Universidade de Aveiro
2022

Fabricio
Botelho

IDENTIFICAÇÃO DE SÍTIOS ARQUEOLÓGICOS EM IMAGEM AÉREA UTILIZANDO APRENDIZAGEM PROFUNDA

IDENTIFICATION OF ARCHAEOLOGICAL SITES IN AERIAL IMAGE USING DEEP LEARNING

Universidade de Aveiro
2022

Fabricio
Botelho

IDENTIFICAÇÃO DE SÍTIOS ARQUEOLÓGICOS EM IMAGEM AÉREA UTILIZANDO APRENDIZAGEM PROFUNDA

IDENTIFICATION OF ARCHAEOLOGICAL SITES IN AERIAL IMAGE USING DEEP LEARNING

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

Professora Doutora Susana de Jesus Mota
Professora Auxiliar, Universidade de Aveiro

vogais / examiners committee

Doutor João Mário Martins da Fonte
Postdoctoral Research Fellow, University Of Exeter - Department Of Archaeology

Professor Doutor António José Ribeiro Neves
Professor Auxiliar, Universidade de Aveiro

agradecimentos / acknowledgements

Gostaria de agradecer a todas as pessoas que tornaram possível a realização desta dissertação.
A todos os professores da Universidade de Aveiro que ao longo do meu percurso académico transmitiram o seu conhecimento.
Agradeço ao professor António Neves e à professora Pétia Georgieva pela orientação, sugestões e apoio oferecido.
Ao Daniel Canedo por toda a ajuda e conselhos.
À Comunidade Intermunicipal do Alto Minho e à empresa ERA-arqueologia pela disponibilização dos dados arqueológicos que tornaram possível a realização desta dissertação.
Aos meus pais e restante família que sempre fizeram tudo para que nada faltasse.
A todos os meus amigos um obrigado por todo o apoio dado ao longo destes 5 anos.

Palavras Chave

Arqueologia, LiDAR, Aprendizagem profunda, Aprendizagem automática, Sensoriamento remoto

Resumo

Esta dissertação foi desenvolvida no âmbito do projeto ODISSEY financiado pelo programa PORTUGAL2020. Este projeto tem como objetivo desenvolver uma plataforma integrada de informação geográfica destinada a arqueólogos e técnicos de património. A identificação de sítios arqueológicos produz informações complementares, que permitem consolidar as fontes de informação patrimonial já existentes. A identificação e deteção são realizadas de forma automática através de técnicas de aprendizagem profunda, com base em dados provenientes de métodos não intrusivos, como por exemplo, LiDAR. Contudo, é necessário um pré-processamento e anotação destes dados por forma a tornar possível o uso de algoritmos de aprendizagem profunda. Nesta dissertação foram usados dados adquiridos na área do distrito de Viana do Castelo situado no norte de Portugal, onde existem 136 objetos arqueológicos identificados e conhecidos por mamoas. Após todo o processamento de dados estar finalizado, é possível aplicar algoritmos de aprendizagem automática, tendo sido explorados nesta dissertação os modelos YOLOv5, Mask R-CNN e CNN. O treino com YOLOv5 e Mask R-CNN é feito através da afinação de modelos pré-treinados. É ainda realizado um estudo acerca do impacto do tamanhos da caixa delimitadora a serem usados na anotação das mamoas. O algoritmo CNN personalizado apenas é usado para a melhor dimensão encontrada com YOLOv5 e Mask R-CNN. No final, é feita a inferência a toda a imagem do distrito por forma a verificar o comportamento dos modelos obtidos e descobrir possíveis objetos arqueológicos ainda não identificados por ação humana. Desta forma, esta descoberta é transmitida através de coordenadas geográficas para permitir aos especialistas uma ida ao terreno e confirmar a descoberta.

Keywords Archaeology, LiDAR, Deep Learning, Machine Learning, Remote Sensing

Abstract

This dissertation was developed within the scope of the ODISSEY project financed by PORTUGAL2020 program. This project aims to develop an integrated platform for geographic information aimed at archaeologists and heritage technicians. The identification of archaeological sites produces complementary information, which allows the consolidation of existing heritage information sources. Identification and detection are performed automatically through deep learning techniques based on data from non-intrusive methods, such as LiDAR. However, pre-processing and annotation of these data are necessary to make possible the use of deep learning algorithms. This pre-processing is necessary because the data is not correctly annotated to be used. In this dissertation, the data acquired in the area of the district Viana do Castelo located in the north of Portugal were used, where there are 136 archaeological objects identified and known as 'mamoas'. After all data processing is completed, it is possible to apply machine learning algorithms, having been explored in this dissertation the YOLOv5, Mask R-CNN and CNN models. Training with YOLOv5 and mask R-CNN is done by fine tuning with pre-trained models. A study is also carried out on the impact of the sizes of the bounding box to be used in the annotation of 'mamoa'. The custom CNN is only used for the best dimension found with YOLOv5 and Mask R-CNN. In the end, an inference is made to the entire image of the district to verify the behavior of the models obtained and to discover possible archaeological objects not yet identified by human action. In this way, this discovery is transmitted through geographic coordinates to allow specialists to go to the field and confirm the discovery.

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

ALS	Airborn Laser Scanning
ASRS	Airborne and Spaceborne Remote sensing
CART	Classification and Regression Tree
CNN	Convolutional Neural Network
COCO	Common Objects In Context
DEM	Digital Elevation Model
DFM	Digital Feature Model
DSM	Digital surface Model
DTM	Digital Terrain Model
JSON	JavaScript Object Notation
GEE	Google Earth Engine
GUI	Graphical User Interface
LiDAR	Light Detection and Ranging
LRM	Local Relief Model
mAP	mean Average Precision
MSRM	Multi Scale Relief Model
MSTP	Multi Scale Topographic Position
ReLU	Rectified Linear Unit
R-CNN	Region Based Convolutional Neural Networks
RF	Random Forest
SAR	Synthetic Aperture Radar
SLRM	Simple Local Relief Model
YOLO	You Only Look Once

CAPÍTULO 1

Introdução

Este capitulo faz uma introdução ao tema desta dissertação. Apresenta as principais características e motivos que levam à sua realização. Contém os objetivos que se pretendem alcançar ao longo do trabalho. Por último é feita uma descrição da dissertação.

1.1 Motivação

Fazer pesquisas pela superfície em busca de objetos arqueológicos consiste em longas caminhadas realizadas por arqueólogos e estudantes, as quais envolvem percorrer uma determinada área que pode ser muito extensa e de difícil acesso [1]. Automatizar todo este processo resolve essencialmente diversos problemas associados à mão de obra, custo e a métodos intrusivos frequentemente usados.

Nos últimos anos o uso da inteligência artificial na arqueologia tem crescido de uma forma exponencial, embora existam alguns métodos menos precisos. Com a chegada de modelos relacionados com deep learning, especialmente Convolutional Neural Network (CNN)'s, que se baseiam na análise de dados multi dimensionais para o padrão de reconhecimento faz a precisão aumentar e proporcionar resultados com uma maior confiabilidade [2].

Grande parte dos estudos publicados que fazem o uso de aprendizagem automática focam-se essencialmente nas modificações das paisagens, contudo isto não produz informações complementares acerca de elementos arqueológicos. Este tipo de informações pode permitir, por exemplo, visualizar o impacto que algumas populações antigas tiveram num certo lugar.

O deteção remota são todas as técnicas que usam dispositivos de contacto não intrusivo para observar alvos de interesse na superfície da terra [3]. Permite identificar sítios arqueológicos com um custo bastante baixo em comparação com outros métodos, como datação por C14 e a realização de escavações que são bastante mais dispendiosos em termos de tempo e custo. Exemplos de métodos não intrusivos podem ser o Light Detection and Ranging (LiDAR) e imagem espectral. O LiDAR, por vezes também conhecido por Airborn Laser Scanning (ALS), facilita a visualização com algum detalhe de uma superfície coberta por grandes ou pequenas densidades florestais permitindo fornecer um mapeamento claro do terreno. Esta tecnologia

tem sido usada nos últimos tempos, nomeadamente na deteção de ruínas arqueológicas cobertas por vegetação. Esta deteção baseia-se na procura de anomalias no terreno, as quais podem ser chamadas de vestígios arqueológicos. O uso de LiDAR é normalmente realizado por aviões ou drones, contudo embora a utilização de drones permita obter dados mais precisos, não permite abranger uma grande área de terreno.

A conjugação de dados provenientes de LiDAR e de deep learning pode permitir realizar a identificação de sítios arqueológicos com níveis promissores de precisão e exatidão [4]. Alguns dos sítios possuem características próprias que os torna fácil de identificar através de LiDAR, tendo este uma resolução suficiente.

1.2 Objetivos

Esta dissertação foi desenvolvida no âmbito do projeto ODISSEY financiado pelo programa PORTUGAL2020. Este possui como objetivo desenvolver uma plataforma integrada de informação geográfica destinada a arqueólogos e técnicos de património. Pretende-se realizar a identificação de sítios arqueológicos e produzir informações complementares, que permitam consolidar as várias fontes de informação patrimonial. Pretende-se que a identificação e deteção sejam realizadas de forma automática aplicando técnicas de deep learning com base em dados provenientes de métodos não intrusivos, como por exemplo, LiDAR ou imagem multi espectral.

Por sua vez os dados são previamente fornecidos e a intenção é realizar uma revisão da literatura por forma a adquirir conhecimentos acerca da realização do seu pré-processamento. O objetivo é formar um conjunto de dados anotados que servem para fazer o treino, validação e teste. A anotação deve ter em conta as diferentes dimensões usadas para as caixas delimitadoras que envolvem o objeto em estudo numa imagem, uma vez que o terreno envolvente pode ter impacto na deteção.

Por fim, planeia-se realizar a inferência na imagem que representa toda a área do distrito em estudo. Para isso, o objetivo é usar os modelos obtidos para o melhor tamanho da caixa delimitadora e no final obter informação acerca da possibilidade de novos sítios identificados.

A Figura 1.1 apresenta o exemplo de um sistema LiDAR e um drone para o processo de aquisição de dados.

Figura 1.1: Fotografia de um aparelho LiDAR e drone para um processo de aquisição de dados em Viana do Alentejo, Junho de 2022.

1.3 Estrutura da dissertação

A dissertação está dividida em 6 capítulos. O primeiro faz uma introdução ao tema deste trabalho, bem como os motivos que levam à sua realização. De seguida, seguem os conceitos fundamentais que permitem obter o conhecimento necessário. Na terceira parte, é realizado um trabalho introdutório, que permite entrar em contacto com algoritmos de deep learning e adquirir conhecimentos necessários para o resto do trabalho. O Capítulo 4 apresenta todo o procedimento realizado e no seguinte os respetivos resultados de acordo com o procedimento usado. Por fim é apresentada a conclusão acerca do trabalho realizado.

CAPÍTULO 2

Conceitos Fundamentais

Neste capítulo são apresentados tópicos importantes relacionados com o tema desta dissertação que permitem transmitir o conhecimento necessário para a realização deste trabalho. Será abordado o sensoriamento remoto, os diferentes modelos digitais de elevação de terreno, algumas técnicas de visualização de relevo e por fim a aprendizagem profunda.

2.1 Sensoriamento remoto

O sensoriamento remoto é uma ferramenta não intrusiva, sendo cada vez mais usada por especialistas. Existem várias técnicas que não usam contacto direto com a superfície terrestre para obter informações acerca de um determinado alvo. Além de possuir um baixo custo, permite ainda uma análise rápida de dados com diversas origens.

2.1.1 ASRS (Airborn and Spaceborne Remote Sensing)

O ASRS é usado na arqueologia para investigar e compreender algumas características arqueológicas. Para este trabalho a identificação é realizada através de imagens aéreas. Dependendo da técnica que possa ser usada nas imagens, as principais ferramentas ASRS podem ser de 4 tipos [3], sendo eles:

- Fotografia
- imagem multiespectral e hiperespectral
- Synthetic Aperture Radar (SAR)
- LiDAR (Light Detection and Ranging)

De notar que destes 4, a fotografia, imagem multiespectral e hiperespectral podem ser classificados como passivos e o SAR e LiDAR como ativos.

2.1.2 Fotografia

A fotografia aérea pode ser de dois tipos, vertical ou obliqua. As primeiras imagens aéreas foram adquiridas no início do século XX através de um balão militar. Com o avanço da tecnologia foi possível adquirir imagens que permitissem obter com maior rigor diferenças existentes no solo.

A Figura 2.1 é um exemplo de uma fotografia obtida para auxiliar a reconstrução da antiga cidade de Pequim.

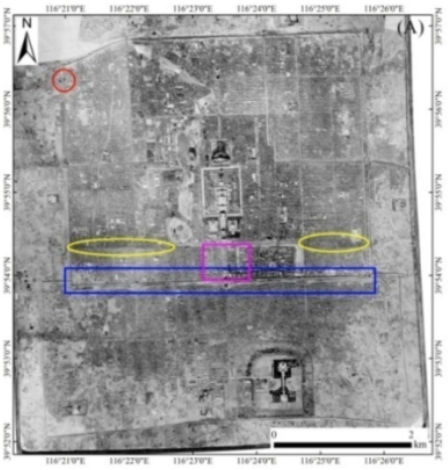

Figura 2.1: Fotografia aérea de 1945 usada na reconstrução da antiga cidade de Pequim [3].

2.1.3 Imagem multiespectral e hiperespectral

A principal diferença entre imagens multi e hiperespectral está no número de frequências medidas. No caso das imagens hiperespectrais são consideradas as bandas mais estreitas, na qual uma imagem pode ter centenas ou milhares de bandas. Nas imagens multiespectrais algumas bandas podem possuir uma resolução espacial cerca de 30 metros. Um exemplo de sensor multiespectral é o satélite Landsat 8 [5][6]. Para o uso destas imagens em aplicações arqueológicas é necessário ter em conta que os vestígios arqueológicos podem afetar as propriedades físicas, químicas e biológicas do solo [3]. Na Figura 2.2 é possível verificar que estas propriedades por sua vez afetam o crescimento de vegetação. É ainda possível observar alterações em solo arado, nomeadamente na sua cor.

A energia refletida permite fazer um registo dos comprimentos de onda. No caso de existir conhecimento de um certo sítio arqueológico é possível selecionar uma faixa espectral ideal e usá-la para melhorar a eficiência de observações do terreno através do uso de dados.

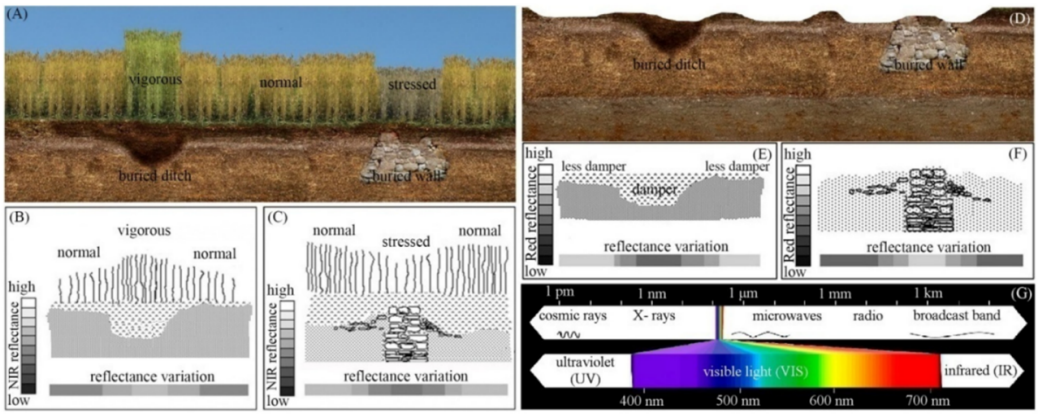

Figura 2.2: Variações na reflexão devido a marcas existentes no solo[3].

2.1.4 SAR

O principio de funcionamento de um sistema SAR aerotransportado baseia-se no uso de antenas duplas em que, permitem mapear áreas de interesse com alta resolução espacial. Em comparação com sistemas SAR de satélite, estes sistemas aerotransportados possuem tempos de repetição mais rápidos e permitem ultrapassar problemas relacionados com cobertura de nuvens ou mau tempo visto ser uma técnica ativa no sentido de detetar um determinado alvo independentemente das condições ambientais.

Em comparação com algumas técnicas de sensoriamento remoto, alguns estudos indicam que o SAR é superior na deteção de objetos enterrados, bem como, marcas no solo de pequeno e médio relevo. O SAR gera imagens através da radiação que recebe do retro espalhamento dos sinais radar emitidos. As imagens são compostas pela amplitude e pela fase de retro espalhamento. Deste modo, as principais características dos objetos que se encontram na superfície terrestre podem ser obtidas através da amplitude. Com a fase é possível obter dados topográficos. Um parâmetro importante para a investigação aqueológica incluído nos dados SAR é o valor de intensidade.

Este tipo de imagens já foi diversas vezes usado em aplicações arqueológicas, como por exemplo, através do uso direto destas imagens e da interpretação visual das mesmas foi possível identificar características e estruturas no solo, tais como, micro relevo ou relevo mais acentuado [3]. Esta identificação foi possível através do retro espalhamento que tal como foi referido, fornece informações sobre características da superfície. Para obter melhores resultados, é ainda possível usar métodos de filtragem, na qual, permite obter modelos digitais de terreno que são frequentemente usados na análise de superfícies.

A Figura 2.3 apresenta a transmissão e receção do retro espalhamento com diferentes tipos de superfície[3].

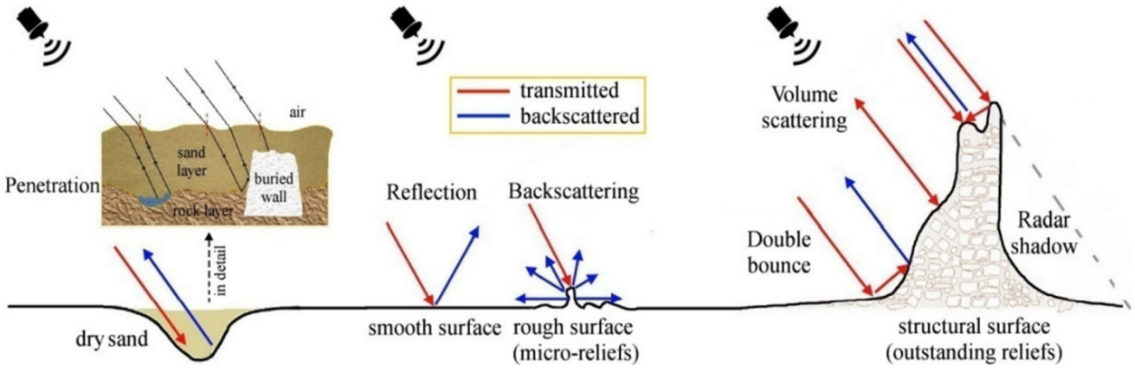

Figura 2.3: Transmissão e receção do retro espalhamento com diferentes tipos de superfície[3].

2.1.5 LiDAR

O LiDAR é uma tecnologia ótica de deteção remota que mede propriedades da radiação refletida, de modo, a obter a distância ou outra informação a respeito de um determinado objeto distante. O método mais usado para obter a distância de um determinado objeto é a utilização de laser pulsado. Cada pulso enviado pode ser refletido por um objeto, desta forma,

o tempo que demora entre o envio e a chegada de um pulso permite obter a distância entre o sensor e o objeto que se situa no solo. Um pulso emitido pode ter mais do que um retorno, por vezes pode ter 4 ou 5, como por exemplo, uma árvore através das suas folhas pode fazer diversos retornos, sendo que o último pulso recebido poderá ser do solo, dependendo da zona.

Existem sensores passivos e ativos. No caso dos passivos, não têm fonte própria de energia e podem fazer reflexão. No caso do LiDAR é um sensor ativo, pois gera pulsos laser e emite-os. Esta tecnologia é semelhante ao SAR, visto anteriormente, sendo algo que permite identificar características do terreno. Deste modo, é atualmente usado na arqueologia e pode ser usado em aplicações aerotransportadas, na qual, concentram-se na obtenção de modelos de superfície altamente precisos e na deteção de estruturas arqueológicas, mesmo que o solo seja constituído por uma cobertura vegetal bastante densa ou subaquática. É possível, por exemplo, colocar um sistema LiDAR num drone, na qual, este emite pulsos rápidos de luz laser e mede o tempo que a luz demora a retornar ao sensor, sendo insensível às condições de iluminação. Ao colher milhões de medidas por segundo, é possível criar um mapa 3D sobre a topografia da superfície terrestre de uma determinada área, sem árvores ou outros objetos que possam dificultar a análise do terreno.

Quando o sensor recebe as reflexões de pulsos é formada uma nuvem de pontos. Normalmente, este conjunto de pontos é guardado num ficheiro com o formato .LAS. Este, é um formato próprio para armazenar os dados obtidos através do LiDAR. As imagens são obtidas após o processamento destas nuvens de pontos. Numa primeira etapa é realizada uma classificação de pontos, na qual, pode ser feita a diferenciação entre vegetação e terreno. O passo seguinte é a interpolação onde é criada a superfície através da nuvem de pontos. O modelo com mais interesse na arqueologia é o Digital Terrain Model (DTM), que irá ser abordado mais à frente, na qual, todos os pontos que não correspondem ao terreno são eliminados. Após a finalização deste processo é possível aplicar várias técnicas de visualização. A Figura 2.4 ilustra uma nuvem de pontos obtida através de LiDAR com os pontos de retorno da vegetação e sem vegetação.

