Leonardo
Maia

Identificação de sítios arqueológicos em imagem aérea utilizando aprendizagem profunda - projeto ODYSSEY

Archaeological site identification on aerial imagery using deep learning - ODYSSEY project

Leonardo
Maia

Identificação de sítios arqueológicos em imagem aérea utilizando aprendizagem profunda - projeto ODYSSEY

Archaeological site identification on aerial imagery using deep learning - ODYSSEY project

"*The greatest challenge to any thinker is stating the problem in a way that will allow a solution*"

— Bertrand Russell

Universidade de Aveiro
2023

Leonardo
Maia

Identificação de sítios arqueológicos em imagem aérea utilizando aprendizagem profunda - projeto ODYSSEY

Archaeological site identification on aerial imagery using deep learning - ODYSSEY project

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações , realizada sob a orientação científica do Doutor António Neves, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e da Doutora Pétia Georgieva, Professora associada do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Amaro Fernandes de Sousa
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor António José Ribeiro Neves
Professor Auxiliar da Universidade de Aveiro

Prof. Doutor José Silvestre Serra da Silva
Professor Associado Com Agregação, Academia Militar - Departamento de Ciências e Tecnologia de Engenharia

agradecimentos / acknowledgements

I would like to thank all the people who made this dissertation possible. To all the professors who spent time after class explaining concepts often unrelated to the curricular unit.
To professor António Neves a special thanks for accepting to be my tutor, to professor Pétia Georgieva for all help and to Daniel Canedo for all help and advice.
To my parents who during all these years have made an effort so that I lacked nothing.
And finally, to all my friends, especially Guilherme Maniezo, who has accompanied me during these years, in good and bad moments.

Palavras Chave Arqueologia, Deep learning, LiDAR, Unet, YOLOv7

Resumo

Esta dissertação foi desenvolvida no âmbito do projeto ODYSEY, que visa o desenvolvimento de uma plataforma destinada a arqueólogos. Neste contexto, esta dissertação tem como objetivo a identificação de sítios arqueológicos a partir de imagens formadas através dos dados fornecidos por um sistema LiDAR. A área de estudo é o Alto Minho, uma sub-região portuguesa pertencente à região Norte, e famosa pela preservação de estruturas históricas. Este trabalho centra-se no estudo das mamoas, que são construções de pedra e areia que teriam a função de esconder e proteger sepulturas, e dos castelos, que são construções urbanas da Idade do Cobre e da Idade do Ferro. De uma forma clara, o objetivo é elaborar um sistema capaz de localizar estes objectos históricos a partir de uma imagem aérea. O trabalho vai desde a criação da base de dados, a implementação de modelos de deep learning, até a inferência dos resultados.

Keywords Archaeology, Deep learning, LiDAR, Unet, YOLOv7.

Abstract

This dissertation was developed within the ODYSEY project, which aims to develop a platform intended for archaeologists. Within this context, this dissertation aims to identify archaeological sites from images formed through the data provided by a LiDAR system. The study area is Alto Minho, a Portuguese sub-region belonging to the Northern region, and famous for the preservation of historical structures. This work focuses on the study of tumuli, which are buildings of stone and sand that would have the function of hiding and protecting graves, and the hillforts, which are urban constructions of the Copper Age and Iron Age. In a clear way, the goal is to elaborate a system capable of locating these historical objects from an aerial image. The work ranges from the creation of the database, the implementation of deep learning models, to the inference of the results.

Contents

List of Figures

List of Tables

Glossary

YOLO	You Only Look Once
CNN	Convolution Neural Network
LOF	Local Outlier Factor
DTM	Digital Terrain Model
LRM	Local Relief Model
mAP	mean Average Precision
LiDAR	Light Detection and Ranging
ACH	archaeological and cultural heritage
MSTP	Multi Scale Topographic Position

CHAPTER 1

Introduction

This chapter provides an introduction to the topic of the dissertation. It describes the problems that were tried to be solved and the motivation behind them.

1.1 Motivation

Cultural and archaeological heritage is one of the most important vehicles of cultural diversity and a way of preserving our history so that we can learn from the past and improve the future. Archaeology provides tools to understand how societies functioned and why the world and human society have changed.

Discovering a site rich in historical artifacts is every archaeologist's dream. Revealing to humanity a piece of its history that has been hidden for centuries or millennia is an arduous task that can result from successive trials and errors. To mitigate the costs associated with this activity, investments are being made in the development of artificial intelligence technologies[1] capable of indicating, with significant accuracy geographical regions where archaeological objects are more likely to be found. In this way, the excavation team can work more efficiently, saving time and money.

One of the reasons for this dissertation is the fact that the most common collection of archaeological information consists of walks in large fields. The ODDYSEY project consists in automating this process, using methods that have been increasingly used in archaeology in recent years, such as artificial intelligence and Light Detection and Ranging (LiDAR), which is encouraged by UNESCO and scientific communities[2], including all remote sensing techniques. Remote sensing is the name given to all non-invasive techniques that use non-direct contact to observe targets of interest, either on the surface of the earth or from below, one of which is LiDAR.

LiDAR is a technology that uses light lasers to measure the distance of objects, allowing the creation of 3D models of the environment. By measuring the time it takes for the light to travel to the object and back, LiDAR can accurately determine the distance to the object.

Another technique that has been used extensively with remote sensing in recent years is artificial intelligence, specifically convolutional neural networks (CNN). This technique allows the algorithm itself to learn through training the necessary filters to extract relevant features from images, which can be automatically adapted to different types of data and contexts. In addition, convolutional neural networks are capable of handling high-resolution and complex images, enabling higher accuracy in real-time object detection and classification. This makes remote sensing even more effective.

The utilization of artificial intelligence tools has experienced a rapid surge in recent years[3]. Various techniques, such as classical random forest machine learning, have been employed to identify archaeological mounds in regions like Mesopotamia, Pakistan, and Jordan [4]. The adoption of deep learning techniques has seen a notable increase in conjunction with LiDAR technology, likely due to the prevalence of tubular structures with archaeological significance worldwide [4]. Among the subjects investigated in this dissertation are tumuli, which are also characterized by their tubular and uniform nature.

1.2 Objectives

The objective of the ODYSSEY project, funded by the COMPETE - Competitiveness and Internationalization Thematic Operational Program and the Regional Operational Programs, in its FEDER component, within the PORTUGAL2020 program, is to develop an integrated geographic information platform for archaeologists and heritage technicians, which allows the consolidation of different sources of heritage information, whether public or private, or directly from the field for specific purposes, using non-intrusive methods (e. g, LiDAR, multispectral imaging) and automate their further processing to identify archaeological elements and produce complementary information.

The project is divided into two parts, the pre-processing of LiDAR data and the detection of archaeological sites. In this dissertation only the detection part was done, having already access to the processed data, in the form of 4 Local Relief Model (LRM) images, each representing a sub-area of Alto Minho.

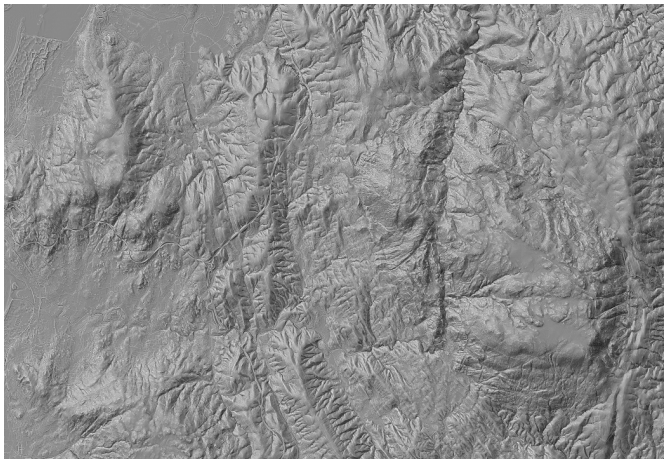

Figure 1.1: Example of a region from the LRM image of Viana do Castelo

To achieve this goal, the 4 LRM images were processed to create a dataset to train the two chosen deep learning models, You Only Look Once (YOLO) and Unet, and augmentation techniques were applied. Finally, a Local Outlier Factor (LOF) model was trained on the LiDAR data to validate the results.

Finally, full inference was performed on all 4 LRM images in an attempt to find undiscovered archaeological sites.

1.3 Dissertation Structure

The dissertation is divided into 6 chapters. In the first chapter an introduction is given to the topic and the motivation behind it. In the second chapter, the method of obtaining archelogical data is discussed. In the third chapter, it is explained how the datasets were created. Chapter 4 is about YOLOv7 and how it was used. Chapter 5 is about Unet and segmentation and finally chapter 6 is a conclusion.

CHAPTER 2

Deep learning in archaeology

In this chapter, it will be discussed various ways of applying artificial intelligence models to archaeology.

Remote sensing is the technique of obtaining information about an object, an area, or a phenomenon without direct contact between the sensor and the object or area being observed.

Its use dates back to the early 20th century[2], and it was then used heavily for military purposes during World War I (1914-1918)[5], when airplanes were equipped with cameras. The information recorded in the photographs was used for target recognition and military planning. Later, this type of technique was widely spread in civil society, where it was used for various purposes.

In the field of this dissertation, its use is for the purpose of indicating geographical regions likely to contain archaeological objects.

2.1 Airborne and Spaceborne Remote sensing

Preservation of archaeological and cultural heritage (ACH) is a critical strategic priority aiming to protect and transmit cultural artifacts and historical evidence to future generations.

Remote sensing from elevated points was initially used in the field of ACH[2], marking one of its pioneering applications. In the field of ACH, remote sensing usually refers to all techniques that use non-direct contact to observe areas of interest on Earth.

More than a century has passed since the inception of airborne remote sensing in the field of ACH around 1900. Dating back to 1839, with the birth of photography, photographers had attempted to lift cameras to capture aerial perspectives of the Earth's surface.

In 1906, a British general used a military hot air balloon[2] to take vertical and oblique photographs of the Stonehenge site. After the 1900s, the majority of aerial photographs were taken from airplanes, revolutionizing the perspective from which images were captured above the Earth's surface.

2.1.1 Photography

The use of photography in archaeology has become more common in the last 110 years[2], as one of the methods that allows the discovery of archaeological sites after they have been destroyed, either by man or by natural phenomena.

There are traces of ancient, man-made transformations of the landscape that can only be seen from above, and this is where photograrchaeology comes in.

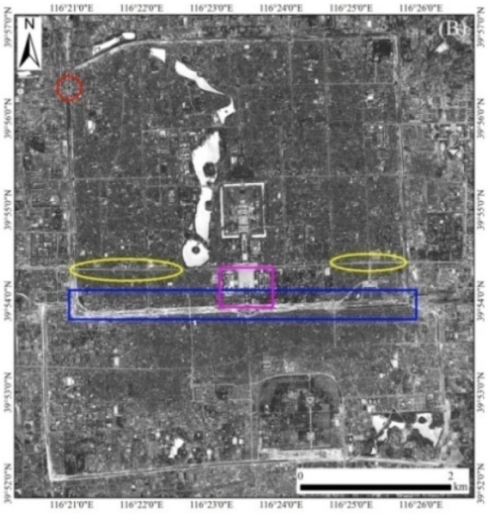

Figure 2.1: Aerial photograph in 1945 of Beijing [2]

2.1.2 Hyperspectral and multispectral image

Unlike photography, which uses only frequencies within the visible range, hyperspectral and multispectral imaging uses a wider range of frequencies that are not visible to the naked eye.

Multispectral has wavelengths usually between the visible and infrared and typically uses 3 to 15 bands with a length greater than 20 nm. An example of the use of multispectral imagery is the Landsat-8 satellite, which produces 11 images from different bands, all but 3 of which have a resolution of 30 meters.

The images 2.2 and 2.3 show the different bands used in Landsat-8 imagery.

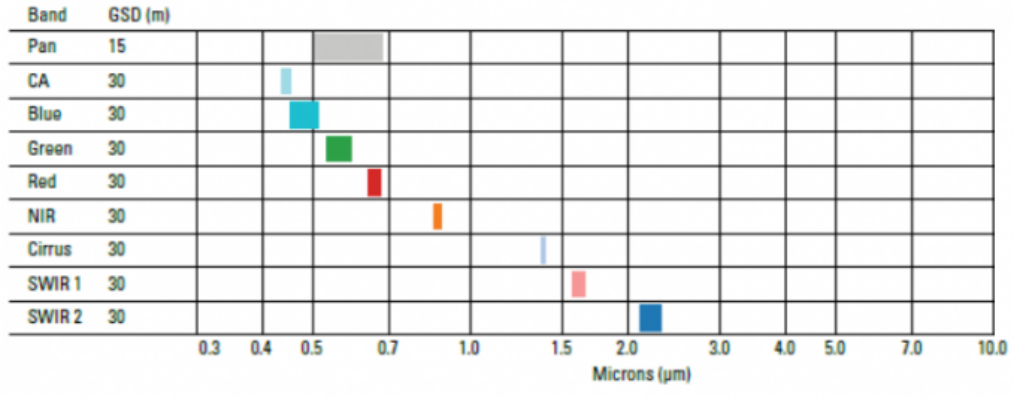

Figure 2.2: Landsat-8 bands from the OLI sensor[6]

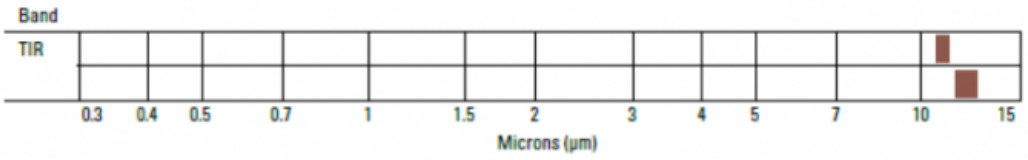

Figure 2.3: Landsat-8 bands from the TIRS sensor[6]

Hyperspectral imaging, on the other hand, typically uses hundreds or thousands of narrower bands (typically smaller than 20nm).

These fundamental differences between multispectral and hyperspectral imaging have a major impact on archaeological prospecting, since the more and narrower the bands, the greater the ability to detect subtly different spectral reflections in the ground.

It is also worth mentioning that buried archaeological objects can have their chemical, physical and biological properties altered, causing differences in spectral reflectance.

