Environmental damage assessment based on satellite imagery using machine learning

Luca Zelioli 1800293
Master of Science Thesis
Supervisors: Prof. Johan Lilius, Dr. Bogdan Iancu
Faculty of Science and Engineering
Åbo Akademi
2019

Vorrei dedicare questa tesi alla mia famiglia che in questi anni mi ha supportato. Un particolare grazie alla mia compagna anche. Impossibile dimenticare la mia seconda famiglia, gli amici di Santa Caterina Valfurva I miei piu fedeli compagni di avventure.

Luca

Abstract

The aim of this thesis is to provide a source of information about damage assessment in forestry using deep learning. A large source of environmental information is provided by satellites imagery. Orbital devices are equipped with sensors that read the frequency variations in the terrestrial electromagnetic field. The information obtained by these devices is composed by collections of dots. Machine learning methodologies, however, have the ability to transform raw data into human-understandable output.

Cloud and blur represent artefacts that need to be tackled to obtain high-quality imagery. For instance, a deep learning neural network, a Generative Adversarial Neural Network, can extrapolate the cloud compound from the image. Moreover, resampling techniques are used to improve their resolution. In this way, it is possible to correct the overall quality of satellite data.

The Finnish Kvarken Region, situated in the province of Vaasa, comprises a delicate forestry zone. Climate changes and the rise of temperature are influencing the forest quality negatively. Moreover, the public company in charge of the operational management needs new tools in order to enhance the environment condition.

Plenty of satellite data analysis frameworks are available for the consumer. In particular, SNAP, QGIS and ArcGIS offer capabilities to analyze environmental damage. Moreover, Google Earth Engine uses powerful programming languages such as Python to elaborate information from the Kvarken Region. It is also possible to study the historic forestry change from the past years until today. Unsupervised and supervised machine learning models are used to underline the difference between techniques. Deforested areas in the Kvarken Region are mapped using state-of-the-art deep learning architectures for image segmentation. The implementation is done using Python programming language and open source libraries such as TensorFlow and Keras.

Keywords: forest assessment, satellite imagery, remote sensing.

Acknowledgement

First of all, I would like to thank the KvarkenSpaceEco Project that gave me the possibility to study and increment my knowledge in computer vision, combining my passion for the environment. I am deeply grateful to Professor Johan Lilius and Dr. Bogdan Iancu for the excellent advices and support. Without the guidance and persistent help of all the Professors, Docents and Personals, of the Åbo Akademi University my studies would have been impossible.

Contents

INTRODUCTION

Computer vision aims to imitate the human vision by electronically understanding the content of images [1]. Camera sensors are usually used to capture images, but during the acquisition time, plenty of quality details are lost due to the incapacity of hardware to perfectly replicate the reality. For example, the geometric properties of the captured scene are re-built using mathematical models. The interpretation of images is a long process in computer vision. Computers need a large amount of data in order to understand items inside images, for humans only previous knowledge of it is necessary [1]. Moreover, one can summarize the logic behind with the following formula:

$$Computer \quad vision \quad interpretation: \quad image \quad data \quad -> \quad model. \tag{1.1}$$

There are multiple extra components that come within images [1]: the noise is present at the moment of the frame acquisition but mathematical algorithms are able to tackle it. Modern camera sensors capture data and before the image representation, pre-processing algorithms digitalize the samples in order to remove the noise. This is attained by multiple phases, for example, capturing phase, processing phase or transmission. The noise is identified by studying the constant power spectrum of the image, because it does not change in intensity. Moreover, it is different than the signal noise because the frequency remains constant. There are two special cases of noise, Gaussian noise and adaptive noise [1]. Sometimes noise is also used for constructive purposes, for example it is possible to detect faces of people and apply blur to them.

The edges of a picture can be enhanced to improve the quality. The process of image creation requires plenty of processing and memory power. Image quality depends on brightness because some objects reflect light and lower the final quality of the picture. Edges are very important in computer vision, because they describe the change in intensity of the neighbour's pixels. Techniques such as optical flow combined with the study of edges show the direction of an object in multiple images. Moreover, the combination between edge and contours is used to extract objects.

Colours are essential components of images. For humans the visible wavelength is from 380 nm to 740 nm [1], for computers it is larger. It is possible to convert the not visible wavelength so that humans have the possibility to see it. They are represented in RGB (red green and blue) format, expressed as a mixture of the primary colours vector. Furthermore, it is very useful for cluster analysis in order to separate the image regions.

The data involved in the various steps of image processing is very different: in the low level it is represented by large arrays of numbers, in the upper layer the content is filtered and only the important features are extracted. However, this process does not follow a specific path, as different operations involve particular algorithms.

Moreover, the computer vision approach attempts to create a relation between an input frame and a pre-made model [1]. Machine vision uses algorithms to correlate the raw image data and the interpretation. At the bottom level, computers have no knowledge about the content, however, top-level Artificial Intelligence methods are used to understand the content of the image [1]. For example, the segmentation process is used to divide the image in different regions to extract features. The border analysis of every segment is used to divide regions. Modern neural networks use this methodology in order to separate the background from an object.

Classification is used for object identification. Machine vision needs previous human knowledge in order to interpret images because at machine level, object identification happens by comparing arrays from raw data and samples provided by humans.

Mathematical models are used to represent images, in 2D coordinates [1]. However, this is the result of porting an image from the real-world scene that is a 3D space, through a process called *perspective projection* [1]. Furthermore, other factors are archived with an image: intensity of the light, surface reflectance and source of brightness. Fortunately, these components are static [1] and they are removed by mathematical algorithms to make clear the object detection. The image captured by a camera sensor is digitalized: this process is divided into two parts, sampling and quantization. The first one refers to the division of pixels representing the image in a grid divided by rows and columns. Rasterization defines the relationship between points in a grid [1]. Moreover, better sampling means good quality of the image. Quantization assigns integer value to every continuous sample of the image [1].

1.1 Motivation

The WMO report on The Global Climate between 2015 and 2019 notifies a great acceleration in climate change with dangerous effect on the environment. Perennial glaciers are becoming smaller, raising the level of water in seas and oceans. For example, the Forni glacier, situated in Santa Caterina Valfurva, a small town in North Italy, and situated between 2600 and 3600 meters above sea level, is descending about 2 meters per year, which confirms that global temperature is rising. This situation has also affected the forest in the area. High temperatures, storms and humans are contributing to damaging the forest, in Finland and around the world. About 75% of Finland is covered with forest, with high contribution of oxygen for Northern Europe. For this reason, it is important to discourage environmental damage to the forest ecosystem remarking how dangerous is the actual climate situation.

It is important that the *Global Community* becomes more sensitive for the climate situation, because it affects human health. If the situation does not change, forest areas around the world will be permanently damaged. However, Finland uses a method to preserve the forest areas by removing old trees and planting new ones.

The aim of this thesis is to present machine learning models able to assess damage, focusing on deforestation. Moreover, the analysis focuses on damage caused by humans or natural disasters. The work is concentrated to two different branches. In unsupervised learning, algorithms like K-Means clustering, PCA analysis, Gaussian Mixed Models and Self-Organizing Maps are used in remote-sensing applications. These models are able to divide the input in clusters given only a few parameters. In the deep learning section, three different models are described. For this kind of analysis, labels need to be provided in order to make the analysis work. Two models are described in this section. The *U-Net* is a very powerful model that is able to discover forestry damage, however, *SegNet* is another good algorithm but does not guarantee satisfactory predictions. In the end, a TensorFlow model with a framework called *Pix2Pix* is studied as well.

Furthermore, different datasets are tested in order to verify how the models perform. However, deeper studies are necessary in order to classify the damage caused by human or by natural disasters. The inquiry focuses on the Kvarken Region. However, the solution is applicable to other areas as well.

1.2 Contribution

I was honoured to present preliminary results of this Master's thesis to the "Finnish Satellite Workshop 2020" with a poster: "Machine Learning methods for environmental damage assessment from satellite imagery, a case of study: The Kvarken Region"

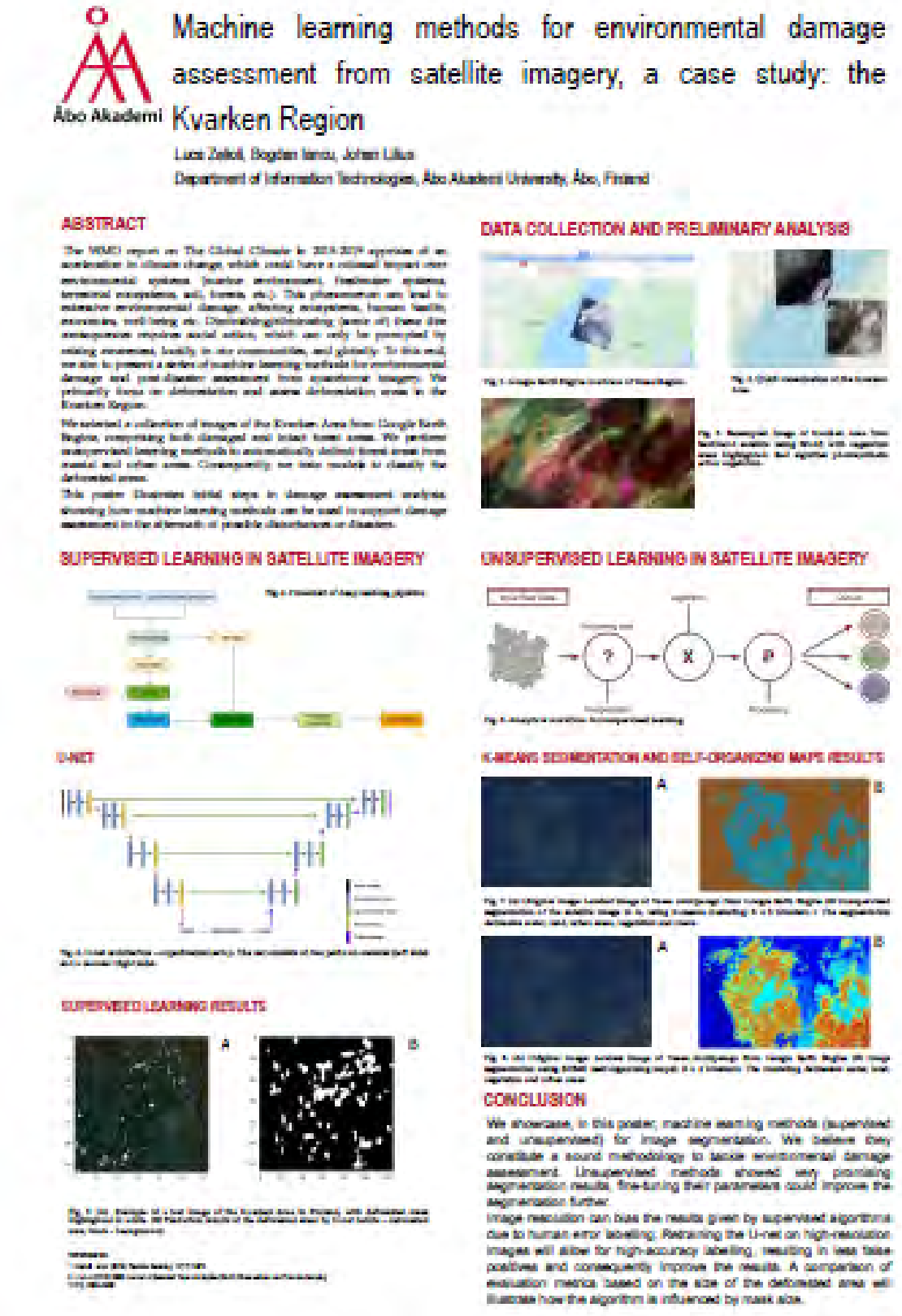


Figure 1.1: Poster showed during the Finnish Satellite Workshop 2020

1.3 Thesis structure

The thesis commences with a brief introduction about the current state-of-the-art in remote sensing technology and essential concepts of computer vision.

In Chapter 2, Background, relevant information such as the definition of the electromagnetic spectrum is given, followed by a short description of the hardware used for remote imagery. At this point, the process of image acquisition by cameras or sensors is described. In the related work section, the Kvarken Region is introduced with valuable information about the geographical conformation of the archipelago and forest distribution in the area. Examples from Sentinel-2 satellite images can be found there as well.

The Resources Chapter describes the current and already made applications of remote sensing. In particular with SNAP, ArcGIS and QGIS software it is possible to conduct analysis with a few clicks and settings. Moreover, the RGB analysis uses specialized algorithms, one can detect from the input image the healthy vegetation of the Kvarken Region. The Google Earth Engine plays an important role for this field because it includes a satisfactory archive of Sentinel, Landsat and Modis data where some of them are updated daily.

The Method Chapter briefly describes the history of *Artificial Intelligence* followed by a machine learning introduction. Mathematical concepts behind the various models used in the field are described. In the Unsupervised learning section, a description for each method used for the forestry damage assessment is reported: *K-Means* clustering, *PCA analysis* and the *Self-Organizing Maps*. The following section is dedicated to the description of the deep learning methodologies: first a general description of them followed by an extensive mathematical review of various activation functions and sensing methodologies. One of the most used backbones for Neural Network, TensorFlow, is briefly described.

The Implementation Chapter starts with an analysis about the forestry situation of the Kvarken Region, the process is done using Python programming language and a dataset found in the Google Earth Engine archive. Moreover, various images, tables and one graph illustrate how many trees are lost from the year 2000 to 2017. In the following section, the workflow used to study deforestation is described. All the packages used to gather results are summarized in tables exercise by exercise. The PCA analysis shows how to extract the components with minimal covariance from a set of bands of the same images. This is used as pre-processing, removing noise from the images before process with

other algorithms as the GMM or K-Means clustering. The results of the Self-Organizing Maps algorithm, realized with MATLAB, are the next topic of the chapter. The following section culminates in an extensive description of three examples, one TensorFlow analysis, implemented with a basic pre-trained SSD, enriched with the Pix2Pix frameworks. U-Net, one of the most used Neural Networks for Remote Sensing, is described with the help of graphs and resulted predicted masks output. The same output is reported also for the SegNet model.

The work concludes with Conclusion and Future Work.

BACKGROUND

In this chapter, the concept of *remote sensing* is briefly introduced, elaborating on the role of the electromagnetic field in the study of satellite imagery. Two main types of sensors, microwave and infrared sensors, are presented. The most popular machine learning techniques used to analyse data from orbital devices are succinctly given. Another critical issue in satellite imagery analysis is image quality enhancement, which is discussed at the end of the chapter. The background culminates with a section describing the Kvarken Region.

2.1 Remote Sensing

Modern technology allows scientists and researchers to inspect lands and oceans from space using remote sensing. Features are detected without any direct interaction with the object itself [2]. The earth and sun are natural emitters of radiation. Therefore, satellite systems are equipped principally with sensors able to read and measure changes in the field.

The electromagnetic field surrounds every object on Earth, travelling at light speed. Studying different frequencies of the waves generated by the electromagnetic spectrum allows observation of different phenomena from space. One principle is that every object reflects, absorbs and transmits radiation. The electromagnetic energy is defined as follows:

$$E = h * c * f,$$

where h is Planck's constant (6.626 * 10-34 Joules-sec), c is a number that expresses the speed of light (3 * 108 m/secs) and f is the wave frequency [2].

Satellite sensors fall into two categories: optical and microwave sensors. Furthermore, the working principle is that optical sensors capture the light reflected by Earth objects and, based on the light intensity, the objects can be identified. However, in dark condi-

tions the resulted outcome is poor, because the natural light during night-time is diffuse and cannot ensure good results. Therefore, modern satellites sense infrared rays from the Earth, guaranteeing optimal results during night-time.

When the sensing phase is terminated, the data collected is composed by a series of dots. Specifically, these dots represent the intensity of the image. Before the dots are combined to engage a visual form, they are converted into binary numbers and stored in a matrix [3].

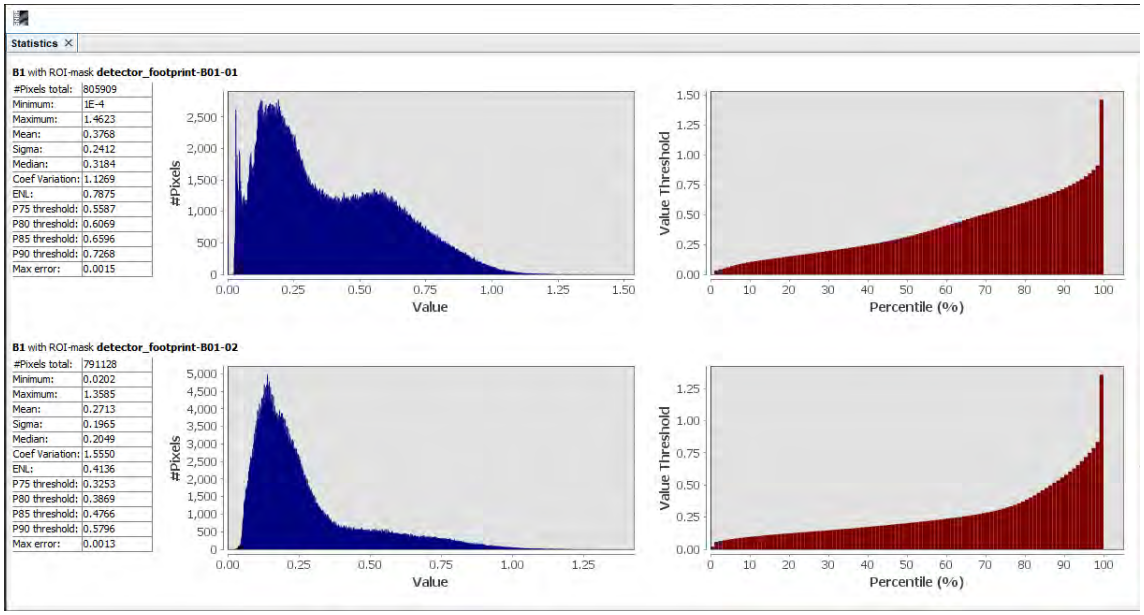


Figure 2.1: Statistical analysis of Sentinel-2 Image. The image shows the frequency of each type of pixels of the image's channel

A typical process of image acquisition is divided into multiple steps. In the first step, the monitoring phase, the vision system is activated in order to estimate the target structure [2]. The data is transferred from the video system to a neural network during a process called acquisition. At this point, the image is constructed joining all similar points collected by the sensors. Moreover, between acquisition and the final output of the visual image, it is possible to use a redundancy system to check if the data has been processed without any problem. It is worth noting that some satellites use sensor calibration and lens distortion alignment in order to achieve the best performance during the process of sensing [4].

Resolution plays an important role in image analysis. Therefore, the image's quality makes the difference between discovering an object from kilometres of distance or less than a metre (spatial resolution). Moreover, there are more factors to take into considera-

tion during the image creation process, for example, the spectral resolution, summarized as the capability of an electromagnetic sensor to capture wavelengths in a range between short-wavelength (about 10^{-14} cycles per second) and long-wavelength (10^{24} cycles per second). The radiometric resolution measures the quality of spectral waves and, with machine learning models, it is also possible to remove noise. Therefore, in modern satellites, sensors are divided into arrays, where every compartment has different characteristics and resolutions. The achievement of a good result is attained operating regular hardware alignment [5]. However, various sensing devices use different scale ratios, computer vision techniques enable the combination of multiple outcomes. Using Gaussian Kernel, for example, the images are intersected eliminating noises deriving from different sensing data or sensors [5].

When satellites sense a particular zone, the data is processed by machine learning algorithms to extract useful information. Moreover, mathematical and statistical algorithms are able to use techniques such as classification and clustering to make a more accurate interpretation of the image. More commonly, clustering is used to group pixels with same characteristics together. In supervised learning, the algorithm needs input labelled data, as input in order to achieve a plausible outcome. Therefore, the quality of training data highly influences the output. Moreover, when the label matches an image's region, the algorithm provides output that is represented by the coordinates' bounding box of the detected object. Along with the bounding boxes, models provide an output number that expresses the accuracy or confidence. In some algorithms, there is a third value, called the mask, that shows the bounding box area.

Machine learning is used also for different types of tasks, for example sensor calibration and maintenance avoiding traditional human supervision. The main reason is to save the huge costs of space missions. Furthermore, working in space is dangerous and also not healthy in the long-term. For example, repair machinery from the space shuttle represents a constraint for the astronauts in the Geo-Stationary Orbits.[4].

The orbit environment creates an appropriate place to collect sensing data from satellites thanks to the diversity of the earth's atmosphere. Moreover, the difference stays in the intensity of background light which in space is almost absent. However, when the light from the stars illuminates our planet, it irradiates the space between the satellite's sensors and the atmosphere, resulting in poor image quality. In some cases, parts of the images could be lost. In order to avoid this problem, perimeters of the instruments are covered with a special foil that reflects back the sun light [4].

The most popular techniques for achieving image clarity are Bagging-based and AdaBoost-based. SVMs, Support Vector Machines, also proved to be efficient in this scope. [6].

The evolution of complexity in orbital devices imagery promotes the use of unsupervised machine learning algorithms. Moreover, K-Means clustering, Fuzzy C-Means (FCM), and Expectation Maximization (EM) algorithms are useful classification models for satellite images [7]. K-Means divides the dataset into groups sharing the same characteristics. Fuzzy C-Means clustering, one of the best algorithms used in remote sensing, is a clustering neural network that produces an optimal number of clusters minimizing different weights in the group. Expectation Maximization finds maximum likelihood parameters where variables are complex. These three algorithms have the ability to distinguish between water, vegetation and land in satellite imagery. In section 4.1, unsupervised learning for satellite imagery is presented. [7].

In particular, the K-Means clustering algorithm is used to study pixels of one input image. The model divides the image into zones with the same characteristics. The resulting outcome is an image where similar zones are shown with the same colours. In the Principal Component Analysis, multiple input images are studied together in order to find uncorrelated patterns. Moreover, the pixels that are constantly repeated are eliminated. The result is a component with maximum variance that is called Principal Component. Self-Organizing Maps detect similarity in the image, removing uncorrelated data. At this point, the pixels with the same number of hits are shown with a particular colour.

To determine the quality of the images, algorithms divide the input into regions assigning to each segment a control point. The same operation is repeated for other images. At this point, every region is compared checking the control point accuracy if it has good quality or not [8]. Therefore, if the difference between two accuracy points in a region is high, due to a lack of scaling or focus in two or more control points of the segment to compare, the process should be repeated. This is used as quality control of the system. However, this technique does not guarantee high-quality output. Machine learning algorithms produce false positives and negatives, distorting the final result.

Stockman proposes an approach by drawing lines between affine control points studying how these intersects between regions. At this point, a parameter that describes the ratio between the line that makes intersection and the one that does not, is used as control point of the image. [8].

This process is divided into four steps: gathering of control points, checking if the control point of the segment corresponds, visitation of the point position in each region, and estimation of the image accuracy. Therefore, a specific dataset for satellite images benefits from using unsupervised learning techniques. Clustering represents a solution to avoid error or bad image labelling. Moreover, calculation of the correspondence between points belonging to the same region of two images can be done using a mathematical approach. However, every situation needs different treatment in order to solve the problem presented [8]. Therefore, unsupervised learning is very powerful and effective because it is able to study the pixel's correlation of the region points with same characteristics.

When a correspondence between regions is distorted, the problem should be solved using different techniques. Furthermore, an analysis of an exclusive point or the correlation between adverse points needs to be conducted. [8]. For this process typically *SVMs* are used. The accuracy calculation between two affine regions is obtained by checking if the two centre points have the same vertex, verifying the resulted accuracy. For example, clustering is used to divide the affine regions of the two images and calculate the variance to check how good the samples are. Stockman uses the best centre region to calculate the threshold for accuracy [8].