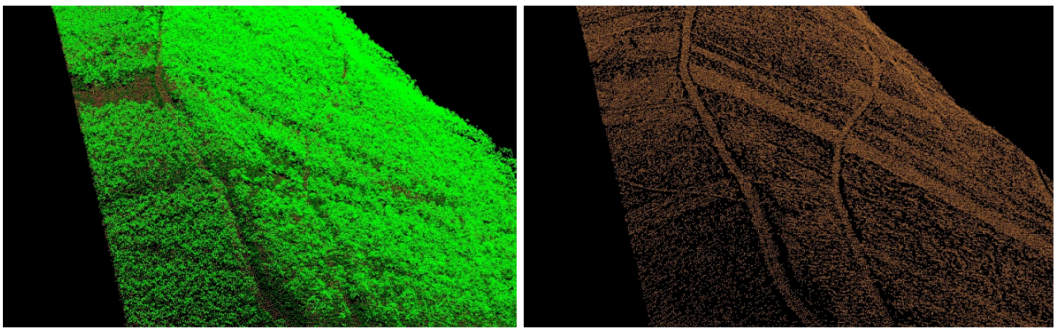

Figura 2.4: Exemplo de nuvem de pontos obtida através de LiDAR com e sem os pontos de retorno da vegetação[7].

Geralmente um sistema LiDAR infravermelho, permite ser usado quando existe vegetação densa. No caso de existir necessidade de efetuar uma pesquisa oceânica é normalmente usado um LiDAR batimétrico que emite luz no comprimento de onda $532nm$, na região da luz verde

[3]. Na Figura 2.5 encontram-se os dois exemplos de LiDAR referidos.

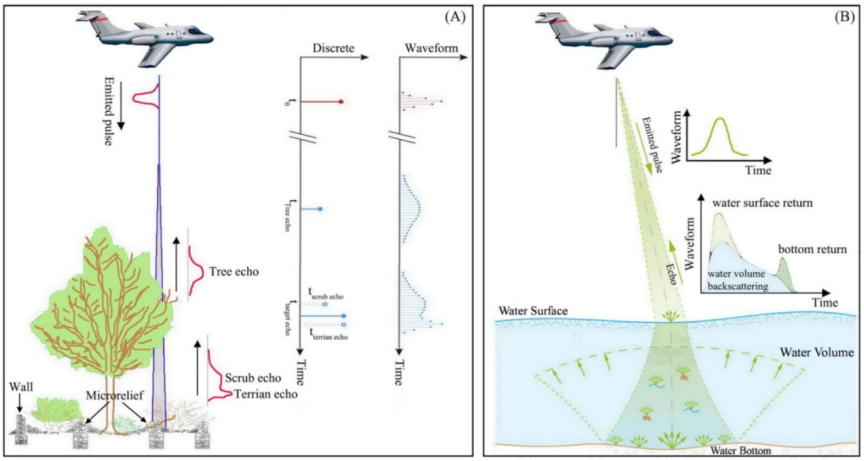

Figura 2.5: Exemplo de LiDAR infravermelho e batimétrico[3].

Quando os arqueólogos de sensoriamento remoto obtêm um DTM, realizam uma análise que permite identificar se o erro é grande ou pequeno, com o uso de métodos de filtragem. Desta forma, permite que possam dizer o quanto uma característica arqueológica é verdadeira ou não. As feições arqueológicas criam anomalias espaciais, contudo pode haver necessidade de fazer uma seleção destas, pois não existe certeza absoluta de que sejam objetos arqueológicos.

A utilização do LiDAR traz benefícios para a arqueologia devido à fácil penetração no solo, uma vez que a vegetação densa é um obstáculo na identificação de sítios arqueológicos. Os drones, ao contrário dos aviões podem voar a baixas altitudes podendo aumentar a precisão. De notar, que esta aumenta com o aumento da densidade do ponto.

O Airborne and Spaceborne Remote sensing (ASRS) é uma ferramenta útil para fazer o estudo de locais arqueológicos, contudo pode ser necessária a ida de um arqueólogo ao terreno para eliminar possíveis falsos positivos de locais que sejam muito pequenos para serem identificados por um sensor deste tipo.

2.2 Modelos digitais de elevação de terreno

Tal como referido é recorrente o uso de dados obtidos através de LiDAR aerotransportado. Um arqueólogo faz uma análise de modelos digitais de elevação, como o Digital Elevation Model (DEM), que surgem através do resultado do processamento de uma nuvem de pontos. É algo usado atualmente e importante devido aos bons resultados obtidos relativamente à eficiência. A Figura 2.6 mostra uma representação dos diferentes modelos que serão explicados de seguida.

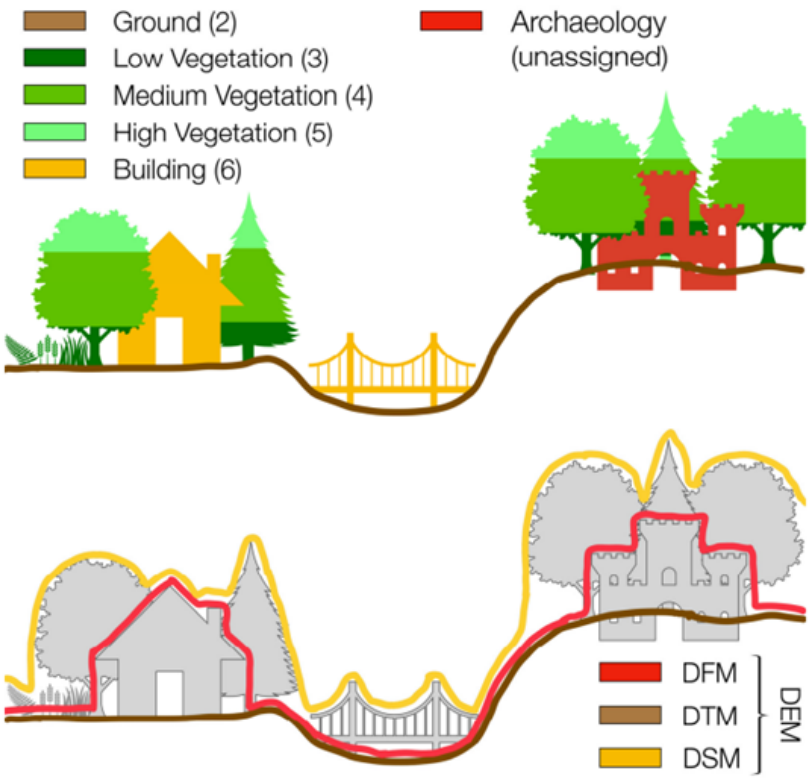

Figura 2.6: Diferença entre os modelos digitais DEM, DFM, DTM, DSM [8].

2.2.1 DEM/DTM

O DEM é considerado fundamental para o processamento de dados. Tanto o DEM, como o DTM referem-se a elevações do terreno. É comum existir pouca clareza sobre a diferença entre estes dois tipos de modelo. Isto acontece devido à variação de definições entre regiões. Por exemplo, na Europa é mais comum ser usado o DTM, sendo um sinónimo de DEM, mas nos EUA é usado principalmente o DEM e existe um significado diferente para DTM. No processamento de dados obtidos através de LiDAR o DEM abrange o DTM.

Um DEM é considerado uma representação topográfica da superfície terrestre. É uma grelha que contém valores de elevação. Esta, é representada por um número real presente em cada pixel. Neste caso elevação significa a altura acima do nível do mar.

Nos EUA, referem-se ao DTM como um conjunto de dados vetoriais composto por pontos em intervalos regulares, sendo considerado uma matriz que contém as características do terreno. Estas características podem ser, por exemplo, inclinações da superfície ou curvaturas. De notar, que estes atributos ou características podem ser obtidos através do DEM.

Em suma, a utilização destes dois modelos pode depender dos tipos de tecnologias usadas, como sensores e dos locais ou países onde é realizado o processamento de dados.

2.2.2 DSM

O Digital surface Model (DSM) (Digital Surface Model) é um modelo que representa a superfície e tudo o que nela está presente, como por exemplo, casas, pontes, estradas ou

vegetação. Deste modo, é fácil afirmar que raramente é ou poderá ser usado em aplicações arqueológicas. Isto, porque grande parte das informações que podem ser obtidas através deste modelo não têm qualquer interesse para o objetivo em causa.

Anteriormente, falou-se acerca do LiDAR aerotransportado e dos pontos de retorno. Deste modo, o DSM é constituído pelos primeiros pontos refletidos, na qual permitem formar a nuvem de pontos, contudo grande parte destes pode não estar associado apenas ao solo. Tem várias aplicações práticas, como por exemplo, na aviação ou planeamento urbano.

2.2.3 DFM

De acordo com os autores de [8], o DEM quando obtido através de dados LiDAR com o objetivo usá-los para fins arqueológicos é denominado por Digital Feature Model (DFM) (Digital Feature Model). Todas as características do solo e micro relevo são mantidas, desta forma é possível detetar, por exemplo, ruínas de paredes ou túmulos. De notar ainda, que este termo não é frequentemente usado e apenas pode servir para distinguir um DEM ou DTM relacionado com arqueologia de outros quaisquer. Normalmente é mais comum usar apenas os termos DEM ou DTM.

Pode ser útil ter o grau de confiança e qualidade do modelo obtido. Ainda no mesmo artigo [8], é apresentada uma técnica denominada de CART (Classification and Regression Tree) que é usada para encontrar anomalias num conjunto de dados que permite fazer uma classificação através do uso de condições ou regras. Na Figura 2.7, é possível ver um exemplo desta técnica. Tal como referido mostra que o seu funcionamento baseia-se no uso de condições lógicas. No final são apresentados 6 níveis de confiança, sendo o nível 6 o melhor resultado. As regras baseiam-se essencialmente nas características do terreno, tais como, densidade, declive e vegetação.

Tal como referido no ínicio deste tópico o DFM apresenta características arqueológicas baseadas no solo e micro relevo. Deste modo, pode-se afirmar que existe uma combinação com o DTM, tal como se pode ver na Figura 2.8. Os principais tipos de morfologia usados por arqueólogos são os dois primeiros, embedded feature e partially embedded features. Os restantes não são usados, pois contêm um número reduzido de objetos ligados ao solo, na qual podem trazer pouco interesse e um aumento de falsos positivos num algoritmo de machine learning, posteriormente aplicado.

Um fator importante neste tipo de modelos é a resolução, na qual refere-se ao tamanho do pixel, desta forma, a resolução será maior quanto menor for o tamanho do pixel. Na prática, quando são usados dados LiDAR é possível detetar objetos com $1m$ de diâmetro no mínimo, ou seja, é provável no futuro deste trabalho serem usados dados de imagem, em que 1 pixel corresponte a $1m$ de comprimento no terreno. Através do gráfico de confiança anteriormente falado pode-se obter uma resolução o mais próxima possível do ideal, que permita obter o maior número de dados possíveis, de modo a serem usados para obter bons resultados com um método de aprendizagem automática.

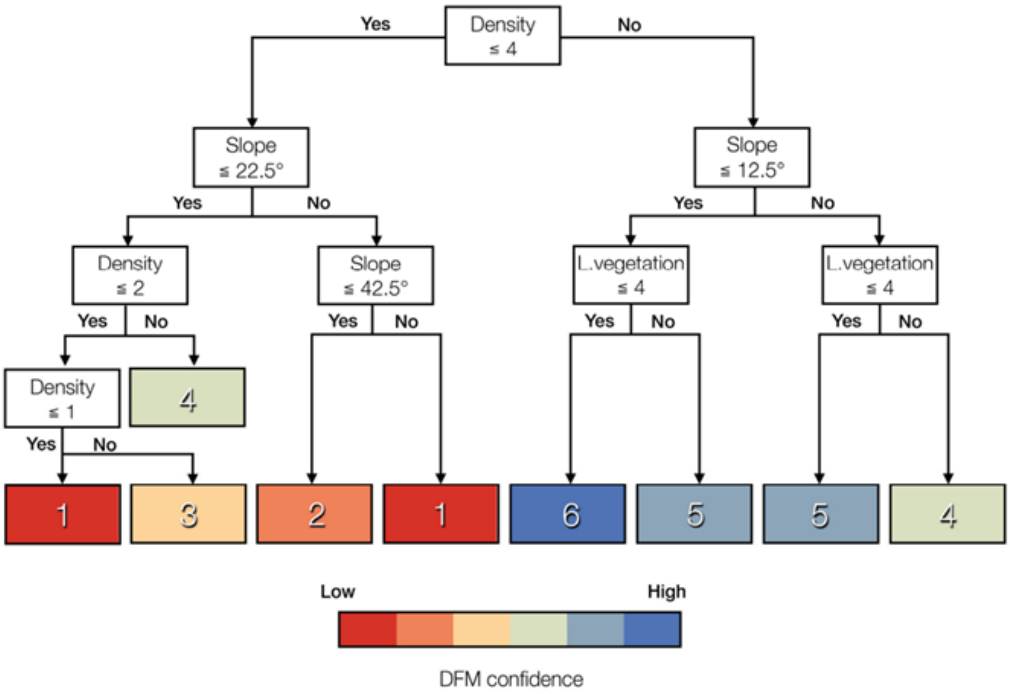

Figura 2.7: Análise de informação com a técnica CART, com o níveis de 1 a 6, em que 6 representa o nível de maior confiança[8].

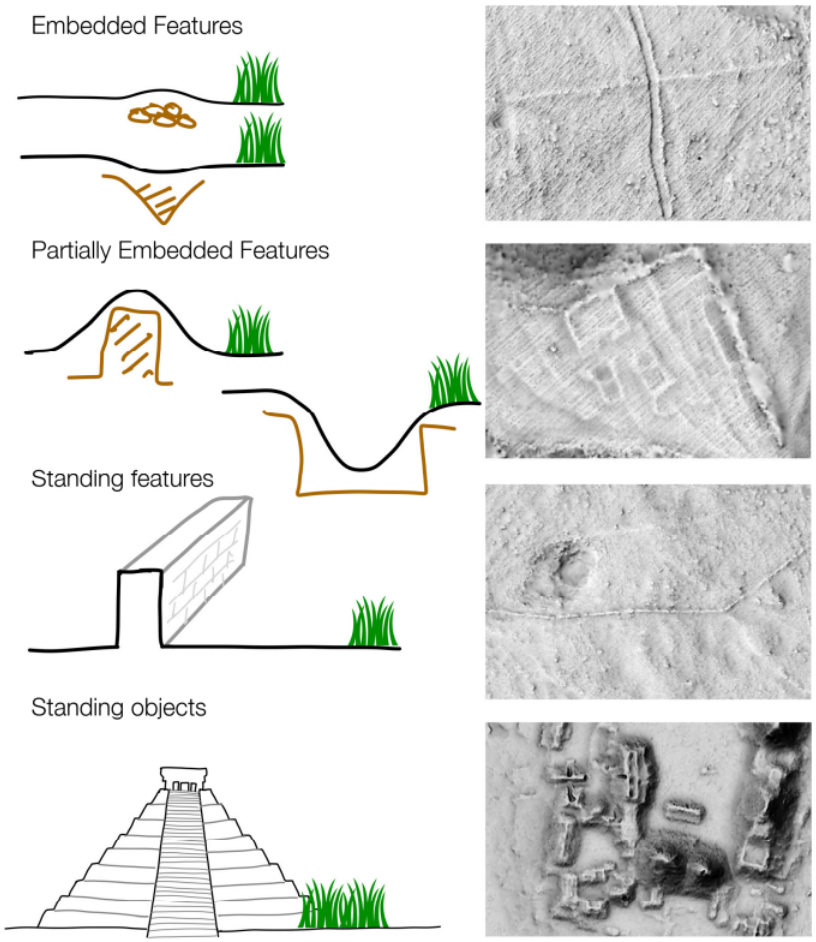

Figura 2.8: Tipos de características arqueológicas detadas via DFM [8].

Para aplicação de algoritmos de deep learning é comum existir um pré-processamento do modelo digital de terreno. Este tipo de modelos por vezes não destacam características necessárias, arqueológicas. No artigo [9] realizou-se uma primeira aproximação usando apenas o DTM original, (sem qualquer aplicação de técnicas de visualização), na qual, foi aplicado um algoritmo de deep learning para detetar sítios arqueológicos. Contudo, verificou-se que era necessário aplicar previamente uma determinada técnica de visualização para destacar objetos arqueológicos presentes no terreno, pois apenas foi obtida uma precisão de 21.81%. Existem diversas técnicas de visualização de terreno. Aqui serão apenas referidas as mais utilizadas, tais como, o LRM, MSRM e MSTP.

Uma ferramenta importante na visualização de relevo é a Relief Visualization Toolbox [10]. Esta recebe como entrada um modelo digital de terreno no formato GEOTIFF, no qual posteriormente podem ser selecionadas distintas técnicas de visualização.

2.3.1 LRM (Local Relief Model)

O Local Relief Model (LRM) é uma das técnicas mais usadas para fins arqueológicos. Isto é possível devido ao facto do LRM representar diferenças de elevação locais após remover as formas de paisagem em larga escala. Desta forma, é possível aumentar a visibilidade das características mais rasas do solo, sendo independente das condições de luminusidade. Por outras palavras [11], pode-se admitir que para obter um LRM aplica-se um filtro passa-baixo ao DTM ou DEM. Algumas características arqueológicas comuns detetadas através desta técnica de visualização, são [12]: estradas, pedreiras, montes funerários/tumulus e terraços agricolas.

2.3.2 MSRM (Multi Scale Relief Model)

O Multi Scale Relief Model (MSRM) possui o método de extrair características morfológicas de um modelo digital de terreno de alta ou baixa resolução, independentemente do formato terrestre em estudo. Deste modo, pode fornecer vantagens importantes em comparação com outras abordagens, visto que não é baseado na qualidade de imagens obtidas e foi desenvolvido para complementar as existentes abordagens de sensoriamento remoto. Esta técnica já foi usada, por exemplo, para ajudar na deteção de uma rede hidrográfica pré-histórica, na qual, foi possível mapear mais de $8000km$ de cursos de água [13] o que mostra a útilidade deste método para visualizar grandes áreas.

O artigo [9] mostra outro exemplo de utilização do MSRM, que tem como objetivo a deteção de túmulos. Neste, realizaram o estudo de três técnicas de visualização, na qual, o MSRM obteve melhor desempenho, pois produziu formas mais consistentes independentemente do tamanho dos túmulos.

2.3.3 MSTP (Multi Scale Topographic Position)

O Multi Scale Topographic Position (MSTP) pode ajudar a resolver alguns problemas relacionados com certas intervenções humanas que existem ao longo do tempo num certo local.

Após uma comparação com outras técnicas de visualização verifica-se que o MSTP destaca o contexto topográfico e fornece informações morfológicas que não são visíveis com outras técnicas [14]. Pode ser útil para a arqueologia, no qual, é possível obter bons resultados, como por exemplo, em [14] [15]. Contudo, apesar das suas vantagens não é possível obter medições topográficas, pois os valores dos pixels não representam elevações ou inclinações, mas sim uma matriz RGB.

2.4 Pré-processamento de dados

O dados representam um papel fundamental na deteção automática de objetos, na qual, podem influenciar o resultado final. Esta secção mostra como é possível gerar dados e fazer a sua anotação. Anteriormente foram referidas diversas técnicas de sensoriamento remoto para adquirir informações acerca da superfície terrestre, deste modo é necessário conhecer formas eficazes de extrair dados que possibilitem, por exemplo, a aplicação de algoritmos de deep learning.

2.4.1 Anotação de dados

Para o processo de aquisição de conjuntos de dados, existem várias abordagens e podem depender do tipo de informações que existe à partida, como por exemplo, ter um modelo de elevação do terreno ou imagens espectrais. No final, o objetivo é dividir o conjunto de dados para treino, teste e validação, e usá-los num algoritmo de aprendizagem automática. Normalmente, antes deste processo, é comum a ida de arqueólogos ao terreno, para validarem e anotarem as coordenadas de objetos arqueológicos que visualizaram através de técnicas ASRS.

Os autores do artigo [4] mostram uma forma de obter dados através do DTM. As imagens tinham o formato .tif e 10000×12500 pixels. Este formato, Tagged Image File Format, representa a imagem RAW. Primeiramente, removeram todos os pixeis que não continham qualquer descrição e para isso usaram uma ferramenta denominada por QGIS. Para visualizarem o DTM usaram uma técnica de visualização denominada Simple Local Relief Model (SLRM), que permite melhorar a visibilidade de anomalias no terreno. O objetivo é conseguir formar o maior número de dados com imagens. Existe um problema que aponta para o facto de existir overfitting, que ocorre quando existe uma memorização dos exemplos, em vez de os generalizar. Para resolução deste problema é sugerido obter exemplos diferenciados.

Na Figura 2.9, é possível ter uma ideia do método que é utilizado. Inicialmente, existe um conjunto de ficheiros constituído por:

- Um ficheiro shapefile por cada classe de objeto.
- Imagens LiDAR no formato GeoTIFF.

