UNIVERSIDADE
BEIRA INTERIOR

Object Detection in Data Acquired From Aerial Devices

Pedro Jorge Franco Brito

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Dr. Hugo Pedro Martins Carriço Proença

Covilhã, Junho 2022

Acknowledgements

The work accomplished in this dissertation would never have been possible without the help of a particular group of people, and thus, this dissertation is dedicated to them.

First and foremost, I would like to sincerely thank my supervisor, Doctor Hugo Proença, for everything he has done throughout the development of this dissertation. I thank him for his guidance, patience, engagement in my work and especially his indispensable and insightful reviews of the different phases of this dissertation, which kept encouraging me to do more and better. Since the first time I heard him teach, I have admired the way he teaches, the way he expresses his knowledge, and his critical thinking. He is one of the main reasons I decided to follow this path of Computer Vision, and I do not think I could have made a better choice. To him, I give my sincerest thank you.

I would also like to thank Bruno Degardin and João Brito for their help these past months. They have been nothing more than patient, communicative and constantly supportive towards me. I thank them for everything they have done, and I sincerely hope this was the start of an incredible journey ahead.

To my parents, I cannot express how much I appreciate everything they have done for me throughout my entire life. All I can say is that I am incredibly grateful for their unconditional love, care, constant support, and for always believing in me and my dreams no matter the circumstances.

To my dearest and closest friends, I thank you all for always being by my side no matter what, always being able to cheer me up in difficult times, and for your care and support throughout this incredible journey. Especially, I want to convey my gratitude to a certain group of friends who helped me gather the data for this work by walking and driving through the wild forest, namely my girlfriend Isabel Marques and my dear friends António Abreu, Miguel Barbosa and Filipe Gameiro; I want to thank them for helping me bring my ideas into life and for their crucial support.

Last but not least, I want to thank the kind, supportive and friendly colleagues I have met in the past months I have spent developing this dissertation at Socialab. Their humour, help, and criticism created a unique working environment in the lab which I am very grateful for.

Resumo

A tarefa de deteção de objetos, tanto em imagem como em vídeo, tem contribuído com inúmeros avanços extraordinários no que toca a arquiteturas inovadoras e ao desenvolvimento de conjuntos de dados cada vez mais completos e de qualidade. Nesse sentido, a maioria dos modelos consegue adaptar-se a quase qualquer cenário do mundo real – se existirem dados suficientes –, uma vez que estes modelos são treinados nestes grandes conjuntos de dados. No entanto, existe um cenário específico – as imagens aéreas –, e que devido às suas caraterísticas naturais, estes modelos tendem a mostrar um desempenho de menor qualidade. Contudo, a diferença de escala do próprio objeto que precisa de ser localizado e identificado é o principal aspeto que marca a diferença entre os conjuntos de imagens típicas e os conjuntos de imagens aéreas. Além disso, fatores como o brilho da imagem, a rotação do objeto, os detalhes do mesmo e as cores de fundo também desempenham um papel crucial no desempenho do modelo, independentemente da sua arquitetura.

Modelos de aprendizagem profunda tomam decisões com base nas características que conseguem extrair do conjunto de imagens de treino. Esta técnica funciona particularmente bem em cenários padrão, em que as imagens representam o objeto numa escala normal, onde os detalhes do objeto são precisos e permitem que o modelo o distinga de outros objetos. Contudo, ao considerar um cenário onde a imagem está a ser capturada a 50 metros de altura, os detalhes do objeto diminuem consideravelmente e, portanto, torna-se mais difícil para o modelo extrair as melhores caraterísticas significativas que permitem a identificação e localização do objeto. Atualmente, muitos sistemas de vigilância utilizam câmaras estáticas colocadas em locais pré-definidos; porém, uma abordagem mais apropriada para alguns cenários poderia passar por utilizar *drones* de modo a vigiar uma determinada área com um percurso pré-definido. Mais especificamente, estes tipos de vigilância seriam adequados a cenários em que não é viável cobrir toda a área com câmaras, tal como florestas.

O primeiro objetivo do presente trabalho passa por reunir um conjunto de dados que se foque na deteção de pessoas e veículos em florestas. O conjunto de dados foi capturado com um *drone DJI* em quatro zonas distintas da Serra da Estrela, e contém gravações que foram capturadas com diferentes condições meteorológicas – sol e nevoeiro – e durante diferentes fases do dia – manhã, tarde e ao anoitecer. Além do mais, contempla também quatro tipos diferentes de terreno, terra, alcatrão, floresta e gravilha, para além de existirem duas classes de objetos, pessoa e veículo.

Posteriormente, o segundo objetivo contempla a análise precisa do modo como os detetores de objetos de vídeo e imagem atuam no conjunto de dados anteriormente descrito. A análise centra-se no desempenho dos modelos em relação a cada classe de objeto e a cada terreno. Com isto, conseguimos demonstrar uma perspetiva das situações exatas em que os diferentes tipos de modelos se destacam e quais os que tendem a não ter um desempenho tão adequado.

Finalmente, com base nos resultados obtidos durante a primeira fase de experiências, o ob-

jetivo final tem como propósito propor dois métodos em que cada um deles visa resolver um problema diferente que surgiu da aplicação destes detetores em imagens aéreas. O primeiro método destaca a utilização de algoritmos de remoção de fundo para melhorar o desempenho dos modelos de deteção de objetos em vídeo em determinadas situações com o objetivo de delimitar áreas específicas nas quais as deteções dos modelos devem ser consideradas válidas.
Um dos principais problemas na criação de um conjunto de dados de alta qualidade a partir do zero é o processo intensivo e moroso de anotação após a recolha dos dados. Com respeito a isto, o segundo método proposto consiste numa arquitetura auto-supervisionada que tem como objetivo enfrentar a escassez particular de conjuntos de dados aéreos de alta qualidade. A ideia principal é analisar a utilidade dos dados não anotados nestes projetos e, assim, evitar o processo demorado e custoso de anotar a totalidade de um conjunto de dados aéreos. Os resultados relatados mostram que, mesmo com um conjunto de dados parcialmente anotado, é possível utilizar os dados não anotados numa arquitetura auto-supervisionada para melhorar ainda mais o desempenho do modelo.

Palavras-chave

Rede Neuronais Convolucionais, Aprendizagem Profunda, Deteção de Objetos, Imagens Aéreas, Aprendizagem Auto-Supervisionada, Aprendizagem Supervisionada, Gravações com *Drone*, Inteligência Artificial.

Resumo Alargado

A tarefa de deteção de objetos, tanto em imagem como em vídeo, tem contribuído com inúmeros avanços extraordinários no que toca a arquiteturas inovadoras e ao desenvolvimento de conjuntos de dados cada vez mais completos e de qualidade. Nesse sentido, a maioria dos modelos consegue adaptar-se a quase qualquer cenário do mundo real – se existirem dados suficientes –, uma vez que estes modelos são treinados nestes grandes conjuntos de dados. No entanto, existe um cenário específico – as imagens aéreas –, e que devido às suas caraterísticas naturais, estes modelos tendem a mostrar um desempenho de menor qualidade. Contudo, a diferença de escala do próprio objeto que precisa de ser localizado e identificado é o principal aspeto que marca a diferença entre os conjuntos de imagens típicas e os conjuntos de imagens aéreas. Além disso, fatores como o brilho da imagem, a rotação do objeto, os detalhes do mesmo e as cores de fundo também desempenham um papel crucial no desempenho do modelo, independentemente da sua arquitetura. Mesmo por isto, torna-se crucial desenvolver métodos que tenham em atenção estes fatores ou adquirir arquiteturas que tenham em consideração os modelos já existentes mas que apliquem métodos de pós-processamento para melhorar as deteções dos mesmos.

A presente dissertarão tem como objetivo principal analisar o modo como os modelos de deteção de objetos, que funcionam moderadamente bem para conjuntos de dados em imagens típicas, se comportam quando confrontados com um conjunto de dados em imagens aéreas. Mais especificamente, ao longo de toda a dissertação, pretende-se: i) explorar os conjuntos de dados já existentes nesta área; ii) desenvolver e anotar um conjunto de dados especificamente para este projeto com todas as suas caraterísticas; iii) realizar experiências com os modelos de estado da arte no conjunto de dados recolhido e analisar meticulosamente os resultados; iv) com base nos resultados obtidos, propor um método de pós-processamento que visa à filtração das deteções destes modelos; v) propor uma arquitetura auto-supervisionada de modo a utilizar os dados não anotados para desenvolver um modelo supervisionado mais robusto.
Primeiramente, começámos por procurar e analisar alguns dos conjuntos de dados já existentes e disponíveis à procura de algum que incluísse as caraterísticas relativas ao problema tratado nesta dissertação. Após analisarmos vários conjuntos de dados, apercebemos-nos que nenhum conjunto de dado atualmente disponível, cumpria os requisitos para ser útil ao desenvolvimento desta dissertação. O fator que é mais comum nestes conjuntos de dados já existentes, é o facto de a maior parte ser captada através de satélites ou câmaras fixas, o que os torna pouco útil em aplicações de vigilância. Tendo isto em consideração, o próximo passo foi analisar as possibilidades de recolher o nosso próprio conjunto de dados. Visto que a presente dissertação tem um foco muito especifico em zonas florestais e na possível prevenção de incêndios, marcámos uma reunião com o ex Comandante dos Bombeiros Voluntários da Covilhã, Fernando Lucas, de modo a estudar todas as opções de recolha de dados. Posteriormente, decidimos que o uso de um *drone* para efetuarmos a recolha dos dados seria a melhor opção, não só porque se adequa mais à vigia, mas como também torna a recolha de dados mais dinâmica do que com várias câmaras fixas. Posto isto, procedemos à recolha dos

dados com recurso a um *drone DJI Phantom 4 Pro* em diferente zonas da Serra da Estrela. Teve-se em consideração várias aspetos tais como, condições meteorológicas – sol e nevoeiro –, diferentes fases do dia – manhã, tarde e anoitecer –, diferentes terrenos – terra, alcatrão, floresta e gravilha – e duas classes de objetos – pessoa e veículo. Após a recolha dos dados, procedemos à anotação dos mesmo com auxílio do *CVAT*. No processo de anotação, tivemos de ser muito precisos na delineação das caixas de delimitação dos objetos pois, o facto de o próprio objeto já ser pequeno torna ainda mais necessário que a zona anotada contenha apenas os detalhes do objeto e não do fundo.

A segunda fase da presente dissertação passou por explorar o estado da arte relativo à deteção de objetos, tanto em imagem como em vídeo. O facto de considerarmos os dois tipos de detetores serve para termos uma noção das vantagens de desvantagens de cada um quando tivermos de considerar qual deles utilizar no sistema final. Na fase inicial de desenvolvimento, começámos por tentar reproduzir vários destes modelos numa máquina local e utilizando os conjuntos de dados típicos (*e.g.*, *COCO* e *ImageNet-VID*). Assim que conseguimos reproduzir os resultados que os autores dos modelos demonstravam nos seus resultados, procedemos à adaptação do nosso conjunto de dados para estes formatos, onde *COCO* é o formato mais utilizado para detetores em imagens e *Imagenet-VID* é utilizado para detetores em vídeo. Após a organização das anotações consoante os critérios de ambos os formatos, procedemos à realização de várias experiências com diferentes modelos.

Na terceira fase da dissertação, conseguimos analisar as deteções de cada modelo em termos de precisão e perceber quais dos modelos se comportam melhor. No entanto, analisar apenas métricas como a precisão ou *recall* torna-se um pouco ambíguo no que toca a perceber exatamente onde cada modelo falha e porquê. Posto isto, prosseguimos a desenvolver vários *scripts* que não só tinham em consideração a precisão e *recall*, mas que também consideravam a especificidade de cada modelo. Esta métrica dá-nos a informação de como o modelo se comporta nos cenários em que é suposto não detetar nenhum objeto. Nesse sentido e visto que o conjunto de dados foi recolhido tendo em consideração vários terrenos, procedemos à analise dos resultados tendo em consideração as mesmas métricas mas separando as deteções por cada terreno. Para conseguirmos esta análise por terreno, tivemos de, primeiro, treinar um modelo de segmentação semântica que fosse capaz de modelar os diferentes terrenos de várias imagens e de cenários diferentes. Procedemos à anotação de 100 imagens do conjunto de dados tendo em consideração vários cenários possíveis e com uma seleção de terrenos similar. Posteriormente, com o modelo de segmentação semântica a funcionar conseguimos atribuir um terreno a cada deteção e assim conseguimos categorizar as métricas que obtivemos previamente por terreno, e consecutivamente, analisar em detalhe onde os modelos falham e porquê.

Na fase seguinte da dissertação, e tendo em conta os resultados obtido anteriormente, propomos dois métodos distintos que visam a melhorar o desempenho tanto dos detetores em imagem como em vídeo. Primeiramente, como verificámos que os detetores em vídeo tin-

ham pior desempenho na terra, e depois de uma análise, concluímos que haviam zonas neste tipo de terreno que se pareciam com uma pessoa quando vistas de tal altitude. Com isto, e sabendo que estes casos específicos eram a razão para o modelo estar a ter um desempenho menos bom, começamos a explorar opções em que conseguíamos limitar as deteções do modelo apenas para zonas onde havia movimento. Começamos por experimentar algoritmos de remoção de fundo que são normalmente usados em imagens de câmara estática, mas como esperado, quando aplicados ao conjunto de dados desta dissertação, os resultados obtidos não conseguiam modelar corretamente o fundo e os objetos. Procedemos há análise de outros algoritmos de remoção de fundo que se especializavam principalmente em câmaras que se estavam a mover, e ao fazermos várias experiências com um dos algoritmos, apercebemos-nos que este conseguia modelar o fundo e os objetos precisamente da maneira como precisávamos. Ao conseguirmos modelar o fundo e os objetos em cada *frame* de teste, conseguimos delinear regiões que está a acontecer movimento e portanto, o método proposta passa por limitarmos as deteções do modelo apenas a essas regiões. Ao utilizarmos este método, obtivemos um aumento significativo do desempenho dos modelos de deteção de objetos em vídeo e conseguimos, quase na totalidade remover os casos em que o detetor dava a presença de um objeto mas que era simplesmente um buraco ou sombra.
Por fim, o segundo e último método proposto na presente dissertação foca-se especialmente numa maneira de conseguirmos utilizar os dados não anotados, com base numa arquitetura auto-supervisionada, de modo a desenvolver um modelo supervisionado mais robusto. Depois da recolha dos dados, reparámos que tínhamos um enorme conjunto de dados e que anotar este mesmo conjunto de dados na sua totalidade seria um trabalho exaustivo. Com isto, começámos a analisar diferentes maneiras de como podíamos usar os dados que não estavam anotados. Seguindo isto, verificámos que a aprendizagem auto-supervisionada foca-se especialmente neste conceito, onde temos apenas dados parcialmente anotados. Sendo assim, explorámos o estado da arte relativo a aprendizagem auto-supervisionada com foco a deteção de objetos e deparámos-nos com uma arquitetura que fazia uso de dados anotados em três modalidades – RGB, térmica e profundidade –, e que utilizava dados não anotados na modalidade de som. Com base nesta arquitetura, considerámos em modificar a mesma mas de maneira a que servi-se o nosso problema, onde apenas teríamos uma modalidades, RGB, para o *teacher* e para o *student*. Seguindo este conceito, adaptámos a arquitetura para este modo e procedemos à realização de várias experiências com diferentes *ratios* de dados anotados e de dados não anotados. No final das experiências, conseguimos demonstrar que, de facto, a partir de um certo *ratio* de dados, é preferível não anotarmos o conjunto de dados na sua totalidade mas ao invés, anotar uma parte de esses dados de uma maneira supervisionada e depois, utilizar os dados não anotados para desenvolver um modelo ainda mais robusto.