2.2 Related work

Scientists and researchers dedicated plenty of resources to find new ways to increase the quality of satellite imagery analysis workflow. Thanks to modern technology, it is possible to study in-depth data from orbital devices. The satellite output is fused with other sensor data to execute a better analysis. Satellite images are often blurry — machine learning methods help in compensating for poor image quality.
Deep learning is a good discipline for the study of satellite images. This is crucial because the deployment cost of new devices is high. Therefore, researchers and data scientists put plenty of effort into finding new ways to improve the state-of-the-art resolution of the current satellite images [9]. Every orbital device has different hardware installed, meaning that it collects data in different frequencies and resolutions. For example, the *Landsat* satellite operates with a scaling between 640 and 670 nanometres. The *Sentinel* satellite uses bands from 634 to 696 nanometres. The sensors installed make use of the non-unique spectral resolution to capture the image. Different sensing instruments are used to obtain various features. For example, *Sentinel* satellites use an array of sensors with about ten spectral bands. These components can spot regions of interest which otherwise would not

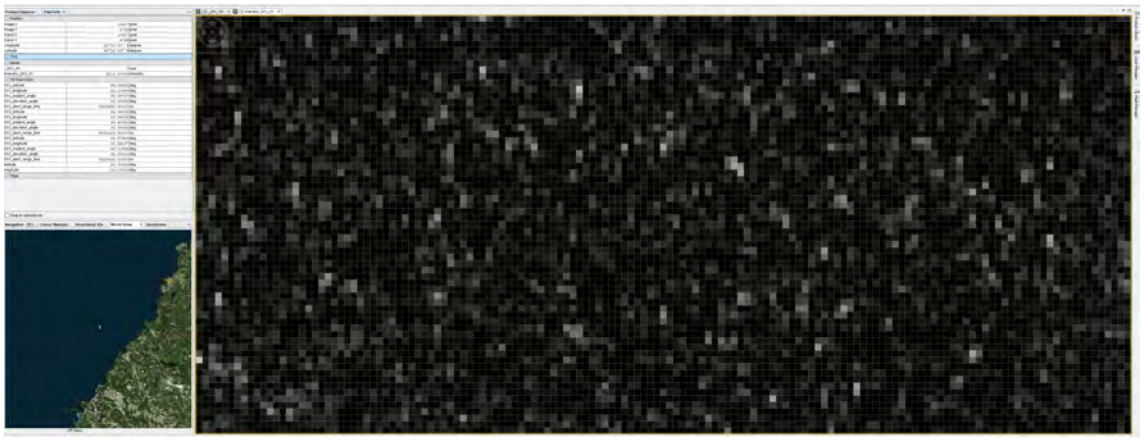


Figure 2.2: Pixels view from Sentinel 2 satellite of the Kvarken Region.

be visible with only one type of device.

However, when the data is not congruent with some parameters of the satellite's hardware, it is stored as garbage collection. Various analyses should be conducted to avoid the problem. Moreover, using only techniques such as resolution resampling that increases the pixels' intensity does not produce acceptable results.

Generative Neural Networks study low-resolution images from satellites with enforcement to improve their quality. Moreover, pixels saturation and clouds need to be considered as noise in image quality [9]. Using filters to eliminate these artefacts is a solution that improves the image quality.

Multiple images are used to enhance the quality of the final one combining numerous low-resolution images into one with a higher resolution through a technique called *multi-image super resolution (MISR)*. The logic behind is the workflow of an algorithm that reorders the pixels of different low-resolution images, reassembling them in the right order and position [10]. However, clouds and brightness changes are artefacts to take into consideration before progressing with the analysis. Temporal resolution, defined as the number of times that the satellite passes the same point capturing the same area, plays an important role. Moreover, the number of low-resolution images acquired in a time range is directly proportional to the final enhanced image. In other words, the more low-resolution images are combined, the better high-resolution images are obtained.

New satellites, such as *PROBA-V*, use software that collects data in the usual satellite resolution, but periodically. One or two times per week, a high-resolution image is ob-

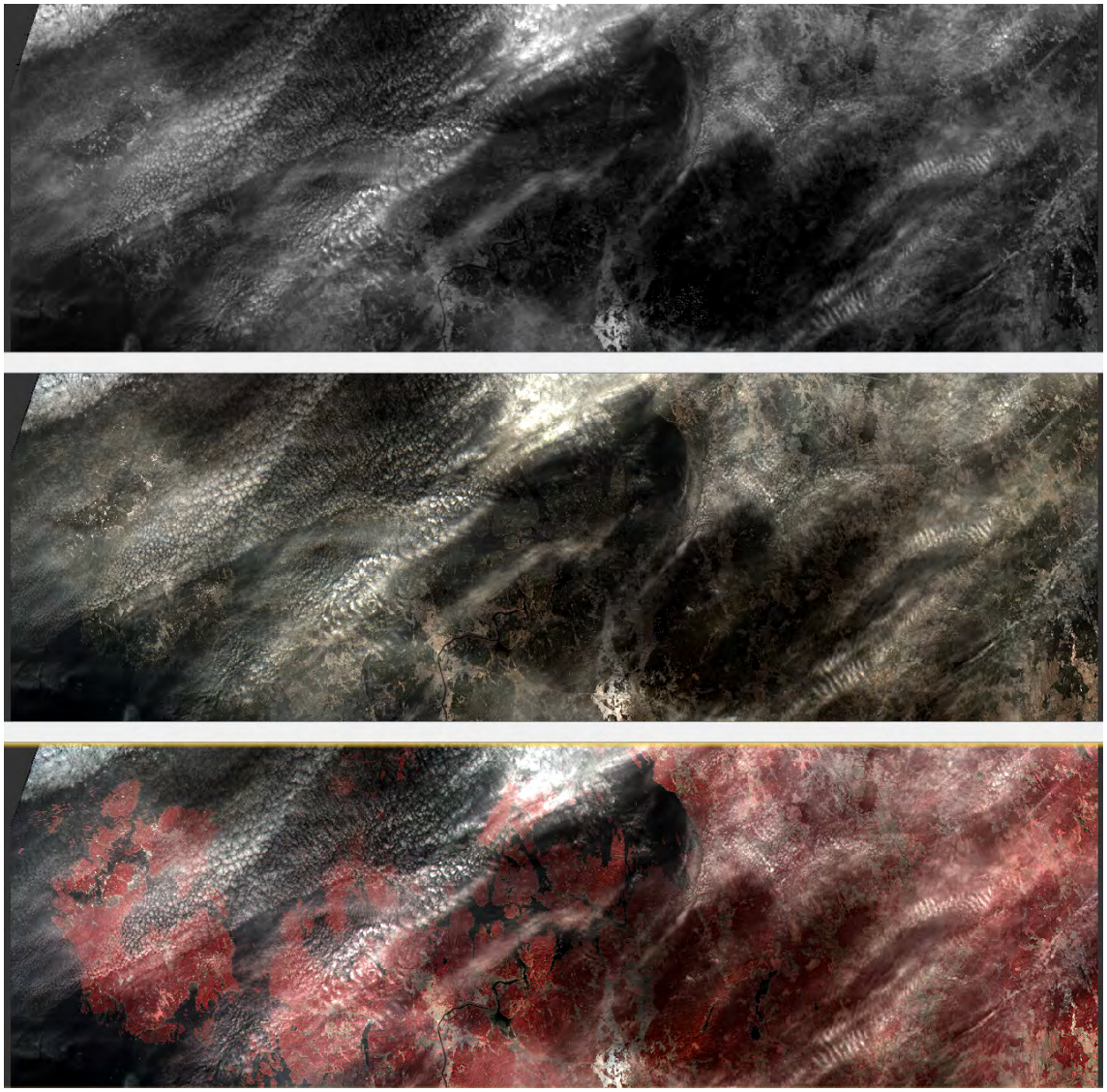


Figure 2.3: This figure shows how the Sentinel 2 image appears when bands are changed.

tained using the MISR technique. A *Convolutional Neural Network* takes as input multiple low-resolution images and a high-resolution one. The resulted output is resampled. Furthermore, a super-sampled image is obtained in three stages. First, the input is resampled to increase the resolution. At this point, the pixels from different images are aligned. Finally, a process of fusion takes place, transforming the original input into a super-sampled high-resolution image [10].

Different frameworks, such as *OpenCV*, the famous library used in computer vision, make use of modern techniques of image enhancement. The combination of neural networks and these frameworks are an essential starting point to arrange better imagery quality.

2.3 The Kvarken Region

The forest coverage of Finland is about 75% of the total land, and it helps the diffusion of oxygen in all the Arctic zones. However, the general rise of temperature could accelerate the fall of plants because the soil does not freeze enough to bear the weight of snow falling on the trees. It is essential to study the forestry situation to prevent damages to the Finnish ecosystem. Moreover, satellite imagery is a beneficial system to analyze the Finnish forestry status. Therefore, ecosystem protection is imperative to preserve the planet and the future.

One interesting location in Finland is Kvarken Archipelago in the province of Ostrobothnia with geographical coordinates of 63.20 degrees for latitude and 21.47 degrees for longitude. The area is mostly uninhabited. Moreover, the population count is about 2500 inhabitants, and it covers five municipalities: Korsholm, Korsnäs, Malax, Vasa, and Vörå [12]. Of the land 94% is privately owned and of the water more than 50% [13]. The Archipelago has unique geological characteristics: the landscape is very particular, and the rise of land is the fastest in the world [13]. This region is rewarded with the title of World Heritage Site. Finnish authority declared the area a National Park in the 1970s [12]. Therefore, the protection under Finnish law guaranteed the conservation of old-growth forests [13]. Due to the yearly elevation of the land, the trees that grow in the new ground are the first ones ever. However, the conservation of this rarity is crucial.

From the sea point of view, the Kvarken Archipelago is called the kingdom of algae. The ecosystem of underwater plants covers most of the archipelago. There are different species of it with various colours. Therefore, deep and machine learning can be helpful to study these plants. A K-Means clustering approach could be able to divide the plants according to their colours. A deep learning system could be useful to reflect to which zones different types of algae are concentrated.

Metsähallitus is the state-owned enterprise in charge of the Kvarken area preservation. Land protection is guaranteed by regulation. Moreover, the intention is to conserve as much as possible of the biological and geological value in the archipelago [13]. They collaborate with the local inhabitants to assure more extensive control over the forest zone and water area. This is a difficult task. Therefore, access to satellite images from Sentinel constellations is essential for the national company to check the zone status. Machine learning promotes the detection state of water areas and tree zones.

Figure 2.4 shows the healthy vegetation status of the Kvarken Region. The image is ob-

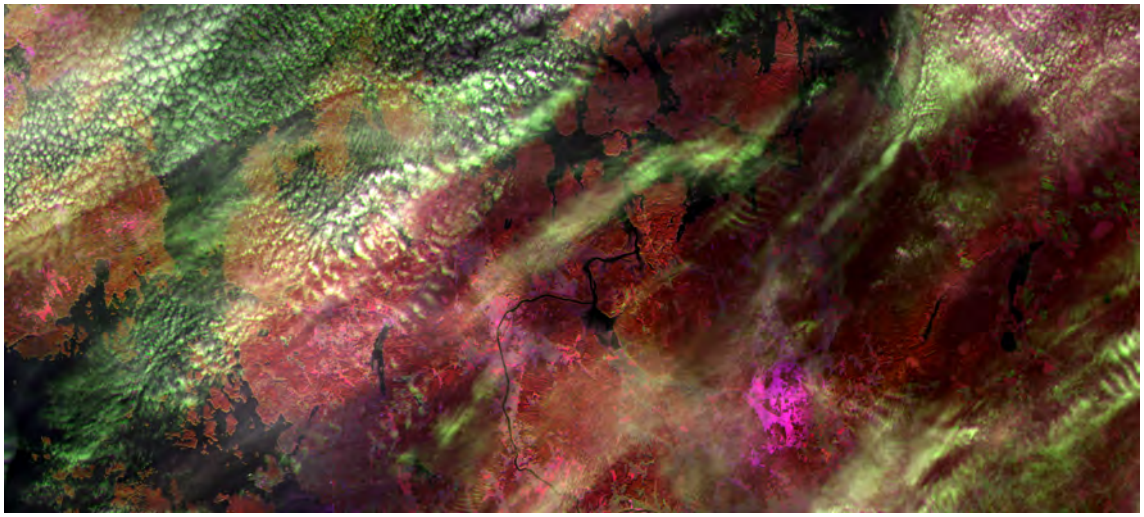


Figure 2.4: This image shows the vegetation status of the Kvarken Region. Trees cover the zones highlighted in red.

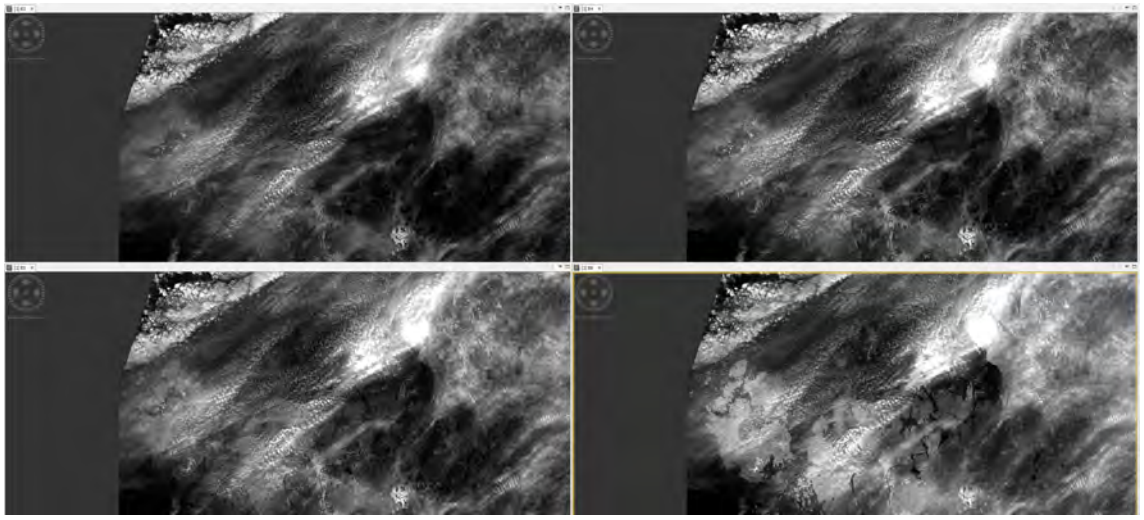


Figure 2.5: This image summarizes the Kvarken Region viewed with different bands.

tained by resampling the raw data from the Sentinel-2 satellite using the *Raster Geometrical* algorithm. The pixels are manipulated to highlight a piece of particular information. In this case, the vegetation is manipulated. However, further algorithms should remove the cloud compound to improve the image.

Figure 2.6 shows differences between the urban areas (upper image) and the vegetation zones (lower image).

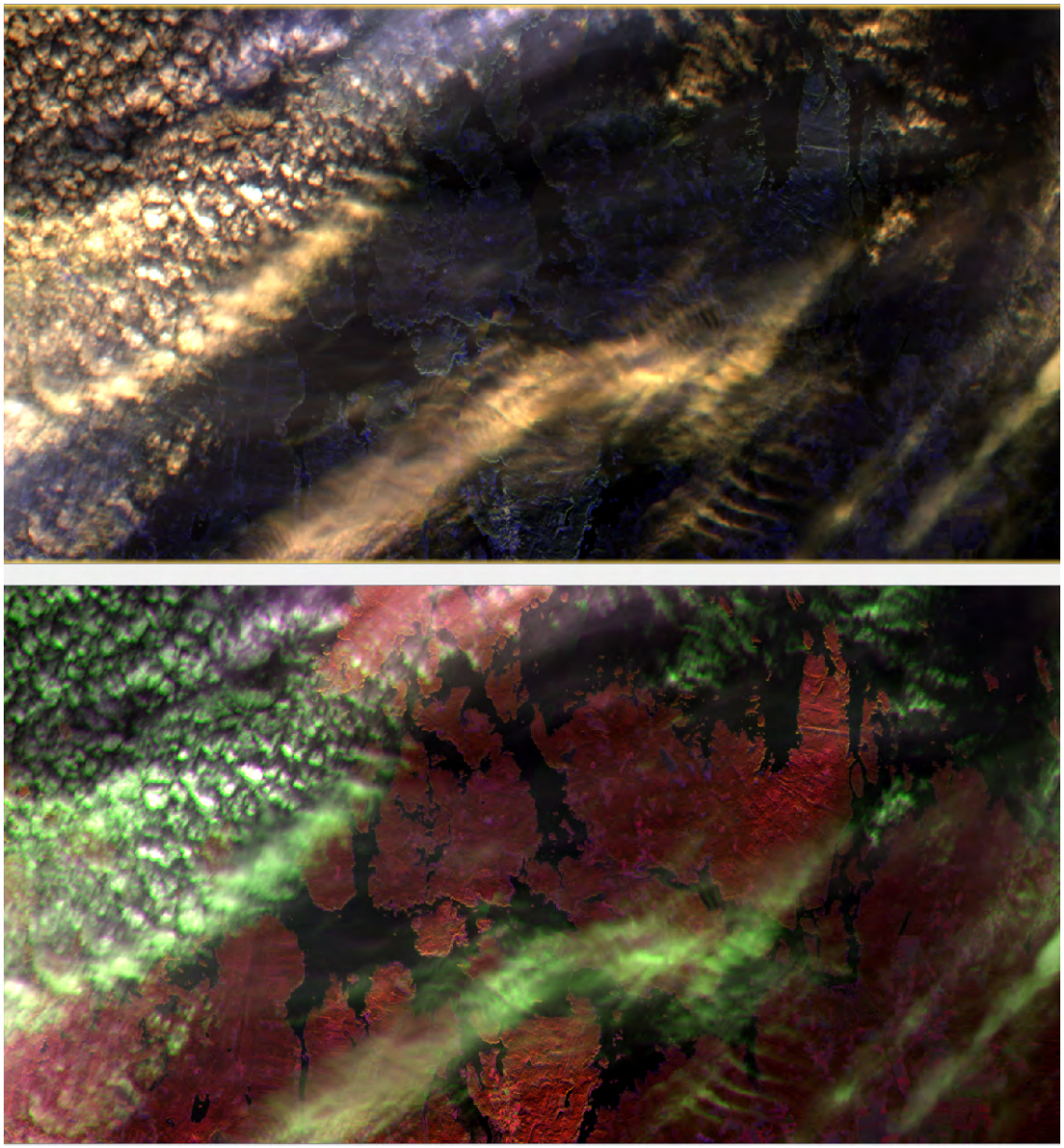


Figure 2.6: Two images from Sentinel-2: The differences between the urban areas (upper image) and the vegetation zones (lower image).

RESOURCES

In this chapter, the resources available for the study of satellite images are analyzed. Google Earth Engine is a useful satellite imagery resource, because it is continuously updated, almost every day. Three alternatives, SNAP, QGIS and ArcGIS, are studied as general software packages. As second part the applicability of these frameworks into the forest field is proposed.

3.1 Google Earth Engine

Google Earth Engine is a platform for remote sensing analysis implemented by Google Inc. It is used in various fields of study, such as urbanization, forestry control, and climate change. The simplistic user interface enhances the usage by technical users or amateurs [14]. New technologies, such as parallel computing and cloud computing, make available computer power that is used to analyze a large quantity of data. Moreover, archives of satellite imagery from the US Government NASA, European ESA, and other national corporations are available to the public for analysis. Unfortunately, merging different types of sensing data is not a simple task, because it involves different frameworks and technologies. Therefore, many researchers are discouraged from using it. However, Google Earth Engine combines different resources and sensing datasets automating algorithms that produce and deploy new information and interactive applications [14].

Google Earth Engine owns a large amount of data organized as a catalogue, sourced from different satellite platforms and sensing systems. Moreover, they are divided by common characteristics. The catalogue includes images from Sentinel 1 and 2, and Landsat. The archive is updated every day. Nearly 6000 images are managed [14]. Historical data is stored, as well. A 40-year collection of Landsat data is available in the Google Earth Engine repository [15]. Each image is associated with information such as location, acquisition time, and which satellite sensed it. The images are tagged with different targets. For example, they can be divided by the type of sensor used to obtain the raw data, or they can be grouped by nation. Therefore, this technique provides a fast filtering capabil-

ity where spatial, temporal, and sensing types are used as filter criteria [14].

The access time to Google Earth Engine resources is reduced thanks to high-performance parallel system service. Algorithms auto-regulate access to the application via authentication servers. The web-based interactive development environment is available ready to use [14]. As an alternative, it is possible to store authentication information into a device and access the server database remotely, with a different *IDE* (Integrated Development Environment) solution. The data exchange between a local computer and Google Earth Engine is done via *REST* (Representational State Transfer) technology. This interface is straightforward and uses an HTML request in order to achieve the data exchange objective. Moreover, it is possible to use Google Earth Engine with *Google-Colab*. It is a web-based IDE where the access to data and information repository is simplified. The analysis is automatically stored on Google Drive.

The architecture of Google Earth Engine fuses various already present Google technologies, such as *Colossus*, *Google File System*, *Google Fusion Tables*, and others. The data is managed by computer clusters able to manage multiple tasks at the same time. About 800 functions are pre-written in order to make the development stage easier [14]. This collection of libraries provides a multitude of pre-made machine learning functions. It contains both supervised and unsupervised algorithms. Operations that require a high computational cost are divided into tiles to take the advantages of parallel computation, contrary to fast operations using virtual machine and just-in-time compiler [14].

Two programming languages provide the interaction between Google Earth Engine API interface: *JavaScript*, a popular client-side programming language powerful for web browser and *Python*, a multi-purpose programming language. Therefore, these procedural coding techniques can personalize any request to the Google Earth Engine. It supports interactive exploration allowing the movement and zoom inside the analyzed image [14]. The script produced by Python or JavaScript is parsed before sending it to the computational server, and the redundant part is removed in order to speed up the calculation process [14].

Google Earth Engine is used across a multitude of disciplines. Forestry change is one of that. Landsat provides a time series of images about forestry. The *Landsat Thematic Mapper (TM)* and *Enhanced Thematic Mapper Plus (ETM+)* are two collections of historical satellite images from the constellation of Landsat that was started in 1988 [16]. Various methodologies find their application in the change monitoring of these imagery

collections over a given time series. Moreover, the woody vegetation lost detection can be derived, making a comparison between the *Foliage Projective Cover* of different years [16].

Classification and Regression Trees enhances the forestry prediction of a given training dataset. For example, it is possible to predict the quantity of lost trees given the historic images series as an input of a particular zone. *Random Forests* algorithm classifies different areas of forests using statistical approaches. However, these techniques produce enough positive results when the training data is large [16]. It is a good practice to enrich the model with a *median filter* in order to reduce the level of false-positives [16]. This is translated into a reduction of the noise produced by the classification process allowing the detection of more explicit samples.

The Global Forest Watch is using Google Earth Engine to create generic machine learning models to predict the future trends of the forest [16]. Having at its disposal a powerful engine like this, the time spent in the calculation is less, and the results are obtained in a shorter time compared to other solutions, such as QGIS or SNAP [14].

3.2 Alternative framework for the analysis of spatial and geographical data

In this section, alternative applications to Google Earth Engine are discussed. In particular, the European Space Agency, in collaboration with software architects, built a framework package to access directly the archives of their constellation of the orbital system. Moreover, QGIS and the new alternative SNAP have been developed with the aim to consult these institutional archives.

Access to Copernicus data occurs via web-browser. The creation of an account is compulsory. Information about the area of interest is retrieved by sending data as perimeter points to the server calling back a series of results correlated to the request. Moreover, there is the possibility to select additional options for precise research. Furthermore, it is possible to use as parameters the amount of clouds and the type of band used by the satellite before interrogating the image archive. However, with Google Earth Engine, one can give different kinds of parameters for a proper specific search. The user interface used by developers and programmers is not recommended for an audience without the associated knowledge.

