POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Remote sensing-based vineyard image segmentation with deep computer vision for precision agriculture

Supervisors

Prof. Marcello CHIABERGE

Francesco SALVETTI

Simone ANGARANO

Candidate

Cesar Andres SIERRA PARDO

July 2022

Abstract

The incorporation of remote sensing in precision agriculture technologies has greatly helped to optimize the productivity and efficiency of both farming and agricultural production processes, especially in applications where the monitoring of an extensive area of land is required and the use of automation and robotics can help reduce the time and costs of it.

The inspection of vineyard fields is one of the precision agriculture applications where remote sensing and deep learning techniques are combined into systems that can, autonomously, assess the state of the crop. Recent research incorporates satellite imagery as well as drone-captured images as inputs for Convolutional Neural Networks (CNN) capable of quickly processing them to output the information relevant to the specific application, ranging from the growth state of the plantation to a list of commands that an unmanned ground vehicle can use to move through the field. For both cases, a clear understanding of where the vineyard rows are located is crucial, so a system in charge of the vine identification is needed.

This thesis aims to present a deep learning approach to solve the problem of segmenting vineyard rows in remotely sensed RGB images. The proposed goal has been achieved by implementing two different CNN that allow comparing traditional and cutting-edge architecture performances. Since, contrary to most studies in the field, that focus on particular cases, generality is a desired characteristic, a dataset consisting of aerial vineyard images of different grape varieties from several wine-growing regions was gathered to account for variable factors such as the illumination condition, the resolution of the images or the growth stage of the crop. Finally, the proposed solutions have been tested with images describing different scenarios with good results, for which a qualitative and a quantitative comparison is done. However, several issues can be further addressed to increase the model efficiency and performance, making this topic interesting for future work development.

Acknowledgements

I would like to thank my supervisor prof. Marcello Chiaberge for the oportunity to collaborate with PIC4SeR, for his patience and support. I give my gratitude as well to Francesco, Simone, Vittorio and Gianluca for the guidance and advice they gave me during the development of this work.

Special thanks to my family as well, for their full emotional and economic support and always believing in my capacities.

To Delia, who has been like a lighthouse to me since the day I met her, and Marco, for accompanying me through many classes and exams. To Sergio and Mayra, my adoptive parents. To my bro Yithzak, for the countless laughs and life lessons. Finally, to Sara, for her precious company and for understanding how to motivate me and push me to work, I wouldn't have gotten to this point without you.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
Artificial Intelligence

ML
Machine Learning

DL
Deep Learning

GPS
Global Positioning System

PA
Precision Agriculture

GPU
Graphics Processing Unit

TPU
Tensor Processing Unit

OS
Operating System

UAV
Unmanned Aerial Vehicle

UGV
Unmanned Ground Vechicle

GNSS

Global Navigation Satellite System

Chapter 1

Introduction

A need of finding new ways to think about agriculture is being created because of the fast growth of the world population. With the food request increasing constantly, also the need of improving the production efficiency and lowering costs does. In the agricultural field, many technologies are under development, for instance, drones and robots are involved in many of them. The future for food production, including harvesting, crop monitoring and planting will inevitably involve service robotics to be applied to smart agriculture.

There are already many promising ideas, like using satellite images to constantly and automatically monitoring fields, and through computer vision, observe the field parameters. A technology with these capabilities would indeed save costs, however it is currently limited by the low resolution of free satellites. A common alternative is to use drone flights instead. The inclusion of specialized sensors and cameras in drones allows a more precise monitoring of a certain terrain area[1].

Another technology that is highly requested in recent smart agriculture is the autonomous navigation of unmanned ground robots, to provide them with the capability of monitoring and harvesting crops with minimal to no human intervention. For this purpose, cameras at ground level are included in such vehicles. To further enhance the quality of the autonomous navigation, the information obtained from, the ground level cameras can be merged with images coming from an unmanned aerial vehicle, usually equipped with an RGB-D camera.

This work can be related with both autonomous navigation and crop monitoring and aims at segmenting vineyard fields from aerial orthoimagery, that consist on only RGB channels, using deep learning methods. The work was developed at the PoliTO Interdepartamental Centre for Service Robotics (PIC4SeR), where it can be used in several agricultural related projects.

Figure 1.1: Unmanned Ground Vehicle for autonomous navigation

1.1 Starting point and objective of the thesis

This thesis is a work which takes direct contribution from two other works performed at PIC4SeR. The first of them, by Simone Cerrato, Vittorio Mazzia, Francesco Salvetti and Marcello Chiaberge [2] titled "A Deep Learning Driven Algorithmic Pipeline for Autonomous Navigation in Row-Based Crops" presented a complete algorithmic pipeline for autonomous navigation in row-based crops, be it vineyards or other types of crops , mainly based on information obtained with low-range sensors and cameras both at ground and in aerial points. The main steps are described in figure 1.3, them being the waypoints estimation, the path planning based on the waypoints, the segmentation of the vineyard rows, fundamental to the waypoints estimations, and the control scheme to be given to an unmanned ground vehicle (UGV). Figure 1.2 shows an example of how this should work.

The second main work was carried by Vittorio Mazzia, Francesco Salvetti, Diego Aghi and Marcello Chiaberge [3] under the name "DeepWay: a Deep Learning Waypoint Estimator for Global Path Generation". This work is at the heart of the waypoints estimation that [2] uses. In this work, a specific neural network architecture , DeepWay, is presented with the goal of estimating waypoints in occupancy maps whose 'obstacles' are parallel lines. To train them, artificial images were employed, and to test the efficacy of the neural network, a dataset of 100 manually segmented aerial images of vineyards was used.

The main goal of this thesis is to present a deep learning approach to vineyard segmentation of aerial images, capable of predicting occupancy maps that could be used for improving the results of these previous works. The second objective is to develop a larger dataset of aerial orthoimages of vineyards, so that it could be used for training or testing in future works. The a priori goal number of samples is set to 1000, ten times bigger than the initial dataset. It is important to mention, that even if autonomous navigation is the main drive for this work, land monitoring is another field where vineyard segmentation can be of use.

Figure 1.2: Aerial view of a vineyard. The estimated waypoints are shown in yellow, the global path in red and the occupancy grid (segmented vineyard) in black. They allow a UGV to move along the vinerows.

1.2 Organization of the thesis

This thesis is organized in 8 chapters. Chapter 1 presented a motivation of the thesis as well as the starting point.
The background knowledge and context needed to understand the work is explained in chapter 2.
Chapter 3 presents a thorough analysis of the state of the art of the techniques used for vineyard image segmentation as well as the main applications where its being used nowadays.
In chapter 4, the process of construction of the newly presented dataset is described, along with the criteria used to chose the images.
Chapter 5 is dedicated to present the architecture of the deep neural networks that are the core of the thesis.
In chapter 6, detailed explanation of the the software and hardware used is provided.
The experimental testing and the results are described in chapter 7, where an analysis of both is also presented.

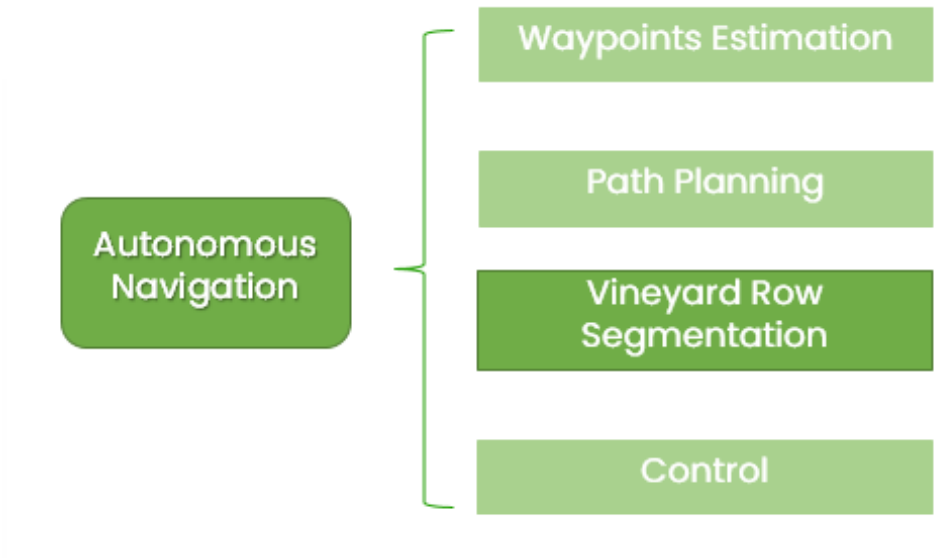

Figure 1.3: Main steps of the pipeline for autonomous navigation. The principal interest is the vineyard row segmentation

Finally, in chapter 8 the conclusions are gathered and the future work is proposed.

Chapter 2

Background

The main focus point in which this thesis is based on are remote sensing, precision agriculture and deep learning. They can also be considered as the most relevant keywords. In this section, an introduction to the most relevant concepts of each of these fields is presented, to put in context the work performed, as well as the state of the art in later chapters.

2.1 Remote Sensing

The technology by which properties of specified objects, or phenomenon can be measured, identifies and analyzed without direct contact with them is called remote sensing. Land-use mapping, enviromental study or weather forecasting are only some of the examples of the applications where remote sensing plays a major role.

Remote sensing allows to collect data from inaccessible or dangerous areas, growing in relevance in nowadays society. Since it provides fast and repetitive coverage of extremely large areas, it can replace slower and costly data collection ground processes for everyday applications, be it reports on climate change or even natural disasters.

It is also considered unobstructive, so the data processing and collection, as well as the geographic information system (GIS) analysis offsite, meaning that delicate objects, targets or areas will not be disturbed. Some specific examples of these are the monitoring of polar bears, chemical concentrations, earthquakes, floods, forest fires or deforestation. In such cases, geospatial remote sensing is a way of obtaining otherwise unattainable global perspectives

2.1.1 Principles

The main principle of remote sensing is sampling the emitted and reflected electromagnetic radiation that comes from the earth's atmospheric, aquatic and terrestrial ecosystems, all to monitor and detect the physical characteristics of the sensed area without the need of making physical contact [4]. To achieve this kind of data collection, typically aircraft-based or satellite-based systems, sensors technologies are involved, being classified as active or passive sensors.

Active Sensors

Active sensors use internal stimuli for the collection of the data. They emit energy to scan areas where a sensor measures the energy that the target reflects. Two typical sensing tools are RADAR and LiDAR. They are capable of measuring the delay of time that happened from the emission until the return of the signal from the objective, determining the location, direction and speed that an object has. This remote sensing data is gathered, then processed and possibly analyzed with remote sensing hardware and software.

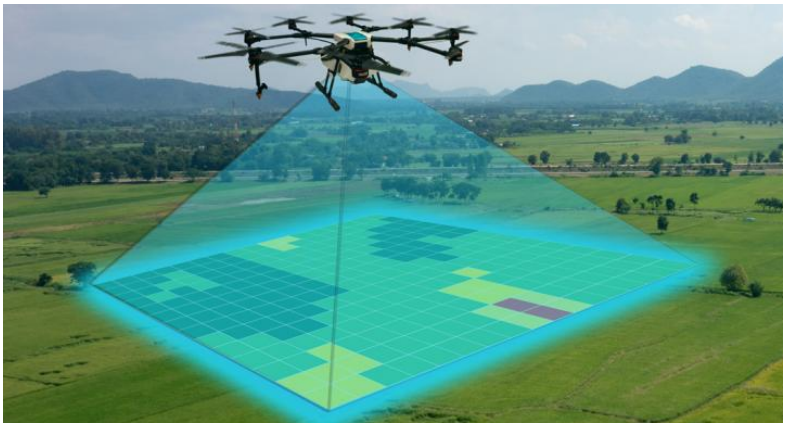

Figure 2.1: Drone-based remote sensing

Passive Sensors

They respond to external stimuli, so they are only capable of gathering radiation that is emitted or reflected by an object or its surrounding space. The reflected sunlight is the most common source of radiation that a passive remote sensing device can have as input. Cheap and popular technologies of passive sensor are charge-coupled devices, cameras, radiometers and infrareds

2.1.2 Advantages of microwave remote sensing

Microwave remote sensing is the sensing of wavelengths ranging from one centimeter to one mete, and they can come from both active and passive remote sensing methods. Since the microwaves can penetrate rainfall, dust, haze, and cloud cover easily than infrared and visible wavelengths, it is a relevant sector. It means that the data can be gathered under most environmental conditions, because the microwave energy is not affected or susceptible to atmospheric scattering. This wavelength is optimal for applications like soil moisture mapping or ice monitoring

2.1.3 Challenges and shortcomings

Some shortcoming of remote sensed images taken with UAV or unmanned aircraft systems UAS need to be [5]. In the recent years, there has been seen an enormous growth of applications based on small UAS, that have limitation caused because the consumer-grade cameras that are not of sufficient quality. They are designed for other purposes and are not optimized for remote sensing applications. Specialized instruments can be too bulky to be incorporated in small UAV and for those instruments that are in fact portable by a small UAV, there is still a need for a precise calibration of the conventional sensors.

The limitations are not only because of the size of the sensors, but there are also important spectral limitations. The spectral curves of consumer-grade cameras are not calibrated correctly, making the coverage of brightness values to radiance, relevant for studies on comparatives. In any case, even sensors that are specifically made for UAS could not meet the scientific benchmarks if not well produced. High contrast can cause problems and detector saturation can arise with consumer grade cameras, like when an image covers at the same time a snow covered field and a dark forest. For some aplications, the lack of a NIR band is an important drawback, given that spetially on vegetation surveys, hidden information can be understand at this wavelenght

The geometry of not specialized cameras also present challenges, because it can be hard to obtain reliable calibrations even under ideal conditions. In retractable lens cameras this is a especially characteristic problem, because the focal lenght may be affected by extraneous factors, like dust. The small size of the image sensor can result in microscopic errors or misalignments. A high-end calibration is wasted in such a low cost device, since they dont last a long time, so instead a quick calibration is preferred, even if it causes inconsistent results. In any case, these cameras can give decent results depending on the aplication. It is also worth noting that in many cases, the low flying height of the UAV compensates for these undesired effects.

The issues relating to the camera geometry and image quality cannot be completley elimiinated if a simple RGB camera is being used, but several steps can be done to improve the final image. A good alternative to consumer-grade cameras, that is also light weight are the micro-four-thirds format cameras. Removing photos that are blurred, over exposed or saturates is another way of improve the general quality. A big difference can be achieved by this simple step. Many research is done to improve image quality of cheap cameras, and they involve ways like improving camera stabilization during flight

Some of the most common problems in remote sensed images taken from drones are:

- Saturated image
- Vignetting
- Chromatic aberration
- Mosaic blurring in overlap area
- Incorrect colour balancing
- Hotspots on mosaic due to bidirectional reflectance
- Relief displacement
- Mosaic gaps caused by incorrect orthorectification

In the case that a drone is equipped with a single GPS unit and more than one camera, an alignment problem is bound to happen, as shown in figure 2.2

2.1.4 Vegetation Remote Sensing and Convolutional Neural Networks

The concepts of precision agriculture and convolutional neural network will be presented later in this same chapter, but for now, it is only necessary to know that most of the recent research that involves both of them, is based on remote sensing. As a matter of fact, [6] perfomed a survey that yielded many interesting conclusions:

- The number of yearly publications on vegetation remote sensing indicate an increase in the number of studies, being triplicated from 2018 to 2020
- open access databases to do research and replicate results are highly demanded but still lacking. Having them would improve not only the efficiency of training, but also the transferability of pretrained models to new domains

Figure 2.2: Alignment error in vineyard images from the same survey but taken with different sensors

- Most CNN are trained in a supervised enviroment, however the effort of annotating data may be reduced by using weakly or semi-supervised learning, that can compensate for coarse annotations.

- The ammount of training data can greatly affect the overall accuracy, and increasing by 10 times the size of the dataset could improve even 10% the accuracy. In almost every case, increasing the dataset size improved the outcome, up to a certain point where the improvement rate seems to flatten

- The mean number of images in studies based on UAV related studies uses a median of 2795 reference observations. This number is bigger than the one of studies that use ground images, but smaller than the satellite based ones, probably because of the big size of satellite image, that allow to cover a really wide area.

- The most commonly used neural network for this kind of study based on aerial image is the traditional yet effective U-Net. More recent architectures are also used, but in general, cutting edge complex deep neural network are not normally used. One of the possible reasons is that they may require a lot of training data to have good predictions

- Contrary to the believe that CNN models are a black box, multiple approaches allow the visualization of a trained model, with its behaviour and the key

patterns on the decision making process. This feature visualization is essential to understand CNN. In distilling new knowledge is the greatest chance of these methods with regard to the interaction of vegetation and remote sensing images.

2.2 Precision agriculture

In global agriculture, Precision Agriculture (PA) is a term with significant relevance, and can no longer be considered a new concept[7]. It emerged in the United States decades ago, in the decade of 80's, and nowadays, many important conferences and workshops are held all around the world, while the first one happened in 1992.

The first formal definition of Precision Agriculture was formulated by the US House of Representatives in the late 90's as an "integrated information- and production-based" farming system. Designed to increase site-specific and whole farm, long term productivity, efficiency and profitability, while minimizing unintended wildlife and environment impacts.

This definition includes a big interval of interpretations and meaning, since is quite general. One of the classic examples of precision agriculture applied is the correct dosage of inputs like water, pesticides, fertilizers, etc. at the correct time to maximize yieldings and increase the productivity. Another very important field of Precision Agriculture is the reduction of environmental impacts, since a correct application of chemicals can benefit the entire crop cycle, including the crops, the soils and groundwater, if applied at the right amount at the right time. Recent precision agriculture makes use of robotics and innovative technologies to perform classical agriculture tasks in an automatic way, be it ploughing, harvesting, monitoring, all while reducing the human intervention, as shown in figure 2.3. Since it respects soils, crops and farmers, precision agriculture has become a big foundation in sustainable agriculture.

