RITA FILIPA
DOS SANTOS
AMANTE

DETEÇÃO DE VEÍCULOS E EDIFÍCIOS EM IMAGENS AÉREAS OBTIDAS POR DRONE

DETECTION OF VEHICLES AND BUILDINGS IN DRONE AERIAL IMAGES

Universidade de Aveiro
2022

RITA FILIPA
DOS SANTOS
AMANTE

DETEÇÃO DE VEÍCULOS E EDIFÍCIOS EM IMAGENS AÉREAS OBTIDAS POR DRONE

DETECTION OF VEHICLES AND BUILDINGS IN DRONE AERIAL IMAGES

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor António José Ribeiro Neves, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor José Silvestre Serra da Silva, Professor Associado com Agregação da Academia Militar.

Dedico este trabalho à minha família por todo o amor e apoio incondicional ao longo da minha formação pessoal e académica.

o júri / the jury

presidente / president

Professor Doutor Joaquim João Estrela Ribeiro Silvestre Madeira
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professora Doutora Catarina Helena Branco Simões Silva
Professora Auxiliar do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

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

agradecimentos / acknowledgements

A realização desta dissertação não seria possível sem a contribuição de algumas pessoas, que merecem o meu agradecimento e às quais estarei eternamente grata.
À Universidade de Aveiro e a todos os professores que me acompanharam no meu percurso académico pela elevada qualidade de ensino oferecido.
Ao orientador Professor António José Ribeiro Neves, pela sua orientação, total apoio, disponibilidade, pelo saber que transmitiu, pelas sugestões, opiniões e críticas e por todas as palavras de incentivo.
Ao coorientador Professor José Silvestre Serra da Silva, pela sua colaboração, visão crítica, oportuna e exigente, contribuindo para o enriquecimento de todas as etapas subjacentes ao trabalho realizado.
À Academia Militar Portuguesa, pela disponibilidade de registos e material para o desenvolvimento desta dissertação.
Ao investigador Daniel Duarte Canedo, pela sua disponibilidade e total colaboração no solucionar de dúvidas e problemas que foram surgindo.
Aos meus amigos César Miranda, Anabela Ribeiro e Lisa Correia, pela sua amizade, apoio, motivação e partilha de ideias nesta etapa da minha vida.
Aos meus avós, por todo o apoio que sempre me prestaram e por todas as boleias que me deram para ir para a Universidade.
Ao meu namorado, que esteve sempre ao meu lado, pelo companheirismo, compreensão, paciência e pelas constantes palavras de confiança.
Ao meu irmão e à minha cunhada, por todos os conselhos preciosos, total disponibilidade e encorajamento nos momentos cruciais desta jornada, bem como pela leitura crítica e atenta das versões preliminares desta dissertação, contribuindo para o seu aperfeiçoamento.
Aos meus pais, um agradecimento especial, tendo consciência que sozinha nada disto teria sido possível, por serem modelos de coragem, pelo seu apoio incondicional, incentivo, amor e paciência demonstrados e total ajuda na superação dos obstáculos que foram surgindo ao longo desta caminhada.
A todos, um sincero e profundo obrigado.

Palavras Chave

Inteligência Artificial, Aprendizagem de Máquina, Aprendizagem Profunda, Aprendizagem por Transferência, Visão Computacional, Deteção de Objetos, UAV

Resumo

A necessidade de desenvolver software para a análise de imagem aérea, capturada por Veículos Aéreos Não Tripulados, tem vindo a aumentar ao longo dos anos devido ao facto de serem cada vez mais utilizadas em diversos cenários do dia-a-dia. A deteção de objetos, técnica da Visão Computacional, é um dos problemas mais explorados nesta área e consiste na identificação e localização de objetos em imagens ou vídeos, com o auxílio de tecnologias de Inteligência Artificial.

Pretende-se com esta dissertação analisar o desempenho de algoritmos de Aprendizagem Profunda, para a deteção de veículos e edifícios em imagens aéreas. Foram escolhidos dois dos principais algoritmos descritos na literatura, Faster R-CNN e YOLO, este último na terceira e quinta versão, por forma a verificar qual apresenta melhor desempenho. Para o treino de cada algoritmo e realização de testes foi utilizado um conjunto de dados fornecido pela Academia Militar Portuguesa, o qual foi anotado e pré-processado.

Os resultados obtidos, no referido conjunto de dados, demonstraram que existe uma discrepância considerável entre os dois algoritmos, tanto a nível do desempenho como do tempo de deteção. O Faster R-CNN apenas se mostrou superior em relação às duas versões do YOLO no tempo de treino, pois foi o algoritmo que precisou de menos tempo. Entre as versões do YOLO, a quinta versão foi a que apresentou melhores resultados.

Keywords

Artificial Intelligence, Machine Learning, Deep Learning, Transfer Learning, Computer Vision, Object Detection, UAV

Abstract

The need to develop software for aerial image analysis, captured by Unmanned Aerial Vehicles, has increased over the years because their use has become more prevalent in different day-to-day scenarios. Object detection, a Computer Vision technique, is one of the most explored problems in this area and consists of identifying and locating objects in images or videos, with the help of Artificial Intelligence technologies.

The aim of this dissertation is to analyze the performance of Deep Learning algorithms for detecting vehicles and buildings in aerial images. Two of the main algorithms described in literature, Faster R-CNN and YOLO, the latter in the third and fifth versions, were chosen to verify which one is capable of better performance. The dataset provided by the Portuguese Military Academy, which was annotated and pre-processed, was used for the training of each algorithm and the performance of tests.

The results obtained in the abovementioned dataset demonstrate that there is a considerable discrepancy between the two algorithms, both in terms of performance and speed. Faster R-CNN only proved to be superior to the two versions of YOLO in terms of training speed, as it was the algorithm that required less time for training. Among the versions of YOLO, the fifth version showed the best results.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
CNN	Convolutional Neural Network
CPU	Central Process Unit
CSV	Comma-Separated Values
CUDA	Compute Unified Device Architecture
CV	Computer Vision
DL	Deep Learning
FC	Fully-Connected
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Networks
Fast R-CNN	Fast Region-based Convolutional Neural Network
Faster R-CNN	Faster Region-based Convolutional Neural Network
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
IoU	Intersection over Union
mAP	mean Average Precision
ML	Machine Learning
MS COCO	Microsoft Common Objects in Context
NMS	Non-Maximum Suppression
PMA	Portuguese Military Academy
PMA-DA	Portuguese Military Academy with Data Augmentation
R-CNN	Region-based Convolutional Neural Network
ReLu	Rectified Linear Unit
R-FCN	Region-based Fully Convolutional Network
RFCN-DF	R-FCN based on Deformable-ConvNets
RoI	Region of Interest
RPN	Region Proposal Network
SSD	Single Shot Multibox Detector
SVM	Support-Vector Machine
TL	Transfer Learning
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicles
VJ	Viola Jones
XML	Extensible Markup Language
YOLO	You Only Look Once

CHAPTER 1

Introduction

This chapter is divided into five subchapters. First, a brief contextualization of the theme and the main motivations which led to choosing this dissertation are presented. Next, the objectives of the work are defined, both at a general and specific level and the organization of the document by chapters is presented with a brief description of the subjects covered in each one. Finally, reference is made to an extended summary carried out within the scope of this dissertation.

1.1 Contextualization

Currently, with the advancement of technology, access to aerial images has been simplified due to the expansion of Unmanned Aerial Vehicles (UAV). Their presence has made large amounts of aerial visual data accessible and, consequently, object detection algorithms have been improved.

UAV, as the name implies, are remotely controlled aerial vehicles which enable autonomous navigation and capture visual data through a high-resolution camera from different locations, angles and altitudes [1].

The facility with which UAV access hard-to-reach places, in a mobile way, has boosted their use in different application areas, both in a civil and military context. At the civil level, they have several benefits in areas such as surveillance, public and private security, disaster assistance, agriculture and the environment, among others [2]. They are of enormous practical interest in a military context, especially for the defense sector, as they help to predict enemy movements and plan preventive measures, saving time and effort [3]. UAV can be used for reconnaissance of enemy positions, thereby protecting the Military Unit through surveillance images, enabling the detection of camouflaged soldiers, as well as enabling offensive operations by carrying small explosives which, in a suicide mission, collide with the target to be destroyed.

Another way of capturing aerial images is through satellites. But although satellites also provide a panoramic view, they are more expensive and the information they provide cannot

be updated. Compared to satellites, UAV are lighter, less expensive and require less human and technological resources to acquire images [1].

Compared to terrestrial images, the aerial images captured by UAV stand out for covering a large field of view and presenting high spatial resolution [1]. However, they present several challenges such as orientation, viewing angles (top view, side view and front view), shadows, complex backgrounds, lighting, reduction of object scale due to the altitude of the flight and the possibility of blurred images due to an unstable trajectory [4].

Increasingly, UAV have been applied in the field of Computer Vision (CV). Just as the human being has the ability to capture and interpret images, CV presents a similar process. Cameras or sensors capture images, the neural networks receive these images and, through Artificial Intelligence (AI) algorithms, the information is extracted. In this light, CV can be portrayed as a process of modeling and identification, extracting information from images or videos. This is only possible due to the evolution of computers, in terms of processing, memory and storage capacity, and it can be said that there is a direct correlation between CV and computers [5].

1.2 Motivation

CV has been playing an important role in digital transformation. Its growth is due to the increase in data that needs to be stored and analyzed, an arduous and time-consuming task to be performed by humans. It can be applied to solve various problems, such as image classification, object detection, object location, among others.

Object detection in aerial images is a recent topic, with a very significant growth outlook, making its study complex and interesting. Vehicle detection, specifically, can provide several real-world applications, such as road traffic monitoring, parking management and control, road accidents, screening for illegal activities, operations to support government agencies and surveillance of enemy troops. Building detection can be useful for urban planning and construction, roof monitoring and control of illegal constructions.

Taking all this into consideration, there is an increasing interest in the study of vehicle and building detection in aerial images obtained by UAV. Two Deep Learning (DL) algorithms, well known in the literature for object detection, will be studied: Faster Region-based Convolutional Neural Network (Faster R-CNN), a two-stage algorithm, and You Only Look Once (YOLO), a one-stage algorithm. Regarding the YOLO algorithm, two of its versions will be studied, YOLOv3, which is one of the most mentioned algorithms in literature, and YOLOv5l, which is the most recent version.

1.3 Objectives

The general objective of this dissertation is to study and analyze the performance of classification algorithms for the detection of vehicles and buildings, using images obtained by UAV.

To achieve the general objective, some specific objectives were defined, namely:

- Research and select pretrained algorithms for the detection of objects in aerial images.
- Annotate the set of images provided by the Portuguese Military Academy.
- Analyze the efficiency of pretrained algorithms in the set of images provided.
- Configure and train the algorithms for the detection of target objects in the set of images provided.
- Evaluate and compare the performance of implemented algorithms.

1.4 Structure of the document

This thesis is divided into 5 chapters, namely:

- Chapter 1 - Introduction: framework, motivation, description of the established objectives, a brief description of the paper's organization and reference to the elaboration of an extended abstract.
- Chapter 2 - Literature Review: survey of theoretical concepts which are key to understanding this dissertation and reference to the current state of knowledge, so that a starting point for the study in question can be established.
- Chapter 3 - Methodology: description of the adopted methodology, using an illustrative diagram.
- Chapter 4 - Results: presentation of the resources used, dataset description, inference process of pretrained algorithms, training configuration of each algorithm and inference process of trained algorithms.
- Chapter 5 - Conclusions: summary of results and references to future work.

1.5 Extended Abstract

The work involved in elaborating this dissertation resulted in the writing of the Extended Abstract, "Discrimination between vehicles and buildings in military aerial images", which was submitted to the Conference of the International Society of Military Sciences, which will take place from October 10^{th} to 13^{th}, 2022 in Lisbon. This document can be found in Appendix E - Extended Abstract.

CHAPTER 2

Literature Review

This chapter outlines some theoretical concepts fundamental to the understanding of this dissertation, a brief contextualization of the chosen algorithms and references to work carried out.

2.1 Artificial Intelligence

In the past, it was unthinkable that machines could perform tasks similar to human intelligence. Over time, AI has become increasingly present, and has turned into a reality.

The pioneer in AI was Alan Turing, considered to be the "father of AI". In 1936, Alan Turing described the fundamental concepts of a computer, which became known as the "Turing Machine", which then gave rise to the concept of the "Intelligent Machine". Turing created a test to evaluate the intelligence of a computer system compared to that of a human being, called the "Turing Test" [6].

Due to authors' different perspectives and sources of inspiration, it becomes difficult to define the term AI in a simple and effective way. Over time, different definitions have emerged. In the 1950s, the objective of AI was first defined as the development of machines capable of behaving intelligently [7]. Subsequently, AI was seen as a discipline whose objective involved the study and construction of artificial entities with knowledge identical to that of the human being [8].

It can be said that AI tries to simulate human reasoning, knowledge, behavior and learning through machines [9]. The advancement of technology has made these intelligent machines faster and more efficient, making it so that most manual tasks can be replaced by automated tasks.

Other concepts are related to AI, such as Machine Learning (ML), DL and CV, which are interconnected, as illustrated in Figure 2.1 [10].

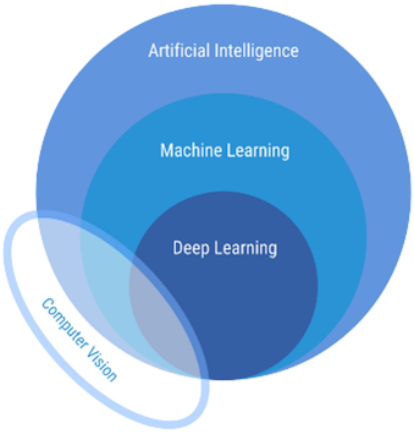


Figure 2.1: Machine Learning and Deep Learning are subsets of Artificial Intelligence (from [11]).

2.2 Machine Learning

Gradually, with access to more extensive computing resources and a greater amount of data, which provide faster and more successful processing to support more complex tasks, ML has emerged.

In 1997, a more engineering-oriented definition appeared by Tom Mitchell, where he cites that *"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E"* [12].

It can be said that ML is a subfield of AI which designs and programs algorithms, enabling machines to learn from data, improve their accuracy and find patterns, without human intervention. The input data used in these algorithms needs to be preprocessed [13].

For a machine to be able to perform tasks independently, three distinct requirements are needed (Figure 2.2) [14]:

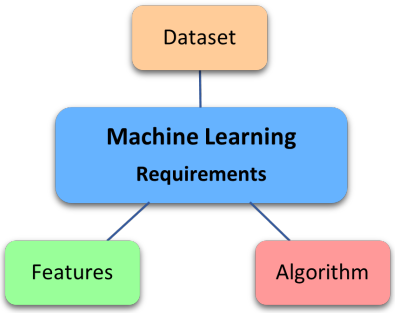


Figure 2.2: The three requirements needed to "educate" a machine (adapted from [14]).

- **Dataset**: one of the most important requirements for machine learning is the input dataset, which, depending on the desired objective, can consist of images, text, videos, among others. Collecting and processing data is a task which requires extensive resources, time and effort. In most algorithms, the dataset is divided into three parts [15]:

  - **Training Dataset**: data used in the algorithm learning process.
  - **Validation Dataset**: data used to impartially evaluate the algorithm's performance after training, to adjust its hyperparameters.
  - **Test Dataset**: data used to evaluate the algorithm's performance, after its training and validation.
- **Features**: attributes associated with the dataset which make it possible for the machine to create a correspondence between the input data and the desired response, enabling the establishment of patterns to optimize learning.
- **Algorithm**: set of instructions executed in a certain order, responsible for executing the machine training process.

ML algorithms can be divided into four categories, depending on the amount and type of supervision they receive during the training process: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning [12].

Supervised learning algorithms predict an output from a labeled input, requiring large amounts of labeled data for model refinement and more accurate results. Each output is assigned a label, which can be a numeric value or a class [16]. These can be divided into two: classification algorithms and regression algorithms. The former is used to predict/classify discrete values, where the output can only assume a set of predefined labels. On the other hand, the latter is used to predict real number outputs [12].

Contrary to supervised learning, the input data of unsupervised learning algorithms are not labeled, that is, the desired response is not informed. These algorithms try to identify patterns among the data, to group them according to the detected similarities [16].

Semi-supervised learning algorithms use labeled and unlabeled data for training, having a higher amount of unlabeled data, since they are more economical and easier to acquire. The algorithms first apply unsupervised learning to gather similar data and then, with the labeled data, apply supervised learning to label the remaining data [16].

Reinforcement learning algorithms involve three components: the agent (entity which interacts with the environment and makes decisions), the environment (space where the agent performs these actions) and actions (what the agent is capable of doing). The agent is responsible for the choices of actions, using trial-and-error tests to maximize the reward or minimize the risk [9].