QGIS and SNAP provide automatic systems to build the images from raw data downloaded from the ESA archive. When using the Google Earth Engine API, the parameters that are used to built the image are more personalized.

QGIS and SNAP offer a machine learning section which can manipulate the data. For example, Google Inc. offers the possibility to write a personalized code to do research. SNAP and QGIS have pre-made algorithms that can be modified only by few parameters. ArcGIS has a toolset that can be used within Python programming language.

However, SNAP and QGIS have interfaces that use pre-trained algorithms. Google Earth Engine and ArcGIS use Python programming language, and they are suitable for more extensive machine learning applications. In the next two subsections, SNAP and QGIS analysis are examined.

3.2.1 SNAP (Sentinel Application Platform)

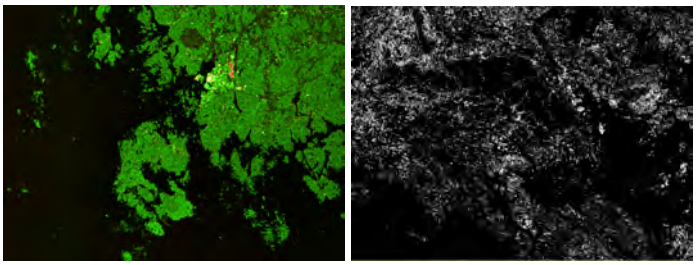
The Sentinel Application Platform, SNAP, is a new software package for the exploration and consulting of Sentinel archive [17]. It is developed as an open-source package by the European Space Agency. It plays an essential role in understanding the evolution of Earth [18]. The archive is public, and it is available at the *Copernicus Open Access Hub*. Moreover, the software can build high-resolution images of vast areas combining raw data from the synthetic aperture radar images *(SAR)* during various days [19].

The framework contains an extensive collection of analysis tools and data display. Images are downloaded in the form of raw data represented by a matrix of points and lines. Therefore, the core *Generic EO data abstraction*, *memory management*, and the *graph processing framework* are responsible for building the final image. Algorithms included in the toolbox cancel the image's noise [17].

SNAP includes machine learning algorithms for supervised and unsupervised classification. Moreover, the *Cloud Exploration Platform Extension (CEP)* decreases the noise produced by clouds in the image [17]. However, to obtain acceptable results, pre-processing of the images is needed. Polarization and correct coordinates are essential elements to control [20].

The *Terrain Observation Progressive Scans* plug-in is used to merge multiple images in order to show more land coverage [18]. Therefore, this plug-in has optimal application in the study of forest areas. However, forestry covers a wide terrain; a system called

Digital Terrain Model package builds a topographic area of the zone that is analyzed by downloading missing data [18]. The tool classifies the presence of flammable vegetation in the forest to prevent disasters or creation of an ignition source. However, the better resolution provided by SNAP does not provide optimal results in all cases [19].



(a) Image 1 (b) Image 2

Figure 3.1: Two different images of the same zone, the Kvarken Region. The images were downloaded from the Sentinel 2 database. 1 represents the RGB image of the Kvarken Region and 2 represents Kvarken Region masked with the tree-cover band

3.2.2 QGIS (Geographical Information System)

QGIS is an open-source software able to retrieve raw data from the archive of the Sentinel constellation of satellites. It transforms the data into a cartographic image. It studies different natural resources, such as water, forest, and nature. The architecture used to build the application is modular.

QGIS is the acronym for *Geographical Information System Environmental Model*, which represents sites and locations. It displays the presence and quality of forestry resources via simple API calls [21]. Moreover, this software helps to solve the conflict of the land's allocation. For example, by consulting the cartographic topology of a zone, it is possible to establish an industrial site far from the forest in order to avoid environmental damages. Forestry companies use the cartography provided by the QGIS system to define which places are suitable to harvest wood, resulting in savings in cost and environment [21].

QGIS finds an application for regional analysis of the land. It collects information about forestry space, that have well-defined boundaries. QGIS offers the opportunity to study the status of the forestry area. However, it is a difficult task to deduce which of the areas are suitable for maintenance to prevent damages. The solution is to pass these images to a deep learning algorithm that detects which part of the forest needs maintenance.

The Geographical Information System can print cartography with a high precision grade.

Machine learning is used in cartographic analysis [21]. Moreover, K-Means clustering is a consistent method to classify and identify regions in the image. However, the study of pixels requires plenty of time. It divides the image into regions that belong to affine characteristics.

The supervised classification should be done by bounding the area within various features, for example, agricultural fields and towns. Other types of analysis can be done to prevent forestry disaster, such as wildfire, since forests prone to fire appear in different colours. However, some areas are simple to classify. For example, the boundaries of agricultural fields are easy to detect, and they are a good starting point to improve the performance of the classification.

Multiple factors determine the state of a forestry area. For example, temperature, soil type, and closeness to agricultural sites or irrigation system are some of these facts [21]. In order to obtain acceptable results, these factors should be taken into consideration. Moreover, to make a fair comparison of various image features, unsupervised algorithms need to be weighted [21]. The usage of *mean* and the *correlation* is an excellent solution to adapt the algorithm to different types of situations. Alternatively, the QGIS software uses a module called *STRETCH*. The users sets multiple parameters based on what they want to search. Consequently, the system decides the suitable number of clusters to study the images [21].

A *pairwise matrix comparison* is another method used to calculate the weight for an algorithm in both unsupervised and supervised learning. The matrix can be used in the deep learning frameworks, such as TensorFlow. Moreover, this process enables the algorithm to be retrained with specific data to assess forestry status.

The QGIS software contains a framework called *IDRISI* that classifies the amount of water inside a map, and a module named *WEIGHT* which is useful to automatize the process of weighting the algorithm [2].

In the forestry analysis, good results are obtained with the multiplication between the general image's weights and the forest's portion of that image. This process can be automatized using the QGIS module called *MCE*, multi-criteria evaluation [21].

QGIS contains *CROSSTAB*, a plug-in that identifies the correlation between forestry and other types of land, such as urban zones or agricultural areas. Moreover, this relationship

shows how pollution impacts the forest's health [21]. However, it is not very easy to verify the correctness of the results.

QGIS is used to gain a complete view of the forestry status. The open-source software captures and isolates natural resources from the images. The outcome data can be used as input for clustering algorithms or in supervised learning [21].

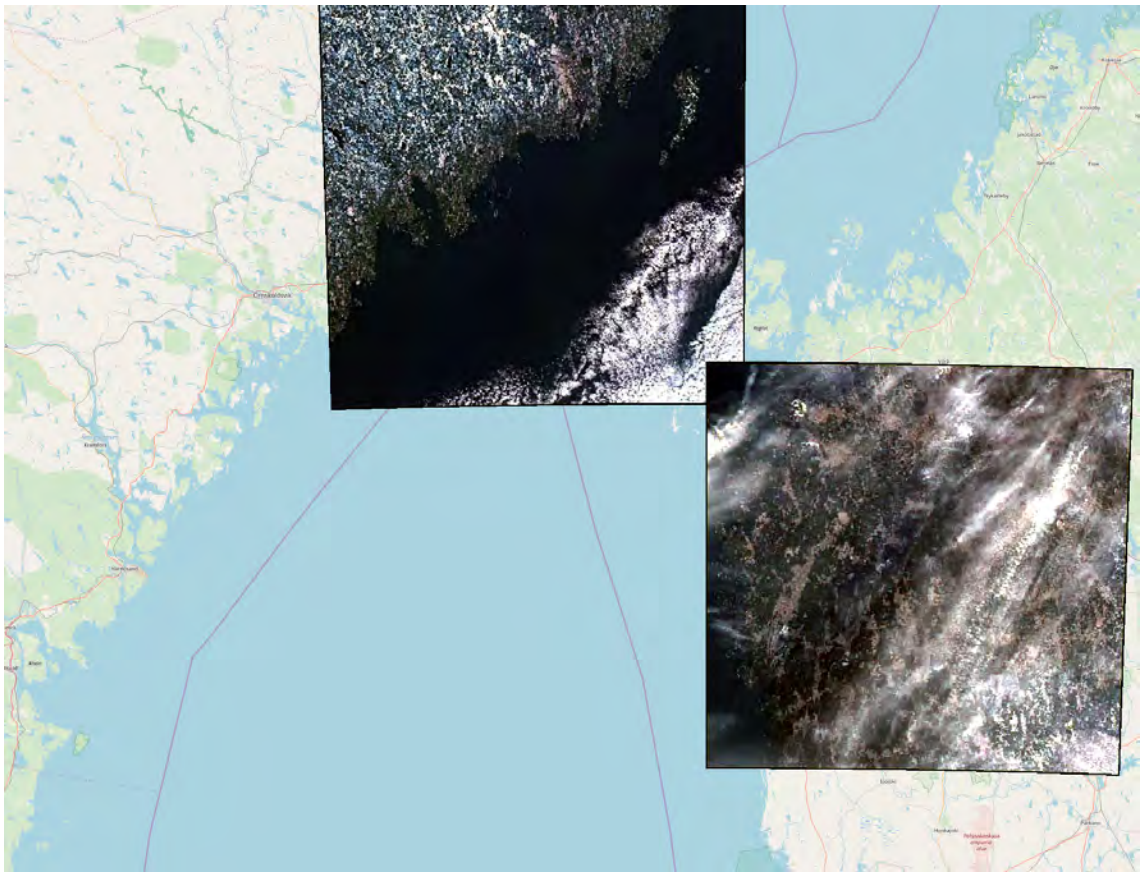


Figure 3.2: Example of QGIS view. The image shows the Swedish and Finnish side of the region.

3.3 ArcGIS Desktop

ArcGIS is a Geographic Information System developed by Environmental System Research Institute, Inc. [45]. The company holds about 40% of the global GIS market. The application engine, *ArcView*, uses a system called the *shapefile* that converts data that contains all the necessary features to conduct analysis. The *shapefile* is similar to the Google Earth Engine's *shapefile* that contains points, lines, and polygons. These components are stored in the database of the system. Moreover, information such as geographic coordinates, land topology, are stored in the geo-database of the application.

These storage units are grouped in a town unit or a nation unit.

The application was developed in 2001 [45]. The ArcView engine is the core that provides instructions to the modern *ArcGIS Desktop*, providing an environment for the analysis, to elaborate and to display geographical data. Moreover, the package also contains a set of toolboxes and routines that manipulate the geographical data. It runs deep learning models, as well. Another essential feature is the direct connection between ArcGIS and Python programming language which provides the chance to develop personalized scripts in order to create a reliable analysis. A set of functions is also available in the Python sub-system to allow better interaction with ArcGIS.

The images provided by ArcGIS Desktop are of high quality. Compared to other systems, like Google Earth Engine, QGIS and SNAP, the satellite images from ArcGIS are better, with a considerable zoom rate. In the other systems it is not possible to distinguish deforested areas, which leads to a laborious process of labelling. Instead, with ArcGIS, the quality is high, and it is possible to obtain zoomed images with a resolution of almost 10 thousand pixels in width and height. However, compared to the other platforms, there is no temporal data.

Moreover, it is not very easy to understand when the images were recorded. Unfortunately, the temporal data is one of the key aspects of the damage assessment. The process of obtaining images after a storm is crucial to quantify damages.

There are multiple datasets tiled inside the ArcGIS software. When one of them is selected, it overlays over the main map creating a new layer, such as the example in Figure 3.3. ESRI also offers a server where new data features can be downloaded from. In particular, there are different types of servers available in the application. Moreover, a map service [45] renders new layers as an overlay, and a feature service [45] makes new traits available to the user as enrichment of the original map. One of the most interesting services offered by ArcGIS is the imagery service, enabling access to a large amount of satellite data from Landsat and Sentinel orbital devices. ESRI also shares from their servers a system called *geo-processing unit* [45], a set of functions and packages to compute sophisticated analysis.

The process of labelling the ArcGIS dataset, that is used for most analyses in this work, has been conducted by verifying the deforested areas by overlapping the original map with the tree-lost overlay layer taken from the ESRI database. The overlayers are very helpful because they delimit the tree loss in forestry of the Kvarken Region, making the

(a) The original map layer

(b) The labelled map layer with tree loss overlay

Figure 3.3: This image shows how ArcGIS overlaps multiple layers in order to highlight the deforested areas.

label creation easier.

METHODS

Due to an increased effort to boost computational resources, *AI* has found success in recent years. The processing of large quantities of information has become crucial for the success of companies and institutions. Today, the development of this field has made possible an increase in performance in machine automation [22], cancer classification [23], face recognition [24], autonomous cars [25] and thousands of other operations. Machines can structure and solve problems that are difficult for humans [26]. However, it was only a few years ago that computers started to perform speech recognition and object detection or tracking. Computers understand statements only if behind them there is a logical inference rule. This principle is known as the basic rule of *Artificial Intelligence* [26]. Behind machines there are humans who implement *logical inference* rules to guide them. In the past, these "rules" were *hard-coded* inside models. Today, thanks to machine learning algorithms, these statements can be acquired by the machine itself. However, the data used to train networks needs to be clear and without noise. Another problem that machine learning tackles is called *representation* of the results. The machine needs to find a way to solve the problem, but also to represent it in a manner that humans can understand [26].

Machine learning introduces new technologies, such as real-time object detection systems, natural language processing and computer assistance. Moreover, modern search engines, such as chatbot assistants, are developed using these models. Algorithms can also be applied to assist software developers in *auto-generating* code and making it more reliable. [27]. The organization behind the input data often greatly influences the outcome and the algorithm performance. The information used for the learning process should represent the specificity of the problem. This process is called *feature finding*. Contrarily, the models can produce non-accurate information due to the generality of the data. Therefore, one of the key points in developing machine learning models is dataset design. The outcome of the learning process has to be comprehensible and simple.

A significant problem of machine learning is the variance of factors in the data [26]. An object of a particular colour can be seen as black during night-time. A shape of an

object depends on the point of view that the image offers. Moreover, for a computer, it is not straightforward to understand different light variations because the data analysis is performed on the pixel level. The learning processes of most neural network algorithms manage massive amounts of data, and they require large computational resources. However, if the data is badly trained, the neural network produces unexpected results with poor learning ratio. For example, the portion of the dataset that is incomplete or unclear should be eliminated. However, when the data is generic, the neural network is not able to be trained correctly, producing *overfitting*. On the contrary, if the model is simpler *underfitting* occurs. However, the data is also regularized in order to make it more consistent. The model validity is checked with a simple methodology. The dataset is divided into two parts: the training set, used to feed the model, and test set, used against the first one to check the validity of the model. One way to quantify if the model has produced acceptable results is the measurement of *RMSE*, Root Mean Square Error calculated as follow. Given the hypothesis h:

$$RMSE(x,h) = \sqrt[2]{\frac{1}{m}\sum_{i=0}^{m}(h(x^i)-y^i)^2}, \tag{4.1}$$

where m is the number of dataset instances, x_i represents the features and y_i is the labelled data [28]. Another example of performance measurement is the Mean Absolute Error (*MAE*):

$$MAE(x,h) = \frac{1}{m}\sum_{i=1}^{m}\left|h(x^i)-y^i\right|. \tag{4.2}$$

In Deep Learning, the basic building blocks are combinations of diverse components. Their primitives are called *layers*. Every layer is composed of a series of neurons, and these layers are interconnected between each other. Different kinds of layers are described in the following paragraphs.

In a *Fully Connected Neural Network*, all input components are inter-connected to the output. Each neuron is in charge of a part of the analysis. When a process is completed, the outcome is merged and sent to the output node [27].

The *Convolutional Neural Network* is specialized in image analysis. Moreover, the input is divided into slices and sent to the network. From a mathematical point of view, they are *matrices* filled with pixels. The neural network studies the relationship between these pixels. Good results are achieved when the dependence among the pixels is ex-

ploited [27].

Table 4.1: Summary of different neural network layers

Neural Network Layer	Symbol	Speciality
Fully Connected Layer	FCL	General purpose
Convolutional Layer	CL	Image processing
Dropout Layer	DL	Increase accuracy
Zero padding Layer	ZPL	Add 0 in the image contours

4.1 UNSUPERVISED LEARNING IN SATELLITE IMAGERY ANALYSIS

Unsupervised learning has an essential place in the machine learning ecosystem. These algorithms do not need labelled data for the learning process. Moreover, in the following sections data filtering by variance, covariance and other methods are explained. The quality of the model is measured through the *error rate* produced by the algorithm itself. However, a model is beneficial when it works well and produces acceptable results based on data that is different from the one used for training or testing. When algorithms become complex, intermediate data is used to evaluate the overall achieved performance.

Unsupervised learning models need to run without *pre-existing labels*. Usually, in machine learning, when a parameter changes during different training sessions, the algorithm produces spurious results. It is time-consuming to adjust the parameters of the model in order to obtain acceptable results. Furthermore, modern machine learning models can auto-understand if a rule has changed and act accordingly to preserve the final results. However, unsupervised learning models are useful for discovering patterns and similarities in datasets. Furthermore, unsupervised learning is beneficial when the ground-truth labels are unavailable. On the contrary to supervised, where the output is limited only to the labels provided by the user, the unsupervised model checks for solutions in all the possible features discovered [29].

For example, in the case of satellite imagery study, if an extensive dataset is provided, the model produces a satisfactory outcome. Contrary, if the input data is weak, the results are inaccurate. In the case of supervised learning, performance is highly dependent on the quality of the training dataset [29].

Unsupervised methods such as *Clustering*, *Self-Organizing Maps*, or *Principal Component Analysis* may perform well when the input variables contain features that continuously change. Moreover, these models are called *reducers* because they are able to classify the data with similar characteristics together. The main rule is to look into the variance of clustering, or covariance in the PCA Analysis.

K-Means Clustering studies the change of the data, distributing it into groups through affinity and similar characteristics. In other words, it can divide the image in different regions, based on several colours' tonality. However, unsupervised learning does not label or identify objects, it infers optimal results by checking the similarity between the image's areas. Furthermore, these algorithm families are able to divide unclassified images into groups [29] with similar patterns that can be used as dataset labels for supervised learning models.

Data scientists use unsupervised models to make a preliminary study in order to find a relationship between unlabelled data points. Eventually, this outcome is used to label data for supervised models. This is possible by using cluster analysis to group data with the same characteristics, giving to each cluster a label. Therefore, similar data points are assigned the same label group [29].

Overfitting is a phenomenon where the algorithms extract too many features from the dataset without taking out the noise, generating poor results. To overcome the problem, the dataset needs a regularization. One way is the usage of unsupervised learning to reduce the complexity of the model, for example, with a process of noise cancellation [29]. This process is called *feature extraction*, where the algorithm reduces the data, capturing only the interesting features. Moreover, from these extracted parts, it is possible to isolate the most promising ones.

Unsupervised learning is effective in reducing the problem of *curse dimensionality*. This phenomenon occurs when a large dataset is used to feed a supervised learning algorithm. The result is a poor solution to the problem. The unsupervised model is able to extract the most important features from a huge dataset, reducing time spent in computation, and improving the final result of the calculation. That is why a vast number of data scientists, with the help of feature engineering, preprocess data with unsupervised learning algorithms in order to extrapolate the most salient features automatically from a generic dataset. Autonomous models are effective when the data used to train the model statisti-

cally differs from the prediction data [29]. The subset of unsupervised learning models proposed is called *Dimensionality reduction*, which aims to filter the unnecessary features and keeping the interesting ones. One approach is the Principal Component Analysis, where the algorithm is able to identify similarity in the features. Moreover, looking at the variance of the patterns, the algorithm reduces the dimension of the data. In the forestry field, it is possible to use *PCA* analysis to extract the *principal components*. In this case, they are the most occurrent features, from the various bands of the images sets. Furthermore, the final set of images is built with a single principal component. One can run a clustering analysis with only the principal features to achieve better results.

Clustering is a robust unsupervised learning algorithm that groups similar patterns in a dataset. There are several variations, but one of the most promising ones is *K-Means* clustering. It takes the desired number of clusters as an initial parameter. Every instance of the dataset is analyzed. The algorithm calculates the *Euclidean distances* between data points. The algorithm regroups the patterns in partitions with the smallest possible variation [29]. This operation is achieved without the help of labels, and the model can make a comparison of different samples by itself. However, the algorithm picks the starting point of the analysis randomly. Clustering applications are various. For example, they are useful in preventing the bank from fraudulent activities, or in the installation of spam filters. For the satellite imagery, clustering occupies an important role. The algorithm is able to segment images from an orbital device. At this point, the outcome of the analysis is regions, separated into different clusters. For example, one region is the water, another the forest, and so on. Using these auto-encoded labels, it is possible to feed supervised models, for instance a neural network *segmenter* such as *U-Net*.

Autoencoders serve as the new frontier for neural networks, and they are able to generate features using a technique called *recurrent learning*. The detail of this particular network is that it is divided into two sections: the *encoder* and the *decoder*, which have the same number of layers [29]. Every layer of the model learns a new feature from the previous one, and it tries to combine the patterns, studying the variance and covariance. The outcome of the *autoencoder* is usually used as input for numerous supervised learning algorithms. Another modern technique is called unsupervised deep learning, where the layers of the network try to create a representation of the problem [29]. To keep the features together, this neural network tracks the weights of each iteration in a gradient function [29].

4.1.1 K-means clustering

The K-Means Clustering divides a set of observations into partitions. Every member of a single group has similar characteristics, but groups are different from one another. Usually, the number of clusters is a parameter of the model. It is essential to feed the algorithm with an initial parameter: the *number of clusters*, which delineates different regions in the image. K-Means Clustering minimizes the variance in a cluster [29]. In the case of the satellite image study, the analysis is concentrated on the differences between the spectral bands. It is important to say that the algorithm can assign parts of the image into clusters but, the identification of the region such as type of plants or field recognition, needs human assessment. The outcome of the K-Means analysis could be assigned as ground data for further supervised analysis. It is good practice to collect multiple results from the K-Means analysis, and assign to each cluster a label in order to create a large dataset.

However, K-Means suffers from the same weakness as the other members of the clustering algorithms family. Due to the random initialization, the results of two different sessions of the system manifest a few variations in the outcome. However, it is possible to fix it assigning a seed into the main script of the program [29]. In satellite imagery, K-Means introduces some problems. The spectral properties of the image can often vary due to the brightness of the sun. Linear objects like power lines and roads are hard to distinguish when the resolution of the image is not perfect. Human effort is required in order to identify the labels of the clustering results [3].

K-Means Clustering makes partitions of a dataset into M disjoint clusters and $C_1, ..., C_M$. The K-Means Clustering Error criterion is summarized as follows:

$$E\left(m_1, m_2, ..., m_M\right) = \sum_{i=1}^{N} \sum_{k=1}^{M} I\left(x_i \in C_k\right) \left\|x_i - m_k\right\|^2,$$

where x_i is a part of the dataset:

$$Dataset = \left\{x_1, x_2, ..., x_N\right\},$$

and m_k is the centroid of the subset c_k, which contains x_i.

$$I\left(P\right) = \begin{cases} 1 & for \quad P = true \\ 0 & otherwise \end{cases}$$

The Euclidean distances is calculated with the following equation:

$$\|x_i - m_k\|^2 .$$

4.1.2 PCA analysis

Principal Component Analysis is a statistical method part of the *linear reduction* algorithms family, aiming to single out *uncorrelated* data points. The method is able to discover the maximum variance in the data. The Principal Component [29] has the most significant variance. With this value, it is possible to regenerate the original data with only the salient features. The outcome of the analysis is smaller than the original dataset. PCA is able to select features that for some other algorithms stay hidden. The process of dimensionality reduction facilitates an increase in computational efficiency for further analysis.

The dimensionality reduction algorithms are essential in terms of clustering because they increase the accuracy of the final result, eliminating unnecessary artefact [3]. When it is not possible to choose a high-quality dataset, running the PCA analysis is a good starting point because it eliminates unnecessary data making the model effective. PCA achieves good results in the detection of edges and corners. In general, it studies the differences between various pixels [3]. In the case of forestry analysis, it is a good idea to use multiple images from the same area in order to check the differences in the trends of the Principal Components.