2.1.3 LiDAR

LiDAR has become an essential part of archaeological prospecting. By attaching it to a drone, for example, it is possible to cover large areas in order to discover new sites with archaeological potential. Its operation is based on emitting high-frequency light beams and then measuring the time it takes for the light beam to reach the object and return. By analyzing the characteristics of the returned signal, it is possible to create a 3D map of the area in question.

The image 2.4 shows an example of a point cloud from an area in Viana do Castelo.

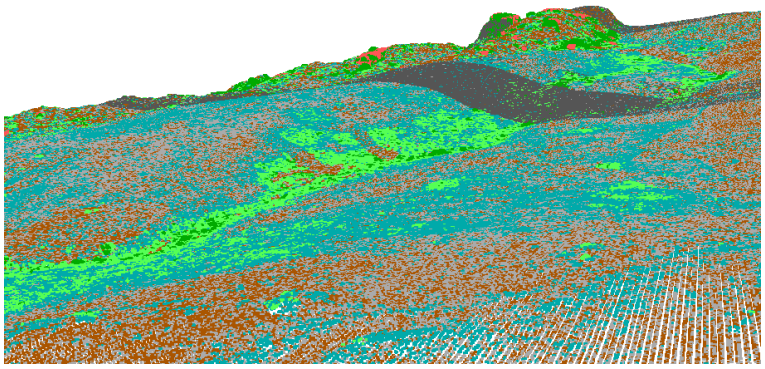

Figure 2.4: Point cloud from Viana do Castelo

In the image each color of the point represents the category the point belongs to - for example green represents vegetation and brown represents soil.

As a result of the work of archaeologists and specialists, airborne lidar has been successfully used to detect archaeological features around the world. However, in order to be useful, it is first necessary to classify and fit the points in the point cloud, which is a crucial process to identify and remove points that do not belong to the ground in order to be able to detect archaeological features. The result of this filtering is called a Digital Terrain Model (DTM). However, it should be noted that this method is not perfect and it is possible to filter too much, in this case filtering points that belong to archaeological objects.

The image 2.5 shows the same zone but filtered showing only the points classified as soil.

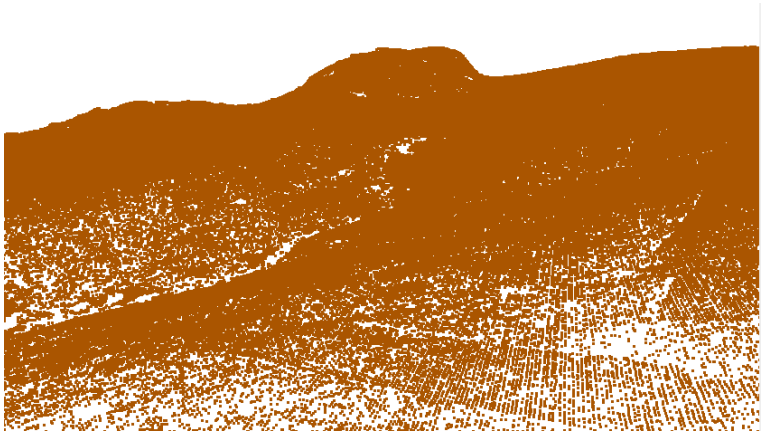

Figure 2.5: Point cloud from Viana do Castelo filtered

2.2 Relief Visualization Techniques

For the implementation of deep learning algorithms, one of the possibilities is to use directly the filtered Lidar radar data, i.e. to use the DTM, as it is shown in the article [7]. However, deep learning methods may have certain limitations, as many human resources are needed to make the annotations. In the case of archaeological annotations, if done by

non-professionals, they may contain errors due to lack of knowledge of the scene and structure of the object in question.

Another option is to pre-process the point cloud to create an image with the relief information. There are several techniques to create such images. In the article[8] 13 methods are listed. However, we will only discuss the two most effective according to the article, e2MSTP and MSTP, and the LRM used in the thesis.

2.2.1 MSTP and Enhanced MSTP

One of the most efficient techniques used is Multi Scale Topographic Position (MSTP). As explained in [9], it works well for visual interpretation and semi-automatic feature detection. By focusing on the topographical context of the structures rather than the structures themselves, it is particularly good for heterogeneous objects. Enhanced MSTP adds a morphological visualization technique to MSTP, creating a more detailed image.

The image 2.6 shows an MSTP image, a morphological visualization, and the result of combining the two, the enhanced MSTP image.

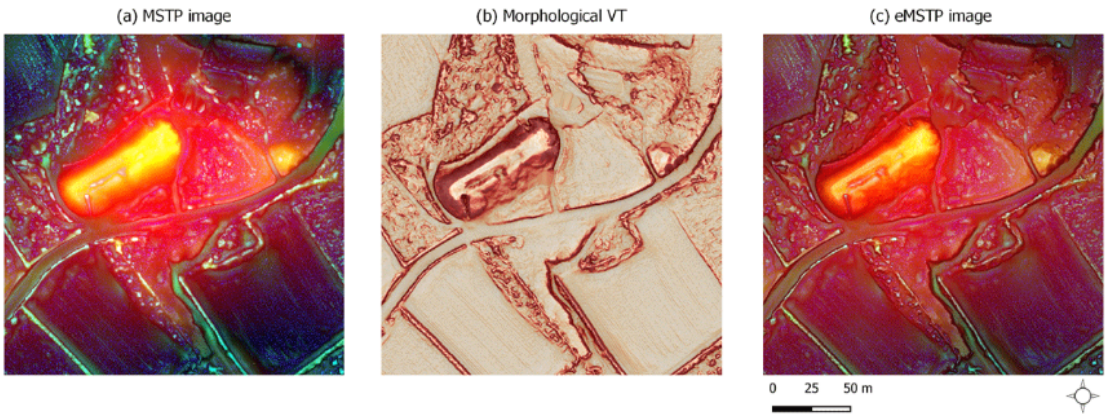

Figure 2.6: MSTP and Enhanced MSTP [10]

It is important to note that this technique does not provide topographical measurements since the RGB values do not represent elevations or slopes.

According to [10], enhanced MSTP allowed researchers to obtain a detection accuracy of 77%, proving thia to be a good technique especially if combined with the right deep learning model.

2.2.2 Local relief model (LRM)

Local Relief Model (LRM) is one of the most widely used techniques for archaeological purposes. In LRM small-scale elevation differences are represented after the landscape is removed from the data, as in DTM.

The LRM significantly improves the detectability of shallow, small-scale topographic features, regardless of the lighting conditions. It enables direct measurement of both their relative heights and volumes[11].

When a low-pass filter is applied to the DTM, it results in a smoothed elevation model, which serves as an initial representation of the large-scale landscape forms. The size of the kernel used in the low-pass filter determines the spatial scale of features captured in LRM.

2.3 Deep learning VS Machine Learning

This section explains how it is possible to apply both deep learning and machine learning methods to archaeology.

Since LRM imaging is one of the most widely used techniques in archaeology, a logical next step is to apply deep learning methods to the detection of new archaeological sites. These methods can be objection detection or semantic segmentation. Both methods are based on CNN (Convolutional Neural Network).

2.3.1 Convolution Neural Network

Convolution Neural Network (CNN) is one of the most recurrent forms of deep learning because it is based on the human brain. Unlike classical machine learning, where it is necessary to first collect features from the dataset and then feed them into the machine learning algorithm, CNNs are also able to train filters that can extract these features.

The operation is based on an input layer, where the image passes through several convolution layers. The convolution layer applies a set of filters or kernels to the input image, each of which extracts a specific feature from the image. For example, one filter might detect horizontal edges, another filter might detect vertical edges, and so on. The filter is moved over the input image, performing a point-to-point multiplication at each position, and the average of the sums of these multiplications is calculated to produce an output value at a given position. At the end, a feature map is created.

Between each convolution layer, it is normal to have a polling layer, which is used to reduce the size of the feature map to reduce the number of features. This layer divides the image into rectangular regions, and then for each region, simply takes the highest value or averages the values. After the last max-poling layer come the dense layers, but for this you need to transform the image from matrix to vector, and for this a flatten layer is used.

The dense layers, also known as fully connected layers, are used to process the data from the previous layers.

Finally, we have the output layer, which can be a sigmoid if the result is binary, or a softmax if it is not.

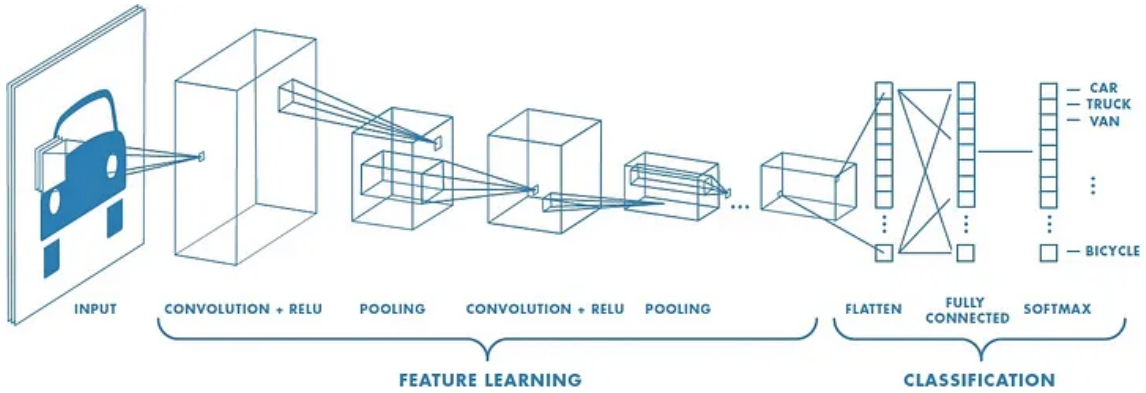

Figure 2.7: Example of neural network [12]

2.3.2 Machine Learning

The main difference between deep learning and machine learning lies in the architecture and the level of complexity in the learned tasks. One notable distinction is the degree of human involvement. In deep learning, the model learns to extract features from the data on its own, while in traditional machine learning, humans need to manually extract features from the data and then provide them as input to the algorithm.

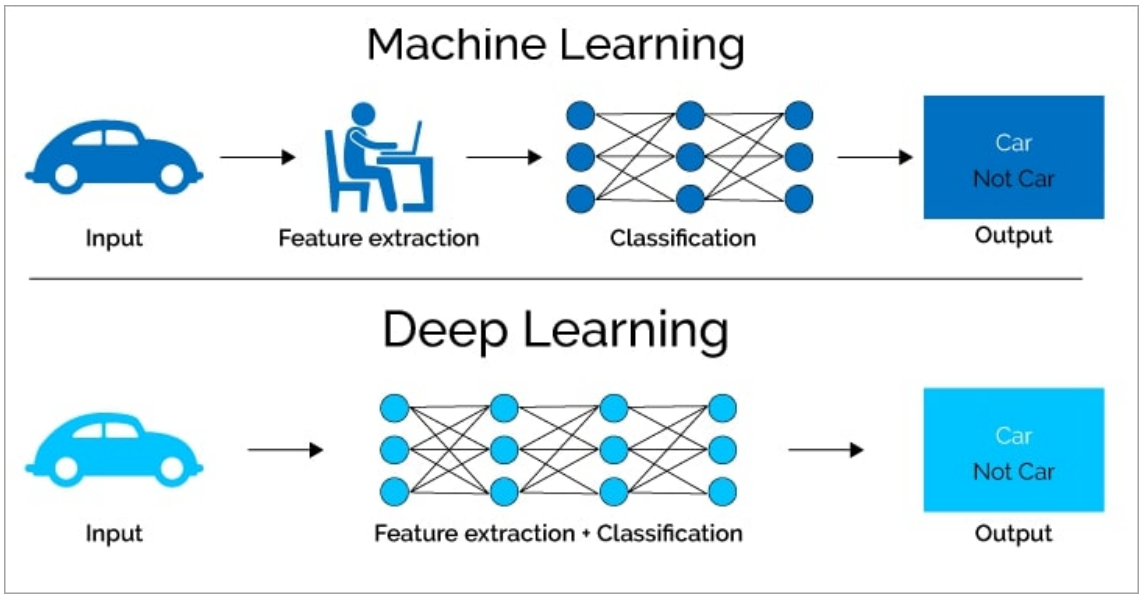

Figure 2.8: Machine Learning VS Deep learning[13]

The use of machine learning methods in archaeology using the LiDAR data without preprocessing is a technique that has been used. The unsupervised clustering based machine learning methods are more suitable for irregular object[7].

When it comes to cultural heritage, surface or structural defects often appear in the form of irregular geometric features. These can include weathering, cracks, and partial defects. As a result, these algorithms are exceptionally well suited for identifying and extracting surface defects in such cases[7].

CHAPTER 3

Dataset preparation

In this chapter, it will be discussed various ways of applying artificial intelligence models to archaeology.

Deep learning is a subtype of artificial intelligence, which relies on neural networks in order to recognize patterns in the given training data. When trained on a dataset, the network is able to learn many features about it, making it capable of making predictions when presented with new data.

The region of Alto Minho, Portugal was chosen for the research presented in this dissertation. With the help of the Comunidade Intermunicipal do Alto Minho[14], LiDAR data of 2018 was collected, covering 2220 km2, to which later was applied the visualization technique, more specifically LRM. As a result of this process 4 LRM images were generated, corresponding to 4 sub-regions of Alto Minho: Viana do Castelo, Paredes de Coura, Arcos de Valdevez and Parque Nacional da Peneda-Gêrez, having a correspondence of 0.5 meters on the terrain per pixel. From these 4 sub-regions, the locations of all known tumuli and hillforts were also provided.

The table 3.1 shows the resolution of each image, as well as the number of known annotations and the image size.

Region	Resolution(px)	Annotations (tumulis)	Annotations (hillforts)	Size (GB)
Viana do Castelo	19,978x46.000	14	41	3.7
Paredes de Coura	13,999x51,999	56	65	2.9
Arcos de Valdevez	15,999x43,999	71	46	2.8
Parque Nacional da Peneda-Gerês	19,999x44,955	135	11	3.6

Table 3.1: Description of the dataset used

The image 3.1 shows the joint of the 4 LRM images, corresponding to the entire Alto Minho area where LiDAR data was collected.