2.3 Deep Learning

In recent years, DL has been the focus of study and implementation, contributing to a significant evolution in ML methods. Its evolution is due to the implementation of algorithms based on neural networks, which have become more complex, and to the progress of computers.

Just as the human being is able to process information, DL can learn and classify objects by itself, through the interpretation of data [13].

Compared to ML, DL focuses on creating algorithms which simulate the functioning of neurons in the human brain. These algorithms are comprised of an interconnected network of

nodes [10], which use vast quantities and a variety of data for training, and Artificial Neural Network (ANN), which enable learning without a manual extraction of features, eliminating the need for the pre-processing of data necessary in ML [13], as illustrated in Figure 2.3.

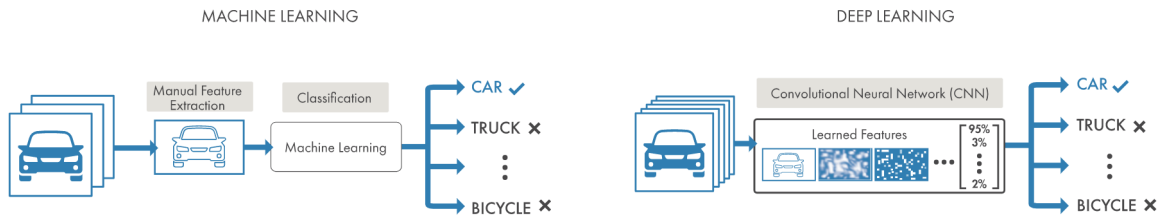


Figure 2.3: Comparison of a machine learning approach to categorizing vehicles (left) with deep learning (right) (from [17]).

The two stages of DL are the training process and the inference process. The first trains the algorithm so that it learns from the data and labels it. The second consists of evaluating and labeling new data, using the network trained in the previous process [13].

DL enables the use of several processing computers to improve the performance, as well as the use of Graphics Processing Unit (GPU) to increase the training speed [10]. These have led to great improvements in the area of CV, with regard to speech recognition, object detection, among other applications.

The inspiration for the creation of an ANN came from the 1943 study on the functioning of biological neurons found in the human brain, by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts [12].

The structure of an ANN consists of vertically stacked elements to which a layer name is attributed. An ANN is therefore comprised of an input layer, one or more hidden layers and an output layer. The input layer receives the input data and transfers it to the next layers. The hidden (intermediate) layers process the incoming data, applying complex functions designed to produce the intended result. Finally, the output layer receives the processed data and produces the final result. In general, the output of the previous layer provides the input for the next layer. In other words, the following layers use information obtained in the previous layers, enabling the construction of complex concepts from simpler ones (Figure 2.4)[18].

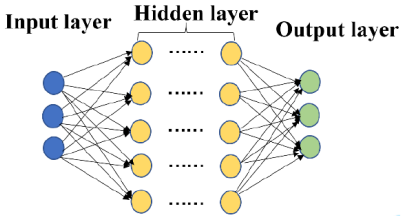


Figure 2.4: Structure of Artificial Neural Networks (from [19]).

In an ANN, each node, or artificial neuron, receives several signals as input, assigning a weighting coefficient ("weight") and an associated limit. If the output layer of any node is above the limit value, that node is activated, sending data to the next layer, otherwise no data is sent. In these networks, learning is achieved by adjusting the weighting coefficients [13].

Thus, the learning is *'Deep'* due to the number of hidden layers in the neural network, which provide part of the learning [6].

In the 1980s, with the study of the visual cortex of the brain, Convolutional Neural Network (CNN) or ConvNet emerged, used in classification and CV tasks. In his study from 1998, Professor LeCun *et al.* [12] introduced the LeNet-5 architecture, adding two new building blocks: convolutional layers and pool layers. After the success of this architecture, others were developed over the years, such as AlexNet, ZFNet, GoogLeNet, VGGnet and ResNet [20].

The function of a CNN is to reduce images into a format which is easier for processing, without wasting features. These are composed of several layers of artificial neurons, where the first layer extracts the basic features and passes them to the next layer which detects more complex features in a continuous process [20].

A CNN is composed of an input layer, several hidden layers and an output layer (Figure 2.5). The most common layers are: Convolutional layer, Pooling layer, Rectified Linear Unit (ReLu) and Fully-Connected (FC) layer. The most important building block is the Convolutional layer, where most of the computation takes place. The ReLu enables faster and more effective training by activating only those features carried over to the next layer. The Pooling layer is responsible for reducing the size and number of input parameters. The FC layer is assigned the classification task based on the features extracted from the previous layers [21].

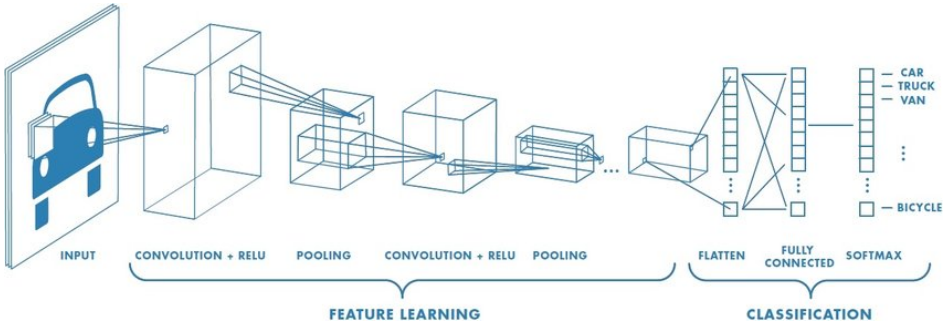


Figure 2.5: Example of a network with many convolutional layers (from [22]).

Most object detection algorithms are based on CNN as they have a weight sharing structure and take advantage of the GPU, an important factor in object recognition [21].

2.4 Transfer Learning

Transfer Learning (TL) is considered a DL technique which transfers the knowledge the trained neural network acquired to a new related task, without the need to train the new network from scratch [23].

In TL there are three possible approaches: pretrained network as a classifier, pretrained network as a feature extractor and fine-tuning (Figure 2.6). The first consists of directly applying the target task to the pretrained network, without making changes to the network and without training it, using the pretrained architecture and weights and, on these, making predictions with the new dataset. The second is to freeze the feature extraction, remove the

classifier, add a new classifier and train it from scratch. The third is to freeze some of the network layers used in feature extraction and train the remaining layers and the network classifier [24].

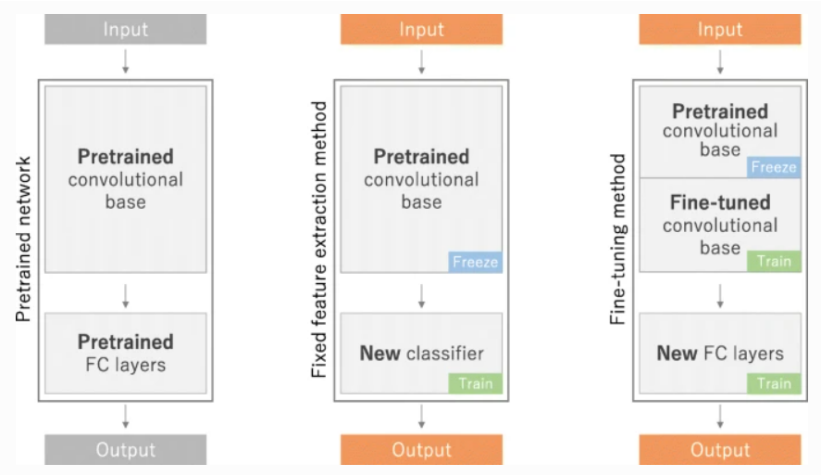


Figure 2.6: Comparison of the three Transfer Learning approaches: pretrained network as a classifier (left), pretrained network as a feature extractor (middle) and fine-tuning (right) (from [25]).

The advantages outweigh the disadvantages in using TL. Relative to the advantages, it does not need a large amount of data for training, reduces training time and improves the performance of neural networks. However, it should only be used when the pretrained task has similarities with the one to be trained and when the algorithm architectures are similar [23].

TL is often used to avoid *overfitting*, which occurs when the algorithm precisely fits the training data, learning irrelevant information about the data, thus underperforming on the new dataset [25].

2.5 Object detection algorithms

In the last 20 years, progress has been made in object detection, with two important periods being highlighted: traditional object detection period (before 2014) and deep learning-based detection period (after 2014) [26]. During the first period, the features base used in object detection algorithms was artisanal, where the image features were manually extracted (color, texture, contours, among others) and then the classifier was trained. The second period can be divided into two-stage and one-stage.

2.5.1 Faster R-CNN algorithm

The Region-based Convolutional Neural Network (R-CNN), proposed by Ross Girshick *et al.*, is one of the first approaches used to determine the object in the image, consisting of the following steps [27]:

- **Determination of a Set of Hypotheses**: through selective research, the list of hypotheses is defined, which includes 2,000 different regions partially overlapping each other. Each of these regions is called a Region of Interest (RoI).
- **Feature extraction using CNN and its encoding into a vector**: each hypothesis is transferred independently and separately from each other to the CNN input.
- **Object classification within each hypothesis**: uses the list of classification models to determine which object is in the region under analysis, through the previously extracted characteristics.

Since the R-CNN performs excessive calculations on numerous proposals, resulting in a slow detection speed, the selective search algorithm is not flexible, and training is time-consuming. The CNN is responsible for extracting the characteristics of all the RoI created [28].

Later, to overcome some shortcomings of the R-CNN, the Fast Region-based Convolutional Neural Network (Fast R-CNN) algorithm, proposed by Ross Girshick, appeared, where, unlike the R-CNN, the image is provided in its entirety to the CNN to create a feature map. This algorithm presents a RoI Pooling layer which, through a selective search parallel network and based on the obtained feature map, creates the RoI and assigns features to each region [28]. However, Fast R-CNN has several limitations such as a high calculation time and a complicated and time-consuming approach [29]. Then came the Faster R-CNN, proposed by Shaoqing Ren *et al.*, to overcome the limitations of the previous algorithms. The latter's main contribution was the introduction of the Region Proposal Network (RPN) [26].

In Faster R-CNN, the input image is provided in its entirety to the CNN, which learns and identifies the RoI, combined with a parallel network, called the RPN [30]. The Faster R-CNN detection process follows several steps. Firstly, it extracts a feature map from images; secondly, it generates hypotheses from the feature map, determining the approximate coordinates; thirdly, it compares the hypothesis coordinates with the feature map, using RoI and, finally, it classifies the hypotheses, updating the coordinates [27].

2.5.2 YOLO algorithm

You Only Look Once (YOLO), as its name implies, looks at the image as a whole, only once, differentiating it from previous models, based on the use of RoI in the classification of images.

Redmon *et al.* [31] developed the extremely fast and accurate unified model for object detection called YOLO, where a single neural network pre-determines limiting boxes and class probabilities directly from complete images, ideal for applications which rely on fast and effective object detection. They concluded that the model contains more localization errors, but predicts fewer false positives, learning very general representations of objects.

All YOLO architectures can be divided into three components [32]:

- **Backbone**: corresponds to a CNN responsible for extracting feature maps from an image.
- **Neck**: mixes and matches the characteristics created in the backbone to capture information to be used in the next step.

- **Head**: with the information received from the previous step, it predicts the limiting boxes and the respective classes.

YOLO uses a fully convolutional neural network which performs the object recognition and localization step at the same time and returns the limiting box position and its class directly in the output layer [33]. Initially, YOLO resizes the input image to a specific size, which is then sent to the CNN system, and finally, the network prediction results are processed to detect the target [34].

The first version has 24 convolutional networks and 2 fully connected layers. It has some limitations. For example, it does not detect low resolution objects; it does not provide more than one limiting box for a given region; it does not detect some objects or their details; it does not incorporate tests with multiple resolutions; it is not flexible and has difficulty in locating [29].

The second version, YOLOv2 or YOLO9000, proposed by Joseph Redmon and Ali Farhadi, can be run in a variety of image sizes, to balance speed and accuracy [35]. It uses Darknet-19 as a backbone for resource extraction and has 19 convolutional layers and 5 max-pooling layers, having removed all fully convolutional layers from the previous version [29].

The third version, YOLOv3, proposed by Joseph Redmon and Ali Farhadi, uses Darknet-53 as backbone for resource extraction and has 53 convolutional layers and 23 residual layers. The novelty of its architecture is that it does not contain a pooling layer and uses three-scale feature maps to predict the position of the object, which is an advantage over previous versions, regarding the detection of small objects. It divides the feature map into grids, each being used to predict limiting boxes and detect the categories of objects [33]. YOLOv3 has two main problems: label rewriting and the imbalance between limiting boxes. The first is due to the detector not using rewritten limiting boxes for the training process. The second happens when the centers of two limiting boxes are present in the same cell, and one can be replaced by the other [34].

The fourth version, YOLOv4, proposed by Alexey Bochkovskiy *et al.*, uses CSPDarknet53 as a backbone for resource extraction, to improve the algorithm's speed and accuracy [36].

The latest version, YOLOv5, presented by the company, Ultralytics, was developed from the open-source PyTorch library and is considered the lightest version of the YOLO algorithms. It uses CSPDarknet53 as a backbone and its structure replaced the first three layers of the YOLOv3 algorithm, reducing Compute Unified Device Architecture (CUDA) memory needed and increasing direct propagation and backpropagation [37].

YOLOv5 can be divided into five different models, trained with the Microsoft Common Objects in Context (MS COCO) dataset [38]:

- YOLOv5n (nano): the smallest, suitable for mobile solutions.
- YOLOv5s (small): best suited for inference execution on the Central Process Unit (CPU).
- YOLOv5m (medium): medium-sized, ideal for vast quantities of training data.
- YOLOv5l (large): relatively large in size, used for the detection of small objects.
- YOLOv5x (extra-large): the largest of all, with better accuracy, but slower.

These differ from each other in depth and size, yielding different values of accuracy and performance [38].

2.6 Related work

After a survey and selection of articles, the ones which fall within the scope of the work developed in this dissertation are summarized below.

Wang *et al.* [39] built a new dataset with 600 images for vehicle detection, using images obtained by Drone and improved the Faster R-CNN algorithm. They found that, when using Faster R-CNN with VGG16, some problems with the objects were detected, such as low resolutions, uncertain directions and background complexity, causing missed and false detections. So, to solve these problems, they selected the best ResNet algorithm to replace the VGG16 and used the Feature Pyramid Networks (FPN) to combine low-level features with high-level ones. They concluded that the improved Faster R-CNN can be used to detect vehicles in aerial images, with an accuracy of 96.83%. Compared to the original Faster R-CNN, they were able to obtain better results by 3.86%.

The research by Xu *et al.* [40] reveals that the Faster R-CNN algorithm is effective regarding changes in lighting and in-plane rotation, with the detection speed not being sensitive to the detection load, which is almost constant. The Faster R-CNN algorithm was analyzed for vehicle detection from low altitude UAV images captured at signaled intersections, compared with the other algorithms, Viola Jones (VJ) Detector and Histogram of Oriented Gradients (HOG) + Support-Vector Machine (SVM). From the analysis of the results, they recorded that the Faster R-CNN obtained Completeness (96.40%) and Correctness (98.43%) with real-time detection speed (2.10 f/s).

In the study by Zheng *et al.* [41], for the detection and identification of buildings from images obtained by UAV, the Faster R-CNN algorithm was implemented on different hardware platforms. The algorithm reached an accuracy of 93.2% with an average processing time of 74 ms in image recognition, concluding that this DL algorithm is viable for the task in question, presenting high efficiency. Another noteworthy aspect was the variation in detection time depending on the hardware platforms used. These authors also mention that the accuracy rate of the algorithm can be improved if the training dataset is broader.

The authors Ammar *et al.* [1] evaluated the performance of three algorithms for vehicle detection using aerial images: Faster R-CNN, YOLOv3 and YOLOv4. The Faster R-CNN algorithm was implemented with two different feature extractors, Inception v2 and ResNet50. They analyzed two datasets with different characteristics such as UAV altitude, camera resolution and object size: Stanford and PSU, where the first is comprised of more data and larger objects. They concluded that the performance of the algorithms depends on the characteristics of the dataset and the representativeness of the training images. Faster R-CNN implemented with Inception v2 was faster than with ResNet50, although they had similar accuracy levels. Faster R-CNN presented better average precision, but a slower inference speed than (YOLO)v4, in the Stanford dataset. YOLOv4 showed better accuracy and faster

inference speed than Faster R-CNN in the PSU dataset. Concerning YOLOv3 and YOLOv4, their prediction difference was insignificant.