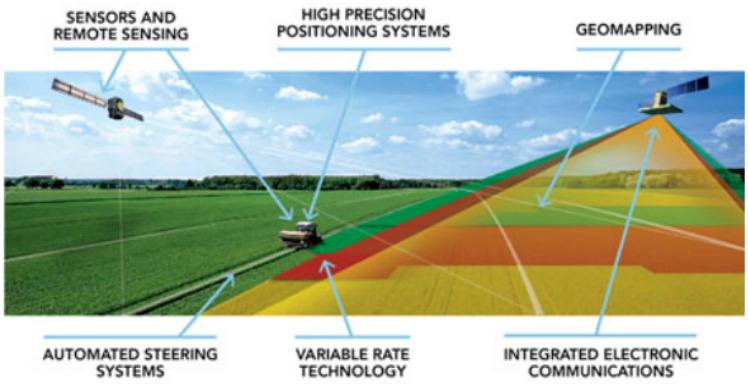

Figure 2.3: A precision agriculture application and all the technologies that can be involved

The need of precision agriculture arises from the unstoppable global population,

that is expected to increase by half the current amount in the next 30 years, making it so that double the food will need to be available. Each step into precision agriculture is thought with this goal in mind, and developing new farming techniques is urgent.

2.2.1 Enabling Technologies

One of the most important technologies that enable precision agriculture applications is the Global Positioning System (GPS), that forms part of the Global Navigation Satellite Systems (GNSS) [8]. By exploiting some correction algorithms, this technology allows farmers to precisely position a field, allowing the creation of maps with the many variables that can be measured. Some of the relevant parameter that are usually measured are pH, EC, nitrogen levels, moisture levels, crop yield, terrain features and topography or organic matter content. The information related to these variables are gathered by sensors that are mounted on GPS-enabled vehicles. This makes it so that every variable can be associated to a point in the planet. If different variables are measured at different times, they can be associated with the element at the same position the GPS indicates.

The use of recent real-time sensors directly in the ground, that can transmit data without human intervention are very relevant as well, since they reduce the time needed to go and manually monitor a small or big area.

The second enabling technology comes from Unmanned Aerial Vehicles (UAVs), that if equipped with multi-spectral or RGB cameras, can be a tool for remote sensing, that allows high quality imagery to optimize crop inputs with the creation of maps.

In addition to GNSS and UAVs, autonomous guidance is enabled on GPS-based tractors and other agricultural machinery, that are each time more independent and capable of of autonomously performing their own tasks, needing human intervention only to emergency cases. Internet of Thing is another relatively recent advancement that allows the collection and aggregation of a lot of data to the farmers, through a network of physical objects that send commands to IoT devices on very strategic and remote places of the farms. IoT is specially relevant for the welfare of animals. On top of all these technologies, machine learning algorithms are used in conjunction with autonomous vehicles, IoT devices and UAVs to study the collected data and have a more studied decision made.

2.2.2 Field Robotics

To put into perspective the the application, a brief description of field robotics is due. It is a wide branch of the robotic world, it deals with the automation of vehicles and platforms in complicated, dangerous, unstructured or harsh environments. This branch involves land, sea and air platforms to be automated and applied to mining, agriculture, cargo handling, planetary exploration, coastal surveillance, rescue, and many other applications[9]. In general, field robots are force to interact with the environment with no human supervision, since they are mobile plataforms that work outdoors.

Field robotics faces many challenges since it tackles a wide range of applications. Those with more relevance for this work are the ones related with presicion agriculture:

Localization issues

The problem of localization has been subject of a large amount of work. Thanks to the GPS, it has become more traceable, in different field robotic applications. In environments where good views of the sky is guaranteed, this is particularly true. Otherwise, the GPS does not work well. In the last years, low cost inertial sensing has been developed and it has allowed and represented an important advance for such applications, they have lead to combining GPS and inertial system with good results. Also, when the GPS localization is very poor, artificial landmarks or beacons are traced by lasers and radars lasers and radars to observe their relative location, referencing the objects to the maps, deducing their position. Natural landmarks use for relative navigation is reasonably well designed. Good examples includes delineating crop lines in agricultural application using artificial vision. Everything considered, this problem is by no means completely solved and depends on the specific robot being used for the specific problem being solved. In any case, it can be partially handled if many sensors are employed to acquire the position by different sources and principles so that they can be merged together to increase the information.

Perception issues

Full 3D perception and understanding on unstructured environments is probably the most complex and demanding research issue that field robotics faces. The problem of land-vehicle requires sensors such as vision and lasers to estimate correctly. The reliable construction of certain terrain is still far from precise, so the reconstruction is not very reliable. If new sensors or sensing strategies were to be developed, then the detecting of surface conditions, like loose gravel, ice or

mud, and the inferring of surfaces that are not fully visible, because rock or grass covers it would be possible. The estimation of more surface properties, such as the slip angle of the sand and the bearing strength of the dirt as well are a problem that could be of interest. The main issue is, in environments with little structure or model, to plan and make robust decisions.

2.2.3 The vineyard environmnet

A plantation of vine plants is called a vineyard. In vineyard, usually, parallel straight lines are used to arrange the vine plants. These conglomeration is denominated a vine row, and the are usually of the same length. Both the plants in the same vine row and different rows are place at a distance fixed from one another. This organization can be considered as a partially structured environment, as well as semi-structured environment, since the same skeleton is share among the vineyards, even if the vine plants can be different in terms of dimensions[10]. The main morphological properties that can make autonomous navigation on vineyards a challenging task are:

Figure 2.4: Different state of growth of the vine plants and of the grass in between the vine rows

- The terrain: The vineyard soil is mainly rough, bumpy and not the same consistency across the fields. Because of this, wheel slipping can be caused, and even be frequent and make a robot platform unable to properly move, making on-board measurements very noisy
- The vegetative state of the environment: The landscape of the vineyards can change a great deal depending on the season. In spring and summer, the vine plants are entirely covered in leaves, and the ground is also full of grass and weeds, while in the autumn and winter, it is mainly dry. The issue is caused because the exteroceptive sensors may have problems distinguishing

between lush vegetation, real obstacles and the real size of the vine rows, since the could seem enlarged to the robot eyes. Many path planning issues can arise because of this problem, because if the planner, that is trying to find a collision-free way among a large number of obstacles that are not really there but are only caused because of the soil vegetation, then it may not be possible.

- The crop type: the terrains that contain vineyards can be of different types depending on their slope, and there can be three main types:

  1. Mountainside crop: Where the vine plants are placed in slope terrain, with slopes higher than 40%
  2. Flat crop: Where the vineyard is planted on terrain that is mainly flat and the same sea-level is present across the whole field
  3. Hillside crop: Where the terrain of the vineyard has a slope, but this slope is smaller than 40%

In the case that the vineyards present slopes, there is still an important issue regarding the orientation on which the vine rows are distributed across the slope, since this factor can affect the battery consumption of a ground vehicle as well as other navigation conditions.

Figure 2.5: Disposition of vineyards parallel to the slope and perpendicular to the slope.

Even if there are many types of vines and grape used for many tasks as the production of wine, juice or just the harvesting of the fruit, the conditions regarding the agricultural process and construction of the fields are almost identical independently of these factors, and the shape of the vineyards is almost in general always that of straight parallel lines. The size of the fields, however, may vary depending on the available soil to cultivate. In some cases, the plantations could be particularly small if they are planted with recreational purposes and not for industrial production of food. Another important annotation to make is that vineyards and some other types of berries are very similar in shape, if we ignore the texture of the leaves and other properties only visible in the infrared range,

which means that in images where the texture of the leaf is not present, like in aerial images, plantations of vineyards and other types of crops may seem identical. For this reason, many applications applied to vineyards can be generalized also to, for example, orchards with no need of important modification of the technology.

Figure 2.6: Aerial view of an orchard in the left, and the aerial view of a vineyard on the right

2.2.4 Sensors for precision agriculture

Usually, unmanned ground vehicles are equipped with a set of sensors that are crucial for them to be proper autonomous vehicles. These devices are expected to be equipped with proper sensors that are able to correctly perceive the environment that surround the UGV to allow it to act in a proper way [11]. One of the tasks that make continuos use of sensors is the autonomous navigation, that wants the vehicle to go from point A to point B without hitting any obstacle. In many cases, such obstacles can also be dynamic, so they could be present in a moment and then move position, like an animal for example. For the correct measuring of the external and internal conditions of both the field and the UGV, there are different kind of sensors:

Exteroceptive sensors

They are in charge of measuring information that regards the surrounding conditions. the most used exteroceptive sensors are:

- GPS: The global navigation satellite system can point the global positioning of the device that is carrying it, be it a robot, another sensor, a beacon or even an animal. It allows to track movement and to track trajectories.
- Laser: This sensor is used to obtain information using light detection and ranging. That is why this technology is commonly known as LiDAR. Mainly

composed of a source of light, it emits light impulses of electromagnetic waves that are detected by a receiver when they are reflected by surrounding obstacles. By measuring the time of arrival (TOA) difference between the emission and the detection, the distance and the direction of the reflecting element can be found. There are two and three dimentional LiDAR sensors that are based on the same principle. The can be tailored to detect from very small particles in the air to greater visible objects, because the wavelength of the laser emitted can be very small, in the scale of nanometers. All considered, LiDAR are good range sensors, but their cost is higher than other range sensors

- Radar: Its working principle is similar to the LiDAR, except that the used wavelenght is different. These sensors use radio waves that travel at the same speed but have greater wavelenght. They can also be used to calculate the velocity of a moving object, because they are able to account for the Doppler shift of the echo withouth much numerical processing. The radar can sense in distances longer than the LiDAR with a good precision and is less sensitive to weather, but its resolution can be lacking if the element to be sensed is too small. Radars are cheaper than LiDARs and are better at keeping track of objects.

- Vision sensors: The most common vision sensor used on precision agriculture are common cameras and stereo cameras. As the stereo part of the name indicate, it consist on two separate lenses that look at the same scene from different but known perspectives, that result in two different perspectives of the same object. With this, a 3D representation of the observed scene can be obtained. The mos commonly used technique to reconstruct the 3D image is the triangulation. Common cameras can be used to obtain an RGB image of the field of view of the device it is mounted in. It can be used for feature identification of the environment.

Proprioceptive sensors

This sensors can collect data about the internal state of a dynamic system. For the case of a UGV or UAV, they can be of help to measure the velocity or orientation with respect to the earth magnetic poles. Among them, the most common are:

- Enconder: Used typically to measure the rotation angle of the objec being sensed with respect to a reference axis, they are usually involved in the estimation of the steering angle of vehicles and moving velocities. They can be mechanical or optical, yielding more reliable results the latter ones. They are composed of a grid disk, a light source

- Inertial Measurement Unit (IMU): It is a combination of sensors that allow to estimate the linear and rotational motion of the vehicle. They are typically equipped with three orthogonal gyroscopes, three orthogonal accelerometers and some of them include also magnetometers. The magnetometers can be used to measure the orientation with respect to the magnetic north, while the gyroscopes and accelerometers can measure angular velocities and linear acceleration respectively. The information coming from the IMU can be merged with the information coming from a GPS to cancel each other flaws and have a much more precise positioning reference. If the signal coming from the GPS is lost, then on-board IMUs can be of help to keep track of the localization of the vehicle. If the terrain causes slipping or very noisy signals, the GPS information can confirm the precise point in which the vehicle stands.

2.3 Deep Leaning

2.3.1 Artificial intelligence, machine Learning and deep learning

A term that is heard often in this digitalization era is Artifitial intelligence. Regardless, people that is not expet tend to mix the concepts of Machine learning (ML) and Artificial Intelligence (AI) [12].

They are for sure, linked concepts, and ML and AI are as well related to the Deep Learning (DL) one. They were connected through time by the progress of the technical sector. AI is a concept bron in 1950s, while DL is a term born not very long ago. The three are related and their relation can be explained with concentric circles as shown in 2.7.

The inner circle, and so the most specific concept is the one belonging to Deep Learning. Artificial intelligence is the wider concept and its related to computer performing tasks usually associated with rational entities. Machine Learning is a sub-field that specializes on learning from data automatically. Deep learning implies the use of artifitial neural networks with a large number of neurons and layers, so a deep network.[13]

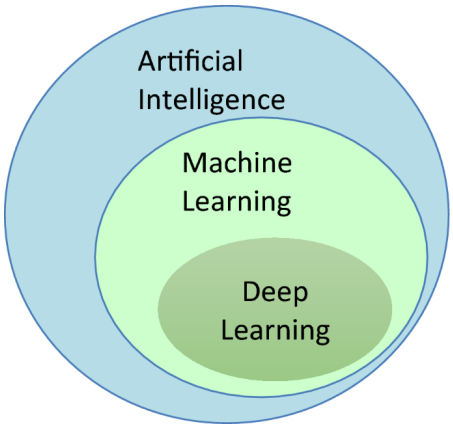

Figure 2.7: Relation between Artificial Intelligence, Machine Learning and Deep Learning

2.3.2 History of Machine Learning

The techniques, theories and technologies that have enable the development of machine learning as implemented in this work were developed along the course of

many decades, and although at some point they may have not seemed relevant, as new research was carried on, the piled knowledge allow to develop this potent tool. Since almost every new improvement that came along the years was based on previous studies, it is important to understand how every concept relates to each other and how of an outbreak were at their development time, as a way to show how nowadays, research in this field can continue making improvements [14]. A summary of the building of machine learning as known today is therefore presented.

The starting phase of Machine Leaning (ML) research was based on very simplistic models of neurons. Another therm for the sequence of trials to mimic the brain function is cybernetics. The credit for creating the first model goes to McCulloch and Pitts, in 1943, year in which they tried using a basic linear binary classifier to simulate a neuron, tought of as a simple digital processor, and the computing machine as the whole brain, called electronic brain by them. For this neuron, it was possible to separate between two classes of input by looking at the sign of the function $f(x, y) = x_1w_1 + ... + x_nw_n$. Setting the correct weights from w_1 to w_n was a responsibility of human intervention.

The next significant step happened in the 1950s, when the perceptron was introduced by Rosenblatt. This perceptron was capable of modifying the weights via an iterative training process, making use of a collection of input for each class. A considerable success was accomplished by this perceptron algorithm. Meanwhile, Alan Turing was devising a test to determine whether a machine could defy a person into believing they were conversing with another human, making a huge progress into the modern computer era.

The next important contribution was made by Widrow and Hoff, in the year 1960, with the introduction of the adaptive linear element, also named as ADALINE. The method used to modify the ADALINE's weights was a prior version of the stochastic gradient descent, that is used in modern deep learning. Nowadays, linear models are often used and reinterpreted, despite their numerous limitations, because of their easy implementation and understanding. As an example, it is common that linear models fail at describing or replicating the XOR function, an implication discovered by Minsky and Papert in 1969. This caused a decline in interest in lineal models that were based on neurons during the following period, in what is known as the first winter of artificial intelligence, because almost no research continued to be carried.

The fuel that started the second important phase of neural network research began in the 80's, with the connectionism approach, in another attempt to reproduce and understand the human mental process and structure.

Connectionism allowed critical contributions to modern deep learning. Its core premise is that the mental process can be described as networks of simple and frequently uniform parts. The connections and units could be different from model to model in appearance, but something remained common between them. More precisely, the network's units could be the neurons and its connections could be like synapses, similar to the human brain

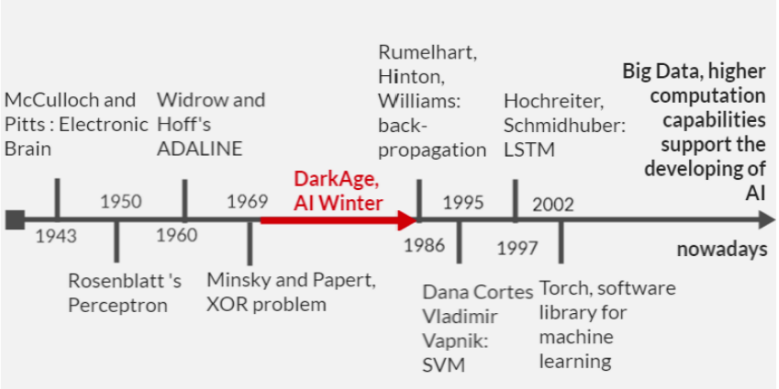

Figure 2.8: Timeline of machine learning

This lead to the creation of a new learning process called back-propagation, for network with this neuron units, coined by Rumelhart, Hinton and Williams in 1986. This is a process that, through iterations, adjusts the weights of the network's connections as to minimize the difference between the network's desired and actual output vectors. At this point, the networks are starting to be constructed by putting together hidden layers and weights that are not included in the input of output, and that, nevertheless, could represent critical aspects of the task domain.

Dana Cortes and Vladimir Vapnik, in 1995, presented the support vector machine (VSM) as a system for mapping and identifying data with simmilar features. The next important characteristic was introduced in 1997 with the inclusion of long short-term memory (LSTM) by Hochreiter and Schmidhuber, who devised a network to overcome the complications inherent in modelling mathematically extended sequences.

Even if progress was being made over the course of the years, there have always been many opposing views on machine learning, deemed by some as too ambitious or impractical. At some point, even the moral aspects of it was discussed.

One very relevant fact is the launching of ImageNet, a free database with more

than 14 million labeled images, by Fei-Fei. With the help of this amount of free data and the work of people like Yoshua Bengio, Geoffrey Hinton, Yann LeCun, it became clear that neural networks could be trained to a specific task. In present days, deep learning methods can beat AI systems that are based on other ML techniques.