Figure 3.1: The 4 LRM images together, corresponding to Alto Minho

In the image 3.2, there are two examples of annotations of tumuli on the left side and hillforts on the right side, using the QGUIS software.

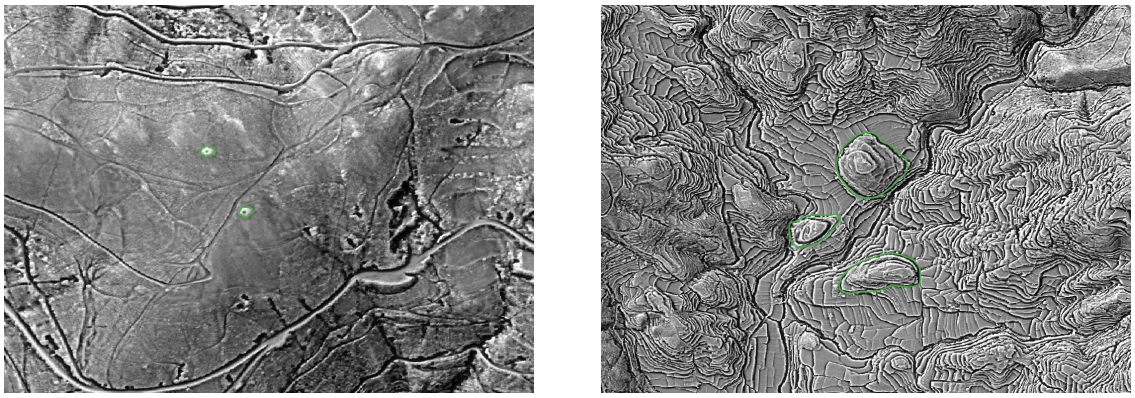

Figure 3.2: Example of annotations of tumuli on the left side and hillforts on the right side, using QGUIS

3.1 Creation of the base dataset

Along with the images the annotations were also provided in the Well-Know Text (WKT) format. This format indicates in polynomial form the geographical location of the site in question.

However, as it is possible to see from the table 3.1, the 4 LRM images are too large, both for training the YOLOv7 model and the Unet model. Because of this, the 4 images were processed in order to obtain a dataset with 640x640 pixels images, being a typical size to train both YOLOv7 and Unet.

For this, a Python script was used. In this script, one LRM image was loaded at a time, as well as its annotations. To simplify it, separate datasets were generated for the tumuli and for the hillforts. Then, the annotations were parsed, and using the LRM image metadata, the geographic coordinates of the image extremities were obtained. With that, the coordinates

present in the annotation polynomials were converted to pixels using the mapping function, as illustrated below:

$$pixel = \frac{(coordenate - inMin) * (outMax - outMin)}{(inMax - inMin)} + outMin \tag{3.1}$$

In the function, inMin and inMax represent the geographic coordinates of the LRM image extremities, while outMin and outMax represent the size of the image in pixels. However, the y coordinate counts from top to bottom. Therefore, to calculate the y-coordinate of the pixel, outMin and outMax are swapped.

Once this is done, the bounding boxes are stored in a list. The script then iterates over the annotations to create the crops. For each unprocessed annotation, a range of coordinates around the annotation is generated, defining the region of interest. These coordinates are randomly selected within a certain range around the annotation. The script then checks if the selected region of interest contains only fully visible objects. If this is true, the annotations present in the region of interest are registered as already processed. This is important to preserve the uniqueness of the original dataset annotations.

The crop is then saved along with the label in YOLO format and a second label containing not the bounding box information, but the segmentation. This will be used later for both the unet and the dataset augmentation.

The script also splits the dataset into training and validation, choosing 85% for training and 15% for validation.

Finally, the bounding boxes are used to store the raw LiDAR data points corresponding to each annotation. These points are stored in separate .las files, with each file containing the points whose coordinates fall within the corresponding bounding box. This approach ensures that the LiDAR data is organized and associated with the specific objects annotated by the bounding boxes. The use of these files is explained later.

The tables 3.2 and 3.3 show the results of the dataset creation.

Tumulis		
Set	Images	Annotations
Training set	139	233
Validation set	23	42

Table 3.2: Tumulis dataset

Hillforts		
Set	Images	Annotations
Training set	139	141
Validation set	21	22

Table 3.3: Hillforts dataset

3.2 Dataset augmentation

Deep learning has revolutionized the field of artificial intelligence, enabling significant advances in areas such as image recognition, natural language processing, and many other complex domains. However, for deep learning models to reach their full potential, it is essential to overcome challenges such as sparse training data and limited generalization.

The main reason why augmentation is so important is its ability to improve the generalizability of deep learning models. With more examples available for training, the model is exposed to a wider variety of cases and patterns, allowing it to learn more robust and variation-resistant features and avoid overfitting. Overfitting occurs when a model performs very well on training data, but struggles to make accurate predictions when presented with new data. This can happen for several reason[15]:

- There is too little training data, creating a dataset that is unrepresentative of all possible inputs.
- The model is trained for a long time and thus, without stopping early, increasingly makes the model only an expert on the given type of training data.
- The model is too complex, causing it to learn not only the necessary predictive features, but also the noise that the inputs may have.
- The training data contains irrelevant data, called noisy data.

This is clearly an undesirable phenomenon.

Analyzing the github repository that contains the code needed to run YOLOv7, it is possible to verify that there are no recommendations for creating a custom dataset, not even in the published paper [16], which only states that the COCO dataset was used to train the model and nothing else is mentioned about datasets.

However, in the YOLOv5 repository, there is a section where it is given recommendations on how to train a custom dataset, and as said before, it is recommended to have more than 1500 images and more than 10000 annotations per object type [17], and as it is possible to see, the original dataset is far from containing this amount of data and it was to overcome this problem that two augmentation algorithms were used. I will refer to the first algorithm as copy-paste augmentation and the second as simple augmentation.

3.2.1 Copy paste augmentation

The copy-paste augmentation method used consists of copying annotated objects and pasting them into areas where it is very certain that there are no unannotated objects.

When creating the base dataset, only a small portion of the LRM images were used, leaving a large number of potential crops with unannotated backgrounds. From here, it would be easy to use the segmentation already provided for the archaeological sites, crop and paste these backgrounds. However, there is a problem, the provided annotations do not result in intensive terrain analysis, leaving the possibility of unannotated sites, which means that if the algorithm crops an image where there is an unannotated site, this would lead to a degradation of the deep learning algorithm.

It becomes necessary for the copy-paste algorithm to be able to discard backgrounds with possible unannotated sites, which in itself is a paradox since the goal of the project is to detect unannotated sites. To do this, we used the yolov5 model trained on the base dataset. This model is then responsible for analyzing candidate crops, discarding all those that have possible new sites.

Since the model was trained on limited data, it is extremely sensitive, so it ends up detecting many shapes that are actually false positives. Because of this sensitivity, the model is useful in discarding possible crops, and because the background is quite large, even if the model discards many images, the background is not compromised.

A second problem that arises from the copy-paste algorithm is the choice of the paste location. If it is chosen randomly, it is possible that the archaeological site will be over a river, a house, a road, and so on, which would also lead to a deterioration of the deep learning algorithm. For this reason, the LBR (Location-Based Ranking) technique was used. This technique consists in classifying the land areas as subsoil or land use. For this purpose, the Charter of Land Use and Occupation of Portugal from 2018[18] was used. This charter is basically a comprehensive document that provides detailed information about the different types of land use and occupation in Portugal. It serves as a reliable reference for categorizing land areas into subsoil or land use regions.

The LBR technique uses this charter to determine the classification of land areas. By referring to the specific guidelines and classifications outlined in the Charter, the technique can accurately label each area as either subsoil or land-used, thereby providing valuable context for the augmentation process.

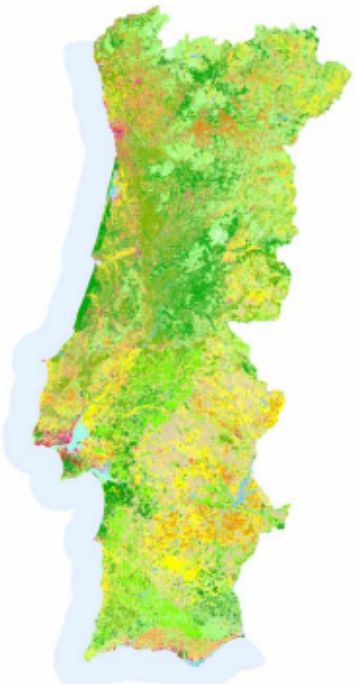

Figure 3.3: Land-use and Occupation Charter of Portugal, 2018. Shade of green represent different types of forest, shades of yellow represent different types of agriculture and shades of red represent infrastructures

Of the 83 classes present on the chart, only the areas with forest, sparse vegetation and agriculture were kept, and all the others were removed.

In the image 3.4, it is possible to see part of the LRM image of Paredes de Coura, without and with the proposed LBR. The green zones represent the areas of interest, resulted from the charter mentioned.

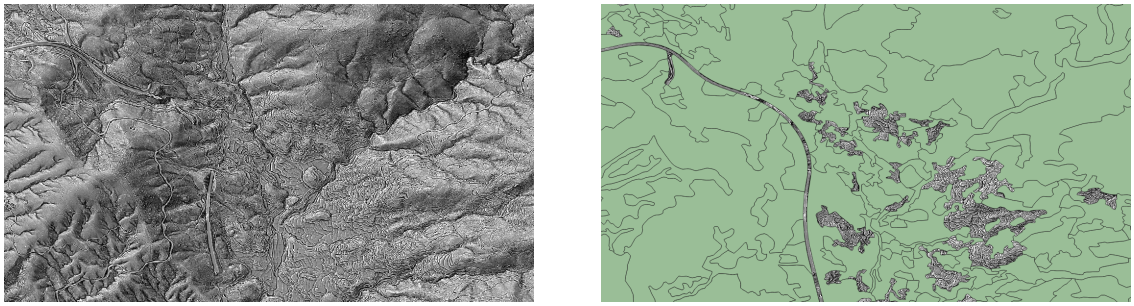

Figure 3.4: Croppred image from Paredes de Coura LRM on the left, and the same image with the LBR on the right

Using these two methods, the dataset was then created. It is worth mentioning that before pasting the annotation, one of the following geometric transformations was applied: flip left to right, flip top to bottom, rotate 90, 180 or 270 degrees and transpose. For each annotation, a transformation was randomly selected and then applied.

Additionally, 10% background was added, following yolov5's recommendations.

Along with the respective labels for the yolo, labels with the segmentation were also saved, which will be used later in Unet.

For the tumuli, being a relatively small object, it was chosen to add 8 tumulis per image, for the castros, being larger objects, it was chosen to add 5 hillforts per image.

In the image 3.5, there are two examples of the copy paste augmentation, on the left side tumuli and on the right side hillforts.

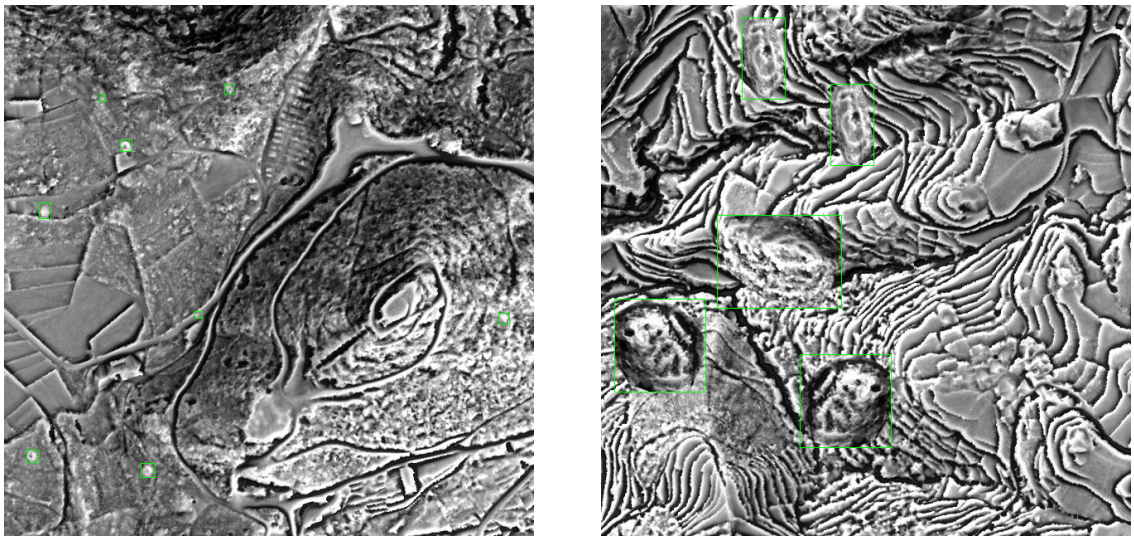

Figure 3.5: Examples of copy paste augmentation

The tables 3.4 and 3.5 contain the results from copy paste augmentation.

Tumulis		
Set	Images	Annotations
Training set	2363	15787
Validation set	391	2611

Table 3.4: Tumulis copy paste dataset

Hillforts		
Set	Images	Annotations
Training set	2304	9668
Validation set	256	1071

Table 3.5: Hillforts copy paste dataset

3.2.2 Simple augmentation

The second augmentation algorithm, which it will be referred to as simple augmentation, is a much more straightforward approach compared to the first one. It involves centering each object within a 640x640 pixel box and randomly shifting the box. Subsequently, the algorithm verifies if the object, along with other nearby objects, remains completely inside the box. If all the objects are contained within the box, a similar transformation to the previous algorithm is applied, not to the individual object but to the entire image. Finally, the modified images and their corresponding labels are saved.

This method was applied several times for the same annotation enabling the generation of annotated data that retains the same objects but positions them differently within the image, resulting in varied backgrounds.

The images 3.6 and 3.7 show examples of the same annotation but on different images resulting from the augmentation.