However, other various methodologies study differences in the satellite images. For example, the *Harris corner detection* is used to distinguish between the different types of terrains [3]. The *Independent Component Analysis* is another model that can segment multispectral images where the pixels are very uncorrelated with one another [3].

In the case of the satellite imagery analysis, it is necessary to maximize the information contained in each image, removing the non-necessary data, for example, the noise. As said previously, a satellite image is composed of the aggregation of multiple bands that are correlated with each other [31]. An image is summarized by a matrix as follows:

$$IMAGE_{b,n} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{8,1} & \cdots & x_{8,n} \end{bmatrix}$$

where n is the number of pixels and b the band number. Every band from the *IMAGE* matrix can be represented by a vector v:

$$B_k = \begin{bmatrix} x_1 \\ \vdots \\ x_8 \end{bmatrix}$$

where k is the band number. The uncorrelated components are calculated using the covariance $\sigma_{b,j}$ between bands in the following equation:

$$\sigma_{b,j} = \frac{1}{N-1} \sum_{p=1}^{N} \left(PN_{p,b} - \overline{m}_b\right) * \left(PN_{p,j} - \overline{m}_j\right),$$

where $PN_{p,b}$ is the pixel that belongs to the band b and $PN_{p,j}$ is the pixel that belongs to the band j:

$$PN_{p,b} \in band_b \quad and \quad PN_{p,j} \in band_j,$$

$\overline{m}$ is the mean calculated as follows:

$$\overline{m} = \frac{1}{n} \sum_{i=1}^{n} x_i,$$

the covariance σ is grouped in a matrix:

$$C_b = \begin{bmatrix} \sigma_{1,1} & \cdots & \sigma_{1,8} \\ \vdots & \ddots & \vdots \\ \sigma_{8,1} & \cdots & \sigma_{8,8} \end{bmatrix}$$

The eigenvalues ev, defined as the container of the original information of certain bands [31], are calculated as the determinant of the resulting characteristic equation:

$$det\left(C - evl\right) = 0 \rightarrow ev = \frac{det(c)}{det(I)},$$

furthermore, the Principal component vector is expressed by:

$$PCA_8 = \begin{bmatrix} PCA_1 \\ \vdots \\ PCA_8 \end{bmatrix} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,8} \\ \vdots & \ddots & \vdots \\ w_{8,1} & \cdots & w_{8,8} \end{bmatrix} * \begin{bmatrix} x_1 \\ \vdots \\ x_8 \end{bmatrix}$$

from the matrix PCA_8 one can calculate the eigenvector w solving the following equation [31]:

$$(C - ev_k I) * w_k = 0$$

4.1.3 Self-organizing maps (SOMs)

Teuvo Kohonen is the inventor of the *Kohonen's map*, or Self-Organizing Maps neural network, that was introduced in 1980. It is able to structure the representation of essential features in an input image. This unsupervised model detects features by removing the noise part of the data [32]. It plays a vital role in satellite imagery for damage assessment because it discovers differences between the image's regions without losing time in the fine-tuning of the model. The algorithm classifies the objects' positions and the differences between the various areas of the map. The algorithm outcome is a series of data that shows how every pixel of the input image is correlated within other inputs.

The neural network has a unique characteristic called *dynamic weight*. During a session of training, the model can adjust the neuron's weight [33]. Self-organizing Maps have a simpler architecture concerning other neural networks. It uses only two layers, the *input layer* and the *output layer*. The working principle is *competitive learning* [33]. During the process, there is a competition where every neuron competes with others in discovering different patterns, acting as separate decoder for the input [32]. From a mathematical point of view, the *Vector Quantization* is the most suitable method to achieve results with Self-Organizing Maps [32]. Neurons are grouped based on the best match. Each iteration activates only one node, which is selected based on the similarity between the input and the rest of the nodes. All input vectors have the same element number. The working principle is that the model computes the Euclidean distance between the input vectors and all the nodes in the input layer; the one with the lowest weight is the winner. However, the neurons member of the *SOMs* has to communicate one with the others. Otherwise, the competitive learning fails. In order to solve this problem, the components of the neural network are grouped in logical subsets where similar weights are imposed [32]. The vector weight propagates the values to all neurons, and it receives updates every stage of the learning process. At this point, the algorithm calculates the distance between each victor neuron. At this point, another cycle of neighbour calculation starts, and the neuron with minimal distance value is the winner.

The SOMs algorithm takes the input x at a time t, returning a winning node c at the same time of the input t that is selected using the following formula [34]:

$$c(t) = arg \quad min_i \left(\|x(t) - w_i(t)\|_2 \right),$$

where $\|x(t) - w_i(t)\|_2$ is the the Euclidean distance between the input x and the weights vector at the time t. *Argmin* is the argument of the minimum. The weights vectors are

continuously updated using the following formula [34]:

$$w_i(t+1) = w_i(t) + \Delta w_i(t),$$

where $\Delta w_i(t)$ is deduced by the following equation:

$$\Delta w_i(t) = \alpha(t)\, h_{c,i}(t) * [x_i(t) - w_i(t)],$$

$h_{c,i}$ is the scaling function of the winning node c. α is the learning rate at the time t. It decreases according to the neighbourhood size until the fitting into the network becomes impossible [34].

4.2 DEEP LEARNING

The first hints of deep learning occurred back in the 1940s. The concept might seem new because during various decades the discipline's name changed more than one time [26].

Table 4.2: Summary of different names of deep learning during the years

Cybernetics	1940s - 1960s
Connectionism	1980s - 1990s
Deep learning	from 2006

Adaptive Linear Element was one the first neural networks, written as a code in the 1960s by Widrow and Hoff. The task assigned was simple: prediction of numbers from a dataset using the stochastic gradient descent. The first recall of image recognition is from Fukushima in 1980, where connected layers and neurons were used to study pixels [26].

Deep learning is a subset of machine learning family, and it enables multi-steps extraction of features from the input data using complex functions in order to outcome results.

Parallel programming techniques and computational resources achieve more ample and complex operations that can be deployed efficiently. Moreover, the deeper the network is, the more complex operations are executed. For example, a *perceptron network*, a neural network with only one layer, detects only a few features inside the data. A more in-depth system is able to discover extra features from the input data.

The damage assessment of forest area can be studied through historical sequences of satellite data. It is possible to underline the differences in the forest regions by looking into the data from different days. For example, the orbital devices can notice some gaps

in the coverage of the trees in certain zones, and with other series of satellite images or within the usage of different machines such as drones, quantify the damage impact.

A classical deep learning model divides the analysis of a satellite image into sub-tasks. The network input-layer maps the image. At this point, it is sent to hidden layers in charge of discovering features. Moreover, in the case of forestry assessment, neurons from the layer study the forest, comparing differences in brightness of adjacent pixels [26], such as corners, and contours, discovering the forestry perimeter. In the last layer, neurons combine the outcome provided by the upper-layers, collecting the different regions found. Figure 4.1 shows a schematic view of a typical neural network. However, clouds represent a problem. When some regions are covered by them, the deep learning model, if it is well trained, can understand which part of the image is clean.

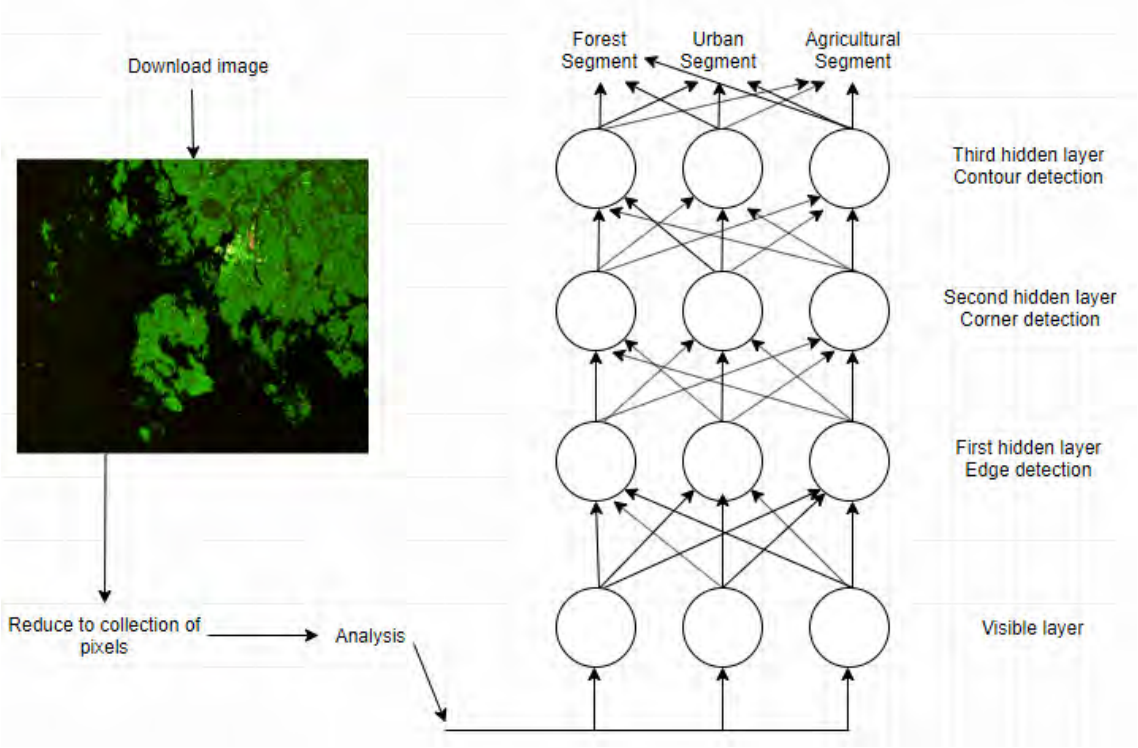


Figure 4.1: A schematic representation of a dense neural network.

Deep learning algorithm layers can be combined to produce accurate results. The primitive components of the network, neurons, are connected, and they communicate with each other to exchange information. Generally, by combining more neural network components the performance of the model grows, increasing the algorithm's deepness.

There are plenty of neural network architectures available to researchers and developers. Some of them are shown in the few following paragraphs.

LeNet is one of the first architectures developed in the 1988s. However, the power required to accomplish a task was higher than the one available in the 80s. The *GPU, (Graphic Processing Unit)*, development made the calculation achievable for almost all computers. They were able to parallelize computations, saving time.

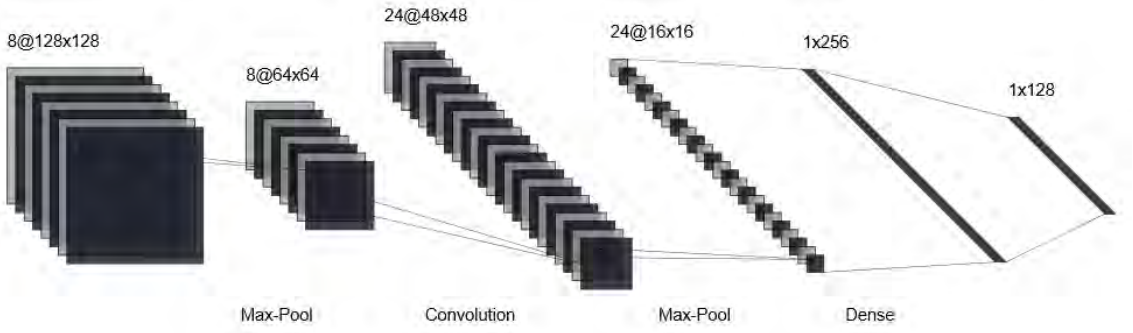


Figure 4.2: Simplified schema of LeNet network

In the 2010s *AlexNet* was developed [27]. This neural network performed great outcome in object recognition. However, a problem was found. When the network layers grew deeper, the results produced by the algorithms were not precise due to a gradient reduction problem. To avoid this issue, in 2015, *ResNet* was developed, having higher object recognition capabilities than other networks available at that time [27].

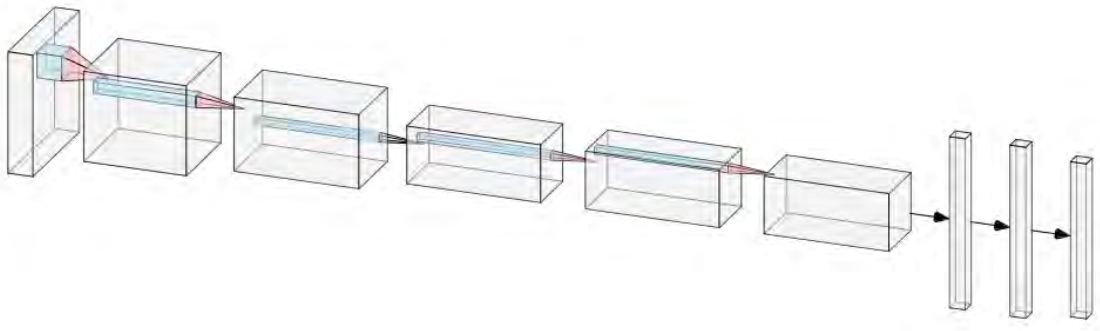


Figure 4.3: Simplified schema of AlexNet network

Google Inc. has developed a neural machine translator, used to translate documents. It understands the sentence's meaning. It helps in the achievement of better work. *One-shot learning model* architecture was used. It is another excellent example of how new technologies learn. This one, in particular, uses the same principles as humans during the learning period [27].

The *Generative Adversarial Network* is based on two different neural networks where

one competes with the other. It achieves good results in new image generation. The first architecture, called the *generator*, generates the image. The second one, called the *discriminator*, takes the real one, and it checks if there are some differences between the real figure and the generated one [27].

One example of deep learning usage is *the Deep Blue* chess system, developed by IBM. The neural network was designed to win a chess match against Kasparov, one of the best chess players of all time. Furthermore, Deep Blue was able to understand all the possible combinations of moves of the human player, consequently, it was able to solve the problem faster [26].

There are many software packages available online for deep learning. *TensorFlow*, for instance, it uses the mathematical concepts of *tensors* to build a complex neural network. However, this framework and the other packages are slow to operate. The activation and expansion of the neural network require plenty of time.

Dataset plays an essential role in the development of the neural network. Thanks to the digitalization era, nowadays a massive quantity of information can be used to build new and specialized datasets. The study of large data requires computational power. Moreover, packages such as TensorFlow, use the power of GPU and CPU combined to reduce the time spent during the calculation.

In conclusion, new forms of learning such as reinforcement learning, are engaging scientists, researchers and programmers. This unique learning system uses trial and error to understand the rules of learning route without any external intervention [26].

4.2.1 Deep convolutional neural network

The primary component of the neural network is neurons. They are organized in layers exchanging information between them when certain conditions are verified. The most common working principle is based on the *Perceptron* model:

$$f(x) = \begin{cases} 1 & wx + b > 0 \\ 0 & in \quad other \quad cases \end{cases}$$

where x is a vector and w is the weights' vectors and b is bias. The output is divided into two parts: 1 if the condition is verified and contrary 0 [35].

A typical neural network is composed of multiple parts. The *input layer* is in charge

of taking the input for the whole network. The *hidden layers* discovers relationships in the data by finding common patterns in it. The numbers of neurons in a layer varies. For example, in a Perceptron network, they are only two, but in complex models, such as *VGG16*, there are multiple of it. They are called *Deep Neural Networks*. The *output layer* is responsible for presenting the results. Usually, the number of neurons differs by the type of operation that the neural network has to do. For example, in *binary classification*, the output layer is just a neuron; in other cases, such as the *multi labels classification*, the number is proportional to the expected output [36]. The following model gives the total number of layers:

$$DCCN \quad layers = \quad hidden \quad layers \quad + \quad output \quad layer.$$

The input layer is not considered when counting the dimension of the network. The working principle of a basic neural network is summarized as follows: the input layer prepares the image to analyze. At the same time, the weights' matrix and bias are randomly initialized. It is noticeable that the weights' matrix is continuously updated by the neurons when they discover new features. The process is repeated until the data arrive into the output layer in charge to prepare the results.

The interaction between layers happens through the activation functions that establish which neurons are *activated* and which are not into a layer. There are different types of transfer functions; they are summarized in the following paragraphs.

The *sigmoid* function performs the activation when the values are in the range between 0 and 1 [36]:

$$f(x) = \frac{1}{1+e^{-x}},$$

this function is useful for *logistic regression*.

The *hyperbolic tangent* or *tanh*, triggers itself when the values are between -1 and 1:

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}},$$

the main difference between the two models, is that the *tanh* is quicker than the first one.

The *Rectified Linear Unit* function has the activation range between 0 and $+\infty$ and it is expressed by [36]:

$$f(x) = \begin{cases} 0 & for \quad x < 0 \\ x & for \quad x \geq 0 \end{cases}$$

where the activation value is 0 if x is less than 0 and if contrary, it become x. This function can be applied in several different cases [36], such as object recognition or feature detection.

A subset of the model mentioned before is the *leaky ReLU*, where also the values less than 0 are used for activation. The formula is [36]:

$$f(x) = \begin{cases} \alpha x & for \quad x < 0 \\ x & for \quad x \geq 0 \end{cases}$$

where α is a constant and usually takes the value of 0.01.

The *Exponential Linear Unit* function, *ELU*, is similar to the *leaky ReLU* but instead of having a straight ascension of x, it is smooth like a curve [36]:

$$f(x) = \begin{cases} \alpha(e^x - 1) & for \quad x < 0 \\ x & for \quad x \geq 0 \end{cases}$$

Google Inc. has recently introduced the *Swish*, a non-monotonic function with better performance than the ReLU [36]:

$$f(x) = 2x\sigma(\beta x),$$

it is noticeable that when the σ is 0 the activation becomes linear. Another important activation function is $softmax$. It is common to use it as last activator, returning the probability of success of each class analyzed by the neural network:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_i}},$$

where x is an element of the classes vector. Generally, the output is regularized with a loss function that checks the performance of the neural network. The formula is generalized as follow [36]:

$$loss = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2,$$

where y is the actual output and p is the predicted output. The *loss function* also considers the weights randomly initialized, discovering their optimal values with a technique named

gradient descent. Moreover, the predictions are more accurate, and the loss function spreads to the minimum [36]. They are calculated as follows:

$$new \quad weights = old \quad weights - learning \quad rates * \frac{\alpha J}{\alpha W},$$

where $\frac{\alpha J}{\alpha W}$ is the *gradient descent* of the loss function. Other important models are *Adaptive Moment Estimation* and *Root Mean Squared Propagation* [37].

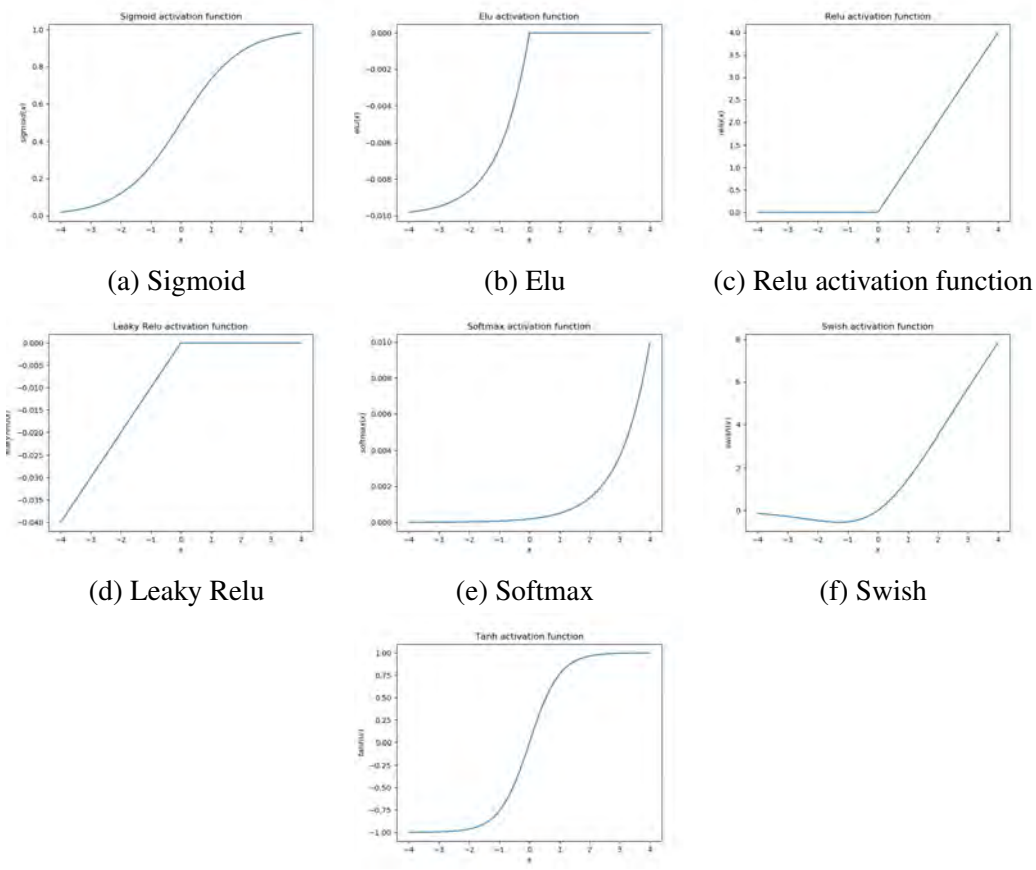
A Convolutional Neural Network (CNN) is suitable as an image classifier. It can study input images pixel by pixel, recognizing the beginning of edges and contours of objects. Moreover, the network becomes deep, when plenty of hidden layers are inserted. In this way, it recognizes more sophisticated objects such as the human body, forest in a determined area, etc. However, plenty of computational power is required to exploit good results. Typically, in a *Deep Convolutional Neural Network* (DCNN), there are various types of layers. Two of the most important are the *Convolutional layer* and the *Pooling layer*.

Convolutional layers have the characteristics of preservation of the spatial information of the pixels [36]. Moreover, the input is divided into tiles. Each of that is sent to a particular neuron that preserves the local structure of the data. The dimension of the matrix is called stride. It is a *hyper-parameter* that needs to be tuned by hand. It should be noted that every convolutional layer in the network has to have the same weights and bias.

In a *Pooling layer* the network extracts the maximum activation from a matrix [36]. It is also called *Max-pooling layer*.

Other important layers are the *Flatten layer* and *Dense layer*. The first one transforms the matrix analyzed by previous layers into a vector. A set of dense layers, called *Fully Connected Network*, is in charge of preparing the output.

However, adding a *Droupout layer* is an excellent practice to normalize and regularize the output. Another important fine-tuning component is the *Zero-padding layer* added between Convolutional layers, where zeros are added as contours of the matrix analyzed. This process achieves better precision when the network discovers edges and shapes of objects.



(a) Sigmoid (b) Elu (c) Relu activation function

(d) Leaky Relu (e) Softmax (f) Swish

(g) Than

Figure 4.4: The most common activation function of neural network.

4.2.2 TensorFlow

TensorFlow is an open-source framework developed by Google, which became one of the most used libraries for numerical computation. TensorFlow uses CPU, GPU, or both, to parallelize the calculation making it faster. It works with almost all operating systems and embedded devices. The Google package operates in symbiosis, acting as the backbone for other systems such as *Keras*. TensorFlow is a declarative framework [27]. Moreover, the operations to compute are added to a core called *computational graph* with the particularity that is just added and not executed. In order to perform the calculation, a session object is created [27].

The TensorFlow's core is based on the concept of tensor that is widely used in mathematics and physics. The tensor is defined as an array of numbers or a multiplication between vector and real number. Moreover, it is possible to represent a satellite image as a tensor. For example, one index on the vector is the height, another one is the width, and the third one is colour. The process is called *featurization*, the representation of world

objects as an entity, storing the properties in vectors [27].