One explanation is that the resources to support these complicated algorithms are available now, contrary to the past. As a matter of fact, the world we are living in now can be called the age of big data, as we see cities become more digitized and networked. Traditional computer hardware was not design to perform algorithms such as those that try to replicate the brain and so it becomes very complicated to execute, and the computational power they required and the magnitude of complexity became much higher than the human brain.[14]

As with many other fields in the world, it was the development of new technologies that made it possible for its rising. In this case, the dramatic rise in computational resources, such as better and faster CPUs and GPUs enabled the operation of highly sophisticated neural networks. Even if it is true that artificial neural networks are structurally similar to the human brain in therms of neurons, topology, learning process and weights, they are not yet capable of simulating many complicated activities. Out brains accumulate a lifetime of data and it is very complicated to capture life experience in a particular dataset, so we still do not know how far this technology can reach. It is very promising, however, and so a great amount of research is being carried out.

2.3.3 Basic concepts of machine learning

The concept presented in the previous section will be now introduced to have a low-level idea of how a neural network works based on the work presented in [15]

Threshold Logic Unit (TLU)

A said before, neural networks derive from the concept of human brain. The foundation of neural networks, is of course, the neurons and they biological properties. Researches tried to emulate them into ANN and have several thing in common: Neurons in nature are made up of a cell nucleus that gets information from adjacent neurons through a network of input terminals, or branches, also called dendrites. With an electrical exchange of neurotransmitter, a chemical substance, the dentritic tree receives inhibitory or excitatory messages from other neurons. The quantity of substance deposited in the synaptic gap defines the synapse conductivity, and that can be interpreted as a measure of how much the synapse attenuates of boost the

signal on the axon. The nucleus interpret these electrochemical signals.

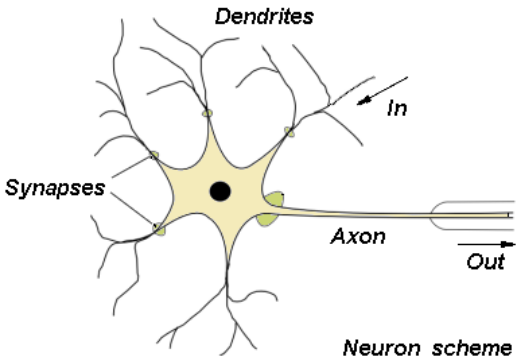

Figure 2.9: Biological neuron

After a process called neural summation, if the sum is higher than a synaptic threshold, the neuron is activated or inhibited, or, in another interpretation, switched on of off. The final output is then sent into other neurons, and the entire process is repeated.

After understanding the described process, an attempt of reverse engineering derived into artificial neural networks. The mathematical description that resulted is called the Threshold Logic Unit (TLU). It is the simplest type of an artifitial intelligence computational unit, coned by McCulloch Pitts in 1943. It mimics the high-level scheme of biological neurons.

Normally, a neuron gets different inputs, each of which is associated with a weight w that would represent the synapses conductivity. In a TLU, neurons have binary output and identical weight. If no inhibitory signals are found, then all the weighted inputs are added. If the value is higher than a threshold of a particular activation function, then an output is generated. This can be matematically described as follows:

$$f(x) = \begin{cases} 1\ , & if\ \Sigma_{j=1}^{n} \geq \theta \wedge no\ inhibition \\ 0, & otherwise \end{cases}$$

Perceptron

The Perceptron of Rosenblatt, is the natural evolution of the TLU proposed by McCulloch and Pitts. It uses some of the compositional policies (MCP) concepts in a more relaxed way, and achieves learning from data in a supervised environment.

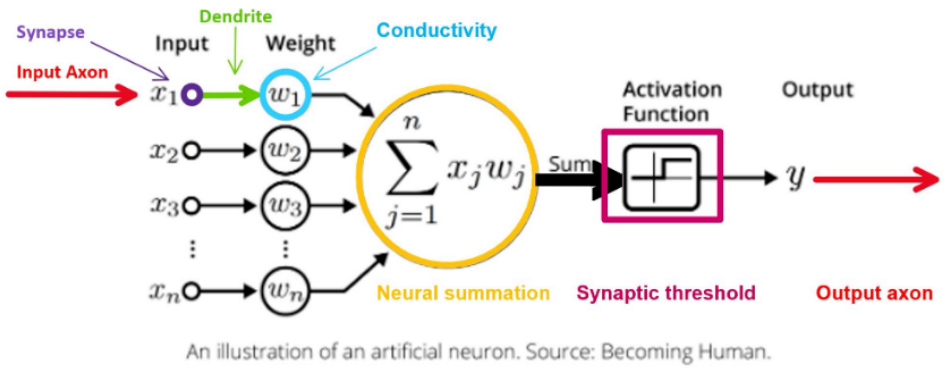

Figure 2.10: Description of a threshold logic unit (TLU)

The main differences with the TLU's structure are:

- The neurons also include a constant bias input term b or weight w_0
- The input can be negative with an inhibitory influence
- The weights can be different from each other.

The model can be written as:

$$y = \sigma(\sum_{j=1}^{n} w_j^T x_j + w_0) = \sigma(w^T x + b) \tag{2.1}$$

This can be translated into:

$$\sigma(w^T x + b) = \begin{cases} 1 if \sum_{j=1}^{n} w_j^T x_j + b \geq 0 \\ 0, \qquad\qquad otherwise \end{cases} \tag{2.2}$$

Here, the activation function σ is the Heaviside function ($H(z) = 0 if z < 0, H(z) = 1 otherwise$). However, other activation functions can be also considered with equivalent results:

- Signium activation function, that can output either 1 or -1

$$sgn(z) = \begin{cases} 1, if z \geq 0 \\ -1, \quad otherwise \end{cases} \tag{2.3}$$

- Threshold gate activation function has as output 1 if the sum of the inputs is higher than a threshold

$$sgn(z) = \begin{cases} 1, if z \geq \theta \\ 0, \quad otherwise \end{cases} \tag{2.4}$$

The decision function linearly depends on the input, in any of the cases, so all of them can be considered linear classifiers.

Binary classification can be done with this efficient and simple learning process, if the synaptic weights w_i are correctly learned from the examples of the training. The iteration through the data can help in the convergence to weights of the optimal value, as well as trying to update these values if a prediction error different from zero is present. The name of first generation neural networks is given to the ones based on these kind of neuron model, and they, with a single layer, can implement every binary discrete function. They, however, since are linear, can only tackle effectively linearly separable features, so it is necessary no include non linearity in the architecture in many real-case problems that cannot be faced with linear functions

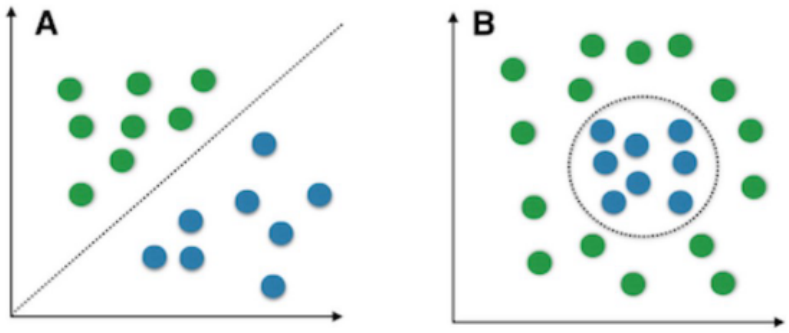

Figure 2.11: Advantages of non-linear activation functions

Non-linear activation functions

The next step towards the evolution of the perceptron into artifitial neural network is the inclusion of non-linear activation functions, and non-linearity in general to the model. The condition for a function to be an activation function is that it must be differentiable everywhere, countinuos and monotonic. Some very used examples are:

- Sigmoid: The most used one initially. It converts the input in a range in between 1 and 0, expressed as:

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(w^T x+b)}} \tag{2.5}$$

  With this formulation, large positive number tend to become 1 and large negative number get close to 0. It has fallen out of favor in recent days, but it

was historically preferred because of how it interpreted the neural behaviour. One drawback is that it could lead to a network barely learning, since it can kill the gradient because of how flat it is in the tails of 1 and 0

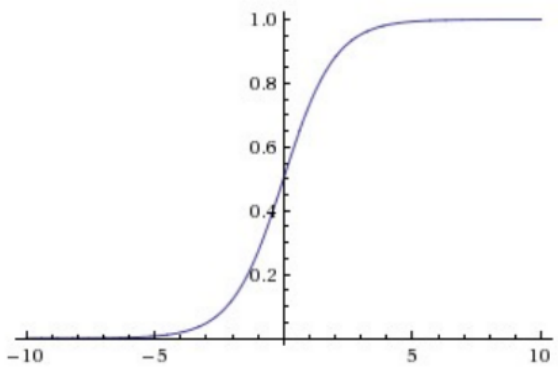

Figure 2.12: Sigmoid activation function

- Tanh It is zero-centered and preferred over the sigmoid non-linearity. In this case, the output is squashed in the range (-1,1):

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = s\sigma(2z) - 1 \tag{2.6}$$

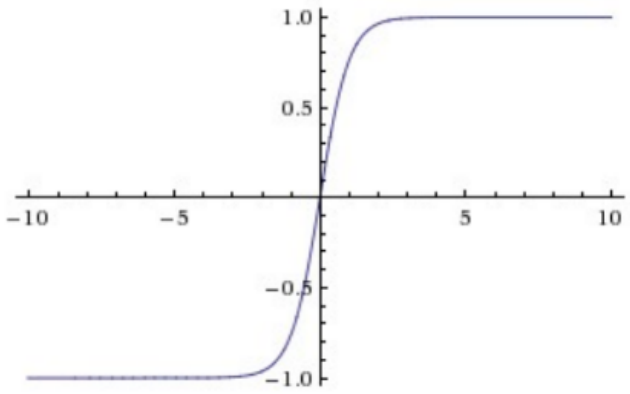

Figure 2.13: Tanh activation function

- ReLu also known as Rectified Linear Unit, is defined as:

$$\sigma(z) = max(0, w^T z + b) \tag{2.7}$$

  This activation function is thresholded at zero, meaning that the output is zero for negative input, and a linear ramp for positive inputs. It is computationally better, since its mathematical operation is simple, and is preferred to the tanh or the sigmoid, mainly because it accelerates the convergence of the stochastic gradient descent with its non-saturating, linear form.

- Leaky ReLu is a variation of the ReLu, that allows a gradient for negative inputs, which can be useful in certain specific training:

$$LReLu(z) = \begin{cases} z, if\ z \geq 0 \\ \alpha z, \quad otherwise \end{cases} \tag{2.8}$$

with α being a parameter to be chosen by trial and error in the design phase

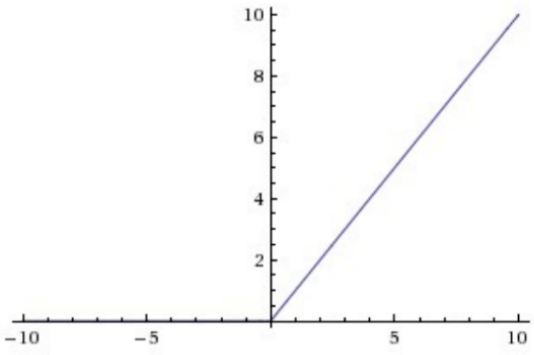

Figure 2.14: ReLu activation function

2.3.4 Artifitial neural networks

The more potent CPUs and GPUs allowed to consider more complex architectures. [16] Most of the neural models that are used now consider at least three main components: An output layer, an input layer and one or mode hidden layers:

- The raw or pre-processed data is received by the input layer. This layer recieves the input that the system is trying to recognize, interpret or translate.

- The network's processed final response is transmitted through the output layer
- The hidden layer is located in between both output and input and is the part in which the weights are considered, the activation functions enter into the picture and most of the processing is done

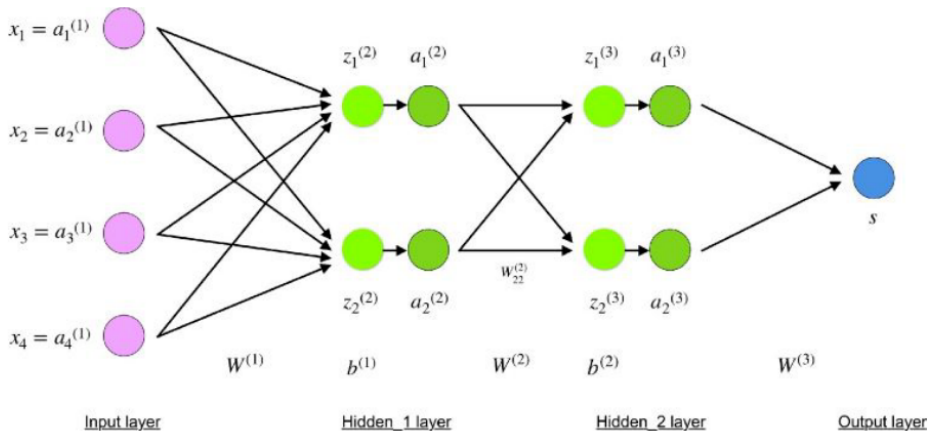

Figure 2.15: Deep neural fully connected network

A network where all the neurons of a layer a connected to all the neurons of the next layer, and non-consecutive layers are not connected is called a fully-connected neural network.

Training an ANN

The set of equations that produces the output s is called forward propagation. As a last step, it evaluates the obtained output s with the expected output y. Since the main goal is to obtain the best bias and weights to have a model that outputs an accurate prediction depending on the task. Starting with some values, either random or pre-trained, the biases and the weights are updated in the training phase through the optimization of a cost function, that summarizes how well the model is performing in its predictions.

$$L(y, s) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - s_i)^2 \tag{2.9}$$

The formula presented above is an example of a quadratic cost function, commonly known as mean square error (MSE). It bases its calculation on the differences between the desired output y and the actual output of the network s, for every training input i. It shows how far away is the prediction from the actual ground

truth. Being s the output of the activation function that depends on weights w and biases b, the training problem then can be described as finding the best parameters that minimize the value of the cost function. To achieve this goal, an algorithm called gradient descent is implemented. It takes the opposite direction of the gradient to reach the minimum. For a generic n-dimensional array v, if there is a small variation of every component v_j, it is possible to define the change of the variation of the cost function as:

$$\Delta L \approx \frac{\delta L}{\delta v_1}\Delta v_1 + ... + \frac{\delta L}{\delta v_j}\Delta v_j + ... + \frac{\delta L}{\delta v_n}\Delta v_n \tag{2.10}$$

By also including the definition of gradient:

$$\Delta L \approx \nabla L \cdot \Delta v$$

Whit Δv being the vector of the variations of v.

This process can be thought of as reaching the valleys of a mountain starting from the top as a metaphor for minimizing the cost function, if possible, taking the shortest path. At each iteration, the negative gradient must be calculated and update the weights in the direction this specifies, until the cost function reaches the bottom of the graph. How big the steps are is controlled by the learning rate η:

$$\Delta v = v' - v = -\eta \nabla L \tag{2.11}$$

And mixing the last two equations:

$$\Delta L \approx -\eta \nabla L \cdot \nabla L = -\eta \left\| \nabla L \right\|^2 \tag{2.12}$$

Making the rule for the update of the parameters at each iteration:

$$v \longleftarrow v' = v - \eta \nabla L \tag{2.13}$$

Choosing an appropriate learning rate is fundamental for the training. Too high of a learning rate means that the steps are faster, but there is a risk of overshooting and missing the lowest point, or event of diverging from it. In the other hand, a small learning rate can reach the optimum more carefully but can be very time consuming . Finally, the rule for updating the weights and biases are:

$$w_j \longrightarrow w'_j = w_j - \eta \frac{\delta L}{\delta w_j}, b_j \longrightarrow b'_j = b_j - \eta \frac{\delta L}{\delta b_j} \tag{2.14}$$

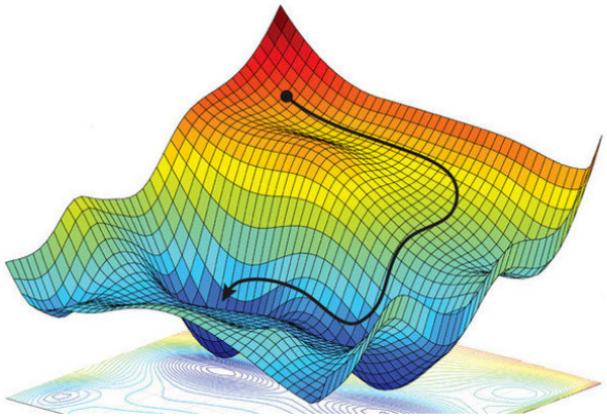

Figure 2.16: Local optimum found by gradient descent

Metodologies of gradient descent

To calculate the update of the weights there are many approaches. In Batch Gradient Descent [17], all training data is processed in a single step. For each input the gradient cost is calculated and then the average is computed.Being this the case, if the number of data is very large, the amount of calculations can lead to very slow convergence. A more efficient approach is mini-batch gradient descent, where a small amount of the training set is considered in each step, a mini-batch , and the average is calculated over them to update the weights. A training epoch is considered finished when all have been calculated,and at this point, a new cycle is started. This approach speeds up the calculation, even if the cost varies a little. An extreme version of mini-batch GD is stochastic gradient descent, where only one sample at a time is considered for the single step, so that the weights are updated after each training input. This is like setting the stack size to one. This leads to highly fluctuating costs, but it can be used for large datasets and converges faster. Batch gradient descent is guaranteed to produce a loss decrease, but the other two methods cannot assure that steps in the optimal direction are taken, that is why they may vary over iteration.