According to Zhang *et al.* [42], there is still some difficulty in detecting objects in UAV images with good performance due to certain factors, such as: altitude (high altitudes produce low resolution images, making objects small and confusing), varied object orientations and reduced amount of labeled aerial imagery data. Starting from this base, the authors created a dataset of high-resolution UAV images, which they called MOHR, containing 10,631 annotated images with five classes of objects, displayed with three cameras. Six object detection algorithms were applied to this dataset, based on DL, namely Single Shot Multibox Detector (SSD), Region-based Fully Convolutional Network (R-FCN), Faster R-CNN, YOLOv2, YOLOv3 and R-FCN based on Deformable-ConvNets (RFCN-DF). They concluded that the R-FCN solves the problem of position sensitivity relative to the Faster R-CNN, as well as improves the average detection accuracy; RFCN-DF is more effective in detecting objects with irregular contours; two-stage detectors produce a relatively higher average accuracy compared to one-stage ones, which can be attributed to the generation of region proposals.

Dikbayir *et al.* [43] developed an application for vehicle detection in aerial images to increase the performance of the YOLO algorithm. These authors state that the performance of one-stage algorithms, such as YOLO, decreases as the object size decreases, giving more importance to speed than to performance. When comparing Faster R-CNN and YOLO, they observed that Faster R-CNN has better performance but slower detection speed. They implemented Faster R-CNN to increase the performance of YOLOv3, taking advantage of the best characteristics of both algorithms, accuracy and speed. They used the Munich dataset which contains high-resolution aerial images above 100 meters. The dataset was labeled and adapted to the Faster R-CNN algorithm which, after training, was used in the YOLO algorithm. They concluded that this implementation increased the performance of the YOLO algorithm by 3.2%.

Xu *et al.* [44] compared the performance of YOLOv3 with YOLOv4 and the parameterization of YOLO-based models for the detection of small objects in the AU-AIR dataset. They concluded that YOLOv4 is slightly more efficient in detecting smaller objects and reduces the cost of hardware training. However, they questioned whether YOLOv4 would be the most suitable in terms of detection accuracy and speed for small objects.

Yin *et al.* [45] improved the Faster R-CNN model, in terms of feature extraction, to obtain better detection accuracy, using feature fusion at various scales, combining the residual module and the pool layer. As the input image is processed by several layers and the receptive field increases, some information is lost, which makes it difficult to locate the boundaries of small objects, affecting detection accuracy. By utilizing the fusion of low-level structural features and high-level semantic features, small objects can have more information and improve the results. They concluded that the average precision increased by 1.06%.

Saetchnikov *et al.* [27] conducted a comparative study on the efficiency of different deep neural networks for the detection of objects in satellite and aerial images. They used R-CNN, Fast R-CNN, Faster R-CNN, SSD and YOLOv3, for three datasets. In the "Stanford

Campus" dataset, they concluded that YOLOv3 achieved better average accuracy (87.12%) and faster analysis (26.82 fps) compared to the R-CNN, Fast R-CNN, Faster R-CNN and SSD algorithms. Even with other datasets, "Dota v1.5" and "xView 2018 Detection", YOLOv3 continues to show advantages over the other algorithms, despite a slight decrease in accuracy. In addition to accuracy, YOLOv3 excelled in processing speed for real-time object detection by UAV.

Nepal *et al.* [37] analyzed object detection algorithms to spot safe landing spots in case the UAV suffer an in-flight failure. They compared the accuracy and speed of the YOLOv3, YOLOv4 and YOLOv5l algorithms using the DOTA dataset. They concluded that the three algorithms satisfy the requirements for object detection in real time. However, YOLOv5l stood out due to its greater precision.

Several approaches and methodologies have been developed to solve object detection problems, which have become more viable with the use of algorithms based on DL. Most of the studies related to object detection in aerial images focus on two and one-stage detectors, mainly Faster R-CNN and YOLO, respectively. Although satisfactory results for object detection in aerial images have already been obtained using these algorithms, this task continues to be an expanding line of research, in which even better results are expected by establishing the best relationship between the average precision and detection speed.

CHAPTER 3

Methodology

This chapter describes the methodology implemented to meet the established objectives.

Several steps were taken to obtain the final results, which are illustrated in Figure 3.1.

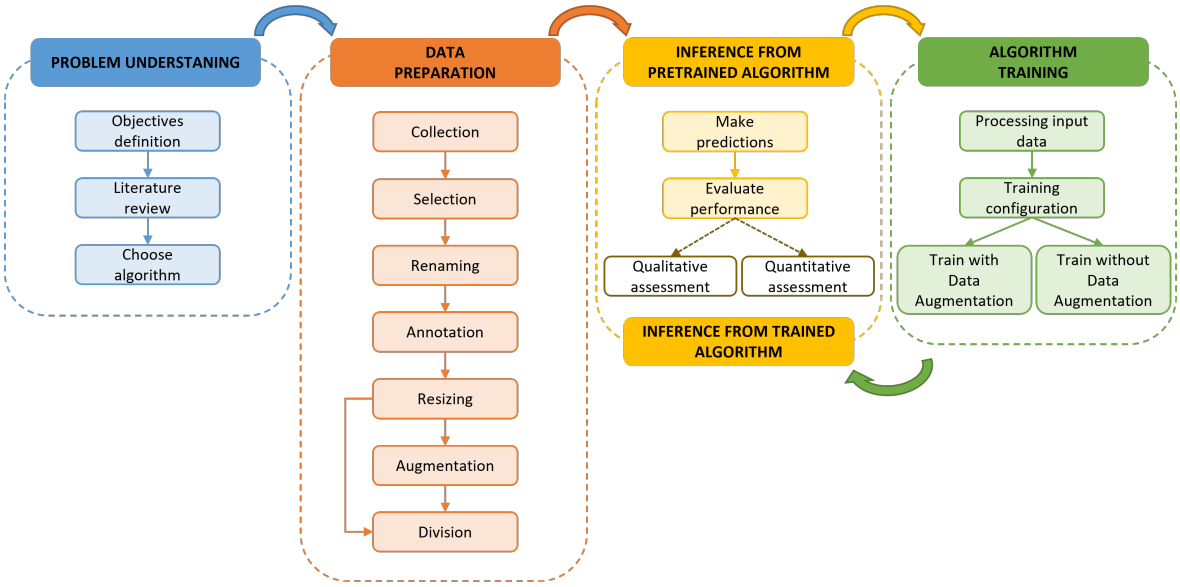


Figure 3.1: Diagram of the methodology used to detect vehicles and buildings in aerial images.

After defining the general and specific objectives described in section 1.3, a literature review was prepared which presented the specific theoretical foundations to understand the present dissertation, as well as a brief survey of the current state of knowledge on the subject. The choice of "deep learning based detection period" algorithms was also included in this first step. These were chosen because traditional methods are slow and inefficient compared to the more recent methods based on CNNs. Of these, a two-stage (Faster R-CNN) and a one-stage (YOLO) algorithm were chosen. Regarding YOLO, two versions were used, YOLOv3 and YOLOv5l.

The next step was based on the preparation of data provided by the Portuguese Military Academy. It started by selecting the images which demonstrated the conditions to be used. These were renamed, annotated and resized. Subsequently, two datasets were created, with and without data augmentation, called Portuguese Military Academy (PMA) and Portuguese Military Academy with Data Augmentation (PMA-DA), respectively. Finally, both datasets were divided into three subsets: training, validation and testing.

The third step included the inference process of the pretrained algorithms, using the pretrained network as a classifier approach of TL. Having a pretrained algorithm, whose authors claim can recognize a given object, an untrained input image was inserted and the detections predicted by the algorithm were analyzed. The performance of the algorithms was assessed qualitatively, based on the observation of the predictions obtained for a new image, and quantitatively, where the detection times were analyzed. The algorithms were then trained because they were not sufficiently capable of detecting target objects.

In the fourth step, the input data were processed, some training parameters were configured and the algorithms were trained. The following were the adjusted training parameters:

- *batch size*: number of training examples used in an iteration, which depends on the amount of memory available; the larger the batch size the more memory is needed.
- *epcohs*: number of complete passes through the training dataset, set between 1 and infinity (∞), one epoch corresponding to one cycle of the complete training dataset.
- *steps*: integer that determines how many training steps the algorithm will perform
- *learning rate*: decimal number, between 0.0 and 1.0, that controls how fast the network updates its weights throughout the training.
- *Intersection over Union (IoU) limit*: decimal number, between 0.0 and 1.0, which quantifies the overlap between the true and predicted limiting box. If the overlap result is greater than or equal to the threshold, the detection is considered correct, otherwise, the detection is considered incorrect.
- *confidence limit*: decimal number, between 0.0 and 1.0, which can be interpreted as a confidence percentage. If the confidence value of the prediction is lower than the established limit, the algorithm does not return this prediction.

The Faster R-CNN algorithms are configured with the step parameter, while YOLO is configured with the epoch parameter. Therefore, it was necessary to calculate the number of steps relative to the number of epochs, following the equation 3.1:

$$steps = \frac{\text{number of epochs} \times \text{number of training examples}}{\text{batch size}} \tag{3.1}$$

Each algorithm was trained with the PMA dataset and the PMA-DA dataset, according to a pretrained network as a feature extractor approach of TL.

The last step included the inference process of the trained algorithms, to make predictions about the PMA and PMA-DA, test subsets, verifying whether the algorithm was well trained for the objective in question. The algorithms were evaluated according to a qualitative and a quantitative approach. The first, as mentioned in the third step, corresponds to the direct observation of the detection results and the second corresponds to the interpretation of the results obtained after training the algorithms. In the quantitative assessment, four aspects were taken into consideration: the training time, the results obtained by the evaluation metrics, the interpretation of the classification and localization loss graphs and the detection time.

To compare the algorithms, metrics had to be obtained to assess their quality. Among the various existing metrics, the following were used to evaluate the algorithms: IoU, Precision, Recall and mean Average Precision (mAP). Precision and Recall were evaluated for just a single 50% IoU value for each class, named P@0.5 and R@0.5, respectively. The mAP was evaluated for a single value of 50% IoU, named by mAP@0.5.

The metrics are created from a confusion matrix, which is obtained by comparing the real class and the predictive class, defining the set of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) [46]. These concepts can be described as follows [43]:

- TP: examples classified as positive in the original data and correctly predicted as positive by the algorithm.
- TN: examples classified as negative in the original data and correctly predicted as negative by the algorithm.
- FP: examples classified as negative in the original data and incorrectly predicted as positive by the algorithm.
- FN: examples classified as positive in the original data and incorrectly predicted as negative by the algorithm.

Intersection over Union

The Intersection over Union (IoU) metric is necessary for the determination of TP or FP, quantifying the similarity between the real and predicted limiting box, to verify if the predicted detection is valid. According to the equation, 3.2, IoU is the ratio between the intersection area and the union area between the actual and predicted limiting boxes [47].

$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}} \tag{3.2}$$

The IoU value is normalized, ranging from 0.0 to 1.0. The closer the value is to 1, the closer the limiting boxes are (actual and predicted). That is, if a score is 0.0 there is no overlap between the limiting boxes, whereas if a score is 1.0, it means that there is a total overlap of the limiting boxes, which is the ideal situation [48] (Figure 3.2).

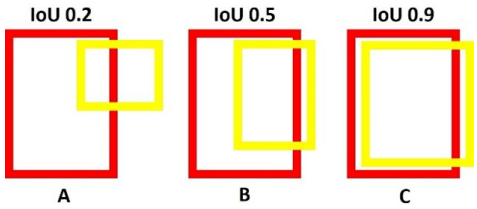


Figure 3.2: Example of IoU values: (a) 20% overlap between the 2 boxes; (b) 50% overlap between the 2 boxes; (c) 90% overlap between the 2 boxes (from [49]).

By comparing this result with a certain defined limit (also between 0.0 and 1.0), a correct or incorrect detection can be classified. If the IoU result is greater than or equal to the stipulated limit, the detection is considered correct; otherwise, the detection is considered incorrect [47].

Furthermore, if there are several limiting boxes for the same detected object, only the one with the highest overlap rate will be considered as positive detection, while the rest are not considered. This is called Non-Maximum Suppression (NMS) [50].

Precision

The Precision metric, as the name implies, refers to the accuracy of the algorithm being used to determine how many of the positively detected objects are positive. In other words, it evaluates the detection of true positives. This metric is defined by the ratio between the number of positive objects classified correctly and the total number of objects classified as positive (correctly or incorrectly), according to the equation 3.3. Thus, the higher the Precision value, the greater the number of positive objects detected correctly (maximizing the true positives) and the lower the number of positive objects detected incorrectly (minimizing the false positives) [43].

$$Precision = \frac{TP}{TP + FP} == \frac{TP}{\text{All detections}} \tag{3.3}$$

Recall

The Recall or Sensitivity metric complements the Precision metric, analyzing the amount of object identification the algorithm obtained. That is, it is used to determine how many objects should have been positively predicted. This metric is defined by the ratio between the number of correctly classified positive objects and the total number of positive objects (equation 3.3). Thus, the higher the Recall value, the greater the number of positive objects detected (desired) [43].

$$Recall(R) = \frac{TP}{TP + FN} = \frac{TP}{\text{All ground truth}} \tag{3.4}$$

Mean Average Precision

The Average Precision (AP) metric evaluates the number of detections made and how many of those detections were correct, according to the actual number of existing objects. This metric is defined by the ratio between Precision and Recall (equation 3.5), calculated for each class [47]. Thus, the higher the AP value, the better the algorithm's performance [20].

$$AP = \frac{Precision}{Recall} = \frac{TP + FP}{TP + FN} \tag{3.5}$$

Once the AP has been calculated for each class, the average of these values for the entire algorithm can be determined by calculating the Mean Average Precision (mAP) metric. This corresponds to the sum of all the AP values over all classes, with the APi variable corresponding to the AP in class i and N the total number of classes evaluated (equation 3.6) [47].

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3.6}$$

Regarding the classification and localization loss graphs of each algorithm, these were obtained through TensorBoard, a TensorFlow tool which enables the visualization of statistical information of a neural network.

CHAPTER 4

Results e Discussion

The experimental stage is presented in this chapter, referring to the resources used, the dataset, the inference process of the pretrained algorithms, the training configuration of each algorithm and the inference of the trained algorithms.

4.1 Resources used

To carry out the experimental stage of this dissertation, several aspects had to be taken into consideration, such as the operating system, hardware, programming language, libraries and source codes used.

The execution environment was carried out on the Windows 11 operating system, on the author's personal computer, with the characteristics shown in Table 4.1.

CPU	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
RAM memory	16.0 GB
GPU	NVIDIA GeForce GTX 1060
Dedicated GPU memory	6.0 GB
Shared GPU memory	7.9 GB
GPU memory	13.9 GB

Table 4.1: Characteristics of the computer used.

To make the training process significantly faster for the algorithms, they were configured supporting the GPU. However, of the algorithms used, only Faster R-CNN worked with the GPU; YOLOv3 and YOLOv5l failed to recognize it. Algorithms need to be run in the same environment to compare them. Thus, the possibility of using the GPU was discarded, and they were executed with the CPU, knowing that the training time is longer.

Python, version 3.9, was used as the programming language. It is a high-level, dynamic, interpreted, modular, cross-platform and object-oriented language. This language is widely applied in areas of data analysis, research, algorithm development and AI. The following were used from the existing Python libraries:

- *Numpy*: provides functionalities for vectors and matrices, basic numerical operations, linear algebra operations, Fourier transform functions, random number features, among others, in order to work with large datasets efficiently.
- *Matplotlib*: provides functions for data visualizations, e.g., graphs and histograms, with the ability to customize the layout of each visualization and even enabling their exportation to various possible file formats.
- *Scikit-learn*: presents simple and efficient tools for predictive data analysis.
- *Pandas*: makes it possible to manipulate and analyze data, providing tools for reading and writing data between in-memory data structures and different formats, such as Comma-Separated Values (CSV) and text files, among others.
- *ElementTree*: includes tools for analyzing, creating and modifying Extensible Markup Language (XML) files.
- *Time*: enables the handling of time-related tasks.
- *Shutil*: offers various high-level operations of files and directories, e.g., file attributes, copying and removing directories, among others.
- *Glob*: used to find all file paths which correspond to a specific pattern.
- *Os*: enables interaction with the operating system's command line and works with the environment, processes, users, files and directories.
- *Argparse*: includes tools for constructing command-line arguments and option processors, facilitating the writing of command-line interfaces and generating help, usage, and error messages when the program receives invalid arguments.
- *OpenCV-Python*: designed to solve CV, ML and image processing problems.
- *PyTorch*: used to develop and train algorithms based on neural networks.
- *Tensorflow*: used to develop and train neural networks to detect and recognize patterns and correlations between data.