For machine learning, it is critical to find the correct way to store data as tensor [27]. Furthermore, one of the basic operations that a neural network computes is vector multiplication. The rule behind is the same rule applied to the matrix multiplication. If A has shape (m, n) where m are the rows and n the columns, it is possible to make a multiplication with a matrix B with the number of columns equal to n [27].

$$Tensor_t = \begin{bmatrix} ax+by \\ cx+dy \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$$

Machine Learning models use vast amounts of tensors with millions of parameters [27].

IMPLEMENTATION AND RESULTS

In this Chapter, I will elaborate on the practical implementation of the methods presented in the Chapter 4.

5.1 EXPLORATORY ANALYSIS OF SATELLITE IMAGERY USING GOOGLE EARTH ENGINE AND PYTHON

The first stage of the proposed analysis techniques is the exploratory forestry status of the Kvarken Region. Over the years, the number of trees in the area has changed rapidly. This analysis aims to study the number of tree loss and the ones replanted. The API chosen is the Google Earth Engine, and the code is written using Python programming language and Jupyter Notebook. The structure is simple, and it shows how to manipulate diverse bands of the Landsat satellites. Moreover, the bands used in this work are summarized in the Table 5.1.

Table 5.1: Summary of the Landsat satellites bands used for the forestry assessment. [38]

Name	Wavelength	Description
treecover2000		tree cover
loss		tree loss
gain		tree gain
first_b3	0.63 - 0.69 micrometre	red band for year 2000
first_b4	0.77 - 0.90 micrometre	NIR band for year 2000
first_b5	1.55 - 1.75 micrometre	SWIR band for year 2000
first_b7	2.09 - 2.35 micrometre	SWIR band for year 2000
last_b3	0.63 - 0.69 micrometre	red band for year 2012
last_b4	0.77 - 0.90 micrometre	NIR band for year 2012
last_b5	1.55 - 1.75 micrometre	SWIR band for year 2012
last_b7	2.09 - 2.35 micrometre	SWIR band for year 2012

The dataset is the *Hansen Global Forest Change v1.6* from a time frame of 2000 to 2018. The data collect information on the status of forestry. The constellation of satellites used

is the Landsat. There are four spectral bands: *band B3 band B4 band B5* and *band B7*, divided into two groups as the Table 5.1 summarizes [38]. The images have a resolution of 30 metres.

After the initialization of the Google Earth Engine API, the world map is loaded into one object. The Kvarken geographical information is given through a parameter. It is a list of five elements: the upper left point, upper right point, lower left point, and lower right point and centre point. The output appears on the screen using the special function *Image*. The first outcome is the infrared frame of the zone. The band used is B3 with a wavelength between 0.63 and 0.69 micrometres [38]. However, this is not enough to understand the gain and loss of the forest.

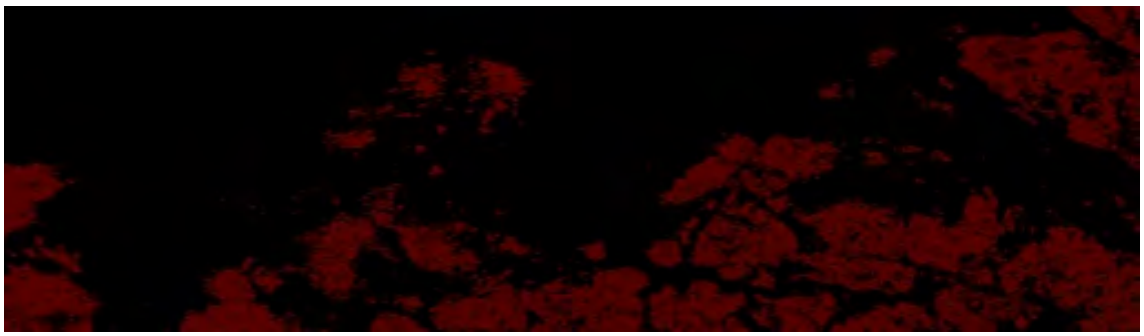


Figure 5.1: The Kvarken Region viewed selecting the band 3, "B3", with a wavelength between 0.63 and 0.69 micrometres.

The next step is to verify how the particular band called *treecover2000* works. It represents a percentage of trees in the selected areas. However, the output is not sharp because it is in greyscale. The sea is masked as black. Here, the trees coverage has different colours of greyscale. Instead, black pixels represent the area without trees.

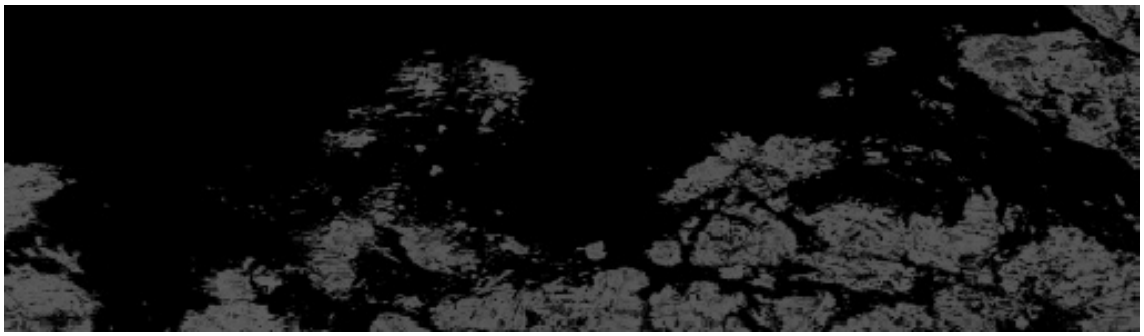


Figure 5.2: The Kvarken Region viewed selecting the treecover2000 layer. Excluding the sea (coloured in black outside the perimeter of the land, represented in grey), the black points into the grey zones represent the most damaged forest areas.

Combining the various bands, such as *the loss*, *the treecover2000*, and *the gain*, the image

is overlayed with different pixels, blue and orange. The green shows the land covered by forest, the orange shows the tree loss and the blue underlines new trees. The working principle is that the band contains an array with three different colours tonalities. These numbers serve to highlight with assorted colours, a part of the images. The outcome shows that many areas of the Kvarken Region has lost forestry. Looking at the result, only a few zones obtain the same number of trees in addition to the gain zone.

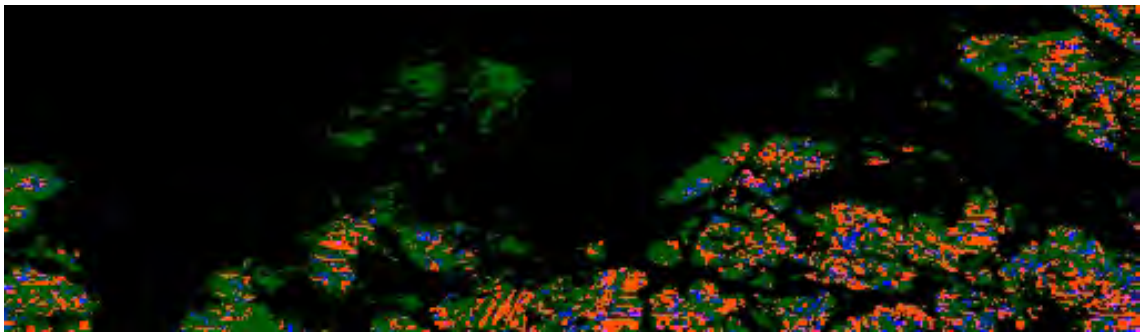


Figure 5.3: This image shows the zones where loss of trees is detected (in orange), and the areas where new trees are planted (in blue). The tree loss and tree gain layers date back to 2017.

Changing the dataset version, into *V1.5 of 2015* and adopting the same three bands as before, the forestry situation is entirely different. Only a few zones are in orange, with loss, and almost 70% of the map stays green, meaning that the terrain is covered by forestry. The difference is due to an increase in agriculture in recent years.

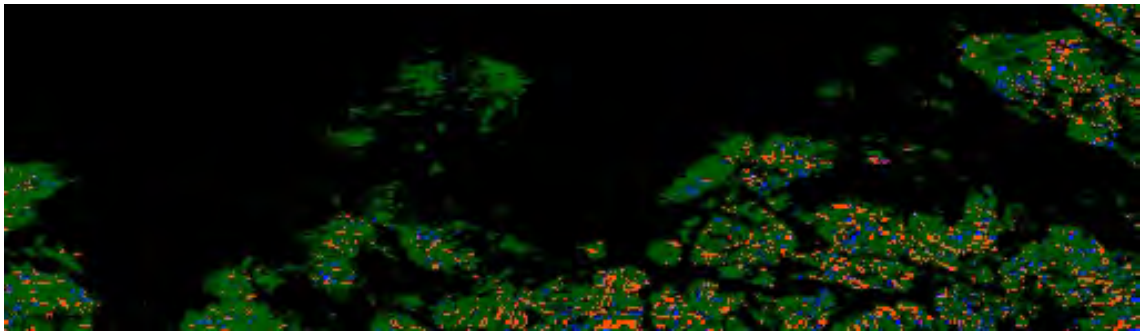
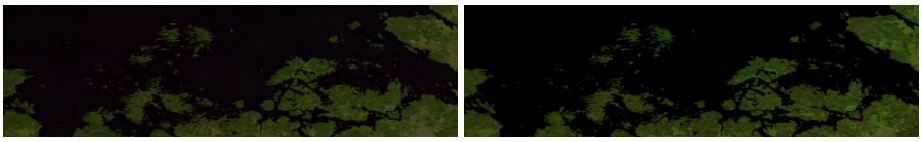


Figure 5.4: Same representation of the previous image, but in 2015.

The dataset contains six important bands: *B3_first*, *B4_first*, *B5_first*, *B3_last*, *B4_last*, and *B5_last*. The first three describe the healthy vegetation status of the area in the 2000s. Opposite, using the last three, the image shows the status of the trees twelve years later, in the 2012s. As described before, the two images show the varied scale of green to highlight the areas with more trees. The light-coloured zones identify housing or urban areas. For example, there are big changes in the low right corner, that in the year 2000 had plenty of

vegetation, and in the year 2012 the trees had disappeared because new agricultural fields and new houses were built. See Figure 5.5.



(a) The Healthy vegetation in 2012 (b) The Healthy vegetation in 2000

Figure 5.5: These two images show the union of bands 3, 4 and 5. This combination shows the status of healthy vegetation in the region.

Using the short wave infrared bands it is possible to see with different shades of purple, the loss and gain plants in the 2012s. See Figure 5.6.

Figure 5.6: The Kvarken Region viewed with short infrared waves in 2012.

Furthermore, another good example is the usage of the *B4 band*. The process is the same as before: looking at the differences in colours. In this case, the image is clearer, but the zones of interest are with bright colouration. The blue pixels, which mean plant loss, are barely visible. However, using B7 bands, the image is darker, but the zones with a decrementation of forestry are more visible. See Figure 5.7.

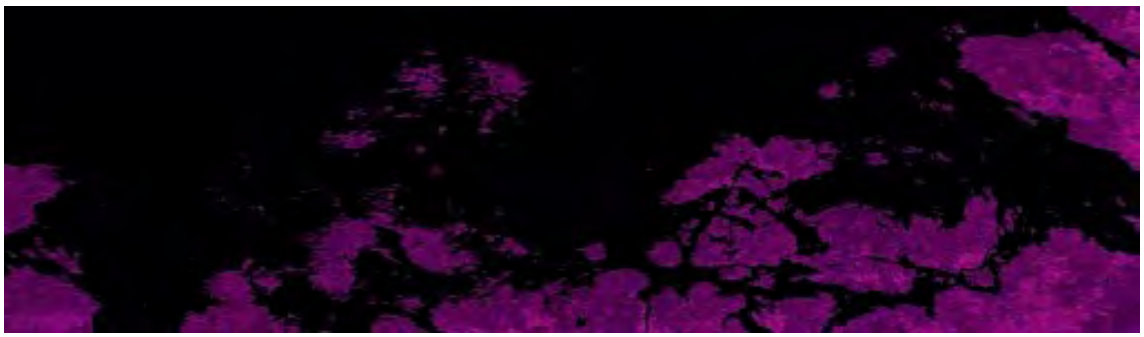


Figure 5.7: The Kvarken Region viewed with short infrared waves in 2017.

Google Earth Engine also offers a numerical analysis of the dataset. One principle is the selection of a band; in this case, loss and gain. The class in charge of the job is called

Reducer. At a low level, the code can sum the same kind of pixels, grouping them with affine properties. This set of functions makes it possible to aggregate trees loss and gain in two variables, over time and space.

However, the choice of the correct band is essential for achieving satisfying numerical results. The calculation of the total forestry loss is the result of the function called *reduceRegion*. The reducer sums all the orange pixels of the zone in the selected place. In the case of the tree gains, the analysis sums the blue pixels. There is the opportunity to feed the reducer functions with different parameters, for example, the *mean* or *standard deviance*. The results are in the Table 5.2

Table 5.2: The table summarizes the gain and loss of trees in the Kvarken Region from 2000 to 2012.

Type	Number
Trees loss	63579
Trees gain	18358
Mean loss	0.0063
Mean gain	0.0018
Dev. Std. loss	0.072
Dev. Std. gain	0.042

The *lossyear* member of the dataset analyzed in this Chapter contains vital information regarding the area in square metres of trees. It is grouped by different year, from 2000 to 2017. These numbers are constructed within a ratio between the band of trees gained and the one with the lost. A reducer selects the area with the geographical coordinates to obtain precise results for the zone.
Table 5.3 shows that in 2000 the Kvarken Region appears to have not registered any tree loss, but from 2001 onward the process of deforestation started. The data is also grouped in Figure 5.8.

Table 5.3: Forest loss in square metres of the Kvarken Region from 2000 to 2012.

Year	Forest loss in square metres
2000	0
2001	2307335
2002	1152117
2003	2198331
2004	2747143
2005	3060944
2006	1328213
2007	2874593
2008	1757349
2009	1989698
2010	804306
2011	985641
2012	1681498
2013	1094693
2014	1185838
2015	824775
2016	1436848
2017	21466462

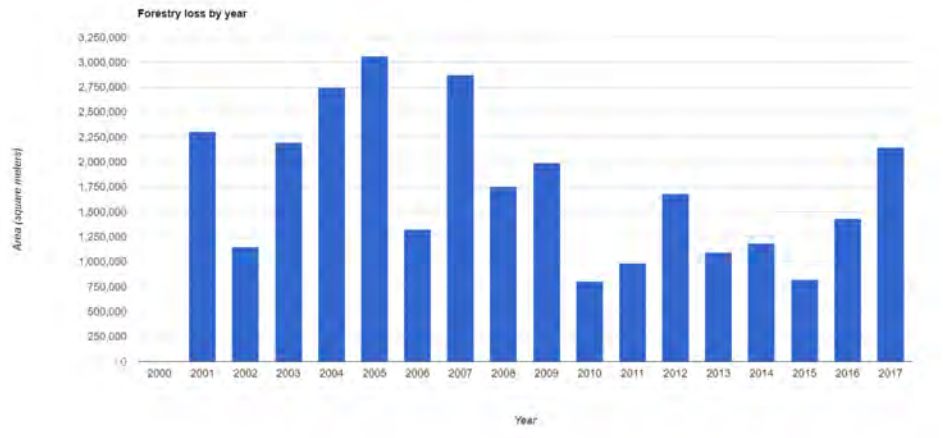


Figure 5.8: The graph shows the area in square metres of forestry loss. Every column summarizes a year.

5.2 UNSUPERVISED LEARNING

5.2.1 K-Means Clustering

In the following pages, two different implementations of K-Means clustering are proposed. The first analysis uses the famous image framework *OpenCV*. The second one

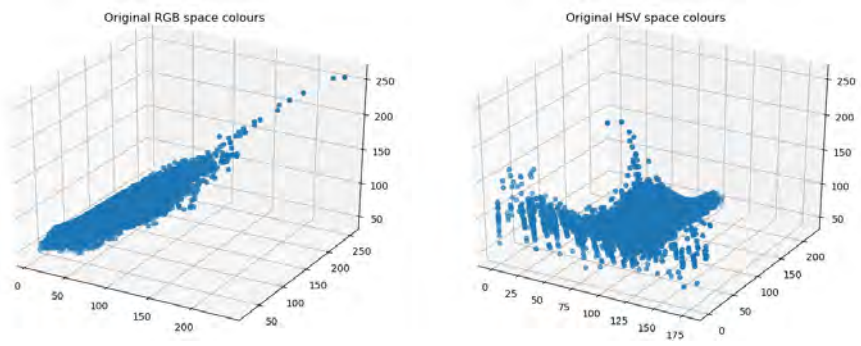
uses the *MATLAB Machine Learning package* that provides various functions to conduct different operations. The images used for these examples come from the Google Earth Engine and ArcGIS, in order to check the differences between low-resolution and high-resolution images. In addition, the dataset provided by Google Earth Engine contains images from 2015 during the summer season in an interval between June and July. The second dataset is downloaded with ArcGIS without any temporal references.

For the first part of the study, the programming language used is Python with the packages listed in Table 5.4.

Table 5.4: Packages used for the OpenCV analysis.

Packages	Purposes
OpenCV	Image editor
Matplotlib	Plot figure
Numpy	Array management

The first step is to load the input image into a variable. The task is accomplished using the *image read* function of OpenCV. The image is converted into a matrix. The data structure is simple: rows and columns; and it is loaded using the *RGB colours space*. This format is a good standard for images because it distinguishes well the light that is reflected by objects. In terms of clustering, it is a good practice to convert the RGB format in *HSV format* (Hue, Saturation, Value) because the algorithm can calculate the variance of every single pixel and discover changes into the colours' saturation. OpenCV can transform the colour space with the *cvtColor* function. It takes as parameters the original image and the expected colour schema. To better understand the differences in the two colours spaces, the Figure 5.9 is beneficial.



(a) RGB space colours. (b) HSV space colours.

Figure 5.9: The two graphs show the difference between RGB and HSV colour spaces.

In the original RGB image, the colours are grouped linearly from 0 to 250. However, there are a few pixels that differ from the central pixels' assemblies. In the HSV plot, the image components are more dispersed. In a very few cases, their values except 200.

The input image is reshaped into a matrix of three columns, and the components inside are converted into floating-point numbers to infer a more precise result. The *OpenCV K-Means cluster* algorithm needs a setup to produce proper outcomes. Furthermore, after the reshaping, the desired number of clusters and a special stopping parameter is prepared, prompting the algorithm to stop and display the results, in case of a hit. The two essential outputs for this analysis are the *labels* and the *centroids*. At this point, before the image is regenerated, the algorithm groups every cluster centre based on their respective label. The Figure 5.10 shows the measured outcomes with two clusters.

Figure 5.10: K-Means clustering analysis with two clusters.

Figure 5.11: K-Means clustering analysis with four clusters.

Figure 5.12: K-Means clustering analysis with five clusters.

Figure 5.13: K-Means clustering analysis with three clusters.

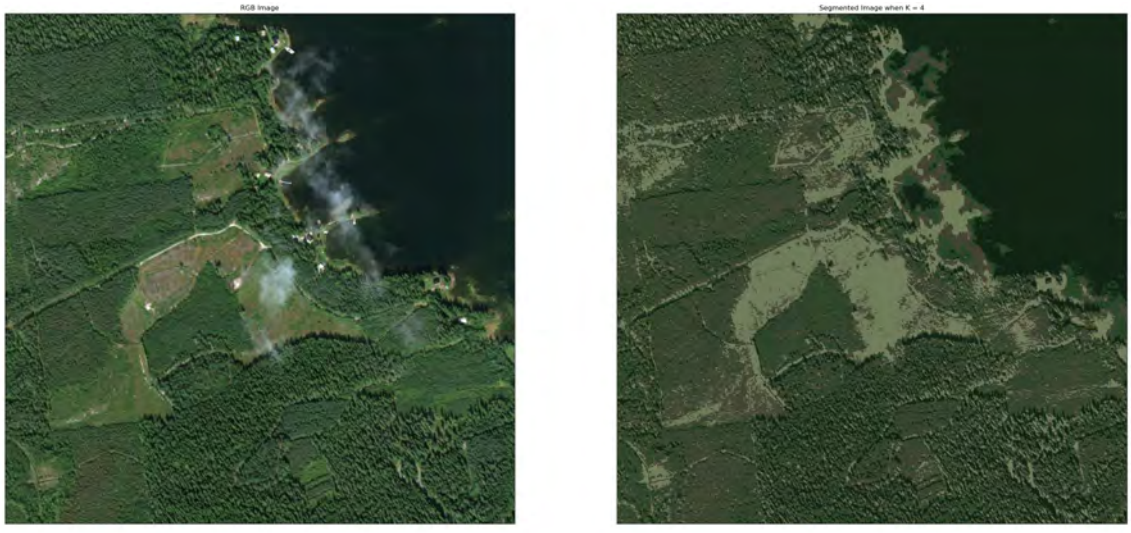


Figure 5.14: K-Means clustering analysis with five clusters.

The K-Means clustering algorithm is tested with two, four, and five clusters. The differences between 2 and 4 clusters are noticeable in the low-definition images. After that, the results only lightly change, showing that four clusters illustrate the consistent result. The output is beneficial when the K-Means clustering studies the images provided by ArcGIS (from Figure 5.10 to Figure 5.14). The two-cluster analysis shows that the model hits the deforested zones and streets. The four-cluster model assures better results. The five-cluster analysis is interesting because the deforested zones are accentuated. In conclusion, *four* is a good cluster number because it underlines the sea, land, and forestry areas.

Another powerful clustering tool-set is provided by *MATLAB*. It is part of the *Machine Learning package*. The point of effort here is the achievement of good results only using few lines of code. The short-code is summarized in a function that takes two arguments: the image and the desired clusters number. Furthermore, the algorithm, called *kmeans*, returns the labels and centroids. At this point, the image is reshaped and sorted based on its cluster centre and the corresponding label. The image is converted to RGB. This model achieves clearer results than the OpenCV ones.

Two tests are reported, one has the number of clusters fixed to four and the other one has six instead, see Figures 5.15 and 5.16.

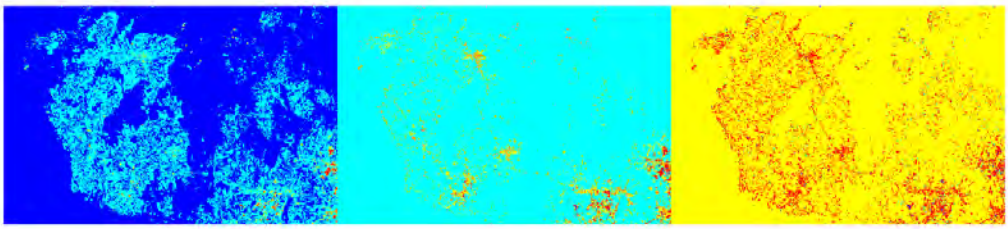


Figure 5.15: K-Means clustering result with four clusters.

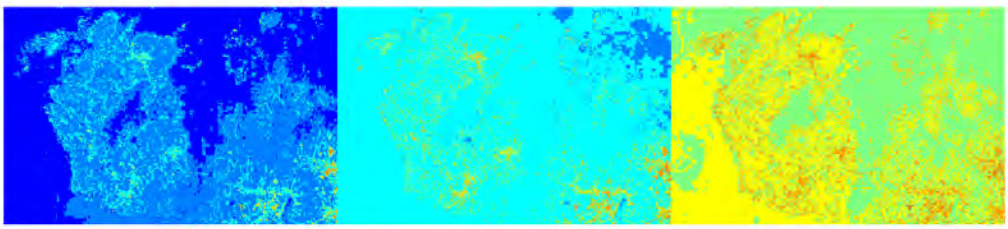


Figure 5.16: K-Means clustering result with six clusters.