O ficheiro shapefile é um formato de ficheiro que armazena informações acerca das características geográficas. Neste caso contém bounding boxes ou anotações de coordenadas X e Y. A Figura 2.9 ilustra o processo que é aqui descrito. O objetivo é associar as bounding boxes às imagens LiDAR. Para isso, realizaram um script em linguagem python. Usaram o comando `gdalinfo`, que permitiu recuperar as coordenadas das imagens por forma a associá-las às

bounding boxes. Após isto, geraram um ficheiro CSV que contém várias informações, tais como, o tipo de objeto, as posições das bounding boxes e imagem associada. A equação:

$$f(x) = \frac{x - x_{min}}{x_{max} - x_{min}}(x^{'}_{max} - x^{'}_{min}) + x^{'}_{min}. \tag{2.1}$$

permitiu converter as coordenadas destas bounding boxes para pixels. De seguida extraíram as imagens, no qual foi verificado posteriormente se estas correspondiam a objetos do ficheiro CSV. Deste modo, foi possível anotar e formar o ficheiro com o conjunto de dados no formato .txt. Este ficheiro contém todas as informações acerca das coordenadas de pixel de cada bounding box e classe de objeto.

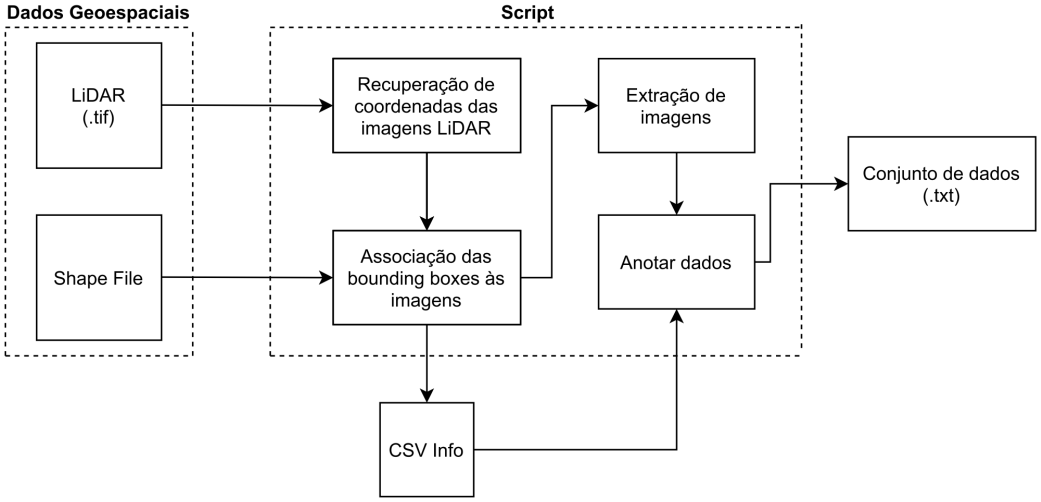

Figura 2.9: Pré-processamento de dados.

Uma abordagem diferente para obter dados pode ser encontrada no artigo [16]. Através de drones foi possível obter fotografias aéreas de um determinado sítio, no qual, posteriormente permitiu gerar um ortofotomosaico. Este, foi gerado através do alinhamento das fotografias e a criação de uma nuvem de pontos que permitiu obter um ficheiro .tif.

O objetivo é identificar cacos de cerâmica, para isso, usaram uma plataforma geoespacial, neste caso o Google Earth Engine, que além de conter bons recursos computacionais, pode implementar algoritmos de machine learning. A identificação não é baseada na cor, mas sim na textura. Teve em conta o facto da cerâmica apresentar uma superfície suave (valores semelhantes entre pixels) ao contrário do solo e objetos que se encontravam no arredor. Após uma análise de métodos de análise de textura observou-se que os melhores resultados obtidos correspondiam à análise do gradiente de magnitude, que calcula a variação dos pixels.

2.5 Aprendizagem profunda

A utilização de algoritmos deep learning revelou novos desafios, como por exemplo, maior quantidade de dados necessários para obter bons resultados. Estes algoritmos têm sido cada vez mais usados em aplicações arqueológicas, nomeadamente na deteção de objetos

arqueológicos numa determinada área. Contudo, tal como foi mostrado em [9], a forma comum semi-hemisférica dos túmulus permite a utilização de menos dados para realizar a deteção deste tipo de objetos através de uma CNN.

Um problema que pode ser associado a este tipo de abordagens é o facto de obter possíveis falsos positivos, como por exemplo, o formato pode ser semelhante a telhados de casas, piscinas ou outros tipos de estruturas que possam estar presentes no terreno, que não sejam objetos arqueológicos. Uma solução para resolver este problema é também demonstrada no artigo [9], no qual, usaram dados obtidos através do satélite Sentinel-2 e aplicaram um classificador Random Forest (RF) (Random Forest) para criar um raster binário que informa acerca dos locais arqueológicos onde podem estar esses túmulos arqueológicos. Posteriormente este resultado pôde servir como entrada no algoritmo de deep learning que fez diminuir o número de falsos positivos. Contudo, nada é ideal e é necessário ter em conta o facto do algoritmo RF eliminar alguns objetos arqueológicos. O Google Earth Engine (GEE) (Google Earth Engine) foi usado para processar a classificação RF através dos dados do satélite Sentinel-2, o que permitiu obter resultados de forma mais rápida.

2.5.1 CNN

O CNN é uma das arquiteturas de deep learning mais utilizadas sendo um algoritmo de classificação inspirado no córtex visual humano. É considerado um bom algoritmo para ser usado no processamento de imagens. O princípio de funcionamento baseia-se numa camada de entrada e de saída, na qual, pelo meio podem existir várias camadas ocultas. Este algoritmo recebe como entrada uma imagem, no qual, é processada e classificada como um determinado objeto. Esta imagem de entrada é vista como um array de pixeis, com dimensão igual ao tamanho da imagem, tal como, *largura*$\times$ *altura*$\times n^o$ *de bandas*. O número de bandas depende da imagem, no caso de ser RGB conta com 3 bandas e no caso de ser na escala de cinza possui apenas 1 banda. A estrutura de uma rede neural convolucional pode ser vista na Figura 2.10.

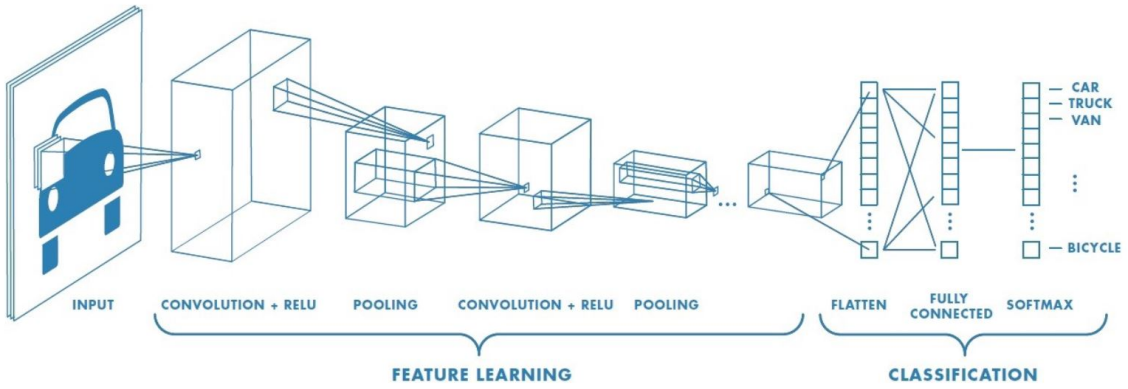

Figura 2.10: Exemplo de rede neural convolucional[17].

Cada imagem passa por camadas de convolução com filtros (kernels), pooling, camadas FC (fully connected) e a camada final pode ser uma função de ativação *softmax*, ou apenas uma *sigmoid* no caso de ser uma classificação binária. A camada de convolução é uma operação

matemática realizada com os valores de pixels da imagem e um filtro. Sendo a imagem com as 3 dimensões (height, width, depth): $h \times w \times d$ e o filtro: $fh \times fw \times d$. A saída terá o seguinte formato: $(h - fh + 1) \times (w - fw + 1) \times 1$. Este filtro move-se sobre os pixels de entrada. O número de deslocamentos de pixels realizados pelo filtro é denominado por stride. Por exemplo, quando o stride possui um valor igual a 2 significa que este filtro desloca-se 2 pixels de cada vez. Um aspeto importante poderá ser o facto do tamanho do filtro não encaixar ou coincidir com a imagem de entrada, nesse caso a imagem poderá ser preenchida com zeros, processo conhecido como zero padding, até o filtro encaixar perfeitamente na imagem de entrada. Outra opção poderia ser desprezar a parte da imagem onde o filtro não encaixa. A utilização da ReLU (Rectified Linear Unit) é importante, uma vez que possui bom desempenho [17]. O objetivo da sua utilização é introduzir não linearidade. A camada de pooling permite reduzir o número de parâmetros, mas ao mesmo tempo manter os mais importantes e pode ser feito de várias formas, como por exemplo, através da média ou simplesmente escolher o valor máximo de um conjunto de valores. Este, pode variar consoante o número aplicado de stride.

Por fim, para a classificação os mapas de features criados são totalmente conectados com o objetivo de criar um classificador. No final existe uma função de ativação que pode ser *softmax* ou uma *sigmoid*. A função *softmax* estima a probabilidade de um exemplo pertencer a uma das classes. De notar, no caso de existirem apenas duas classes, classificação binária, basta fazer o uso da função de ativação *sigmoid*.

Um dos problemas já falados acerca da arqueologia e da deteção de objetos é o facto da existência de pequenos conjuntos de dados. Desta forma, o CNN permite a utilização de modelos pré-treinados que possibilita a utilização dos respetivos parâmetros durante a transferência de aprendizagem reduzindo o tempo de treino e aumenta a generalização do algoritmo [18].

2.5.2 YOLO

O You Only Look Once (YOLO) é uma das arquiteturas CNN ainda não muito explorada na arqueologia. Permite a classificação de imagens, localização e deteção. O algoritmo YOLO permite detetar objetos de forma rápida visto que pode ser usado em tempo real. Além disso, permite obter bons resultados em termos de exatidão, uma vez que o número de erros relativos ao background é reduzido. Tal como é mostrado, [19], o YOLO tem atualmente várias aplicações, como por exemplo, na condução autónoma, vida selvagem e na segurança.

Exemplos da utilização do YOLO na arqueologia podem ser encontrados nestes artigos, [9], [20]. Ambos usaram dados LiDAR para obter o dataset e realizar o treino usando o algoritmo YOLO. Concluíram que obtiveram um bom desempenho e resultados favoráveis.

O algoritmo recebe como entrada imagens com as respetivas anotações escritas num certo ficheiro. Estas anotações possuem a indicação da classe, coordenadas do centro, largura e altura da bounding box. Como saída o algoritmo além da classificação tem a localização do objeto na imagem através de uma bounding box e respetiva probabilidade de um determinado objeto pertencer à classe. A imagem de entrada é dividida numa grelha com várias células, desta forma, cada célula é analisada com o objetivo de encontrar o ponto central de cada

objeto. Cada célula está associada a um vetor, na qual, a sua dimensão varia consoante o número de classes. $y = [p_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3, ...]$

- p_c é (0 ou 1) e especifica se há ou não um objeto na célula.
- $[b_x, b_y, b_h, b_w]$ especifica a bounding box se houver um objeto na célula.
- $[c_1, c_2, c_3, ...]$ especifica a qual classe pertence o objeto presente na célula.

Num simples caso, em que a imagem é divida em células 3x3 e existem 3 classes o volume total da saída será 3x3x8. Um dos problemas associados a esta abordagem na deteção de objetos é o facto de poder existir mais do que um bounding box para um determinado objeto numa imagem. O $IoU(IntersectionOverUnion) = \frac{intersectarea}{unionarea}$, mede a sobreposição entre duas bounding boxes e diz se a deteção de objetos está a funcionar bem. O valor IoU máximo igual a 1 significa uma deteção e precisão perfeita. O non-max suppression é a forma de garantir que cada objeto é detetado apenas uma vez. Basicamente descarta todas as bounding boxes com baixa probabilidade, em que, apenas resulta a que tiver maior probabilidade.

Um outro problema está relacionado com o facto de uma célula poder ter mais do que um ponto central de bounding boxes que representam dois objetos. Deste modo, para o exemplo anterior com 3 classes essa célula terá associada a si um vetor de 16 elementos e a solução poderá passar pela utilização de 2 anchor boxes e associar cada previsão a cada uma das anchor boxes.

A Figura 2.11 mostra a arquitetura da rede usada pela primeira versão do YOLO, na qual, é possível observar uma rede CNN para classificação e localização de objetos através de bounding boxes.

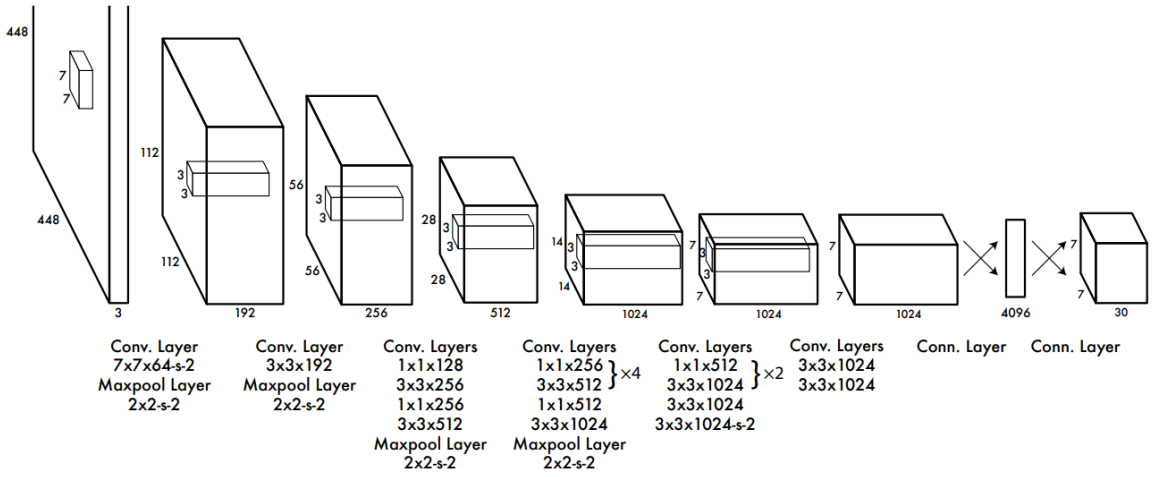

Figura 2.11: Arquitetura da rede CNN do YOLOv1 com 24 camadas convolucionais + 2 camadas fully connected. (As camadas convolucionais são pré-treinadas através do ImageNet dataset) [21]

2.5.3 Métricas de desempenho

Existem várias métricas que permitem analisar o desempenho dos algoritmos de aprendizagem profunda. Neste sub-capítulo é apresentado o conceito de cada, bem como, a sua importância neste trabalho.

Uma das formas mais fácil e conhecida para avaliar o desempenho é a matriz de confusão. Esta, não é nada mais do que uma simples tabela com duas dimensões que representam o que é real e previsto pelo algoritmo. Os valores apresentados por si, são: TP (True Positive), FP (False Positive), TN (True Negative) e FN (False Negative). Os TP são todos os casos que a classe prevista é positiva e igual à classe real. Os FP são os casos em que a classe prevista é positiva e não corresponde à classe real. Os TN representa a quantidade de classes negativas previstas corretamente. Os FN correspondem ao número de classes previstas como negativo, mas na realidade são positivas.

	Positivo	**Negativo**
Positivo	TP	FP
Negativo	FN	TN

Tabela 2.1: Exemplo de matriz de confusão.

Uma das métricas mais comuns é conhecida por accuracy que é a relação entre as previsões corretas e as previsões todas realizadas. O valor é obtido através da seguinte equação:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.2}$$

A proporção de identificações feitas corretamente é conhecida como sendo o precision, no qual, pode ser visto com a seguinte equação:

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

A proporção que representa os positivos reais identificados corretamente, é conhecido por Recall e calculado através da equação:

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Estas duas últimas métricas por si só não permitem avaliar o desempenho de um algoritmo. Para isso, existe uma outra métrica chamada F1-score que permite dizer qual é o melhor equilíbrio entre o precision e recall e é obtido da seguinte forma:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.5}$$

Nos algoritmos de deteção de objetos é muito comum aparecer uma métrica chamada mean Average Precision (mAP) que é obtida através do cálculo da área interior de um gráfico produzido através de vários valores de precision e recall.

Um fator importante na análise de desempenho são as perdas de treino e validação. Estas são frequentemente apresentadas em forma de gráfico, na qual, podem ajudar a verificar se a aprendizagem foi feita corretamente e estas perdas convergiram para um valor mínimo, uma vez que, a aprendizagem máquina pode ser definida como o objetivo de minimizar uma função de custo. De notar, que no caso das perdas de validação a não convergência para um valor mínimo pode significar o aparecimento de overfitting.

CAPÍTULO 3

Primeira abordagem com dados arqueológicos

Numa primeira abordagem ao tema desta dissertação foi refeito o trabalho demonstrado no artigo, [22]. Este apresenta no seu material suplementar todo o trabalho realizado, bem como, o dataset usado. Deste modo, foi útil para adquirir conhecimentos.

Os autores realizaram uma Graphical User Interface (GUI) a qual permite interagir com o utilizador, mais concretamente realizar anotações, treino e segmentação. O trabalho realizado mostra os resultados preliminares. Para o treino de dados foi usada a técnica de deep learning, Mask Region Based Convolutional Neural Networks (R-CNN). O CNN não requer extração de features, pois o sistema aprende a extraí-las e gera features invariáveis por convolução das imagens e filtros que são passadas para a próxima camada. Os dados de treino refletem as regiões onde as classes ou objetos estão presentes. Um dos motivos que levaram os autores ao uso do Mask R-CNN foi o facto deste método ter sido anteriormente aplicado com sucesso em outros dados arqueológicos.

O Mask R-CNN [23] permite segmentação de imagens e é construído a partir do algoritmo Faster R-CNN, contudo enquanto este último apresenta como saída uma bounding box e a classe de um objeto, o mask R-CNN possui ainda uma máscara que mostra a região de interesse do objeto presente no interior da bounding box. No artigo [22] é usada a biblioteca PixelLib [24] sendo construída através da biblioteca do machine learning, TensorFlow.

A anotação de imagens foi realizada através da ferramenta do python, LabelMe, em que, os objetos são individualmente classificados e no final é adquirido um ficheiro no formato json que é usado no treino. Algo importante é o facto de ter sido usado o modelo pré-treinado 'mask_rcnn_coco.h5' [25] obtido com um dataset Common Objects In Context (COCO). Isto permite usar a transferência de aprendizagem tirando proveito de um modelo pré-treinado através da reutilização de partes relevantes para treinar com um novo conjunto de dados. Os principais benefícios do seu uso é reduzir a necessidade de um conjunto de dados maior e permitir obter um modelo mais generalizado.

O modelo da rede neuronal usado foi o RESNET101 que possui 101 camadas. A GUI desenvolvida permite também o uso de apenas 50 camadas, RESNET50, que permite obter resultados de forma mais rápida, embora possam ser piores.

O dataset é composto por fotografias adquiridas através de uma camera montada num helicóptero. O número total de imagens é 383 e são consideradas 3 classes de sítios arqueológicos. Os dados foram organizados da seguinte forma:

- 286 imagens para treino
  - 94 mounded sites
  - 68 qanats
  - 124 ruined structures
- 71 imagens para validação
  - 18 qanats
  - 22 mounded sites
  - 31 ruined structures
- 26 imagens para teste
  - 7 mounded sites
  - 7 qanat
  - 12 ruined structures

A Figura 3.1 ilustra um exemplo de imagem para cada classe.

Figura 3.1: Exemplos de 3 fotografias das 3 classes de dados: 1-mounded site, 2-qanat, 3-ruined structure. [22]

Os ficheiros JavaScript Object Notation (JSON) obtidos através da ferramenta LabelMe são anotações de polígonos sendo compostos pelo ID de cada imagem, nome, valores de pixels, coordenadas das bounding boxes e as dimensões das imagens. Este tipo de ficheiros são considerados descritivos, uma vez que são legíveis por humanos e de fácil leitura. Apresentam a descrição ou propriedades de um determinado objeto.

Os resultados apresentados em [22] são baseados no mean average precision, mAP e perdas de validação, contudo pretendem no futuro realizar um estudo mais detalhado com precision, recall e F1 score. A avaliação para o melhor modelo obtido durante o treino apresentou perdas

de validação de 1.84 e um mAP de 0.2. A classe qanat foi a que apresentou o pior resultado de identificação, enquanto que a classe ruined structure obteve melhores resultados. Contudo, isto pode ser justificado pelo facto dos dados não serem balanceados e o dataset conter um número muito baixo de exemplos para aplicar um modelo de deep learning.

Nas subsecções 3.1 e 3.2 é detalhado todo o trabalho que foi refeito baseado no que foi explicado anteriormente. A secção 3.1 mostra a abordagem com a técnica Mask R-CNN. Contudo, para comparar resultados com outras técnicas de deep learning foi usado o YOLOv5 que se encontra descrito na secção 3.2. Embora os resultados apresentados não sejam muito favoráveis o uso destas técnicas é muito importante, pois serve para adquirir capacidades e usar futuramente durante esta dissertação, tal como irá ser mostrado nos seguintes capítulos.

3.1 Abordagem com mask R-CNN

Com o dataset apresentado na introdução deste capítulo aplicou-se o Mask R-CNN. O código foi escrito através da ferramenta Jupyter Notebook no Google Colaboratory, pois permite o uso de recursos computacionais necessários, visto que apresenta uma GPU Tesla k80 de 12GB de RAM com um limite de utilização até 12 horas. A versão do tensorflow e keras usada foram 1.13.1 e 2.0.8, respetivamente. O treino usa um modelo pré-treinado, 'mask_rcnn_coco.h5', constituído por 80 classes do dataset COCO. O número de épocas utilizadas foi 100, contudo poderia ser um número menor, pois os valores de perdas de validação tendem a subir quando são atingidas cerca de 15 épocas. Deste modo, é guardado o melhor modelo que neste caso corresponde à época número 15. Foi ainda usado um batch size igual a 4 e um learning rate igual a 0.001. Durante o treino é aplicado o aumento de dados, aplicando técnicas de rotação e ruído nas imagens existentes. Isto permite ao modelo aprender diferentes representações dos objetos e diminuir o overfitting. A mean average precision, mAP, obtida foi 0.27 com um valor mínimo de perdas de validação de 1.75.

