayName="Unexploited
Lands",name="UnexploitedLands",datatype="DERasterDataset",parameterType="Requi
red",direction="Input")
        param2 = arcpy.Parameter(displayName="Exploited
Lands",name="ExploitedLands",datatype="DERasterDataset",parameterType="Require
d",direction="Input")
        param3 = arcpy.Parameter(displayName="Indices
Table",name="IndicesTable",datatype="DETable",parameterType="Required",directi
on="Input")
        param4 =
arcpy.Parameter(displayName="Month",name="Month",datatype="GPLong",parameterTy
pe="Required",direction="Input")
        param4.filter.type = "Range"
        param4.filter.list = [1, 12]
        param5 =
arcpy.Parameter(displayName="Year",name="Year",datatype="GPLong",parameterType
="Required",direction="Input")
        param5.filter.type = "Range"
        param5.filter.list = [2016, 2100]
        param6 = arcpy.Parameter(displayName="Indices Mosaic
dataset",name="mosaicdataset",datatype="GPMosaicLayer",parameterType="Required
",direction="Input")
        param7 = arcpy.Parameter(displayName="Parcels Feature
Class",name="Parcels",datatype="DEFeatureClass",parameterType="Required",direc
tion="Input")

        #param3 = arcpy.Parameter(displayName="Elevation
DEM",name="DEM",datatype="DERasterDataset",parameterType="Required",direction=
"Input")
        #param4 =
arcpy.Parameter(displayName="Slope",name="Slope",datatype="DERasterDataset",pa
rameterType="Required",direction="Input")
        #param5 = arcpy.Parameter(displayName="Distance From
Roads",name="DistRoad",datatype="DERasterDataset",parameterType="Required",dir
ection="Input")

        params = [param1,param2,param3,param4,param5,param6,param7]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True
    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return




    def execute(self, parameters, messages):
        UnexploitedLands=parameters[0].valueAsText
        ExploitedLands=parameters[1].valueAsText
        IndicesTable=parameters[2].valueAsText
        Month=parameters[3].valueAsText
        Year=parameters[4].valueAsText
        IndicesMosaicSataset=parameters[5].valueAsText
        Parcels=parameters[6].valueAsText



        #LandsRaster,Date=makeMosaic(Lands,'mosaic',Month,Year)
        #Expression='Con((L==2))'
        #outputLands = arcpy.ia.RasterCalculator([Lands],["L"],Expression)
        #outputLands = arcpy.sa.Con(Lands,1,0,"VALUE = 2")


        IndexRasters,Names,Date=SelectRastersFromMosaic(IndicesMosaicSataset,'
mosaic',Month,Year)
        for indexRaster,Name in zip(IndexRasters,Names) :
            arcpy.AddMessage(Name)
            Expression='L*i'
            output1 =
arcpy.ia.RasterCalculator([indexRaster,UnexploitedLands],["L","i"],Expression)
            output2 =
arcpy.ia.RasterCalculator([indexRaster,ExploitedLands],["L","i"],Expression)
            DateFormatted=Date.strftime("%Y_%m_%d")

            arcpy.AddMessage("Unexploited Area1")
            arcpy.sa.ZonalStatisticsAsTable(Parcels,
"ID",output1,arcpy.env.workspace + r"\tempTable1", "DATA", "MIN_MAX_MEAN")
            arcpy.AddMessage("Unexploited Area2")
            Searchrows=arcpy.SearchCursor(arcpy.env.workspace + 
r"\tempTable1")
            row_values=[]
            for row in Searchrows :
                ParcelID=row.ID
                Min=row.Min
                Mean=row.Mean
                Max=row.Max
                row_values.append((ParcelID,Date,3,Name,Min,Mean,Max))
            cursor = arcpy.da.InsertCursor(IndicesTable,['ParcelID', 
'Date','Class','Index','Min','Mean','Max'])
            for row in row_values:
                cursor.insertRow(row)
            del cursor

            arcpy.AddMessage("Exploited Area1")
            arcpy.sa.ZonalStatisticsAsTable(Parcels, 
"ID",output2,arcpy.env.workspace + r"\tempTable2", "DATA", "MIN_MAX_MEAN")
            arcpy.AddMessage("Exploited Area2")
            Searchrows=arcpy.SearchCursor(arcpy.env.workspace + 
r"\tempTable2")
            row_values=[]
            for row in Searchrows :
                ParcelID=row.ID
                Min=row.Min
                Mean=row.Mean
                Max=row.Max
                row_values.append((ParcelID,Date,2,Name,Min,Mean,Max))
            cursor = arcpy.da.InsertCursor(IndicesTable,['ParcelID', 
'Date','Class','Index','Min','Mean','Max'])
            for row in row_values:
                cursor.insertRow(row)
            del cursor
            arcpy.management.Delete(arcpy.env.workspace + r"\tempTable1")
            arcpy.management.Delete(arcpy.env.workspace + r"\tempTable2")

        return


class CalculatPermanentStatisticalIndicesTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Calculate Permanent Statistical Indices"
        self.description = "Calculate Permanent Statistical Indices and import 
them to the Permanent Statistical Indices table"
        self.canRunInBackground = False