Back-propagation

The process which acts on the loss function's gradient computation for each layer with a chain rule is called back-propagation. This algorithm takes this name because the error is propagated backwards across the network, starting from the last layer L. The main feature of the procedure is a quantity called error δ_j^l and it

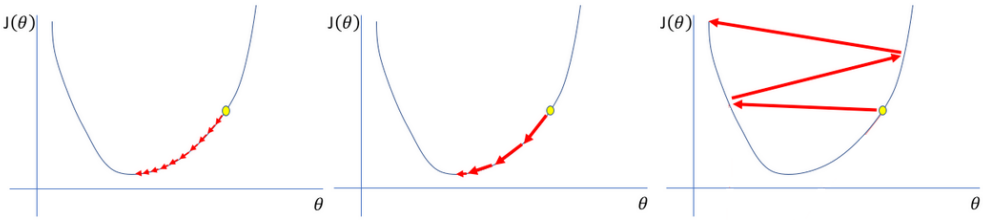

Figure 2.17: Diferent outcomes wiht different learning rates

serves the purpose of expressing the error commited by the neuron.

$$\delta_j^l = \frac{\delta L}{\delta z_j^l} \tag{2.15}$$

On the output layer, the error will then be formed by a couple of terms, the first one highlighting the influence of the last neuron's output in the cost function, while the second one concerns the influence of the input z_j^l in the activation function σ

$$\delta_j^l = \frac{\delta L}{\delta a_j^l}\sigma'(z_j^L) \tag{2.16}$$

The back-propagation algorithm makes use of the chain rule and applies it to the loss function partial derivatives. In this way, the gradient for a single weight w_{jk}^l of the layer l is described by:

$$\frac{\delta L}{\delta w_{jk}^l} = \frac{\delta L}{\delta z_j^l}a_k^{l-1} \tag{2.17}$$

To the bias term b_j^l a similar equation can be obtained:

$$\frac{\delta L}{\delta b_j^l} = \frac{\delta L}{\delta z_j^l}\frac{\delta z_j^l}{\delta b_k^l} = \frac{\delta L}{\delta z_j^l} \tag{2.18}$$

In this equation, the error term $\delta_j^l = \frac{\delta L}{\delta z_j^l}$ of every hidden layer, propagates the error from a specific layer to the previous layer, taking care of the activation function meanwhile:

$$\delta_j^l = [(w^{l+1})^T\delta^{l+1} * \sigma^{l\prime}(z^l)] \tag{2.19}$$

With $*$ denoting the element-wise product. These past four equations contain the essence algorithm.
This algorithm together with the mini-batch gradient descent follow the next

steps, while assuming an adequate learning rate η, a batch size m and a random initialization of the parameters:

1. Take a random mini-batch as input
2. For each input x in the mini-batch:
   - x enters the input layer
   - Feed forward propagation: For each layer l, the results $z^{x,l}$, $a^{x,l}$ and the final prediction $\widehat{y}$ are calculated
   - Output error: The error $\delta^{x,L}$ of the output of the final layer L is calculated
   - Back-propagation: The error is propagated along the network $l = L, L-1, L-2, ...,2$
3. Following the rule of mini-batch gradient descent for weights and biases:

$$w^l \longrightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} a^{x,l-1} \tag{2.20}$$

$$b_j \longrightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l} \tag{2.21}$$

4. The successive mini-batch is processed until there is no more data input left, so the epoch can be considered over, as well as the weight actualization.

Momentum

Gradient Descent can be further improved to make and advance version that includes Momentum, which objective is to guide the gradient in the correct direction in a faster way. The real world cost functions may have more than one local minima, and reaching the global one could require large amounts of time. Optimizing the loss function in machine learning consist in finding ways to find the global minimum. The **Momentum** parameters is thought as a way to include history to the parameter update equation, updating the weights instead of the gradients directly, using the exponentially weighted averages of the gradients.

$$V_{dW} = \beta_1 V_{dW} + (1 - \beta_1) dW$$
$$V_{db} = \beta_1 V_{db} + (1 - \beta_1) db$$

With β_1 being a hyper parameter to be tuned. In this case the updating rule is:

$$W = W - \eta V_{dW}$$
$$b = b - \eta V_{db}$$

Just including Momentum into the picture may improve by a good margin the convergence times as well as the overall performance.

RMSprop

In a similar way that the momentum does, the Root Mean Square Propagation (RMSprop) is considered when a reduction in the oscillations in the vertical direction is wanted, allowing also a faster convergence. It is described by:

$$S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2$$
$$S_{db} = \beta_2 S_{db} + (1 - \beta_2) db^2$$

With β_2 being a hyper parameter to be tuned. In the case that dW^2 was smaller than db^2 then the algorithm starts to take steps horizontally, in the direction of W, because the updating of b is more dumped in that situation:

$$W = W - \eta \frac{dW}{\sqrt{S_{dW}}}$$
$$b = b - \eta \frac{db}{\sqrt{S_{db}}}$$

The difference on the updating ratios of the two can lead to a better convergence

Adam Optimizer

Known as Adaptive Momentum Estimation, its idea is to consider a combination of RMSprop and Momentum because it can handle sparse gradients on problems with noise, it is widely used in many applications. It calculates an exponentially weighted average of the past gradients and the squares of the past gradients, having two hyper-parameters β_1 and $beta_2$ control the decay rates of these moving averages, and updates the parameters in a direction based on the combination of both results. This combination can be expresed in the following way:

$$W \longrightarrow W - \eta \frac{dW}{\sqrt{S_{dW}} + \epsilon}$$
$$b \longrightarrow b - \eta \frac{db}{\sqrt{S_{db}} + \epsilon}$$

Usually, for $\beta 2$ and $\beta 1$ the fixed values of 0.99 and 0.9 are used respectively, whereas $\epsilon = 10^-8$. This optimizer has proven useful in a large variety of problems and is usually considered more robust and efficient, in overall better than the SGD

Fine-tuning

The concept of fine-tuning is related very closely to the concept of transfer learning, where we apply the knowledge gained in solving one problem to a new but somewhat related problem.
Fine-tuning is a technique where a model that has already been trained for one task is tuned to a second, similar task. For example, an already trained network for a large and diverse dataset such as Cityscape captures universal features like curves and edges in its early layers, which are primordial and useful for most classification problems. Whena model is developed from scratch, normally, there is the need to try many different approaches by trial and error, and a large dataset or a lot of time to train the network is needed.For this reason, the fine-tuning method is so useful. If it is possible to identify a trained model that already does a task in an adequate way, and that task can be remotely associated to the one that is being tackled, then it is possible to use what the model has already learned and apply it to the new specific task. Fine-tuning is done by using the weights of a pre-trained neural network as initialization for a new model that is trained on data from the same domain (such as images, for example). Additional operations can be done in the fine-tuning process. one of them is to "freeze" some of the trained weights,which are usually part of the first layers that usually capture general features that are also relevant the new task. Some other important considerations are to choose a low learning rate; it is not wanted to bias the pre-trained weights too much or too quickly, as it is assumed that they are already quite good compared to being randomly initialized. The fine-tuning process can be considered the most crucial part when a neural network is adapted to solve a new problem, so a big amount of time can and should be spent on this hyper-parameters fine tuning.

2.3.5 Data overfitting and underfitting

The problem of a model fitting the data used to train it very well, but failing against additional, new and unknown data is known as overfitting and is an important issue in neural networks, since it basically is contrary to the main goal a neural network has[18]. If an artifitial neural network is unable to handle never-seen-before data, it is completely useless for the classification or prediction task that it was designed for. It can happen when a model is too complex or has been trained for too long, causing it to learn the dataset intrinsic error and irrelevant information. A model is said to have high variance if it is sensitive to minimum variations in the training set. The overfitting of a model is evidenced when there are low error rates in the training stage and high rates in the test set. To correct it, some strategies can be used, such as having an early stopping of the training, which is to take the state of the model in a middle point of the whole optimization process, or removing some less relevant inputs. This, however, can lead to the opposite problem, called as a

matter of fact, underfitting, an error that happens when the model is either not trained for a sufficiently long time, when the inputs are not significant enough or when the model is not complex enough. Solving overfitting and underfitting may require a great deal of trial and error. Some of the most popular approaches to solve them are:

Augmentation of data

Since the amount of data that the a model is fed controls somehow the outcome it can have, data augmentation is done to increase the size of the dataset. In the case of images, the data augmentation consist in applying specific transformations to either the hole image or part of it. Some of them are random rotations, flipping, color modification, cropping, tone scale changes or merging of parts of different images.Data augmentation can help in solving some innate bias in the original dataset. A simple example is a dataset of cats where all the cats are facing to the right. Applying a random rotation to some of the images can help the model to also recognize cats that are not facing to the right if they are ever presented after the training.

Splitting data

One very common practice in machine learning is to split the dataset in three parts, training, validation and test dataset. The training set is used to actually train the network, the validation set is used to evaluate the performance of the network after at a given point of the training, to help determining the direction that the optimization should take. Finally, the test set is used to test the network. Splitting the data may reduce the data available for the training, which can have a large impact if the dataset is small. each of the splits should contain a somewhat homogeneous sample of the dataset to avoid a biased evaluation.

Regularization

Regularization is another method that adds a penalty term to the cost function, to leave out excessive fluctuations on the function. The main regularization techniques are called L1, L2 and dropout, being the L2 the most common of them, also called as weight decay. their description is as follow:
Let J be the cost function to optimize:

$$J(w^1, b^1, ..., w^L, b^L) = \frac{1}{m} \sum_{i=1}^{m} L(\widehat{y}_i, y_i) + \frac{\lambda}{2m} \sum_{l=1}^{L} \left|\left|w^l\right|\right|^2$$

The regularisation term is composed of the sum of the squared weights with a multiplicative factor that depends on λ, the regularisation rate. Increasing this parameter gives greater importance to the second term, with the effect that smaller weights are preferred

$$w \longrightarrow w^{'} = w(1 - \frac{\eta\lambda}{m}) - \eta\frac{\delta L}{\delta w}$$

An additional subtraction is introduced by the regularization into the current weights. In a similar way, the L1 loss description instead of having the square sum, presents the absolute value of the weights

$$J(w^1, b^1, ..., w^L, b^L) = \frac{1}{m}\sum_{i=1}^{m} L(\widehat{y}_i, y_i) + \frac{\lambda}{2m}\sum_{l=1}^{L}\left|\left|w^l\right|\right|$$

meaning that the updating rule will be:

$$w \longrightarrow w^{'} = w\frac{\eta\lambda}{m}sgn(w) - \eta\frac{\delta L}{\delta w}$$

While the L1 regularization produces matrices with several values close to zero, also called sparse matrices, in the L2, the regularization is proportional to the weight, keeping weights low but without setting them to zero. What these regularization do in a more intuitive way is that, since the weights are smaller, it is like the hidden neurons have less impact, or are not present, effectively reducing the total complexity of the neural network. A less complex model will avoid overfitting, but increasing the value of λ too much can possibly cause underfitting, making this regularization term a hyper parameter that need to be tuned as well.

Dropout

This regularization technique acts not on the cost function, but on the architecture of the neural network itself. During the training phase, there is a probability that a neuron of the network gets "turned off", and its weight value and bias is not updated. It is like trimming the neural network, reducing its complexity, but possibly introducing underfitting, so it need to be a controlled process as well.

2.3.6 Convolutional Neural Networks

Up to this point in the other sections, a fully connected network has been considered a reference network, meaning that every neuron in one layer is connected to every

neuron in both the next and the previous layer. However, when processing images, a fully connected network incurs high computational costs, making the performance of the system poor rather than optimal. The input layer is linked to a hidden layer, but each neuron is only linked to a group of pixels in the input layer. This group of pixels forms a window called the local receptive field. There is a bias and a weight for each connection that the receptive field has. This type of architecture allows the network to focus on low-level characteristics in the first hidden layer in order to assemble the information into a larger higher-level feature in the next hidden layer, which is the convolutional part of the network. To form the entire hidden layer, the window of the receptive field slides by a size called stride through the input pixels of the image, forming a layer of hidden neurons step by step. All pixels in the receptive field share the same neuron, i.e. they share the same weight, which greatly simplifies the number of parameters in the model. The union between all the biases, shared variables and weights is called the kernel or philtre.

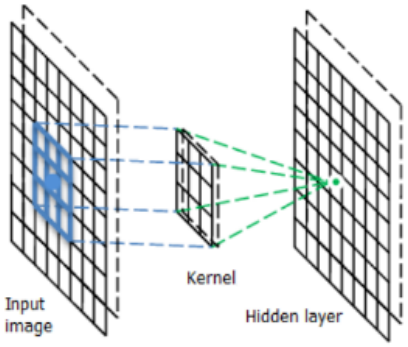

Figure 2.18: Receptive field

To coin the definition of convolution, lets have σ as a generic activation function associated to the neuron and with a $a(x, y)$ the input activation function positioned at (x, y). The output produced by the (j, k) hidden neuron is then:

$$out_{j,k} = \sigma\left(b + \sum_{l}\sum_{m} w_{(l,m)} a_{(l+j,m+k)}\right)$$

From this definition we can deduce that, since the weights are being shared, then the total number of parameters used in the network is reduced by a big margin allowing a faster training process compared to a fully connected architecture. A filter can be considered as a mask that can ignore every shape in their receptive field but the one that the mask defines. A shape detected by a filter can be as

simple as a horizontal line or as complex as eyes or more intricate shapes. A hidden layer that uses a filter outputs a filter map, enhancing the zones in the input image that are being activated because of the filter. The implementation of these convolutional filters sees the layers stacked into a variety of trainable layers, so every pixel is associated to a neuron in each individual feature map, and since every neuron in such feature maps share the same parameters, the total amount of trainable values is reduced.

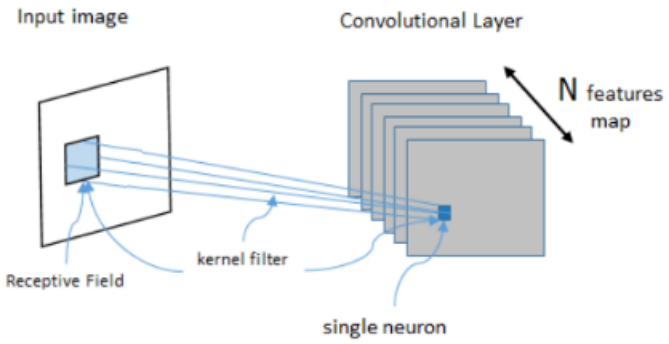

Figure 2.19: Feature Map in a CNN

It is possible to obtain a more precise control on the output size of the layer by setting to zero the pixels along the image border. A zero padding is added, that consist on zeros around the last layer, with the purpose of having the same dimension compared to the previous layer, in an attempt to make sure that no information is lost.

There are two kind of padding, actually, both looking to produce a larger or equally sized output of the layer:

- Valid padding, that consist in no padding at all, where the input pixels fallen outside because they don't fit the dimensions are dropped
- Same padding, where the output layer is equal to the input one

The number of filters is another parameter that can be chosen when composing a convolutional neural network, and it affects the depth of the output. Four filters would result in four different feature maps.

To perform dimension reductions, a pooling layer is due, that, without weights, is applied to small input regions, simplifying the contained information. Also in this case, two types of pooling can be found:

- Max pooling: Where the filter selects only the maximum value to be the output

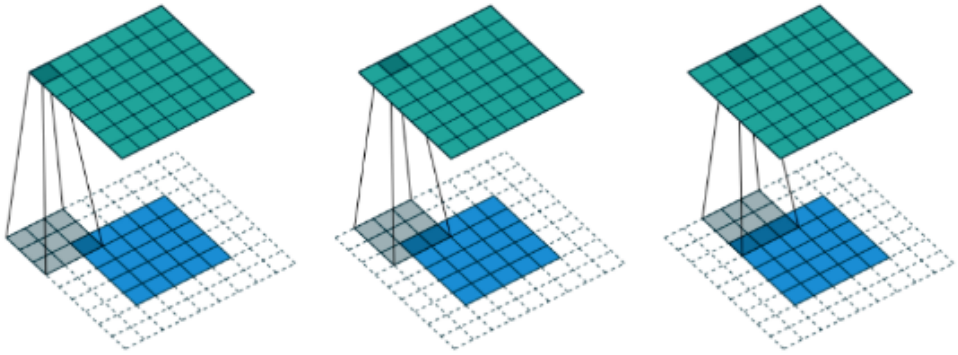

Figure 2.20: Zero Padding

- Average pooling: Where the filter outputs the average of the input region.

Basic Architecture of a CNN

A network based on the working principles just presented, and that may seem odd at first glance is present in figure 2.13. In this network, Weights and biases are the parameters to be trained, and the optimizer with backpropagation are used to this task. Differently than with fully connected layers CNN gives more attention to the spatial part of its input and treat every input in a different manner. The problem of recognition is subdivided along the entire lenght of the network and its different layers. The first ones are better at recognizing simple features, like circles, edges and lines, leaving deeper layers, with the help of pooling layers, to detect more complex characteristics. In a common CNN network, after a cascade of convolutional and pooling layers, the data is flattened in a one dimensional vector. This vector feeds one or multiple fully connected layers, that can use or not dropout regularization, attached to a softmax layer at the end, and is used to make a prediction of the image that was input. A way of thinking about the CNN is to think about a cascade of convolutional layers as a detector of increasingly more complex patterns, and a last fully connected layer as a model that takes the detected feature and associate them to the output. Just this simple logic combination to form the neural network can perform much better than a similar network with many fully connected layers. Replacing the sigmoid functions with ReLu activation functions can lead to an even faster training process.