For the implementation of the algorithms, two different source codes were used, both aimed at the easy and efficient development, training and implementation of ML algorithms. As for the implementation of the Faster R-CNN algorithm, the TensorFlow 2 Object Detection API was used, an open-source structure based on TensorFlow. And as for the implementation of the YOLO algorithm, for both version 3 and version 5, the open source developed by Ultralytics was used. Of the five models previously presented for YOLOv5, the YOLOv5l model was used, ideal for detecting smaller objects.

The chosen algorithms were pretrained on the MS COCO dataset, which is a set of annotated images created by Microsoft to obtain the state of the art of object recognition algorithms. This dataset contains about 90 categories of objects easily recognizable by a child, such as a person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, dog, bird, chair, cake, tv, among many others [51].

4.2 Dataset

The set of images used in this dissertation was provided by the Portuguese Military Academy, consisting of two sets of images obtained by UAV, with 146 RGB images and 227 IRG images, for a total of 373 images. An RGB image (short for Red, Green, Blue) is based on an additive color system consisting of three primary colors, red, green and blue. An IRG image is an RGB system, where red prevails, giving the images a reddish appearance.

There being no further knowledge about the set of images, additional information was obtained about each type of image using its metadata (information on image creation), using an implemented Python script. From the information obtained through the metadata, as illustrated in Figure 4.1, the following stands out:

- Images were taken in January 2013; the RGBs were captured by a Canon IXUS 220HS camera and the IRGs by a Canon PowerShot ELPH 300HS camera.
- All images were saved in the Joint Photographic Experts Group (JPEG) format, with a size of 3000x4000, which corresponds to 3000 pixels per inch of height and 4000 pixels per inch of width, understood as the amount of information the image contains per unit of length, ppi (pixels per inch).
- Both types of images were captured at an image display resolution of 180.0, with an "upper left" orientation of the camera relative to the scene when the image was captured.

```
Get metadata from RGB image:
Filename                : ./images_RGB/IMG_0003_RGB.JPG
Image Size              : (4000, 3000)
Image Height            : 3000
Image Width             : 4000
Image Format            : JPEG
Image Mode              : RGB
Image is Animated       : False
Frames in Image         : 1
GPSInfo                 : 4688
ResolutionUnit          : 2
ExifOffset              : 238
ImageDescription        :
Make                    : Canon
Model                   : Canon IXUS 220HS
Orientation             : 1
DateTime                : 2013:01:08 12:44:41
YCbCrPositioning        : 2
XResolution             : 180.0
YResolution             : 180.0
```

```
Get metadata from IRG image:
Filename                : ./images_IRG/IMG_0003_IRG.JPG
Image Size              : (4000, 3000)
Image Height            : 3000
Image Width             : 4000
Image Format            : JPEG
Image Mode              : RGB
Image is Animated       : False
Frames in Image         : 1
ResolutionUnit          : 2
ExifOffset              : 240
ImageDescription        :
Make                    : Canon
Model                   : Canon PowerShot ELPH 300HS
Orientation             : 1
DateTime                : 2013:01:08 12:04:18
YCbCrPositioning        : 2
XResolution             : 180.0
YResolution             : 180.0
```

Figure 4.1: Metadata of an RGB image (left) and an IRG image (right).

To create a more effective dataset, the two sets of RGB and IRG images were joined, resulting in the PMA Dataset.

To ensure coherence, precision and quality in the final results, the dataset had to be prepared and organized, as its quality can directly influence the result of the algorithms. This process was divided into several steps: data selection, renaming, annotating, resizing, data augmentation and division of the dataset (train, validation and test).

Data selection

Of the 373 images provided, 10 images were discarded, since they were excessively unfocused and did not provide a clear idea of the contours of the objects, and thereby, did not meet the

conditions for their treatment. Thus, the set of images used consisted of 363 images.

Renaming the dataset

The name originally assigned to each image had the format "IMG_XXXX.JPG", where "XXXX" corresponds to the number assigned to the image, for example, "IMG_0135.JPG".

To work with a more composite dataset, the two sets of images, RGB and IRG, were joined into one, as previously mentioned. However, as there were RGB images with the same name as the IRG images, a Python script was implemented which renamed the images according to their "IMG_XXXX_YYY.jpg" format, where "XXXX" corresponds to the number assigned to the image and "YYY" to the image type, for example, "IMG_0135_RGB.jpg".

Annotation of the dataset

Image annotation consists of identifying, selecting and classifying a specific object in an image. This task needs to be performed with the utmost care and precision for the algorithm to recognize the objects accurately.

The PMA dataset had to be annotated for the desired classes, since the pretrained algorithms with the MS COCO dataset do not contain annotated aerial images, or any class corresponding to a building.

The tool LabelImg, written in Python, was used for the annotation of the images. It uses Qt (cross-platform framework) for its graphical interface and is executed locally. A directory, with the images to be annotated and a text file with the names of all the necessary classes written in it, had to be created to use the tool. Annotations were saved as XML files in PASCAL VOC format.

The XML file was generated for each annotated image, as shown in Figure 4.2. This file contains the values of the four coordinates of the limiting box, identified as *xmin*, *ymin*, *xmax*, and *ymax*, where the coordinate pair (*xmin*, *ymin*) corresponds to the upper left corner and the pair (*xmax*, *ymax*) corresponds to the lower right corner of the limiting box. As the original images have a resolution of 3000x4000, the minimum values for these coordinates vary between 0 and 4000 for x and between 0 and 3000 for y.

The XML file can contain zero or more annotated objects in an image, creating a new <object> element for a new object. The example in Figure 4.2 below illustrates the annotation of a dataset image which contains two annotated objects.

```xml
<annotation>
    <folder>images_IRG</folder>
    <filename>IMG_0135_IRG.jpg</filename>
    <path>C:\Users\User\Desktop\IMG_0135_IRG.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>640</width>
        <height>640</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>vehicle</name>
        <pose>Unspecified</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>182</xmin>
            <ymin>629</ymin>
            <xmax>190</xmax>
            <ymax>640</ymax>
        </bndbox>
    </object>
    <object>
        <name>vehicle</name>
        <pose>Unspecified</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>191</xmin>
            <ymin>627</ymin>
            <xmax>199</xmax>
            <ymax>640</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 4.2: Example of an XML file in a PASCAL VOC annotation format.

In the scope of this dissertation, as the objective is the detection of vehicles and buildings in aerial images, only two classes of objects were created: *vehicle* and *building*. Figure 4.3 illustrates the number of object instances of each class, a total of 11945 instances, more specifically, 9500 (79.53%) of vehicles and 2445 (20.47%) of buildings.

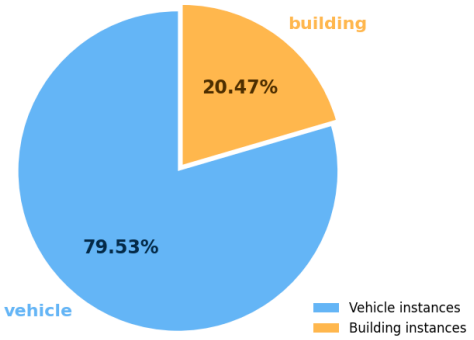


Figure 4.3: Percentage of instances of each class in the PMA dataset.

– **Vehicle annotation**

According to the Highway Code, a vehicle is comprised of a propulsion engine, equipped with at least four wheels, with a tare weight greater than 550 kg, whose maximum speed is, by construction, greater than 25 km/h, and which is intended to travel on public roads, without the use of rails [52].

Examples of vehicles in this category are passenger vehicles (vehicles with a capacity not exceeding 9 seats, including the driver's, used for the transport of people), light goods vehicles (vehicles with a capacity not exceeding 9 seats, including the driver's, used for the transport of cargo), heavy passenger vehicles (vehicles with a capacity of more than 9 seats, including the driver's, used for the transport of passengers) and, finally, heavy goods vehicles (vehicles with a gross weight of more than 3500 kg, used for the transport of cargo).

In the images from the PMA dataset, all vehicles which could be easily detected with the naked eye, which did not raise any doubts, were considered vehicles, having taken into consideration the complete objects (Figure 4.4) and those in which the intended object could be identified, even with overlapping or incomplete objects (Figure 4.5). All those causing doubt were not considered (Figure 4.6).

Figure 4.4: Examples of considered vehicles (complete objects).

Figure 4.5: Examples of considered vehicles (incomplete objects or with other overlapping objects).

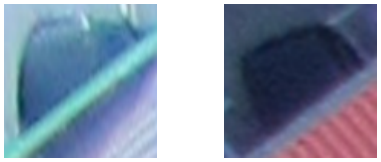


Figure 4.6: Examples of vehicles not considered.

– **Building annotation**

In an aerial view, the detection of a building involves the detection of a roof/covering. Currently, there are several types of building roofs, such as ceramic tiles, glass tiles, photovoltaic tiles, metal tiles, PCV tiles, among others.

In the images from the PMA dataset, all objects that, following the same criteria of selection of vehicles, with the naked eye, were easily recognized as a building were considered

a building. Again, in the building annotations, complete objects were taken into consideration (Figure 4.7), as well as those in which the intended object could be identified, even with overlapping or incomplete objects (Figure 4.8). All those causing doubt were not considered (Figure 4.9).

Figure 4.7: Examples of considered buildings (complete objects).

Figure 4.8: Examples of considered buildings (incomplete objects or with other overlapping objects).

Figure 4.9: Examples of buildings not considered.

Resizing the dataset

The images from the PMA dataset have a size of 3000x4000, as said before. The input image resolutions were reduced to allow for a larger batch size and to deal with computational limitations.

To standardize the input data set for the implemented algorithms, the PMA was resized to 640x640, in pixels per inch. The only difference in terms of annotations can be seen in the minimum and maximum range of the x and y coordinates, which both then vary between 0 and 640.

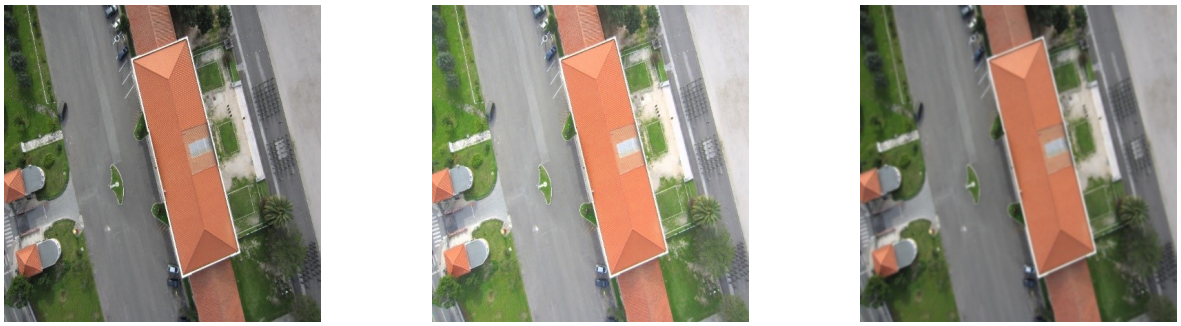
A Python script was implemented which enabled the local resizing of both the images and the respective XML annotations. This script recursively traversed the input directory, resized the image, changed the xycoordinates of each object in the XML file and saved the new files in the output directory path, making it possible to save the images with the limiting boxes drawn.

Data augmentation

Data augmentation methods were applied to the original dataset, which resulted in an increase in the data amount.

The transformations applied to each image of the PMA dataset were brightness and Gaussian blur, as illustrated in Figure 4.10. The purpose of applying these transformations to

the images was to simulate different flight conditions, different times of day (brightness effect) and flight instability when capturing the image (blur effect).



(a) (b) (c)

Figure 4.10: Example of Data Augmentation application: (a) original image; (b) image with brightness; (c) image with Gaussian blur.

The application of data augmentation resulted in a new dataset, named Portuguese Military Academy with Data Augmentation (PMA-DA), which is three times larger than the PMA dataset (original image, image transformed with the application of brightness and image transformed with the application of Gaussian blur). The new images were named according to the "IMG_XXXX_YYY_Z.jpg" format, where "XXXX" corresponds to the number assigned to the image, "YYY" to the image type, and "Z" to the name of the transformation applied to the image, for example, "IMG_0135_RGB_brightness. jpg".

Division of the dataset

The datasets, PMA and PMA-DA, were divided into three subsets: training, validation and testing. Figure 4.11 illustrates the number of instances of each class per subset, for both datasets.

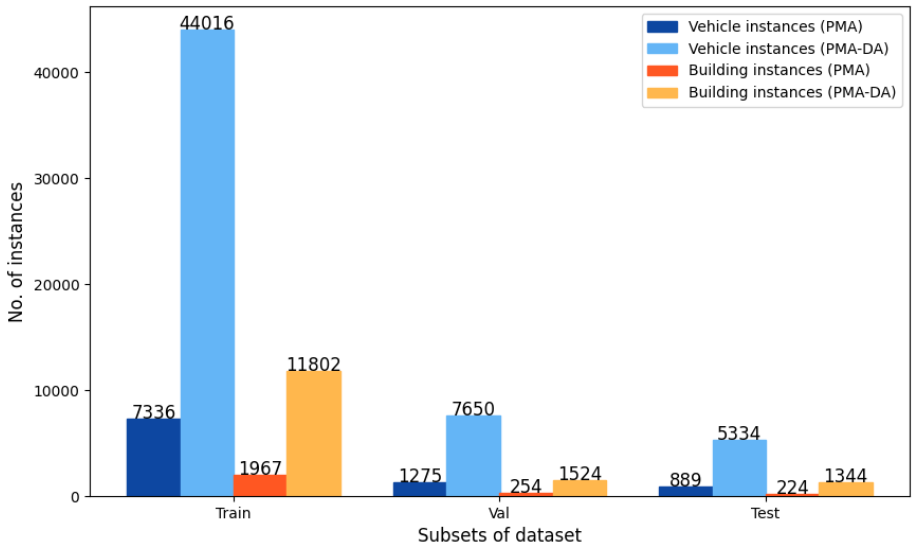


Figure 4.11: Number of instances of each class per training, validation and testing subsets of the PMA and PMA-DA datasets.

The division was executed with the aid of an implemented Python script, so that the training subset was the largest. Specifically, 80% of the images of the dataset were used for training, 10% were used for validation and the remaining 10% were used for testing. The script, in addition to dividing the dataset, also enabled the redirection of the new subsets to their respective directories (training, validation and testing). Table 4.2 presents the concrete division of images carried out for each data set.

Dataset	**Train** (80%)	**Validation** (10%)	**Test** (10%)
PMA	290	36	37
PMA-DA	870	108	111

Table 4.2: Number of images by training, validation and testing subsets of each dataset.

4.3 Inference from pretrained algorithms

To obtain conclusions about the inference of algorithms pretrained with the MS COCO dataset, a Python script was developed and executed for each algorithm, which enabled going through all the images of the PMA test subset and return information about each prediction. The script, given an input image and the algorithm to be analyzed, loads its pretrained weights and its architecture, detects the target objects present and returns the image with the final result of the predictions, with limiting boxes, classes and confidence percentages for the original image.

Initially, tests were performed to verify that the script was working correctly, using pretrained algorithms and images from the MS COCO dataset. Then, all images from the PMA test subset were used.

Since the images, used both for training and for validation and testing, are images of aerial views which may contain a considerable number of small objects, the resulting image may be difficult to read for a detailed analysis. Therefore, to enable a more accurate analysis, another Python script was implemented which, given an image of the test subset, returns the total number of predictions the algorithm should detect for each class, the total number of predictions it detected, the time of detection and the confidence percentage for each detected object, recording the information in a text file.

The inference results of the pretrained algorithms can be found in a Table 1 in Appendix A - Detection results from pretrained algorithms. For each image of the test subset, the number of vehicles (V_T) and buildings (B_T) the algorithm should detect was noted. All images were analyzed using the three algorithms, where the detection time in milliseconds (T), the number of vehicles (V) and the number of buildings detected (B) were recorded and the following was observed:

- None of the algorithms are prepared to recognize a building, which is to be expected, since they were trained with the MS COCO dataset, which does not contain any category of buildings.
- Algorithms should be able to detect vehicles, as the MS COCO dataset has been trained to detect categories integrated in the vehicle definition (e.g., trucks and vehicles).

However, the performance of the algorithms was quite low, possibly because the dataset images are not aerial. Comparing the algorithms, YOLOv3 and YOLOv5l stand out compared to the Faster R-CNN algorithm because it managed to detect some vehicles.

- The YOLOv5l algorithm has a lower detection time, approximately 1.66 seconds; therefore, it is the fastest algorithm. On the other hand, Faster R-CNN and YOLOv3 have a similar detection time of approximately 2 seconds, although YOLOv3 is slightly faster than Faster R-CNN.

As an example of detection result visualization, two images belonging to the PMA test subset were randomly chosen, one RGB (left) and another IRG (right) images, illustrated in Figure 4.12.

By visualizing the images in Figure 4.12, it was found that the algorithms detected few objects. Faster R-CNN and YOLOv3 exhibit multiple False Positives (FP) compared to YOLOv5l. Faster R-CNN was only able to detect a single vehicle, exhibiting a vast number of False Negatives (FN). YOLOv3 was able to detect several vehicles, as was YOLOv5l. Another observation is related to the size of the objects. The smaller the object is, the worse the performance of the algorithms, as can be seen in the images on the right, where the vehicles are smaller compared to the images on the left.

After having made these observations, it was concluded that the Faster R-CNN, YOLOv3 and YOLOv5l algorithms, pretrained with the MS COCO dataset, are not prepared for the PMA dataset.

Figure 4.12: Detection results of the pretrained algorithms for an RGB image (left) and an IRG image (right): (a) original annotations; (b) Faster R-CNN; (c) YOLOv3 and (d) YOLOv5l.

The training of any algorithm involves processing the input data, setting the training parameters and, finally, training the algorithm itself.

Because the structure of the input data differs for each algorithm, it was necessary to process the PMA and PMA-DA datasets.

The Faster R-CNN algorithm, implemented with the TensorFlow 2 Object Detection API, receives two TFRecord files as input data, one corresponding to the training data and the other to the validation data, which store the data as a sequence of binary strings. Creating TFRecord files involves two steps: converting XML files into CSV files and converting CSV files into TFRecord files.

The XML annotations needed to be converted into three CSV files, corresponding to the training, validation and test, with the aid of an implemented Python script. For each annotation, a new line was added to the CSV file, with the format [*filename*, *width*, *height*, *class*, *xmin*, *ymin*, *xmax*, *ymax*], as shown in Figure 4.13. The *filename* field, as the name implies, corresponds to the name of the file, more precisely, the name of the image with the extension jpg. The *width* and *height* fields correspond to the image's size. The *class* field corresponds to the object's class. The remaining fields correspond to the coordinates (*xmin*, *ymin*) of the upper left corner and the coordinates (*xmax*, *ymax*) of the lower right corner of the limiting box.