Abstract

The object detection task, both in images and in videos, has been the source of extraordinary advances with state-of-the-art architectures that can achieve close to perfect precision on large modern datasets. As a result, since these models are trained on large-scale datasets, most of them can adapt to almost any other real-world scenario if given enough data. Nevertheless, there is a specific scenario, aerial images, in which these models tend to perform worse due to their natural characteristics. The main problem differentiating typical object detection datasets from aerial object detection datasets is the object's scale that needs to be located and identified. Moreover, factors such as the image's brightness, object rotation and details, and background colours also play a crucial role in the model's performance, no matter its architecture.

Deep learning models make decisions based on the features they can extract from the training data. This technique works particularly well in standard scenarios, where images portray the object at a standard scale in which the object's details are precise and allow the model to distinguish it from the other objects and background. However, when considering a scenario where the image is being captured from 50 meters above, the object's details diminish considerably and, thus, logically, making it harder for deep learning models to extract meaningful features that will allow for the identification and localization of the said object. Nowadays, many surveillance systems use static cameras placed in pre-defined places; however, a more appropriate approach for some scenarios would be using drones to surveil a particular area with a specific route. More specifically, these types of surveillance would be adequate for scenarios where it is not feasible to cover the whole area with static cameras, such as wild forests.

The first objective of this dissertation is to gather a dataset that focuses on detecting people and vehicles in wild-forest scenarios. The dataset was captured using a *DJI* drone in four distinct zones of Serra da Estrela. It contains instances captured under different weather conditions – sunny and foggy – and during different parts of the day – morning, afternoon and evening. In addition, it also includes four different types of terrain, earth, tar, forest, and gravel, and there are two classes of objects, person and vehicle.
Later on, the second objective of this dissertation aims to precisely analyze how state-of-the-art single-frame-based and video object detectors perform in the previously described dataset. The analysis focuses on the models' performance related to each object class in every terrain. Given this, we can demonstrate the exact situations in which the different models stand out and which ones tend to perform the worse.
Finally, we propose two methods based on the results obtained during the first phase of experiments, where each aims to solve a different problem that emerged from applying state-of-the-art models to aerial images. The first method aims to improve the performance of the video object detector models in certain situations by using background removal algorithms to delineate specific areas in which the detectors' predictions are considered valid.
One of the main problems with creating a high-quality dataset from scratch is the intensive

and time-consuming annotation process after gathering the data. Regarding this, the second method we propose consists of a self-supervised architecture that aims to tackle the particular scarcity of high-quality aerial datasets. The main idea is to analyze the usefulness of unlabelled data in these problems and thus, avoid the immense time-consuming process of labelling the entirety of a full-scale aerial dataset. The reported results show that even with only a partially labelled dataset, it is possible to use the unlabelled data in a self-supervised matter to improve the model's performance further.

Keywords

Convolutional Neural Networks, Deep Learning, Object Detection, Aerial Images, Self-Supervised Learning, Supervised-Learning, Drone Recordings, Artificial Intelligence.

Contents

List of Figures

List of Tables

Acronyms

Bi-FPN Bi-directional Feature Pyramid Network. 9, 10

CNN Convolutional Neural Network. 1, 6–8, 12–14, 18

CVAT Computer Vision Annotation Tool. 33

DFF Deep Feature Flow. xvi, 42, 43, 46, 61

DOTA Dataset for Object Detection in Aerial Images. xv, 11, 12, 29

FGFA Flow Guided Feature Network. xvi, xvii, 46, 53, 61

FPN Feature Pyramid Network. 9, 10

GAN Generative Adversarial Network. xv, 20

GPU Graphical Processing Unit. 6

HR-Net High Resolution Network. 19

IoU Intersection-Over-Union. 7, 39

MTA Multi-Teacher Alignment. 23, 24

NLP Natural Language Processing. 20

POLS Partially-overlapping Local Subspaces. 26

R-CNN Region-based Convolutional Neural Network. xv, 6–8, 11, 17

R-FCN Region-based Fully Convolutional Neural Network. 11

RPA Region Proposal Algorithm. 6

RPN Region Proposal Network. 7, 8, 13

SE-Sync A Certifiably Correct Algorithm for Synchronization over the Special Euclidean Group. 26

SSD Single-Shot Detector. 11

STN Spatial Transformer Net. 26

SVM Support Vector Machine. 5, 7, 18

Chapter 1

Introduction

Computer vision is, nowadays, one of the essential sub-fields of machine learning due to its importance in modern-day applications (*e.g.,* autonomous vehicles, health, security, and surveillance systems). These applications are assembled based on various computer vision tasks such as image classification, object detection, and image segmentation. With this being said, it is also a highly dependent field on data quality, meaning that tons of resources must be invested into data collection and labelling tasks to develop new techniques that achieve great results.

Even though many advances in this field have been made, a few topics still need further investigation and improvement; one of those topics is object detection in aerial data. On the one hand, object detection in typical images has made significant progress, and the methods developed can achieve impressive results. However, there are few large-scale datasets for aerial data, and the problems' natural obstacles are slowing down the advancements in this field. In order to obtain large-scale datasets, there is the need to make significant investments in terms of money and time. Due to this, it is crucial to design methods that perform well on these tasks but do not require an immense amount of annotated data that is sometimes impossible to acquire.

1.1 Problem Statement

The object detection task involves identifying and locating an object in images and videos. There are two main techniques to perform this task in computer vision: machine learning and deep learning-based approaches. The first group of techniques uses more statistical methods (*e.g.,* edge detection and pixel values histograms) to define a region of pixels where an object may be located considering those hand-crafted features. On the other hand, the latter assigns the responsibility of feature extraction to Convolutional Neural Network (CNN)s. In these networks, if provided with enough data, they can extract the most meaningful features for a given problem. Due to this, most state-of-the-art object detection methods use a CNN as the backbone to extract features from images. Nonetheless, every convolutional network is only as good as the input data.

Object detection has been around for a while, and many state-of-the-art techniques have achieved impressive results on the most known large-scale datasets. However, these techniques tend to perform worse when applied to aerial image problems. In aerial images, the objects are on a much smaller scale, thus, making it harder for the network to find meaningful features to identify that particular object. Note that if the camera is straight in front of a

person, many details can be taken into account (*e.g.*, clothing, face details, colour, and many more). However, considering an image from a person from 50 meters above, the quantity and quality of the details that characterize that object diminish immensely and thus, making it harder for the network to find the best features. Further, this will be thoroughly analyzed and discussed, illustrating how this lack of detail in objects impacts the performance of state-of-the-art methods.

1.2 Motivation and Objectives

With more and more resources being invested into computer vision and more techniques being proposed every day, it is essential to start applying these in systems that help surveillance on more restricted natural areas and avoid any disasters from criminal activities. The primary motivation for this work was developing a system capable of performing surveillance with a drone in wild areas such as forests. This surveillance would be performed in interdicted areas for people and vehicles, especially in the summer.
The initial goal of this work is to provide a comparative analysis between state-of-the-art video object detectors and state-of-the-art single-frame-based methods. Thoroughly analyze how they perform on this project's dataset, not only in precision but also on how they perform in different terrains. Additionally, after analyzing the scenarios in which these models perform worst, we aim to solve these problems with post-processing methods that can tackle the problem's nature difficulty. Furthermore, given how costly and complicated the process of collecting and labelling a large-scale dataset of aerial images and videos is, this work also provides a comparative overview of supervised and self-supervised learning methods in terms of the model's precision and the dataset's size. With these experiments, we aim to prove that, instead of labelling the entirety of the dataset, we can partially label it and then use the rest of the unlabeled data to create a more robust method through self-supervised learning methods.

1.3 Main Contributions

As the first contribution of this work, we present a new dataset explicitly gathered for object detection in data taken from aerial devices. Moreover, unlike any of the existing datasets, this dataset focuses mainly on wild-forest scenarios and the data is gathered using a drone. As described in chapter 3, the dataset provides a wide range of scenarios from different zones and terrains, with two classes: people and vehicles.
For the second contribution, we thoroughly analyze how the different state-of-the-art object detection methods perform in aerial data, identifying the specific scenarios in which they perform better or worse.
After specifically analyzing each method's pros and cons so far as their performances' in each specific terrain, this work's third contribution is using post-processing methods to improve the models' performances in these difficult situations. Moreover, as proven in subsection 4.2.4, we significantly improved the models' performances using this proposed method.

Last but not least, the fourth contribution of this work is based on the self-supervised architecture proposed in section 4.3, which has proven that it can use unlabeled data to improve the supervised model's performance further. This method can be incredibly fitting in scenarios where the amount of unlabeled data is considerably more than the labelled data, thus avoiding allocating even more resources to label the entire dataset.

1.4 Document Organisation

The organization of this document is divided into the following chapters:

- Chapter 1 – **Introduction** – provides an overview of the problem statement. It also describes the main motivations and objectives for this work, along with the organization of the present document.
- Chapter 2 – **Object Detection: Related Work** – presents an analysis of state-of-the-art methods for object detection, both single-frame-based and video. It also depicts an overview of the state of aerial image datasets and the methods developed specifically for these scenarios. Furthermore, it summarizes the methods and techniques used in semantic segmentation. This chapter also provides a broad overview of state-of-the-art self-supervised techniques and in which scenarios they are useful. Finally, this chapter also illustrates the different background subtraction algorithms and the types of scenarios for which they were designed.
- Chapter 3 – **Serra Dataset** – illustrates the state-of-the-art datasets in the aerial image field and explains why these datasets were not a good fit for this project. Furthermore, a contextualization is provided for the dataset, along with its specifications and statistics. Finally, this chapter also describes the dataset's second iteration, including the amount of data gathered and its purpose.
- Chapter 4 – **Experiments and Discussion** – primarily explains the different metrics used to evaluate the different object detection models. Afterwards, this chapter describes the experiments made for the supervised methods, thoroughly analyzes the result, and posteriorly explains the experiments performed using the background subtraction algorithms. After that, this chapter illustrates the experiments performed using the self-supervised method, and its results are analyzed and used to make conclusions about the proposed method.
- Chapter 5 – **Conclusions and Further Work** – concludes the document with the most relevant points taken from the work's development and defines the main aspects in which it can be further improved.

Chapter 2

Object Detection: Related Work

This chapter reviews state-of-the-art techniques and methods related to object detection, describing the advantages and disadvantages of each one. Section 2.1 presents a brief introduction to deep learning, underlining its main concepts and single-frame state-of-the-art object detection architectures. Afterwards, section 2.2 outlines the main differences between regular and aerial object detection. Subsequently, section 2.5 describes another approach to object detection, self-supervised learning, stating its pros and cons, the principal state-of-the-art methods employed, and their benefits to the problem. Finally, section 2.7 briefly summarizes the major points of the previous sections, emphasizing the most relevant conclusions and aspects related to object detection in aerial data.

2.1 Deep Learning

Deep learning is a branch of machine learning that uses deep neural networks inspired by how the human brain works. These networks can achieve exceptional results in many problems and are applied to different types of problems (*i.e.,* text, vision, speech). However, note that these models are data devour, which means they need an immense amount of data to train and achieve great results. If the amount of data is small, then lower complexity methods are a better alternative to these deep neural networks (*e.g.,* SVMs and Naive Bayes).

The first artificial neural networks were proposed as a computational model for neural networks by Warren McCullough and Walter Pitts, two researchers from the University of Chicago, in 1943 [22]. These networks could receive an input, calculate the weighted sum, and return a binary output based on a certain threshold. A neural network consists of layers of these perceptrons interconnected to each other. The field of artificial intelligence stood still for about 12 years, where no institutions, large or small, would accept any projects that involved neural networks. The field started to come back to life in 1985 with the re-discovery of backpropagation which made the task of ”learning” much faster than the previous methods. Backpropagation is considered the workhorse of learning in neural networks [23] due to its significant importance in the function of every single neural network.

Afterwards, a significant advance in this field happened in 1980 by Dr Kunihiko Fukushima, which proposed the Neocognitron [24]. It was an artificial neural network with simple and complex cells. The core idea was that simple cells could detect simpler patterns of an object, and complex cells could detect more complex patterns. Even though this was an outstanding advancement in the field, it was only in 1998 that Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, in their paper [1], implemented the first convolutional neural network, LeNet, which

architecture can be seen in figure 2.1. The *LeNet* network was trained on the MNIST dataset, which consists of hand-written digits and characters, having 60 thousand images for training and 10 thousand images for testing.