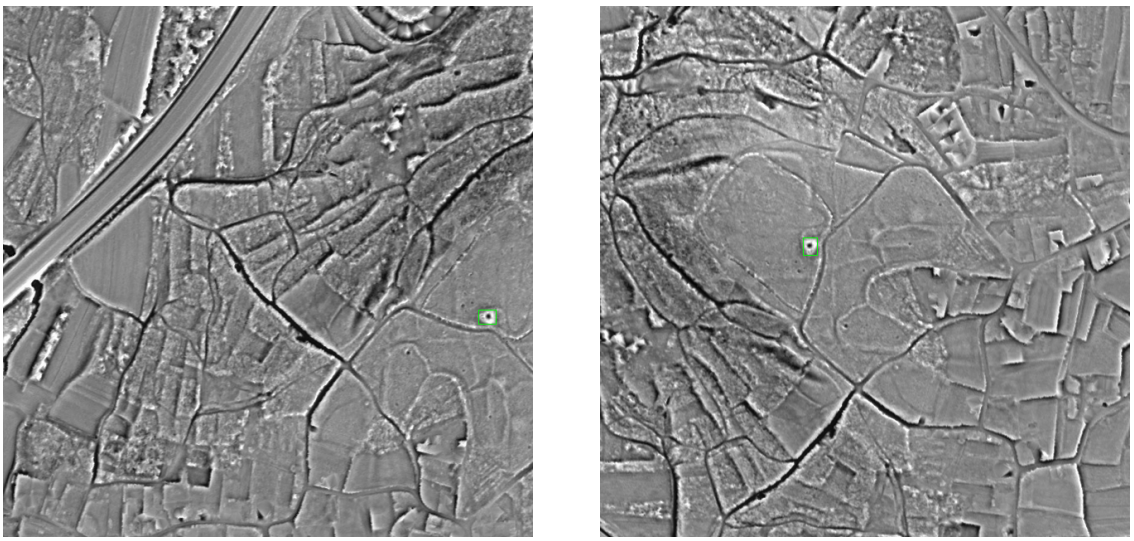

Figure 3.6: Example of the same tumuli but in different position of the image

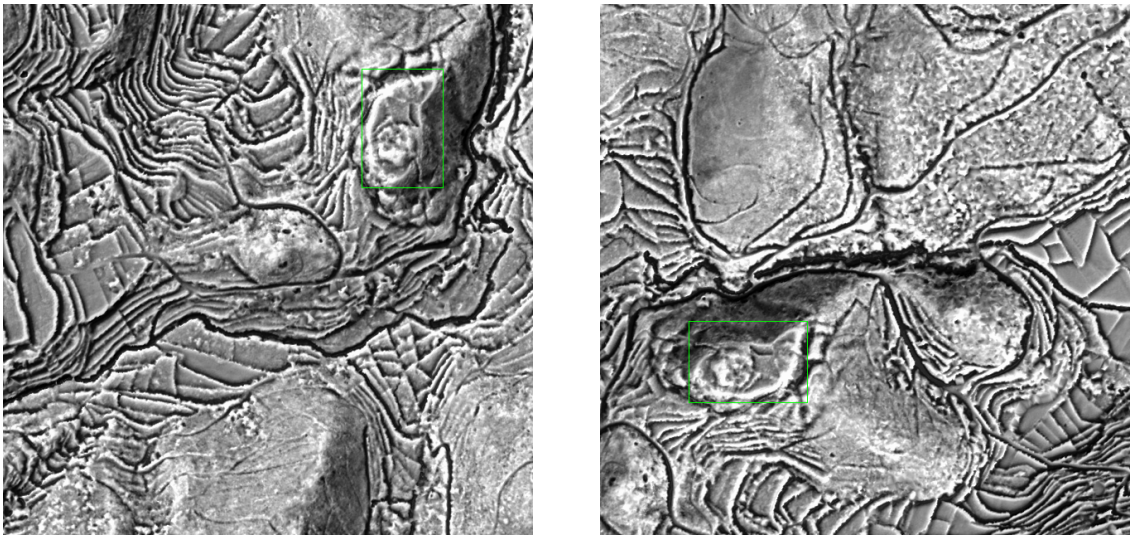

Figure 3.7: Example of the same hillforts but in different position of the image

The tables 3.6 and 3.7 show the results of the dataset creation.

Tumulis		
Set	Images	Annotations
Training set	1171	2658
Validation set	209	466

Table 3.6: Tumulis simple dataset

Hillforts		
Set	Images	Annotations
Training set	691	716
Validation set	124	128

Table 3.7: Hillforts simple dataset

Having implemented these augmentation algorithms, the dataset was significantly augmented, however with the simple augmentation, the results does not reach the miminum recommended by YOLOv5, it was chosen not to increase more the dataset because, being only a transformation of the image, increasing it more could lead to a similar images.

CHAPTER 4

Object detection

In this chapter we will present one of the techniques used to detect archaeological sites, object detection.

One of the challenges of artificial intelligence is image classification, i.e. identifying which class the content of the image belongs to. However, another more challenging problem is object detection, i.e. identifying where the object is within the image.

One of the first algorithms developed for this purpose is RCNN (Region-based Convolutional Neural Network). This algorithm involves generating approximately 2000 regions of interest in an image using the selective search algorithm, and subsequently performing image classification on these proposed regions using a CNN. While this algorithm can yield satisfactory results, one of its drawbacks is its slow processing speed, averaging around 47 seconds per image.

Later, two other algorithms were created, the Fast RCNN and the Faster RCNN, showing improvements, the latter being able to process an image in mere seconds.

Later in 2016, a paper was published on a model with an architecture called You Only Look Once (YOLO), which has since had several versions, the latest being YOLOv8. YOLO is considered state of the art for object detection.

4.1 YOLO

While the 3 methods presented, RCNN, Fast RCNN, and Faster RCNN, all of them run the image several times through the algorithm, YOLO comes with a different idea, the image only passes through the model once.

For this, the image is divided into a grid[19] with several cells, the image is then processed by the model, generating a two dimensional tensor, with the number of rows equal to the number of cells, and columns equal to 5 + number of object classes. In other words, each cell has a vector associated with the probability of having an object with the central point of the bounding box inside it.

y = [pc, cx, cy, w, h, c0, c1, c2,...].

- pc is the probability of being the identified object.
- cx and cy is the center point of the bounding box.
- w and h are the width and height of the bounding box.
- c0, c1, c2, ... indicates which class the detection belongs to.

However, this approach presents a problem, because for the same object several bounding boxes can appear, for this the algorithm uses a function called NMS (Non Maximum Suppression).

Non Maximum Suppression

Non Maximum Suppression [20] is a method for selecting a region from a set of overlapping regions. First, it discards all proposed regions with a probability below a certain threshold. For the remaining regions, it selects the region with the highest probability and discards all others that have an IoU higher than a certain threshold.

IoU is the intersection over union, that is, the area of the intersection of the overlapping regions to be divided by the area of the union. It will be better explained later in the dissertation.

4.1.1 Training

Neural network training is a fundamental step in deep learning, it is in this phase that the network will "learn" how to perform the task at hand, this can range from image classification, object detection, language processing and many other applications.

To do this it is common to divide the dataset into 3 parts, training, validation and testing. During training the net goes through several iterations called epochs, where it learns to make predictions by adjusting its internal parameters based on the provided training data. The training process consists of feeding the neural net with batches of training data. A batch is a subset of the training data set, instead of feeding the entire training data set to the neural net, it is divided into batches, making the process more computationally efficient, and also allowing the weights of the neural net to be adjusted more often, since they are updated at the end of each batch.

At the end of the batch processing, the error is calculated relative to the predictions made by the model and the labels. This error is usually measured by a loss function, such as cross entropy or least squares error. After the error is calculated, the optimization algorithm, such as SGD or Adam, is used to adjust the weights of the neural network in a direction that minimizes the error. To do this a back propagation function is used, where the error is propagated from the output layer to the input layer, allowing each neuron to update its weights in order to contribute to the reduction of the total error.

At the end of each epoch, a validation of the model is then performed with the validation set, that is, the data from the validation set is used to evaluate the performance of the trained model up to that point. Model validation is crucial for monitoring the generalization ability of the model over the course of training. It provides information on how the model is performing on data not used during training, and helps identify problems such as overfitting. Overfitting

occurs when the model overfits on training data, losing the ability to generalize to other input data.

It is at this stage that the hyperparameters, values that control the learning process, are updated, one of them being for example the learning rate. At the end of training, inference is performed on the test set, which consists of data never before seen by the model. While the validation set is used to adjust the hyperparameters and is thus indirectly seen during training, the test set remains completely unknown to the model until that moment.

Pre-trained models

One of the techniques widely used in deep learning is the use of pre-trained models, especially when the available dataset is relatively small.

The main advantage of pre-trained models is that they capture knowledge and patterns learned from massive datasets, which can be applied to specific tasks with smaller datasets. They are able to extract relevant features and abstract representations from the input data, making them more effective at the task at hand.

Although pre-trained models have almost the same accuracy as untrained models, the robustness and uncertainty of the model changes dramatically, as can be seen in the article [21].

This means that for new input data that the model has never seen before, the model can make better predictions and be more certain of the results, giving a higher probability of the results.

YOLOv7 training

YOLOv7 already provides pre-trained models with the COCO dataset, offering a total of 6 models, in the following table it is possible to analyze the performance of the models in the coco dataset.

	YOLOv7	X	W6	E6	D6	E6E
Parameters (Milions)	36.9	71.3	70.4	97.2	154.2	151.7
mAP@0.5	69.7	71.2	72.6	73.5	74.0	74.4

Table 4.1: Parameters of yolov7 models trained on the COCO dataset and the respective mean Average Precision calculated with and intersection over union threshold of 0.5(mAP@0.5), being X the YOLOv7-X, W6 the YOLOv7-W6, E6 the YOLOv7-E6 and E6E the YOLOv7-E6E.

By analyzing the table it is possible to see that the best model in terms of mean Average Precision is only 4.7% better, however it has 4.1 times more parameters. For this reason the YOLOv7 pre-trained model was chosen. The optimizer used was the stochastic gradient descent with a learning rate of 0.01.

The generated datasets were divided into training and validation data, 85% training and 15% validation. The test dataset was not used, because, at the end of training, an inference was performed on the original LRM images.

The models were trained between 200 and 350 epochs, with an expected stop if there was no improvement after 40 epochs, using a batch size of 8, the highest allowed for the hardware used, which was an Nvidia GeForce RTX 3080 GPU and an AMD Ryzen 5 5600X 6-Core 3.7GHz CPU.

Cross fold validation

Another widely used technique is cross-fold validation, which is known to give good results, especially when the dataset is small. This technique consists of dividing the entire dataset into K folds, and then using each fold as validation data and the others as training data to generate K models.

This method is useful because the entire available dataset is used to train K-1 models, which makes it possible to analyze whether there are significant differences in the use of different parts of the dataset. This was only applied to the dataset generated by copy-paste augmentation. The dataset was divided into 5 folds, with each fold containing 550 images.

4.1.2 LOF model training

One of the problems encountered earlier in the dissertation was that the inference results contained many false positives; in an attempt to solve this problem, a Local Outlier Factor (LOF) model was trained on the raw LiDAR data.

Using the point cloud files generated in LAS (LASer) format when creating the dataset, which contain only the points within the bounding box surrounding the object in question, two LOF models, one for tumulis and one for hillforts, were trained using the scikit learn library[22].

LOF calculates the local outlier factor for each sample in the dataset, taking into account the local density of neighboring samples. It compares the density of the sample with the average density of its nearest neighbors. If the density of the sample is significantly lower than the average density of its neighbors, this indicates that the sample is a possible outlier.

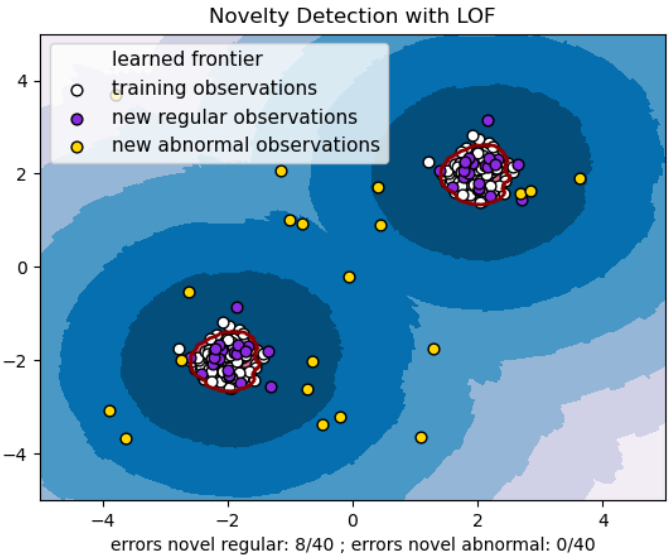

Figure 4.1: Example of clusters generated by LOF[22]

To train the model, the jakteristics library[23] was used. This library allows 14 features to be taken from each point in the point cloud, these features are: Eigenvalue Sum, Omnivariance, Eigenentropy, Anisotropy, Planarity, Linearity, PCA1, PCA2, Surface Variation, Sphericity, Verticality and Normal Vector (nx, ny, nz).

Since each point contributed 14 features, this results in a huge amount of features, the mean, median, variance, standard deviation and covariance are calculated, resulting in an amount of 70 features per .las file.

4.1.3 Metrics

YOLOv7 already provides the python script needed for training and also saves the training results. These training results are fundamental to understand if the model is well trained or not. Several metrics are used for this, some of them being the confusion matrix, accuracy, mAP and F1.

The confusion matrix is nothing more than a table with the values of true positives, false positives, true negatives and false negatives (TN). True positives (TP) are all predictions generated by the model where an object actually exists, false positives (FP) are predictions where the object in question does not exist, true negatives (TN) are predictions where the model is all the background given by the model, which actually corresponds to background and finally, false negatives are all objects that were not detected by the model.

Accuracy is the ratio between all correct predictions and all predictions generated by the model, given by the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Two other values that can be obtained are accuracy and recall[24]. Precision refers to what percentage of the predictions made are true positives,in other words, how accurate is the

model in identifying only relevant objects, and is given by the following formula:

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Recall on the other hand tells us the ability of the model to identify ground truths and is given by the following formula:

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

To obtain the graph of both precision and recall, several confidence values are used, i.e. the value that is required to be trusted for each detected object, so that it is not discarded. As a rule, a higher confidence value leads to a higher recall, while a lower confidence value leads to a higher precision. Taking the precision and recall values in pair for several confidence values, the graph recall vs precision is generated, being the area under the curve the corresponding AP (Average precision). To calculate if a detected object is a false positive or not, a function called IoU (intersection over union) is used.