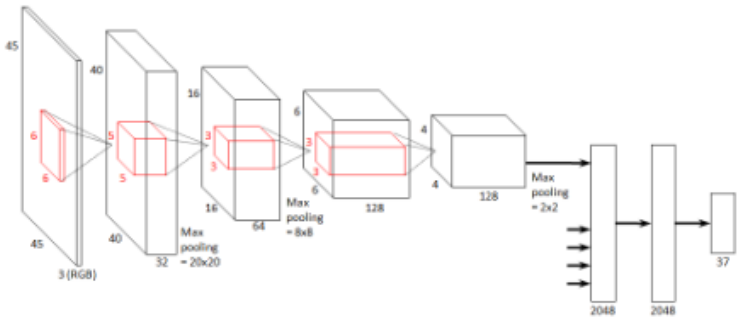

Figure 2.21: Common CNN with zero padding set to zero and convolutional and pooling layers

Chapter 3

State of the art

In this section, the methods used for vineyard segmentation are presented. A description of their advantages and disadvantages will also be presented.

3.1 Traditional methods for crop identification

One of, if not the most relevant study that involves computer vision techniques with the main focus being the vineyard segmentation on aerial imagery was done by Lorenzo Comba, Paolo Gay, Jacopo Primicerio and Davide Ricauda Aimonino in 2015 [19] under the name "Vineyard deteciton from unmanned aerial systems images" This work was perform before deep learning completely took over the image segmentation field over other non-machine learning based techniques, yet the results presented are outstanding. The method presented involved three main steps: Dynamic segmentation, Hough Space Clustering and Total Least Square techniques. The work was quite innovative because it allowed to extract vine row and inter-row information when it was a very complicated task and was a reference in the development of other precision viticulture tasks, like path planning for UGVs.

The metodology that this work presented needed no previous data of segmented vineyards, since it was not based on any machine learning that had to see a large amount of data, labeled or not, to learn from. Instead, the whole study presented 4 near-infrared images of 2048x1536 pixels of size, that contained various vineyard fields with different orientations as well as other types of vegetation like trees, houses or roads. The four images were all taken on the south region of the Tuscany, in Montalcino (Siena) and it is not specified whether or not the images were taken on the same time of the day, or season. The images have a 0.056 m/pixel resolution and were taken by a UAV at an altitude of 150m.

The first step of their procedure aimed at making a rough detection of the pixels that may represent vineyard vegetation, by binarizing the image, with potential vine plants represented by nonzero values. This problem is tackled with a dynamic local neighboring window segmentation procedure, that increases the contrast between the region of interest (ROIs) and the backgroun pixels of ground and other vegetation. The window is big enough to contain at leats part of two vine rows at the same time, and it slides trough the image calculating a histogram and a normalized moment $J(\Omega_{x,y})$

$$J(\Omega_{x,y}) = \frac{\sum_{\nu=1}^{255} n(\nu) \cdot (\nu - \mu_{x,y})^2}{N} \tag{3.1}$$

Where $n(\nu)$ is the number of pixels charaterized by digital number intensity value $\nu, \mu_{x,y}$ and $N = l_w^2$ is the number of pixels belonging to $\Omega_{x,y}$. If the value $J(\Omega_{x,y})$ is greater than the fixed threshold $T_j = 5 \cdot 10^2$, then, pixels that beling to $\Omega_{x,y}$ are considered part of a vine row and their counters are increased. This will result in an image with more contrast between the potential vine rows and the other classes. This is the image that is binarized, however, it still includes part of the vegetation and non desired elements as part of the vine rows.

The second step is the cluster isolation. Each group of the binarized image is clusterized, resulting in some clusters that describe several vine rows wrongly connected togheter. The goal with this step is to recognize the straight lines that are supposed to be the vineyards. For this line detection, the Hough transform was used, and then a moving average filter as well.

The vine rows are then separated in the case that more than one line intersects them, and finally, Total Least Squares is applied to calculate the best line, since it is invariant to rotation. A final vine row reconstruction is needed to recover all the information from the straight lines. Figure 3.1 Summarizes the workflow presented by [19]

Figure 3.1: Scheme of the methodology implemented for vine row segmentation using traditional methods

Some criticism about about this study could be that the amount of data it was

tested on was small, or too specific, and that there could be a lack of generalization in the methodology because only 4 images were considered. Since this work is based on Near Infrared images, and the current thesis presents a collection of RGB images, it is not possible to test on it expecting the same results. It demonstrates, however, that a classic and methodical approach can be used in the process of vineyard segmentation. Another result to highlight is that vine rows that are only partially visible are harder to recognize

3.2 Deep learning-based methods

Two works that include deep learning in the study of vineyard segmentation can be found in the literature, [20] and [21]. both of them more oriented towards field monitoring than they are to the problem of autonomous navigation.

Classification with multispectral and RGB images

In the first work, T. Barros, P. Conde *et.al*. titled "Multispectral vineyard segmentation: A deep learning comparison study". In this work, it is presented a comparisson between deep segmentation networks and conventional unsupervised methods. The study was done with the objective to assess which bands or band combinations of state-of-the-art sensors were more contributing to this segmentation task. The second goal was to determine whether the inclusion of high resolution RGB images had greater impact than the inclusion of multispectral low resolution layers.

According to them, most recent studies related to crop monitoring with aerial imagery include multispectral information, while only a few of them are based only on RGB images.

The images for the study came from three vineyard located in the centre of mainland Portugal. They are subjected to similar conditions but contain different varieties of vine plants. The distance between the vine rows ranges from 1.1m to 2.4 meters.

The survey flights were done so that the apparent size of the vineyards across the three fields was similar, so altitudes ranging from 120 to 60 m were used. Finally, the images were split into 240x240 pixel size and standardize using the following relation:

$$\chi'_b = \frac{\chi_b - \mu_b}{\sigma_b} \tag{3.2}$$

Where χ_b represents the image band b, μ_b is the mean and σ is the standard deviation. After the standardization, the images were fed to the neural networks.

The neural networks used in this work are both of the encoder-decoder type. The U-net that learns new indices for the transposed convolution in the upsampling, concatenating each new feature space, obtained after each upsampling step, while the SegNet architecture uses the same indices of the max pooling operations learned in the respective encoder steps.

For the unsupervised methods, K-means and OTSU were used for comparison purposes.

The evaluation procedure they used was to implement the F1-scroe as a metric, that mixes the True Positives (TP), True Negatives(TN), False Negatives(FN) and False Positives (FP) in the following way:

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{3.3}$$

According to this study, the best segmentation performance with only RGB bands are on average 4.1% of lower performance, which if multispectral sensors are not available, could be acceptable. with the best of the results of 81% accuracy for the U-Net and SegNet methods

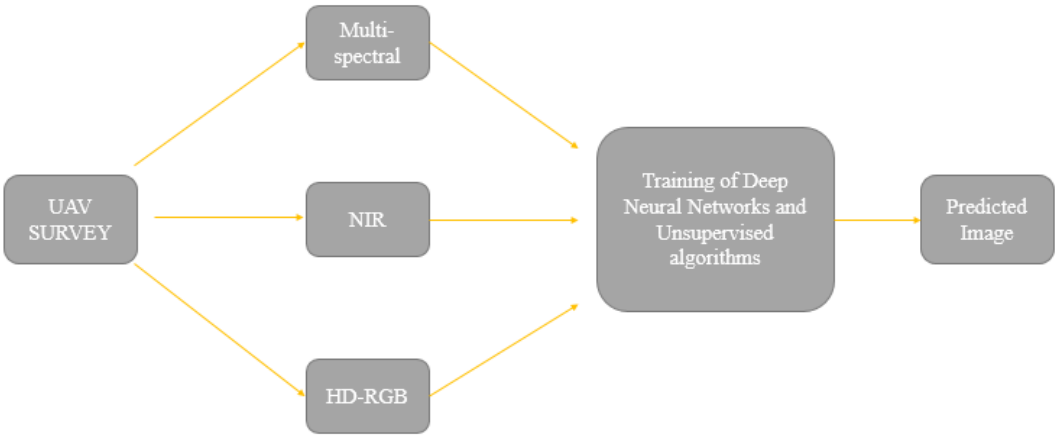

Figure 3.2: Methodology for the multispectral vs RGB imagery study

As a final remark, they conclude that the deep learning methods completely

outperforms the non supervised traditional methods regarding these vineyard segmentation of aerial image problem.

Vineyard disease detection

The second work, [21] carried by Mohammed Kerkech, Adel Hafiane and Raphael Canals, called "Vine disease detection in UAV multispectral images using optimized image registration and deep learning segmentation approach"

The areas of interest are two parcels of vines located in France, with surveys performed at 110 and 114 meters. One of the two areas was infected with the Mildew disease while the other was completley healty and was used for the purpose of validation. The fligth were done in such a way as to obtain optimal lightning and avoid shadows. The UAV used to do the survey was equipped with both a multispectral camera and an RGB camera

The first step of the study is to merge the visible images and the infrared images by using a proper registration algorithm. Since the images were not necessary aligned. A thorough process for the image acquisition was done, that included image normalization, thresholding and RANSAC, as well as the calculation of homographic matrix, allowing for a precise imagery.

Once the two vineyard fields were surveyed, the process of labelling and of data generation using data augmentation was carried. The dataset was then composed of visible and infrared range images. The infrared sensor operated at 850 nm, a wavelength chosen for its sensitivity to changes in the states of the vegetation. Four classes were labeled: ground, shadow, symptomatic foliage and healthy vegetation, differently from all previous studies.

A metodology for semi-autocally labeling the data was also presented as part of this work as a solution for the high time cost that manually labeling the images has. For this purpose, each image was classified by a neural network, the LeNet5, used for pre-labelling. A manual labelling was performed afterwards to form the ground truth based on the information of the technicians on the field. The images of the two fields were cropped to patches of 32x32 pixels and a sliding window with a displacement step of 2x2 was used. This images were manually corrected after the neural network had initially pre-labeled them.

The infrared and the RGB images were used to train two different SegNET, in charge of segmenting the RGB and infrared layers. Once their prediction was made, the results are simply added to get a prediction. The F1 score reported for the

classes of Healthy and symptomatic vineyards were 91.68 and 92.81 respectively.

3.3 Autonomous navigation on row based crops

Simone Cerrato, Vittorio Mazzia, Francesco Salvetti and Marcello Chiaberge presented in [3] the work named "A Deep Learning Driven Algorithmic Pipeline for Autonomous Navigation in Row-Based Crops". This recent work makes use of aerial imagery of vineyards and merges it with many other precision agriculture technologies assembled into an UGV, with the goal of achieving autonomous navigation.

This work proposes an algorithmic pipeline that has as starting point, a georeferenced occupancy grid of the field of interest. With this image, that is basically a segmented aerial view of the vineyard field, the metology they proposed is summarized in figure 3.3.

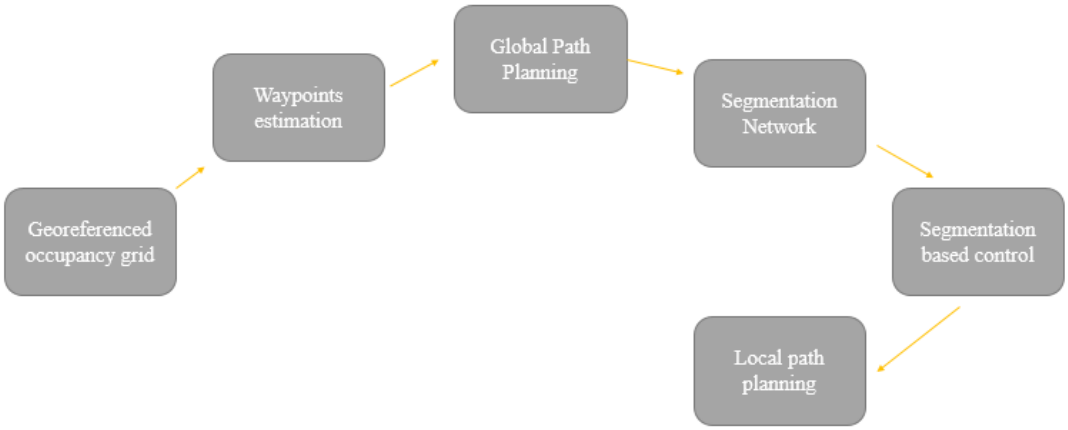

Figure 3.3: Methodology for the autonomous navigation study

The work was first tested on a simulation environment to test the UGV trajectory with the proposed methodology. Finally, it was tested in a real environment evaluation in both a vineyard and a pear orchard, with results of the UGV following the proposed path with a margin on a maximum of 0.296 meters in the vineyards and 0.659 meters for the orchard.

Chapter 4

Dataset Construction

Labeled data captured by remote sensing methods is not common, since the area covered by such images is huge and the time and cost for manually labelling them tend to be also very elevated. For this reason, most datasets conformed by satellite images contain usually a small number of samples. In the case that the data is capture by an unmanned aerial vehicle, the number of applications that incorporates them is wider. However, when it comes to the specific case of vineyard image segmentation, with very few exceptions, all the datasets are restricted in the diversity of the samples

4.1 Commonly used dataset

Reference	Used dataset
Comba *etal.* [19]	four 2048 x 1536 with a resolution of 0.056m/pixel from 4 different fields
Padua *etal.* [22]	88 images from 1 vineyard field at noon with 2 cm/pixel
Hajjar *etal.* [23]	591 patches of 224 × 224 pixels of 1 vineyard
Mazzia *etal.* [3]	100 images of 800x800 pixels taken from many locations
Kerkech *etal.* [21]	two 4608x3456 images of two fields cropped into 360x360 patches
Ronchetti [24]	520 1280 x 960 4 bands 1 farmstead

Table 4.1: Composition of the datasets used in studies that involve aerial imagery of vineyards.

Table 4.1 shows summarises the fact that most studies carry out a UAV survey in 1 to 4 fields, gather the information of each vineyard in one big image, and then make patches of them. In some cases, the resolution, growth state of the plants, and the illumination conditions are the same.

4.2 Data recollection

For the recollection of the data, the main objective was to have a final dataset as heterogeneous as possible, while still presenting the main characteristics of vineyards. For this purpose, images from all over Italy (North, South and Center) and some of them were also taken from area in California, United States. Since the method relies only on the rgb bands of information, the images could be taken from the google maps database or any other available aerial orthoimagery source, like the USGS aerial orthoimagery database.

The dataset is then composed by 1000 RGB images of size 800x800 pixels with many varying factors:

- Sun position: There is presence of images with the sun on top of the crop, where shadows are barely noticeable, as well as images where the sun projects a long shadow from the vineyard
- Illumination condition: There are images that were taken from morning to sunset, and images where the sky is clear as well as images where it is not
- Ground contrast: A high contrast between the vine rows and the soil usually facilitates the recognition of the vineyards. To increase the robustness of the model, many images with low contrast were also included.
- Plantation size: There are present vineyard plantation that range from very few rows to huge plantation that occupy dozens of ha
- Resolution: Images where the texture can be appreciated because of the high amount of detail, as well as images where mostly the shape is the only indication of the presence of vineyards.

A small collage of images present in the database with high differences between them in shown in figure

4.3 Image preprocessing

The captured images where resized to squares of 800x800 pixels, independently of the initial size of the image. If at least one of the height or width of the image is bigger than 800 pixels, then the image is shrunk to make the bigger size equal to 800 pixels. A padding is added to the other size to obtain a square image. If none of the initial dimensions exceed the 800 pixels, then padding is also added keeping the image at the center

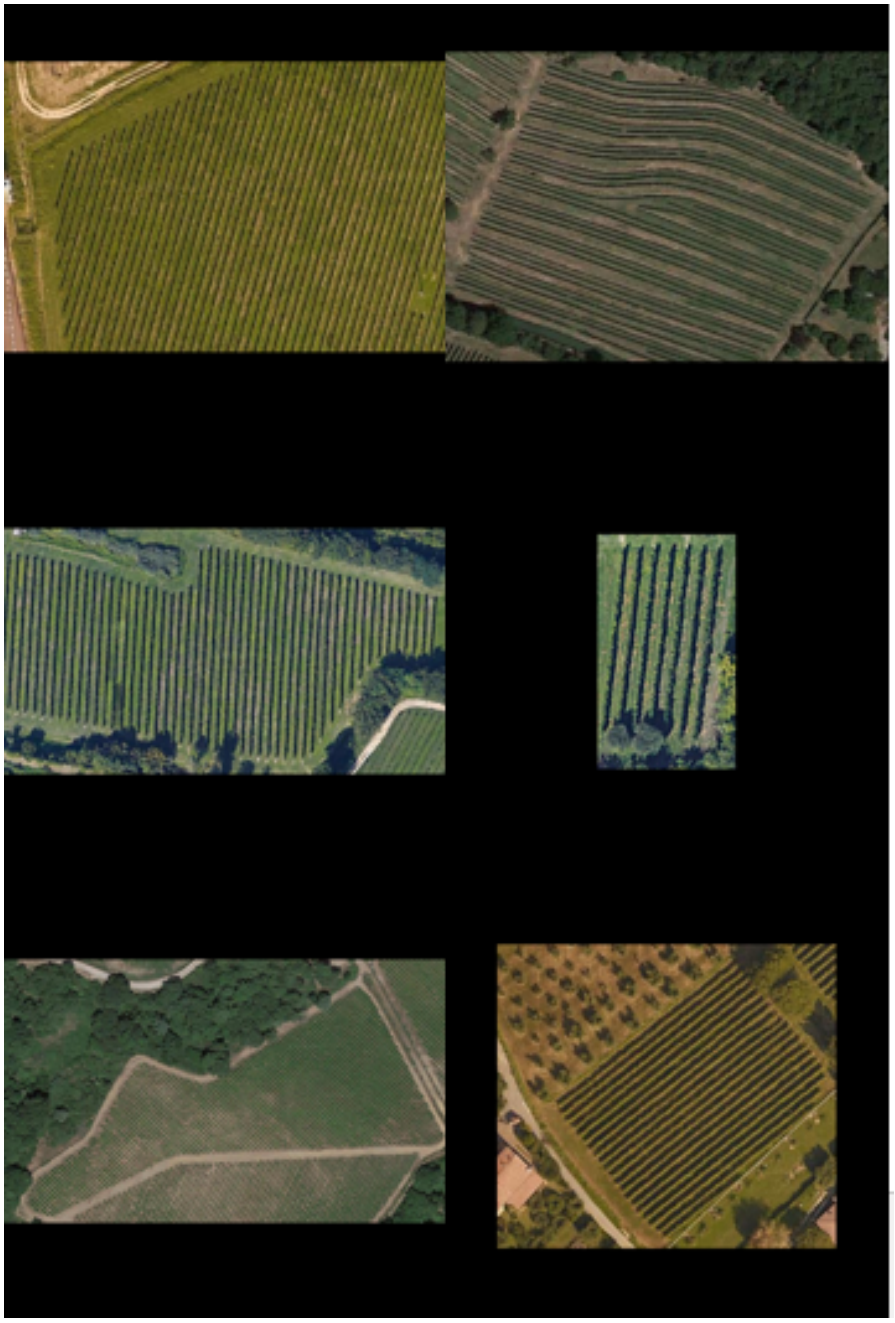

Figure 4.1: Heterogeneous images in the dataset

The order in which these two operations are performed is important, since if the image is first square and the shrunk, the size reduction operation may mix the new added pixels with the existing ones.