```
filename,width,height,class,xmin,ymin,xmax,ymax
IMG_0135_IRG.jpg,640,640,vehicle,182,629,190,640
IMG_0135_IRG.jpg,640,640,vehicle,191,627,199,640
IMG_0135_IRG.jpg,640,640,vehicle,200,620,210,640
IMG_0135_IRG.jpg,640,640,vehicle,209,619,219,640
IMG_0135_IRG.jpg,640,640,vehicle,217,616,226,636
IMG_0135_IRG.jpg,640,640,vehicle,227,612,237,633
IMG_0135_IRG.jpg,640,640,vehicle,236,609,246,631
IMG_0135_IRG.jpg,640,640,vehicle,245,606,255,630
IMG_0135_IRG.jpg,640,640,vehicle,254,603,264,625
IMG_0135_IRG.jpg,640,640,vehicle,272,596,281,617
IMG_0135_IRG.jpg,640,640,vehicle,282,596,292,620
IMG_0135_IRG.jpg,640,640,vehicle,290,591,300,614
IMG_0135_IRG.jpg,640,640,vehicle,299,586,310,609
IMG_0135_IRG.jpg,640,640,vehicle,308,584,318,604
IMG_0135_IRG.jpg,640,640,vehicle,346,629,354,640
IMG_0135_IRG.jpg,640,640,vehicle,423,621,440,635
IMG_0135_IRG.jpg,640,640,vehicle,466,605,483,619
IMG_0135_IRG.jpg,640,640,vehicle,486,624,503,640
```

Figure 4.13: Example of an XML file converted into CSV.

After obtaining the three CSV files, these were converted into TFRecord files, with the help of an existing Python script from the TensorFlow 2 Object Detection API.

The YOLOv3 and YOLOv5l algorithms receive the same input data structure, so this step is similar for both. To obtain the input format accepted by the algorithms, the XML files were converted into text files with the following format: [*classID xCenter yCenter width height*], as illustrated in Figure 4.14. The *classID* field corresponds to the class index. The fields,

xCenter and *yCenter*, correspond to the coordinates of the center of the limiting box. The *width* and *height* fields correspond to the width and height of the limiting box, respectively. Both the coordinates and the limiting box width and height are normalized.