A Figura 3.2 apresenta as perdas relativas ao treino e validação.

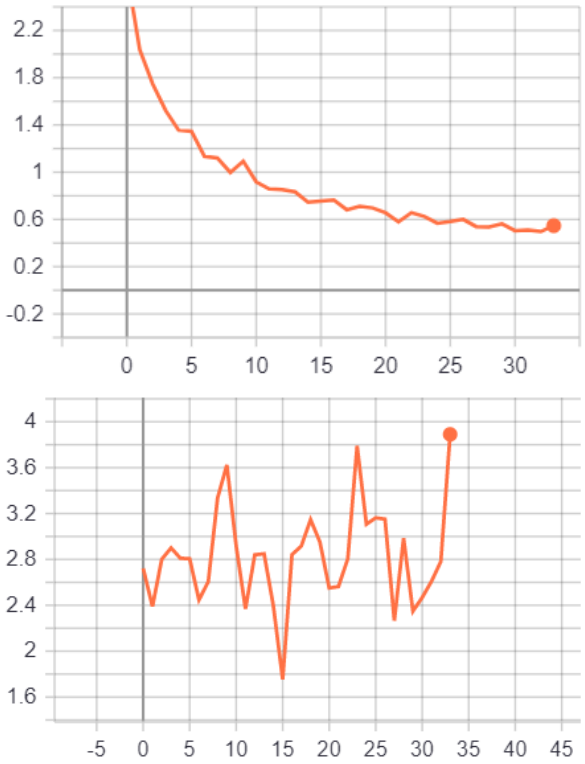

Figura 3.2: Gráficos com as perdas obtidas através do mask R-CNN. Superior: perdas de treino, Inferior: perdas de validação.

3.2 Abordagem com YOLOv5

Por forma a comparar os resultados com outro algoritmo deep learning foi usado o YOLOv5. Contudo, o dataset anteriormente usado necessitou de ajustes nas anotações, uma vez que o YOLO usa um formato especifico de anotação. Para isso foi desenvolvido um script que permitiu converter as anotações no formato COCO para o formato YOLOv5 e obter a informação necessária para formar um novo dataset, tal como, o nome de cada classe, dimensões da imagem e coordenadas de bounding boxes.

O dataset usado pelo YOLOv5 pode ser composto por 3 pastas, sendo elas: treino, validação e teste. Os dados são constituídos pelas imagens e pelos respetivos ficheiros no formato de texto. Cada linha nestes ficheiros possui a classe, coordenadas do centro e tamanho da bounding box, que corresponde a um objeto na imagem. De notar ainda que estes valores de coordenadas e dimensões são normalizados, divididos pelo tamanho da respetiva imagem.

Após a finalização e verificação do dataset foi possível realizar o treino através do YOLOv5. Este algoritmo foi descarregado do GitHub, [26]. Foram usadas cerca de 200 épocas, através da Figura 3.3 verifica-se que o modelo ainda não convergiu e possivelmente pode nem convergir mesmo com o uso de mais épocas.

O valor das perdas de validação logo nas primeiras épocas tende a aumentar. Desta forma, observa-se a presença de overfitting, em que o modelo ajusta-se bem com os dados de treino, mas perde generalização face a novos dados, pois decorou os dados de treino. Contudo, o YOLO através desta avaliação obtém dois tipos de modelos, tais como, o 'last.pt' e o 'best.pt'.

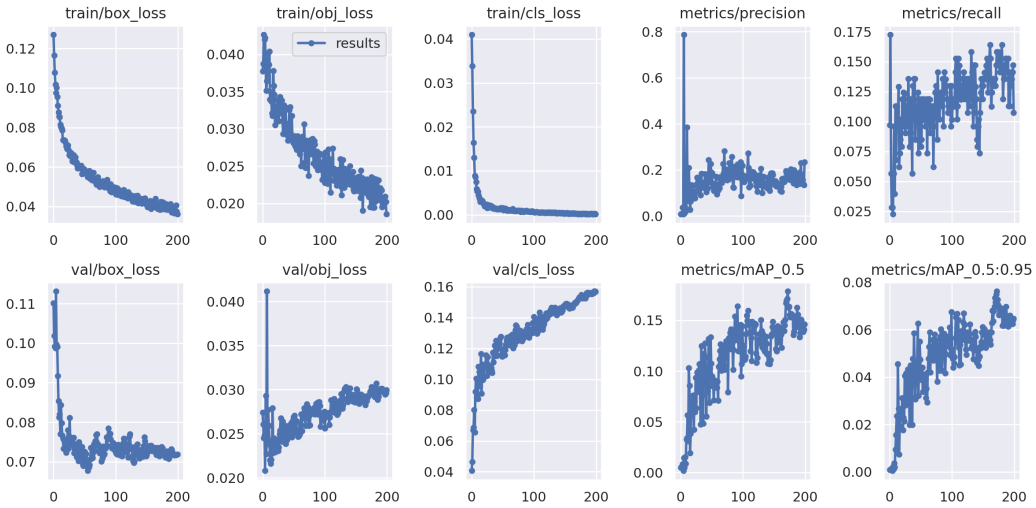

Figura 3.3: Resultados obtidos com YOLOv5 com o dataset mencionado.

Sendo que o último modelo pertence à última época e o melhor modelo corresponde à época com o menor valor de perda de validação, antes do começo da existência de overfitting. Os gráficos com as métricas precision, recall e mAP ilustram os valores obtidos em cada época, contudo para uma melhor análise com o melhor modelo adquirido durante o treino, a Figura 3.4 mostra as curvas para cada uma das classes. De notar, que estas métricas são obtidas através dos dados de validação. É possível verificar uma grande divergência nos valores obtidos para as 3 classes, no qual a classe qanat apresenta um valor muito reduzido, próximo de zero. O valor médio de average precision obtido foi 0.18.

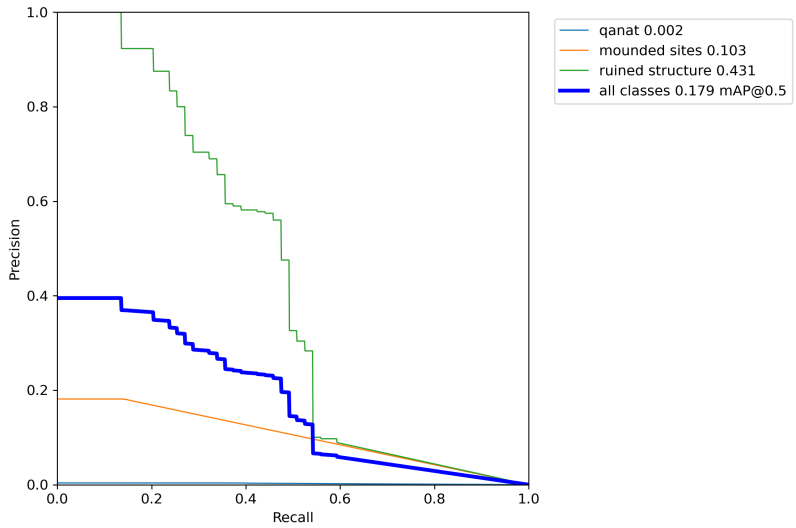

Figura 3.4: Curva de precision-recall.

A Figura 3.5 representa um exemplo de sucesso na deteção de um sítio arqueológico com a classe ruined structure, no qual, obteve boa pontuação de classificação, 0.93.

Verifica-se que não existiram melhorias face ao método anteriormente usado, Mask R-CNN, o que já seria de esperar tendo em conta o baixo número de dados. A classe qanat possui

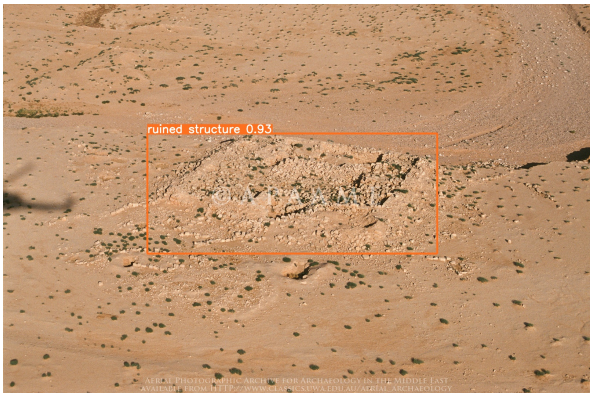

Figura 3.5: Exemplo de deteção da classe ruined structure com YOLOv5.

uma forma constante, circular com uma certa elevação e orifício no meio, contudo em algumas imagens possui exemplos não percetíveis, devido à forma como a imagem foi adquirida, deste modo, juntando o facto da existência de menos imagens para esta classe estes motivos serão a causa mais provável para os maus resultados. A classe mounded sites, embora possua mais imagens contem exemplos que diferem uns dos outros, o que faz dificultar o processo de aprendizagem do modelo.

A Tabela 3.1 apresenta os resultados presentes no artigo e os obtidos neste trabalho.

	Mask R-CNN (artigo)	Mask R-CNN	YOLOv5
mAP	0.2	0.27	0.18

Tabela 3.1: Resultados obtidos com Mask R-CNN e YOLOv5.

Comparando os resultados obtidos com os do artigo [22] pode-se dizer que foram identicos devido às razões referidas.

Em suma, este estudo serviu para adquirir conhecimentos acerca da utilização destas duas técnicas de deep learning que serão usadas durante esta dissertação.

CAPÍTULO 4

Identificação de sítios arqueológicos com dados LiDAR

Neste capítulo é demonstrado o trabalho realizado com dados LiDAR para a identificação de sítios arqueológicos. Os dados foram fornecidos através da Comunidade Intermunicipal do Alto Minho [27]. Estes dados por sua vez não constituem um dataset pronto para ser aplicado num modelo de deep learning. Os dados são constituídos pelo LRM no formato GEOTIFF de todo o distrito do norte de Portugal, Viana do Castelo, bem como a localização em formato shape file de 136 pontos. O objeto em estudo é denominado de 'mamoa'. Existem mais objetos, contudo estes 136 foram validados em campo por especialistas nas regiões de Arcos de Valdevez e do Laboreiro. A área em estudo encontra-se representada na Figura 4.1.

Após a aquisição do dataset o objetivo é a aplicação de diferentes abordagens com deep learning, como por exemplo, YOLO, Mask R-CNN e um algoritmo mais simples, como um CNN personalizado. Desta forma, será possível realizar um estudo com várias abordagens e verificar aquela que permite obter melhores resultados com este tipo e quantidade de dados. Um dos tópicos abordados é o aumento de dados realizado para melhorar o desempenho dos algoritmos usados.

O tipo de objeto arqueológico em estudo possui uma forma comum, circular com elevação, sendo o seu relevo visível com a técnica de visualização LRM.

Na Figura 1, em anexo, é possível observar o fluxo de trabalho detalhado neste capítulo.

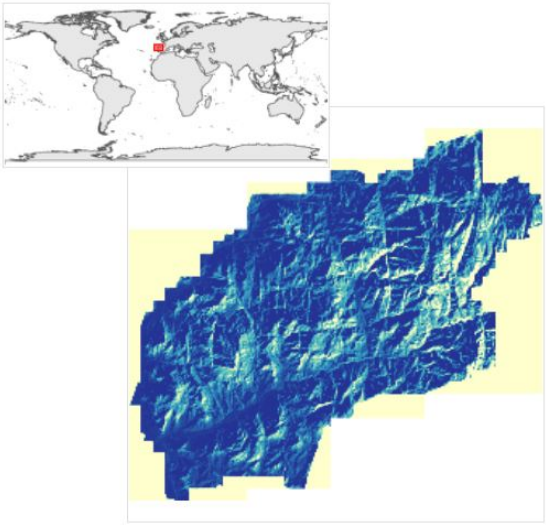

Figura 4.1: Área geográfica em estudo.

4.1 Pré-processamento dos dados

A Figura 4.2 mostra de uma forma resumida e simples todo o processo que irá ser descrito desde a aquisição dos dados até obter o dataset final.

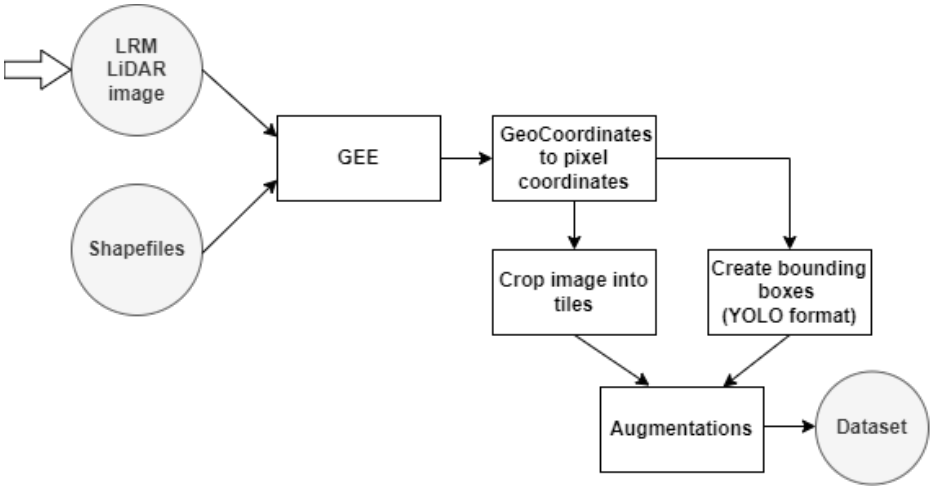

Figura 4.2: Esquema do pré-processamento de dados LiDAR para obter o dataset.

Os dados são constituídos pelo LRM que é a técnica de visualização aplicada ao DTM. O DTM resultou do processamento da nuvem de pontos obtida através do LiDAR aerotrasnportado. Deste modo, o pré-processamento de dados tem como principal objetivo obter o dataset com imagens e as respetivas anotações para posteriormente ser aplicado um algoritmo de deep learning.

O GEE foi usado para visualizar os dados e extrair o dataset. Esta ferramenta é uma plataforma baseada numa cloud para realizar análises científicas e visualizar datasets geográficos, tal como acontece neste caso. De notar, que esta plataforma oferece uma interface de programação gráfica bem como recursos computacionais para habilitar a análise de grandes datasets. Usa linguagens de programação python e JavaScript [28]. Uma das diferenças do GEE em relação a qualquer ferramenta de GIS é o facto desta plataforma ser baseada em cloud. Um script não é executado diretamente nos servidores da Google. É codificado num conjunto de objetos JSON, envia os objetos à Google para o processamento no lado do servidor e aguarda que a resposta seja enviada para o lado do cliente.

O GEE permite ser usado através do Google Colaboratory, [29]. Para isso foi necessário configurar a API do GEE no colab, usando a biblioteca `import ee`. Além disso é necessário efetuar a autenticação com utilizador e iniciar a sessão. Após estas etapas é possível trabalhar com os dados geoespaciais presentes no GEE.

Numa primeira análise aos dados no mapa do GEE foram associadas as coordenadas geográficas presentes nos shapefiles à imagem LRM, deste modo, foi possível visualizar a localização geográfica das mamoas. Através disto foi possível criar um dataset preliminar, em que foi obtida uma imagem para cada mamoa. Contudo, para obter exemplos de imagens sem mamoas desenhou-se um polígono no mapa numa região sem interesse e foram criados 136 pontos aleatórios que permitiram recortar imagens que foram posteriormente associadas às imagens com exemplos de mamoas. De notar ainda, que estas imagens foram recortadas da imagem LRM com uma escala de $0.5m/pixel$. Este dataset poderá ser usado, por exemplo, num algoritmo de machine learning ou um simples algoritmo de deep learning, CNN, tal como foi referido recebe como entrada uma imagem que pode representar ou não representar um objeto. Porém, para cumprir o objetivo de usar diferentes abordagens de deep learning, como YOLO e mask R-CNN foi necessário obter imagens com uma ou mais mamoas. Desta forma, em cada imagem existem dois tipos diferentes de estruturas que é necessário diferenciar, tais como, o background e o objeto em estudo.

Para a extração destas imagens foi necessário dividir a imagem LRM num mosaico. Contudo, foi necessário converter as coordenadas do mundo real, geográficas, para coordenadas de píxeis. Para isso, observou-se as informações que a imagem apresentava, sendo uma delas os seus limites e tamanho (largura e altura). Assim foi possível associar o canto superior esquerdo à coordenada de pixel (0,0) e o canto inferior direito com os valores do tamanho da imagem, ou seja, (69977,63086). Deste modo, foram obtidas duas equações para cada eixo que permitem converter uma coordenada geográfica (latitude e longitude) para coordenada de pixel:

- $px = 82639.50 \cdot LAT + 736092.97$
- $py = -110569.01 \cdot LON + 4661960.56$

De notar, que o sistema de coordenadas geográficas foi o EPSG:3763, sendo considerado um sistema global para Portugal continental.

Um dos problemas já referido em outros artigos é o facto de existirem poucos dados arqueológicos, o que também ocorre neste trabalho. A utilização de uma escala com $0.5m/pixel$

permite obter boa resolução. Decidiu-se dividir a imagem LRM num mosaico/blocos, em que cada bloco possuía uma resolução de $640 \times 640 pixels$, o que equivale a $320 \times 320m$ no terreno. Desta forma, cada imagem que compõe este dataset representa uma área equivalente a $102400m^2$. Para serem apenas recortadas imagens com mamoas e também obter o número presente em cada imagem foi realizada a função, `CountNumberOfMamoas(Xmin, Ymin, Xmax, Ymax)`, que recebe como parâmetros as coordenadas dos limite mínimo e máximo de cada bloco. Esta, por sua vez possui a lista de coordenadas em píxeis de cada mamoa já identificada na imagem LRM. Deste modo, através de um processo iterativo verifica-se se as coordenadas que correspondem ao bloco e é feita a contagem do número de objetos no seu interior. De notar, que esta função contém a lista de mamoas com as coordenadas em píxels obtidas através das equações mostradas acima. As imagens foram obtidas através da função `ee.Image.clip()` do GEE e guardadas na drive da conta Google. Foram ainda convertidas do formato .tif para .jpg através da biblioteca OpenCV para puderem ser usadas pelos algoritmos de deep learning, como por exemplo, o YOLO. No final foi possível obter cerca de 80 imagens.

Além disto foi necessário obter as anotações. No caso do YOLOv5 é necessário obter um ficheiro no formato texto para cada imagem. Este ficheiro é organizado da seguinte forma:

- Id da classe do objeto
- X e Y que corresponde às coordenadas do centro da bounding box
- Width e height são a largura e altura da bounding box

As coordenadas em pixels de cada mamoa na imagem LRM foram convertidas para as coordenadas da imagem recortada, deste modo, as coordenadas do centro de cada bounding box foram adquiridas.

O tamanho ideal para uma bounding box necessita de ser estudado, pois neste caso o tamanho de cada sítio arqueológico pode variar e além disso não se sabe se o que está em redor de cada mamoa pertence ou não pertence. Desta forma, realizou-se um estudo com diferentes tamanhos para as bounding boxes. Para isso, foram gerados diferentes ficheiros de texto com anotações. Os tamanhos escolhidos foram: 15×15, 20×20, 22×22, 25×25, 30×30 e 35×35. Posteriormente realizou-se um estudo através de uma análise de resultados para obter o melhor tamanho que permite uma melhor identificação das mamoas.

Antes de obter os ficheiros de texto foi ainda necessário realizar a normalização dos valores, para isso, dividiu-se as coordenadas, largura e altura de cada bounding box pelo tamanho da imagem associada.

O próximo passo foi realizar a divisão do dataset com as imagens e anotações. Para isso, usou-se a biblioteca splitfolders para dividir 60% para treino, 20% para validação e 20% para teste. Desta forma, o dataset inicial para cada dimensão de bounding box é mostrado a partir da Tabela 4.1.

Dados	**Imagens**	**Nº de objetos**
Treino	48	75
Validação	16	31
Teste	16	30

Tabela 4.1: Divisão dos dados originais.

A Figura 4.3 é um exemplo de uma das imagens que compõem o dataset. Para verificar todo o trabalho descrito neste subcapítulo decidiu-se realizar uma função que permite ler as imagens e anotações, na qual o objetivo foi utilizar a biblioteca OpenCV para desenhar bounding boxes que representam o ground thruth do objeto arqueológico associado na imagem.

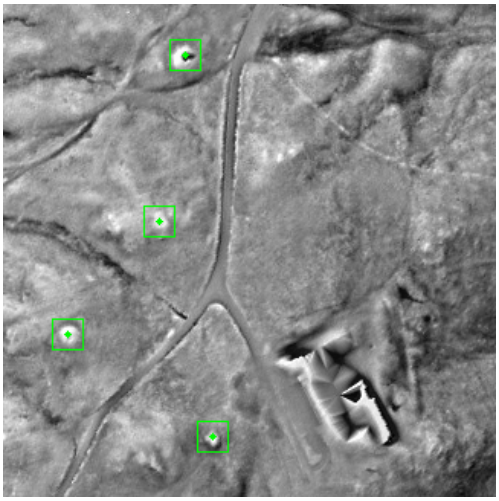

Figura 4.3: Exemplo de imagem do dataset criado.