Figure 4.2: Original image

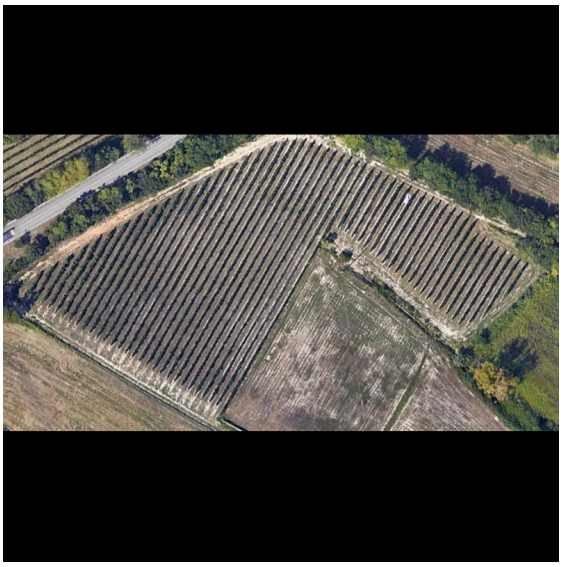

Figure 4.3: Resized image to fit the neural network input

4.4 Semi-Automatic labelling

Since manually producing finely segmented masks takes an immense amount of time, a more semi-automatic approach was taken. Several steps were required to do this.

The first thing to put into consideration about the initial dataset of 100 images is that the masks consist of straight lines that go along the vineyard rows, whether or not there is crop present in all the line. This lines are also all of the same width, and, normally, only the central field is segmented, leaving flank vineyards unsegmented.

The main idea was to use a simple neural network, like the presented as a baseline in this work, the Resnet50-Unet, to segment most of the image, therefore

Figure 4.4: Process of mask generation

reducing the amount of manual intervention needed for each image. However, since the dataset consisted of images with only the central rows segmented, the performance obtained by a first training was poor, and often times, not even the central vineyard was correctly marked, which is normal, since the network is receiving contradictory information on what is a vineyard and what is not.

To make these images more consistent with what the masks said, all the non-segmented vineyard were blackened, and a retraining of the neural network was done, achieving much more satisfactory results, reducing the amount of time spent in correcting each image from around 6 hour to around 30 minutes. Once enough finely segmented images were present, the network was again retrained and the time required for each image was reduced to about 15 minutes.

This process was done for the first 100 images, while for the other 900, since no mask was present at all, inference with the Resnet50-Unet was done to obtain a starting mask to modify. The neural network was retrained each time 50 new finely segmented images were available, until no further improvement on the quality of the images was noticed. The total amount of time spent on this stage rounds the 300 hours.

4.4.1 Paint.net

Since each pixel is important to the neural network, because using a supervised training it can only learn what is taught through the masks, a software that allows pixel-wise modification of the images was needed. For this, Paint.net is a perfect software.

It is perfect for the task because it allows the superposition of images and the modification in a pixel-wise level of the layer of interest

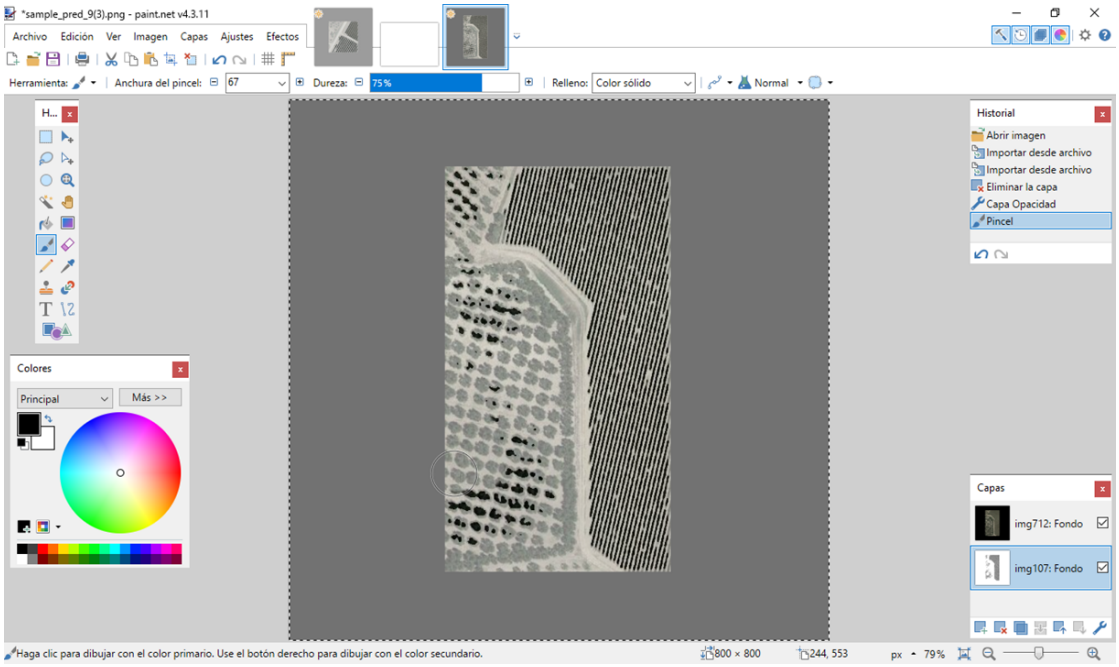

Figure 4.5: Pain.net layout

4.4.2 Image post-processing

In some cases, the neural network failed in ways that could be solved by traditional computer vision operations.

Erosion and Dilation

The first and second of these problems were caused mainly because the starting dataset was composed of lines with the same width, so the neural network tried to replicate it in situations where it was not precise.

Dilation of the image A with the structuring element B is represented as

$$A \oplus B \tag{4.1}$$

And is an operation that is typically used to enlarge or cause the growth in a region of the iamges. Erosion in the other hand, is represented as

$$A \ominus B \tag{4.2}$$

And since normally removes pixels from regions in an image, it can be used to eliminate little noisy dots. One example of these morphological operation applied

in this work is presented in the following figure, where erosion was used to correct the width of the mask of the vine rows

Figure 4.6: Original prediction

Figure 4.7: Eroded mask to further correct

Manual Traslation

In some cases, the shadow was segmented instead of the vineyard, which can be considered a fatal error and a completely incorrectly classified image. Fortunately, in those cases, due to the parallel nature of the vineyard rows, a simple movement of the mask is enough to correct this error

Figure 4.8: Original prediction where the shadow is segmented instead of the vine row

Figure 4.9: Manually corrected image by means of the translation of the predicion

Chapter 5

Deep learning model architectures

In this seciton, the architecture of the two neural networks used for prediction is presented, as well as the reason they were chosen for the study

5.1 Resnet50-Unet

The widely used Unet architecture [25] is exploited in this work because of its relative simplicity, as well as its capacity to rely on data augmentation and not only on large datasets. It is further enhanced with a resnet50 backbone as the feature extractor, making it a competitive model, used in remote sensing aplications.

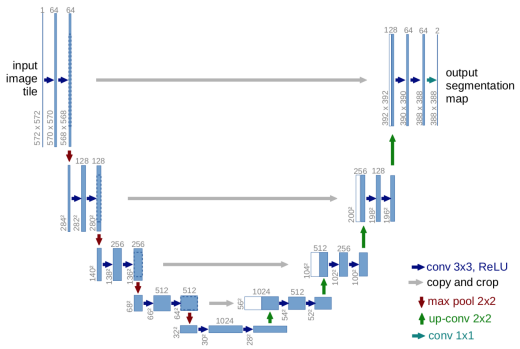

Figure 5.1: Unet Architecture

This neural network is pretrained on the imagenet dataset and transfer learning is used to adapt the network to the new clases used in this specific application.

This model was chosen not only because of its popularity, but also because of its good performances according to the state of the art.

5.2 High Resolution Network with Object Context Representation (HRNet+OCR)

The combination of the Object Context Representation with High resolution network as backbone [26] is a state of the art technique capable of producing high quality results since it is a more complex architecture. Because of the fact that the prediction are to be done offline, the complexity of the network as well as the inference time is not a critical factor, however, because of the same reason, memory consumption could be an issue. The HRNet module takes advantage of

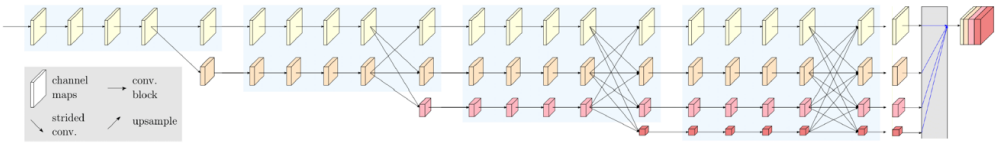

Figure 5.2: HRNET backbone

high resolution convolutions and by mixing them with low resolution layers of operations is able to extract features from images. The object context representation

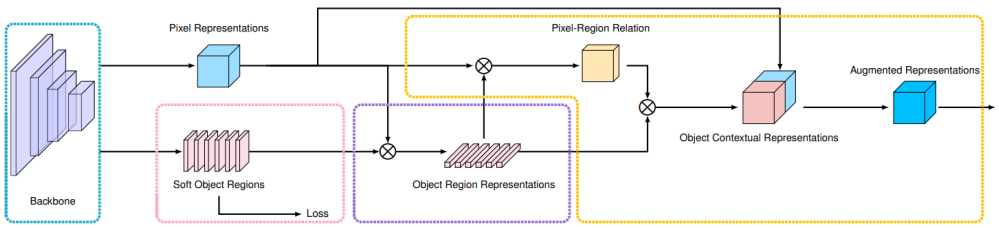

Figure 5.3: OCR Architecture

module uses the features from the previous stage and by its two channel object region representation pipeline is able to assign each pixel of the image to an object region. This cutting-edge model was chosen because of its top performance on the cityscapes dataset, becoming nowadays the base for the development of better networks.

Chapter 6

Work Plataform

This chapter describes the hardware in which the neural network training were carried, their respective capacity and the advantages of each one of them.

6.1 Google Colab

Google colab is a free and online alternative for implementing deep learning techniques, since it provides an environment with python and allows the remote connection to one of Google's own GPU, allowing the user to speed up the testing of applications that include artificial intelligence by executing the code with the GPU instead of the users CPU. It is design specially for learning and can be easily connected with Google Drive.

Both Neural Networks were first implemented on Google Colab and all the testing was possible there. The GPU that are available and some important specifications are the following:

- NVIDIA Tesla K80 with 12GB of RAM memory
- NVIDIA Tesla P100-PCIE with 16GB of RAM memory
- NVIDIA Testa T4 with 15 GB of RAM memory

Although sometimes very useful there are limitations to this service. For example, which GPU is chosen is not up to the user, but is automatically assigned by the online platform itself. Since the main idea of this google service is to make it available for everyone, the use of a GPU is not guaranteed if you have been using one frequently. Additionally, no more than 12 hours of continuous use is possible, limiting training times.

6.2 NVIDIA GEFORCE RTX 2080 Ti

Once all the testing was done in Google colab, next, a new testing and training was performed on a local supercomputer available at the PIC4SeR facilities. It have a two ram memories, one with 12GB and the other one with 8GB, which made possible the use of higher batch sizes for the training of the Highly complex OCR+HRNet network. Different working environments were set and a more extensive training was performed to make sure that the results obtained in google colab were reproducible.

Chapter 7

Experimental test and results

In this section, a comparison between the results of both neural network is done and a qualitative and quantitative assessment of them is also presented. For the final tests, both networks were trained with the same training, validation a testing dataset, in a distribution of 682, 170 and 148 images respectively. The evaluation metric, as is common for many segmentation task, is chosen to be the IoU.

7.1 Training, fine tuning and quantitative assessment

7.1.1 Resnet50-Unet

Transfer learning was used to retrain an neural network already specialized in segmentation to adjust it to this specific problem. In this case, the starting weights of the Resnet50-Unet (From now on, only referred to as Unet) were obtained by the training done with the Imagenet dataset. Since the objective classes are only two, and the classes are different from those of the Imagenet dataset, a reshaping of the last layer is due.

To further improve the results and speed the training, the original 800x800 pixel images were cropped to a size of 256x256, and one hot encoded masks were calculated (see figure 7.1)

A summary of the final training parameters is shown in table 7.1

The convergence of the network is relatively fast, reaching its maximum value at 15 out of the 16 epochs, so an early stopping is done.

Figure 7.1: Original image, ground truth and one-hot enconded mask

Batch size	16
Number of epochs	16
Activation function	sigmoid
Loss Function	Dice Loss
Optimizer	Adaptative Moment Estimation (ADAM)
Data augmentation	Random 90 degrees rotation, horizontal flip, vertical flip, random crop
learning rate	0.0001
RAM consumption	4 GB

Table 7.1: Training parameters for the Resnet50-Unet

7.1.2 HRNet+OCR

Similarly to the Unet, transfer learning was used to take advantage of a previously trained model. In this case, the model wheights were taken from the training done with the Cityscapes dataset, that ranked first in the semantic segmentation category for Cityscapes for more than 2 year from 2019. The parameters used for the training are presented in table 7.3

The training of this network was performed using two different versions of the MMsegmentation package because of the differnt CUDA versions of the used GPU's. The training on the Google Colab Plataform used the version 0.26.0 while the training on the NVIDIA RTX 2080 Ti was done with version 0.22.0. This change alone could potentially change the results obtained. The configuration file that fully describes the training details for the HRNet+OCR are present in appendix A.

Batch size	4
Number of epochs	58
Activation function	ReLu
Optimizer	Adaptative Moment Estimation (ADAM)
Data augmentation	Random crop, random flip and photometric distortion
Learning rate	0.00001
Learning rate reduction factor	0.05 at each iteration
Learning rate ending value	0.000002 at epoch 58

Table 7.2: Training parameters for the OCR+HRNet

7.1.3 Quantitative Results

The metric used for this work is the Intersection over Union, that is also preferred in many studies for semantic segmentation. It is deifined as:

$$IoU = \frac{TP}{TP + FP + FN} \tag{7.1}$$

Where TP stands for True Positives, FP stands for False Positive and FN stands for False Negatives. This metric tends to penalize single instances of misclassification more than others. The validation curves for both of the networks are shown in figures 7.2 and 7.3 for the Unet and the OCR+HRNet respectiveley. The validation curve for the Unet shows a smooth convervenge to the maximum

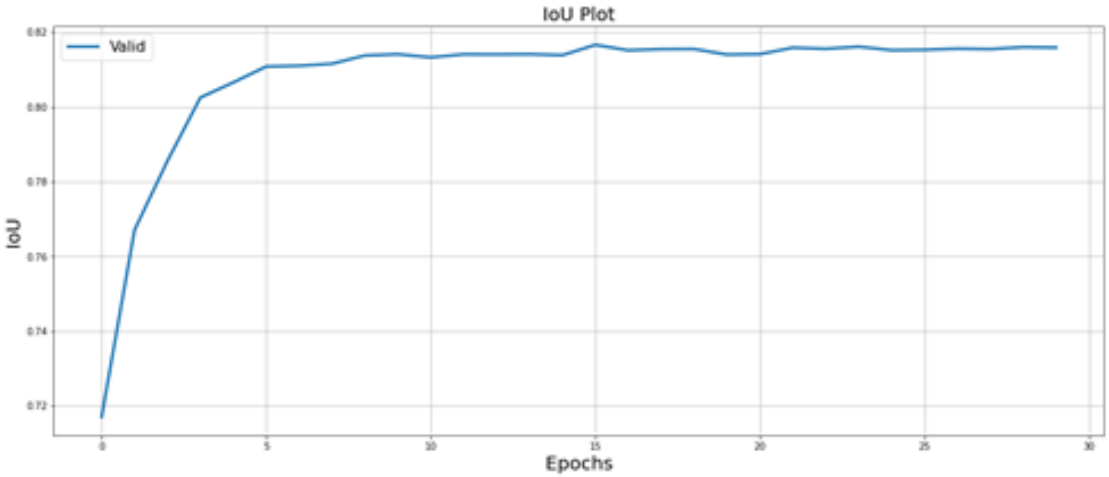

Figure 7.2: Unet Performance. Maximun peak at 81.66

value, reached at 81.66. The curve for the OCR+HRNet instead, shows some

fluctuations aven at the last epoch, however, the shape is indicative that the model is learning from the data, and that if the data was improved, the results could possible improve as well. The maximum peak that the OCR+HRNet reached on the RTX 2080 Ti GPU was 85.48, however, the training in google colab reached a value of 86.67 for the IoU. The number of parameters and the inference time was also computed, even if they are not of much relevance, given that this is an offline application, however they show the complexity difference of the networks.