For a detected object to be considered a true positive, it is necessary that the area of intersection between the bounding box of the detected object and the bounding box of the ground truth divided by the union of the two areas is greater than a certain threshold. When this value is 50%, the AP is called AP50.

$$IoU = \frac{\text{Intersection area}}{\text{Union area}} \tag{4.4}$$

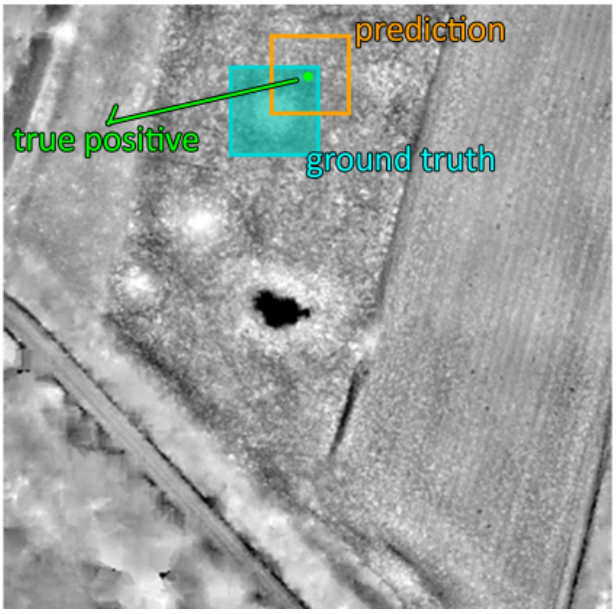

Figure 4.2: Example of IoU [25]

Having the APs of all classes, the mean Average Precision (mAP) is then calculated by taking the average of the APs, however in this project separate models were trained for each class, making the AP equal to the mAP.

Another metric is F1, which is given by the following formula:

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{4.5}$$

This metric is especially interesting, when both accuracy and recall are important, enabling an analysis of which is the most favorable confidence value.

Another metric that is also commonly used, is mAP0.5:0.95, the difference with mAP50, is that instead of using only a minimum value of IoU to consider a true positive, several values between 50% and 95% are used, with 5% jumps.

When it comes to lof, training data is unavailable. Additionally, unlike yolo, the original dataset cannot be expanded due to its raw lidar data nature. Given the limited size of the dataset, all the generated .las files were utilized and as mentioned in [22], performing inference on training data can yield inaccurate outcomes.

4.1.4 Training results

In this subsection the training results of yolov7 will be shown. Training was performed with all the generated datasets, which are 4 for the tummies and 3 for the hillforts.

As said before, the dataset was divided into 85% for training and 15% for validation, no test set was used because the original dataset provided was small, and also because an inference will be made on the LRM images, so it will be possible to analyze the model's performance, analyzing if it found all the annotated objects as well as new ones.

First the training results for the tumuli will be analyzed, and then the results for the hillforts will be analyzed.

In the case of yolov7 training with the original dataset, the result is shown in the image 4.3:

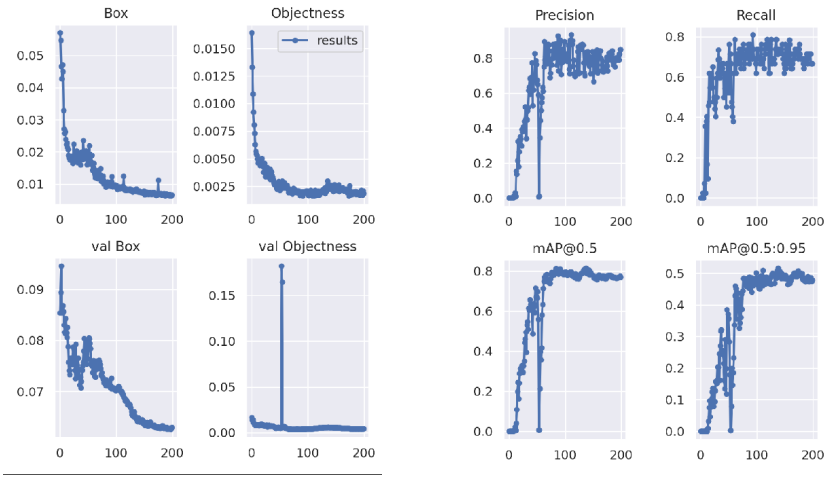

Figure 4.3: Results of Yolov7 training with tumulis non augmented dataset

In the case of training with the dataset augmented with the copy paste algorithm, the result is shown in the image 4.4:

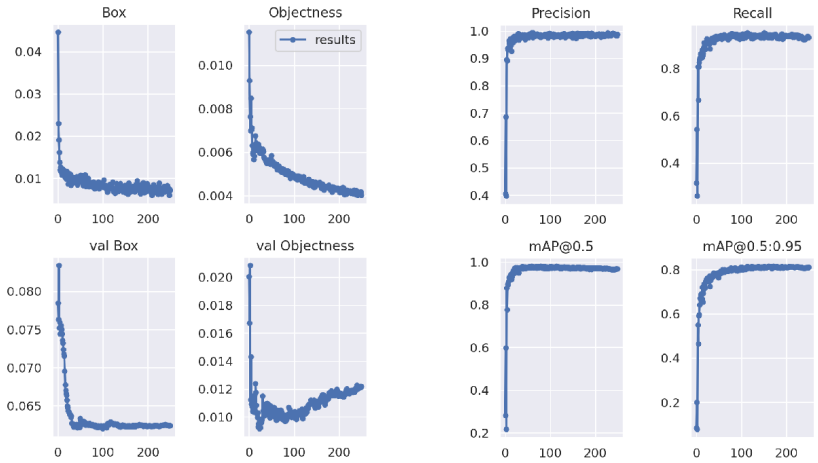

Figure 4.4: Results of Yolov7 training with tumulis augmented dataset (copy paste)

In the case of training with the dataset augmented with the simple algorithm, the result is shown in the image 4.5:

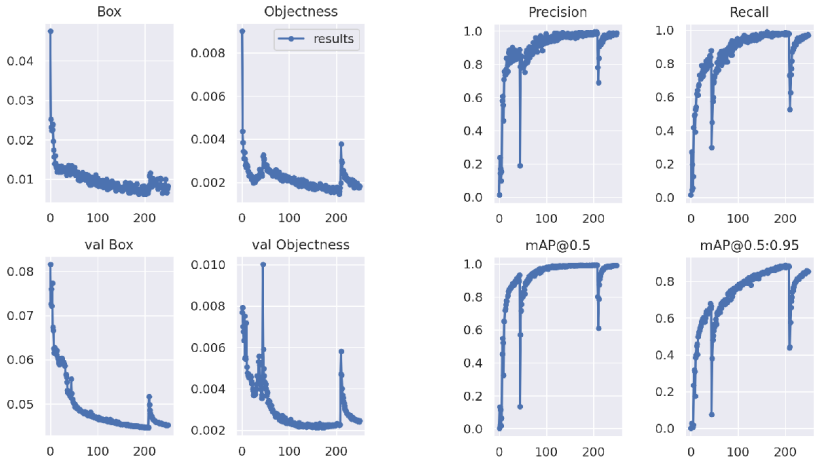

Figure 4.5: Results of Yolov7 training with tumulis augmented dataset (simple)

To analyze these results, only the mAP0.5 metric will be used, not the F1 score. The reason for this choice is that mAP provides a consolidated value that combines both precision and recall for each epoch, while F1 score represents a graph for each epoch, making it a more complex metric to work with.

As can be observed, there was no overfitting in any of the three models, even when trained on the unaugmented dataset, which is the most prone to overfitting. It is interesting to note that in the copy-paste dataset, the model stabilized after a few seasons and became very close to 1.

The training results of the 5 folds corresponding to the cross validation are shown in the figures 4.6, 4.7 and 4.8.

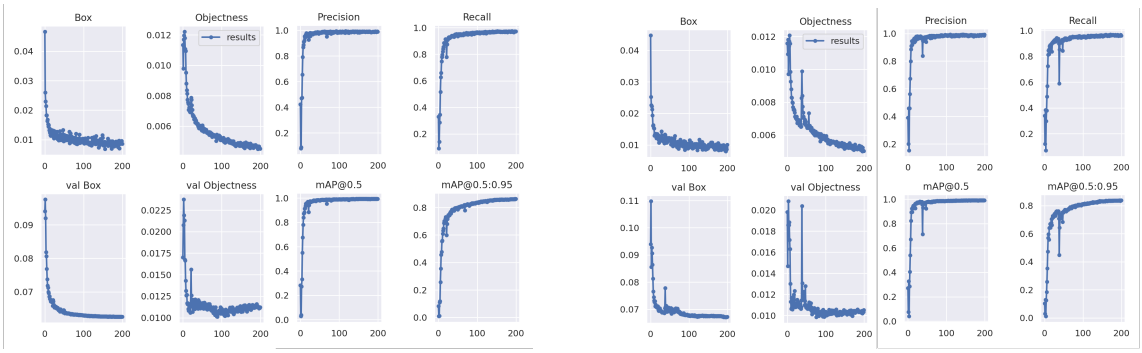

Figure 4.6: Fold 0 and Fold 1 training results for cross validation

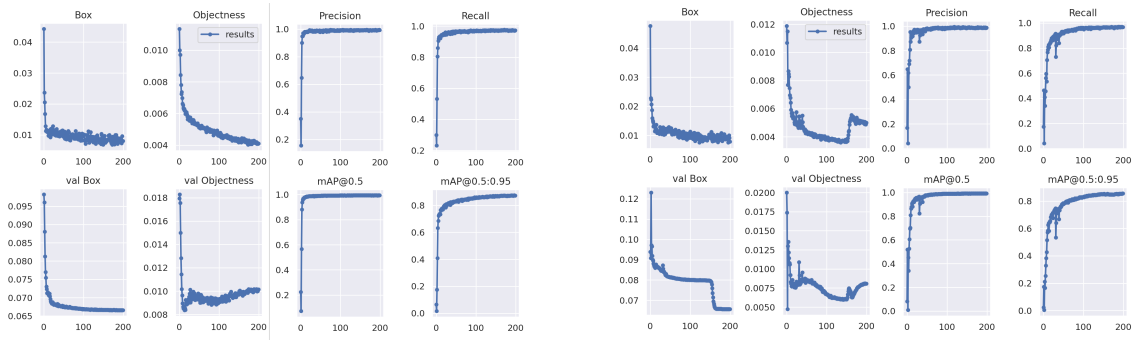

Figure 4.7: Fold 2 and Fold 3 training results for cross validation

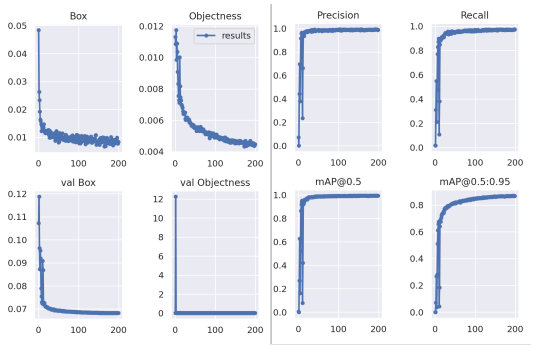

Figure 4.8: Fold 5 training results for cross validation

As can be seen from the folds training, at least with 5 folds, there is no subset in which there is a big difference in performance, meaning that the data set is homogeneous and does not contain a subset with data that is very different from the rest.

In the case of hillforts, the results are shown in the image 4.9:

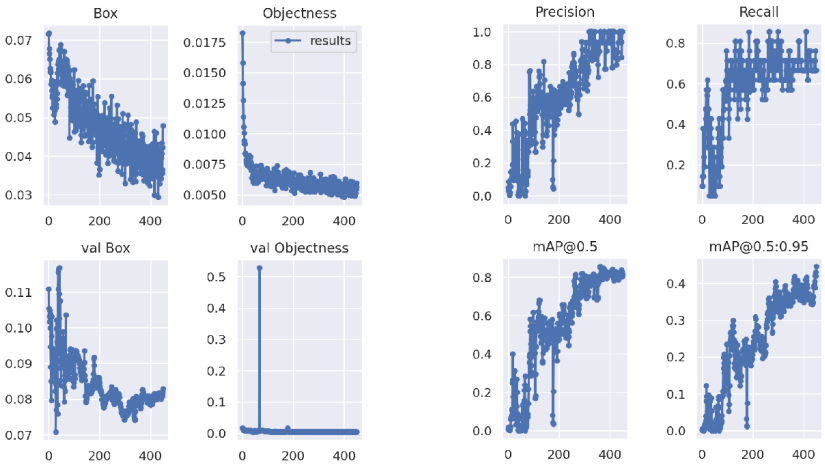

Figure 4.9: Results of Yolov7 training with hillforts non augmented dataset

In the case of training with the dataset augmented with the copy paste algorithm the results are shown in the image 4.10:

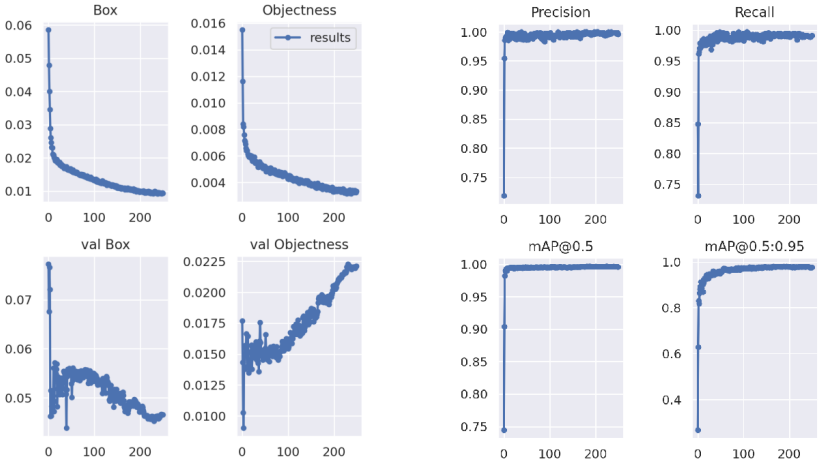

Figure 4.10: Results of Yolov7 training with hillforts augmented dataset (copy paste)

In the case of training with the dataset augmented with the simple algorithm the results are shown in the image 4.11:

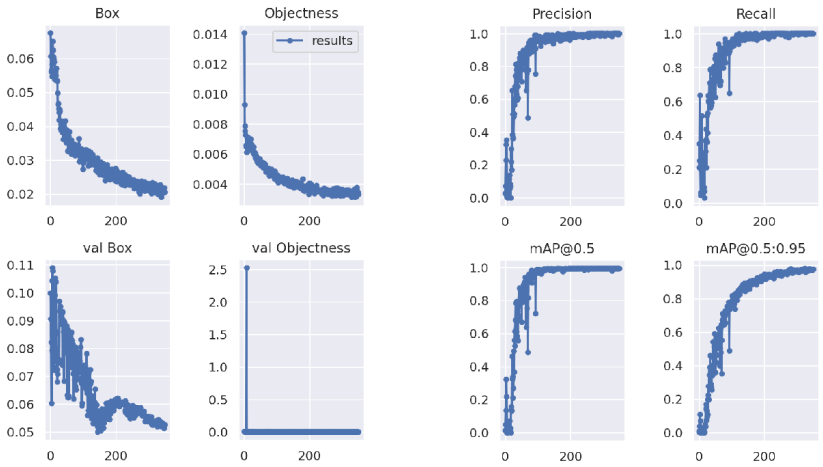

Figure 4.11: Results of Yolov7 training with hillforts augmented dataset (simple)

Similar to the tumulis, there was no overfitting. In the case of the training with the non augmented dataset, it was necessary to increase the number of seasons in order to make the model converge. And similarly to the tumuli training, the hillforts training with the copy paste dataset, after a few seasons, the mAP was already close to 1.