```
0 0.567969 0.990625 0.039062 0.018750
0 0.535938 0.875000 0.043750 0.031250
0 0.534375 0.832813 0.037500 0.031250
0 0.184375 0.333594 0.031250 0.060938
0 0.375781 0.153125 0.035938 0.043750
0 0.354688 0.064063 0.028125 0.040625
0 0.338281 0.020313 0.039062 0.040625
1 0.464063 0.080469 0.165625 0.160938
1 0.572656 0.478906 0.335938 0.751563
1 0.671875 0.896875 0.181250 0.206250
1 0.035156 0.571094 0.070312 0.114063
1 0.078906 0.807813 0.085938 0.115625
```

Figure 4.14: Example of an XML file converted yo text.

Then the training parameters were configured for each algorithm. The batch size, learning rate, IoU limit and confidence limit parameters were defined as constants, corresponding to the values indicated in Table 4.3.

Parameters	**Value**
Batch size	2
Learning rate	0.013
IoU limit	0.5
Confidence limit	0.3

Table 4.3: Values of training parameters.

The epochs parameter used in the YOLO algorithms was set to two values, 100 and 200, for the PMA dataset and 100 epochs for the PMA-DA dataset. For the Faster R-CNN algorithm, a relationship had to be established between the number of epochs and the number of steps, so that the number of steps corresponded to 100 and 200 epochs.

In the PMA-DA dataset, the algorithms could only be trained for 100 epochs, since the training time was too long for 200 epochs (1 epoch $\approx$ 1h), becoming excessive for the computer and causing overheating.

4.5 Inference from trained algorithms

The Faster R-CNN algorithm was trained for 14500 and 29000 steps, equivalent to 100 and 200 epochs, respectively, with the PMA dataset. The algorithm was also trained for 43500 steps, equivalent to 100 epochs with the increase in the dataset for the PMA-DA dataset. The results obtained are recorded in Table 4.4.

The analysis of Table 4.4 revealed that the Faster R-CNN training time was fast for any number of steps. In both classes, vehicles and buildings, the Precision value was high (above 91%) and the Recall value was low (less than 50%), which indicates that most objects were correctly detected, but with detection losses. It was also noted that the Recall in the building

class was slightly lower than in the vehicle class. Comparing the results of 14500 steps PMA and 43500 steps PMA-DA, it was observed that without data augmentation the algorithm obtained a superior performance. The algorithm showed better results for 14500 steps with the PMA dataset, with a mAP equal to 58.1%.

		PMA Dataset		**PMA-DA Dataset**
		14500 steps	29000 steps	43500 steps
training time	(hours)	8	16	25
mAP	(all)	58.1%	56.2%	52.5%
Precision	(vehicle)	91.9%	93.7%	94.8%
Recall		50.4%	47.9%	58.2%
Precision	(building)	98.9%	97.7%	97.2%
Recall		47.0%	44.5%	50.1%

Table 4.4: Faster R-CNN algorithm training results.

Comparing the confusion matrices generated by the algorithm (Figure 4.15), it was found that the training of 14500 steps with the PMA dataset had the lowest detection confusion in the building class, no detection confusion in the vehicle class and the lowest percentage of false negatives for the building class. The training of 29000 steps with the PMA dataset had the highest detection confusion in both classes, the highest percentage of false positives and the lowest percentage of true positives. Although the training of 43500 steps with the PMA-DA dataset achieved the best percentage of true positives and false positives, it had a high percentage of false negatives and detection confusion in the building class.

(a) **(b)** **(c)**

Figure 4.15: Faster R-CNN Confusion Matrix for: (a) 14500 steps (PMA); (b) 29000 steps (PMA) and (c) 43500 steps (PMA-DA).

The results of training the YOLOv3 algorithm, for 100 and 200 epochs with the PMA dataset and for 100 epochs with the PMA-DA dataset, were recorded in Table 4.5.

The results recorded in Table 4.5 revealed that the algorithm training time was slow for any epoch value. Precision and Recall values, for both classes, were high, above 91% and 76%, respectively, which indicates most objects were correctly detected. As for Recall in the building class, it was slightly lower than in the vehicle class. It was noted that training the algorithm with the PMA-DA dataset, compared to the PMA dataset, for 100 epochs

presented worse performance. The best results of the YOLOv3 algorithm were achieved in the training of 200 epochs, with a mAP equal to 92.2%.

		PMA Dataset		PMA-DA Dataset
		100 epochs	200 epochs	100 epochs
training time	(hours)	30	60	127
mAP	(all)	91.4%	92.2%	88.7%
Precision	(vehicle)	94.1%	93.2%	91.9%
Recall		91.1%	92.3%	88.9%
Precision	(building)	93.7%	94.5%	97.1%
Recall		82.3%	85.4%	76.0%

Table 4.5: YOLOv3 algorithm training results.

Analyzing the confusion matrices generated by the algorithm (Figure 4.16) revealed that the training results of 100 epochs with the PMA dataset were similar to those of 200 epochs, differing in the percentage of true positives in the building class, which was lower, in the increase in the percentage of false negatives and in the increase in detection confusion in the building class. The training of 200 epochs with the PMA dataset was the one that achieved the best percentage of true positives, no detection confusion and the lowest percentage of false negatives in the building class. The training of 100 epochs with the PMA-DA dataset demonstrated the lowest percentage of true positives, the highest detection confusion in the building class, the highest percentage of false negatives in both classes and the highest percentage of false positives in the vehicle class.

(a) **(b)** **(c)**

Figure 4.16: YOLOv3 Confusion Matrix for: (a) 100 epochs (PMA); (b) 200 epochs (PMA) and (c) 100 epochs (PMA-DA).

Relative to the YOLOv5l algorithm, it was also trained for 100 and 200 epochs with the PMA dataset and for 100 epochs with the PMA-DA dataset, with the results recorded in Table 4.6.

Through Table 4.6, it was observed that the algorithm training time was relatively fast for any epoch value. In both classes, the Precision and Recall values were high, above 92% and above 81%, respectively, which means that most objects were correctly detected. The Recall value in the building class was slightly lower than in the vehicle class. When comparing the training performance of 100 epochs for the PMA and PMA-DA datasets, it was noticed that

the algorithm performed better without data augmentation. The best mAP value obtained by the algorithm was 93.3% for the training of 100 epochs with the PMA dataset.

		PMA Dataset		PMA-DA Dataset
		100 epochs	200 epochs	100 epochs
training time	(hours)	24	48	73
mAP	(all)	93.3%	91.8%	91.4%
Precision	(vehicle)	95.4%	92.8%	93.4%
Recall		90.8%	91.6%	91.4%
Precision	(building)	96.0%	97.2%	96.4%
Recall		85.4%	91.5%	81.6%

Table 4.6: YOLOv5l algorithm training results.

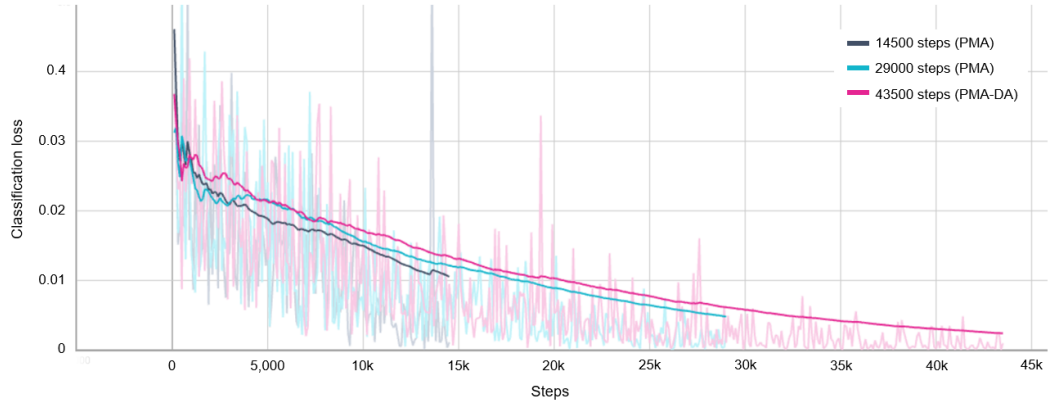
Observing the YOLOv5l confusion matrices (Figure 4.17), it was determined that the results of the training of 100 epochs with the PMA dataset achieved the highest percentage of true positives in the building class, the lowest percentage of false positives in the vehicle class, the lowest percentage of false negatives in the building class, the highest percentage of false positives in the building class and the highest detection confusion in the building class. The training of 200 epochs with the PMA dataset showed the highest percentage of true positives in the vehicle class. However, it obtained the lowest percentage of true positives in the building class and the lowest percentage of false negatives in the vehicle class. Training of 100 epochs with the PMA-DA dataset showed similarities to the training of 100 epochs with the PMA dataset in terms of the percentage of true positives and false negatives in the vehicle class. It also showed similarities to the training of 200 epochs with the PMA dataset regarding the percentage of false positives in both classes, the percentage of false negatives in the building class and the detection confusion which was null in both classes.

(a) **(b)** **(c)**

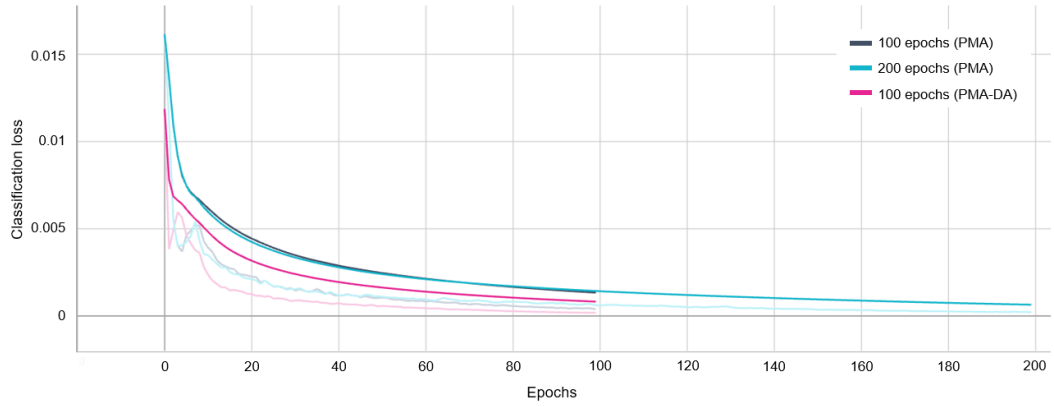
Figure 4.17: YOLOv5l Confusion Matrix for: (a) 100 epochs (PMA); (b) 200 epochs (PMA) and (c) 100 epochs (PMA-DA).

Figure 4.18 presents the classification loss graphs for each algorithm. In general, at the beginning of training, it was found that classification loss was quite high, followed by an abrupt decrease, which tends to occur gradually, with a tendency of approaching zero classification losses as the number of steps/epochs increases. Regarding the behavior of the Faster R-CNN algorithm, it was observed that there was significant oscillation, which led

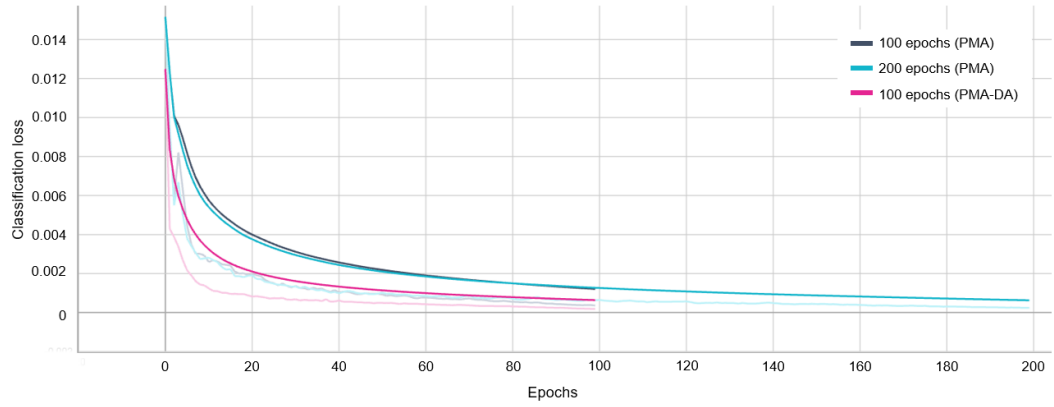
to a greater loss of performance relative to the other algorithms. Unlike YOLO algorithms, Faster R-CNN showed a higher classification loss for dataset growth. The YOLO algorithms achieved a much lower classification loss than the Faster R-CNN algorithm.



(a)



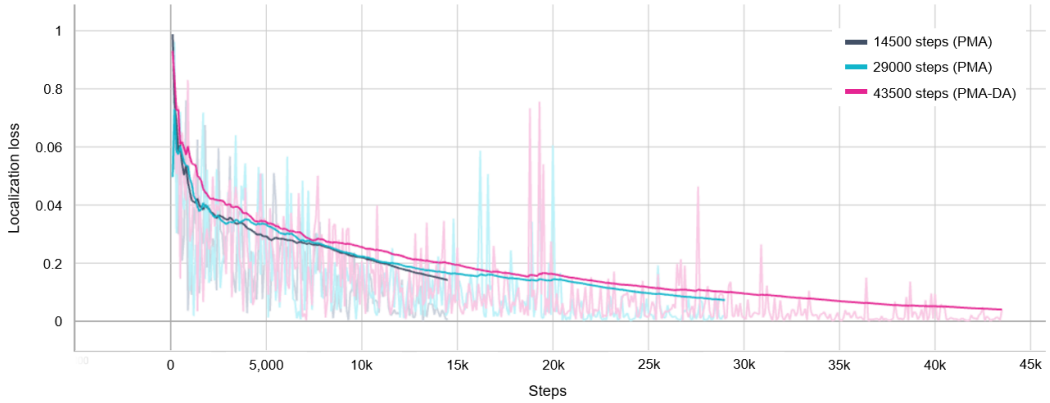
(b)



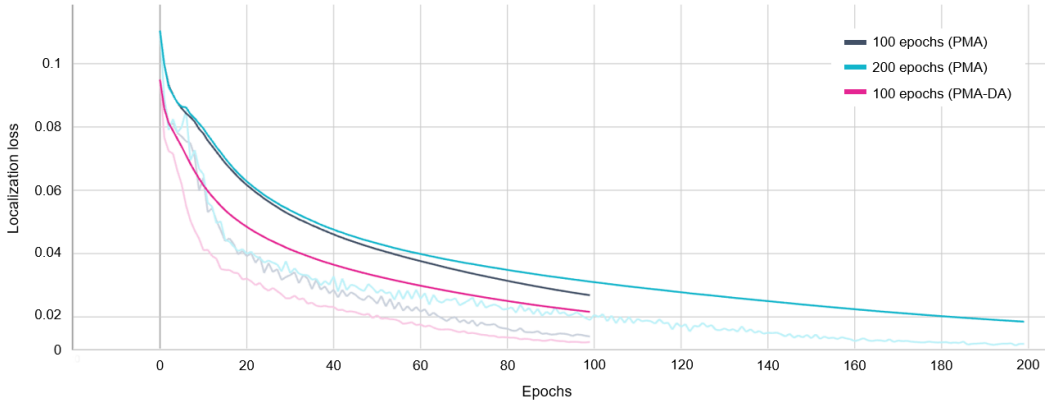
(c)

Figure 4.18: Classification loss graph of the algorithms: (a) Faster R-CNN, (b) YOLOv3 and (c) YOLOv5l.

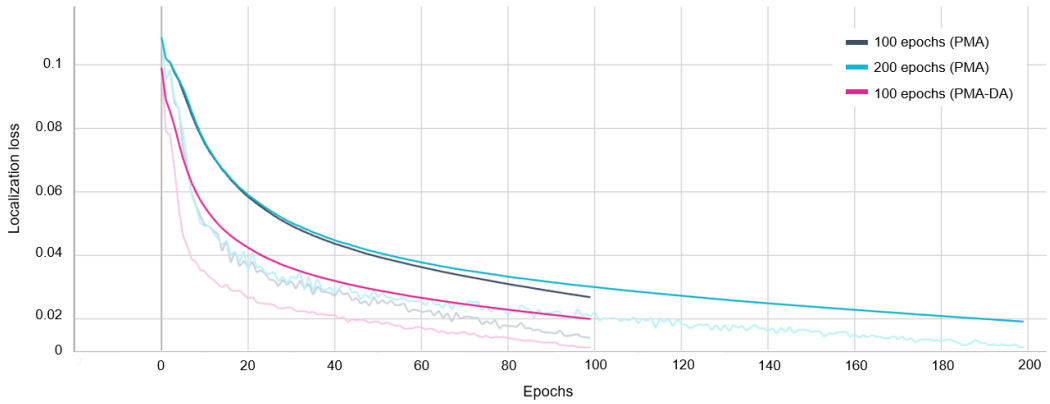
Figure 4.19 presents the localization loss graphs for each algorithm. The localization loss behavior of algorithms is similar to classification loss. However, it was found that there was more localization loss than classification loss for all algorithms.



(a)



(b)



(c)

Figure 4.19: Localization loss graph of the algorithms: (a) Faster R-CNN, (b) YOLOv3 and (c) YOLOv5l.

Like the inference step of the pretrained algorithms, a Python script was implemented, which went through all the test subset images and returned information about each prediction. This script, when inserting an input image, loads the trained weights, detects the target objects present and returns the image with the final result of the predictions, with limiting boxes, classes and confidence percentages on the original image.

As for the detection results, these were recorded in Tables 2, 3 and 4 found in Appendix B - Detection results for Faster R-CNN algorithm, Appendix C - Detection results for YOLOv3 algorithm and Appendix D - Detection results for YOLOv5l algorithm, corresponding to the Faster R-CNN, YOLOv3 and YOLOv5l algorithms, respectively. For each image of the test subset, the number of vehicles (V_T) and buildings (B_T) the algorithm should detect, the detection time in milliseconds (T), the number of vehicles (V) and the number of buildings (B) detected were noted. The main observation of these records refers to the detection time, which varies from 1450 to 2291 milliseconds for the Faster R-CNN algorithm, from 999 to 1592 milliseconds for the YOLOv3 algorithm and from 716 to 1439 milliseconds for the YOLOv5l algorithm. It was also found that the detection time is not affected by the number of objects in an image, nor with the number of steps/epochs.

Figure 4.20 illustrates the detection results of two images, RGB on the left and IRG on the right for each algorithm. To visualize the results without overlapping the classes assigned to each limiting box, vehicles were identified in blue and buildings in yellow. Through a direct observation of the images, relative to the Faster R-CNN, the presence of several false negatives and the existence of limiting boxes with a localization deficit were obvious. For example, some boxes did not include the object in its entirety. The YOLOv3 algorithm presented better detections compared to Faster R-CNN, being able to detect practically all target objects. However, it still had some problems with false positives and negatives. As for the YOLOv5l algorithm, it presented the best detection results, mainly due to the reduction of false negatives. It should be noted that the problem of the object size, which occurred in the inference stage of the pretrained algorithms, was overcome after the algorithm training process, being then able to detect the smallest objects.

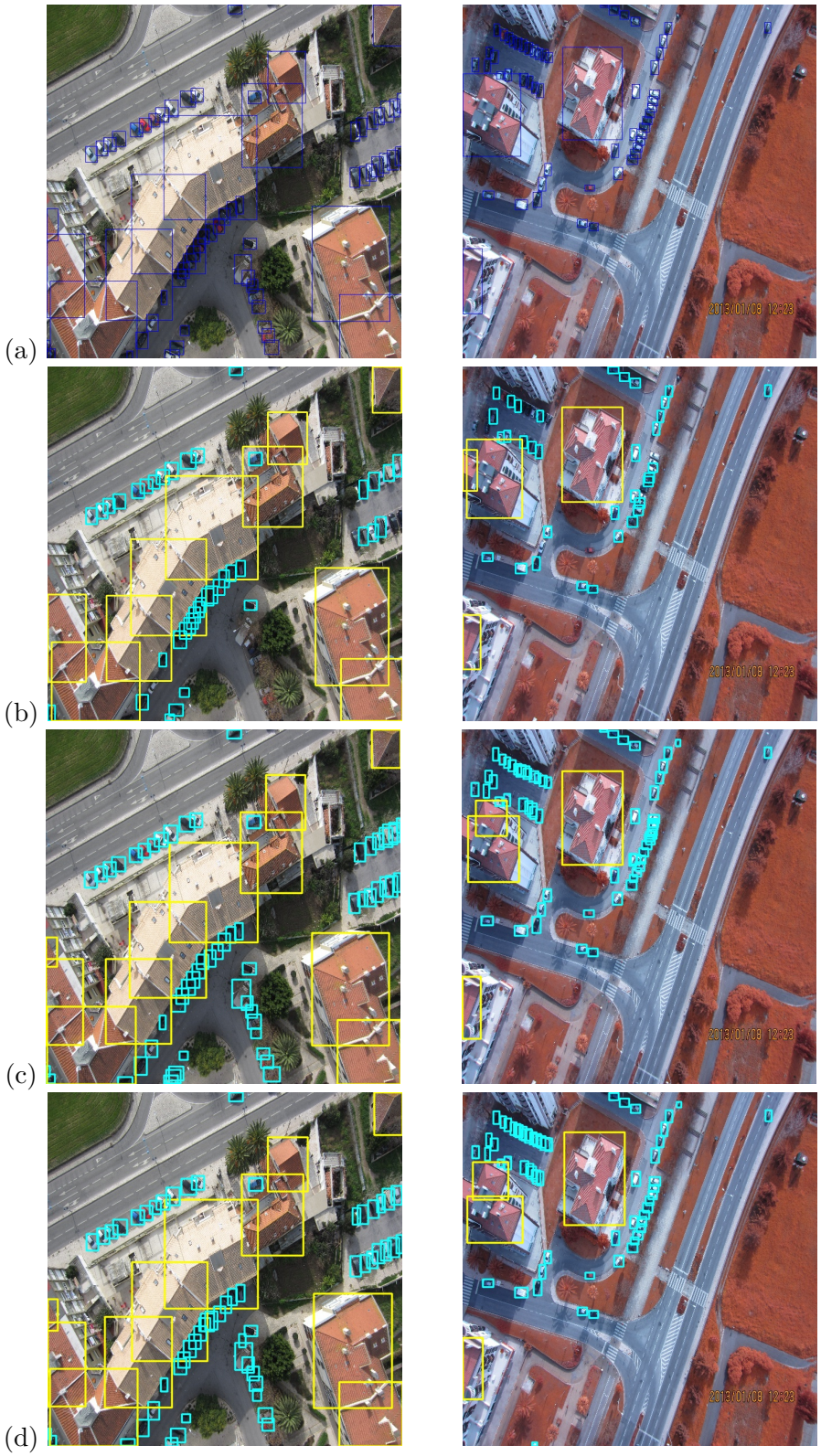


Figure 4.20: Detection results of the trained algorithms, for an RGB image (left) and an IRG image (right): (a) original annotations; (b) Faster R-CNN; (c) YOLOv3 and (d) YOLOv5l.

CHAPTER 5

Conclusions

The main conclusions of the work and some possible suggestions for future work are presented in this chapter.

Human beings have long been developing technologies to facilitate and assist everyday tasks. There has currently been a great interest in the area of aerial image processing, mainly in object detection tasks, which is useful for several applications, such as in surveillance, security and military areas.

The purpose of object detection algorithms is to learn to predict annotations similar to real ones, both in terms of the object class and the size of the limiting boxes. In this dissertation, object detection algorithms were implemented, specifically, Faster R-CNN, a two-stage detector and YOLOv3 and YOLOv5l, one-stage detectors.

The adopted methodology was based on several steps. These included understanding the problem, with the defining of objectives, a brief literature review and choice of algorithms; data preparation, with image selection, renaming, annotation, resizing, data augmentation and division of the PMA and PMA-DA datasets; the inference of pretrained algorithms with the MS COCO dataset, according to the TL pretrained network as a classifier approach, where predictions and performance evaluations of the algorithms were made in terms of detection time; the training of the algorithms according to the pretrained network as a feature extractor approach of TL, where it was necessary to process the input data and configure the training parameters; and, finally, the inference of the trained algorithms, where predictions and evaluations of the performance of the algorithms were also carried out, in terms of training time, evaluation metrics, loss of classification and location and detection time.

Based on the results obtained in the inference stage of the trained algorithms, it was found that the data augmentation technique did not produce improvements in the quality of the algorithms, possibly because there are new added difficulties and the algorithms would need more training time to learn; the number of object instances in an image and the number of epochs or steps did not influence the detection time; the building class showed a higher