    def getParameterInfo(self):
        param1 = arcpy.Parameter(displayName="Unexploited
Lands",name="UnexploitedLands",datatype="DERasterDataset",parameterType="Requi
red",direction="Input")
        param2 = arcpy.Parameter(displayName="Exploited
Lands",name="ExploitedLands",datatype="DERasterDataset",parameterType="Require
d",direction="Input")
        param3 = arcpy.Parameter(displayName="Index
Name",name="IndexName",datatype="GPString",parameterType="Required",direction=
"Input")
        param4 = arcpy.Parameter(displayName="Index
Raster",name="IndexRaster",datatype="DERasterDataset",parameterType="Required"
,direction="Input")
        param5 = arcpy.Parameter(displayName="Permanent Indices
Table",name="PermanentIndicesTable",datatype="DETable",parameterType="Required
",direction="Input")
        param6 = arcpy.Parameter(displayName="Parcels Feature
Class",name="Parcels",datatype="DEFeatureClass",parameterType="Required",direc
tion="Input")


        params = [param1,param2,param3,param4,param5,param6]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return





    def execute(self, parameters, messages):
        UnexploitedLands=parameters[0].valueAsText
        ExploitedLands=parameters[1].valueAsText
        IndexName=parameters[2].valueAsText
        IndexRaster=parameters[3].valueAsText
        IndicesTable=parameters[4].valueAsText
        Parcels=parameters[5].valueAsText

        Expression='L*i'
        output1 = 
arcpy.ia.RasterCalculator([IndexRaster,UnexploitedLands],["L","i"],Expression)
        output2 = 
arcpy.ia.RasterCalculator([IndexRaster,ExploitedLands],["L","i"],Expression)
        arcpy.sa.ZonalStatisticsAsTable(Parcels, 
"ID",output1,arcpy.env.workspace + r"\tempTable1", "DATA", "MIN_MAX_MEAN")
        Searchrows=arcpy.SearchCursor(arcpy.env.workspace + r"\tempTable1")
        row_values=[]
        for row in Searchrows :
            ParcelID=row.ID
            Min=row.Min
            Mean=row.Mean
            Max=row.Max
            row_values.append((ParcelID,3,IndexName,Min,Mean,Max))
        cursor = 
arcpy.da.InsertCursor(IndicesTable,['ParcelID','Class','Index','Min','Mean','M
ax'])
        for row in row_values:
            cursor.insertRow(row)
        del cursor


        arcpy.sa.ZonalStatisticsAsTable(Parcels, 
"ID",output2,arcpy.env.workspace + r"\tempTable2", "DATA", "MIN_MAX_MEAN")
        Searchrows=arcpy.SearchCursor(arcpy.env.workspace + r"\tempTable2")
        row_values=[]
        for row in Searchrows :
            ParcelID=row.ID
            Min=row.Min
            Mean=row.Mean
            Max=row.Max
            row_values.append((ParcelID,2,IndexName,Min,Mean,Max))
        cursor = 
arcpy.da.InsertCursor(IndicesTable,['ParcelID','Class','Index','Min','Mean','M
ax'])
        for row in row_values:
            cursor.insertRow(row)
        del cursor
        arcpy.management.Delete(arcpy.env.workspace + r"\tempTable1")
        arcpy.management.Delete(arcpy.env.workspace + r"\tempTable2")

        return
```

```
class GenerateSuitabilityScoreTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Generate Suitability Score"
        self.description = "Calculate Suitability Score by merging multiple
rasters using weigthed suitability calculation"
        self.canRunInBackground = False

    def getParameterInfo(self):

        """Define parameter definitions"""


        param1 = arcpy.Parameter(displayName = "Selected records",name =
"value_table",datatype = "GPValueTable",parameterType =
"Optional",direction="Input")
        param1.columns =([["DERasterDataset", "Raster Dataset"],
["GPLong","Percentage"],["GPBoolean","Inverse Index"]])
        param2 = arcpy.Parameter(displayName="Min
Value",name="Min",datatype="GPLong",parameterType="Required",direction="Input"
)
        param3 = arcpy.Parameter(displayName="Max
Value",name="Max",datatype="GPLong",parameterType="Required",direction="Input"
)
        param4 = arcpy.Parameter(displayName="Suitability
Raster",name="Output",datatype="DERasterDataset",parameterType="Required",dire
ction="Output")
        param5 = arcpy.Parameter(displayName="Parcels Feature
Class",name="Parcels",datatype="DEFeatureClass",parameterType="Optional",direc
tion="Input")
        param6 = arcpy.Parameter(displayName="Suitable
Parcels",name="SParcels",datatype="DEFeatureClass",parameterType="Optional",di
rection="Output")
        params = [param1,param2,param3,param4,param5,param6]

        return params

    def isLicensed(self):

        """Set whether tool is licensed to execute."""

        return True

    def updateParameters(self, parameters):

        """Modify the values and properties of parameters before internal

        validation is performed.  This method is called whenever a parameter
        has been changed."""

        return

    def updateMessages(self, parameters):