4.1.5 Inference

In this sub-section, it will be shown how the inference was carried out on the 4 images of the LRM.

In order to do the inference, a python script was created, in this script the image is loaded using the PIL library and then the geographic coordinates of the 4 corners of the image are obtained using the rasterio library, this is possible because tif images can contain additional information such as geographic coordinates.

Since the coordinates and the images are too large to make inferences, an interactive window was used to perform 640x640 pixel cropping. However, this approach leads to a problem, the cropping can be done on top of an object, leaving part of the object in one image and another part in the next image, to work around this a sliding window was used instead of a fixed window, with the sliding window instead of cropping in a fixed way at each position, the image is scrolled in a systematic way by moving the window pixel by pixel or at a defined interval. This approach allows each object or region of the image to be covered by multiple inference windows, thus avoiding the problem of having parts of an object split between different crops.

Sliding windows divide the image into overlapping windows that move along the image. Each sliding window captures a specific region of the image, and then inference is applied to that region individually. As the window slides, new regions are processed, ensuring that all objects or regions are covered.

For the tumuli a 50% slide was used, because they are small objects, and for the hillforts a 30% slide was used, because some hillforts occupy almost half of the cropped image.

Once the crop is obtained, the topographic map is used to determine whether the crop contains areas that are likely to contain objects. This means that there are areas in the crop where, for example, there are no houses or streets, so crops that represent, for example, a residential area in its entirety are discarded. After that, the inference is done with YOLO. As mentioned before, YOLO can make multiple detections for the same object, so to overcome this, the NMS function is also used here, discarding overlaps and all inferences with a confidence lower than 25%. Then the topographic map is used again to check that the object is not in an area where it would be impossible to detect.

Once the inference is complete, the identified objects are validated using the LOF model. To validate each object, a temporary LAS file is created. This file is created using the geographic coordinates of the bounding box. The temporary LAS file is then populated with points that fall within the bounding box. Next, various statistical measures such as mean, median, variance, standard deviation, and covariance are calculated using the 14 features extracted from each point. Finally, a prediction is made, returning a value of 1 if the object is determined to be a true positive, or 0 if it is determined to be a false negative.

Finally, due to the window slider, there may be multiple inferences for the same object, because when the slider is moved, the same object may still appear in the next image. To mitigate this, Euclidean distance was initially used to remove all bounding boxes whose center was less than 15 pixels, or 7.5 meters on the ground, from another bounding box. Initially, this approach worked well for the tumuli. However, for hillforts, a problem arose: the Yolo detected hillforts inside larger hillforts, which does not happen in reality, making it difficult to choose a threshold for the Euclidean distance.

In order to get around this, the Euclidean distance was replaced by IoU, as explained before, if the IoU was smaller than a certain value, the smaller hill was discarded, however, the problem still remains, as I checked later in article [25], if the smaller hillfort is fully contained in the larger one and if the area of the smaller one is smaller than the defined threshold value, IoU does not discard, so the problem still remains. So the approach was adopted, instead of being the normal IoU the intersection. Therefore, instead of using the normal IoU, the intersection to be divided by the union, the division of the intersection to be divided by the area of the smallest detection was used. But before that the objects were ordered by area, from the biggest to the smallest, by doing that and using this modified version of the IoU, in the cases where there are detections completely contained inside another, or almost completely contained, they are removed, keeping the largest.

In the end, the results are saved in a csv file, containing the geographic coordinates of the bounding box, the confidence given by YOLO and the result of the LOF validation.

Once the inference is complete, it is necessary to evaluate the effectiveness of the model by analyzing the discovered ground truths and new objects. To facilitate this analysis, a Python script was developed. Within this script, the ground truth coordinates are loaded and converted into corresponding image pixels, which are further converted into bounding boxes. Similarly, the coordinates of the objects resulting from the inference are processed. Using the Shapely library, both the ground truth objects and the inferred objects are converted to

polygons. Next, a filtering process is applied to the inference results, removing all inferences with an area smaller than 90% of the smallest ground truth area. Finally, an interaction is performed on the ground truth objects to determine which inference results intersect the ground truth.

In the case of inference with the 5 models generated by cross-validation, YOLOv7 provides a technique called model ensemble, which allows multiple models to be used simultaneously.

Model ensemble

Model ensemble is a technique that uses predictions from multiple deep learning models to produce more robust results than a single model could provide.

By using multiple models trained on different datasets, in this case the cross-validation models, each model generates its own predictions. The predictions are then combined using techniques such as majority voting, weighted averaging, probability averaging, and others to make a final decision.

The idea behind this is that each model can capture different features or aspects of the training data set, and by combining these different models it is possible to compensate for the failures of one model by the successes of others, making the result more reliable.

4.1.6 Inference results

In this sub-section it will be showed the results of the inferences made on the 4 LRM images. For this, a script was created, in which the annotations were loaded as well as the results, both the inference results and the annotations were converted to polynomials using the shapely library, then in an interactive way it was checked if any ground truth was intersected by any inference, if so it was accounted for, if there was more than one inference to intersect, only the first one was accounted for as a true positive and all the others as false positives.

The tables 4.2 and 4.3 show the results from inference, it shows the total number of objects detected, within these objects the number of validations made by the LOF, as well as the true positives, false positives and false negatives, the last 3 being related to the total number of objects detected and not to those validated by the LOF.

	Non augmented	Copy Paste	Simple	Cross Validation(Copy paste)
Total	2324	2366	1800	11446
Validations	1604	1500	1354	6385
True positives	240	224	210	242
False positives	2084	2142	1590	11204
False negatives	36	52	66	34

Table 4.2: Results of inference for tumulis(YOLOv7)

	Non augmented	Copy Paste	Simple
Total	8002	1913	5630
Validations	2799	628	2145
True positives	153	153	113
False positives	7849	1760	5517
False negatives	10	10	50

Table 4.3: Results of inference for hillforts(YOLOv7)

It is important to emphasize that the false positives and true positives are relative to the available ground truth, i.e., the training and validation data, and as mentioned above, this data is not the result of intensive terrain analysis, and there may be some false positives that are actually true positives, i.e., new sites discovered by the algorithm that have not yet been annotated.

However, according to the provider of the annotations, most of the sites have been annotated, with only a few that may have been missed.

Analyzing the data resulting from the inferences, we can verify that there were many false positives, both in the tumuli and in the hillforts, but in the tumuli the worst case was the one resulting from cross validation, giving about 7 times more false positives than the model resulting from simple augmentation, on the other hand it was the one that had fewer false negatives, that is, it was the one that managed to detect more ground truth.

In the case of the Hillforts, the model that gave fewer false positives was the one resulting from the copy-paste augmentation, giving 1750 false negatives and detecting almost all the ground truths, failing only 10 out of 163.

The pictures 4.12 and 4.13 are examples of the results of some inferences.

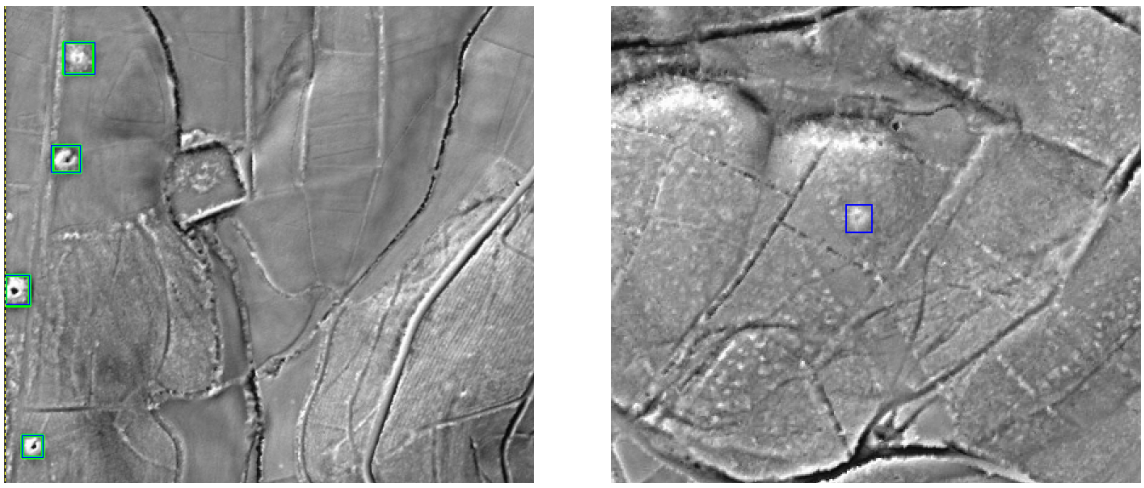

Figure 4.12: Example of the result of inference from the copy-paste model, the green corresponds to a ground truth, while the blue corresponds to a detection from YOLOv7, on the left are 4 tumuli correctly found and on the right is a possible new tumuli.

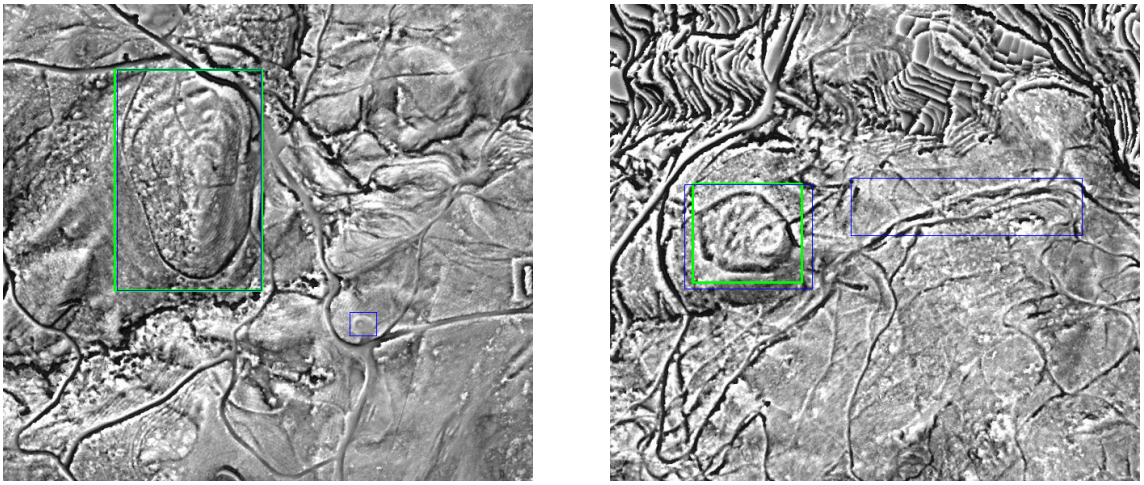

Figure 4.13: Example of hillforts, the green corresponds to a ground truth, while the blue corresponds to a detection from YOLOv7, on the right it is possible to see a clear false positive case.

In the context of Hillforts, it was anticipated that the outcomes would not yield considerable accuracy. This is primarily due to the heterogeneous nature of Hillforts, characterized by their diverse shapes and sizes, which poses a challenge for the Yolo model in distinguishing between background and foreground elements within the bounding box.

CHAPTER 5

Semantic segmentation

In this chapter we will present the second technique used to detect archaeological sites, semantic segmentation.

Identifying objects in images can be achieved by various methods, not limited to object detection alone. While object detection involves placing a bounding box to detect the object, there are alternative approaches such as instance segmentation and semantic segmentation. These methods aim to classify the image at the pixel level by assigning a class to each individual pixel within the image.

The goal of instance segmentation is to classify and distinguish individual objects in an image. In this method, each pixel is given a classification of which class it belongs to. In addition, each object is assigned to an individual class, which makes it possible to distinguish between objects of the same class. In this way, instance segmentation provides detailed information about the location and class of each object.

Semantic segmentation, on the other hand, focuses only on classifying the class of the pixel, without distinguishing between different objects of the same class. In this method, the image is divided into semantically meaningful regions, and each pixel within a region is assigned to the same class.

The image 5.1 illustrates the difference between semantic segmentation and instance segmentation.

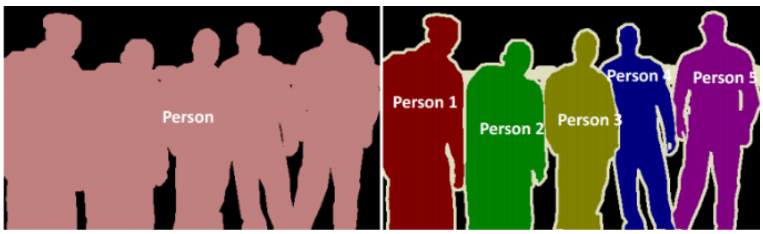

Figure 5.1: Semantic segmentation VS Instance segmentation [26]

Both instance segmentation and semantic segmentation require first predicting the instances of objects and their binary masks. It is done object detection and then semantic segmentation. The existing methods are generally divided into two categories[27], two-stage and one-stage instance segmentation.