Precision than the vehicle class, indicating that the algorithms are more accurate in detecting larger objects; the building class showed a lower Recall than the vehicle class, probably because there are fewer instances of this class in the PMA and PMA-DA data sets; relative to training time, the Faster R-CNN algorithm took the least time, followed by YOLOv5l and, finally, YOLOv3; the Faster R-CNN algorithm was the least accurate and had the longest detection time, its training time being its only advantage; the YOLOv3 algorithm, despite presenting a relatively fast detection time and a performance exceeding that of Faster R-CNN, could not surpass YOLOv5l; the YOLOv5l algorithm was the one which achieved the best results, with the best mAP of 93.3%, the highest percentage of Precision and Recall, as well as standing out for having a faster detection time.

Thus, it was concluded that the one-stage detector, YOLOv5l, presented the best performance, proving to be efficient for the detection of vehicles and buildings in aerial images.

5.1 Future work

In terms of future work, it would be advantageous to train, analyze and evaluate YOLOv5, to improve the performance of the algorithm which presented better results in this dissertation, using the following approaches:

- Expand the dataset, with images from other environments and locations, to improve the accuracy and generalization of the algorithm.
- Train the YOLOv5 algorithm on the YOLOv5m model, which is suitable for large amounts of training data.
- Adjust the hyperparameters, finding the best value for each one, to gradually improve the algorithm's performance.

References

[1] A. Ammar, A. Koubaa, M. Ahmed, A. Saad, and B. Benjdira, "Vehicle detection from aerial images using deep learning: A comparative study," *Electronics*, vol. 10, no. 7, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10070820. [Online]. Available: https://www.mdpi.com/2079-9292/10/7/820.

[2] S. Srivastava, S. Narayan, and S. Mittal, "A survey of deep learning techniques for vehicle detection from uav images," *Journal of Systems Architecture*, vol. 117, p. 102 152, 2021, ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2021.102152. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001107.

[3] F. Kamran, M. Shahzad, and F. Shafait, "Automated military vehicle detection from low-altitude aerial images," in *2018 Digital Image Computing: Techniques and Applications (DICTA)*, 2018, pp. 1–8. DOI: 10.1109/DICTA.2018.8615865.

[4] X. Xie and G. Lu, "A research of object detection on uavs aerial images," in *2021 2nd International Conference on Big Data Artificial Intelligence Software Engineering (ICBASE)*, 2021, pp. 342–345. DOI: 10.1109/ICBASE53849.2021.00070.

[5] S. J. Prince, *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, Aug. 2012, p. 2, ISBN: 9781107011793.

[6] T. Taulli, "Artificial intelligence basics: A non-technical introduction," in Apress, Aug. 2019, ch. 1,4, ISBN: 9781484250280.

[7] W. Ertel, *Introduction to Artificial Intelligence*, 2nd ed. Springer, 2017, p. 1, ISBN: 9783319584867. DOI: 10.1007/978-3-319-58487-4.

[8] E. Costa and A. Simões, *Inteligência Artificial: Fundamentos e Aplicações*, 3rd ed. FCA, Mar. 2011, p. 3, ISBN: 9789727223404.

[9] P. Ongsulee, "Artificial intelligence, machine learning and deep learning," in *2017 15th International Conference on ICT and Knowledge Engineering (ICT KE)*, 2017, pp. 1–6. DOI: 10.1109/ICTKE.2017.8259629.

[10] T. Markiewicz and J. Zheng, "Getting started with artificial intelligence," in 2nd ed. O'Reilly Media, Inc., Oct. 2020, ch. 1, ISBN: 9781492083436.

[11] *What is ai?* [Online]. Available: https://www.resquared.com/blog/what-is-ai.

[12] A. Géron, "Hands-on machine learning with scikit-learn, keras, and tensorflow," in 2nd ed. O'Reilly Media, Inc., Sep. 2019, ch. 1,10,14, ISBN: 9781492032649.

[13] M. Dol and A. Geetha, "A learning transition from machine learning to deep learning: A survey," in *2021 International Conference on Emerging Techniques in Computational Intelligence (ICETCI)*, 2021, pp. 89–94. DOI: 10.1109/ICETCI51973.2021.9574066.

[14] *Artificial intelligence vs. machine learning vs. deep learning: Essentials*, Last accessed 21 May 2022, Apr. 2020. [Online]. Available: https://serokell.io/blog/ai-ml-dl-difference.

[15] *About train, validation and test sets in machine learning*, Last accessed 21 May 2022, Dec. 2017. [Online]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

[16] M. Mohammed, M. B. Khan, and E. M. Bashier, "Machine learning: Algorithms and applications," in CRC Press, Aug. 2016, ch. 1, ISBN: 9781315354415.

[17] *What is deep learning?* [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html.

[18] F. Chollet, *Deep Learning with Python*. Manning Publications Co., 2018, p. 8, ISBN: 9781617294433.

[19] R. Um and X. Zeng, "A review of deep learning research," 4, vol. 13, Apr. 2019. DOI: 10.3837/tiis.2019.04.001. [Online]. Available: https://doi.org/10.3837/tiis.2019.04.001.

[20] A. Bouguettaya, H. Zarzour, A. Kechida, and A. M. Taberkit, "Vehicle detection from uav imagery with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021. DOI: 10.1109/TNNLS.2021.3080276.

[21] H. Eriş and U. Çevik, "Implementation of target tracking methods on images taken from unmanned aerial vehicles," in *2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2019, pp. 311–316. DOI: 10.1109/SAMI.2019.8782768.

[22] *What is a convolutional neural network?* [Online]. Available: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[23] M. A. Wani, F. A. Bhat, S. Afzal, and A. I. Khan, *Advances in Deep Learning*. Springer, 2020, vol. 57, p. 51, ISBN: 9789811367939. [Online]. Available: https://doi.org/10.1007/978-981-13-6794-6.

[24] M. Elgendy, "Deep learning for vision systems," in Manning Publications, Nov. 2020, ch. 6.

[25] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611–629, Aug. 2018. DOI: 10.1007/s13244-018-0639-9. [Online]. Available: https://doi.org/10.1007/s13244-018-0639-9.

[26] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *CoRR*, vol. abs/1905.05055, 2019. arXiv: 1905.05055. [Online]. Available: http://arxiv.org/abs/1905.05055.

[27] I. V. Saetchnikov, E. A. Tcherniavskaia, and V. V. Skakun, "Object detection for unmanned aerial vehicle camera via convolutional neural networks," *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 2, no. 2, pp. 98–103, 2021. DOI: 10.1109/JMASS.2020.3040976.

[28] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. arXiv: 1504.08083. [Online]. Available: http://arxiv.org/abs/1504.08083.

[29] A. K. Shetty, I. Saha, R. M. Sanghvi, S. A. Save, and Y. J. Patel, "A review: Object detection models," in *2021 6th International Conference for Convergence in Technology (I2CT)*, 2021, pp. 1–8. DOI: 10.1109/I2CT51068.2021.9417895.

[30] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497.

[31] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[32] A. Ismail, M. Mehri, A. Sahbani, and N. ESSOUKRI BEN AMARA, "Performance benchmarking of yolo architectures for vehicle license plate detection from real-time videos captured by a mobile robot," Jan. 2021, pp. 661–668. DOI: 10.5220/0010349106610668.

[33] F. Lin, X. Zheng, and Q. Wu, "Small object detection in aerial view based on improved yolov3 neural network," in *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA)*, 2020, pp. 522–525. DOI: 10.1109/AEECA49918.2020.9213538.

[34] W. Ding and L. Zhang, "Building detection in remote sensing image based on improved yolov5," in *2021 17th International Conference on Computational Intelligence and Security (CIS)*, 2021, pp. 133–136. DOI: 10.1109/CIS54983.2021.00036.

[35] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. arXiv: 1612.08242. [Online]. Available: http://arxiv.org/abs/1612.08242.

[36] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. arXiv: 2004.10934. [Online]. Available: https://arxiv.org/abs/2004.10934.

[37] U. Nepal and H. Eslamiat, "Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs," *Sensors*, vol. 22, no. 2, 2022, ISSN: 1424-8220. DOI: 10.3390/s22020464. [Online]. Available: https://www.mdpi.com/1424-8220/22/2/464.

[38] *Custom object detection training using yolov5*, Last accessed 21 May 2022, Apr. 2022. [Online]. Available: https://learnopencv.com/custom-object-detection-training-using-yolov5/#What-is-YOLOv5.

[39] L. Wang, J. Liao, and C. Xu, "Vehicle detection based on drone images with the improved faster r-cnn," pp. 466–471, Feb. 2019. DOI: https://doi.org/10.1145/3318299.3318383. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3318299.3318383.

[40] Y. Xu, G. Yu, Y. Wang, X. Wu, and Y. Ma, "Car detection from low-altitude uav imagery with the faster r-cnn," *Journal of Advanced Transportation*, vol. 2017, 2017, ISSN: 0197-6729. DOI: https://doi.org/10.1155/2017/2823617. [Online]. Available: https://www.hindawi.com/journals/jat/2017/2823617/.

[41] L. Zheng, P. Ai, and Y. Wu, "Building recognition of uav remote sensing images by deep learning," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 1185–1188. DOI: 10.1109/IGARSS39084.2020.9323322.

[42] H. Zhang, M. Sun, Y. Ji, S. Xu, and W. Cao, "Learning-based object detection in high resolution uav images: An empirical study," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 886–889. DOI: 10.1109/INDIN41052.2019.8972320.

[43] H. S. DIKBAYIR and H. İbrahim BÜLBÜL, "Deep learning based vehicle detection from aerial images," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 956–960. DOI: 10.1109/ICMLA51294.2020.00155.

[44] H. Xu, Y. Cao, Q. Lu, and Q. Yang, "Performance comparison of small object detection algorithms of uav based aerial images," in *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, 2020, pp. 16–19. DOI: 10.1109/DCABES50732.2020.00014.

[45] X. Yin, Y. Yang, H. Xu, W. Li, and J. Deng, "Enhanced faster-rcnn algorithm for object detection in aerial images," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9, 2020, pp. 2355–2358. DOI: 10.1109/ITAIC49862.2020.9339038.

[46] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation," in *AI 2006: Advances in Artificial Intelligence*, A. Sattar and B.-h. Kang, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1015–1021, ISBN: 978-3-540-49788-2.

[47] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[48] F. Ajmera, S. Meshram, S. Nemade, and V. Gaikwad, "Survey on object detection in aerial imagery," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 2021, pp. 1050–1055. DOI: 10.1109/ICICV50876.2021.9388517.

[49] *Evaluating object detection models using mean average precision (map)*. [Online]. Available: https://blog.paperspace.com/mean-average-precision/.

[50] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery : A small target detection benchmark," *Journal of Visual Communication and Image Representation*, vol. 34, pp. 187–203, 2016, ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2015.11.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320315002187.

[51] T. Lin, M. Maire, S. J. Belongie, *et al.*, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[52] V. Monteiro, *O Código da Estrada*, 60th ed. Edições Segurança Rodoviária, pp. 285, 286, ISBN: 9789895469321.

Appendix A - Detection results from pretrained algorithms

	Theoretical Detections		Faster R-CNN			YOLOv3			YOLOv5l		
Image	V_T	B_T	T	V	B	T	V	B	T	V	B
IMG_0003_RGB.jpg	56	11	8773.0	1	0	2046.0	9	0	1552.0	7	0
IMG_0015_IRG.jpg	0	2	1905.0	0	0	2083.0	0	0	1613.6	0	0
IMG_0017_RGB.jpg	1	4	1897.0	0	0	1891.0	0	0	1838.0	0	0
IMG_0025_IRG.jpg	0	0	1887.0	0	0	2011.0	0	0	1623.0	0	0
IMG_0025_RGB.jpg	0	2	1848.0	0	0	2420.0	0	0	1598.0	0	0
IMG_0030_IRG.jpg	0	0	1799.0	0	0	1905.0	0	0	1559.0	0	0
IMG_0030_RGB.jpg	0	0	1755.0	0	0	1862.0	0	0	1604.5	0	0
IMG_0031_IRG.jpg	0	0	1747.0	0	0	1904.0	0	0	1601.5	0	0
IMG_0038_RGB.jpg	54	26	1758.0	1	0	1878.0	13	0	1901.0	8	0
IMG_0039_IRG.jpg	30	8	1906.0	0	0	2059.0	0	0	1664.0	0	0
IMG_0041_IRG.jpg	136	25	1860.0	0	0	1954.0	0	0	1596.5	0	0
IMG_0049_RGB.jpg	0	1	1963.0	0	0	1864.0	0	0	1618.0	0	0
IMG_0050_IRG.jpg	2	6	1963.0	0	0	1864.0	0	0	1618.0	0	0
IMG_0053_RGB.jpg	16	4	1872.0	0	0	1867.0	0	0	1602.0	0	0
IMG_0057_IRG.jpg	70	2	1744.0	0	0	1914.0	0	0	1873.0	0	0
IMG_0057_RGB.jpg	4	3	1787.0	1	0	2145.0	0	0	1716.0	0	0
IMG_0060_IRG.jpg	0	0	1780.0	0	0	1923.0	0	0	1589.0	0	0
IMG_0062_RGB.jpg	0	0	1836.0	0	0	1862.0	0	0	1586.0	0	0
IMG_0066_IRG.jpg	14	3	1794.0	0	0	1902.0	0	0	1590.0	0	0
IMG_0077_IRG.jpg	0	3	1737.0	0	0	1834.0	0	0	1672.0	0	0
IMG_0079_RGB.jpg	6	2	1772.0	0	0	2060.0	0	0	1778.0	0	0
IMG_0084_RGB.jpg	4	3	1697.0	0	0	2131.0	4	0	1758.0	4	0
IMG_0089_RGB.jpg	42	3	1862.0	0	0	1906.0	0	0	1580.0	0	0
IMG_0094_IRG.jpg	6	5	1802.0	0	0	1922.0	0	0	1611.0	0	0
IMG_0100_IRG.jpg	0	0	1792.0	0	1	2173.0	0	0	1601.0	0	0
IMG_0108_RGB.jpg	0	0	1787.0	0	0	2101.0	0	0	1661.0	0	0
IMG_0120_IRG.jpg	101	48	1951.0	0	0	2393.0	1	0	1762.0	18	0
IMG_0120_RGB.jpg	0	6	2007.0	0	0	2803.0	0	0	1888.0	0	0
IMG_0124_IRG.jpg	8	7	1862.0	0	0	1916.0	0	0	1580.0	0	0
IMG_0131_IRG.jpg	53	8	1752.0	0	0	2016.0	0	0	1545.0	0	0
IMG_0131_RGB.jpg	25	3	1733.0	0	0	1861.0	0	0	1594.0	0	0
IMG_0133_IRG.jpg	46	2	1789.0	0	0	2017.0	0	0	1656.0	0	0
IMG_0164_IRG.jpg	12	4	1784.0	0	0	2029.0	0	0	1721.0	0	0
IMG_0190_IRG.jpg	25	5	1788.0	0	0	1836.0	0	0	1824.0	0	0
IMG_0199_IRG.jpg	103	17	1932.0	0	0	1857.0	0	0	1621.0	0	0
IMG_0211_IRG.jpg	22	8	1860.0	0	0	1860.0	0	0	1586.0	0	0
IMG_0215_IRG.jpg	53	3	1874.0	0	0	1914.0	0	0	1579.0	0	0

Table 1: Detection results from pretrained algorithms.

Appendix B - Detection results for Faster R-CNN algorithm

	Theoretical Detections		**PMA Dataset**						**PMA-DA Dataset**		
			14500 steps			***29000 steps***			***14500 steps***		
Image	V_T	B_T	T	V	B	T	V	B	T	V	B
IMG_0003_RGB.jpg	56	11	4379.0	40	10	5788.6	41	9	4122.5	41	9
IMG_0015_IRG.jpg	0	2	1790.0	0	2	2266.0	0	2	1804.0	0	2
IMG_0017_RGB.jpg	1	4	1908.0	1	5	2173.0	0	4	1466.0	0	4
IMG_0025_IRG.jpg	0	0	1775.0	0	0	2022.0	0	0	1463.0	0	0
IMG_0025_RGB.jpg	0	2	1691.0	0	2	2066.0	0	2	1460.1	0	2
IMG_0030_IRG.jpg	0	0	1825.0	0	0	1973.0	0	0	1455.0	0	0
IMG_0030_RGB.jpg	0	0	1766.0	0	0	1967.0	0	0	1461.0	0	0
IMG_0031_IRG.jpg	0	0	1708.0	0	0	2237.0	0	0	1508.0	0	0
IMG_0038_RGB.jpg	54	26	1800.0	28	22	2291.0	27	23	1733.0	27	23
IMG_0039_IRG.jpg	30	8	1950.0	11	5	1815.0	12	4	1583.0	7	2
IMG_0041_IRG.jpg	136	25	1900.0	47	3	1821.0	47	3	1972.0	48	2
IMG_0049_RGB.jpg	0	1	1945.0	0	1	1895.0	0	1	1940.0	0	1
IMG_0050_IRG.jpg	2	6	1854.0	1	6	1789.0	0	6	1842.0	0	5
IMG_0053_RGB.jpg	16	4	1968.0	15	4	1814.0	15	4	1634.0	15	4
IMG_0057_IRG.jpg	70	2	1833.0	39	2	1722.0	34	2	1605.0	29	3
IMG_0057_RGB.jpg	4	3	1988.0	3	3	1748.0	5	2	1570.0	4	2
IMG_0060_IRG.jpg	0	0	1967.0	0	0	1699.0	0	0	1637.0	0	0
IMG_0062_RGB.jpg	0	0	1987.5	0	0	1748.0	0	0	1858.5	0	0
IMG_0066_IRG.jpg	14	3	1982.0	6	1	1750.0	4	2	1849.0	6	0
IMG_0077_IRG.jpg	0	3	1926.5	0	3	1737.0	0	3	1682.0	0	3
IMG_0079_RGB.jpg	6	2	1922.0	6	3	1698.0	5	3	1713.0	3	2
IMG_0084_RGB.jpg	4	3	1818.0	3	4	1834.0	3	4	1959.5	5	5
IMG_0089_RGB.jpg	42	3	1758.0	36	4	1732.0	36	3	1694.0	42	2