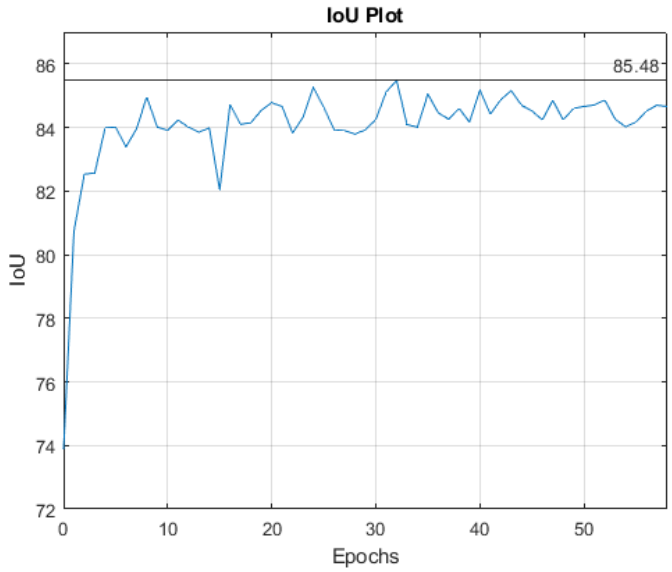

Figure 7.3: HRNET performance. Maximum peak at 85.48

Metric	Resnet50-UNet	OCR+HRNet
IoU	81.66	85.58
Number of parameters	32.52 M	70.37 M
Inference time	79.22	239.57

Table 7.3: Performance Comparative

7.2 Qualitative comparative

To see the impact of the improvement of the HRNet+OCR with respect to the traditional Unet, images on the test dataset are compared:

7.2.1 Close distance images

Lets consider as "Close distance images" those images where the vine-row width, independently of the growth state of the vine plant, occupies 8 or more pixels. Under this conditions, the HRNet+OCR network performs specially well, being capable of easily identifying even vine plants whose vine rows are not visible from start to end. Since a close view of the vineyard means either a very high resolution or a very low flight height, it is normal that non-vineyard objects are clear as well, and the network does not mistake them for vineyards. The Unet Has a pretty acceptable performance as well, even if the recognition of the plants at the borders is not could be better. Figures 7.4 and 7.5 show the predictions of both networks on the same close image.

Figure 7.4: Resulting predicted mask on a close distance image with the UNet

In this comparative, it can be seen the significance of the higher IoU that the OCR+HRNet achieved, since its output has less misclassifications and the performance at the borders of the image is better. The higher accuracy of the network can be seen also in the fact that in the lower vineyard of the image, the Unet classified mostly the darkest part of the vineyard, while the HRNet+OCR correctly

Figure 7.5: Resulting predicted mask on a close distance image with the HRNet+OCR

assessed the size of the crop. For both cases, the predictions are continuous straight lines that could be used as a grid occupancy map for autonomous navigation without much need of post-processing.

7.2.2 Medium distance images

Lets consider as "Medium distance images" those images where the vine-row width, independently of the growth state of the vine plant, occupies from 4 to 8 pixels. In this kind of images, both networks perform similarly and their results generally yield to a very precise identification of the vine rows, only missing one if it is very ambiguous whether or not it belongs to the vineyard class and very rarely mixing two rows, independently of the low contrast with the ground.

Examples of the predictions on the same image by both the HRNet+OCR neural network and the UNet are presented in figures 7.6 and 7.7

The main differences in this case are shown in a red box in figure 7.7, where two mistakes were made by the weaker network: the top of an electricity pole was marked as belonging to the vineyard class, and two vine rows were merged together. As mentioned before, in this resolution it can still be seen that the HRNet+OCR makes has less false positives and correctly classified both of the above mentioned situations. It is worth mentioning that in neither case, the shadow of the house near

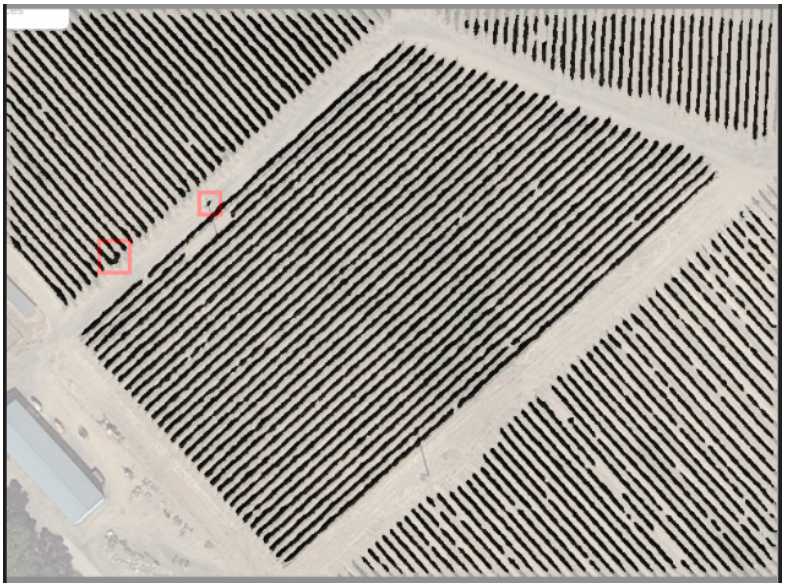

Figure 7.6: Resulting predicted mask on a medium distance image using the Unet

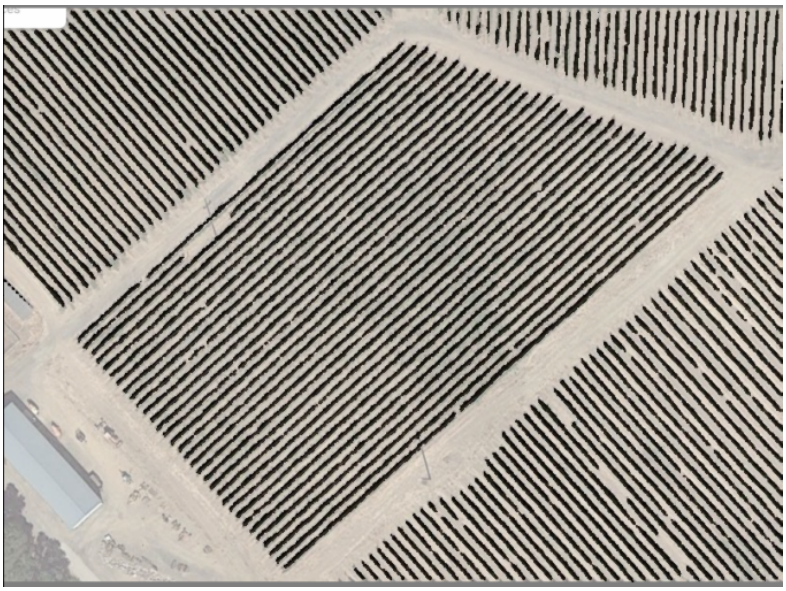

Figure 7.7: Resulting predicted mask a medium distance image using the HR-Net+OCR

the vineyards was mistaken. Again, for both cases, the predictions are continuous straight lines that could be used as a grid occupancy map for autonomous navigation without much need of post-processing.

7.2.3 Long distance images

Lets now consider as "Long distance images" those images where the vine-row width, independently of the growth state of the vine plant, occupies from 1 to 3 pixels. This type of images proved to be very challenging to correctly predict by both the networks, which is reasonable given the small margin that such small vine rows give. For the HRNet+OCR, the lowest performing images where of this kind

Examples of the predictions on the same image by both the HRNet+OCR neural network and the UNet are presented in figures 7.9 and 7.8 respectively

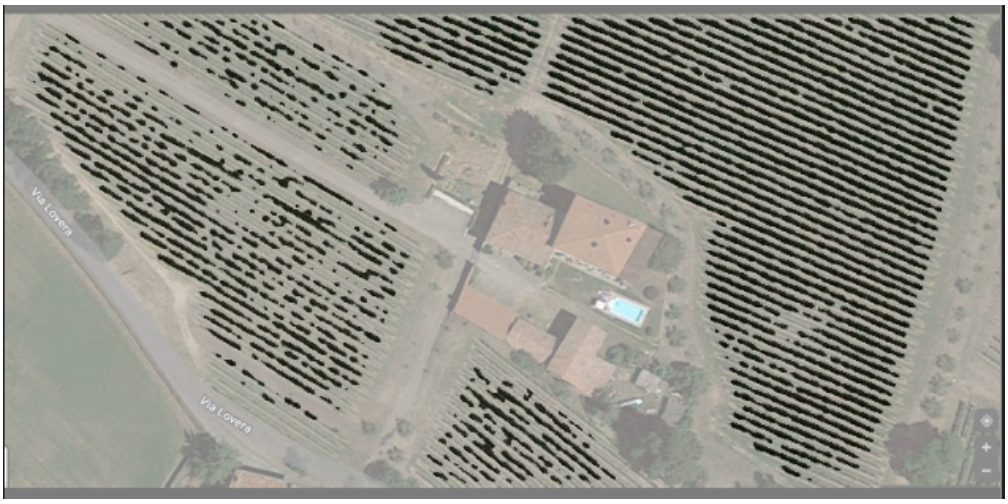

Figure 7.8: Resulting predicted mask on a long distance image using the Unet

The general performance in these kind of images is rather low. In most cases, the OCR+HRNet is not able to identify neatly the vine rows, producing instead of almost straight lines, wobbly ones that in most of the cases are not continuous. It is important no notice that false positive are uncommon even in images taken at this height, but the false negatives are unacceptably high.

The predictions of the UNet are qualitatively better in some of these cases because of one reason only: Even if some parts of the some vineyards are missing, the segmented ones are more alike to a straight line.

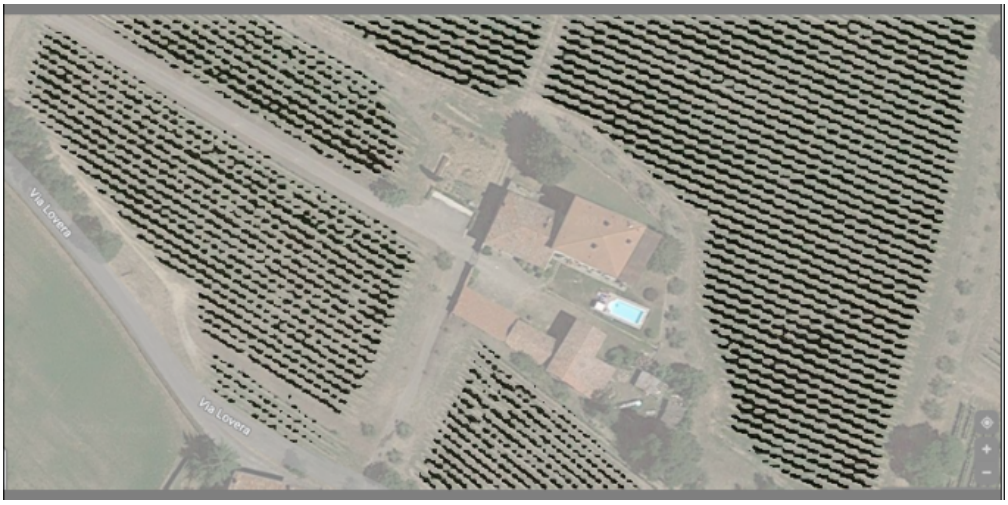

Figure 7.9: Resulting predicted mask on a long distance image using the HR-Net+OCR

7.2.4 Non-straight vineyards

Up until now, most of the research has been focused on vineyards planted in straight lines. In this work, some images of non-straight vineyards were included to provide a starting point in this field. The main reason it is possible to include them in this study is that the problem is similar in every other aspect except for the geometry of the vine rows. That is to say that all that the other information that the neural networks have learned about recognizing vineyards from aerial images (Texture of the vine plant, parallel vine rows, contrast with the ground, illumination conditions, etc.) is valid. An test of segmentation for non-straight vineyards is presented in figures 7.10 and 7.13.

The proposed image is interesting from different perspectives: It contains vineyards on different states of growth, with different orientations, alignments, ground contrast and field size

There is one important remark about the segmentation mask predicted for this image: Both the OCR+HRNet and the UNet were capable of identifying the vine rows, including the plants that make the curve shape of the vineyard. In this case, the Unet was able to recognize the more smaller vineyards, but incorrectly classified dark grass as a vine row.

Both segmentation masks of the central vineyard are clean and could be used as am occupancy grid map. It is demonstrated, then, that by taking a deep learning approach for vineyard segmentation on aerial image, path planning and autonomous

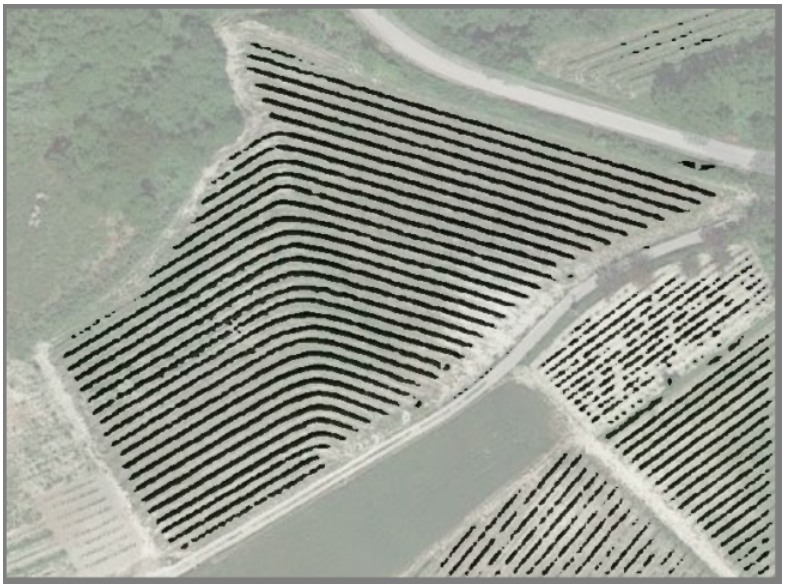

Figure 7.10: Resulting predicted mask on a non-straight vineyard image using the OCR+HRNet

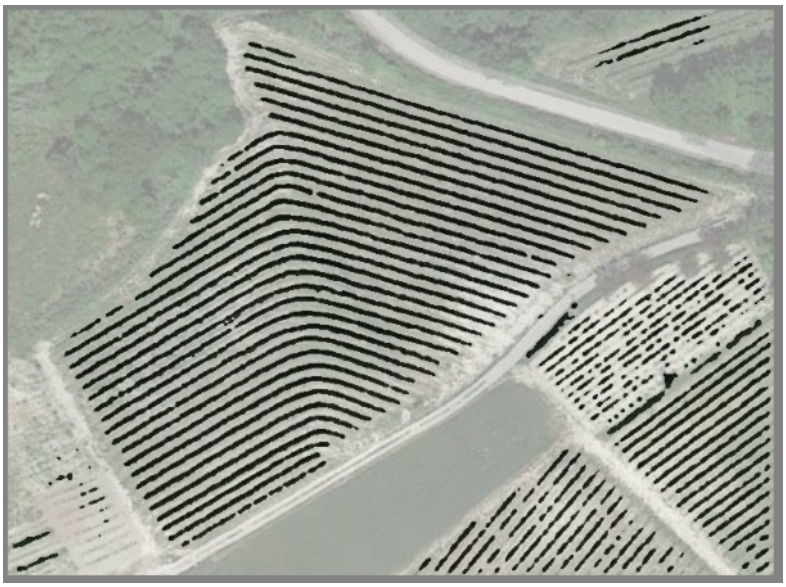

Figure 7.11: Resulting predicted mask on a non-straight vineyard image using the Unet

navigation could be possible even in non-straight vineyards.

7.2.5 General remarks

This section showed how the trained network performed on some images that were never shown to the neural networks during the training. In general, the cutting edge HRNet+OCR outperforms the traditional Unet, specially on close and medium distance images, mostly because of the few false positives its classifications has. On the other hand, the Unet can outperform the HRNet+OCR in particular scenarios, such as when the image spatial resolution is very low, like in long distance images, where the latter network cannot neatly identify the vine rows from one another as in . One possible explanation is that the training set has not enough images with a similar resolution, and being this a very complex network, it needs a bigger number of samples to learn. The test set contains much more low resolution images compared with the other sets, so a shuffle of samples between the test set and the train set could solve the problem and improve the obtained results.

7.3 Path planning with the predicted masks

Since the methodology showed that it is possible to obtain a clear mask for the vineyards in most cases, and one of the goals of this thesis is to apply these masks to applications in precision agriculture, a simple demonstration of how path planning could be carried in vineyards with the obtained masks as a starting point. For this purpose, a simple Dijkstra algorithm was implemented, manually introducing the starting and ending point. The task of selecting the waypoints as already discussed, can be automatically done in autonomous navigation by a neural network.