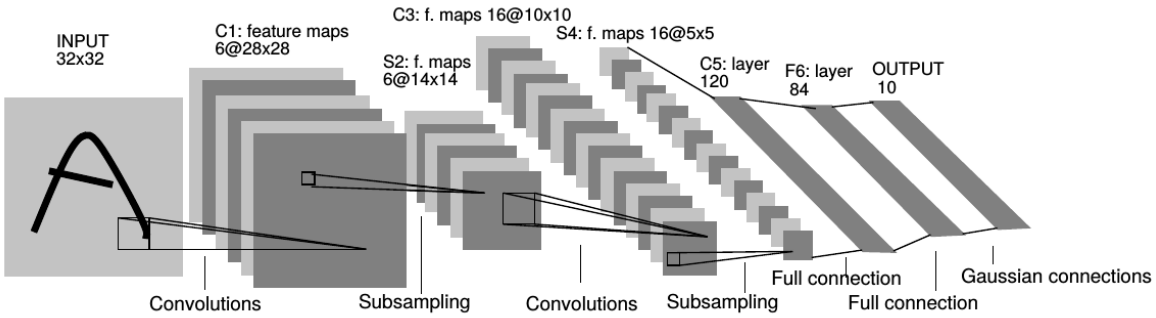

Figure 2.1: LeNet architecture, taken from [1].

Following the previous advancements, CNNs just kept increasing in complexity and were trained with larger and larger datasets. Another significant advance occurred in 2012, when Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton proposed AlexNet. This network consisted of a deep convolutional neural network that was trained using a GPU and achieved an error rate of 15.3%; after this, the use of GPUs to train CNNs became the standard.

2.1.1 Region-based Convolutional Neural Network

After the appearance of more complex convolutional networks that could find the most meaningful features, there was no need anymore to perform manual hand-crafted features for each image in the task of object detection. Afterwards, many proposed architectures used convolutional networks as backbones. The region-based networks will be discussed in more detail in this section, given that most methods for video object recognition use them.
There are multiple versions of the Region-based Convolutional Neural Network (R-CNN) architecture; each one tries to improve the previous one mainly in terms of computational efficiency. The R-CNN family consists of the following architectures:

- R-CNN;
- Fast R-CNN;
- Faster R-CNN;

The most prevalent version – **Faster R-CNN** – was proposed in 2015. The architectures in this family have four main components: a **Region Proposal Algorithm (RPA)**, which generates the possible locations of an object in the image, **bounding boxes**; then, there is the CNN backbone, which extracts features from the image; afterward, there is a classification layer which classifies the image and, finally, there is also a regression layer to adjust the parameters of the bounding box to fit the object.
Note the description of the following terms:

- Proposed Region – is a region that has a probability of containing an object;

- Anchor Boxes – are a predefined set of bounding boxes that are used in the Regional Proposed Network;
- Regional Proposal Network – iterates through the previously mentioned anchor boxes in different image locations and calculates the IoU. Based on its results, the RPN classifies each image region as either background, meaning that there is no object, or as foreground, identifying that an object is indeed present in that region.

The first proposed architecture, R-CNN, starts by generating region proposals using edge boxes. Moreover, the CNN runs for each of the proposed regions, which can be costly given that there can be around 2000 proposed regions. Each of these proposed regions' bounding boxes is adjusted with a SVM model, as shown in figure 2.2. Due to this, this architecture is considered the slowest model of the family.

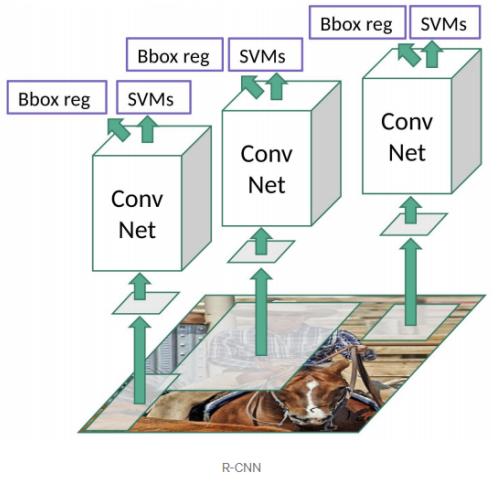

Figure 2.2: R-CNN architecture, taken from [2].

The subsequent architecture that followed was, Fast R-CNN, shown in figure 2.3, also uses the EdgeBoxes algorithm; however, the underline is in the CNN application. This model uses the CNN once on the whole image and only adjusts the proposed regions to fit the object. This approach is around 25 times faster than the original R-CNN.

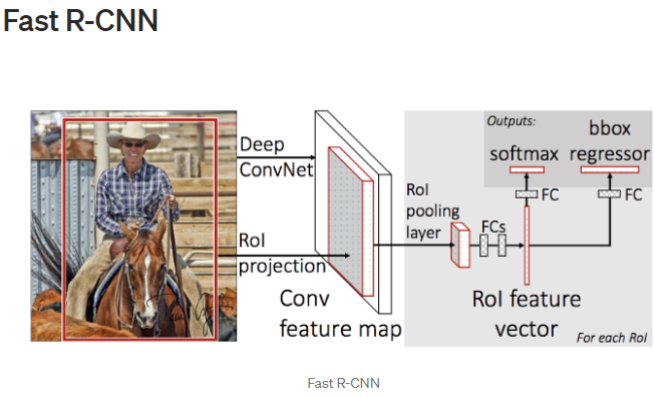

Figure 2.3: Fast R-CNN architecture, taken from [2].

Lastly, the architecture Faster Faster-R-CNN makes use of a Region Proposal Network (RPN) which is used to generate the region proposals instead of having to use an external algorithm such as *EdgeBoxes* [25], as illustrated in figure 2.4.

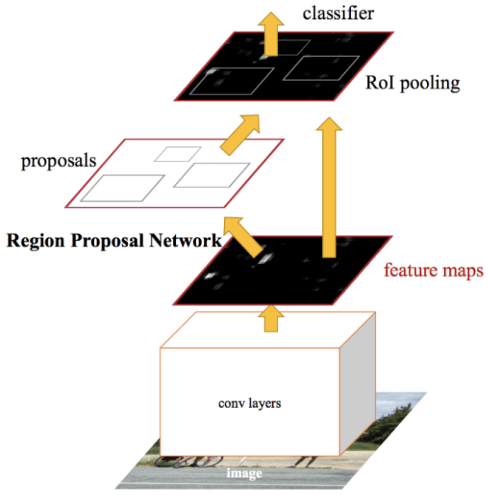

Figure 2.4: Faster-R-CNN architecture, taken from [3].

The output of this RPN is given to a Region of Interest pooling, which will be responsible for reducing the size of these maps to a fixed size; this is since the anchor boxes have different sizes, and so the feature maps come in more than one size. Finally, these features maps, already flattened, are then given as input for both the regressor, which refines the bounding boxes and the classifier, which classifies the image as an object or background. This architecture is around 250 times faster than the original architecture and 100 times faster than the Fast-RCNN.

2.1.2 *EfficientNet* and *EfficientDet*

Most state-of-the-art object detectors correlate accuracy and computational resources, meaning that models sacrifice part of their accuracy if they are not very complex. This is not a viable long-term solution since, in modern applications, these models must run on devices that sometimes do not have such advanced hardware. Thus, most of the systems that involve CNNs are first developed with a fixed budget and then are improved later if needed. To avoid this correlation between accuracy and computational resources, in 2019, *EfficientNet* [4] was proposed with the intent to scale these networks. Which was done by balancing all the network's dimensions (*i.e.,* width, depth, resolution) which was done by scaling each one with a fixed ratio; the following figure, 2.5, illustrates the proposed scaling methods.

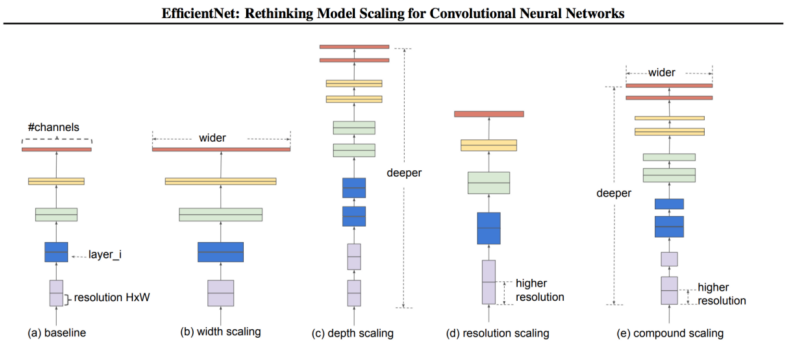

Figure 2.5: EfficientNet scaling methods, where (a) is the baseline model, (b),(c), and (d) only focus on scaling one dimension, and (e) is the proposed method for uniformly scaling all three dimensions with a fixed ratio. Image taken from [4].

The core idea is that scaling each of these dimensions provides a significant advantage:

- **Depth** – by increasing the depth of a network, we are increasing the number of convolutional layers and thus, allowing the model to learn more complex features.

- **Width** – by scaling the width of the network, it can extract more specific features.

- **Resolution** – by scaling the input resolution, we are giving more detail to the network, which allows improves the model to find more specific patterns (*e.g.*, smaller objects).

Even though scaling each dimension has been proven to improve the accuracy, these gains diminish in larger models. This is not the case when scaling the three dimensions together since it is much more beneficial for the model [4]. This method would then become the base for another proposed method, *EfficientDet* [6] which, by itself, is also a foundation for another architecture that will be discussed further in 2.5. The main difference between *EfficientNet* and *EfficientDet* is that the latter has the addition of a Bi-directional Feature Pyramid Network.

Feature Pyramid Networks were first proposed in 2016 by Tsung Yi Lin proposed in [5]. It provided a more efficient alternative to the previously used feature pyramids. The idea is that to detect objects at different scale levels, we must obtain the image's feature maps at those scales. Primarily, this technique would extract each image's feature maps individually for each image scale; however, it is a very costly process in terms of memory and a highly time-consuming process. FPNs propose a pyramid of feature maps in a top-down architecture that aims to obtain high-level semantic features for each different scale. The idea is that, since object detectors only use feature maps from the highest layers, given that those have the most semantic value, it is also crucial to consider those bottom layers for specific scenarios (*i.e.*, as previously mentioned, for tiny objects' detection).

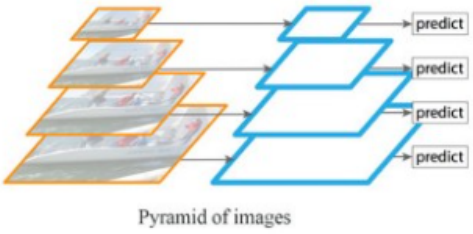

Figure 2.6: Pyramid of Images. Taken from [5].

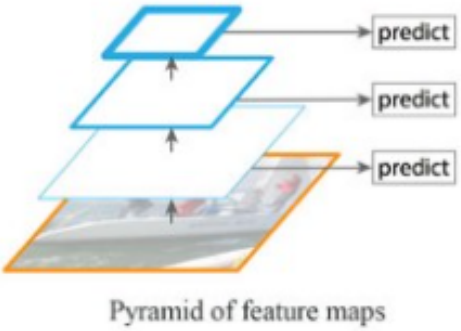

Figure 2.7: Pyramid of Feature Maps. Taken from [5].

Other approaches, such as [26], propose an additional bottom-up pathway in the network's architecture that aims to find the best feature network topology with a neural architecture. Furthermore, and as already stated previously in this section, the *EfficientDet* uses another type of FPN, the Bi-FPN. The previous networks are limited by either having only one pathway and thus, only having one information flow. This is addressed in other proposals by adding cross-scale connections, but these require heavy computation resources. On the other hand, Bi-FPN proposes a set of optimizations for this type of cross-scale connection, such as only maintaining edges that possess more than one input. Then, the network considers each path (*i.e.*, top-bottom and bottom-top pathways) as one feature network layer [6]. Consider the following figure, 2.8 that represents the *EfficientDet* architecture.

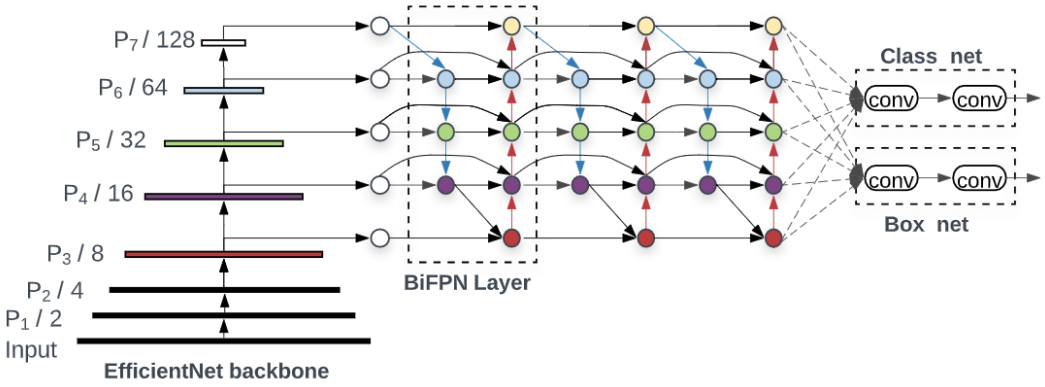

Figure 2.8: Proposed *EfficientDet* architecture, taken from [6].

As illustrated in the previous figure, *EfficientDet* proposes an architecture where the backbone is an *EfficientNet* pre-trained on the *ImageNet* dataset. Afterwards, the last chosen features (P3, P4, P5, P6, P7) are then given as input to the Bi-FPN which is responsible for aggregating these features bidirectionally; finally, after the feature's fusion, these are fed to a class-box network to perform object class classification and box regression, respectively.

2.2 Object Detection in Aerial Data

The main difference between standard images and aerial images is the scale. In aerial images, considering a ”bird’s view”, the objects are much smaller than usual, making the object to detect smaller and, thus, harder to detect. Additionally, objects have more orientation changes in many aerial images, increasing the problem’s complexity even further. State-of-the-art methods for object detection are based on finding the best features to characterize an object and then performing classification and detection based on those features. However, if the object is tiny, then the number of features that can be analyzed and learned is much small, making it harder for these models to find suitable features to detect those objects. Given these technique difficulties, a well-labelled and large dataset is a must when training any model to perform this task. In terms of object detection, there are already many relevant datasets such as PascalVOC [27], COCO [28] and *ImageNet* [29]. When it comes to aerial object detection, there are no such datasets.
The main one that is used and mentioned in section 3.1 is the most complete of them. The same authors that proposed the DOTA [7] dataset also selected a few state-of-the-art object detection models (*i.e., YoloV2 [30], Region-based Fully Convolutional Neural Network (R-FCN) [31], Faster-R-CNN and SSD [32]*) and evaluated them on their dataset, the results are shown in the following figure, 2.9.