IMG_0094_IRG.jpg	6	5	1782.0	3	5	1740.0	4	4	1805.0	3	5
IMG_0100_IRG.jpg	0	0	1761.0	0	0	1734.0	0	0	1758.0	0	0
IMG_0108_RGB.jpg	0	0	1786.0	0	0	1739.0	0	0	1900.0	0	0
IMG_0120_IRG.jpg	101	48	1852.0	47	3	1911.0	48	2	1607.0	48	2
IMG_0120_RGB.jpg	0	6	1767.0	0	6	1852.0	0	6	1627.0	0	6
IMG_0124_IRG.jpg	8	7	1716.0	7	6	1851.0	7	5	1581.0	3	5
IMG_0131_IRG.jpg	53	8	1772.0	28	8	1764.0	37	9	1638.0	22	8
IMG_0131_RGB.jpg	25	3	1930.0	15	3	1691.0	12	3	1576.0	11	3
IMG_0133_IRG.jpg	46	2	1813.0	36	2	1799.0	32	2	1610.0	27	2
IMG_0164_IRG.jpg	12	4	1788.0	9	4	1728.0	7	4	1935.0	5	4
IMG_0190_IRG.jpg	25	5	1881.0	13	5	1733.0	11	5	1930.0	8	4
IMG_0199_IRG.jpg	103	17	1841.0	45	5	1730.0	45	5	1674.0	41	5
IMG_0211_IRG.jpg	22	8	1833.0	16	8	1732.0	17	8	1776.0	13	8
IMG_0215_IRG.jpg	53	3	1859.0	24	4	1701.0	34	2	1618.0	34	3

Table 2: Detection results for Faster R-CNN algorithm.

Appendix C - Detection results for YOLOv3 algorithm

	Theoretical Detections		**PMA Dataset**						**PMA-DA Dataset**		
			100 epochs			***200 epochs***			***100 epochs***		
Image	V_T	B_T	T	V	B	T	V	B	T	V	B
IMG_0003_RGB.jpg	56	11	1027.0	57	11	1080.0	57	11	1052.0	62	11
IMG_0015_IRG.jpg	0	2	1069.0	1	2	1023.0	0	2	1058.0	0	2
IMG_0017_RGB.jpg	1	4	1015.0	1	5	999.0	1	4	1084.0	4	5
IMG_0025_IRG.jpg	0	0	1046.0	0	0	1079.0	0	0	1008.0	0	0
IMG_0025_RGB.jpg	0	2	1039.0	0	2	1004.0	0	2	1016.0	0	2
IMG_0030_IRG.jpg	0	0	1013.0	0	0	1121.0	0	0	1365.0	0	0
IMG_0030_RGB.jpg	0	0	1062.0	0	0	1128.0	0	0	1137.0	0	0
IMG_0031_IRG.jpg	0	0	1050.0	0	0	1137.0	0	0	1317.0	0	0
IMG_0038_RGB.jpg	54	26	1175.0	56	27	1057.0	53	27	1089.0	53	27
IMG_0039_IRG.jpg	30	8	1158.0	29	6	1126.0	30	5	1667.0	24	5
IMG_0041_IRG.jpg	136	25	1133.0	139	23	1125.0	147	23	1583.0	141	23
IMG_0049_RGB.jpg	0	1	1154.0	0	1	1114.0	0	1	1420.1	0	1
IMG_0050_IRG.jpg	2	6	1230.0	4	6	1080.0	3	6	1129.0	3	6
IMG_0053_RGB.jpg	16	4	1126.0	20	4	1064.0	19	4	1304.0	19	4
IMG_0057_IRG.jpg	70	2	1178.0	84	2	1124.0	82	2	1127.1	81	2
IMG_0057_RGB.jpg	4	3	1083.0	5	2	1075.0	6	3	1320.0	4	3
IMG_0060_IRG.jpg	0	0	1168.0	1	0	1227.0	0	0	1327.0	1	0
IMG_0062_RGB.jpg	0	0	1134.0	0	0	1074.0	0	0	1339.0	0	0
IMG_0066_IRG.jpg	14	3	1161.0	16	0	1014.0	21	0	1368.0	19	0
IMG_0077_IRG.jpg	0	3	1119.0	0	3	1035.0	0	3	1454.0	0	3
IMG_0079_RGB.jpg	6	2	1121.0	6	4	1085.0	6	4	1261.0	6	2
IMG_0084_RGB.jpg	4	3	1119.0	5	5	1061.0	4	6	1327.0	4	7
IMG_0089_RGB.jpg	42	3	1102.0	45	4	1111.0	49	4	1111.0	45	4
IMG_0094_IRG.jpg	6	5	1198.0	7	5	1096.0	7	5	1172.0	6	5
IMG_0100_IRG.jpg	0	0	1051.0	0	1	1166.0	0	1	1441.0	0	1
IMG_0108_RGB.jpg	0	0	1063.0	0	0	1036.0	0	0	1326.0	0	0
IMG_0120_IRG.jpg	101	48	1034.0	108	50	1085.0	108	47	1355.0	109	47
IMG_0120_RGB.jpg	0	6	1061.0	1	6	1099.0	0	6	1243.0	0	6
IMG_0124_IRG.jpg	8	7	1070.0	11	7	1126.0	11	7	1176.0	10	7
IMG_0131_IRG.jpg	53	8	1141.0	61	9	1098.0	61	9	1201.0	6	8
IMG_0131_RGB.jpg	25	3	1257.0	21	4	1085.0	24	4	1351.0	25	3
IMG_0133_IRG.jpg	46	2	1259.0	48	2	1097.0	52	2	1441.0	56	2
IMG_0164_IRG.jpg	12	4	1157.0	11	4	1050.0	15	4	1252.0	12	4
IMG_0190_IRG.jpg	25	5	1264.0	23	5	1082.0	25	6	1281.0	25	5
IMG_0199_IRG.jpg	103	17	1094.0	113	17	1063.0	115	17	1228.0	117	17
IMG_0211_IRG.jpg	22	8	1124.0	24	9	1087.0	30	9	1592.0	25	8
IMG_0215_IRG.jpg	53	3	1094.0	57	5	1136.0	61	4	1162.0	59	4

Table 3: Detection results for YOLOv3 algorithm.

Appendix D - Detection results for YOLOv5l algorithm

	Theoretical Detections		PMA Dataset						PMA-DA Dataset		
			100 epochs			*200 epochs*			*100 epochs*		
Image	V_T	B_T	T	V	B	T	V	B	T	V	B
IMG_0003_RGB.jpg	56	11	1306.0	56	11	1166.1	58	11	832.0	60	11
IMG_0015_IRG.jpg	0	2	1305.1	0	3	1186.0	0	2	886.0	0	2
IMG_0017_RGB.jpg	1	4	1245.0	1	5	1107.0	1	4	813.0	1	5
IMG_0025_IRG.jpg	0	0	853.0	0	0	1224.0	0	0	846.0	0	0
IMG_0025_RGB.jpg	0	2	931.0	0	2	1167.0	0	2	826.5	0	2
IMG_0030_IRG.jpg	0	0	1022.0	0	0	784.0	0	0	831.0	0	0
IMG_0030_RGB.jpg	0	0	894.0	0	0	791.0	0	0	833.6	0	0
IMG_0031_IRG.jpg	0	0	999.0	0	0	800.0	0	0	848.0	0	0
IMG_0038_RGB.jpg	54	26	1411.0	52	26	791.0	54	27	818.0	52	28
IMG_0039_IRG.jpg	30	8	924.0	34	4	759.1	25	4	803.0	29	4
IMG_0041_IRG.jpg	136	25	844.0	142	23	782.0	150	24	832.0	149	23
IMG_0049_RGB.jpg	0	1	870.0	0	1	813.0	0	1	882.0	0	1
IMG_0050_IRG.jpg	2	6	899.0	0	6	716.0	3	7	1016.0	2	6
IMG_0053_RGB.jpg	16	4	872.0	20	4	808.0	19	3	1051.0	20	4
IMG_0057_IRG.jpg	70	2	830.0	73	3	823.0	75	3	912.0	80	3
IMG_0057_RGB.jpg	4	3	827.0	5	3	8333.0	5	2	940.0	5	3
IMG_0060_IRG.jpg	0	0	948.0	2	0	821.0	1	0	920.0	1	0
IMG_0062_RGB.jpg	0	0	1172.0	0	0	805.0	0	0	1001.1	0	0
IMG_0066_IRG.jpg	14	3	1439.9	15	2	889.0	17	1	980.0	17	0
IMG_0077_IRG.jpg	0	3	1040.0	0	3	816.3	0	3	1093.0	0	3
IMG_0079_RGB.jpg	6	2	1044.0	6	3	957.0	6	3	1083.1	6	4
IMG_0084_RGB.jpg	4	3	1263.9	5	5	1022.0	5	6	1365.0	4	5
IMG_0089_RGB.jpg	42	3	1396.8	49	4	1022.0	48	4	1150.0	47	3
IMG_0094_IRG.jpg	6	5	1050.0	6	6	784.0	7	5	1212.0	7	5
IMG_0100_IRG.jpg	0	0	1105.0	0	1	820.0	0	1	1173.0	0	1
IMG_0108_RGB.jpg	0	0	1402.0	0	0	979.0	0	0	1044.0	0	0
IMG_0120_IRG.jpg	101	48	1211.0	105	46	1071.9	105	49	1103.0	113	45
IMG_0120_RGB.jpg	0	6	1044.0	0	6	898.0	0	6	920.0	0	6
IMG_0124_IRG.jpg	8	7	1128.0	8	7	966.0	7	7	1177.0	5	7
IMG_0131_IRG.jpg	53	8	1166.0	57	9	868.0	64	8	1082.0	66	8
IMG_0131_RGB.jpg	25	3	1052.0	22	4	869.0	21	4	1023.5	22	3
IMG_0133_IRG.jpg	46	2	1068.0	54	2	897.0	60	2	1033.0	65	2
IMG_0164_IRG.jpg	12	4	1062.0	11	4	1011.0	11	4	1021.0	12	4
IMG_0190_IRG.jpg	25	5	1108.0	24	7	887.0	25	7	1012.0	25	6
IMG_0199_IRG.jpg	103	17	1042.0	112	17	1046.0	121	17	1038.0	116	17
IMG_0211_IRG.jpg	22	8	1118.0	23	9	880.0	22	8	999.1	19	8
IMG_0215_IRG.jpg	53	3	1046.0	60	4	839.0	56	4	1048.1	61	4

Table 4: Detection results for YOLOv5l algorithm.

Appendix E - Extended Abstract

The Extended Abstract, “Discrimination between vehicles and buildings in military aerial images”, written for the Conference of the International Society of Military Sciences, is presented on the following pages.

Discrimination between vehicles and buildings in military aerial images

Rita Amante (1), José Silvestre Silva (2) and António Neves (3)

(1) DETI – Dep. of Electronics, Telecommunications and Informatics, University of Aveiro, Portugal, rita.amante@ua.pt
(2) Portuguese Military Academy & CINAMIL & LIBPhys-UC, jose.silva@academiamilitar.pt
(3) DETI & IEETA, University of Aveiro, Portugal, an@ua.pt

ABSTRACT
The advancement of technology has facilitated the detection of objects in aerial images, a task which must be performed quickly and accurately. This work analyzed the performance of object detection algorithms, Faster R-CNN, YOLOv3 and YOLOv5l, for the detection of vehicles and buildings in aerial images obtained by UAV. The results showed that YOLOv5l had the best performance and the fastest detection time.

KEYWORDS
Deep Learning; Object Detection; Transfer Learning; UAV.

INTRODUCTION

The expansion of Unmanned Aerial Vehicles (UAV), remotely controlled aerial vehicles, made it possible to access hard-to-reach places and vast amounts of data, as well as enabled the improvement of object detection algorithms, a growing topic of great interest and applicability, in both civil and military contexts. UAVs capture images which can be used in traffic monitoring, road accidents, operations supporting government agencies, predicting enemy movements, locating areas occupied by enemy troops, disaster assistance, detecting illegal activities, planning offensive operations, among others.

In recent years, several studies have been carried out on the detection of objects in aerial images. Saetchnikov *et al.* (2021) showed that the You Only Look Once (YOLO) version 3 algorithm had better average accuracy and a faster detection time than the Region-based Convolutional Neural Network (R-CNN) algorithms, Fast R-CNN and Faster R-CNN. Nepal *et al.* (2022) revealed that the YOLOv3, YOLOv4 and YOLOv5l algorithms are efficient for the detection of objects in real time, highlighting the YOLOv5l for its accuracy.

This gave emphasis to the development of algorithms for the detection of vehicles and buildings in aerial images, namely in military scenario. It is a pertinent task which can be useful for several applications in the real world. Two object detection algorithms based on Deep Learning were studied, Faster R-CNN, a two-stage detector, and YOLO, a one-stage detector. The latter was implemented in the YOLOv3 and YOLOv5l versions.

METHODOLOGY

The adopted methodology started with the choice of Faster R-CNN, YOLOv3 and YOLOv5l algorithms. Then, the data were prepared. A set of 363 images was selected and renamed, resulting in the Portuguese Military Academy (PMA) dataset. All target objects were annotated with limiting boxes and respective classes (*vehicle* or *building*), using the LabelImg tool, and the annotations were saved in Extensible Markup Language (XML) files, in PASCAL VOC format. A total of 11945 instances, 9500 vehicles and 2445 buildings, were registered. The images were resized to 640×640 (pixels per inch), to which brightness and Gaussian blur transformations were applied, resulting in the Portuguese Military Academy with Data Augmentation (PMA-DA) dataset. Both sets were divided into three parts: training (80%), validation (10%) and testing (10%).

The following step consisted of inferring the pretrained algorithms using the Transfer Learning (TL) pretrained network as a classifier approach, to carry out qualitative (direct observation of detection results) and quantitative (observation of the detection times) evaluations. The TensorFlow 2 Object Detection API was used for Faster R-CNN and the open-source code developed by Ultralytics was used for YOLO, both pretrained with the Microsoft Common Objects in Context (MS COCO) dataset (Lin *et al.*, 2015).

Subsequently, the input data were processed, converting the XML files into TFRecord for Faster R-CNN and into text files for the YOLO algorithms. The training parameters of batch size, learning rate, Intersection Over Union (IoU) limit and confidence limit were configured with values of 2, 0.013, 0.5 and 0.3, respectively. The YOLOv3 and YOLOv5l were trained for 100 and 200 epochs with the PMA dataset and for 100 epochs with the PMA-DA dataset. The Faster R-CNN was trained for 14500 and 29000 steps with the PMA dataset, the equivalent to 100 and 200 epochs, respectively, and for 43500 steps with the PMA-DA dataset, the equivalent of 100 epochs with the augmented data. The training of the algorithms followed the pretrained network as a feature extractor approach of TL.

The last step consisted of the inference of the trained algorithms, where a qualitative and quantitative evaluation (interpretation of metrics and detection time) of the performance of the trained algorithms was carried out.

RESULTS AND DISCUSSION

In the inference of the pretrained Faster R-CNN, YOLOv3 and YOLOv5l algorithms, it was confirmed that the building class was not detected, possibly because this class does not belong to the categories of the MS COCO dataset, and the performance of the algorithms in the detection of vehicles was low, probably because the pretrained algorithms did not learn to detect objects in aerial images.

Table 1 shows the results of each algorithm after training. It shows that Faster R-CNN has the lowest mean Average Precision (mAP) because, although it shows high Precision, Recall is low, which indicates that most objects were detected correctly, but the number of false negatives was high. And the YOLO algorithms have a higher mAP since both Precision and Recall are high, which indicates that most objects were detected correctly, decreasing the number of false positives and false negatives.

Table 1 – Results of the trained algorithms: Faster R-CNN, YOLOv3 and YOLOv5l.

	FASTER R-CNN			YOLOv3			YOLOv5l		
	PMA		PMA-DA	PMA		PMA-DA	PMA		PMA-DA
	14500 steps	29000 steps	43500 steps	100 epochs	200 epochs	100 epochs	100 epochs	200 epochs	100 epochs
Training time (hours)	8	16	25	30	60	127	24	48	73
mAP (All)	58.1%	56.2%	52.5%	91.4%	92.2%	88.7%	93.3%	91.8%	91.4%
Precision (Vehicle)	91.9%	93.7%	94.8%	94.1%	93.2%	91.9%	95.4%	92.8%	93.4%
Recall (Vehicle)	50.4%	47.9%	58.2%	91.1%	92.3%	88.9%	90.8%	91.6%	91.4%
Precision (Building)	98.9%	97.7%	97.2%	93.7%	94.5%	97.1%	96.0%	97.2%	96.4%
Recall (Building)	47.0%	44.5%	50.1%	82.3%	85.4%	76.0%	85.4%	91.5%	81.6%
Detection time (milliseconds)	1914	1954	1758	1118	1088	1273	1079	911	985

Figure 1 illustrates the YOLOv5l algorithm detection results for two images from the PMA dataset, where the blue color corresponds to vehicles and the yellow to buildings. The YOLOv5l was capable of detecting all vehicles and buildings. However, there were still some false positives.

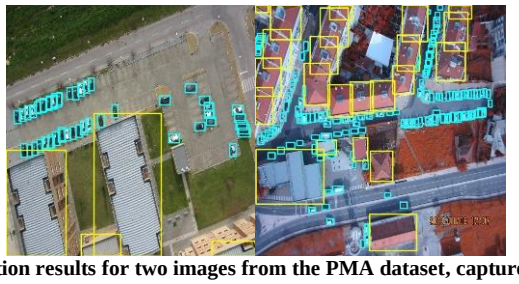


Figure 1 – YOLOv5l algorithm detection results for two images from the PMA dataset, captured in area UEO of Portuguese Army.

CONCLUSION

The performance of the algorithms show that the Precision for the building class was higher than that of the vehicle class, which indicates that the algorithms recognise larger objects more effectively; the Recall of the building class was lower than that of the vehicle class, due to the smaller number of instances of this class in the datasets; the number of steps or epochs did not influence detection time; training with data augmentation did not improve the performance of the algorithms, probably because there was an increase in new difficulties, requiring more training time for the algorithms to learn; Faster R-CNN needed less training time, but took longer to detect objects and had the worst performance; YOLOv3 had the slowest training time and YOLOv5l achieved the best performance, with a mAP of 93.3%, fast training time and the best detection time. It was concluded that YOLOv5l was the most efficient for detecting vehicles and buildings in aerial images and useful for real-time applications, which is in line with the two studies mentioned.

In terms of future work, two approaches can be applied to the YOLOv5l algorithm: the expansion of the dataset with images from other locations and environments, and the adjustment of hyperparameters.

ACKNOWLEDGEMENTS

This research was supported by the Military Academy Research Center (CINAMIL), the Center for Research and Development of the IUM (CIDIUM). The images were made available by the Portuguese Military Academy.

REFERENCES

Lin, T., Marie, M., Belongie, S.J., Bourdev, L. D., Girshick, R. B., Hays, J., … Zitnick, C. L. (2015, Feb). Microsoft COCO: Common Objects in Context. doi: 10.48550/ARXIV.1405.0312v.

Nepal, U. & Eslamiat, H. (2022, Jan). Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors 22*(2). doi: 10.3390/s22020464.

Saetchnikov, I. V., Tcherniavskaia, E. A. & Skakun, V. V. (2021, Jun). Object Detection for Unmanned Aerial Vehicle Camera via Convolutional Neural Networks. *IEEE Journal on Miniaturization for Air and Space Systems, 2*(2). doi: 10.1109/JMASS.2020.3040976.