The output is divided into three images: the first one shows the overall result of the K-

Means cluster analysis, where the dark blue represents the sea, the yellow shows the dense forestry, the light blue refers to the vegetation, and the red represents urban areas. The second image, shows the status of the vegetation and the urban regions. The third image is reserved for the cluster with more labels.

The low-resolution images are also experimented with MATLAB, concluding that: until four clusters, it is possible to appreciate evident changes in the final segmentation. The results are very good. Increasing the number of clusters, the differences in the various outcomes are not noticeable. Using six clusters, MATLAB can underline the streets in the image, and the different islands that are connected to the town.

Taking in consideration the high-quality image gathered from ArcGIS Desktop, the results are great. They are reported in Figure 5.17 and 5.18

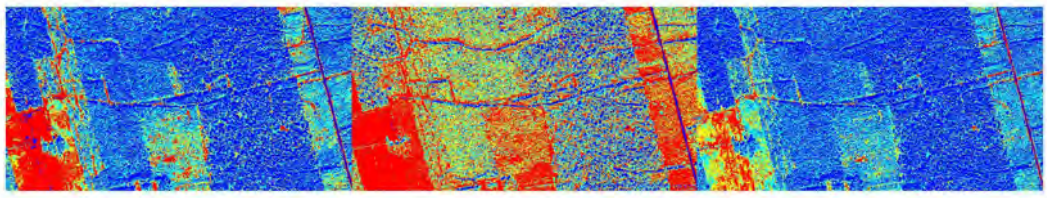


Figure 5.17: K-Means clustering result with fours clusters executed with high-quality image.

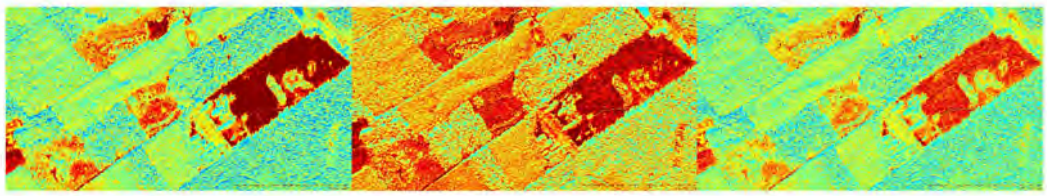


Figure 5.18: K-Means clustering result with eight clusters executed with high-quality image.

OpenCV and MATLAB are suitable frameworks able to assess the status of the forestry. Moreover, they analyze every pixel of the input image, discovering in which image's portion the trees are missing. However, the final comparison between the healthy forestry areas and the faulty ones still needs to be done by a manual comparison. However, by using the high-resolution images, the results become excellent.

Two more clustering algorithms achieve optimal results, showing more precisely the vegetation quality: the *K-Means image segmentation function* and the *Gaussian Mixture*

Model.

The *KmeansClustering* function can segment an image provided as input. The model is tested with four and six clusters. It gives different final information than the traditional K-Means clustering algorithm. It shows more detailed data. The analysis shows the region of the image where there is healthy vegetation, coloured in dark blue, and the status of the loss of trees that is in light blue. However, the region with a high-density of trees and housing are visualized with the same colour. This model is more specialized in the density calculation of the pixels than the division of the image into clusters. The six clusters analysis shows more clearly the forestry zones, but in some places, the output is full of noise. It should be noted that the model with six clusters shows more precisely the forestry status, concluding that it is an important tool to be considered for the forestry damage assessment. The results of the *KmeansClustering* are shown in the Figures 5.19 and 5.20. Using the high-resolution images, the results are excellent, see the Figure 5.24.

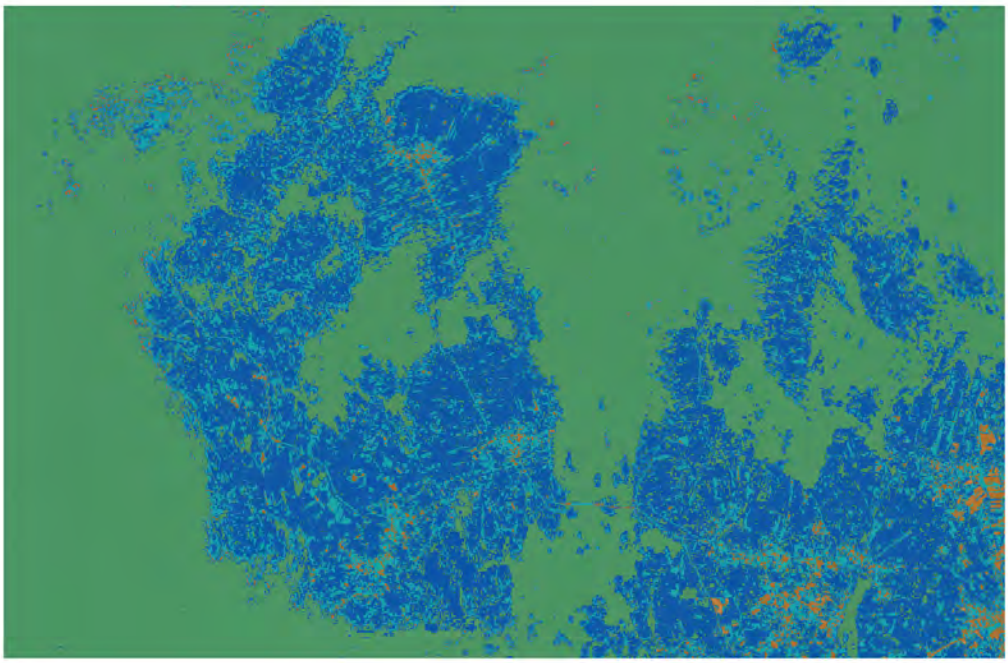


Figure 5.19: K-means image segmentation with four clusters, using *KmeansClustering* from MATLAB.

The Gaussian Mixture Model (GMM) function, assumes that the input image has a certain amount of Gaussian distribution. MATLAB uses complex functions that quantify this value. Moreover, it can segment the image and makes a comparison of their distribution. The input parameters are the number of clusters and the input image. A series of three images represent the resulted analysis, see the Figures 5.21 and the Figure 5.22. In the

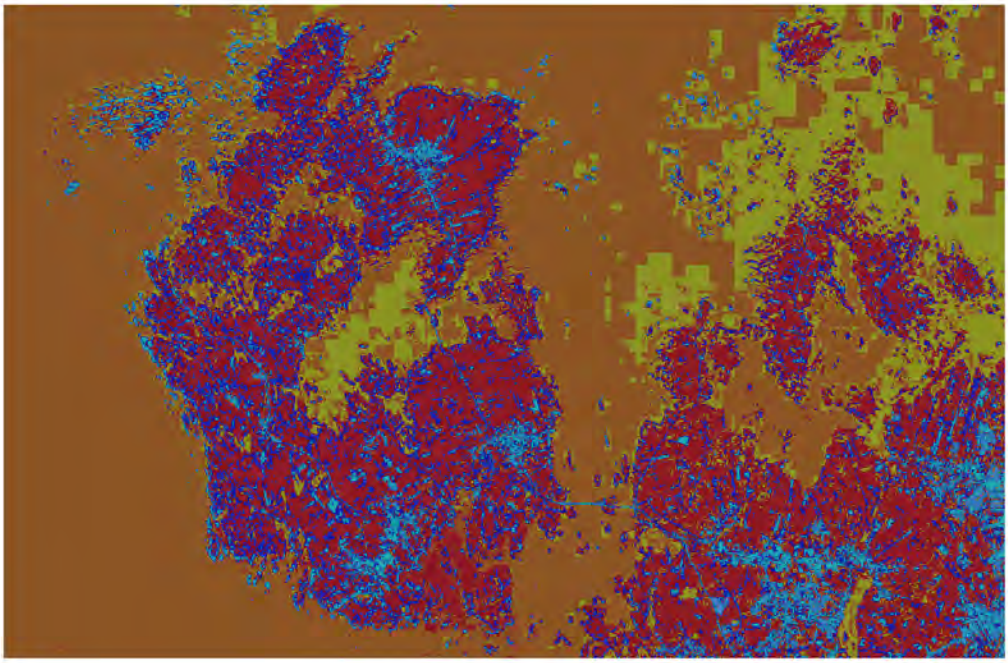


Figure 5.20: K-Means image segmentation with six clusters, using *KmeansClustering* function outcome from MATLAB.

first one, the towns and the zones with plenty of vegetation are highlighted in red. The second image shows a better view of the result. The zones with more interesting pixels are marked clearly. However, the algorithm studies the pixels' density better than the cluster division of the image. GMM is another potent tool to detect anomalies in the vegetation because it shows the difference between zones from a various image, such as the Figure 5.19 and the Figure 5.20. The Figure 5.23, shows the result for the high-quality image.

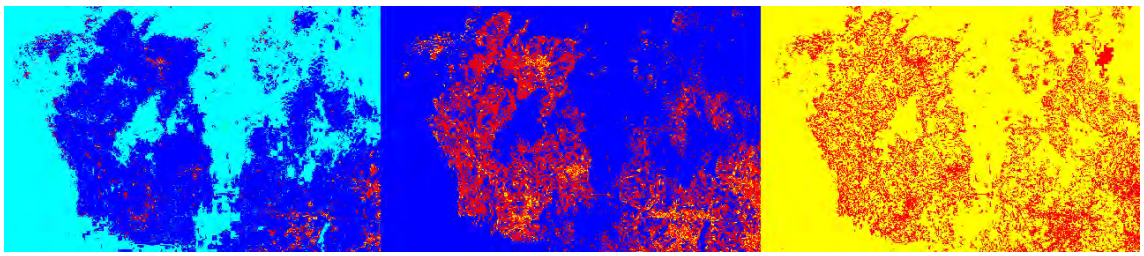


Figure 5.21: GMM segmentation result with with four clusters.

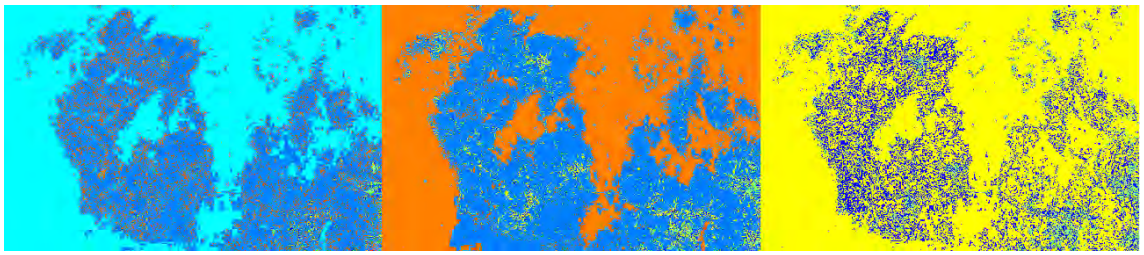


Figure 5.22: GMM segmentation result with six clusters.

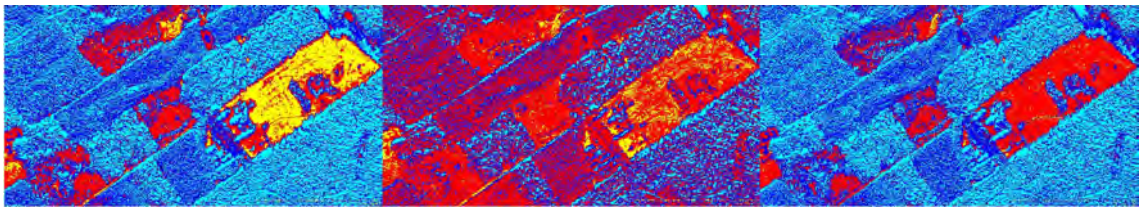


Figure 5.23: This image shows the GMM analysis executed with high-quality image

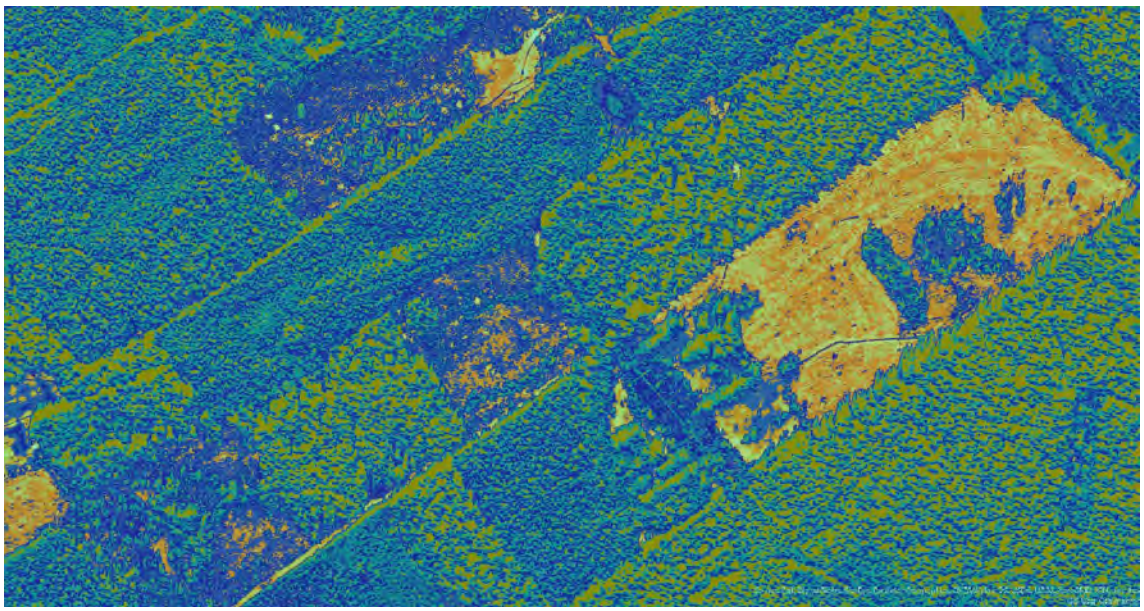


Figure 5.24: This image shows the result achieved using the tool K-means segmentation of MATLAB.

5.2.2 PCA analysis

In this subsection, a set of satellite images are taken. They have the same coordinates, approximately 63.23 degrees in latitude and 21.29 degrees in longitude. The repository used is the Google Earth Engine Sentinel archive. I collected the total number of eight images. Each one, has only one band. The idea is to extract the *Principal Component* from every image, rebuild another one with only the salient features, eliminating the noise.

The analysis uses Python programming language. The following packages are installed and reported into the Table 5.5.

Table 5.5: Packages used for the PCA analysis

Packages	Purposes
OpenCV	Image editor
Matplotlib	Plot figure
Numpy	Array management
Panda	Data management
Seaborn	Plot figure

The images are resized with the same width and height, (600 pixels x 600 pixels). The data is loaded into a *Numpy* array and converted into greyscale reducing the images' channels to one. Moreover, each input data is an element of the array. The structure is a matrix, where all pixels have defined positions located by *n-rows* and *n-columns*. The images collected are presented in Figure 5.25.

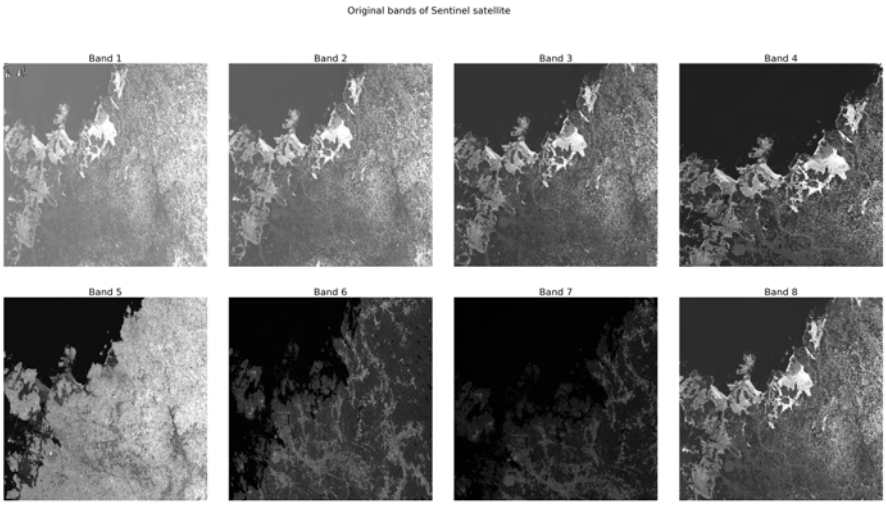


Figure 5.25: This image lists eight bands from the Sentinel-2 satellite of the Kvarken Region.

Figure 5.25 illustrates that every band is different because the satellite used various parameters to capture the image. In the bands 1, 2, 3, 4 and 8 it is possible to distinguish the snow. In the bands 6 and 7, there is a clear view of the forestry zones and the urban areas. At this point, it is crucial to ensure the consistency of the images. An excellent way to do so, is to transform the array in vector. Moreover, the real standardization is obtained by calculating for each element the following value:

$$element[i] = \frac{(element[i] - element.mean())}{element.devStd()}$$

With this standardization, the PCA model calculates the *correlation* between all components and it checks which of them goes in a different direction. At this point, the *Eigenvalues* are calculated [39]. During the vector transformation, it is possible to represent them by the number that helps the linear transformation of the vector itself:

$$new\_vector = \beta \times vector$$

where β is the Eigen-value. Fortunately, this complex set of values are an easy step to set using Python and Numpy utilities. After the covariance calculation, the values are obtained with the following Numpy function:

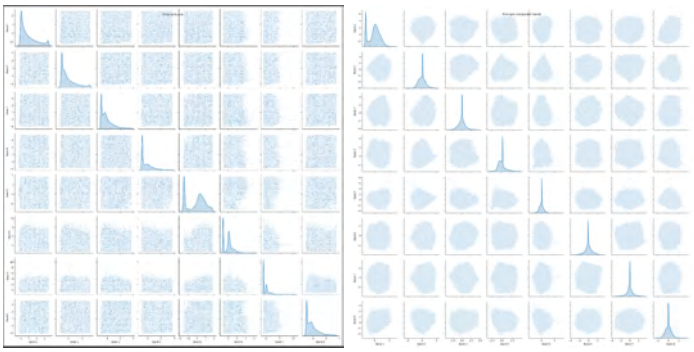
$$eigen\_values, \quad eigen\_vectors = np.linalg.eig\,(np.cov\,(bands))$$

the resulted values are summarized in the following Table 5.6:

Table 5.6: Summary of the Eigen value

Band	Eigen-Value
1	3.158
2	1.298
3	0.282
4	0.838
5	0.75
6	0.509
7	0.607
8	0.558

These values are crucial because they show the direction of every variance into the vector. The larger the Eigen-values is, the bigger the component significance is [39]. To better understand the reductions algorithm, the graph in Figure 5.27 reports the variance of each component. It is noticeable that from the first band, the variance of the components decreases because the reduction operates by eliminating plenty of data. At this point, the Principal Component is obtained, with the multiplication between the *Eigen-vector* and the image matrix. One can see the differences between original bands and the principal component in the following graphs in Figure (5.26a and 5.26b).



(a) The collection of graph shows the covariance of the pixels in the original bands

(b) The collection of graph shows the covariance of the pixels in the PC bands

Figure 5.26: This set of images shows information about the bands of the satellite image

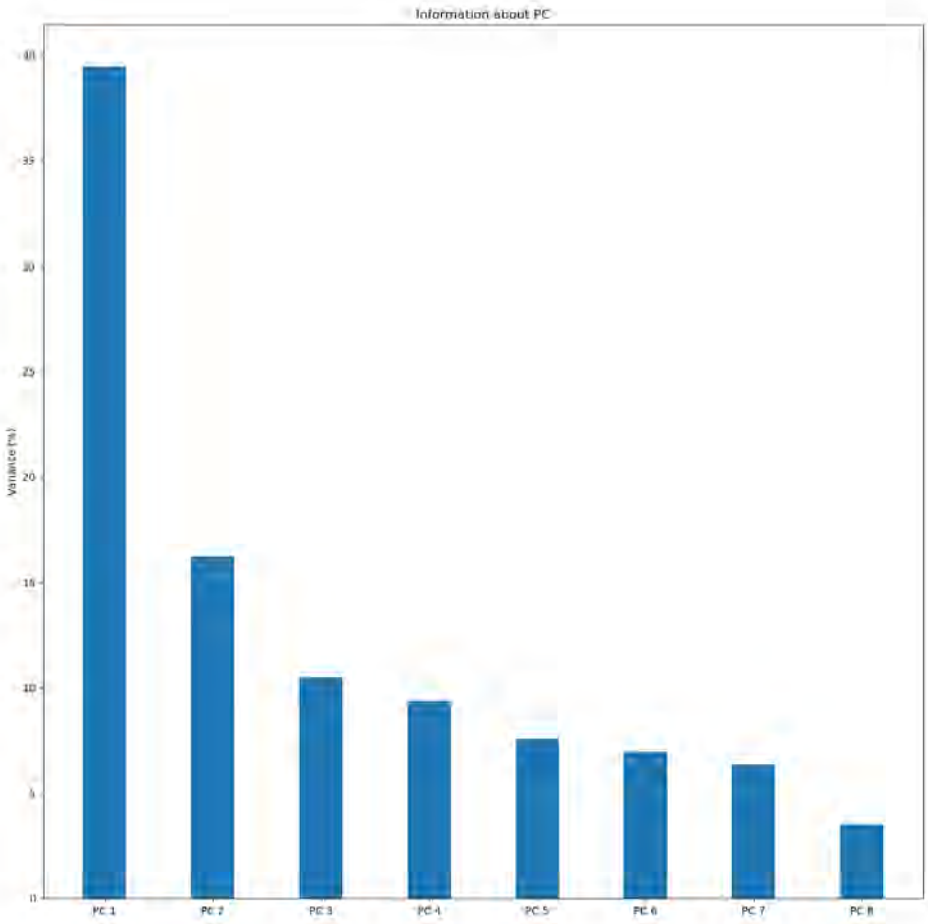


Figure 5.27: Histogram representing the variance of various bands

In conclusion, the eight bands can be exploited with PCA analysis, see Figure 5.28. There are many differences between the first image and the last one because the reduction algorithm has to operate by isolating the component with less variance respect to others. For example, in image number four, the snow is visible and instead in the number eight, only the forestry and housing can be seen.

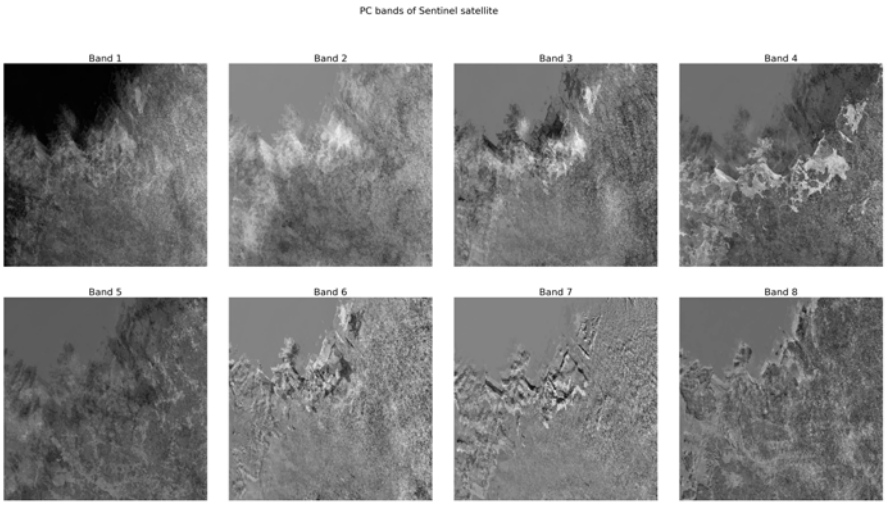


Figure 5.28: This image shows the Principal Component from different bands

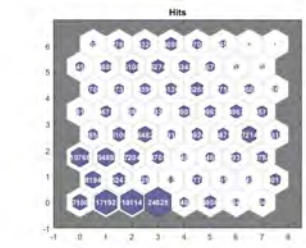
5.2.3 Self-organizing maps (SOMs)

Here I propose a *Self-Organizing Maps* implementation using the MATLAB library Machine Learning. The code is organized as a function that takes two arguments, the input image and the dimension of the neural network. In this case, these layers have dimension of 8 x 8. The image is read and stored into a three columns array, that represents the three colours channels (RGB). At this point, the software resizes the input with a ratio of 0.5 concerning the original input shape. It is crucial to organize the input image into two variables called *rows* and *cols* that are equal to the dimensions of the image. They are also necessary for the final reconstruction of the image.