	YOLOv2 [25]	R-FCN [4]	FR-H [26]	SSD [16]
Plane	76.9	81.01	80.32	57.85
BD	33.87	58.96	77.55	32.79
Bridge	22.73	31.64	32.86	16.14
GTF	34.88	58.97	68.13	18.67
SV	38.73	49.77	53.66	0.05
LV	32.02	45.04	52.49	36.93
Ship	52.37	49.29	50.04	24.74
TC	61.65	68.99	90.41	81.16
BC	48.54	52.07	75.05	25.1
ST	33.91	67.42	59.59	47.47
SBF	29.27	41.83	57	11.22
RA	36.83	51.44	49.81	31.53
Harbor	36.44	45.15	61.69	14.12
SP	38.26	53.3	56.46	9.09
HC	11.61	33.89	41.85	0
Avg.	39.2	52.58	60.46	29.86

Figure 2.9: State-of-the-art object detection models’ results on the DOTA dataset, taken from [7].

The results shown in the previous table should be anticipated right away. Large objects such as harbours, planes, swimming pools, and tennis courts obtain a decent average precision, whereas smaller objects such as ships and vehicles obtain below-average results. This illustrates how these state-of-the-art object detectors might perform well for typical computer vision problems. However, when confronted with aerial images, their performance is highly affected by the difference in their characteristics.

A point still needs to be addressed regarding aerial image datasets. There are different methods of obtaining aerial data, yet, the most commonly used is to obtain images from public satellites. The main problem with this method is that there is no control over the actual camera and its characteristics. Even though it is a straightforward and uncomplicated method to

obtain aerial images, it might not be the best for specific scenarios. Another way of obtaining aerial data is through drones, which are, nowadays, compelling and can possess different types of cameras (*e.g., RGB*, sensor, thermal and many more), and we have control over the height, camera settings and route. Given this, there are also disadvantages when recording our dataset with drones, such as:

- **Low battery life** – The Phantom V4 Pro, which was used to record the data used for this project, can only last for 20 minutes in the air recording;
- **Manual flight** – There can be danger zones where the drone can crash if not guided carefully;
- **Costly** – It is time-consuming, and there is the need to travel to the places where the recording scenarios must happen.

Furthermore, given the problem tackled in this work, obtaining aerial data in the forest with drones is a demanding task. There are many zones with high tree density, and the drone must be constantly looked after to adapt its height to prevent it from crashing. Another critical factor is that recording many optimistic scenarios (*i.e.,* cars driving by and people walking, running, or riding bicycles) requires additional personnel to perform these scenarios since, most of the time, there is not much activity.
After the release of DOTA, a few object detection methods in aerial images have been developed that mainly focus on the objects' rotation. Because a typical CNN does not model orientation variation, and given that this is a crucial factor in detecting objects in aerial images, there is the need to develop models that consider rotation. With this being said, the *reDet* model [8], and *CoRR* [33] are the main state-of-the-art methods that focus on modeling these variations of orientation towards object detection in aerial images. The following figure 2.10, illustrates the architecture of the method proposed in [8], *ReDet*: A Rotation-equivariant Detector for Aerial Object Detection.

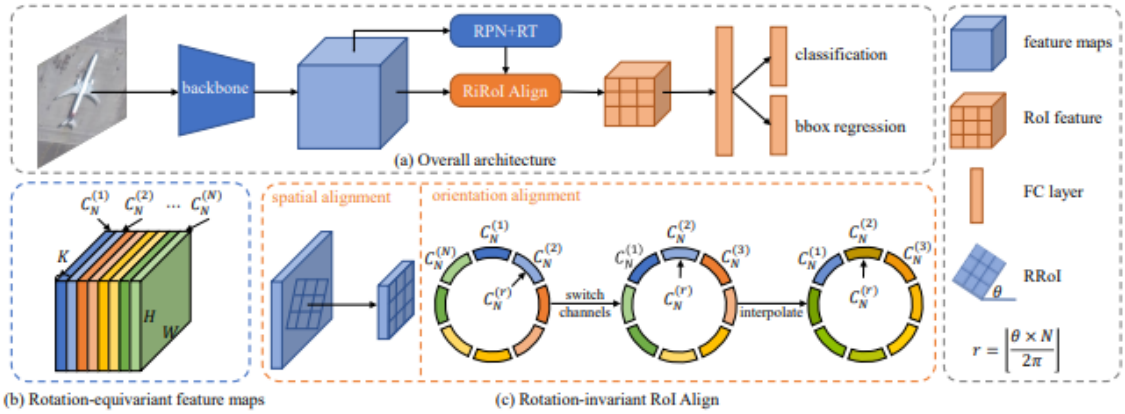

Figure 2.10: *ReDet* architecture, taken from [8].

Note that, as the previous figure shows, these methods possess a network that is specific for extracting the objects' rotation features (*i.e.,* rotation-equivariant features) which are then

fed to a RPN and, subsequently, to a region of interest transformer. The comparison results of this model with the other state-of-the-art object detection models are shown in figure 2.11.

method	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC	CC	mAP
OBB results:																	
RetinaNet-O [18]	71.43	77.64	42.12	64.65	44.53	56.79	73.31	90.84	76.02	59.96	46.95	69.24	59.65	64.52	48.06	0.83	59.16
FR-O [27]	71.89	74.47	44.45	59.87	51.28	68.98	79.37	90.78	77.38	67.50	47.75	69.72	61.22	65.28	60.47	1.54	62.00
Mask R-CNN [11]	76.84	73.51	49.90	57.80	51.31	71.34	79.75	90.46	74.21	66.07	46.21	70.61	63.07	64.46	57.81	9.42	62.67
HTC [2]	77.80	73.67	51.40	63.99	51.54	73.31	80.31	90.48	75.12	67.34	48.51	70.63	64.84	64.48	55.87	5.15	63.40
OWSR* [15]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	74.90
ReDet (Ours)	79.20	82.81	51.92	71.41	52.38	75.73	80.92	90.83	75.81	68.64	49.29	72.03	73.36	70.55	63.33	11.53	66.86
ReDet* (Ours)	88.51	86.45	61.23	81.20	67.60	83.65	90.00	90.86	84.30	75.33	71.49	72.06	78.32	74.73	76.10	46.98	**76.80**
HBB results:																	
RetinaNet-O [18]	71.66	77.22	48.71	65.16	49.48	69.64	79.21	90.84	77.21	61.03	47.30	68.69	67.22	74.48	46.16	5.78	62.49
FR-O [27]	71.91	71.60	50.58	61.95	51.99	71.05	80.16	90.78	77.16	67.66	47.93	69.35	69.51	74.40	60.33	5.17	63.85
HTC [2]	78.41	74.41	53.41	63.17	52.45	63.56	79.89	90.34	75.17	67.64	48.44	69.94	72.13	74.02	56.42	12.14	64.47
Mask R-CNN [11]	78.36	77.41	53.36	56.94	52.17	63.60	79.74	90.31	74.28	66.41	45.49	71.32	70.77	73.87	61.49	17.11	64.54
OWSR* [15]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	77.90
ReDet (Ours)	79.51	82.63	53.81	69.82	52.76	75.64	87.82	90.83	75.81	68.78	49.11	71.65	75.57	75.17	58.29	15.36	67.66
ReDet* (Ours)	88.68	86.57	61.93	81.20	73.71	83.59	90.06	90.86	84.30	75.56	71.55	71.86	83.93	80.38	75.62	49.55	**78.08**

Figure 2.11: *ReDet* comparison results, taken from [8].

The significant difference between the results obtained with the *ReDet* networks and the results obtained by state-of-the-art methods is in the smaller objects. While using these state-of-the-art detectors, objects such as vehicles, helicopters, ships, and container cranes, the results are not favourable due to the characteristics of the object; however, the results are significantly better when using a method that considers the objects' rotation features.

Further, the few aerial image datasets found in search of similar projects will be described and analyzed based on the number of instances, categories and images.

2.3 Video Object Detection

If obtaining large quality image databases is difficult, video databases become even more challenging. There are already a few available databases that allow researchers to develop and evaluate new architectures towards video object detection, such as *ImageNet VID* 2015 [34] and *Cityscapes* [35]. While these two datasets are extensive and have quality data, the variety of scenarios is still minimal.

CNNs have been showing tremendous results in various tasks. In particular, one technique mainly serves as the foundation for the feature propagation methods approached in this subsection: a CNN that can perform optical flow estimation. Optical flow estimation has been the target of different approaches with improved methods since it was proposed by Berthold K.P.Horn [36]. In contrast, all of these methods' parameters must be inserted manually. Other techniques have been proposed that apply machine learning methods to learn optical flow, both using supervised learning methods ([37]) and also unsupervised methods ([38]).

A relevant contribution for the following papers is *FlowNet* [9] which is proposed in two architectures that are shown in the following figure,2.12.

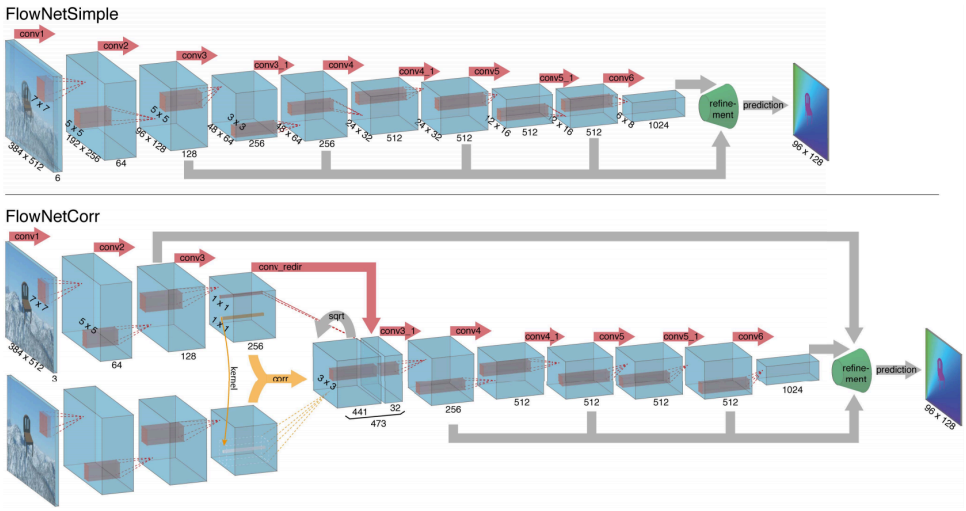

Figure 2.12: *FlowNetSimple* architecture in the top and, *FLOWNETCORR* shown in the bottom, taken from [9].

The first architecture, *FlowNetSimple*, is the most basic of the two and consists of stacking the two input images together and then feeding them through a CNN. This is a relatively simple approach, but if the network is extensive enough and provided with quality data, it has been proven to achieve great results. On the other hand, the second architecture, *FlowNetCorr*, initially has two separate streams responsible for extracting meaningful features for each input image. Later, both the features extracted from each image are combined, passing through a final CNN, much like the simple architecture.

2.3.1 Deep Feature Flow for Video Recognition

The task of object recognition in images is often performed with the help of deep learning architectures; however, even though these architectures achieve great results, it is also essential to consider the large number of computational resources they require. This problem only aggravates when the tasks are performed on video. This is one of the reasons why it is essential to design new architectures that achieve great results and are also efficient in terms of computational resources.

Additionally, Paper [11] proposes *Deep Feature Flow*, a flow-based method that aims to solve these computational boundaries by taking advantage of the similarity between adjacent frames. The proposed architecture, illustrated in figure 2.13, consists of having two separate networks (*i.e.,* two-stream network). The first one is a *ResNet-101* which is responsible for extracting the key-frames feature maps; the second one is a *FlowNet* [9] which is used to warp the key-frames feature map with non-key frames.

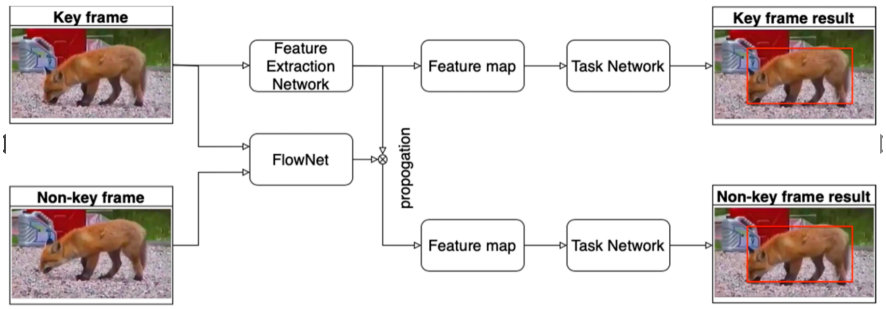

Figure 2.13: Deep Feature Flow feature propagation architecture, taken from [10].

There is a significant similarity between adjacent frames in videos, and thus, there is an equal similarity between each frame's feature maps. Based on this, the authors propose not to extract feature maps from every frame but only from those which are distinct and then propagate the features of these key-frame, feature propagation. Note that, after extracting features maps from the key-frame, these features are then propagated through a flow field to the next **n** frames, as shown in figure 2.14.

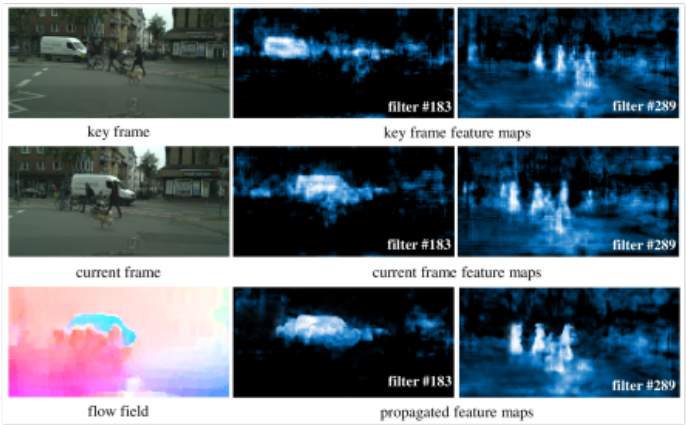

Figure 2.14: Proposed flow field mechanism, taken from [11].