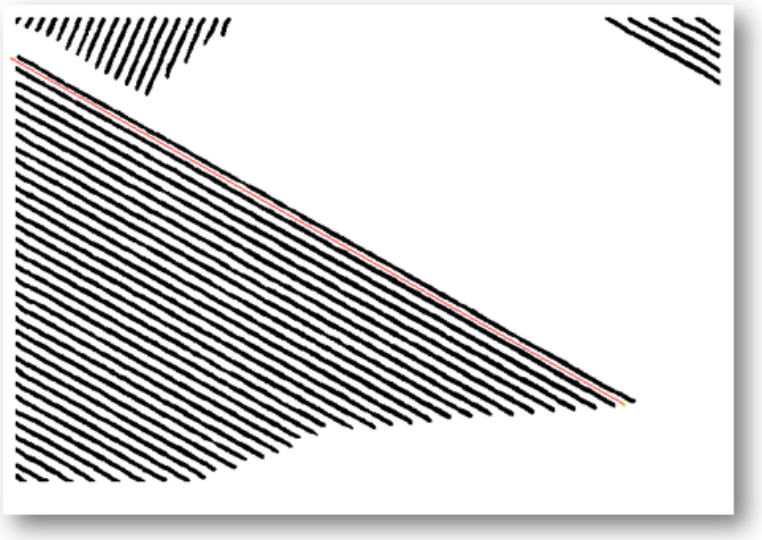

Figure 7.12: Path for inter-row navigation

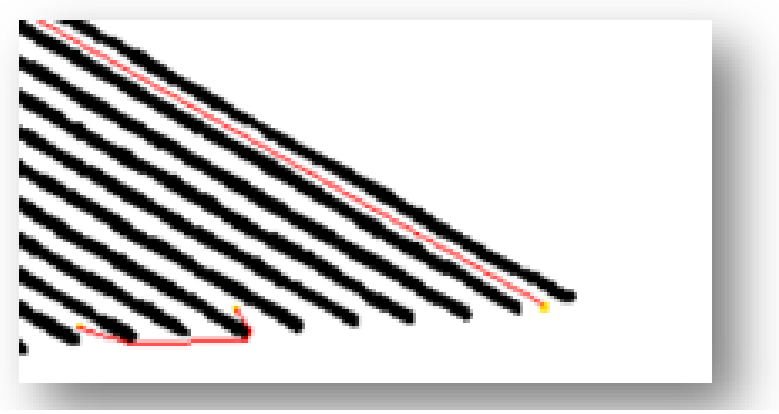

Figure 7.13: path for moving from one row to another

These path calculations were performed for the purpose of checking the fitness of the produced mask to be used as an obstacle map, an occupancy grid that allows path planning methods to be used. After all, the segmented vineyard image is a binary image.

Chapter 8

Conclusions and future work

8.1 Conclusions

In this thesis, a deep learning approach for remote sensing vineyard segmentation was presented. This work was divided in two main steps: The dataset generation and the neural network training.

The new dataset presented in this work consists on 1000 aerial RGB images of vineyards. Because the most challenging part of vineyard segmentation is the variability in the environmental conditions surrounding the crop, like grass presence or illumination conditions, the dataset was formed with the intentions of gathering an heterogeneous set of images, containing vineyard fields located mainly in the center, south and north of Italy. The dataset contains images taken with different camera resolutions, at different times of the year, whit many different contrasts between the vine plants and the soil. A methodology for semi-automatic labelling of the gathered images was also presented, and could serve as a reference for the generation of similar datasets in the precision agriculture field that lacks big public datasets to effectively carrying research.

With the created dataset, the training of two deep learning neural networks, the more traditional UNet with a resnet50 backbone and the state-of-the-art OCR+HRNet, was done. For training both networks, transfer learning was used to take advantage of already good performing models, retrained to fit the vineyard segmentation task. The results obtained for the UNet is in line with other state of the art studies, achieving an 81.66 average IoU score, result improved by almost 5 points by the more complex OCR+HRNet.

Specific situations were presented to assess the difference in the performance

of both networks. They showed that for images featuring vineyards at a close to medium distance, the OCR+HRNet outperforms the UNet and creates more clean masks, with much less pixels misclassified as vineyards. The UNet, however, proved to be competitive in images where the spatial resolution is very low, being able to create more clean vine row masks. Both neural networks were also tested for non-straight vineyards with very positive results. This methodology for vineyard segmentation on remote sensed image can, therefore, be used in precision agriculture applications, like autonomous navigation or field monitoring.

8.2 Future work

The presented methodology is an off-line one, which means that the images are processed much more time after they are taken. Future improvements of this work include the implementation of on-line methods, able to do the segmentation as the images are being taken. Depending on the application, the automatic calculation of vegetation indexes or the computing of waypoints to navigate the segmented vineyard can also be considered.

Appendix A

Appendix

```
    Config:
norm_cfg = dict(type='SyncBN', requires_grad=True)
model = dict(
    type='CascadeEncoderDecoder',
    num_stages=2,
    pretrained='open-mmlab://msra/hrnetv2_w48',
    backbone=dict(
        type='HRNet',
        norm_cfg=dict(type='SyncBN', requires_grad=True),
        norm_eval=False,
        extra=dict(
            stage1=dict(
                num_modules=1,
                num_branches=1,
                block='BOTTLENECK',
                num_blocks=(4, ),
                num_channels=(64, )),
            stage2=dict(
                num_modules=1,
                num_branches=2,
                block='BASIC',
                num_blocks=(4, 4),
                num_channels=(48, 96)),
            stage3=dict(
                num_modules=4,
                num_branches=3,
                block='BASIC',
                num_blocks=(4, 4, 4),
```

```
                num_channels=(48, 96, 192)),
            stage4=dict(
                num_modules=3,
                num_branches=4,
                block='BASIC',
                num_blocks=(4, 4, 4, 4),
                num_channels=(48, 96, 192, 384)))),
    decode_head=[
        dict(
            type='FCNHead',
            in_channels=[48, 96, 192, 384],
            channels=720,
            input_transform='resize_concat',
            in_index=(0, 1, 2, 3),
            kernel_size=1,
            num_convs=1,
            norm_cfg=dict(type='SyncBN', requires_grad=True),
            concat_input=False,
            dropout_ratio=-1,
            num_classes=19,
            align_corners=False,
            loss_decode=dict(
                type='CrossEntropyLoss', use_sigmoid=False, loss_weight=0.4)),
        dict(
            type='OCRHead',
            in_channels=[48, 96, 192, 384],
            channels=512,
            ocr_channels=256,
            input_transform='resize_concat',
            in_index=(0, 1, 2, 3),
            norm_cfg=dict(type='SyncBN', requires_grad=True),
            dropout_ratio=-1,
            num_classes=19,
            align_corners=False,
            loss_decode=dict(
                type='CrossEntropyLoss', use_sigmoid=False, loss_weight=1.0)),
    ],
    train_cfg=dict(),
    test_cfg=dict(mode='whole'))
dataset_type = 'CityscapesDataset'
data_root = 'data/cityscapes/'
```

```
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
crop_size = (512, 1024)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations'),
    dict(type='Resize', img_scale=(2048, 1024), ratio_range=(0.5, 2.0)),
    dict(type='RandomCrop', crop_size=(512, 1024), cat_max_ratio=0.75),
    dict(type='RandomFlip', prob=0.5),
    dict(type='PhotoMetricDistortion'),
    dict(
        type='Normalize',
        mean=[123.675, 116.28, 103.53],
        std=[58.395, 57.12, 57.375],
        to_rgb=True),
    dict(type='Pad', size=(512, 1024), pad_val=0, seg_pad_val=255),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_semantic_seg'])]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(2048, 1024),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(
                type='Normalize',
                mean=[123.675, 116.28, 103.53],
                std=[58.395, 57.12, 57.375],
                to_rgb=True),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img'])
        ])
]
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type='CityscapesDataset',
```

```
        data_root='data/cityscapes/',
        img_dir='leftImg8bit/train',
        ann_dir='gtFine/train',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(type='LoadAnnotations'),
            dict(
                type='Resize', img_scale=(2048, 1024), ratio_range=(0.5, 2.0)),
            dict(type='RandomCrop', crop_size=(512, 1024), cat_max_ratio=0.75),
            dict(type='RandomFlip', prob=0.5),
            dict(type='PhotoMetricDistortion'),
            dict(
                type='Normalize',
                mean=[123.675, 116.28, 103.53],
                std=[58.395, 57.12, 57.375],
                to_rgb=True),
            dict(type='Pad', size=(512, 1024), pad_val=0, seg_pad_val=255),
            dict(type='DefaultFormatBundle'),
            dict(type='Collect', keys=['img', 'gt_semantic_seg'])
        ]),
    val=dict(
        type='CityscapesDataset',
        data_root='data/cityscapes/',
        img_dir='leftImg8bit/val',
        ann_dir='gtFine/val',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(
                type='MultiScaleFlipAug',
                img_scale=(2048, 1024),
                flip=False,
                transforms=[
                    dict(type='Resize', keep_ratio=True),
                    dict(type='RandomFlip'),
                    dict(
                        type='Normalize',
                        mean=[123.675, 116.28, 103.53],
                        std=[58.395, 57.12, 57.375],
                        to_rgb=True),
                    dict(type='ImageToTensor', keys=['img']),
                    dict(type='Collect', keys=['img'])
```

```
                ])
        ]),
    test=dict(
        type='CityscapesDataset',
        data_root='data/cityscapes/',
        img_dir='leftImg8bit/val',
        ann_dir='gtFine/val',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(
                type='MultiScaleFlipAug',
                img_scale=(2048, 1024),
                flip=False,
                transforms=[
                    dict(type='Resize', keep_ratio=True),
                    dict(type='RandomFlip'),
                    dict(
                        type='Normalize',
                        mean=[123.675, 116.28, 103.53],
                        std=[58.395, 57.12, 57.375],
                        to_rgb=True),
                    dict(type='ImageToTensor', keys=['img']),
                    dict(type='Collect', keys=['img'])
                ])
        ]))
log_config = dict(
    interval=50, hooks=[dict(type='TextLoggerHook', by_epoch=False)])
dist_params = dict(backend='nccl')
log_level = 'INFO'
load_from = None
resume_from = None
workflow = [('train', 1)]
cudnn_benchmark = True
optimizer = dict(type='SGD', lr=0.01, momentum=0.9, weight_decay=0.0005)
optimizer_config = dict()
lr_config = dict(policy='poly', power=0.9, min_lr=0.0001, by_epoch=False)
runner = dict(type='IterBasedRunner', max_iters=160000)
checkpoint_config = dict(by_epoch=False, interval=16000)
evaluation = dict(interval=16000, metric='mIoU', pre_eval=True)
```

Bibliography

[1] Australian Centre for Precision Agriculture. «A General Introduction to Precision Agriculture». In: (). URL: http://www.agriprecisione.it/wp-content/uploads/2010/11/general_introduction_to_precision_agriculture.pdf (cit. on p. 1).

[2] Simone Cerrato, Vittorio Mazzia, Francesco Salvetti, and Marcello Chiaberge. *A Deep Learning Driven Algorithmic Pipeline for Autonomous Navigation in Row-Based Crops*. 2021. DOI: 10.48550/ARXIV.2112.03816. URL: https://arxiv.org/abs/2112.03816 (cit. on p. 2).

[3] Vittorio Mazzia, Francesco Salvetti, Diego Aghi, and Marcello Chiaberge. «DeepWay: A Deep Learning waypoint estimator for global path generation». In: *Computers and Electronics in Agriculture* 184 (May 2021), p. 106091. DOI: 10.1016/j.compag.2021.106091. URL: https://doi.org/10.1016%2Fj.compag.2021.106091 (cit. on pp. 2, 46, 47).

[4] Mohammad Haji Gholizadeh, Assefa M Melesse, and Lakshmi Reddi. «A comprehensive review on water quality parameters estimation using remote sensing techniques». In: *Sensors* 16.8 (2016), p. 1298 (cit. on p. 6).

[5] Mutlu Ozdogan, Yang Yang, George Allez, and Chelsea Cervantes. «Remote sensing of irrigated agriculture: Opportunities and challenges». In: *Remote sensing* 2.9 (2010), pp. 2274–2304 (cit. on p. 7).

[6] Teja Kattenborn, Jens Leitloff, Felix Schiefer, and Stefan Hinz. «Review on Convolutional Neural Networks (CNN) in vegetation remote sensing». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 173 (2021), pp. 24–49. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2020.12.010. URL: https://www.sciencedirect.com/science/article/pii/S0924271620303488 (cit. on p. 8).

[7] Dimosthenis C Tsouros, Stamatia Bibi, and Panagiotis G Sarigiannidis. «A review on UAV-based applications for precision agriculture». In: *Information* 10.11 (2019), p. 349 (cit. on p. 11).

[8] Paul D Groves. «Principles of GNSS, inertial, and multisensor integrated navigation systems, [Book review]». In: *IEEE Aerospace and Electronic Systems Magazine* 30.2 (2015), pp. 26–27 (cit. on p. 12).

[9] Spyros Fountas, Nikos Mylonas, Ioannis Malounas, Efthymios Rodias, Christoph Hellmann Santos, and Erik Pekkeriet. «Agricultural robotics for field operations». In: *Sensors* 20.9 (2020), p. 2672 (cit. on p. 13).

[10] José A. Martínez-Casasnovas, M. Concepción Ramos, and Roser Cots-Folch. «Influence of the EU CAP on terrain morphology and vineyard cultivation in the Priorat region of NE Spain». In: *Land Use Policy* 27.1 (2010). Soil and Water Conservation Measures in Europe, pp. 11–21. ISSN: 0264-8377. DOI: https://doi.org/10.1016/j.landusepol.2008.01.009. URL: https://www.sciencedirect.com/science/article/pii/S0264837708000239 (cit. on p. 14).

[11] Chenghai YANG. «High resolution satellite imaging sensors for precision agriculture». In: *Frontiers of Agricultural Science and Engineering* 5.4, 393 (2018), p. 393. DOI: 10.15302/J-FASE-2018226. URL: https://journal.hep.com.cn/fase/EN/abstract/article_22425.shtml (cit. on p. 16).

[12] Luisa Sangregorio. «Estimating Depth Images from Monocular Camera with Deep Learning for Service Robotics Applications.» PhD thesis. Politecnico di Torino, 2022, pp. 18–29 (cit. on p. 19).

[13] Stefano A. Bini. «Artificial Intelligence, Machine Learning, Deep Learning, and Cognitive Computing: What Do These Terms Mean and How Will They Impact Health Care?» In: *The Journal of Arthroplasty* 33.8 (2018), pp. 2358–2361. ISSN: 0883-5403. DOI: https://doi.org/10.1016/j.arth.2018.02.067. URL: https://www.sciencedirect.com/science/article/pii/S0883540318302158 (cit. on p. 19).

[14] Alexander L. Fradkov. «Early History of Machine Learning». In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 1385–1390. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1888. URL: https://www.sciencedirect.com/science/article/pii/S2405896320325027 (cit. on pp. 20, 22).

[15] Jaime G. Carbonell, Ryszard S. Michalski, and Tom M. Mitchell. «1 - AN OVERVIEW OF MACHINE LEARNING». In: *Machine Learning*. Ed. by Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. San Francisco (CA): Morgan Kaufmann, 1983, pp. 3–23. ISBN: 978-0-08-051054-5. DOI: https://doi.org/10.1016/B978-0-08-051054-5.50005-4. URL: https://www.sciencedirect.com/science/article/pii/B9780080510545500054 (cit. on p. 22).

[16] M.W Gardner and S.R Dorling. «Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences». In: *Atmospheric Environment* 32.14 (1998), pp. 2627–2636. ISSN: 1352-2310. DOI: https://doi.org/10.1016/S1352-2310(97)00447-0. URL: https://www.sciencedirect.com/science/article/pii/S1352231097004470 (cit. on p. 27).

[17] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747 (cit. on p. 30).

[18] Haotian Zhang, Lin Zhang, and Yuan Jiang. «Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems». In: *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. 2019, pp. 1–6. DOI: 10.1109/WCSP.2019.8927876 (cit. on p. 34).

[19] Lorenzo Comba, Paolo Gay, Jacopo Primicerio, and Davide Ricauda Aimonino. «Vineyard detection from unmanned aerial systems images». In: *Computers and Electronics in Agriculture* 114 (2015), pp. 78–87. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2015.03.011. URL: https://www.sciencedirect.com/science/article/pii/S0168169915000861 (cit. on pp. 41, 42, 47).

[20] T. Barros, P. Conde, G. Gonçalves, C. Premebida, M. Monteiro, C.S.S. Ferreira, and U.J. Nunes. «Multispectral vineyard segmentation: A deep learning comparison study». In: *Computers and Electronics in Agriculture* 195 (2022), p. 106782. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2022.106782. URL: https://www.sciencedirect.com/science/article/pii/S0168169922000990 (cit. on p. 43).

[21] Mohamed Kerkech, Adel Hafiane, and Raphael Canals. «Vine disease detection in UAV multispectral images using optimized image registration and deep learning segmentation approach». In: *Computers and Electronics in Agriculture* 174 (2020), p. 105446. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2020.105446. URL: https://www.sciencedirect.com/science/article/pii/S016816991932558X (cit. on pp. 43, 45, 47).

[22] Luís Pádua, Telmo Adão, Jonáš Hruška, Nathalie Guimarães, Pedro Marques, Emanuel Peres, and Joaquim J. Sousa. «Vineyard Classification Using Machine Learning Techniques Applied to RGB-UAV Imagery». In: *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. 2020, pp. 6309–6312. DOI: 10.1109/IGARSS39084.2020.9324380 (cit. on p. 47).

[23] Chantal Hajjar, Ghassan Ghattas, Maya Kharrat Sarkis, and Yolla Ghorra Chamoun. «Vine Identification and Characterization in Goblet-Trained Vineyards Using Remotely Sensed Images». In: *Remote Sensing* 13.15 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13152992. URL: https://www.mdpi.com/2072-4292/13/15/2992 (cit. on p. 47).

[24] Giulia Ronchetti. «UAV SURVEYS FOR CROP MONITORING AND MANAGEMENT IN PRECISION AGRICULTURE». PhD thesis. 2019 (cit. on p. 47).

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: 10.48550/ARXIV.1505.04597. URL: https://arxiv.org/abs/1505.04597 (cit. on p. 55).

[26] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. «Segmentation Transformer: Object-Contextual Representations for Semantic Segmentation». In: (2019). DOI: 10.48550/ARXIV.1909.11065. URL: https://arxiv.org/abs/1909.11065 (cit. on p. 56).