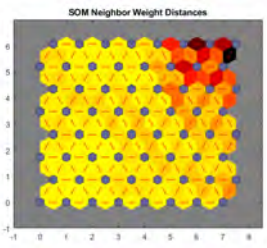
Before processing with the SOMs analysis, the data needs management. This is achieved by transforming the initial matrix into three different vectors, where each of them represents a colour channel. However, MATLAB uses the input number as double, so *casting* (the conversion process of a data type to another) is necessary. The normalization stage uses a straightforward methodology. It starts with a subtraction between the vector element and the minimum of the vector. The result is divided again by the difference between the maximum and the minimum of the vector. Moreover, the process is repeated for the three RGB channels of the image. The formula is:

$$normalizedVectorChannel = \frac{(vectorElement - min(vector))}{(max(vector) - min(vector))}$$

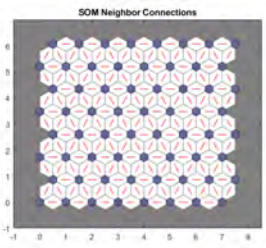
The neural network characteristics are summarized in the Figure 5.29.



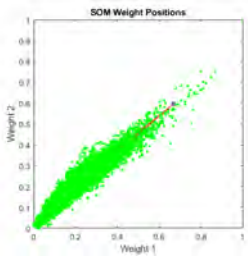
(a) The summary of the winner neurons



(b) The SOMs Neighbour weight distances



(c) The SOMs Neurons connections



(d) The SOMs Neighbour weight positions

Figure 5.30: Results of the SOMs analysis. The output shows the creation of four different graphs. The first one shows the winning neuron in the final stage, the second one shows the weights of neighbours neurons, and the third one indicates the connection of various neurons, the last one shows the weight position.

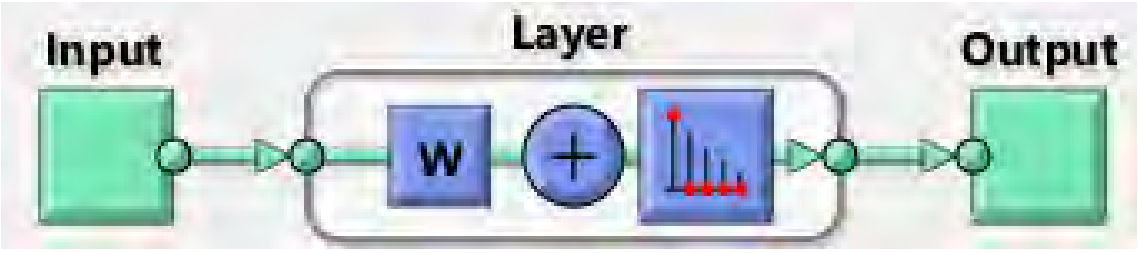


Figure 5.29: Schematic representation of SOMs layers

At this point, the training of the network starts, and the output is presented. MATLAB shows plenty of different figures, such as Figure 5.30. In the end, the pixels' labels are rebuilt with the following output, shown in Figure 5.31

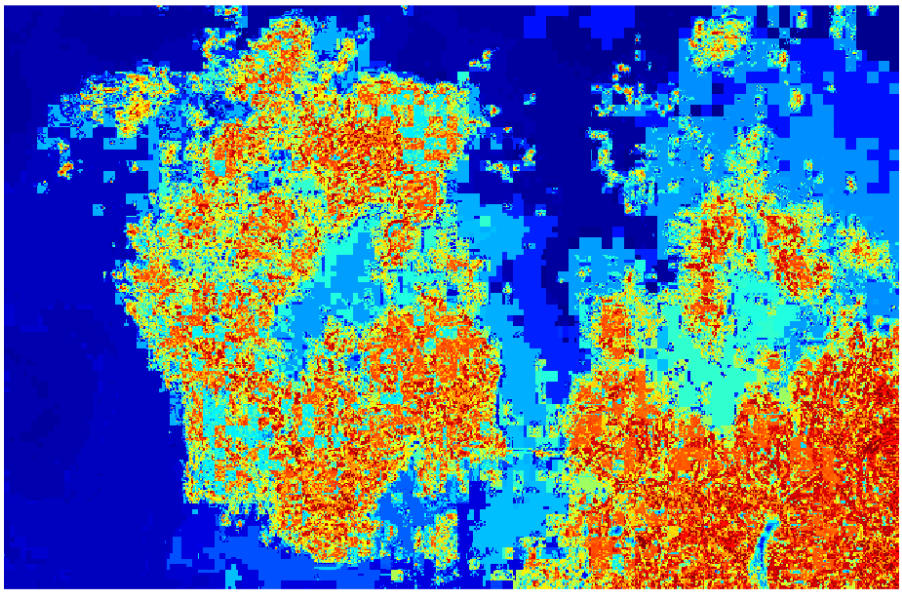


Figure 5.31: The resulted image summarizes the output of the SOMs. The clustering algorithm delineates water, land, vegetation and urban areas, and so forth.

In conclusion, looking into the Figure 5.30a, the summary of the *hit graph*, shows the most interesting pixels and their respective grade of connection with the others. In total, there are 64 neurons involved in this analysis. In the Figure 5.30a, the hitting points in the lower part of the graph assume significant values, between 17 thousand and 24 thousand. Moreover, in this portion of the image, the neurons have discovered forestry because in the input photos, there was the most common pattern. The same results come with the *Neighbour weight distance graph* where there are a significant concentration of them in the upper part of the graph. This connection becomes very strong underlining the green area of the input image.

The *connection graph* shows the fully connected network with all the 64 element members (Figure 5.30c). Figure 5.30d shows the scatter plot of the weight, which changes every time a neuron hits a point. In the final reconstruction of the image (see Figure 5.31), the SOMs model summarizes the pixels' density detecting zones with forestry, and also the one with few trees. The outcome shows bridges, lakes, and so forth.

5.3 DEEP LEARNING

5.3.1 Image Segmentation with TensorFlow

TensorFlow is an open-source framework for *tensorial programming* with vast capabilities for image analysis. It can manage large scale of multidimensional arrays within a few lines of code. For the purpose of this exercise the packages in Table 5.7 are used.

Table 5.7: Summary of the packages used

Package	Purpose
OpenCV	Image editor
TensorFlow	Backbone for neural network
NumPy	Array management
Pix2Pix	Image to image translator
Matplotlib	Plotting utility

The dataset is small, formed by seven satellite images, and their masks. The source of the data is the Google Earth Engine archive. The masks are created by using MATLAB tool called *Image-Segmenter*. The image's sources are the *Landsat* and the *Sentinel* repositories. The input data is prepared, combining each image with each mask. Two *Numpy* arrays hold the input. The images are normalized in order to achieve better results. The input schema have four dimensions:

$$X\_batch, Y\_batch = [batch\_number, image\_counter, image\_width, image\_height, image\_channel]$$

The Python script is formed by a class that groups the neural network builder and a set of utilities. The model is *pre-trained*, and it called the *Mobile Net Version 2*. This network is a member of the family SSD, *Single Shot MultiBox* detector. The training part occurs in a network with only one hidden layer. Therefore, the execution speed is improved [40].

The Mobile Net is a light-weight model. It is computationally fast compared to other networks specialized in semantic segmentation. The speed analysis has in average 30 milliseconds per frame, for example, the *faster RCNN-NAS* has a speed of about 1800 milliseconds.

The base model is loaded into the memory, it saves the weights for the further analysis conduction. The neural network has two branches: the *encoding layer* and the *decoding layer*. The first one is used to decompose the image to discover features. The building

blocks use the *Rectifier-Linear-Unit* activation. The characteristics are summarized into the Table 5.8.

Table 5.8: Summary of the encoder

Block	Kernel Stride
Block 1	64x64
Block 2	32x32
Block 3	16x16
Block 4	8x8
Block 5	4x4

The second branch of the neural network has a stack that re-built the image, (the decoding layer). *Softmax* is the final activator. The layer has characteristics summarized into the Table 5.9.

Table 5.9: Summary of the decoder

Block	Kernel Stride
Block 1	4x4 -> 8x8
Block 2	8x8 -> 16x16
Block 3	16x16 -> 32x32
Block 4	32x32 -> 64x64
Block softmax	64x64 -> 128x128

The stack layer uses a new TensorFlow utility called *pix2pix*. It is a *mini conditional GAN* [41] where a generator and a discriminator are constructed with a *Convolution Batch Normalization* architecture [41]. The union between the encoder and the decoder happens with a *Concatenate layer*. The input layer is on the top of the encoding stack. At this point, the model is created and compiled. The history is calculated with a batch size of 7 members and 200 epochs per batch. With so few images, the accuracy is not very incisive. However, the prediction works very well, as shown in the Figure 5.32.
The predicted mask shows the most interesting points. In particular, the blue, the white and the red pixels underline where the trees are absent. It is noticeable that the sea and even the lake on the upper left corner are not been represented in the final mask, which means that this model is a good starting point for the forestry analysis.

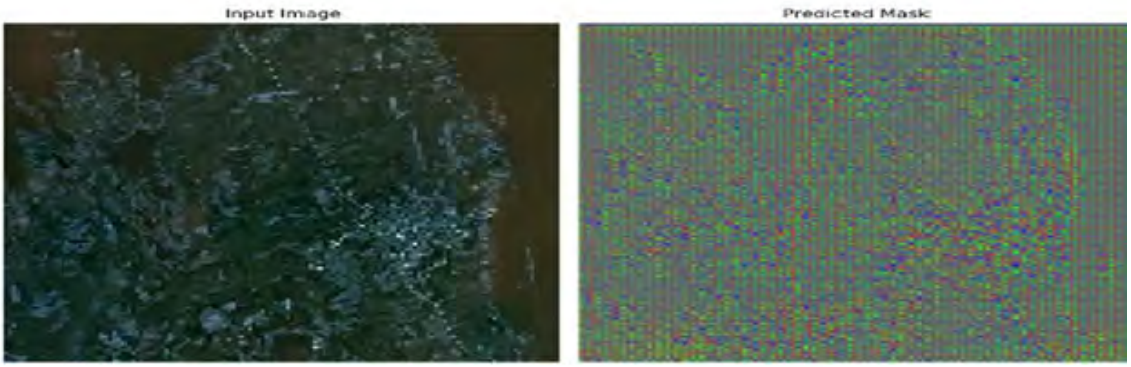


Figure 5.32: This image shows the outcome of the pix2pix analysis of the Kvarken Region.

5.3.2 U-Net

In the following section, the *U-Net neural network* is implemented to detect forest zones and subsequently to assess the damage. This model studies the satellite image at the pixel level. When these pixels are regrouped, a label that describes the object is assigned. The output of the U-Net usually is represented by a predicted mask where it is possible to find the instances or objects.

U-net has many applications, for example, biological analysis [42], study of cancer [43] and also remote sensing [44]. In the *Geo-Sensing* field, the model can classify the elements of a satellite image in different areas, such as urban zones, agricultural fields and industrial areas.

The model is divided into two parts: an *encoder* and a *decoder*. The first one is composed by a combination of a *Convolutional layer* and a *Max-pooling layer*. In between, in order to increase the accuracy of the model, a *Dropout layer* regularizes the output before sending it to the next layer. The number of filters start with 16 and scale-up until 256. Moreover, the activation function is *ELU*, Exponential Linear Unit, explained in 4. The *kernel-size* has three dimensions. At this point, a *Transpose layer* prepares the image for the next branch, the decoder. The workflow is similar to the previous one, but the number of neurons for each layer start from 256 and it decrease to 16. The input layer takes as input the image with shape (128, 128, 3). However, it is not the only allowed input shape. A normalization process needs to be done before starting the analysis. The output layer

is activated when the neural network perceives the *Sigmoid* function. The outcome is a predicted mask of the forestry zone assessed. The accuracy is used as the final metrics' assessment.

For this example, the packages are summarized in Table 5.10.

Table 5.10: Summary of the packages used

Package	Purpose
OpenCV	Image editor
TensorFlow	Backbone for neural network
NumPy	Array management
Keras	Neural network architecture API
Matplotlib	Plotting utility

One of the weaknesses of this exercise is the dataset. The images used are in low-quality, and the pixel study can be affected. Moreover, for encouraging better results, three different datasets are used. The first one contains 30 images for training, 10 for testing and 10 for validation. The second one is composed of 6 images for training, 2 for testing ad 2 for validation. In the last one, it is possible to find 8 images for training, one for testing and one for validation. However, there are no significant differences between the three, because, in the second and third one, only the best images are taken and grouped by similarity. Every dataset member has a mask associated. For two datasets the application *MATLAB image- Segmenter* is used. In the third one *Adobe Photoshop* is used instead, in order to increase the quality of the input feed.

The data-point of the dataset is loaded into two different batches. The first one contains the images, and the second one the masks. Therefore, for each element of the X image, a Y mask is associated:

$$X_{image[i]} = Y_{mask[i]}$$

Before delivering the image to the neural network, every member of the dataset is resized, and the mask is converted into greyscale. The final input content has the following characteristics:

$$X_{image[i]} = \left(\left(batch_{number}, image_{width}, image_{height}, image_{channel}\right), dtype = np.float32\right)$$

$$Y_{mask[i]} = \left(\left(batch_{number}, mask_{width}, mask_{height}, mask_{channel}\right), dtype = [0,1]\right)$$

At this point, the model is ready for training, generating weights that are saved for every dataset, in order to keep truthful results. The model assumes the structure summarized in the Figure 5.33.

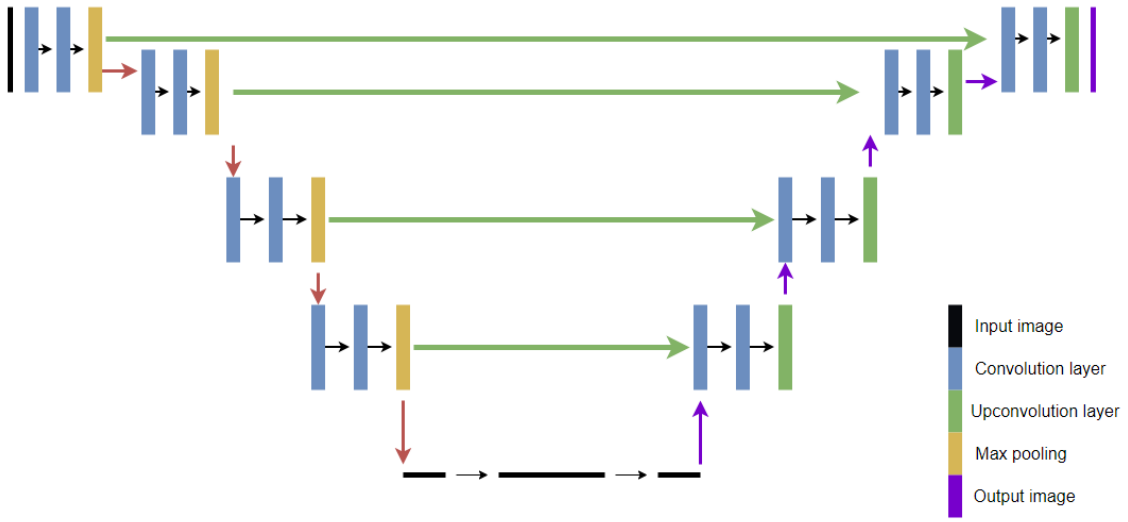


Figure 5.33: Description of the U-Net model

The prediction is performed through an utility function. After calling the method of the *Keras API*, the predicted mask is saved into a variable holder, and it is printed on the screen. In order to remove the noise and false positive from the outcome, a threshold that eliminates the hits with lower accuracy is put in place. The coloured mask is converted into black and white, where the first colour shows the background and the second displays the predicted areas. At this point, to increase the visibility of the outcome, the software draws bounding boxes overlaying the original image. This process does assessments by using an algorithm provided by the OpenCV framework.

Moreover, the OpenCV algorithm can detect contours discovering edges and lines into the mask, returning the possible coordinates of the bounding boxes. However, this process is not very pleasant, producing plenty of false-positive hits. The results are in the Figure 5.34.
The U-net is capable of detecting damage from the forestry. It is noticeable that the sea around the Kvarken archipelago and the towns do not generate false positives, increasing the reliability of this neural network. The model produces an accuracy in a range between 89% and 95% depending on the quality of the proposed data.

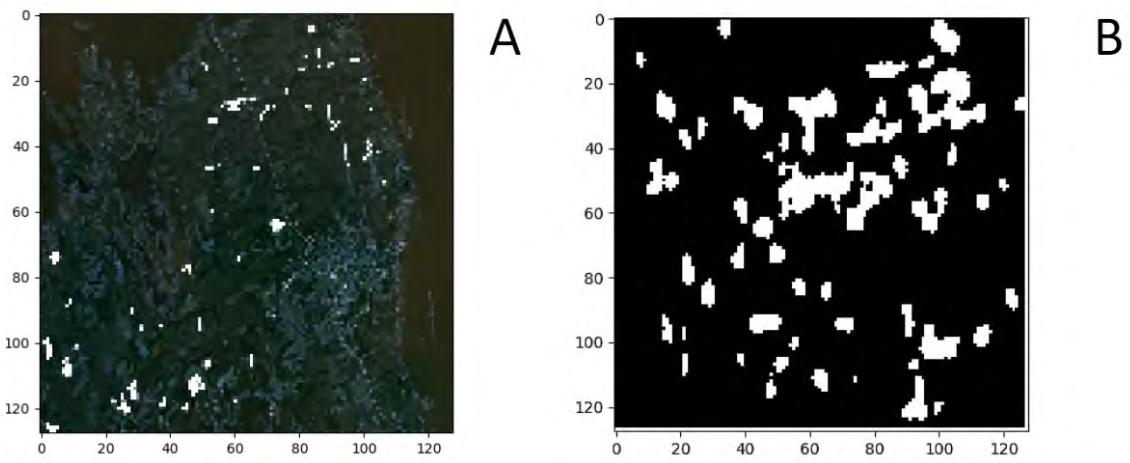


Figure 5.34: Test image comprising deforested areas (mask is in white); (B) Mask prediction of the deforested areas by U-net

In addition, 26 images are downloaded from the *ArcGIS_Desktop* package. These satellite imagery data are in high-resolution with dimensions of 9984 pixels in width and height. Moreover, the data appears more detailed and zoomed-in comparing to the previous sets. The elements are masked using MATLAB within the same process described above. For every test, one image is taken away from the dataset in order to make accurate predictions. It is not good practice to use the same image for prediction after it has been used for testing or training. The dataset goes into two containers, one for the images and one for the masks and it is split using *Sklearn* Python package as follow:

$$X\_train, X\_test, Y\_train, Y\_test = train\_test\_split\,(X\_container, Y\_container, test\_size = 0.33)$$

where X_train, X_test, $,Y\_train$, Y_test are the batches that contain the actual data, $test\_size$ is the split ratio and $shuffle$ is set to false in order to take the images and the masks in the same order avoiding confusion.

The model produces accurate results. However, the evaluation always has accuracy between 80% and 86%. In order to show the detection zones, a function provided by OpenCV framework is used. Moreover, it is able to draw contour using an algorithm that calculates the distances between various points and edges of the predicted mask. Some resulted samples are reported in Figure 5.35 to Figure 5.37.
In general, this model meets the expectation of an excellent instrument for remote sensing and damage assessment. However, the usage of a more specialized dataset with many high-resolution images and mask is crucial. The usage of a more accurate dataset can

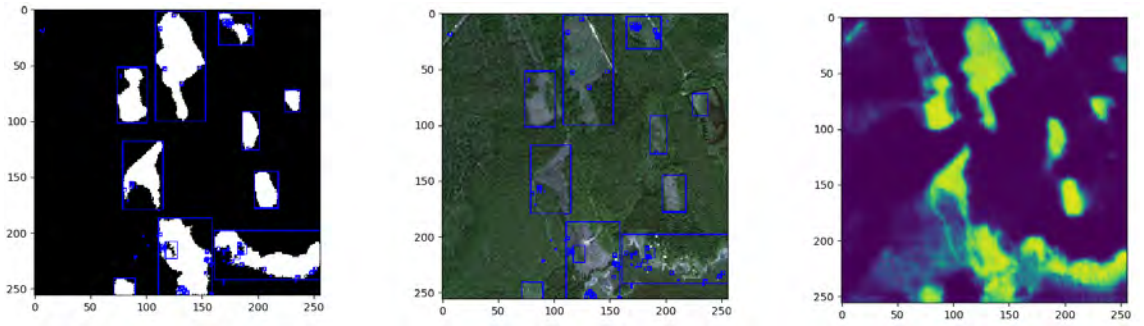


Figure 5.35: In the first sub-image, it is possible to discover the predicted contours of the image. In this example, the areas without trees are obvious. However, this sample is without plenty of water and houses. In the third image, there is the original predicted mask.

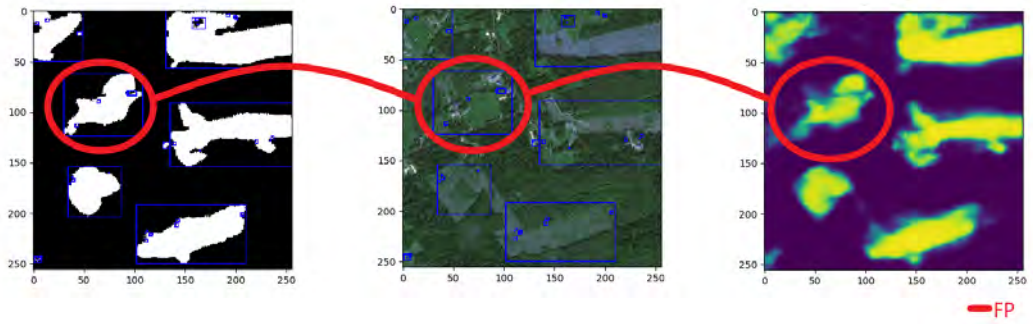


Figure 5.36: This prediction is fascinating because a significant false positive is detected. The system did not distinguish houses into the zone highlighted in red. The problem is that near the houses there is a damaged area and the algorithm has merged them.

Figure 5.37: In this example, the algorithm did not detect the sea, making a very interesting prediction of the damaged area.

increment the accuracy of the whole model and also the detection of the area of damage.

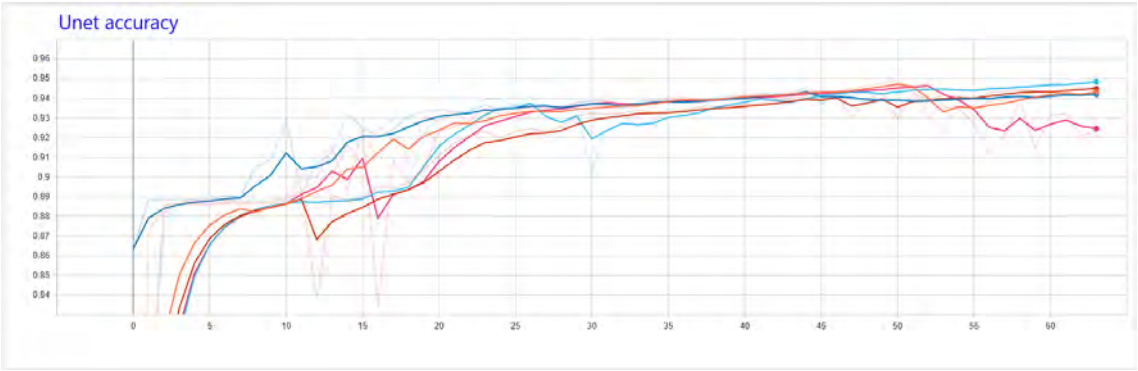


Figure 5.38: In this graph are reported the values of accuracy in 5 runs.