Em suma, após todas as análises do dataset obtido terem sido realizadas com sucesso é possível aplicar diversos algoritmos de deep learning. Embora o dataset com as anotações esteja focado para o uso de YOLO, é possível converter estas anotações para outro tipo de formato, como por exemplo, o formato COCO. Este tipo de formato já foi anteriormente usado no capítulo anterior com o Mask R-CNN. Desta forma, escreveu-se um script em python que permite converter as coordenadas anteriormente obtidas do formato YOLO para o formato COCO. Para isso, foi usado um ficheiro no formato .json, que possui informações associadas à imagem e anotações dos objetos em estudo.

4.2 Aumento de dados

Um dos principais problemas que tem vindo a ser discutido é o facto de existirem poucos dados, algo que pode provocar maus resultados. Desta forma, para reduzir este risco decidiu-se realizar um aumento do dataset através da aplicação de várias técnicas. Para isso foi usado a biblioteca albumentations para aumentar o dataset de treino e validação. O dados de teste não sofrem aumento de dados, pois o objetivo é testar o modelo obtido com imagens reais. Existem diversas técnicas que permitem aumentar os dados. Contudo foram apenas usadas algumas delas, pois outras alteravam o tamanho da bounding box, tal como, a rotação, algo que neste caso não é favorável, visto que um dos objetivos é realizar um estudo com vários tamanhos fixos conhecidos à partida. As técnicas usadas foram as seguintes: transposição, horizontal e vertical flip, RGB shift, blur e colorJitter. O objetivo com este aumento de dados, além de ser obter um maior dataset é a tentativa de obter exemplos diferentes que permitam diminuir o overfitting e aumentar o desempenho do modelo. A biblioteca mencionada realiza a leitura das anotações no formato YOLO e obtém as novas anotações para as imagens provenientes do

aumento de dados. Por cada imagem do dataset original de treino e validação foram obtidas através de um ciclo for mais 7 imagens com as respetivas anotações. Deste modo, o dataset ficou organizado de acordo com a Tabela 4.2.

Dados	**Imagens**	**Nº de objetos**
Treino	384	600
Validação	128	248
Teste	16	30

Tabela 4.2: Divisão dos dados aumentados.

Cada uma das técnicas aqui mencionadas representa uma função da biblioteca albumentations e foi usado um parâmetro de entrada conhecido por 'p'. Este, indica a probabilidade da técnica ser aplicada à imagem.

Foi usada a transposição que realiza uma troca de linhas e colunas através de uma rotação de 90º à imagem e foi aplicado com uma probabilidade p de 50%.

Para alterar as imagens de forma horizontal e vertical realizou-se uma viragem que realiza um deslocamento dos pixels da imagem de entrada tendo como referência o eixo y e o eixo x, respetivamente. Basicamente é como realizar uma dobra na horizontal ou vertical na imagem e virar o objeto de uma dobra para a outra. O valor de probabilidade p usado para esta técnica foi 50%

O RGBShift também foi aplicado, uma vez que muda aleatoriamente os valores de pixels de cada banda da imagem de entrada. Para esta técnica foi definida uma probabilidade cerca de 80% e o intervalo limite para alterar os valores para cada banda foram (-25,25).

Foi também aplicado um filtro que permite simular ruído nas imagens, designado por Blur. Este é importante para obter imagens com ligeiras distorções ou desfocagens. Este método recebe como argumento de entrada o tamanho máximo do kernel igual a 3 para desfocar a imagem e a probabilidade p usada foi cerca de 50%.

O ColorJitter também foi aplicado, uma vez que altera o brilho, contraste e saturação da imagem de entrada. Esta técnica foi aplicada com uma probabilidade de 50% e valores de brilho, saturação, contraste e tonalidade com valores entre (0,0.2).

A biblioteca albumentations permitiu desta forma obter novos exemplos para treino e validação a partir dos dados existentes. O objetivo do aumento de dados é aumentar a qualidade e desempenho de um algoritmo deep learning [30] no sentido de diminuir a ocorrência de overfitting. A Figura 4.4 ilustra três exemplos de imagens do dataset de treino que resultaram do processo anteriormente descrito.

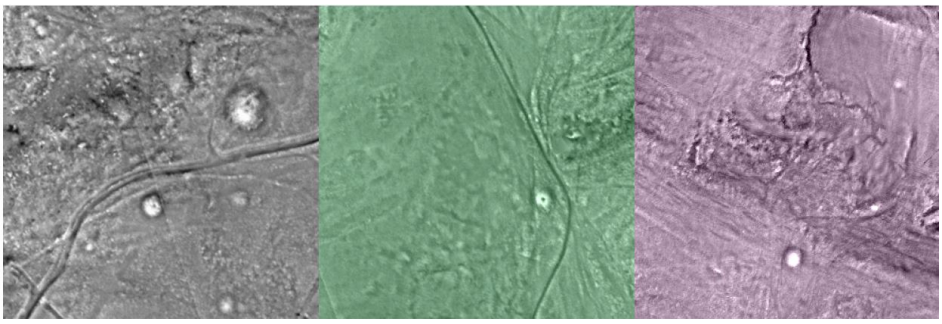

Figura 4.4: Exemplos de 3 imagens que resultaram do aumento de dados.

4.3 Modelos de treino

Neste sub-capítulo pretende-se falar acerca dos parâmetros do modelo usado para cada algoritmo, ou seja, YOLOv5, Mask R-CNN e CNN. Estes parâmetros são usados durante o treino sendo importante a sua escolha para aumentar o desempenho. Alguns deles foram escolhidos através de várias tentativas realizadas, como por exemplo, o batch size e o número de épocas. De notar, que o Google Colaboratory foi usado para realizar o treino através de YOLOv5 e Mask R-CNN, visto que possui os recursos computacionais necessários e assim permite usar um batch size maior. Aumentar o valor deste parâmetro não implica que irá haver melhores resultados, contudo permite realizar um treino mais rápido e usar melhor os recursos computacionais disponíveis. O Google Colaboratory permite o uso de uma GPU Tesla k80 ou GPU Tesla T4, em que, esta última possui um melhor desempenho. Para usar esta ferramenta basta possuir uma conta associada Google e pode ser usada livremente durante um período limitado a 12 horas.

4.3.1 YOLOv5

O YOLOv5 é um dos algoritmos aqui usado para detetar objetos. Desta forma é feita a explicação detalhada do seu uso.

Primeiramente, foi realizado o clone do repositório do GitHub [26] e instalados todos os requerimentos necessários. A versão python usada deve ser $\geq$ 3.7.0. A framework usada é o PyTorch, na qual, a sua versão é $\geq$ 1.7. Antes do treino começar é necessário selecionar um modelo pré-treinado, na qual, a sua seleção é importante para obter bons resultados [31]. Aqui existem vários modelos, tal como é possível verificar na Figura 4.5.

O uso de modelos pré-treinados é recomendado para este caso, pois existe uma quantidade de dados reduzida. O YOLOv5s foi o modelo escolhido visto ser considerado pequeno e rápido. Estes modelos foram previamente treinados no conjunto de dados COCO que conta com 80 categorias de objetos [25]. As camadas pré-treinadas deste modelo são todas carregadas, contudo neste caso não existem 80 classes, mas sim apenas a classe 'mamoa'. Deste modo, o modelo irá ser composto pelos weights pré-treinados, exceto a camada de saída que neste caso possui um formato diferente e irá ser inicializada com weights aleatórios [32].

Os parâmetros do modelo foram atribuídos da seguinte forma:

- Nº de épocas = 200

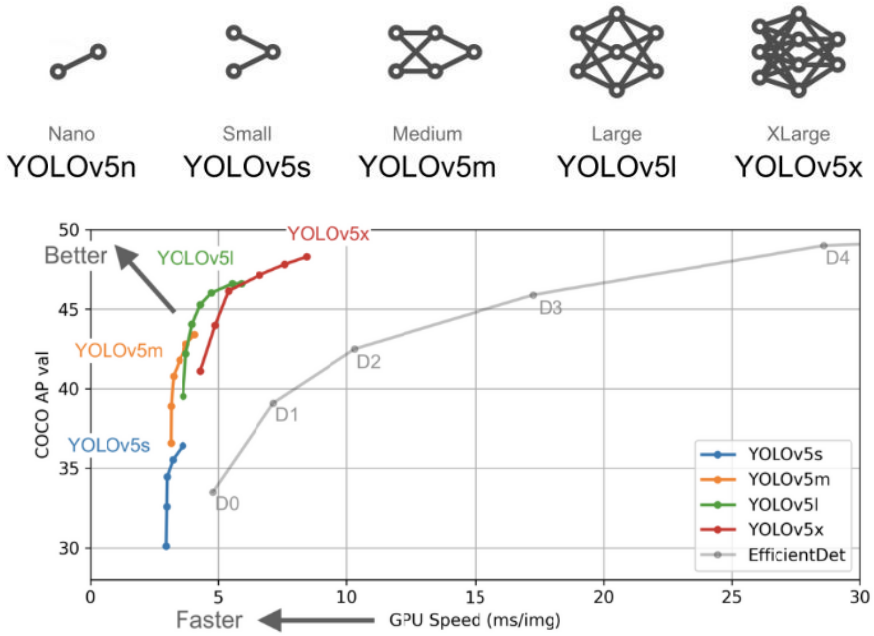

Figura 4.5: Modelos pré-treinados do YOLOv5 [26].

- Batch size = 16
- Patience = 50
- Nº de camadas = 270
- Nº de parâmetros = 7022326
- Learning rate = 0.01
- IoU threshold = 0.2

O número de épocas usado inicialmente foi um número superior, cerca de 500 épocas. Contudo, verificou-se a presença de overfitting após as primeiras épocas e decidiu-se reduzir este número para 200 épocas. O batch size usado foi 16, pois os recursos computacionais permitiam o uso deste valor. O learning rate foi previamente definido com o valor 0.01, no qual, observou-se que permite obter uma boa convergência. De notar, que a escolha correta deste parâmetro é essencial, pois um valor demasiado baixo provoca uma convergência mais demorada necessitando de mais iterações. Um valor demasiado elevado pode provocar uma não convergência. O parâmetro patience foi ajustado para 50, ou seja, no caso de não existirem melhorias após 50 épocas o treino finaliza. O tamanho usado para as imagens de entrada foi previamente definido no pré-processamento de dados, ou seja, $640 \times 640 pixels$ que é a dimensão recomendada. Os dados são lidos através de um ficheiro no formato .yaml que é organizado com os diretórios das pastas de treino, validação, número de classes e o nome da classe:

train: dataset/train
val: dataset/val
nc: 1
names: ['mamoa']

Para facilitar a escrita deste ficheiro e automatizar este processo, foi realizada uma função em python que recebe como argumentos de entrada os diretórios correspondentes ao dataset de treino e validação e o número e nome de classes.

4.3.2 Mask R-CNN

O Mask R-CNN é outro algoritmo aqui usado para detetar objetos. Relembra-se que este algoritmo permite realizar a deteção e segmentação de objetos na imagem. O backbone usado é o RESNET101 que usa 101 camadas. Primeiramente, começou-se por realizar o clone do repositório GitHub [23] e instalação de todos os requerimentos necessários. A versão das bibliotecas mais relevantes usadas foram as seguintes: python 3.7, tensorflow-gpu 1.13.1, keras 2.0.8, h5py 2.10.0, scikit-image 0.16.2. Foi ainda usado um modelo pré-treinado do conjunto de dados COCO, 'mask_rcnn_coco.h5'.

Os parâmetros do modelo foram atribuídos da seguinte forma:

- Nº de épocas = 100
- Batch size = 8
- Nº de camadas = 101
- Nº de parâmetros > 44M
- Learning rate = 0.001
- IoU threshold = 0.5

O processo de atribuição de valores aos parâmetros foi semelhante ao YOLOv5. Primeiramente, começou-se com um número superior de épocas, realizou-se o treino e observaram-se os resultados. Como se verificou a presença de overfitting a partir das primeiras épocas decidiu-se reduzir o número de épocas para 100. O batch size usado foi o valor 8 devido ser aconselhado pela documentação deste algoritmo. O learning rate usado inicialmente era 0.01, contudo as perdas de treino não convergiam, por isso, foi usado o valor inferior 0.001. O dataset usado é exatamente igual ao anteriormente usado pelo YOLOv5, ou seja, com um tamanho de imagens igual a $640 \times 640 pixels$.

4.3.3 CNN

Após a utilização dos algoritmos anteriores decidiu-se usar uma rede neuronal convolucional mais simples devido ao facto do dataset ser reduzido e esperar obter melhores resultados. Deste modo, ao contrário dos algoritmos anteriores foram adquiridos durante a fase do pré-processamento exemplos de mamoas e exemplos sem mamoas. Cada imagem conta com um tamanho em pixels de 40×40 que equivale a $20 \times 20m$ no terreno, ou seja, uma área equivalente a $400m^2$.

Este algoritmo CNN usa como framework o tensorflow e keras. A escolha destas ferramentas surgiu devido à sua frequente utilização na resolução deste tipo de problemas. As versões usadas foram as seguintes: tensorflow 2.9.0 e keras 2.8.0.

Antes da atribuição dos parâmetros do modelo e da realização do treino foi necessário realizar um pré-processamento do dataset, como por exemplo, leitura das imagens e conversão para um numpy array, atribuição de labels, baralhar os dados através da função `random.shuffle()` e normalização dos valores de pixel. Estes dados foram divididos em três partes, sendo elas,

treino, validação e teste através da função, `train_test_split()`, da seguinte forma: 70% treino, 15% validação e 15% teste.

O número de camadas convolucionais a usar foi alvo de estudo no sentido de obter os melhores resultados. Desta forma, o treino foi realizado várias vezes, à qual, obtiveram-se os respetivos valores de accuracy. Foram testadas até três camadas, em que, verificou-se o melhor resultado com a utilização de apenas uma camada convolucional. No próximo capítulo é possível observar os valores obtidos que confirmam este resultado.

O modelo de treino foi definido da seguinte forma:

- Nº de épocas = 300
- Patience = 100
- Nº de camadas = 3 (1 camada convolucional)
- Nº de parâmetros = 14529
- Learning rate = 0.001 (Optimizer Adam)

Em suma, além da camada convolucional existem ainda outras duas, sendo elas, a camada de entrada e de saída.

A imagem de entrada tem apenas uma banda e por isso possui $40 \times 40 = 1600$ entradas (features). Após a passagem por cada camada de convolução foi usada a função de ativação Rectified Linear Unit (ReLU). A utilização desta função permite introduzir não linearidade. A utilização de bacth normalization permite tornar a rede mais rápida e estável através da normalização das entradas das camadas. Após o processo de feature learning segue-se a etapa de classificação, em que, as features são conectadas e usou-se a função de ativação *sigmoid*, visto que este problema exige uma classificação binária. Deste modo, sempre que o resultado for < 0.5 a deteção é considerada uma mamoa e caso contrário conclui-se que não é.

O número de épocas usadas foi superior ao necessário, porém usou-se a função `EarlyStopping()` com um valor patience de 100, ou seja, se após 100 épocas não se verificar melhorias nos resultados o treino é finalizado. Esta verificação é feita através do valor de perdas, sendo que o melhor modelo obtido corresponde à menor perda de validação. Em termos de otimização foi usado o otimizador 'adam' que implementa um learning rate com o valor 0.001, sendo considerado um método computacionalmente eficiente.

4.4 K-Fold cross validation com YOLO

O K-fold cross validation é uma técnica muito comum usada no machine learning e conhecida por proporcionar bons resultados. Esta técnica pode ser considerada interessante e útil, uma vez que pode minimizar o problema da falta de dados através da utilização dos dados de treino para validação. Deste modo, o dataset deixa de ser dividido em 3 partes passando apenas a ser usado dados de treino e teste. Uma outra vantagem é o facto de realizar mais testes que torna o resultado final mais legítimo e obter métricas semelhantes significa que os dados são consistentes ou fáceis de aprender. Uma desvantagem deste processo pode ser a exigência temporal, pois necessita de mais tempo para realizar vários treinos e validar resultados.

Tal como foi mencionado o dataset é dividido em duas partes, treino e teste. Deste modo, o conjunto de dados de treino é dividido em vários subconjuntos, denominados por folds. O parâmetro k-fold significa o número de folds que o dataset de treino irá ser dividido e o número de validações a ser realizadas. De reparar, que cada subconjunto é usado no treino $k-1$ vezes.

O algoritmo YOLO não costuma ser usado com a técnica k-fold cross validation por isso necessitaria de algumas alterações de código [33]. Contudo, tendo em conta a teoria e o que foi explicado foi possível criar um processo iterativo que permitiu organizar o dataset de treino e dividi-lo em vários subconjuntos (folds). O treino YOLO é aplicado k vezes e usa $k-1$ subconjuntos de cada vez. Os dados de validação foram associados aos dados de treino. Obteve-se a seguinte divisão:

Dados	**Imagens**	**Nº de objetos**
Treino	480	768
Teste	20	40

Tabela 4.3: Divisão dos dados aumentados para usar k-fold cross validation com YOLOv5.

Os parâmetros usados pelo modelo foram os mesmos. Usou-se um valor k-fold igual a 6, no qual, divide o dataset de treino em 6 folds e cada um conta com 80 imagens. Para tornar este processo iterativo realizou-se uma função python, `splitTrainData()`, que recebe como argumento de entrada o número de folds, o diretório dos dados de treino e por fim o diretório onde são guardados os dados já divididos em folds. Esta função é aplicada antes da realização de qualquer treino. Através de um ciclo for que por sua vez é executado k-fold vezes é aplicada uma função previamente realizada denominada de `crossValidation()`. Esta função retorna o diretório que contém os dados usados no treino e na validação correspondentes ao número de fold que estiver a ser executado no momento. Através da escrita automatizada dos diretórios de treino e validação num ficheiro yaml é possível alcançar o objetivo pretendido realizando o treino da forma esperada.

Uma das alterações efetuadas no código do YOLOv5 foi a escrita de um ficheiro de texto com as métricas obtidas no final de cada treino. No final foi possível ler estes ficheiros e calcular a média dos valores obtidos. Através disto é possível avaliar a performance do conjunto de treinos realizados. Por cada fold obteve-se um modelo final, no qual, foram aplicados aos dados de teste para obter os resultados e posteriormente obter o valor médio destes. O YOLOv5 permite a utilização de model ensemble que foi também usado e irá ser abordado de seguida.

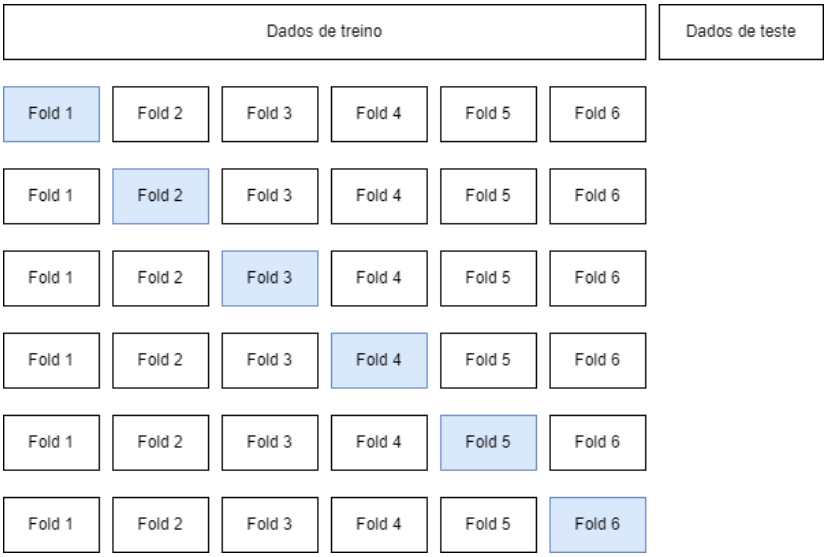

Figura 4.6: Esquema K-Fold cross validation com k-fold=6. A azul é representado o Fold usado na validação.

4.4.1 Model ensemble

O model ensemble foi usado para testar os modelos obtidos após a finalização do treino com k-fold cross validation. De notar, que estes modelos surgiram do treino com o mesmo tipo de dados, embora através do cross validation cada um acabe por ser treinado com exemplos diferentes. Deste modo, decidiu-se tirar proveito do uso destes diferentes modelos em conjunto para prever objetos.

É uma técnica que o YOLOv5 permite usar e conhecida por produzir melhorias no mAP e no recall durante o teste e inferência. Permite a utilização de vários modelos para prever um resultado final com dados nunca dantes vistos. Sendo que, a motivação para o seu uso é a diminuição do erro de generalização de previsão. Numa fase de inferência quando é necessário prever um resultado esta abordagem procura a resposta dada por cada um do conjunto de modelos usados, no qual, o que é dito pela maioria ganha.

Desta forma, embora o model ensemble use um conjunto com vários modelos acaba por funcionar como se fosse usado apenas um único modelo. Esta técnica pode ser usada com vários modelos treinados no mesmo conjunto de dados [34][35].

4.5 Inferência na Imagem LRM

Após todo o procedimento realizado até ao momento estar completamente finalizado decidiu-se aplicar os modelos obtidos à imagem LRM de todo o distrito de Viana do Castelo.

Primeiramente, começou-se por testar a ideia apenas numa pequena região, como por exemplo, a região dos Arcos de Valdevez que contém cerca de 85 mamoas já identificadas. O objetivo além da possibilidade de descobrir novos sítios arqueológicos é avaliar o desempenho dos modelos obtidos.