A crucial factor in this method is deciding which key frames and how long their feature maps should be propagated. In the paper's experiments, finding these crucial frames was done by setting a constant interval of frames in which this feature extraction operation was performed. This method works for this scenario, but it is highly volatile given that every scenario is different in terms of frames' content. In order to further improve this architecture, there is the need to find a more reliable way to find the best key frames and propagate their feature until there is a frame where the information highly shifts. Note that there will be experiments conducted to evaluate which value to choose between key-frames by analyzing the particular scenario for this project.

2.3.2 Flow Guided Feature Aggregation

In videos, many aspects make the task of object detection harder, such as motion blur, video defocus, part occlusion, and rare poses. In this paper [12], the core idea is to aggregate feature maps from adjacent frames to maintain crucial information so that the model can easily detect a given object in more specific scenarios. The following figure, 2.15 illustrates the different frames' features warping.

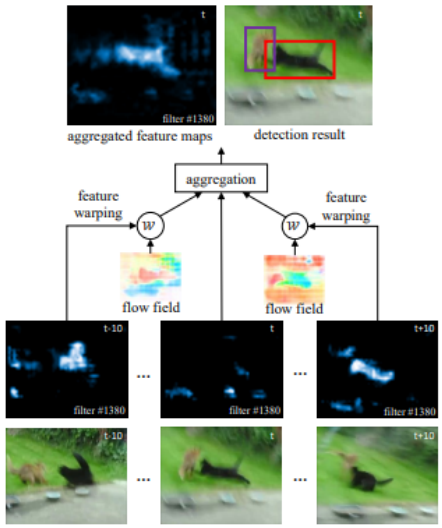

Figure 2.15: Flow Guided Feature Aggregation method proposed, taken from [12].

The previous figure illustrates why, in some scenarios, it is essential to consider the neighbour frames when detecting an object. In the figure's middle frame, the image is blurred; thus, the feature maps extracted from it have low activations. An architecture that only considers that frame's feature maps would almost certainly not detect any object in the frame. Nevertheless, by considering the frame's neighbours, which have high activations (left and right images on the figure), the architecture can detect that there is indeed an object in the image. The overall architecture of this model is shown in figure 2.16.

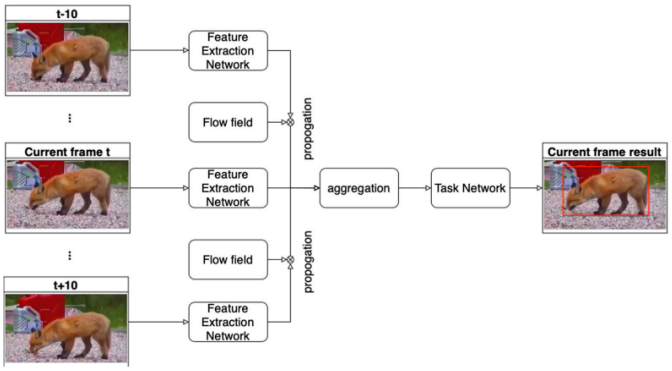

Figure 2.16: Flow Guided Feature Aggregation architecture, taken from [12].

These features are extracted using a *ResNet-101*, also pre-trained on *ImageNet*; afterwards, it is calculated the flow estimation between the reference frame and its neighbours by us-

ing *FlowNet*, which is called feature warping. Proceeding this process, these sets of feature maps are then aggregated as a way to combine their information (*i.e.,* different object poses, lightning variances, and many more variables). Logically, since every neighbour frame possesses different information levels than the reference frame, they should not have the same impact when aggregated. The authors propose to feed these features through a sub-network to assign different weights to the different feature maps. This is done by using a sub-network based on cosine normalization to compute these weights; these weights are then used to aggregate the features accordingly to their importance. Following the features' aggregation, these are then fed to the detection task network, similarly to the previous model, a R-CNN which will then perform regression and classification.

2.4 Semantic Segmentation

Image segmentation is a computer vision problem aiming to outline an object's boundaries precisely. Instead of finding a region of pixels where the object might be, we want to label each pixel of an image with its respective class. There are two main sub-fields in image segmentation: semantic and instance segmentation. The first is only responsible for finding each pixel's class. The latter is responsible for classifying each pixel according to its class and identifying every pixel belonging to a particular instance. On the other hand, when using instance segmentation, it not only classifies and outlines both dogs but also identifies each as a unique instance of that class, as shown in figure 2.17).

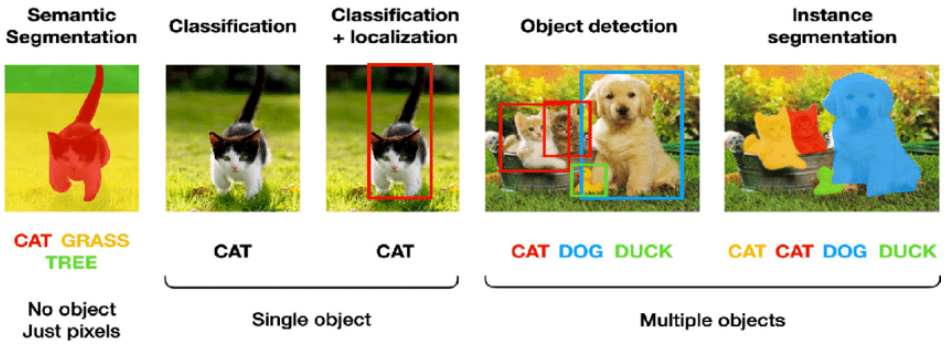

Figure 2.17: Illustration of the different computer vision tasks, taken from [13].

With this being said, and now considering this work's scenario, wild forests have a few possible terrains, such as dirt roads, forestation zones, gravel paths (*i.e.,* terrain mainly composed of rocks), and tar roads, as shown in figure 2.18. The main idea is that by applying instance segmentation, we can outline these terrains on any dataset image, figure 2.19.

Figure 2.18: Dataset's example frame.

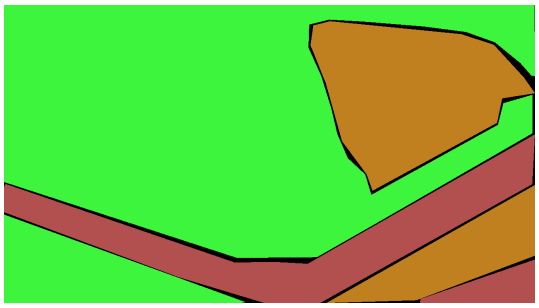

Figure 2.19: Terrain segmentation mask for the example frame.

The terrains segmentation colors is as follows:

- **Forest** – Green;
- **Gravel** – Orange;
- **Dirt Road** – Red;
- **Tar Road** – Gray.

This type of data is intriguing because when experimenting with different state-of-the-art methods, thoroughly analyzing the results is crucial to making the correct conclusions. Hence, by segmenting an image and outlining all terrains, we can then use this information to analyze the model's performance on each one (*i.e.,* conclude if the model performs better on a specific type of terrain). Before CNNs, image segmentation was done using SVMs, random forests, and k-means clustering; however, after the appearance of convolutional networks, methods like U-Net [14] started to surge which main goal was to consider different features with different levels of maturation.

Olaf Ronneberger proposed the *U-NET* in 2015, and the main idea is that there are two paths in the network. These paths are also known as the encoder and the decoder. The first is used to find the image's context, and it consists of only convolutional and max-pooling layers where it will learn the representation of the different classes. On the other hand, the expansion path is responsible for up-sampling these features back to the original image's size; this path uses transposed convolutions. It also has to skip connections so the model can consider features with different maturation levels.

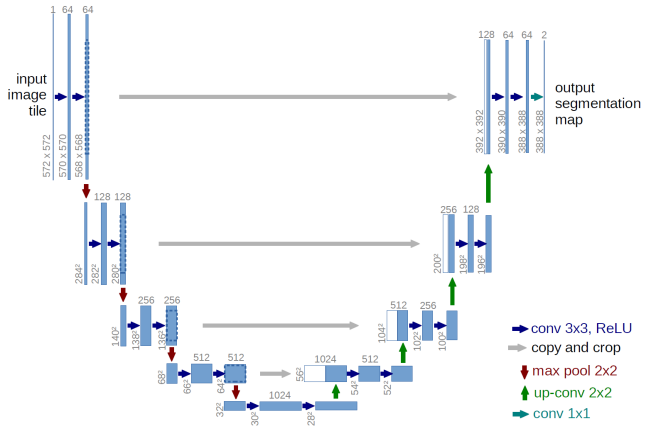

Figure 2.20: Proposed *U-NET* architecture, taken from [14].

Note that the final output of this network is an image of the same size as the original's; however, the depth of the output corresponds to the number of the problem's classes, and every pixel has its corresponding class value in that class's depth.
After this architecture was proposed, many methods were proposed, [39] [40] [41]; however, given that most of these architectures were proposed around 2015 and 2016, in this project, a more contemporary state-of-the-art architecture was used, the High Resolution Network (HR-Net) [15]. It was proposed in 2019 by Ke Sun, and it relies on the fact that high-resolution solid representations are an essential part of region labelling problems. Thus, making it crucial that systems can main high-level representations of the original data throughout the whole network's flow. The authors found that to maintain these representations by connecting different convolutions paths with different resolutions, as shown in figure 2.21.

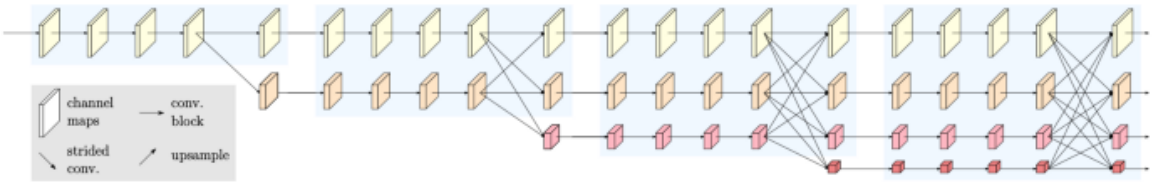

Figure 2.21: Proposed HR-Net architecture, taken from [15].

As the previous figure illustrates, the first feature maps have a high resolution. However, as the depth of the networks starts to grow, other blocks are created that possess the exact resolution as the previous block, and more miniature representations are created in parallel. Every feature map at the end of a block is fully connected to the following block's feature maps, also known as multi-resolution convolution. This makes it so that the network keeps that high-resolution information throughout the process. The final feature maps possess more semantic value by performing convolutional with these different resolutions.

2.5 Self-Supervised Learning Towards Object Detection

Supervised learning methods require an immense amount of data to achieve good results. It is challenging to obtain such amounts of data in distinct scenarios, but the task of annotating

all that data is also an arduous, very costly, and time-consuming process. Self-supervised learning is a form of unsupervised learning that removes the need for labelled data by giving the task of learning useful representations for that unlabelled data to the method; this makes it so that these models can label and categorize the data themselves. The most recent and promising advances in this field are transformers. For example, BERT [42] is a self-supervised based transformer applied to Natural Language Processing which is first trained on unlabelled data and then fine-tuned on a smaller set of labelled data. Self-supervised learning started to get more attention around 1990 due to its potential applications in Natural Language Processing problems. There are two main types of methods used in this field:

- **Self-training** – This method's core idea was to, primarily, train an initial model with available labelled data and then, with the help of this previous one, train a secondary model with unlabelled data [43].
- **Structural Learning** – In this method, only unlabeled data is used to find simpler hypothesis spaces by modelling the regularities of this unlabelled data [44].

Afterward, self-supervised learning also started being applied to computer vision and, an exciting technique in this field is **generative models** (*e.g.,* Generative Adversarial Network). A GAN is an architecture that combines two neural networks: the generator and the discriminator, which compete against each other continuously as if they are playing a game, as shown in figure 2.22. On the one hand, given a random noise, the generator network generates new synthetic data instances, but that needs to be as real as possible to fool the discriminator. On the other hand, the discriminator evaluates the image generated and tries to evaluate it as real or fake. The core idea is that the discriminator can find a correlation between the data and the corresponding output [45] and the generator reaches a point where it generates images that are as real as possible. One of these methods' problems is that they are challenging to train.

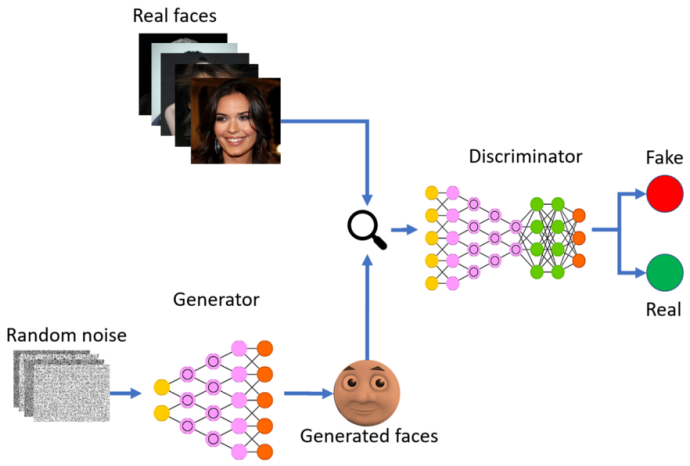

Figure 2.22: Basic GAN architecture, taken from [16].

Another method used in self-supervised learning is **contrastive learning**, proposed in 2018 by Aaron van den Oord [46]. The main idea behind this method is to identify and learn

representations by contrasting positive and negative data samples. This is important because we do not want the model to only identify the objects with characteristics like the ones it was trained on; it must generalize. We, as humans, learn to identify objects by specific characteristics similar to most of that same object's type (*e.g.,* identity a car no matter the colour, shape, or brand), and thus, our power of generalization is perfect. Given this, the model is learning high-level features that distinguish it from other objects by using contrastive learning. A state-of-the-art model developed using this technique is **A Simple Framework for Contrastive Learning of Visual Representations** [47] proposed by Ting Chen in 2020, which architecture can be seen in figure 2.23.