        """Modify the messages created by internal validation for each tool

        parameter.  This method is called after internal validation."""
        p=0
        for i in range(0,len(parameters[0].values)):
            p=p+parameters[0].values[i][1]
        if p != 100:
            parameters[0].setErrorMessage("Sum of Percentages should be 100")

        return

    def execute(self, parameters, messages):

        Min=parameters[1].value
        Max=parameters[2].value
        Output=parameters[3].valueAsText
        Parcels=parameters[4].valueAsText
        OutputParcels=parameters[5].valueAsText
        outputs=[]
        rasters=[]
        exp=""
        for i in range(0,len(parameters[0].values)):
            raster = arcpy.Raster(str(parameters[0].values[i][0]))
            p=parameters[0].values[i][1]
            min=raster.minimum
            max=raster.maximum
            if min==max:
                if min<0:
                    max=0
                else:
                    min=0
            Expression='((R-{0})/({1}-{0}))*({3}-{2})'.format(min,max,Min,Max)
            if (str(parameters[0].values[i][2]) == "True") :
                Expression=str(Max) + "-(" + Expression + ")"
            outputs.append(arcpy.ia.RasterCalculator([raster],["R"],Expression
))
            rasters.append("R"+str(i))
            exp=exp+'{0}*{1}+'.format("R"+str(i),p)
        exp="(" + exp[:-1] + ")/100"
        final=arcpy.ia.RasterCalculator(outputs,rasters,exp)
        final.save(Output)
        if Parcels != '':
            arcpy.sa.ZonalStatisticsAsTable(Parcels,
"ID",Output,arcpy.env.workspace + r"\tempTable1", "DATA", "MEAN")
            arcpy.AddField_management(Parcels, "Mean", "DOUBLE")
            arcpy.MakeTableView_management(arcpy.env.workspace +
r"\tempTable1", "table")
            arcpy.MakeFeatureLayer_management(Parcels, "parcels")
            arcpy.AddJoin_management("parcels","ID", "table", "ID")
            arcpy.CalculateField_management("parcels", "parcels.Mean",
"!tempTable1.MEAN!","PYTHON")
            arcpy.Delete_management(arcpy.env.workspace + r"\tempTable1")
        return


class FindSuitabilityZonesTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Find Suitability Zone"
        self.description = "Find Suitability Zone by merging multiple rasters
using Binary suitability calculation"
        self.canRunInBackground = False

    def getParameterInfo(self):

        """Define parameter definitions"""
        param1 = arcpy.Parameter(displayName = "Selected records",name =
"value_table",datatype = "GPValueTable",parameterType =
"Optional",direction="Input")
        param1.columns =([["DERasterDataset", "Raster Dataset"],
["GPDouble","Min"],["GPDouble","Max"]])
        param2 = arcpy.Parameter(displayName="Suitability
Raster",name="Output",datatype="DERasterDataset",parameterType="Required",dire
ction="Output")
        param3 = arcpy.Parameter(displayName="Parcels Feature
Class",name="Parcels",datatype="DEFeatureClass",parameterType="Optional",direc
tion="Input")
        param4 = arcpy.Parameter(displayName="Suitable
Parcels",name="SParcels",datatype="DEFeatureClass",parameterType="Optional",di
rection="Output")
        params = [param1,param2,param3,param4]

        return params

    def isLicensed(self):

        """Set whether tool is licensed to execute."""
        return True
    def updateParameters(self, parameters):

        return

    def updateMessages(self, parameters):

        """Modify the messages created by internal validation for each tool

        parameter.  This method is called after internal validation."""

        return

    def execute(self, parameters, messages):


        Output=parameters[1].valueAsText
        Parcels=parameters[2].valueAsText
        OutputParcels=parameters[3].valueAsText
        outputs=[]
        rasters=[]
        exp=""
        for i in range(0,len(parameters[0].values)):
            raster = arcpy.Raster(str(parameters[0].values[i][0]))
            min=parameters[0].values[i][1]
            max=parameters[0].values[i][2]
            outputs.append(arcpy.sa.Con(((raster >= min) & (raster <= max)),
1, 0))
            rasters.append("R"+str(i))
            exp=exp+'{0}*'.format("R"+str(i))
        arcpy.AddMessage(exp)
        exp=exp[:-1]
        final=arcpy.ia.RasterCalculator(outputs,rasters,exp)
        final.save(Output)
        if Parcels != '':
            arcpy.sa.ZonalStatisticsAsTable(Parcels,
"ID",Output,arcpy.env.workspace + r"\tempTable1", "DATA", "MEAN")
            arcpy.AddField_management(Parcels, "Mean", "DOUBLE")
            arcpy.MakeTableView_management(arcpy.env.workspace +
r"\tempTable1", "table")
            arcpy.MakeFeatureLayer_management(Parcels, "parcels")
            arcpy.AddJoin_management("parcels","ID", "table", "ID")
            arcpy.management.SelectLayerByAttribute("parcels",
"NEW_SELECTION", "tempTable1.MEAN > 0.9")
            arcpy.conversion.FeatureClassToFeatureClass("parcels",
arcpy.env.workspace, "Result")
            arcpy.Delete_management(arcpy.env.workspace + r"\tempTable1")
        return
```