Para esta primeira abordagem foi realizado um script através do jupyter notebook com ligação ao GEE por forma a percorrer toda a área pretendida. Neste caso decidiu-se usar o

`predict()` do algoritmo CNN para prever a existência de objetos, uma vez que apresentou bons resultados anteriormente. Para isso, realizou-se a construção de uma janela que de forma iterativa efetua o clip da imagem através da função `clipToBoundsAndScale()` do GEE com um tamanho de 40×40 pixels, $(20 \times 20m)$. Após esta etapa é usada a função `sampleRectangle()` que por sua vez retorna os valores de pixels no interior da janela. De seguida procede-se à normalização destes valores que posteriormente entram no `model.predict()`. No caso do valor obtido ser < 0.5 o objeto presente na imagem é considerado uma mamoa. De notar, que isto é um processo iterativo e só termina após a janela percorrer a região pretendida. No próximo capítulo é possível observar os resultados obtidos para esta primeira abordagem.

Um dos problemas associados a este tipo de abordagem já observados em alguns artigos é o facto do objeto estar perto dos limites da imagem o que provoca um corte do mesmo. Desta forma, uma solução para resolver este problema baseia-se na utilização de uma janela deslizante com um determinado valor de step. Este valor representa o deslocamento em pixels que a janela deve percorrer horizontal e verticalmente. Para este trabalho decidiu-se realizar um estudo com valores de step diferentes, 20% e 50% no caso do CNN, para verificar qual permite obter os melhores resultados. Um step de 50% significa que a imagem irá ter metade da sobreposição. O step 20% indica um deslizamento de 20% da imagem, ou seja, existe uma sobreposição equivalente a 80% da imagem.

4.5.1 Extração da imagem do GEE

Um outro problema observado foi a questão temporal visto que isto é um processo de trabalho online através do GEE. Para esta pequena região com o uso de uma janela com um step de 20% o processo demorou cerca de 2 dias e 6 horas. Deste modo, optou-se por extrair a imagem LRM total de todo o distrito com a escala $0.5m/px$ e trabalhar no modo offline. Como é possível prever o GEE possui um limite de tamanho de imagem que pode extrair. Desta forma, tiveram-se em consideração os seguintes passos:

- Extração de imagens em blocos $(500 \times 500)px$.
- Concatenação das imagens na horizontal.
- Concatenação das imagens na vertical.

Esta etapa foi realizada através de um pc com 8 GB de RAM que não permitiu concatenar a imagem na sua totalidade devido a erros de alocação de memória. Para solução obteve-se a concatenação da imagem dividida em duas partes com uma ligeira sobreposição de 40 pixels de altura. Esta sobreposição permite evitar o corte de possíveis objetos. Para o posterior pós-processamento de dados esta parte não causa qualquer limitação, uma vez que estas duas imagens do ponto de vista de código são tratadas como uma única imagem.

4.5.2 Deteção com modelos treinados

Após a conclusão das etapas anteriores pretende-se aplicar os modelos treinados com os melhores resultados obtidos para as várias dimensões em estudo. Para tornar este trabalho mais autónomo resolveu-se realizar um programa em linguagem python, no qual, o utilizador pode escolher a forma como realiza a inferência através de YOLOv5, Mask R-CNN ou CNN.

Um outro parâmetro a ser atribuído é o valor step que permite usar o conceito da janela deslizante entre 0 e 100%. Além disso necessita de identificar o diretório do projeto. Este, contém as imagens LRM anteriormente concatenadas e os modelos obtidos do treino com os 3 algoritmos. A **figura 4.7** ilustra um exemplo do terminal da interface com o utilizador.

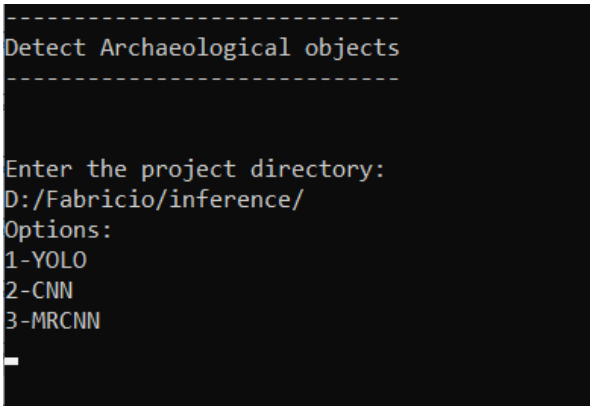

Figura 4.7: Interface com utilizador para realização da inferência.

O principio de funcionamento para realizar a inferência é semelhante à primeira abordagem usada com o CNN na região de Arcos de Valdevez. Tal como foi mencionado os objetivos desta etapa do trabalho passam pela avaliação do desempenho dos modelos obtidos e descobrir possíveis sítios arqueológicos. Deste modo, um dos procedimentos é inserir as 136 coordenadas geográficas das mamoas previamente identificadas que servem para calcular a quantidade de TP, TN, FP e FN. Embora o número de TN seja apenas relevante para o caso do CNN. Ainda que nesta situação o nome "false positives"seja usado, o mesmo pode gerar alguma confusão, uma vez que estes podem ser mamoas ainda não descobertas.

A imagem total conta com um tamanho de $140000 \times 126040 pixels$, desta forma as equações que permitem a conversão das coordenadas geográficas para as coordenadas em píxeis são:

- $px = 165334.516 \cdot LAT + 1472680.354$
- $py = -220838.328 \cdot LON + 9311284.277$

Através disto é possível converter as coordenadas das 136 mamoas já identificadas para coordenadas de píxeis. Através destas, é possível verificar se os objetos detetados na imagem foram anteriormente detetados, sendo contabilizados como true positives. Posteriormente, através destas equações é também possível obter a localização geográfica das mamoas detetadas pelos algoritmos. Para o caso do CNN esta verificação é realizada por meio de uma condição, no qual, verifica-se alguma das coordenadas das mamoas coincide com o interior da janela deslizante atual. Para o YOLOv5 e Mask R-CNN esta verificação é feita através do IoU(Intersection over union) ≥ 0.25 que usa o ground truth.

Algo importante a ter em conta é o carregamento dos modelos treinados. Tanto o CNN como o Mask R-CNN usam apenas o melhor modelo obtido para a melhor dimensão. Contudo, a deteção através do YOLOv5 usa o conceito de model ensemble com 6 modelos obtidos através do uso de 6 folds para a melhor dimensão de bounding box.

Os resultados são guardados num ficheiro csv para posteriormente ser feito o pós-processamento. Este ficheiro é organizado da seguinte forma: nome da imagem (nº de

linha e coluna da janela), a previsão (0=mamoa, 1=not_mamoa), conclusão (true ou false), as coordenadas em píxeis e as respetivas coordenadas geográficas.

Um dos problemas do uso da janela deslizante é a repetição dos objetos identificados, sendo contabilizados mais do que uma vez devido à sobreposição. Deste modo, é necessário realizar uma filtragem destes resultados. Para isso, no caso do CNN é realizada a leitura do ficheiro CSV, no qual, quando se encontra um caso de deteção positiva é realizada uma analise do resultado obtido relativamente aos blocos ao lado, em cima, em baixo e nos cantos superiores e inferiores, ou seja, todos os resultados que rodeiam esse caso positivo. Aqui, é realizada a contagem dos blocos que detetaram um caso positivo. Se esta der um número igual ou superior a 5 o bloco atual em análise é considerado uma mamoa, caso contrário não o é.

Um novo ficheiro CSV é escrito com os resultados atualizados. Para o caso do Mask R-CNN e YOLOv5 este problema é resolvido através da eliminação das coordenadas repetidas tendo em conta a dimensão de bounding box usada.

Em suma, todas coordenadas geográficas de possíveis mamoas são adicionadas ao GEE, por forma a adicionar uma coleção de multiponto como layer ao mapa para obter uma representação visual dos resultados obtidos.

CAPÍTULO 5

Resultados

Este capítulo tem como principal finalidade expor todos os resultados obtidos através do procedimento anteriormente descrito. Para proporcionar uma fácil análise destes resultados foram elaboradas tabelas e gráficos. É realizada uma explicação acerca do significado das várias métricas obtidas através do treino, validação e teste para os vários algoritmos aqui usados. Um outro fator a ter em conta são as condições, equipamento e tempo usado para obter estes resultados. Por fim são mostrados os resultados relativamente à inferência na imagem total do distrito de Viana do Castelo.

Tanto o YOLOv5 como Mask R-CNN usam o mAP (mean average precision). Para obter esta métrica é usado o recall e precision. Um elevado valor de precision significa a existência de poucos falsos positivos e vice-versa. Os objetos não detetados são contabilizados como false negative e são considerados background da imagem. Um elevado número de false negatives significa um baixo valor de recall. Devido à importância destas métricas é formado o gráfico precision vs recall. Este, permite selecionar o melhor limite que permite maximizar ambas as métricas. O mAP é a área correspondente à parte interior do gráfico precision vs recall.

Para obter vários valores de precision e recall são usados vários valores de threshold. Ambas as métricas são calculadas para cada valor de limite, no qual, posteriormente torna possível obter o gráfico anteriormente falado. Para o caso do YOLOv5 são apresentados gráficos que mostram a tendência de cada curva para os diferentes valores de threshold denominados por confidence. Estes valores limite são usados pelos algoritmos na previsão, no qual, quando um dado objeto é detetado com uma pontuação igual ou superior a um dado limite é atribuído uma certa classe. Por vezes a análise do gráfico precision vs recall pode tornar-se complexa e difícil de visualizar quais os melhores valores. Deste modo, é usado o gráfico F1 score com os respetivos valores de confidence associados. Aqui visualiza-se o valor de confidence para o qual existe o melhor equilíbrio entre precision e recall. Um valor elevado para F1 score pode significar um alto valor para precision e recall.

O mAP é a média de todos os APs(Average precision). Para cada classe é calculado o AP e no final é obtido a média. Para este trabalho apenas foi usada uma classe, por isso o mAP é

considerado o mesmo que AP e pode ser calculado de acordo com a Equação 5.1, sendo k o número de thresholds.

$$AP = \sum_{k=0}^{\text{k=n-1}} [Recalls(k) - Recalls(k+1)] \times Precisions(k) \tag{5.1}$$

Algo interessante acerca dos algoritmos de deteção de objetos é o facto de saber o quão bem foi detetado um objeto. Isto é possível devido ao cálculo do IoU (Intersection over Union), que é calculado de acordo com as a seguinte Equação 5.2.

$$IoU = \frac{\text{Área de } Interseção}{\text{Área de } União} \tag{5.2}$$

A área de interseção é a área correspondente à interseção entre a bounding box do ground truth e a box que surge da deteção. A área de união corresponde à área total de ambas as boxes. Este cálculo permite avaliar a classificação de um objeto como sendo true positive, false positive, true negative ou false negative. Após a deteção de um objeto obtém-se uma pontuação para o mesmo, deste modo, é usado o threshold de IoU para verificar se essa pontuação é superior a esse valor. Em caso positivo o objeto é considerado verdadeiro.

5.1 YOLOv5

Nesta sub-secção são apresentados os resultados obtidos relativamente ao algoritmo YOLOv5. Tal como é possível verificar foram obtidos diversos resultados, uma vez que este algoritmo foi usado com o dataset dividido em três e duas partes para o caso do uso de cross validation. Para este último estão presentes os resultados relativamente ao uso com e sem model ensemble.

As métricas apresentadas são F1-score, precision, recall e mAP. São ainda exibidas as contagens relativamente ao número de TP, FP e FN. Na Tabela 5.1 é possível verificar os resultados obtidos. Tal como foi referido anteriormente foram realizados diversos treinos para as diferentes dimensões de bounding box usadas. Deste modo, na tabela é possível verificar os resultados obtidos para cada uma dessas dimensões.

Algo a ter em conta relativamente ao algoritmo YOLOv5, é o facto dos valores de precision e recall serem obtidos através de 1000 valores de confidence entre 0 e 1. O F1 score foi obtido através dos valores de precision e recall para os mesmos valores de confidence. Durante o treino estes valores são calculados através do dataset de validação, ou seja, no final de cada época este dataset é usado para obter estas métricas. Os valores apresentados na tabela correspondem ao resultado obtido através do melhor modelo obtido do treino. A avaliação do modelo também foi feita com os dados de teste (exemplos de dados que o algoritmo não viu anteriormente). Neste caso é possível verificar na linha da tabela, "confidence", os valores obtidos através da analise do gráfico F1 vs confidence. Estes valores, tal como foi dito correspondem ao melhor resultado para F1 score, que por sua vez diz qual o melhor equilíbrio entre precision e recall. Os valores apresentados para TP, FP e FN foram obtidos através da matriz de confusão. Contudo, usando estes valores para obter o precision e recall verifica-se

Validação	15x15m	20x20m	22x22m	25x25m	30x30m	35x35m
F1	0.79	0.83	0.82	0.84	0.84	0.82
Precision	0.88	0.89	0.86	0.84	0.89	0.94
Recall	0.72	0.76	0.77	0.76	0.80	0.73
mAP@.5	0.79	0.82	0.82	0.84	0.84	0.82
TP	207	213	195	210	216	182
FP	23	19	13	9	15	4
FN	81	75	93	78	70	91
Teste						
Confidence	0.41	0.22	0.22	0.28	0.18	0.42
mAP@.5	0.71	0.74	0.77	0.8	0.79	0.71
Precision	0.72	0.69	0.73	0.76	0.742	0.90
Recall	0.70	0.73	0.80	0.82	0.82	0.63
F1	0.71	0.71	0.76	0.79	0.78	0.74
TP	24	21	23	22	22	17
FP	13	9	7	7	8	2
FN	6	9	7	5	6	10

Tabela 5.1: Desempenho do YOLOv5 para vários tamanhos de bounding box.

que não correspondem aos valores apresentados. Isto deve-se ao facto do YOLOv5 usar um valor confidence fixo 0.25 para gerar a matriz de confusão e as métricas calculadas usarem o melhor valor de confidence apresentado.

A Figura 5.1 apresenta dois gráficos correspondentes aos resultados de mAP@.5 e F1 presentes na Tabela 5.1 de acordo com as dimensões usadas. Estas 2 métricas foram as escolhidas para análise, uma vez que precision e recall por si só não têm utilidade. Nesta situação tem-se especial atenção para o gráfico dos dados de teste, pois pretende-se verificar o comportamento do modelo obtido relativamente a dados não usados anteriormente. Desta forma, pode-se verificar que existiu uma tendência crescente da average precision até $25 \times 25m$, que apresentou o melhor resultado. Para as dimensões superiores, 30×30 e $35 \times 35m$, o mAP e F1 diminuíram.

Na Tabela 5.2 são apresentados os resultados relativamente ao uso do YOLOv5 com k-fold cross validation. Tal como referido foi usado um k-fold igual a 6. A tabela é dividida em 3 partes, sendo que a primeira corresponde aos resultados de treino gerados através da média dos valores obtidos no final de cada treino, através da validação de cada fold. As outras duas partes da tabela correspondem aos valores obtidos através do uso dos dados de teste. Como foi dito estes valores foram obtidos de forma diferente. Os resultados de teste da segunda parte da tabela foram alcançados através do model ensemble.

A última parte da tabela tem os valores de teste adquiridos sem model ensemble. Estes foram obtidos através da utilização independente de cada modelo e no final realizou-se a média dos valores.

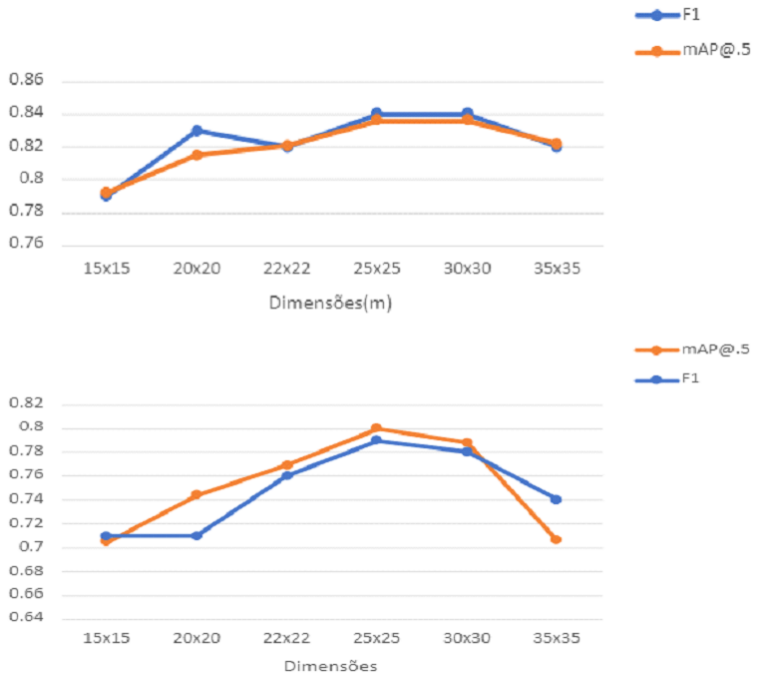

Figura 5.1: Gráficos mAP e F1 vs tamanhos de bounding box usados. Superior: resultados de validação, Inferior: resultados de teste

Uma vez que o uso de K-fold cross validation é uma técnica conhecida por apresentar resultados com maior nível de confiança será realizada uma análise mais detalhada acerca do treino realizado e como as métricas foram obtidas. Embora o conceito e análise sejam idênticos para o caso anterior com YOLOv5.

Na Figura 5.2 é possível observar os gráficos relativamente às perdas do treino e validação. Foram usados 6 folds, desta forma, existe um conjunto de gráficos para cada fold. Estes são relativos às perdas de box e objetividade. Através dos gráficos de perdas de validação é possível verificar a presença de overfitting, após ser alcançado um valor mínimo existe uma tendência crescente nos valores de perdas. Neste caso, as perdas de treino continuam a diminuir o que indica um bom funcionamento do modelo para os dados de treino, mas para novos dados deixa de ter sucesso. Deste modo, é possível afirmar que o melhor modelo obtido após o treino corresponde ao valor mínimo associado às perdas de validação. No geral, pode-se dizer que o overfitting é um problema, visto que logo após as primeiras épocas as perdas de validação aumentam. De notar, que o mesmo acontecia para o caso anterior, sem cross validation. Isto acontece devido ao principal problema falado durante esta dissertação, que é a falta de dados.

A Figura 5.3 ilustra os gráficos relativos às métricas obtidas com os dados de teste através dos modelos adquiridos anteriormente com cross validation e model ensemble. De notar, que por uma questão de simplicidade apenas estão presentes as curvas que correspondem à melhor dimensão vista na Tabela 5.2. Como foi introduzido nesta secção foram usados vários valores de confidence para obter os gráficos de precision e recall. Contudo, o objetivo é obter o melhor equilíbrio entre estas duas métricas, por isso, é mostrado o gráfico F1 score, em que, possui a

K-fold cv	15x15m	20x20m	22x22m	25x25m	30x30m	35x35m
F1	0.74	0.76	0.78	0.78	0.75	0.79
Precision	0.87	0.91	0.88	0.89	0.91	0.93
Recall	0.64	0.48	0.70	0.69	0.70	0.69
mAP@.5	0.70	0.73	0.77	0.75	0.76	0.78
TP	478	434	452	449	451	478
FP	92	32	41	40	30	55
FN	267	258	241	243	230	226
Teste*						
Confidence	0.83	0.53	0.76	0.74	0.44	0.56
mAP@.5	0.82	0.82	0.80	0.77	0.81	0.78
Precision	0.85	0.83	0.70	0.79	0.75	0.70
Recall	0.70	0.83	0.90	0.83	0.83	0.8
F1	0.77	0.82	0.79	0.80	0.79	0.74
TP	30	34	32	33	34	33
FP	6	13	16	16	13	16
FN	10	6	8	7	6	7
Teste**						
mAP@.5	0.75	0.78	0.77	0.78	0.79	0.77
Precision	0.78	0.85	0.79	0.82	0.82	0.79
Recall	0.74	0.72	0.77	0.76	0.78	0.73
F1	0.75	0.78	0.78	0.79	0.79	0.76
TP	30	27	27	29	29	28
FP	12	29	6	6	6	7
FN	10	11	13	9	10	10

Tabela 5.2: Resultados com cross validation, k-fold=6 com YOLOv5.
*Resultados de teste obtidos através de model ensemble.
**Resultados de teste obtidos sem model ensemble.

legenda com o valor de confidence correspondente ao melhor valor F1. Através deste é possível obter o valor de precision e recall presente na tabela. O mAP é representado pelo gráfico precision vs recall que tal como foi dito corresponde ao cálculo da área no interior da curva.

Na Figura 5.4 estão presentes dois gráficos relativamente aos dados mostrados na tabela anterior para as métricas mAP e F1 score. O primeiro gráfico mostra as curvas obtidas para os valores que resultaram do uso de cross validation durante o treino. Por forma a realizar uma analise com e sem o uso de model ensemble, o segundo gráfico presente na figura possui as curvas de mAP e F1 obtidas com os dados de teste. Deste modo, é possível verificar, tal como foi anteriormente previsto, que o uso de model ensemble proporciona melhores resultados tanto average precision, como F1-score. Além disso, comparando estes resultados com a Figura 5.1 verifica-se que foi possível alcançar melhores resultados com o uso desta técnica. O average precision obtido através da técnica cross validation obteve ligeiras melhorias relativamente aquilo que foi visto atrás. Esta melhoria deve-se provavelmente ao facto do número de exemplos usados para o treino ser superior, o que permite reduzir o overfitting e por sua vez aumentar os valores das métricas apresentadas.