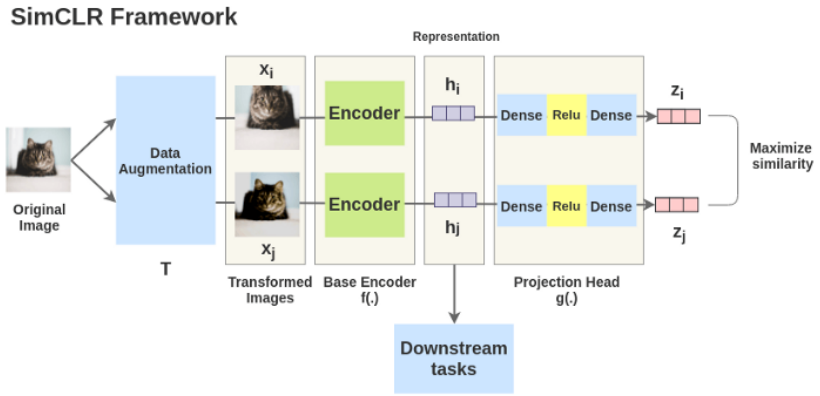

Figure 2.23: SimCLR architecture, taken from [17].

As illustrated in the previous figure, the concept is to apply data augmentation first, get the images' representation by using an encoder, and, afterwards, maximize the similarity between these representations of the same image. The model's generalization can improve and learn the right features to distinguish between objects. Another critical aspect in these networks is transfer learning, which consists of using knowledge from a model, that was previously trained for new tasks and purposes. This is ideal given how much time it takes to train models with a high complexity from scratch. In "CNN Features off-the-shelf: an Astounding Baseline for Recognition" [48], it has been proven that the lower layers of a network can represent the general purpose of a dataset. In contrast, the higher layers focus more on the specific task of the problem.

Note that transfer learning is fundamental to this field, given that self-supervised learning techniques improve their performance when the model is more complex [49]. If we can re-utilize the weights from another big model, then there is no need to spend weeks or maybe months training another one from scratch. While transfer learning consists of transferring the weights from one model to another, there is also another way of transferring information between models, knowledge distillation. In this method, we are not looking to transfer the weights but instead, the representational learning from an extensive network (*i.e.,* teacher model) to a smaller one (*i.e.,* student model). This is also extremely important given that there are specific real-world applications where running such large models is not an option (*e.g.,* mobile phones and embedded devices). This transfer is done by having the teacher model pre-trained on a large-scale dataset and supervising the student model's training. This makes it so that the student network's task is to obtain the most similar output to the

teacher's, as illustrated in figure 2.24.

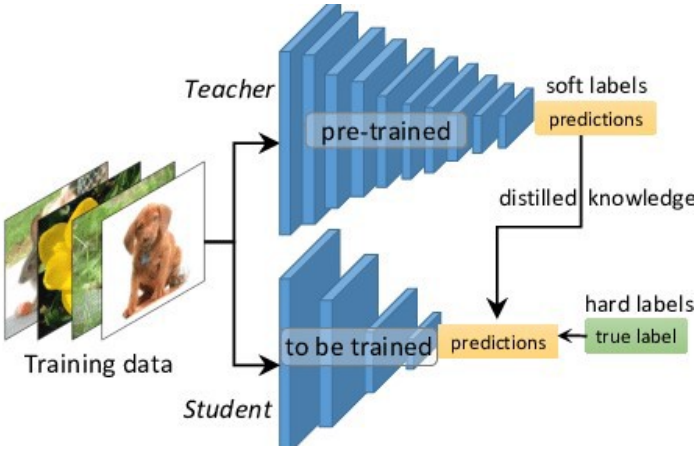

Figure 2.24: Knowledge distillation method's architecture, taken from [18].

When we look more at the specific task approached in this project, object detection methods such as SimCLR and MoCo fail to achieve good results due to being explicitly engineered to perform object classification problems. A few methods have been proposed ([50], [51]) that try to use these techniques in a more object detection-oriented approach. Furthermore, following this project's interests, the self-supervised method that will be analyzed in more detail is **There is More than Meets the Eye: Self-Supervised Multi-Object Detection and Tracking with Sound by Distilling Multimodal Knowledge** [19]. Considering that humans have an excellent perception of their surroundings, whether night or day, this brings up a discussion on how we can design systems with the same level of perception. The main reason for this fantastic perception of humans is that we consider every information from different sensory modalities in the body. This means that, in a scenario where we do not see a person coming from behind us, we can still rely on the sound or smell to conclude that there is a person. Additionally, it is proven that sound attributes that come from objects contain a prosperous time and domain frequency. This paper [52] and a few others have already applied methods that focus on leveraging teacher-student learning based on these two modalities, image and sound. Even though these methods already integrate these different modalities, there are still a few problems that they have: they only work for strict scenario settings and single object detection, there is the need for metadata related to the input, and most of these methods train the teacher only on RGB images, which have many variables (*i.e.,* weather, brightness, object scales and many more).

The main innovation in this paper uses three teacher networks, which were trained with three different modalities, to train a student network with the sound modality, shown in figure 2.25. The combination of these three modalities is very beneficial to finding more cues in the video for more specific scenarios (*e.g.,* if the input video was at night, the RGB modality will most likely not provide any useful features for the detection task, making it crucial that other modalities, such as thermal, are also taken into consideration for more useful features).

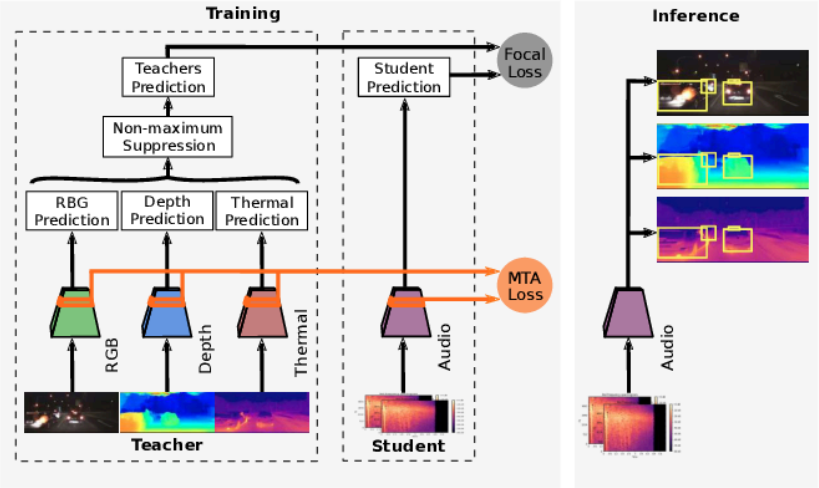

Figure 2.25: *MM-DistillNet* architecture, taken from [19].

Posterior to training the teachers on their specific modalities, these are then used to train the audio student to map the sounds captured from a microphone to bounding box coordinates. The task of finding complementary cues from the different teachers and, at the same time, distilling their object detection knowledge to the student is done by applying the Multi-Teacher Alignment (MTA) loss, which is also a technique proposed by the authors in this paper and will be discussed further into the method's explanations.
The primary aspect of the proposed architecture is the teacher's backbone. The authors chose *EfficientDet-D2* since it was the one that provided the best trade-off between speed and performance. These backbones were trained in different datasets that focused explicitly on their modalities.

- RGB teacher – trained on the *COCO* [28], *PASCAL VOC* [27] and *ImageNet* [29] dataset.
- Depth teacher – trained on the *Argoverse* [53] dataset.
- Thermal teacher – trained on the *FLIR ADAS* [54] dataset.

Note that, as it was explained previously, the *EfficientDet* is not only responsible for extracting the most meaningful features but also for fusing them. Fusing features is a responsibility of the bi-directional weighted feature pyramid.

Any system must have a loss function to learn based on its mistakes. This method has two: Focal Loss and MTA Loss. The first loss is a type of cross-entropy whose goal is to keep the network focused on learning the more hard examples. Equation 2.1 is relative to the focal loss where α corresponds to the weight associated with the more complex examples, γ is a hyper-parameter that controls how much difference in the effort there is between the complex and simpler examples and, where p_t corresponds to the probability of ground-truth class.

$$L_{focal} = \alpha(1 - pt)^{\gamma} * log(pt) \quad (2.1)$$

The latter loss, MTA, is used to perform knowledge distillation from the different teachers to find cues from the intermediate layers of each teacher. This is done by ensuring that specific inner layers of the student network align with the teachers' same layers. We are not interested in using audio, sensor, or thermal modalities in this work. Thus, this architecture will be used only with the RGB modalities further into the document as an attempt to develop an architecture that can create a new model that performs better than the teacher itself.

2.6 Background Subtraction

The main problem that we expect will occur is that these models will often confuse zones of the environment that have similar characteristics to a person or vehicle (*i.e.,* holes and shadows on the terrain). On the one hand, such a problem cannot be easily tackled in the video object detectors' training process. On the other hand, in the *EfficientDet* model, the different fusion of feature maps considers features of more than one scale of the original frame, making it crucial to consider more object details in the training process, even if the object is tiny.
With this being said, the best chance to avoid these miss detections in video object detectors is to apply post-processing to the model's outputs to try and discard these outlier detections. In order to discard any detections, we must first determine an area in which we can consider good detections. Since models often confuse non-moving objects (*i.e.;* ground holes) with moving objects (*i.e.;* people), the main idea would be to determine in which areas objects are moving and only consider models' detections in those areas. Based on this, this section provides an overview of the different types of background subtraction algorithms and for which scenarios they work best.

2.6.1 Static-Camera-Based Background Subtraction Algorithms

The task of background subtraction is mostly used in static camera scenarios, and there are various methods which specialize in this problem, such as [55], [56], [57]. The main idea behind background subtraction is to model the background of the image or video and then detect any changes to the foreground. There are multiple methods to perform this task: average temporal filter, frame difference, and Gaussian filters.

- **Temporal Average Filter –** consists of calculating the median value for each pixel in a sequence of training frames, and then for each new frame, each pixel is compared to the correspondent median value from the training process; if the pixel's value is in between a threshold limit, then it is considered as foreground.

- **Frame Difference –** consists of subtracting the pixel values of two adjacent frames and analyzing the results. If the pixel's subtraction value is lower than a predefined threshold, then the pixel is considered background. On the other hand, if the difference is higher than the threshold, it is considered foreground, as shown in figure 2.26.

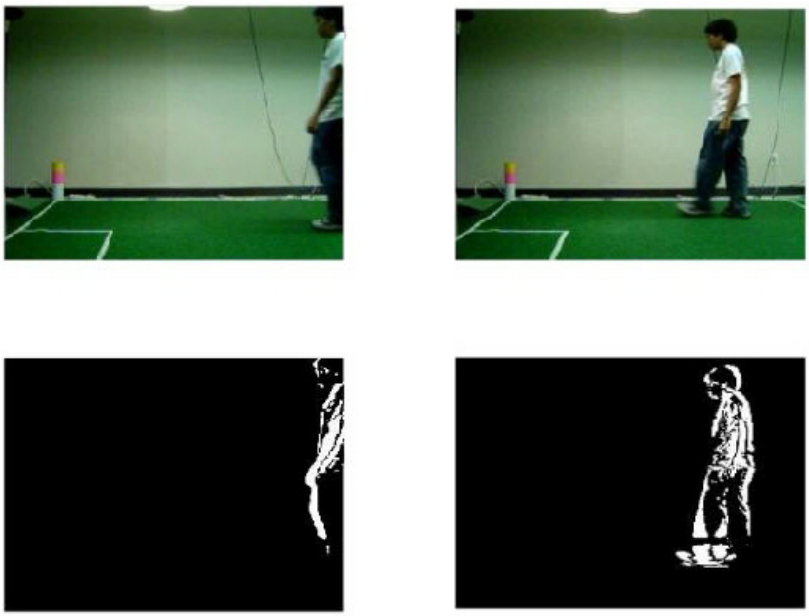

Figure 2.26: Frame difference method example, taken from [20].

- **Mixture of Gaussian Filters [58]** – is a probabilistic algorithm using a set of n predefined Gaussian filters, where each one represents a different cluster. The idea is that different clusters can represent the background and foreground.

These previously mentioned algorithms are based on statistical estimates calculated in the background and then compared to new frames. Additionally, in most of these papers, the experiments are performed on scenarios with a fixed camera with always the same background besides some lightning variations. With this being said, it is essential to note that the scenario being tackled in this thesis consists of a non-static camera (drone), and thus, not only does the foreground change but so does the background. However, there are already algorithms that are developed towards scenarios with a non-static camera where the background changes, such as: [59], [60] and [21].

2.6.2 Moving-Camera-Based Background Subtraction Algorithms

The problem of modelling a background on a sequence of images taken from a moving camera is tackled using innovative approaches to the existing methods. In static-camera-based methods, the primary method for solving this problem consisted of three steps:

1. Align all the frames in a given set;
2. Train a static camera background modelling model using those frames;
3. Inference with unseen frames on them model.

This process works well in scenarios where the sequence of analyzed frames is short. Whenever this sequence starts to get longer and especially with more significant and abrupter camera movements, this process starts to show its flaws. The main problem of these algorithms

developed for static cameras is that they do not maintain any information about the background and foreground of the previous frame, given the problem's nature. With this being said, we focus on one specific method, Moving-camera Background Model via Joint Alignment and Partially-overlapping Local Sub-spaces [21]. The method divides the task of background subtraction into the three following phases:

1. **Unsupervised Joint-Alignment** – is the first phase of the proposed method consisting of finding a fixed global coordinate system relative to a given set of images. Primarily based on an initial set of frames used for training, this first phase uses the A Certifiably Correct Algorithm for Synchronization over the Special Euclidean Group [61], which tries to estimate poses given a few measurements. These transformations are then refined by using an Spatial Transformer Net [62], as shown in figure 2.27.

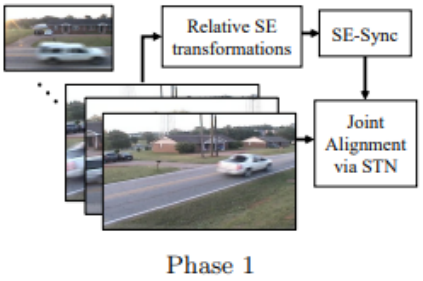

Figure 2.27: Primary phase of the *JA-POLS* method, unsupervised joint alignment. Image taken from [21].