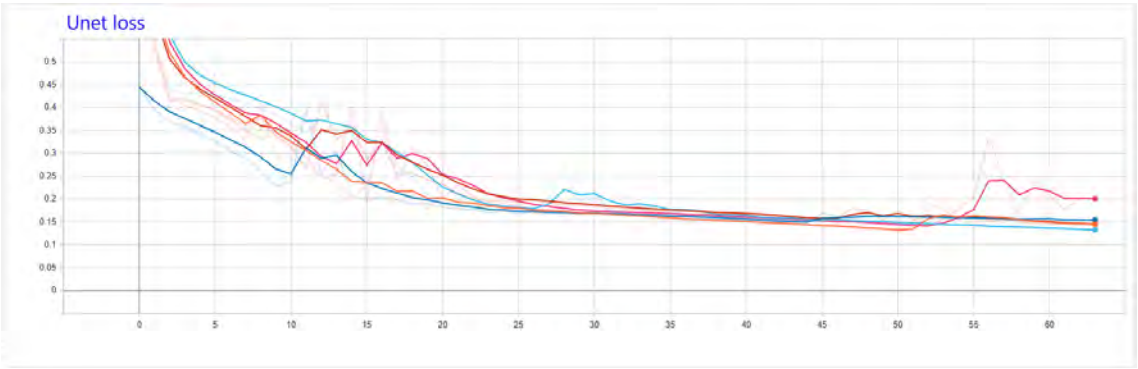


Figure 5.39: In this graph are reported the values of loss in 5 runs.

Table 5.11: Values of accuracy and loss

Run	Accuracy	Loss
Run 1	0.9446	0.1421
Run 2	0.9443	0.1540
Run 3	0.9457	0.1430
Run 4	0.9496	0.1298
Run 5	0.9228	0.1996
Average	0.9414	0.1921

5.3.3 SegNet

In the following example, the usage of *SegNet neural network* is described. The model has two different branches, the *encoding layer* and the *decoding layer*. Each of that has four blocks. Into the encoder, there is the input layer; into the decoder, there is the output layer. The architecture uses the *sequential* API of Keras framework. It achieves excellent results in pixel-wise segmentation. The model synthesis is presented in the Figure 5.40.

The elements of the encoder block are formed by a *Zero-padding layer* followed by a *Convolutional layer*, a *Batch normalization* and a *Max pooling layer*. The neurons activation occurs with the Rectifier Linear Unit. Moreover, the scope of the Zero-padding layer

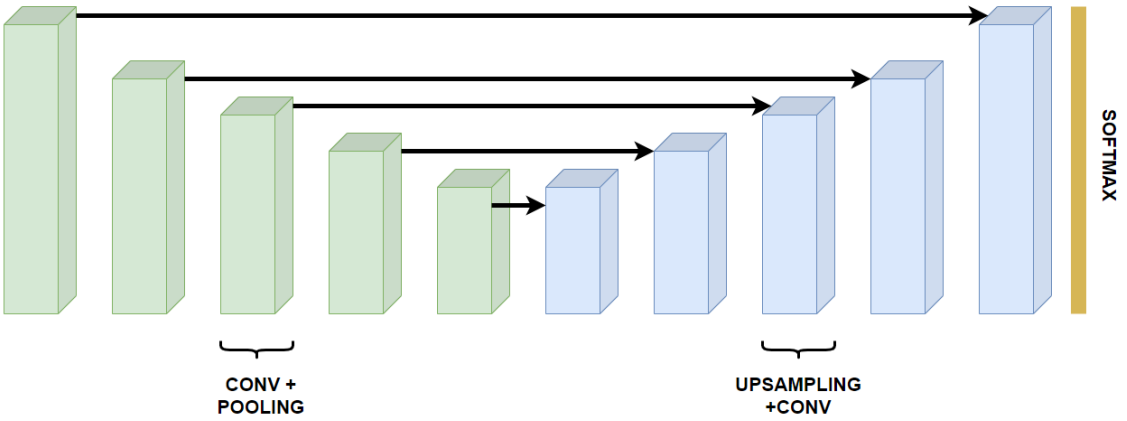


Figure 5.40: Description of the SegNet model.

is to add zeros right before the image's contours helping the Convolutional layers to use shared weights to discover features. The number of neurons increases from a minimum of 56 to a maximum of 512.

However, from layer to layer, the discovered features become small (down-sampled) in order to study neighbouring pixels. Opposite, in the decoder layer, the features' map are up-sampled and normalized. This process stops when the predicted mask assumes the original input dimension. The outcome is provided by $Softmax$ activation returning the predicted mask of the input image.

For the purpose of this example the packages named in Table 5.12 are used.

Table 5.12: Summary of the packages used for the SegNet implementation.

Package	Purpose
OpenCV	Image editor
TensorFlow	Backbone for neural network
NumPy	Array management
Keras	Neural Network architecture API
Matplotlib	Plotting utility

The same low-resolution datasets from the previous section are used to feed this model. However, with SegNet, the input data is sequential. The input function passes to the neural network only one combination of image and mask per time. In order to accomplish this prerequisite, a special class from the Keras framework is used, the *ImageDataGenerator*. Moreover, this utility can read the input from various folders without using an extra set of functions like OpenCV. Resizing and normalization are automatic. The image generator can generate new images. The process is simple. The original image is flipped,

rotated and shifted in order to augment the original dataset. However, this process is very computationally expensive and requires plenty of computational power. The same process using only CPU (Intel(R) Core(TM) i7-7700HQ 2.80 GHz) and 16 Gigabytes of RAM (with a frequency of 2100 MHz) requires between two and three hours. When the model is compiled *SegNet* requires a special function called $fit\_generator$ that takes as argument the train image generator and the validation generator, both created with the Keras *ImageDataGenerator*, described above. The produced masks are summarized in Figure 5.41.

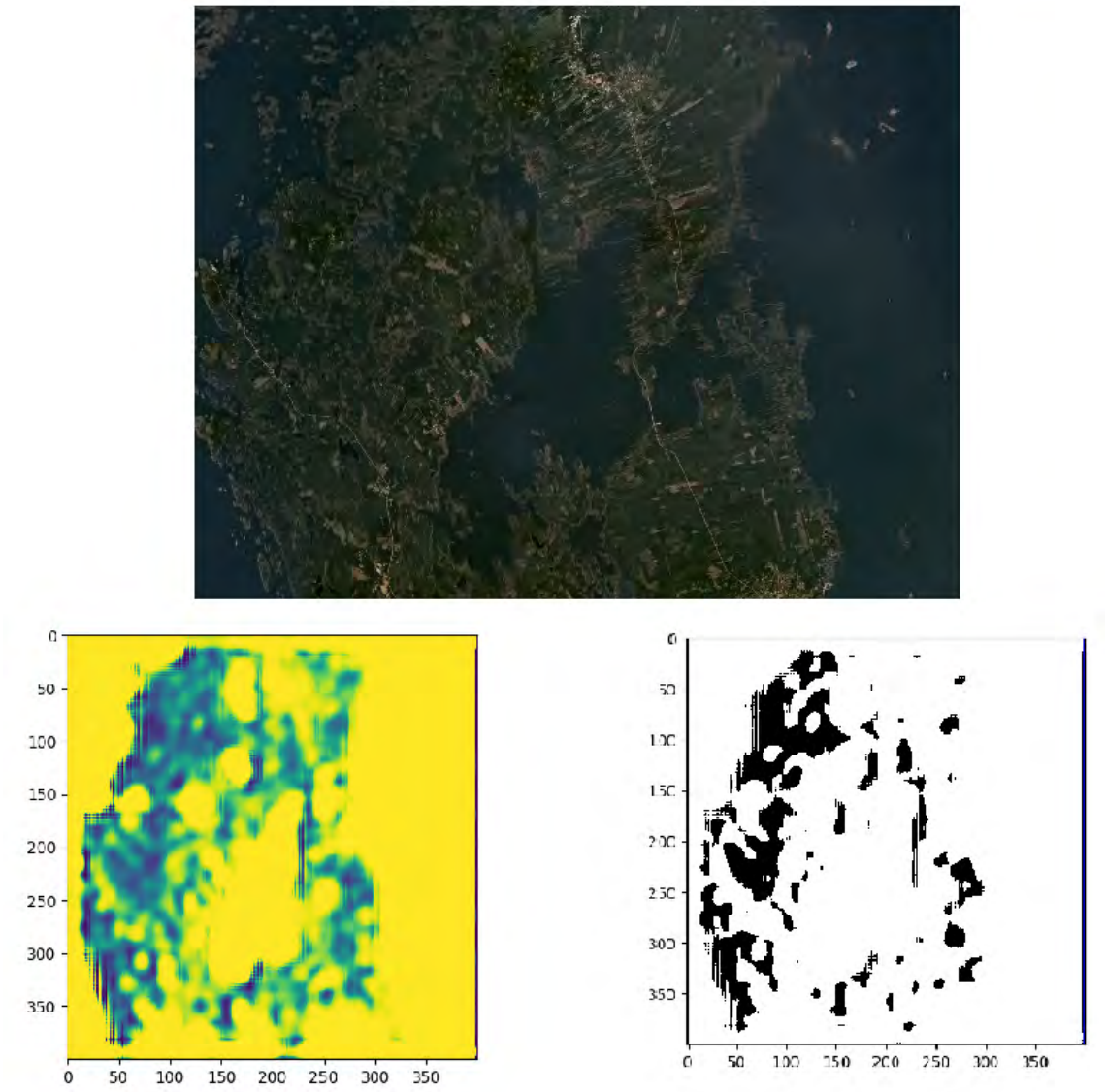


Figure 5.41: Results of the SegNet prediction masks. The upper photo shows the Kvarken Region. In the lower-left image, the predicted mask of the region is shown. On the right is displayed the predicted zones with a probability percentage greater than 0.5.

Using the dataset provided by ArcGIS Desktop the results are summarized in Figure

5.42.

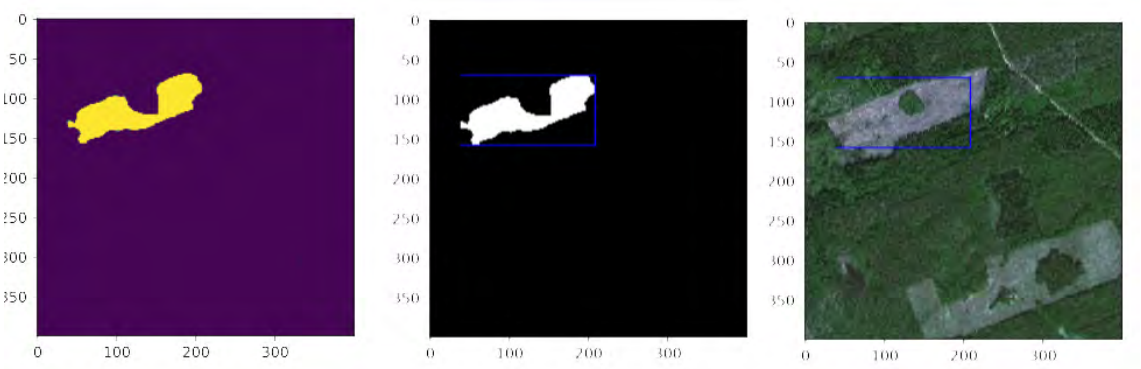


Figure 5.42: Results of the SegNet prediction masks. The first images shows the predicted mask. The second one shows the contours of the prediction, and in the third one it is possible to find the final results. The results show some improvements, but they are not great.

Studying the results, the U-Net produces superior outcomes with better accuracy (about 96%), as well. The SegNet model captures some damaged area, but contrary to the U-net, the significant zones detected are the sea and the land without underlining the interesting forestry areas.

CONCLUSION AND FUTURE WORK

6.1 Summary and conclusion

In this work, several machine learning methods are implemented, both supervised and unsupervised. They constitute a suitable methodology to assess the environmental damage of satellite images. The case study is the Kvarken Archipelago, Finland.

GIS systems, such as SNAP, ArcGIS and QGIS provide an important initial step for assessing forestry area. They contain a set of tools and utilities for land classification. Moreover, Google Earth Engine offers the same kind of analysis with more extensive datasets that are updated almost every day. The SNAP RGB analysis makes possible to gather the status of the healthy vegetation in the targeted areas within just a few clicks. ArcGIS provides a very good set of high-resolutions images that are essential for the neural network.

The unsupervised models presented here show that the K-Means cluster algorithm can highlight the damaged area giving only the number of clusters as the initial parameter. K-Means achieved good results delineating water, land, urban areas, active and barren vegetation. However, the method does not show brilliant capabilities when quantifying the density of trees and their dispersion. Therefore, other sophisticated solutions need to be considered. Self-Organizing Maps are handy for this purpose. The particularity of this model is the study of pixels' density in the image. The outcome shows that the algorithm can segment the interested areas, highlighting the places with more trees, deforested areas, urban areas and water. In addition, using the high-resolution images, the outcome shows more prominently roads and urban areas, suggesting further improvements of the whole quality of the output.

The Principal Component Analysis, PCA, is very useful it can remove the noise in the initial input. Eight bands are analyzed showing that is possible to rebuilt a clearer image from a set of bands considering only its Principal Component. However, with low-quality

input, the final output is not satisfying.

The deep learning models used here show that the input image resolution can bias the outcome of the learning process. Also manually labelling deforested areas on space-borne images is tedious. However, finding high-resolution images, the process of labelling becomes simpler, consequently, the results are improved.

I trained the SegNet architecture on the low-resolution dataset with unsatisfactory results. Subsequently, I trained the U-Net on two different datasets collected from the Kvarken Area. The simulation shows that the U-Net performs significantly better on the dataset with high-resolution imagery, delineating deforested areas with good precision. A quality assessment can be found in Figures 5.35, 5.36 and 5.37.

6.2 Future work

The improvement of the dataset is crucial in order to retrieve better information for all the analysis proposed in Chapter 4. Moreover, incrementing the quality of the image and dividing it in multiple tiles is another key point to obtain better performances. Further research needs to be proposed. For example, a neural network that provides a trees census of the area is good. It enriches the quantity of information that is used for further analysis. This work is achieved, making a comparison between the trees in summer when they all have leaves, and in autumn when they are bare. Therefore, it is possible to study the forestry season by season underlining the differences. However, the creation of a temporal dataset which contains ground-truth is necessary.

Filtering data on days when bad weather has occurred is another important step to study. Moreover, it is possible to create a script where the images are passed as input to the Neural Network only when bad weather occurs by consulting, for example, the Finnish weather archive.

References

[1] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. The MIT press Cambridge, Massachusetts, 2016, pp. 1–18.

[2] D. Kumar, "Basic concept of remote sensing," 2014. [Online]. Available: https://nptel.ac.in/courses/105108077/ (visited on 10/07/2019).

[3] B. Surwekiha, R. T., and D. Nilanjan, *Satellite Image Analysis: Clustering and Classification.* Springer, 2019, pp. 1–2.

[4] Z. Yong-Liang and L. Huan-Zhang, *Vision system for satellite observation in close quarters.* 2008. [Online]. Available: https://ieeexplore.ieee.org/document/4659919 (visited on 10/15/2019).

[5] M. Teke and T. A., "Multispectral satellite image registration using scale restricted surf.," 2010. [Online]. Available: https://ieeexplore.ieee.org/document/5595974 (visited on 10/01/2019).

[6] A. Du, P. Liu, S. L. J., and C. L., *Ensemble extreme learning machines for hyperspectral image classification*, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6732910 (visited on 10/07/2019).

[7] A. Bhaat, *Automated change detection in satellite images using machine learning algorithm for delhi, india.* 2008. [Online]. Available: https://ieeexplore.ieee.org/document/7326109 (visited on 10/09/2019).

[8] A. Goshtasby, C. Stockman, and V. Carl, *A region-based approach to digital image registration with subpixel accuracy.* 1986. [Online]. Available: https://ieeexplore.ieee.org/document/4072476 (visited on 10/15/2019).

[9] S. Fobi, "Blurry?! using deep learning to harmonize satellite imagery across resolutions.," 2019. [Online]. Available: https://medium.com/descarteslabs-team/blurry-no-more-using-deep-learning-to-harmonize-satellite-imagery-across-resolutions-b1fe46c7f8cc (visited on 11/04/2019).

[10] D. Valsesia, "Enhancing satellite imagery with deep multi-temporal super-resolution.," 2019. [Online]. Available: https://towardsdatascience.com/enhancing-satellite-imagery-with-deep-multi-temporal-super-resolution-24f08586ada0 (visited on 10/30/2019).

[11] A. Ojala, *Quaternary studies in the northern and arctic regions of finland.* 2005. [Online]. Available: http://tupa.gtk.fi/julkaisu/specialpaper/sp_040.pdf (visited on 10/30/2019).

[12] K. Svels, *World heritage, tourism and community involvement: A comparative study of high coast (sweden) and kvarken archipelago (finland).* 2015. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/15022250.2015.1009708?src=recsys (visited on 10/30/2019).

[13] P. Berg, H. Syrjälä, and P. Laaksonen, *Natural uniqueness and sustainable tourism business. small tourism enterprises in the finnish kvarken archipelago world natural heritage site*, 2014. [Online]. Available: https://www.cairn.info/revue-management-et-avenir-2014-3-page-187.htm# (visited on 10/30/2019).

[14] N. Gorelick, M. Hancher, M. Dixon, S. Illyushenko, D. Thau, and R. Moore, *Google earth engine: Planetary-scale geospatial analysis for everyone*, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425717302900 (visited on 10/21/2019).

[15] N. Patel, E. Angiuli, P. Gamba, A. Gaughan, G. Lisini, F. Stevens, and A. T. G. Trianni, *Multitemporal settlement and population mapping from landsat using google earth engine*, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0303243414001998 (visited on 10/21/2019).

[16] K. Johansen, S. Phinn, and M. Taylor, *Mapping woody vegetation clearing in queensland, australia from landsat imagery using the google earth engine*, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352938515000051 (visited on 10/21/2019).

[17] L. Veci, J. Lu, M. Foumelis, and M. Engfdahl, *Esa's multi-mission sentinel-1 toolbox*, 2017. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2017EGUGA..1919398V/abstract (visited on 10/23/2019).

[18] M. Foumelis, J. Blasco, Y. Desnos, M. Endahl, D. Fernandez, L. Veci, and C. Wong, *Snap – stamps integrated processing for sentinel-1 persistent scattered interferometry*, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8519545 (visited on 10/23/2019).

[19] C. Storie, *Urban boundary mapping using sentinel-1a sar data*, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8519051 (visited on 10/23/2019).

[20] S. Ullo, A. C., L. Cicala, N. Fiscante, P. Addabbo, M. D. Rosso, and A. Sebasttianelli, *Sar interferometry with open sentinel-1 data for environmental measurements: The case pf ischia earthquake*, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8385270 (visited on 10/23/2019).

[21] J. Manjarrez and L. Ross, *Geographical information system (gis) environmental models for aquaculture development in sinaloa state, mexico.* 1995. [Online]. Available: https://link.springer.com/article/10.1007/BF00117877 (visited on 10/25/2019).

[22] J. Kokina and T. H. Davenport, "The emergence of artificial intelligence: How automation is changing auditing," *Journal of Emerging Technologies in Accounting*, vol. 14, no. 1, pp. 115–122, 2017. [Online]. Available: https://www.nber.org/papers/w24196.

[23] G. Litjens, C. I. Sánchez, N. Timofeeva, M. Hermsen, I. Nagtegaal, I. Kovacs, C. Hulsbergen-Van De Kaa, P. Bult, B. Van Ginneken, and J. Van Der Laak, "Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis," *Scientific reports*, vol. 6, p. 26 286, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417415007101.

[24] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015. [Online]. Available: https://arxiv.org/abs/1502.00873.

[25] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, ACM, 2018, pp. 303–314. [Online]. Available: https://dl.acm.org/doi/10.1145/3180155.3180220.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT press Cambridge, Massachusetts, 2016, pp. 1–18.

[27] B. Ramsundar and R. Zadeh, *TensorFlow for Deep Learning*. O' Reilly, 2018.

[28] G. Aurelien, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly, 2017, pp. 37–38.

[29] A. Pantel, *Hands-On Unsupervised Learning Using Python*. O'Reilly, 2019.

[30] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320302000602.

[31] J. Estornell, J. M. Martí-Gavliá, M. T. Sebastiá, and J. Mengual, "Principal component analysis applied to remote sensing," *Modelling in Science Education and Learning*, vol. 6, pp. 83–89, 2013. [Online]. Available: https://polipapers.upv.es/index.php/MSEL/article/view/1905.

[32] T. Kohonen, *The self organizing maps*, 1990. [Online]. Available: https://sci2s.ugr.es/keel/pdf/algorithm/articulo/1990-Kohonen-PIEEE.pdf (visited on 12/17/2019).

[33] A. Khazri, *Self organizing maps*, 2019. [Online]. Available: https://towardsdatascience.com/self-organizing-maps-1b7d2a84e065 (visited on 12/17/2019).

[34] E. Berglund and J. Sitte, "The parameterless self-organizing map algorithm," *IEEE Transactions on neural networks*, vol. 17, no. 2, pp. 305–316, 2006. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1603618.

[35] A. Gulli and S. Pal, *Deep learning with Keras: implementing deep learning models and neural network with python*. Packt, 2017, pp. 1–106.

[36] S. Ravichandiran, *Hands-On Deep Learning Algorithm with Python: Master Deep Learning algorithms with extensive math by implementing them using tensorflow*. Packt, 2017, pp. 1–50.

[37] R. Atienza, *Advanced Deep Learning with Keras*. Packt, 2017.

[38] M. C. Hansen, P. V. Potapov, R. Moore, M. Hanchen, S. A. Turubanova, A. Tyukavina, D. Thau, S. V. Stehman, S. J. Goetz, T. R. Loveland, A. Kommareddy, A. Egorov, L. Chini, C. O. Justice, and J. R. G. Townshend, *High-resolution global maps of 21st-century forest cover change.* 2013. [Online]. Available: http://earthenginepartners.appspot.com/science-2013-global-forest (visited on 11/2019).

[39] S. Kumar, *Principal component analysis: In-depth understanding throught image visualization*, 2019. [Online]. Available: https://towardsdatascience.com/principal-component-analysis-in-depth-understanding-through-image-visualization-892922f77d9f (visited on 12/16/2019).

[40] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, S. Fu, and C. Berg, "Single shot multibox detector," *Europena conferences in computer vision*, vol. 17, no. 2, pp. 21–37, 2016. [Online]. Available: https://arxiv.org/abs/1512.02325.

[41] P. Isola, J. Zhu, T. Zhou, and A. Efros, "Image to image translation with conditional adversarial networks," *ArXiv*, vol. 17, no. 2, pp. 1–17, 2018. [Online]. Available: https://arxiv.org/pdf/1611.07004.pdf.

[42] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, *et al.*, "U-net: Deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, no. 1, pp. 67–70, 2019. [Online]. Available: https://www.nature.com/articles/s41592-018-0261-2.

[43] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, "Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation," *arXiv preprint arXiv:1802.06955*, 2018. [Online]. Available: https://arxiv.org/abs/1802.06955.

[44] Y. Xu, L. Wu, Z. Xie, and Z. Chen, "Building extraction in very high resolution remote sensing imagery using deep learning and guided filters," *Remote Sensing*, vol. 10, no. 1, p. 144, 2018.

[45] M. Price, *Mastering ASrcGIS*. Mc Graw Hill Education, 2019, pp. 43–57.