A Figura 5.5 é um exemplo de sucesso na deteção de 4 mamoas. Por forma a verificar a

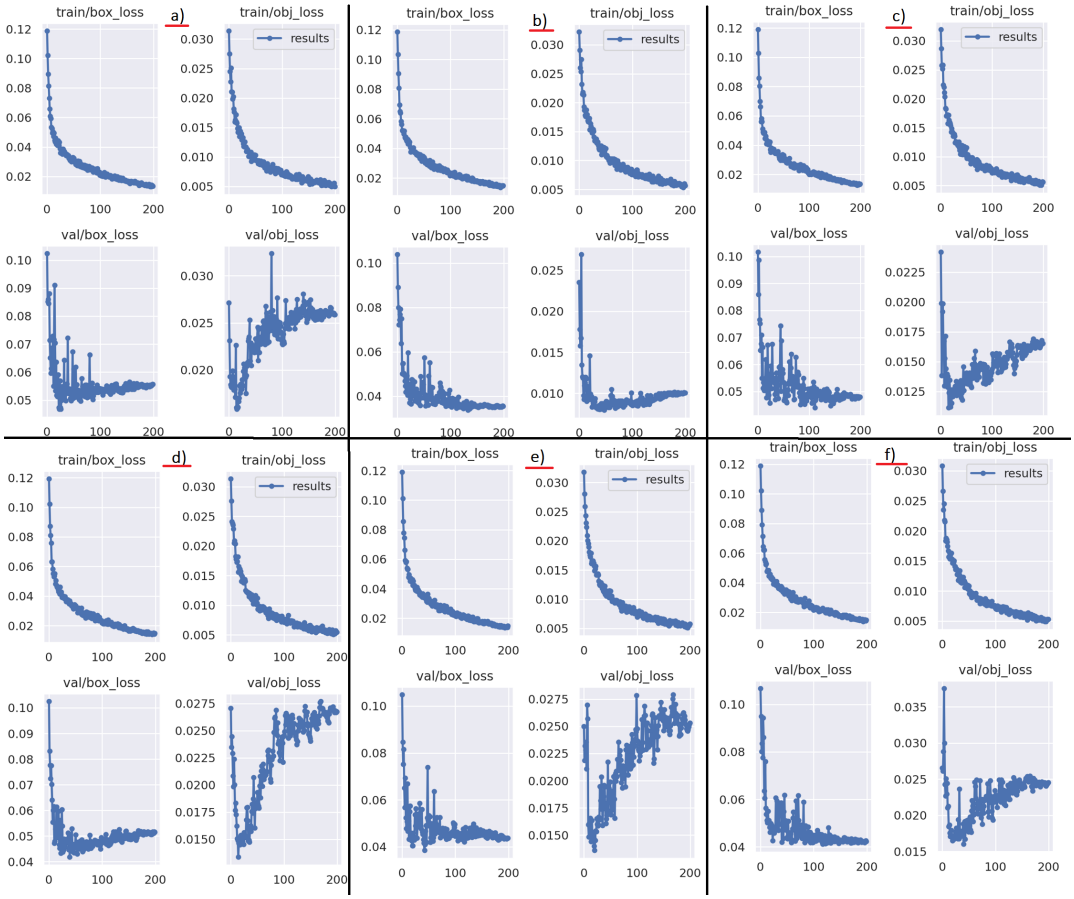

Figura 5.2: Gráficos relativos às perdas de treino e validação com YOLOv5 obtidos para cada fold. a)15x15m, b)20x20m, c)22x22m, d)25x25m, e)30x30m, f)35x35m

existência de possíveis TP, FP ou FN foi associada a bounding box a verde, que representa o ground truth.

Em suma, o treino foi realizado através do Google Colaboratoy e permitiu o uso de uma GPU Tesla T4 que fez diminuir os tempos de treino. Com o dataset dividido em três partes, sem cross validation, o treino demorou cerca de 1 hora e 30 minutos para cada dimensão, ou seja, um total de 6 horas. Com cross validation o treino foi mais demorado, uma vez que para cada dimensão demorou cerca de 6 horas. O google colab possui uma limitação de utilização de 12 horas seguidas, desta forma, o treino com os vários folds acabou por demorar um pouco mais do que o esperado.

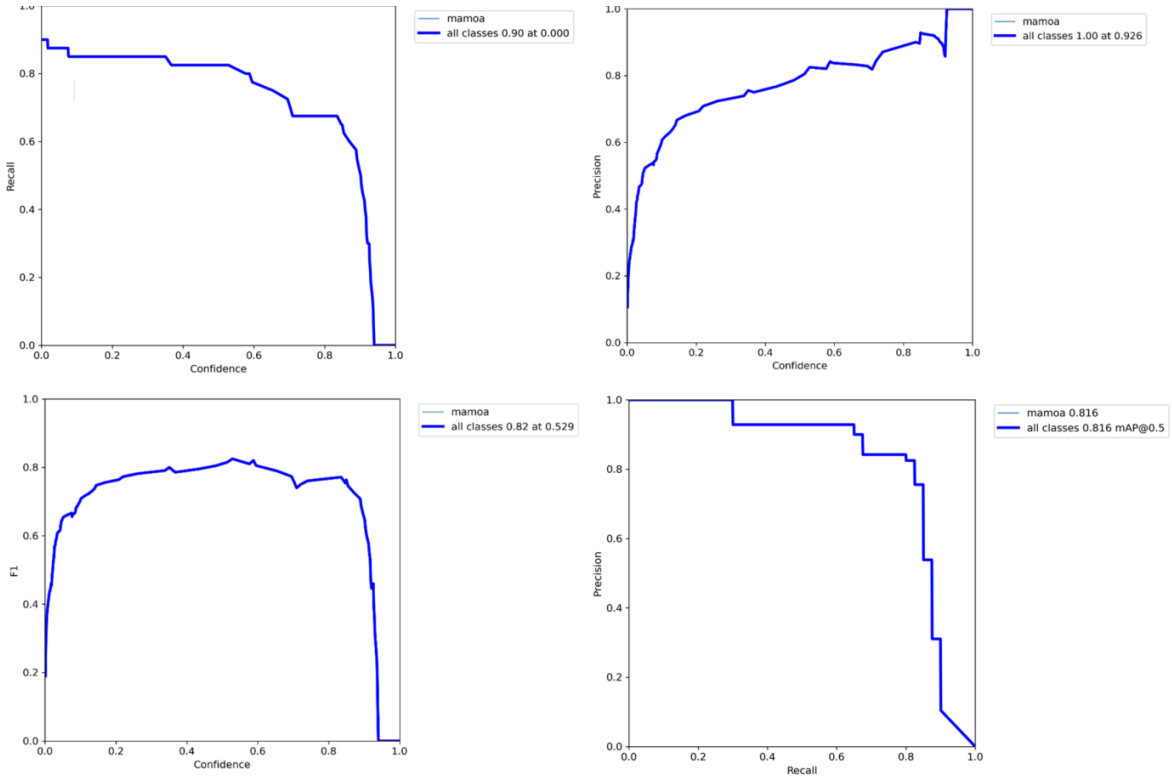

Figura 5.3: Gráficos relativos às métricas precision, recall, pontuação confidence e mAP@.5. Superior esquerdo: recall vs confidence, Superior direito: precision vs confidence, Inferior esquerdo: F1 score vs confidence, Inferior direito: Precision vs Recall.

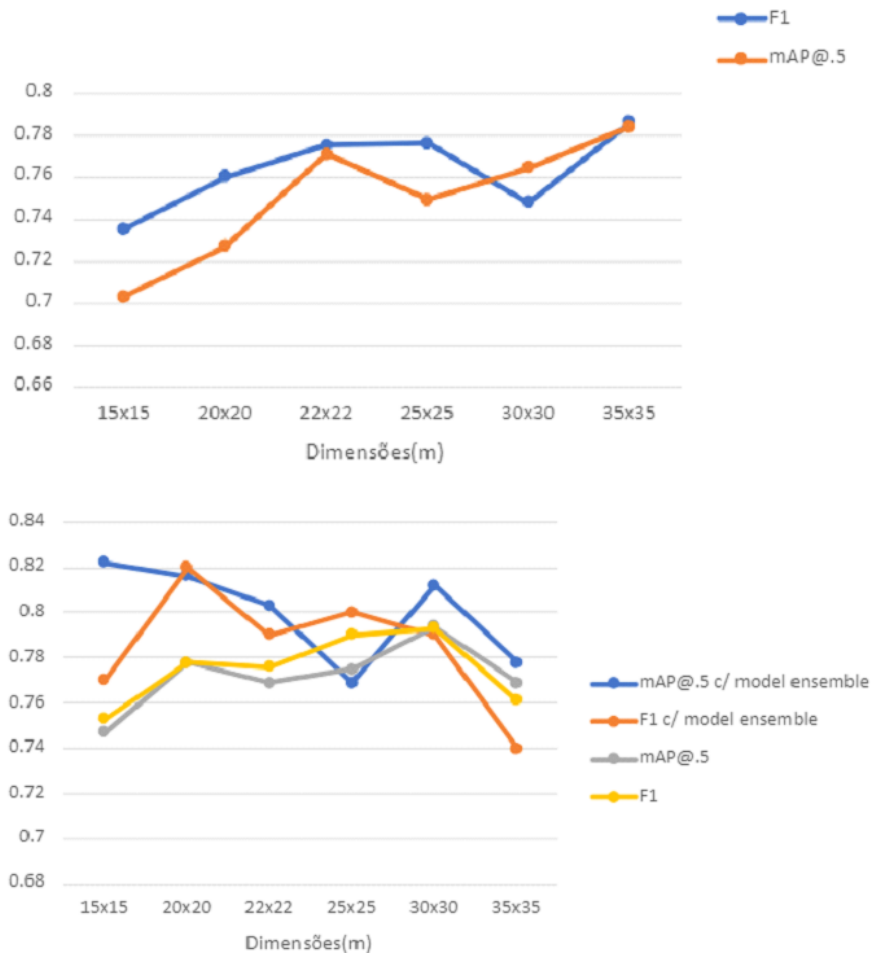

Figura 5.4: Gráficos mAP e F1 vs tamanhos de bounding box, Superior: resultados cross validation, Inferior: resultados de teste com e sem model ensemble.

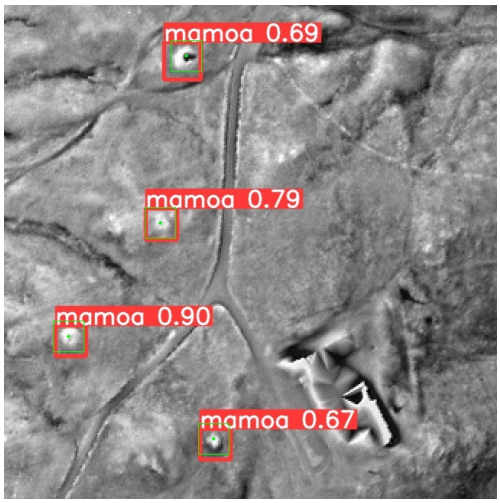

Figura 5.5: Exemplo de deteção de 4 mamoas com YOLOv5 para o tamanho 20x20 metros da bounding box. Bounding box verde representa o ground truth. Bounding box vermelha representa a mamoa detetada.

5.2 Mask-RCNN

Esta sub-secção expõe os resultados de deteção obtidos através da utilização do algoritmo Mask R-CNN. Tal como foi feito anteriormente é realizada uma análise através de uma tabela, gráficos e as respetivas perdas de treino e validação. Um dos objetivos da utilização deste algoritmo é a comparação com outros para auxiliar na validação dos resultados obtidos.

Como é possível verificar na Tabela 5.3 são usadas as mesmas métricas que o YOLOv5, tais como: F1-score, precision, recall e mAP@.5. Esta tabela possui duas partes, sendo a primeira com os valores obtidos através da validação do modelo e a segunda com os valores adquiridos com os dados de teste.

Na Figura 5.6 estão presentes os gráficos com F1-score e mAP@.5 relativamente ao resultado de validação e de teste. Através do gráfico de validação é possível verificar uma oscilação dos valores obtidos para as várias dimensões. O gráfico adquirido através dos dados de teste mostra que foram obtidos os melhores resultados para as dimensões 20×20 e $35 \times 35m$ com um average precision igual a 0.89.

A Figura 5.7 corresponde às perdas de treino e validação relativas à dimensão 20×20. É possível observar a presença de overfitting após 20 épocas. Desta forma, o modelo final obtido corresponde a esta época. O tempo de treino necessário para cada dimensão foi cerca de 2 horas, ou seja, um total aproximado de 12 horas através da GPU Tesla T4 do Google Colab.

Validação	15x15m	20x20m	22x22m	25x25m	30x30m	35x35m
F1	0.75	0.75	0.79	0.87	0.77	0.79
Precision	0.81	0.76	0.87	0.96	0.82	0.84
Recall	0.70	0.75	0.73	0.79	0.72	0.74
mAP@.5	0.79	0.85	0.80	0.76	0.81	0.78
Teste						
mAP@.5	0.84	0.89	0.83	0.71	0.84	0.89
Precision	0.66	0.66	0.78	0.84	0.75	0.82
Recall	0.73	0.84	0.79	0.74	0.78	0.80
F1	0.69	0.74	0.78	0.79	0.76	0.81

Tabela 5.3: Resultados obtidos com o algoritmo mask R-CNN.

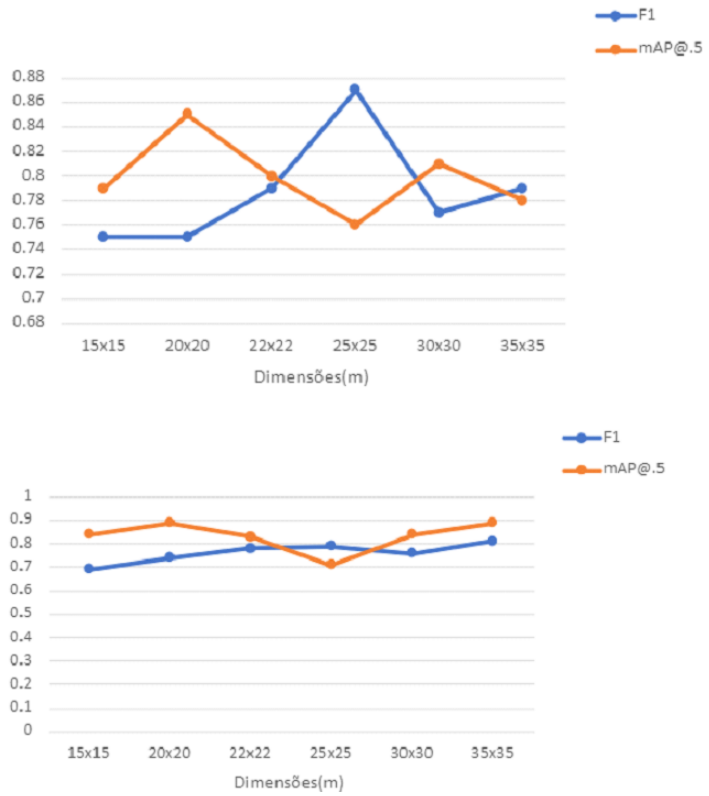

Figura 5.6: Gráficos mAP e F1 vs tamanhos de bounding box, superior: resultados validação, inferior: resultados de teste.

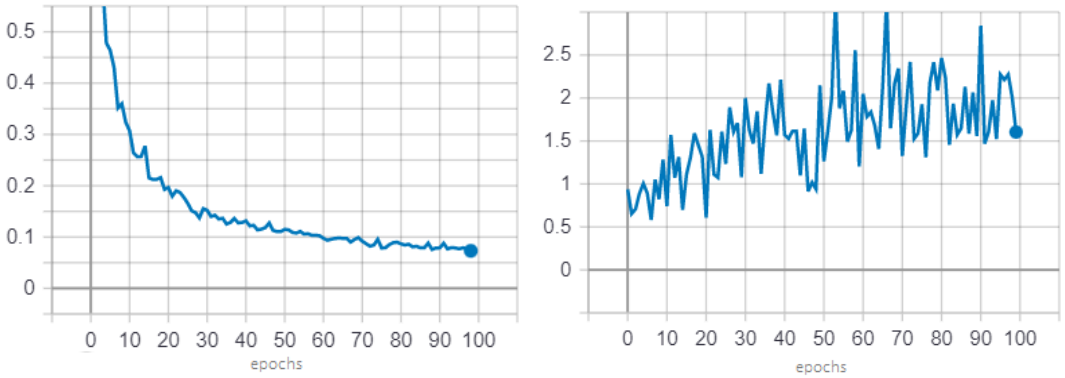

Figura 5.7: Gráficos das perdas resultantes do algoritmo Mask R-CNN vs nº de épocas, Esquerda: perdas de treino, Direita: perdas de validação.

5.3 CNN

Esta sub-secção apresenta os resultados obtidos com o algoritmo CNN personalizado. Aqui são mostrados os valores de accuracy resultantes do treino, validação e teste. São ainda apresentadas as contagens dos TP, FP, TN e FN. De notar que neste caso não foi realizado um estudo com as várias dimensões, mas sim apenas com a dimensão $20 \times 20m$, uma vez que obtiveram-se bons resultados para esta dimensão nos algoritmos anteriormente usados. Como foi referido, para este algoritmo foi realizado um estudo acerca do número de camadas convolucionais para o qual obtém-se os melhores resultados.

A Tabela 5.4 apresenta esse estudo. Começou-se apenas com uma camada e foi-se aumentando o número até verificar que não existem melhorias. Neste caso, o número máximo de camadas usadas foi 3, no qual, verificou-se o melhor resultado de accuracy, 0.95, para os dados de teste com a utilização de apenas uma camada. Algo a ter em conta relativamente aos dados usados é o facto de não ter sido feito qualquer aumento de dados, pois uma vez que estes resultados são considerados bons não exigiram a necessidade para tal.

nº camadas convolucionais	1	2	3
Train Acc	1.00	1.00	1.00
Val acc	0.90	0.93	0.93
TP	17	17	18
FP	1	0	1
TN	20	21	20
FN	3	3	2
Test Acc	0.95	0.93	0.93
TP	19	19	19
FP	0	1	1
TN	20	19	19
FN	2	2	2

Tabela 5.4: Desempenho obtido com o algoritmo CNN.

A Figura 5.8 mostra a presença de overfitting após 60 épocas, pois as perdas registadas no treino tendem a convergir para um valor mínimo, enquanto que as perdas de validação tendem a subir, ou seja, a partir desse ponto o modelo deixa de produzir bons resultados. Isto verifica-se com os valores apresentados na tabela, em que, relativamente ao treino existe uma accuracy de 100%, mas com os dados de validação e teste a accuracy desce ligeiramente.

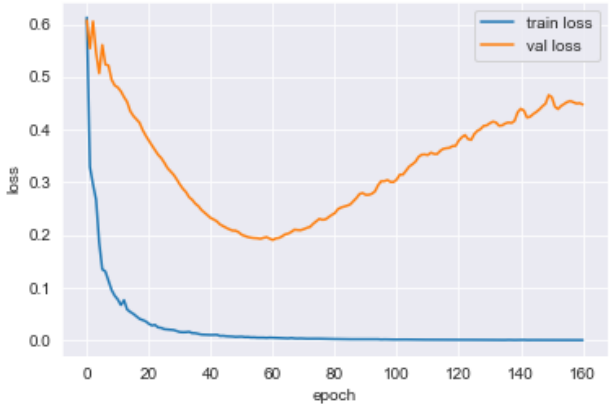

Figura 5.8: Gráfico com as perdas de treino e validação resultantes do algoritmo CNN para 1 camada convolucional.

Na Figura 5.9 é apresentado a accuracy relativamente ao treino e validação. Após a análise destes valores obtidos é possível afirmar que os resultados obtidos através deste algoritmo destacam-se como sendo os melhores. Isto deve-se ao facto de ser mais simples do que os outros, pois possui um menor número de parâmetros, algo também visível através do número de

camadas usadas. Desta forma, devido à baixa quantidade de dados a utilização de algoritmos mais simples pode ajudar na melhoria de resultados.

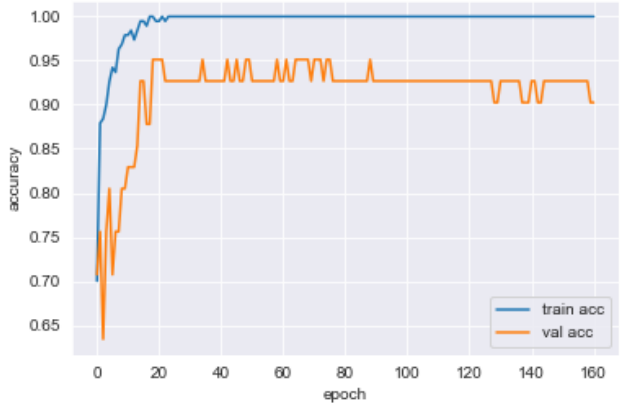

Figura 5.9: Accuracy adquirida no treino e validação com o CNN para 1 camada convolucional.