2. **Alignment Prediction and Learning Multiple Background Models** – is the second phase, which consists of using the output of the joint-alignment to perform two different tasks: learn transformation predictions based and learn Partially-overlapping Local Subspaces, as shown in figure 2.28.

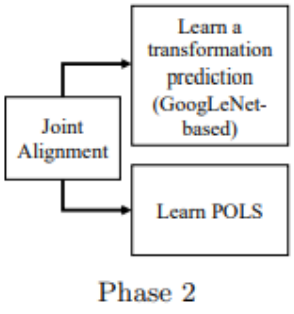

Figure 2.28: Secondary phase of the *JA-POLS* method, which includes both tasks of learning alignment predictions and learning multiple background models over partially-overlapping areas. Image taken from [21].

The first task is done by training a *GoogleNet* [63] model, pre-trained on ImageNet [29] to find the global representation of a given frame. The second task splits the scene into different regions and learns different background modellers for those same regions.

3. **Testing Phase** – is the last phase of the method. Primarily it warps a previously unseen frame to the global scene and afterwards to the local subspace (*i.e.,* the different

scene regions). Finally, we consider both the outputs of the global scene and local regions models and average them to perform background subtraction, as illustrated in 2.29.

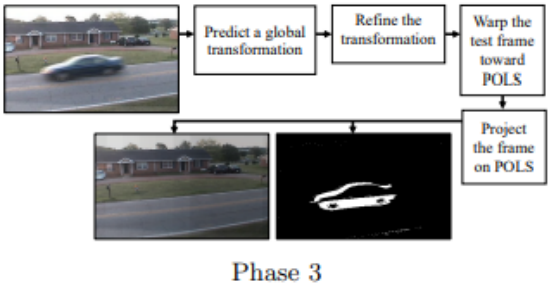

Figure 2.29: Final phase of the *JA-POLS* method, testing phase. Image taken from [21].

There are many methods which also aim to solve the problem of background subtraction on moving cameras problems such as *incPCP-PTI* [64], *DECOLOR* [65], *PRPCA* [66] and Prac-ReProCS [67]. The authors of the *JA-POLS* paper illustrated how their method performed compared to these other methods. Refer to the following figure, 2.30, which illustrates the different background/foreground outputs for an observation frame from the different algorithms.

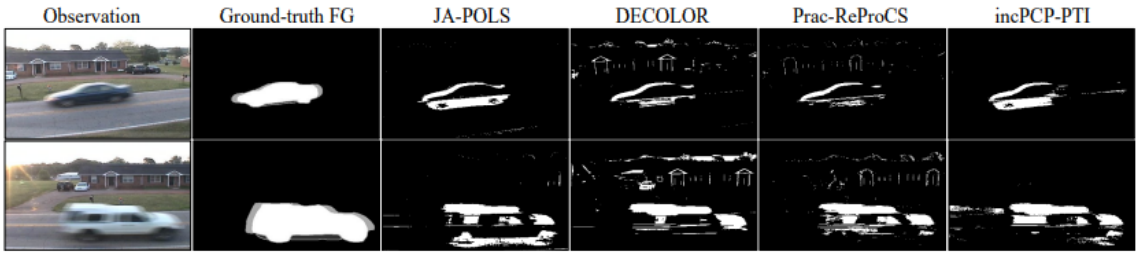

Figure 2.30: Comparison between the different outputs from multiple moving-camera based background subtraction algorithms, taken from [21].

It is noticeable that while all methods can significantly model the background and foreground, the main difference between them is the environmental noise surrounding the object. The *JA-POLS* method, since it considers the global scene of the frame and its local region, can perform a much more localized and precise modelling. This is the main difference between these models. Note that, because of this, the *JA-POLS* method can only focus on the object closer to the camera (*i.e.,* the car) instead of also modelling the house in the back.

2.7 Conclusions

This chapter provides a clear description of the current object detection state-of-the-art related to both supervised and self-supervised learning. A very descriptive analysis was done on object detection in aerial images, showcasing the methods that have been developed for

this particular area even though they are single-frame-based. Moreover, an analysis was performed on state-of-the-art video object detectors based on feature and flow-field propagation to connect the spatial and temporal aspects of the frame sequence.
Furthermore, this chapter also analyzes the evolution and current state-of-the-art semantic segmentation architectures, which will then be used to perform this task on the dataset proposed for this project to analyze the detector's results thoroughly related to every different terrain.
An overview of the state-of-the-art self-supervised methods is presented in this chapter to illustrate how these methods operate and when they can be valuable. Additionally, special attention was given to the architecture mentioned in figure 2.25 since one of the proposed methods in this work is mainly based on it.
Background subtraction is the last topic approached in this chapter. It provides the details that differ between static-camera and moving-camera-based algorithms—illustrating how the different methods work and the specific scenarios in which they should be used.
Every related work presented in this chapter will serve as a foundation to either explain results obtained in the experiments chapter or serve as a foundation for newly proposed methods in this work.

Chapter 3

Serra Dataset

In this chapter, we introduce the dataset collected for this project, explaining the different ideas and methodologies for gathering a new dataset. Furthermore, the dataset's assembling process is explained to provide an overview of the process, from data collection to annotation. Finally, the last section provides an overview of the dataset's specifications.

3.1 Related Sets

Researchers have been trying to fine-tune state-of-the-art models trained on datasets with normal images and adapt them to aerial images in the recent past. However, some factors still become a step-back when trying to achieve great results, such as:

- First and foremost, the scale variation is immense when comparing standard images to aerial images, and this proves to be a significant step-back when adapting other models. Note that the scale difference is not only in terms of spatial resolution (*i.e.,* object is extremely small considering the rest of the image) but also in terms of object categories' scale *i.e.* in standard images, different objects' categories also have different scales, however, in an aerial image, some objects might have the same scale from a top-view).

- Depending on the problem's scenario, there are variations in the number of objects in an image. This means that there are scenarios where one image might have hundreds of objects and others much less, as shown in figure 3.1, creating an unbalanced frequency of instances.

- Lastly, an essential factor in these images is the object's orientation, which can change drastically based on the camera angle, which is also illustrated in 3.1.

For this field to advance, there is the need for extensive and quality datasets that focus on aerial images. This way, researchers and developers can build accurate systems trained and evaluated on a significant amount of data gathered for this problem. Combining a few examples from these typical massive datasets can improve the system's generalization. In 2019, the Dataset for Object Detection in Aerial Images dataset, which consists of 188.282 instances where each one is labelled by scale, orientation, and shape, contains 15 common object categories [7].

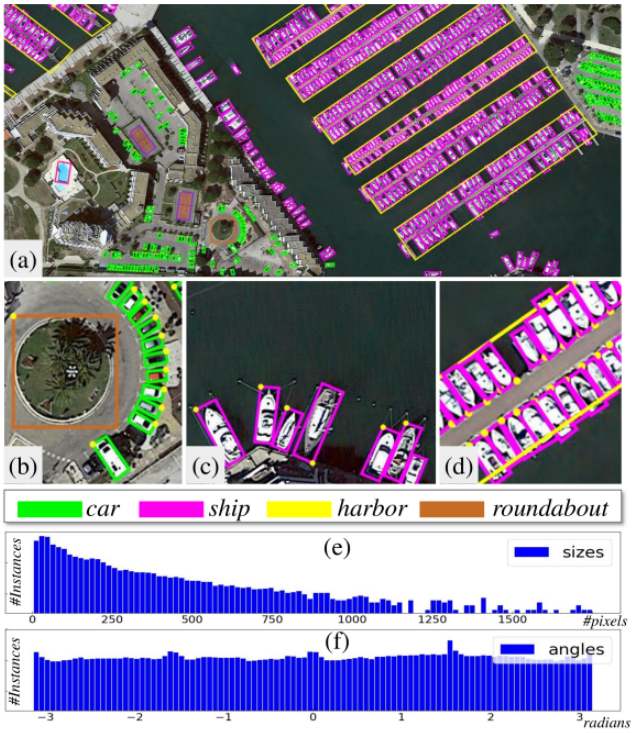

Figure 3.1: DOTA dataset statistics and examples, taken from [7].

Previous to the DOTA dataset, around eight primary datasets focused on aerial images; however, these datasets are not very good regarding the number of instances, classes, and images. These three factors are very low for these eight datasets, as table 3.1 shows. This means that systems trained with these datasets can not be applied to real-world applications since the system would not generalize well enough. With this being said, the DOTA dataset provides a broader range of classes, a much higher number of images, and a higher number of instances.

Table 3.1: Comparison of aerial images datasets with the DOTA dataset, based on [7].

Dataset	# of images	# of instances	# of classes
NWPU VHR-10 [68]	800	3651	10
SZTAKI-INRIA [69]	9	665	1
TAS [70]	30	1319	1
COWC [71]	53	32716	1
VEDAI [72]	1268	2950	3
ICAS-AOD [73]	1510	14596	2
HRSC2016 [74]	1061	2976	1
3k Vehicle Detection [75]	20	14235	2
DOTA [7]	**2806**	**188282**	**14**

After the original release of this data, two more releases added something new to it:

- **DOTA v1.5** – this new version had the tiny objects also annotated, which increased the number of instances to more than 400 thousand.
- **Dota v2.0** – In this latest version, additional images were collected from Google Earth and GF-2 Satellite. This led to an immense increase in the number of images, 11,268.

Likewise, the number of instances also increased to 1,793,658, and the number of categories was now 18.

At last, it is essential to showcase why there was the need to gather the dataset instead of simply using one of the previously mentioned. Most of these datasets were gathered through satellite images; none of them is in a scenario of wild forests like the one intended for this project. Most of the datasets are mainly focused on vehicles and building detection. Furthermore, the problem with these datasets is that they consist primarily of images from different places with different objects with no specific scenario. Moreover, no video datasets focused on forest scenarios were also a downside, given how much information can be gained from analyzing temporal information since objects in aerial images have low spatial information. With this being said, it was decided that it would be more exciting and valuable to gather our dataset with a drone and in a wild forest scenario focused on detecting people and vehicles from aerial videos.

3.2 Contextualization

As part of the development of this thesis, a dataset was gathered that focuses on aerial image data on wild forest scenarios. The main idea for this dataset was to detect possible criminal activities such as causing a wildfire and carrying possible dangerous objects (*i.e.,* lighters, cigarettes, tanks of gasoline). However, these are extremely hard to capture due to their small size and the fact that these videos were gathered through a drone that would be more than 50 meters high; these objects would not be significantly visible.
At the project's initial stage, there was a discussion on whether to use cameras or drones to record the dataset. The question was about which one would bring the most benefits in the long term and benefits in terms of data quality. Given that this specific scenario was the forests, the first approach was to speak with the captain of Covilhã's Fire Department and ask if any cameras would be available to collect the data. The answer was that the forest is filled with cameras, which detect unwanted people in forbidden sites, people setting up fires, and general forest surveillance. We sent an email asking permission to access some of these cameras and use them in the data collection process, but there was no response, and thus, this idea was discarded. After everything was considered, the drone seemed the best option. It also offered a few more advantages when compared to the cameras: complete control of the recorded scenario, the possibility to control the height and camera angles, and, as a long-term solution, it is more likely to have a few drones covering a given forest than thousands of cameras spread throughout the whole zone.

3.3 Dataset Development

The drone used for the data collection process was *DJI Phantom 4 Pro*, illustrated in figure 3.2, provided by the University of *Beira Interior*.

Figure 3.2: *Phantom DJI 4 Pro*.

The following table shows the aircraft and stabilizer specifications for this drone:

Table 3.2: *Phantom DJI 4 Pro* specifications.

Aircraft	
Weight (g)	1388
Max Speed	S-mode: 45 mph (72 kph) A-mode: 36 mph (58 kph) P-mode: 31 mph (50 kph)
Max Tilt Angle	S-mode: 42° A-mode: 35° P-mode: 25
Max Wind Speed Resistance	10 m/s
Max Flight Time	Approx. 30 minutes

Gimbal	
Stabilization	3-axis (pitch, roll, yaw)
Controllable Range	Pitch: -90° to +30°

The data collection process was divided into three phases:

1. Find and define which areas would be better to record videos.
2. Record the necessary amount of data to start experiments.
3. Annotate the data.

The first phase was done by analyzing a few known zones in *Serra da Estrela*, either by Google Maps or by going there in person and considering whether it was a good place. Given that we needed to record people and cars, it had to be a zone that included almost all terrains. With this in mind, the chosen zone to record was Rosa Negra (this link will redirect to the mentioned zone's location); figure 3.3 delimits the main zone of where the recordings took place and also, the sub-zones that were chosen for more specific terrains.

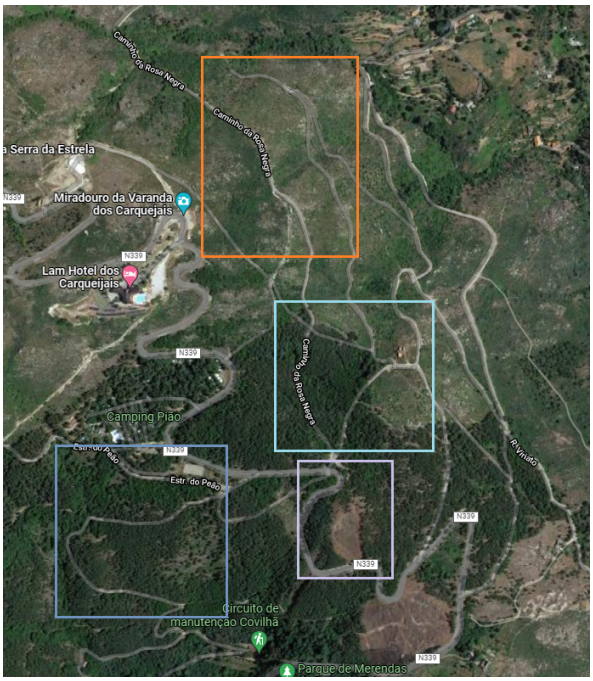

Figure 3.3: Overview of the different recording zones.