The two-stage methods generally involve object detection, obtaining the bounding box of objects followed by obtaining the mask within each bounding box. This is how for example MaskRCNN works, this algorithm uses Faster R-CNN to get the bounding boxes and then adds a new branch to get the segmentation.

The one-stage methods, on the other hand, perform detection and segmentation directly without getting the bouding boxes.

Following these two methods, an example of each was used, for the two-stage method YOLOv7-seg was used, and for the one-stage one Unet was used.

5.1 Unet

Unet is a neural network widely used in semantic image segmentation tasks, having been proposed in 2015, it is where today one of the most popular neural networks for semantic segmentation.

The main motivation behind Unet was the challenge of medical image segmentation[28]. In constrast of the previous model, Unet gives one classification for each pixel.

Unet's architecture consists of two parts: the encoder and the decoder. The encoder is composed of convulse and polling layers, which reduce the resolution of the input image, and filter image features. These features are then passed to the decoder, which performs upsumpling and concatenation operations. It is because of these U-shaped downsampling and upsampling operations that the architecture is called Unet.

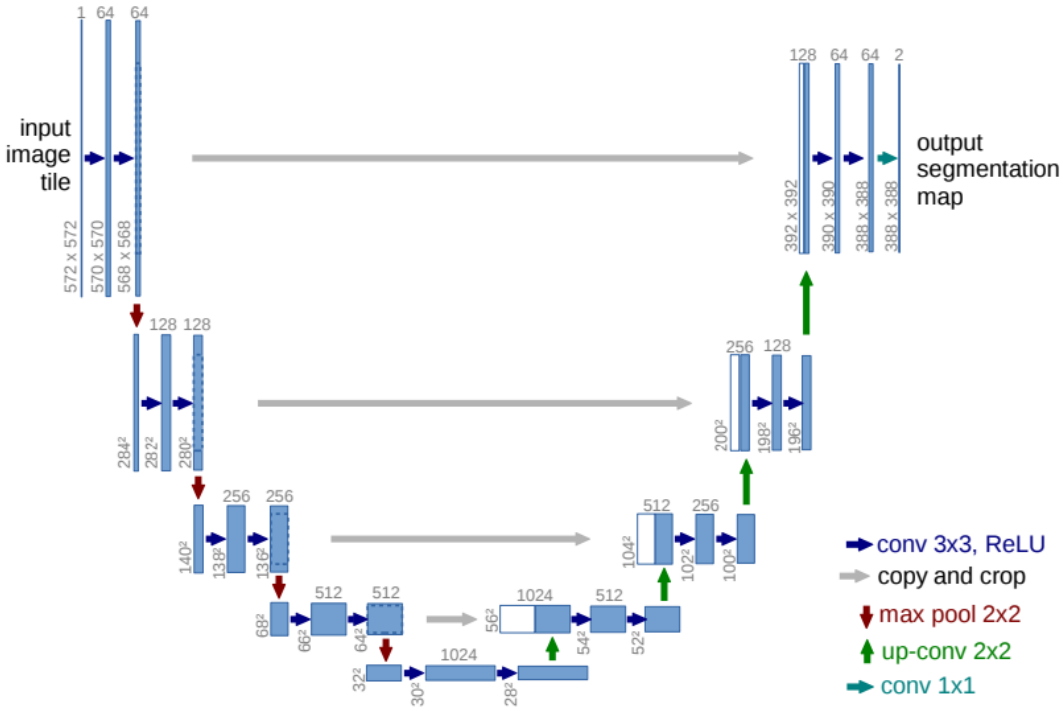

Figure 5.2: Unet's architecture[28]

5.1.1 Model training

In order to train the Unet model it was used the implementation present in the repository[29]. The repository already provides a pre-trained model trained with the carvana dataset.

The training dataset used was the same for YOLOv7. However, unlike YOLO, the annotations in Unet are not in text format, but in image format. To adapt the dataset, a Python script was created to read the annotations with the segmentations created during the dataset creation process and transform each text file into an all-black image with the object masks blank.

The image 5.3 shows a photo of the dataset copy paste of hillforts and its generated label.

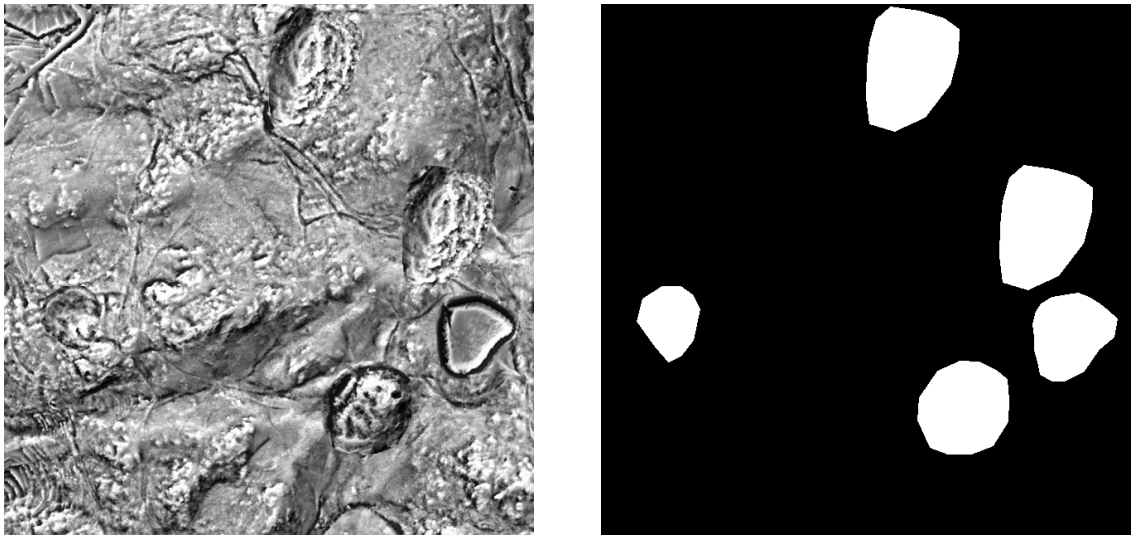

Figure 5.3: Example of label for Unet training

The training script has also been adapted to save the best generated model of all epochs. Also, an early stop was added so that if there is no improvement in 40 seasons, the training will stop.

The metric validation dice was used to evaluate the training. The Dice index (also known as the Dice coefficient) is a widely used metric for evaluating the overlap or similarity between two image segmentation masks. It is often used in segmentation problems, such as the segmentation of objects in medical images. It is calculated based on the comparison between the segmentation mask predicted by the model and the reference mask (ground truth). The calculation of the Dice index involves counting the corresponding pixels in the two masks and comparing the intersection and the sum of the areas of the masks.

The dice index is given by the following formula:

$$Dice = \frac{2 \times \text{Area of Overlap}}{\text{Total Area of Mask 1} + \text{Total Area of Mask 2}} \tag{5.1}$$

The value of the Dice index ranges from 0 to 1, where 0 indicates no overlap between the masks and 1 indicates perfect overlap. The closer the Dice value is to 1, the better the overlap between the masks and therefore the better the segmentation quality.

The images 5.4, 5.5, 5.6 and 5.7 show the evolution of the dice index during the training process.

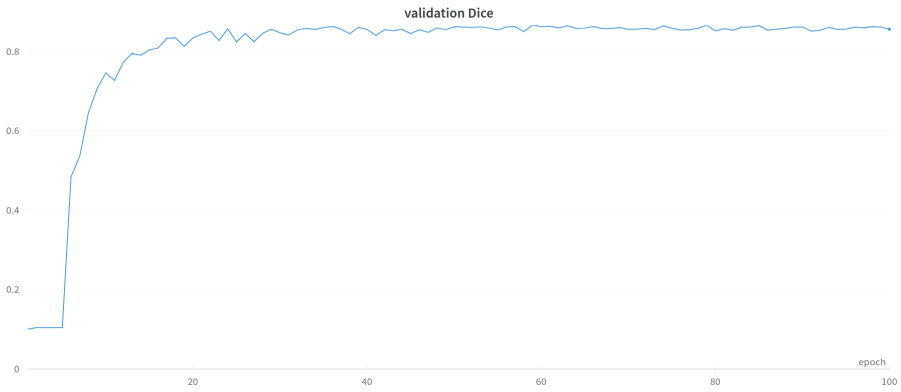

Figure 5.4: Evolution of dice index for tumulis copy paste dataset

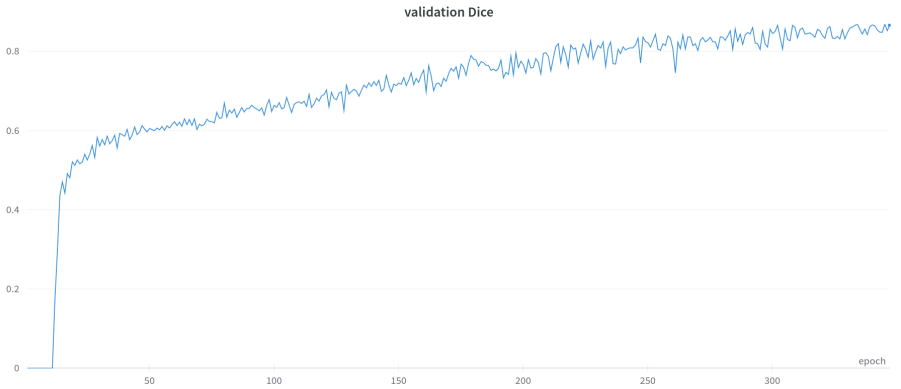

Figure 5.5: Evolution of dice index for tumulis simple dataset

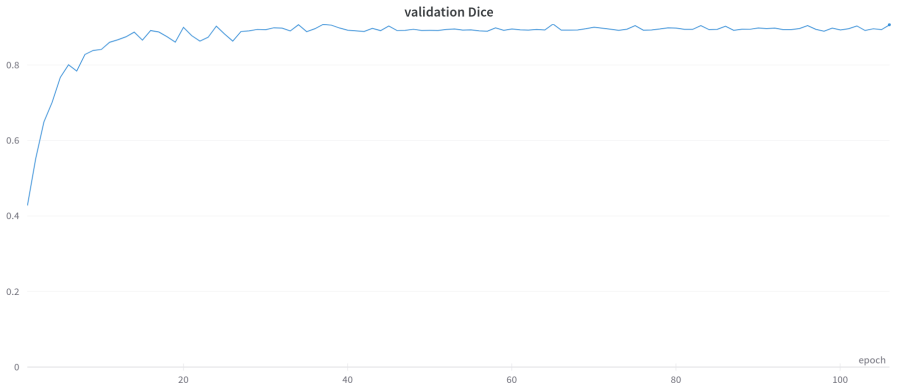

Figure 5.6: Evolution of dice index for hillforts copy paste dataset

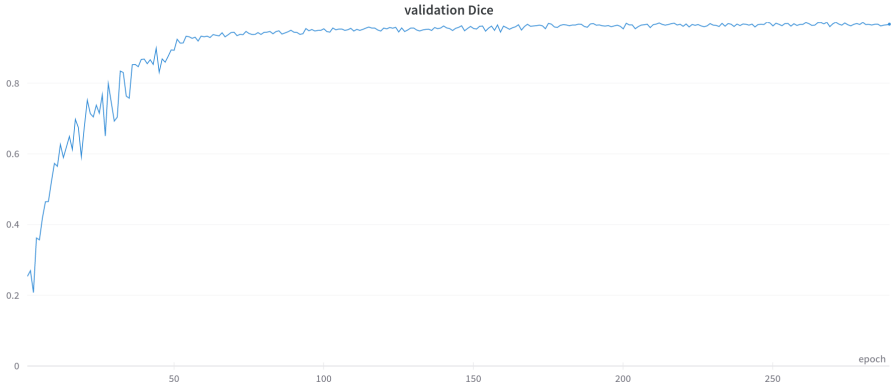

Figure 5.7: Evolution of dice index for hillforts simple dataset

As we can see, all training sessions stabilized with a validation dice around 80%.

One value that was fundamental to the training of the models was the learning rate. The default learning rate was 0.001, but it was verified that with this value the model still hadn't started to converge after almost 50 epochs, even a reduction of the learning rate in 10% was programmed if there was no improvement after 5 seasons. Several values were tried, finally 1e-7 was chosen.

For unet it was not possible to train with the unaugmented dataset, because most likely the dataset is small, the model had not started to converge after 100 epochs for both the tumuli and the hillforts.

5.1.2 Inference

The same script used in YOLO was employed for inferring the four LRM images. A sliding window of 640x640 pixels was created, with a sliding factor of 50% for tumuli and 30% for hillforts. After that, a crop was performed. In cases where the crop contained potential archaeological zones, this was verified using the land occupation map, and the crop was subsequently processed by the model.

The key distinction in the inference process between Yolo and Unet lies in the subsequent step. Unet generates results in a matrix format, where each element corresponds to a pixel in the image and contains the pixel's classification. To obtain segmentation, the OpenCV function findContours was utilized. This function provides the polynomials that enclose the masks. If a polynomial was not located within areas that cannot be archaeological zones (determined using the land use charter), it was saved for further analysis.

Next, the same YOLO inference function was used to eliminate duplicate results from the sliding window. The remaining outcomes were validated using the Local Outlier Factor (LOF) model. Finally, the annotations were saved in CSV files.

The tables 5.1 and 5.2 shows the results from inference, also here was used the same script used in YOLO to analyze the results.

	Copy Paste	Simple
Total	621	1117
Validations	378	808
True positives	121	177
False positives	500	940
False negatives	155	99

Table 5.1: Results of inference for tumulis(Unet)

	Copy Paste	Simple
Total	267	2938
Validations	98	1227
True positives	135	108
False positives	132	2830
False negatives	28	55

Table 5.2: Results of inference for hillforts(Unet)

As in YOLO, the total is the total number of detections of the model, the validations are the total number of validated results of the LOF model, the true positives, false positives and false negatives are relative to the total number of detections and not to the results validated by the LOF.