5.4 Inferência com a Imagem total

Um dos objetivos planeados para este trabalho é a utilização dos modelos adquiridos nos diferentes algoritmos usados, por forma a realizar a inferência em toda a imagem LRM do distrito de Portugal em estudo. Desta forma, esta sub-secção apresenta os resultados adquiridos após a finalização deste processo. Tal como foi referido no capítulo anterior as mamoas previamente identificadas foram usadas para obter as contagens de TP, FP, TN e FN. Estas serviram para calcular as métricas, como por exemplo, precision, recall, F1-score e accuracy. Contudo, para este caso o número de falsos positivos pode ser divido em duas partes, tais como: os que realmente não são mamoas e os que podem ser possíveis mamoas ainda não identificadas.

Algo importante é o facto das imagens analisadas serem vistas como um dataset. Nesse caso, é possível afirmar que este dataset não é balanceado, pois como é possível observar existe um maior número de exemplos negativos do que positivos. Desta forma, para o caso do CNN é necessário recorrer à métrica F1-score e não à accuracy para analisar os resultados.

A Tabela 5.5 corresponde aos resultados de uma primeira abordagem à inferência com CNN. Esta, foi realizada apenas numa pequena região conhecida por Arcos de Valdevez que por sua vez foi escolhida por possuir 85 mamoas já identificadas. Os steps usados para a janela deslizante encontram-se na tabela, sendo 50 e 20%. Tal como é possível confirmar os resultados relativamente à accuracy são enganadores devido à questão do não balanceamento dos dados. De notar, que o número de imagens analisadas é enorme tendo havido um elevado número de true negatives, mas também de false positives. Como é possível verificar nestes resultados preliminares não foi realizada qualquer tipo de filtragem dos valores repetidos consequentes da sobreposição da janela deslizante. A métrica mais importante nesta tabela é o F1 score, no qual, conclui-se que os resultados obtidos estão longe das expectativas sendo

considerados maus. Contudo, acerca do step usado para a janela deslizante verifica-se que os resultados tiveram uma ligeira melhoria com um step de 20%, uma vez que este método permite obter os objetos centrados na imagem e evitar o corte dos mesmos.

Step	50%	20%
Detetado	6761	42419
TP	65	430
FP	6696	41989
TN	98115	611837
FN	268	1662
Accuracy	93.37%	93.34%
Precision	0.96%	1.01%
Recall	19.51%	20.55%
F1	1.83%	1.93%

Tabela 5.5: Resultados através da inferência de um pedaço da imagem total do distrito de Viana do Castelo.

Os modelos obtidos através YOLOv5 e cross validation foram usados para realizar inferência na imagem total através do uso de model ensemble. Para este caso decidiu-se usar um step de 50 e 100%. Um step de 100% neste caso significa que não existe sobreposição da janela deslizante. Além disso, os modelos usados são relativos à dimensão $20 \times 20m$, $40 \times 40px$ que comparado com a imagem $640 \times 640px$ é considerado um tamanho pequeno. Foi também usado o step 50% que significa um deslizamento da janela de 320 píxeis que corresponde a metade da imagem, algo que poderia ser menor, pois as imagens desta forma possuem uma grande sobreposição. Poderia ser usado uma sobreposição de 40 píxeis que corresponde ao tamanho usado para a bounding box da mamoa.

Esta inferência demorou cerca de 2 horas e 30 minutos com um step de 100% e cerca de 10 horas com um step de 50% através de um pc com 8GB de RAM e uma GPU de NVIDIA 940MX.

A figura 5.10 mostra o resultado obtido através dos pontos de coordenadas geográficas adicionados como uma layer no mapa do GEE. Como é possível verificar existe uma quantidade enorme de pontos, ou seja, falsos positivos. Os modelos confundem o objeto arqueológico com outras estruturas com formato idêntico, como por exemplo rotundas, tal como é possível observar na Figura 5.11.

Algo interessante acerca desta inferência, é o facto de existir um maior número de falsos positivos relativamente a locais onde existem rios provavelmente devido à existência de estruturas semelhantes, como por exemplo, rochedos. A Tabela 5.6 apresenta os resultados relativamente ao número de mamoas detetadas, bem como, o número de TP, FP e TN.

A Tabela 5.7 apresenta o resultado obtido através do modelo Mask R-CNN após a realização da inferência com o uso de um step 100% e 50%. É possível verificar que com este modelo obtiveram-se piores resultados, pois existe um maior número de falsos positivos relativamente ao uso de YOLOv5. Deste modo, confirma-se que a junção de YOLOv5 K-fold cross validation e model ensemble permitem obter melhores resultados.

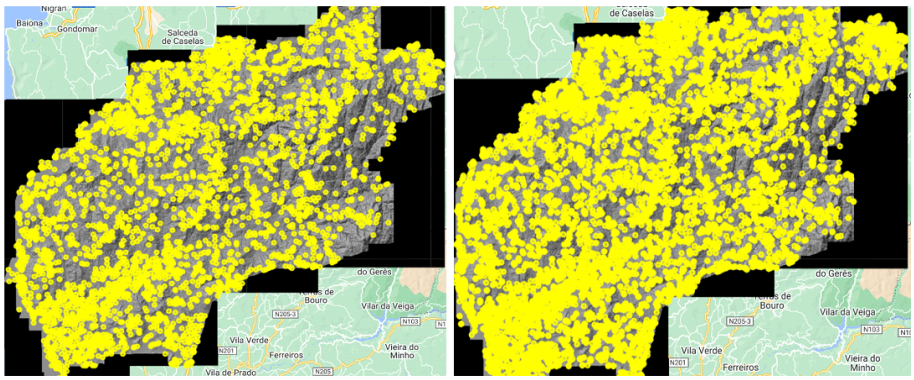

Figura 5.10: Imagem do mapa com layer de mamoas identificadas usando YOLOv5. Esquerda: step de 100% e Direita: step de 50%.

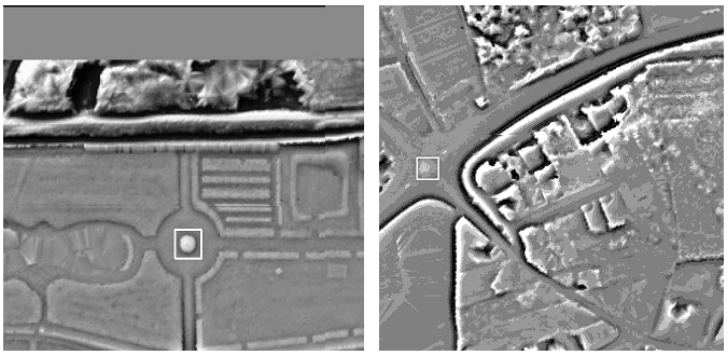

Figura 5.11: Exemplos de falsos positivos obtidos após inferência com YOLOv5.

Step	100%	50%
Detetado	3837	15280
TP	102	424
FP	3735	14856
FN	34	122

Tabela 5.6: Resultados através da inferência da imagem total do distrito de Viana do Castelo com YOLOv5.

Além disso permitiu realizar uma inferência mais rápida, uma vez que o Mask R-CNN demorou 11 horas e 50 minutos com um step de 100% e cerca de 2 dias com um step de 50%. Nesta inferência foi usado um pc diferente com 8GB e uma GPU AMD Radeon(TM) Vega 8.

Step	100%	50%
Detetado	33740	225242
TP	93	353
FP	33647	158177
FN	43	107

Tabela 5.7: Resultados através da inferência da imagem total do distrito de Viana do Castelo com Mask R-CNN.

CAPÍTULO 6

Conclusões

Após todo o trabalho realizado e os resultados adquiridos é possível tirar conclusões acerca dos algoritmos usados tendo em conta as várias dimensões estudadas. Além disso, pode-se falar acerca do que pode ser melhorado no futuro através da introdução de novos dados e diferentes técnicas.

Uma das razões, pela qual foi realizado o estudo acerca da melhor dimensão de bounding box é o facto do terreno envolvente poder ter impacto na deteção. Para isso foi usado mais do que um algoritmo por forma a comparar os resultados obtidos por cada um e obter a melhor dimensão. A Figura 6.1 apresenta um gráfico com os vários average precisions obtidos em cada algoritmo para cada dimensão. Através de uma breve análise é possível concluir que as dimensões $20 \times 20m$ e $30 \times 30m$ permitem obter bons resultados.

Todo o procedimento usado permitiu obter estes resultados com sucesso, embora exista espaço para a exploração de diferentes técnicas por forma a aumentar a eficácia. O principal problema está associado à falta de dados, algo comum nestes trabalhos. Um exemplo disso é o capitulo 3 da primeira abordagem a este trabalho que obteve maus resultados devido ao mesmo problema. Como se verificou, isto gera problemas relacionados com overfitting.

Algo a ter em conta é o facto de por vezes os resultados obtidos com os dados de teste serem ligeiramente melhores do que com os dados de validação. Algumas causas prováveis pode ser o facto de existirem poucos dados, ou estar relacionado com o aumento de dados realizado. Isto, porque o dataset continha poucas imagens reais e o mesmo sofreu um aumento de dados bastante grande. Desta forma, o dataset de validação possuía imagens alteradas, por exemplo, desfocadas, o que pode permitir ao modelo gerar melhores resultados quando está perante imagens reais, inalteradas.

Quando realizada a inferência na imagem de todo o distrito verificam-se maus resultados, pois existe confusão com diversos objetos idênticos a mamoas. Algo que poderia ser feito para ajudar a eliminar este problema seria a introdução de uma nova classe que representaria exemplos de falsos positivos. Desta forma, o algoritmo perante a classificação poderia conseguir distinguir de forma mais eficaz os objetos. Além disso, poderia ser aplicado um raster binário

a toda a imagem para eliminar regiões sem qualquer interesse, como por exemplo, regiões urbanas, permanecendo apenas zonas rurais que são menos alteradas ao longo do tempo devido à ação humana. Deste modo, permitiria ao algoritmo analisar apenas regiões com maior probabilidade de deteção de objetos.

A técnica de visualização usada, local relief model, permite na grande maioria do casos a visualização destes objetos. Contudo, seria interessante usar outro tipo de técnica de visualização, como por exemplo, o MSRM para obter resultados e compará-los com os obtidos neste trabalho.

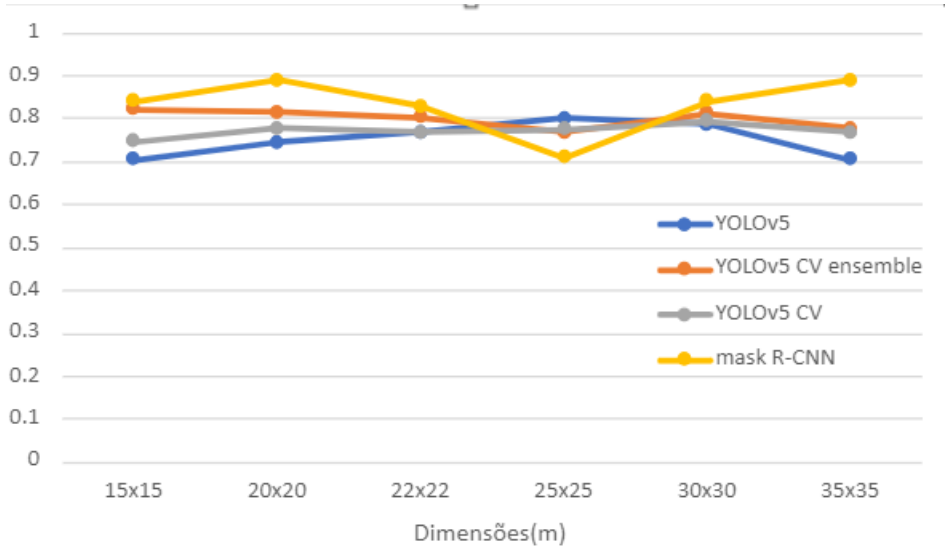

Figura 6.1: Gráfico para comparação do mAP obtido com os algoritmos, YOLOv5 e Mask R-CNN.

Referências

[1] A. H.Orengo, «A brave new world for archaeological survey: Automated machine learning-based potsherd detection using hight-resolution drone imagery,» *ELSEVIER*, vol. 112, 2019. DOI: https://doi.org/10.1016/j.jas.2019.105013.

[2] D.Davis, «Theoretical Repositioning of Automated Remote Sensing Archaeology: Shifting from Features to Ephemeral Landscapes,» *journal of computer applications in archaeology*, vol. 4, pp. 44–109, 2021. DOI: http://doi.org/10.5334/jcaa.72.

[3] L. Luo, X. Z. R. Lasaponara, G. W. N. Masini et al., «Airborne and spaceborne remote sensing for archaeological and cultural heritage applications: A review of the century (1907–2017),» *Remote Sensing of Environment*, vol. 232, 2019. DOI: https://doi.org/10.1016/j.rse.2019.111280.

[4] W.-v. d. V. M.Olivier, «Implementing State-of-the-Art Deep Learning Approaches for Archaeological Object Detection in Remotely-Sensed Data: The Results of Cross-Domain Collaboration,» *Journal of Computer Applications in Archaeology*, vol. 4, n.º 1, pp. 274–289, 2019. DOI: http://doi.org/10.5334/jcaa.78.

[5] URL: https://gisgeography.com/multispectral-vs-hyperspectral-imagery-explained/.

[6] URL: https://www.usgs.gov/landsat-missions/landsat-8.

[7] L. S. João Fonte, *"Módulo 1. Generación de contextos y escenarios digitales a partir de LiDAR aéreo"*, 2021.

[8] S. E. B. Štular E. Lozić, «Airborne LiDAR-Derived Digital Elevation Model for Archaeology,» *Remote Sensing of Environment*, vol. 13, n.º 9, p. 1855, 2021. DOI: https://doi.org/10.3390/rs13091855.

[9] I. Berganzo-Besga, F. L. H. Orengo, J. F. M. Carrero-Pazos e 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 Sens.*, vol. 13, n.º 20, p. 4181, 2021. DOI: https://doi.org/10.3390/rs13204181.

[10] URL: https://iaps.zrc-sazu.si/en/rtv#v.

[11] URL: https://grass.osgeo.org/grass78/manuals/addons/r.local.relief.html.

[12] R. Hesse., «LiDAR-derived Local Relief Models – a new tool for archaeological prospection,» *archaeological prospection*, vol. 17, pp. 67–72, 2010. DOI: https://doi.org/10.1002/arp.374.

[13] A. C. A. Hector, «Multi-scale relief model (MSRM): a new algorithm for the visualization of subtle topographic change of variable size in digital elevation models,» *Earth Surf. Process.*, vol. 245, pp. 1361–1369, 2018. DOI: https://doi.org/10.1002/esp.4317.

[14] T. L. A. Guyot L. Hubert-Moy, «Detecting Neolithic Burial Mounds from LiDAR-Derived Elevation Data Using a Multi-Scale Approach and Machine Learning Techniques,» *Remote Sensing*, vol. 10, n.º 2, p. 225, 2018. DOI: https://doi.org/10.3390/rs10020225.

[15] H. R. J. Lindsay J. Cockburn, «An Integral Image Approach to Performing Multi-Scale Topographic Position Analysis,» *ELSEVIER*, vol. 245, pp. 51–61, 2015. DOI: https://doi.org/10.1016/j.geomorph.2015.05.025.

[16] A.-M. H.A. Orengo, «A brave new world for archaeological survey: Automated machine learning-based potsherd detection using hight-resolution drone imagery,» *ELSEVIER*, vol. 112, 2019. DOI: https://doi.org/10.1016/j.jas.2019.105013.

[17] URL: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[18] K. L. W. Verschoof van der Vaart, «Learning to Look at LiDAR: The Use of R-CNN in the Automated Detection of Archaeological Objects in LiDAR Data from the Netherlands.,» *Journal of Computer Applications in Archaeology*, vol. 2, n.º 1, pp. 31–40, 2019. DOI: http://doi.org/10.5334/jcaa.32.

[19] URL: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/.

[20] W. V. v. d. V. M. Oliver, «Implementing State-of-the-Art Deep Learning Approaches for Archaeological Object Detection in Remotely-Sensed Data: The Results of Cross Domain Collaboration.,» *Journal of Computer Applications in Archaeology*, vol. 4, n.º 1, pp. 274–289, 2021. DOI: https://doi.org/10.5334/jcaa.78.

[21] J.Redmon, R. S.Divvala e A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» *ELSEVIER*, 2015. DOI: http://doi.org/10.48550/arXiv.1506.02640.

[22] M. Altaweel, Z. L. A. Khelifi, T. B. A. Squitieri e M. Ghazal, «Automated Archaeological Feature Detection Using Deep Learning on Optical UAV Imagery: Preliminary Results.,» *Remote Sens.*, vol. 14, n.º 3, p. 553, 2022. DOI: https://doi.org/10.3390/rs14030553.

[23] URL: https://github.com/matterport/Mask_RCNN.

[24] URL: https://pixellib.readthedocs.io/en/latest/.

[25] URL: https://cocodataset.org/#home.

[26] URL: https://github.com/ultralytics/yolov5.

[27] URL: http://www.cim-altominho.pt/.

[28] URL: https://www.google.com/intl/pt_in/earth/education/tools/google-earth-engine/.

[29] URL: https://colab.research.google.com/github/google/earthengine-api/blob/master/python/examples/ipynb/ee-api-colab-setup.ipynb.

[30] URL: https://albumentations.ai/docs/introduction/image_augmentation/.

[31] URL: https://github.com/ultralytics/yolov5/wiki/Tips-for-Best-Training-Results.

[32] URL: https://github.com/ultralytics/yolov5/issues/36.

[33] URL: https://github.com/ultralytics/yolov5/issues/1289#issuecomment-751282209.

[34] URL: https://github.com/ultralytics/yolov5/issues/318#issuecomment-1066623185.

[35] URL: https://docs.ultralytics.com/tutorials/model-ensembling/.

Anexos

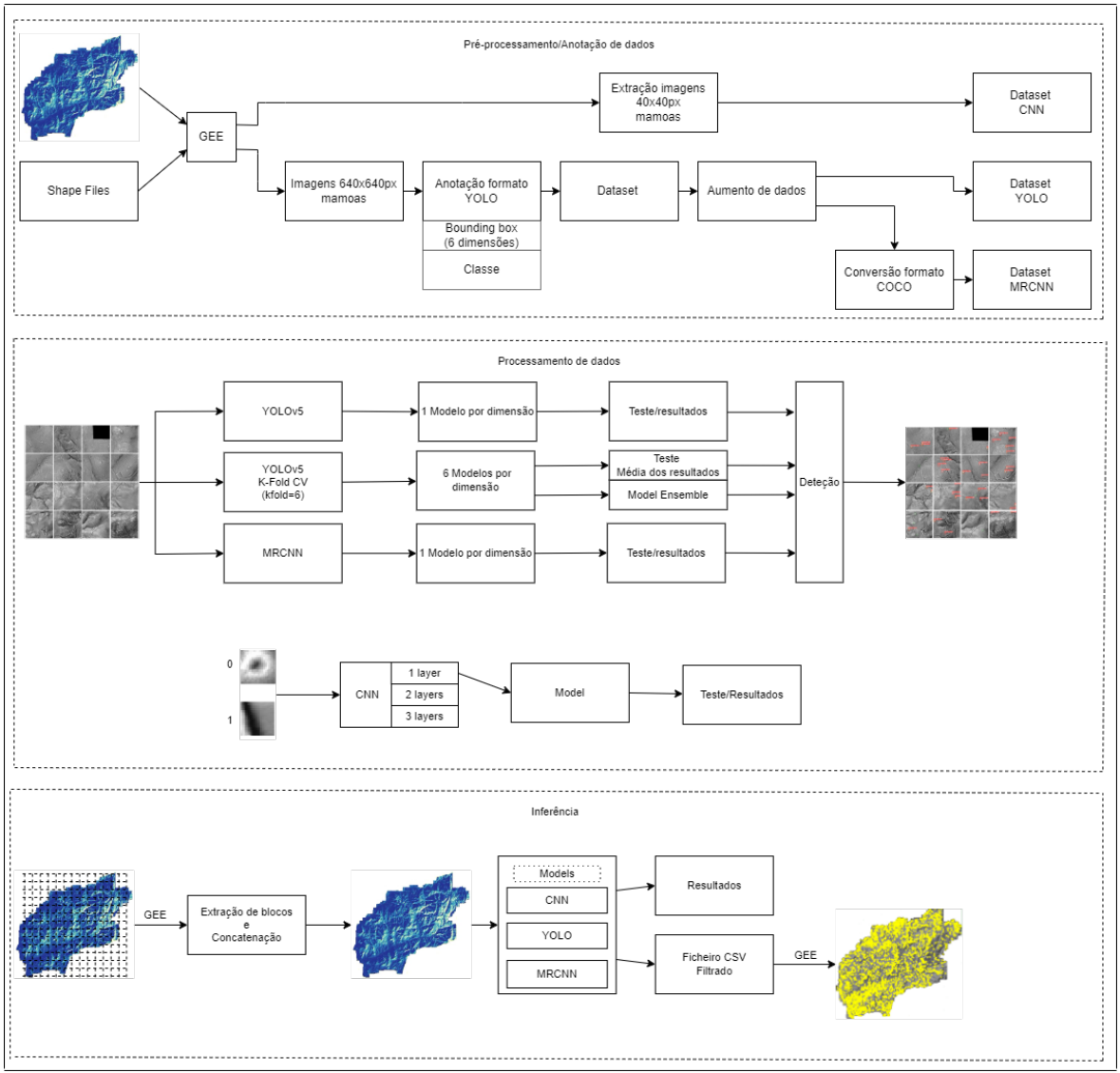

Figura 1: Fluxo de trabalho implementado com dados LiDAR. Pré-processamento e anotação para obter os datasets usados. Processamento de dados com o uso de algoritmos DL. Inferência a toda Imagem do Distrito.