As can be seen, the results are quite different from YOLO, the first difference is in the number of detections, Unet has detected many fewer archaeological sites than YOLO, but it has left many of the sites already noted undetected.

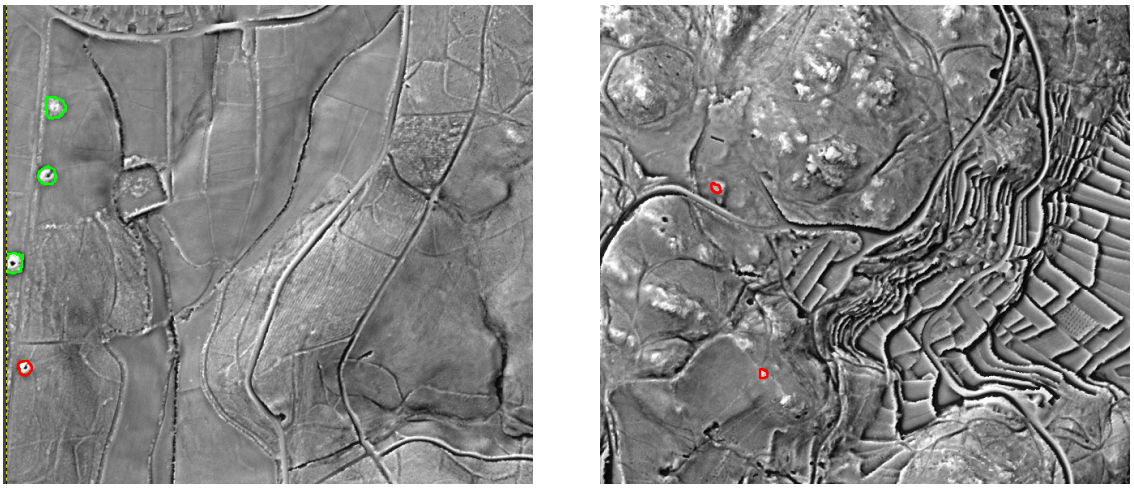

Figure 5.8: Examples of inference results from the copy-paste model for tumulis, the green corresponds to a ground truth, while the red corresponds to a detection from Unet.

In the figures 5.8 and 4.12 it is possible to see the contrast between false negatives, in Yolo and Unet, while Yolo detected four ground truth tumuli on the left, Unet detected only one.

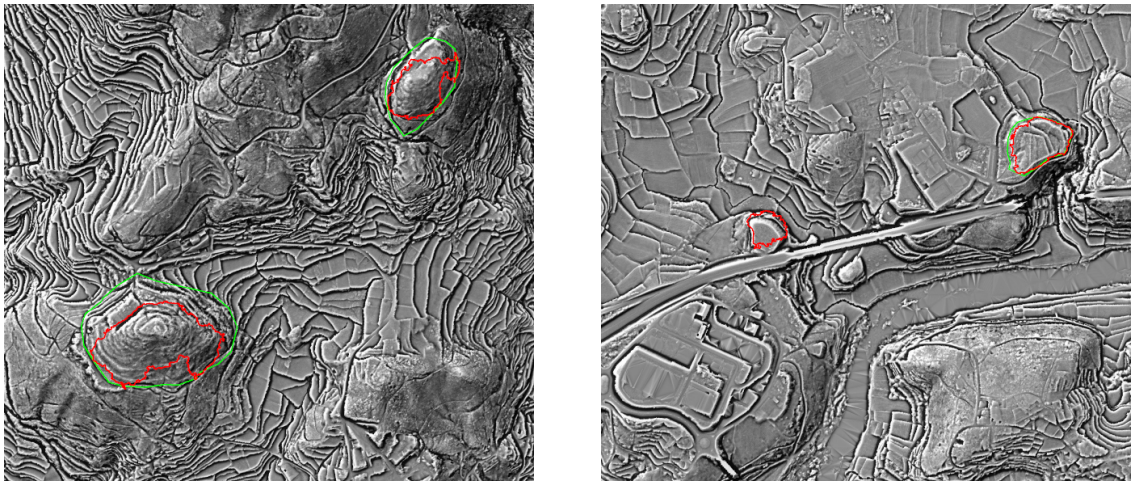

Figure 5.9: Examples of inference results from the copy-paste model for hillforts, the green corresponds to a ground truth, while the red corresponds to a detection from Unet.

5.2 YOLOv7-seg

YOLOv7-seg is a modification of YOLOv7, it is an integration of BlendMask with YOLOv7, each bouding box generated by YOLOv7 is then fed into BlendMask neural network in order to generate the segmentation.

5.2.1 Model training

YOLOv7-seg was trained exclusively using the simple dataset. The decision to use this dataset was driven by the superior performance in YOLOv7. Since the primary focus of the project is archaeological site localization rather than segmentation, the results are expected to be quite similar to those of YOLOv7. However, the ability to obtain segmentation could be advantageous for archaeological purposes.

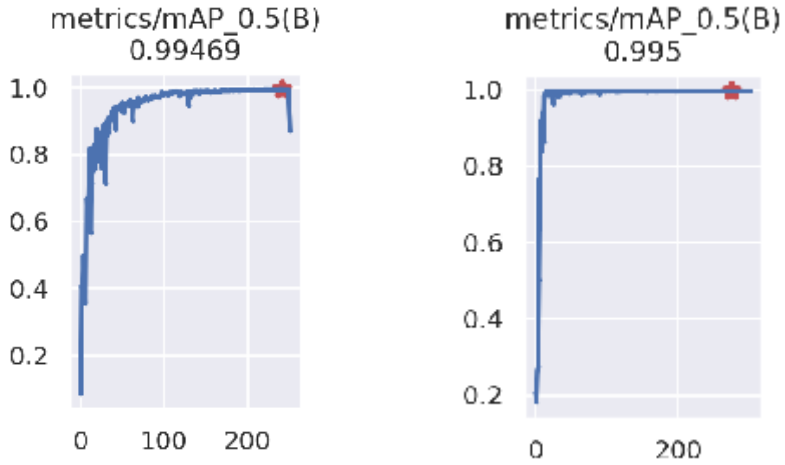

Figure 5.10: Results of YOLO-seg for tumulis(left) and hillforts(right)

As can be seen, the training is identical to YOLOv7 training (figure 4.5 and figure 4.11), for the same dataset, as expected

5.2.2 Inference

In the inference, a script similar to unet's was used, but instead of the model's result format being in table format, the result comes in polynomial format.

The table 5.3 shows the results from the YOLOv7-seg inference.

	Tumulis	Hillforts
Total	1410	1624
Validations	1009	553
True positives	213	90
False positives	1197	1534
False negatives	63	73

Table 5.3: Results of inference(YOLO-seg)

As expected, the results are not much different from YOLOv7.

The image 5.11 shows some examples of the results from the YOLOv7-seg.

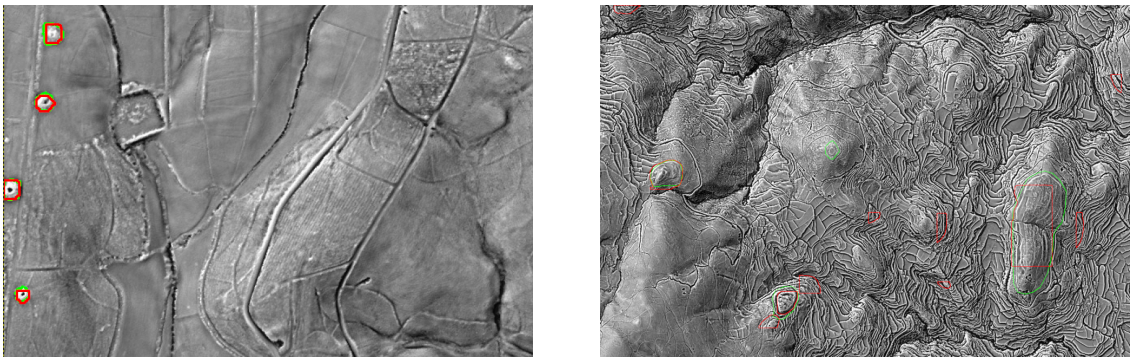

Figure 5.11: Examples of the results from YOLOv7-seg, the green represents a ground truth and red represents detections from YOLOv7-seg

CHAPTER 6

Conclusions

This section highlights the insights gained from the analysis of the work that is currently being developed on the topic of this dissertation, as well as how the existing solutions will support the construction of this dissertation's work

Locating archaeological sites is a difficult task that requires several steps. Over the years, methods have evolved from intensive human sweeping of the terrain to more sophisticated methods such as LiDAR. These modern methods can produce an enormous amount of data. This paper describes an attempt to locate archaeological sites using deep learning methods. As can be seen in the paper, two deep learning architectures were used, YOLOv7 and Unet.

The whole procedure allowed us to obtain successful results even with a small amount of initial data and two archaeological objects with very different shapes, tumuli with a much more regular shape and size, and hillforts that can have different shapes and sizes.

One interesting fact was that YOLOv7 worked much better on the mounds than on the hillforts, because it is an object detection model, and mounds have a much more uniform shape. On the other hand, YOLOv7 performed very poorly on hillforts, probably because of their irregular shape. The fact that mounds have a uniform shape was one of the reasons to try a segmentation model.

Unet, although it is an old model architecture, it performed well on both tumulis and hillforts.

It was also found that the copy-paste augmentation method works much better with Unet than with YOLOv7, most likely because Unet focuses more on the object rather than the background, which is especially useful due to the fact that copy-paste can create collages in unrealistic places.

For future work, in an attempt to improve the results, other methods of creating the images could be used, such as the enhanced MSTP. For validation, other models could be tried, or not all the annotations could be used to get some information on how the model performed, and those annotations could be used to validate the model.

References

[1] A. G.-M. H.A. Orengo, «A brave new world for archaeological survey: Automated machine learning-based potsherd detection using high-resolution drone imagery», *Journal of Archaeological Science*, vol. 112, 2019.

[2] G. W. N. M. e. a. L. Luo X. Z. R. Lasaponara, «Airborne and spaceborne remote sensing for archaeological and cultural heritage applications: A review of the century (1907–2017)», *Remote Sensing of Environment*, vol. 232, no. 3, 2019.

[3] C. P. L. Dylan S. Davis Gino Caspari and M. C. Sanger, «Deep learning reveals extent of archaic native american shell-ring building practices», *Journal of Archaeological Science*, vol. 132, 2021.

[4] H. A. O. Iban Berganzo-Besga, F. Lumbreras, M. Carrero-Pazos, J. Fonte, and B. Vilas-Estévez, «Hybrid msrm-based deep learning and multitemporal sentinel 2-based machine learning algorithm detects near 10k archaeological tumuli in north-western iberia», *Remote sensing*, 2021.

[5] *Aerial reconnaissance in world war i.* [Online]. Available: `https://www.centennialofflight.net/essay/Air_Power/WWI-reconnaissance/AP2.htm`.

[6] *Landsat 8 bands and band combinations.* [Online]. Available: `https://gisgeography.com/landsat-8-bands-combinations/`.

[7] S. L. Su Yang Miaole Hou, «Three-dimensional point cloud semantic segmentation for cultural heritage: A comprehensive review», *Remote sensing*, 2023.

[8] L. H.-M. Alexandre Guyot Marc Lennon, «Objective comparison of relief visualization techniques with deep cnn for archaeology», *Journal of Archaeological Science: Reports*, vol. 38, 2021.

[9] T. L. Alexandre Guyot Laurence Hubert-Moy, «Detecting neolithic burial mounds from lidar-derived elevation data using a multi-scale approach and machine learning techniques», *Remote sensing*, 2018.

[10] T. L. Alexandre Guyot Marc Lennon and L. Hubert-Moy, «Combined detection and segmentation of archeological structures from lidar data using a deep learning approach», *Journal of Computer Applications in Archaeology*, 2021.

[11] R. Hesse, «Lidar-derived local relief models (lrm) – a new tool for archaeological prospection», *Archaeological prospection*, vol. 17, 2010.

[12] *Understanding of convolutional neural network (cnn) — deep learning.* [Online]. Available: `https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148`.

[13] *Machine learning vs deep learning.* [Online]. Available: `https://levity.ai/blog/difference-machine-learning-deep-learning`.

[14] *Comunidade intermunicipal do alto minho.* [Online]. Available: `https://www.cim-altominho.pt/pt/`.

[15] [Online]. Available: `https://aws.amazon.com/what-is/overfitting/`.

[16] H.-Y. M. L. Chien-Yao Wang Alexey Bochkovskiy, «Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors», 2022.

[17] G. Jocher, *Tips for best training results.* [Online]. Available: `https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/`.

[18] DGT, *Cartografia de uso e ocupação do solo*, 2018. [Online]. Available: https://smos.dgterritorio.gov.pt/cartografia-de-uso-e-ocupacao-do-solo.

[19] C. Hughes, *Yolov7: A deep dive into the current state-of-the-art for object detection*, 2022. [Online]. Available: https://towardsdatascience.com/yolov7-a-deep-dive-into-the-current-state-of-the-art-for-object-detection-ce3ffedeeaeb.

[20] [Online]. Available: https://paperswithcode.com/method/non-maximum-suppression.

[21] M. M. Dan Hendrycks Kimin Lee, «Using pre-training can improve model robustness and uncertainty», 2019.

[22] [Online]. Available: https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_novelty_detection.html.

[23] *Jakteristics*. [Online]. Available: https://jakteristics.readthedocs.io/en/latest/.

[24] K. E. Koech, *Object detection metrics with worked example*, 2020. [Online]. Available: https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e.

[25] P. S. Marco Fiorucci Wouter B. Verschoof-van der Vaart, «Deep learning for archaeological object detection on lidar: New evaluation measures and insights», *Remote sensing*, 2022.

[26] A. T. Vinorth Varatharasan Hyo-Sang Shin and N. Colosimo, «Improving learning effectiveness for object detection and classification in cluttered backgrounds», 2019.

[27] A. E.-S. Eslam Mohamed Abdelrahman Shaker and M. Hadhoud, «Insta-yolo: Real-time instance segmentation», 2021.

[28] P. F. Olaf Ronneberger and T. Brox, «U-net: Convolutional networks for biomedical image segmentation», 2015.

[29] *Unet repository*. [Online]. Available: https://github.com/milesial/Pytorch-UNet.