Afterwards, the videos were recorded on particular days to obtain as much data diversity as possible (*i.e.,* morning, afternoon, and night scenarios). Additionally, as shown in the previous figure, multiple zones were chosen based on their different terrains: a few with more tree density and dirt roads and others with more tar roads, less tree density, and more gravel. The last phase of this process, data annotation, was performed in CVAT. This tool makes annotating and tracking objects very simple; furthermore, annotations can be exported in many different formats, making it helpful in trying the dataset on different models.

3.4 Dataset Specification and Statistics

In order to better analyze the dataset, this section describes a few of its statistical properties. The first properties being analyzed in the dataset are the video properties: the resolution chosen for the recording was 1920x1080, and the number of frames per second was 30. The drone's more detailed camera settings, supported video and photo encoding formats are shown in table 3.3. The dataset consists of video clips with a total of 86150 frames. There are 20006 frames with persons in the current dataset status and 6673 frames with cars.

Table 3.3: Drone camera properties.

Camera	
Sensor	1” CMOS Effective pixels: 20M
Lens	FOV 84° 8.8 mm/24 mm (35 mm format equivalent) f/2.8 - f/11 auto focus at 1 m - ∞
Mechanical Shutter Speed	8 - 1/2000 s
Electronic Shutter Speed	8 - 1/8000 s
Photo	PEG, DNG (RAW), JPEG + DNG
Video	MP4/MOV (AVC/H.264; HEVC/H.265)

In order to better showcase the dataset’s properties, a few histograms were plotted to describe different aspects of it. The first is a simple histogram that shows the different brightness levels of all the frames in the dataset, shown in figure 3.4.

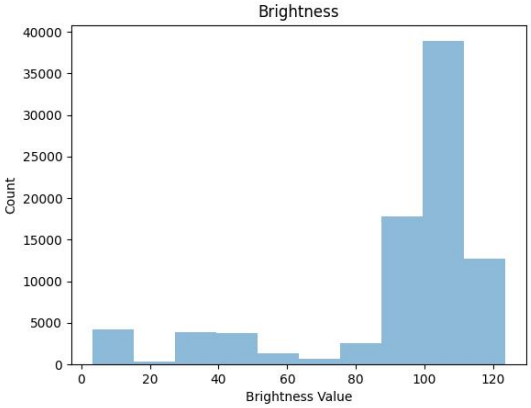

Figure 3.4: Dataset’s brightness distribution.

By analyzing the previous histogram, it is noticeable that most of the frames in the dataset were taken during the day, 3.5, with more brightness with only a few examples, recorded more at night, 3.6.

Figure 3.5: Day recording example.

Figure 3.6: Night recording example.

Since terrains were a very important step in the experiments so far and crucial to making a few conclusions, the following histograms, 3.7, 3.8, 3.9 and 3.10 show the relative frequency

of each terrain relative to the whole frame.

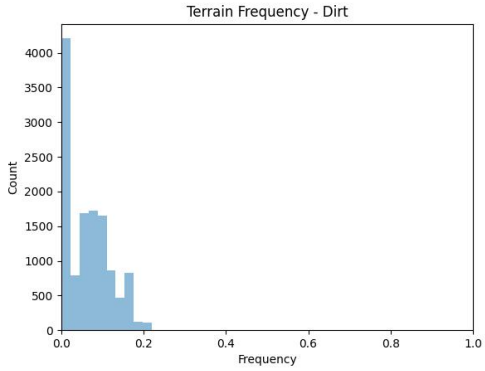

Figure 3.7: Relative frequency of dirt terrain in each frame.

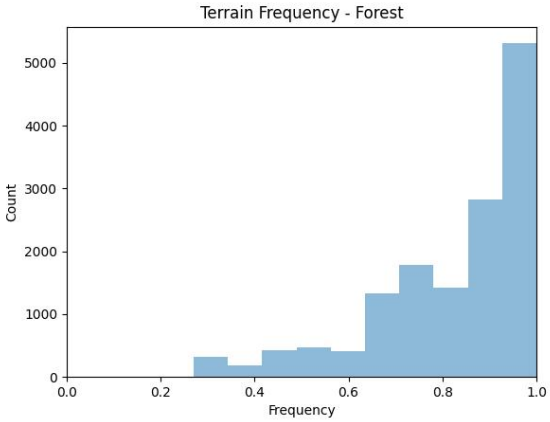

Figure 3.8: Relative frequency of forest terrain in each frame.

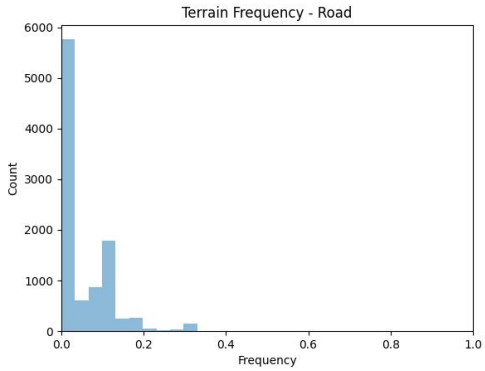

Figure 3.9: Relative frequency of road terrain in each frame

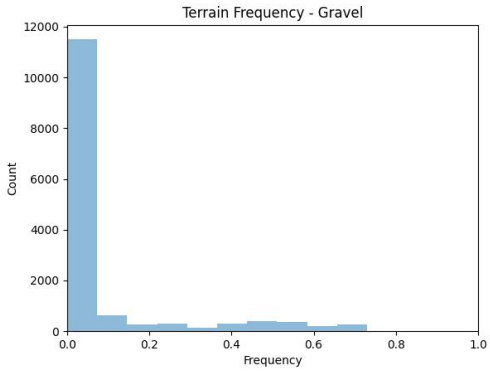

Figure 3.10: Relative frequency of gravel terrain in each frame

The specific scenario in which this dataset was recorded is noticeable since most of the frames are covered mainly through forest, and then there are only a few zones with the other terrains.

The following histograms, 3.11 and 3.12, illustrate the distribution of bounding box areas for each class, car and person. This is important for models where there is the need to setup pre-defined anchor boxes that will be used to detect the objects; since they can be adapted to every specific scenario, this data provides a solid foundation to change the anchor boxes of those methods to ratios and scales that better fit this problem.

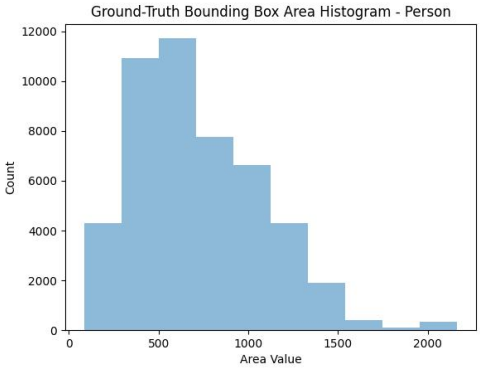

Figure 3.11: Histogram for class person bounding boxes.

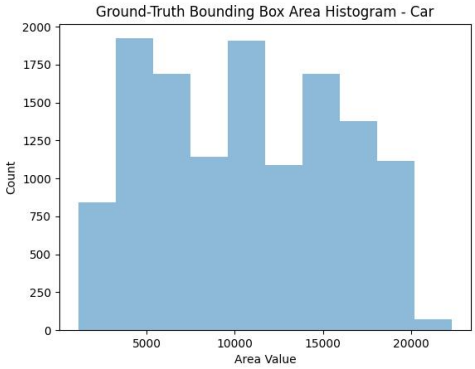

Figure 3.12: Histogram for class car bounding boxes.

More than anything, the previous histograms showcase how small the objects we are trying to detect are. Even though in cars the bounding boxes are more significant, the bounding boxes for almost every person are very small and thus, leading to a few problems that will be discussed in the experiments section.

3.5 Dataset's Second Iteration

The previous section describes the first iteration of the dataset gathered for this work. The main objective until then was to gather a solid amount of data to perform experiments on supervised methods and analyze their performance. However, once the project advanced into the self-supervised phase, there arose a need for a more significant amount of data.

The main difference from this second iteration was that the data itself would only need to be gathered and not annotated since the self-supervised experiment's objective was to analyze how we could use this unlabelled data to improve the model's performance.

With this being said, this part of the dataset was gathered by considering the same four zones as pictured in figure 3.13.

Figure 3.13: Unlabelled examples of the new dataset's iteration.

Furthermore, since the supervised results pictured a clear tendency that the models tend to perform worse on the dirt terrain, the data gathered in this phase focused explicitly on this terrain. The data collected for this second iteration consists of around 71 thousand frames from the four zones. Table 3.4 shows the number of frames for both labelled and unlabelled sets.

Table 3.4: Number of frames for the labelled and unlabelled set.

Type	# Frames
Labelled Set	21 225
Unlabelled Set	71 352

Finally, we used the data gathered from this new recording session to perform more experiments, both on supervised and self-supervised architectures. Close to the final phase of this work, and although we had a significant amount of data to perform various experiments, we noticed that obtaining better results and further analysing the self-supervised results would require many more recording sessions.

Chapter 4

Experiments and Discussion

The present chapter discusses the different experiments performed in this work. The first section 4.1 presents the most relevant object detection metrics and how they are calculated since they are the base for every experiment performed. Furthermore, section 4.2 aims to explain why there is such difficulty applying state-of-the-art methods to aerial data, providing an extensive discussion of the experimental results for each model and every terrain. Moreover, section 4.3 illustrates the different self-supervised experiments using different ratios of labelled and unlabelled data. Finally, section 4.4 provides the main conclusions from the experiments performed in this chapter.

4.1 Object Detection Evaluation Metrics

In order to evaluate the bounding boxes that the system is outputting, we use Intersection-Over-Union (IoU) which computes the intersection area between the ground-truth bounding box and the system's outputted bounding box. Based on the result of this measurement, it is possible to analyze how well the system's bounding boxes are when compared to the ground-truth ones; this is done by calculating the number of true positives (TP), false positives (FP), and true negatives (TN). Note that a prediction is considered a true positive when the IoU is equal to or higher than a predefined threshold (usually 0.5); otherwise, if the IoU is below that threshold, we consider that prediction a false positive.
The precision metric measures how accurate the system's predictions are. We consider the number of true positives (TP) related to all the system's predictions (TP + FP) when calculating precision, as shown in equation 4.1.

$$Precision = \frac{TP}{TP + FP} \quad (4.1)$$

Another critical metric to consider is the model's recall, also known as sensitivity. This metric shows how many positive predictions the model predicts (TP) related to the total number of cases it should have predicted (TP + FN), equation 4.2.

$$Sensitivity = \frac{TP}{TP + FN} \quad (4.2)$$

The main objective of every model is to have a high precision while maintaining a high recall. It usually happens that there are scenarios where one might be more beneficial than the other. For example, in the medical field, a model must have a high recall which means that the model will predict 100% of the positive cases, even if that means having a lower precision

value. Because even if the model outputs more false positives having a higher recall, it is much more important to diagnose a patient with a false positive than a false negative.

Even though specificity is not considered when dealing with object detection tasks, given that in most scenarios, it is not feasible to analyze the model's true negatives. However, after analyzing the model's output and tweaking the network's last layer, we also obtained negative samples from the model. This metric provides an overview of how the model will likely not output a positive prediction when it should not. This is done by measuring the rate of true negatives (TN) concerning the total number of true negatives (TN) and false positives (FP), shown in 4.3.

$$Specificity = \frac{TN}{TN + FP} \quad (4.3)$$

Average precision is the most used metric to measure the performance of an object detector. This metric describes recall and precision in a single value calculated by averaging precision across all the recall values. Primarily, we divide the recall into 11 points from 0 to 1.0 and then compute the average precision for those values independently. Lastly, we divide the sum of all those average precision values at each recall point by the number of points; equation, 4.4, illustrates this precisely, where AP_r is the average precision for each recall point. Note that this was the official way of calculating AP for challenges like the PascalVOC competition. However, after 2010, the average precision started being calculated using every recall point, not only 11.

$$AP = \frac{1}{11} \times \sum_{r \in 0,\ldots,0.1} AP_r \quad (4.4)$$

Afterwards, we have mean average precision, which consists of averaging the AP over all the problem's classes, as shown in equation 4.5, where N corresponds to the number of classes and AP_i is the average precision for each particular class.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad (4.5)$$

As briefly explained, there is a trade-off between precision and recall. Even though the best possible scenario is that the model has high precision and a high recall, this is almost impossible due to the confidence threshold. Most models' predictions come with a value corresponding to the model's confidence that an object exists. By altering this threshold, we are actively impacting recall and precision values. This means that if the confidence threshold is high, the model will shorten its positive predictions, subsequently increasing the number of false negatives (lowering the recall metric). On the other hand, it will decrease the number of false positives (increasing the precision metric). Contrarily, if we decrease the threshold, we lower the precision but increase the recall.

With this being said, it is essential to analyze and observe this trade-off between precision and recall for each model to analyze the best confidence threshold for our specific task. This is done by plotting the precision-recall curve consisting of different points, each corresponding to one particular confidence threshold. The idea is that by analyzing this curve, we can find the point closest to the point (1,1), this being the perfect scenario, and consider the corresponding confidence threshold as the best one possible for this model.

4.2 Supervised Learning Experiments

4.2.1 Deep Feature Flow Evaluatin

This model and the previous one were trained with a learning rate of 0.00025, which was the default that the authors used with the *ImageNetVID*. Since this architecture uses feature propagation, the key-frame interval chosen was 10, the same as in the original proposal, meaning that the feature maps of the key-frame are propagated to the next 10 and then aggregated. The following table, 4.1, includes all the parameters and values used in this experiment.

Table 4.1: Parameters used in the Deep Feature Flow model's training.

Parameter	**Value**
Optimizer	Adam
Learning Rate	0.00025
Batch Size	2
Epochs	50
Key-frame Interval	10

The first step in object detection experiments is plotting the precision-recall curve. This will show us which confidence threshold provides the best trade-off between recall and precision for this problem. Note that, as previously said, after plotting the PR curve, the way to find the best threshold is to calculate how far each point is from the perfect scenario (*i.e.,* point (1,1), which represents perfect recall and perfect precision, as shown in figure 4.1.

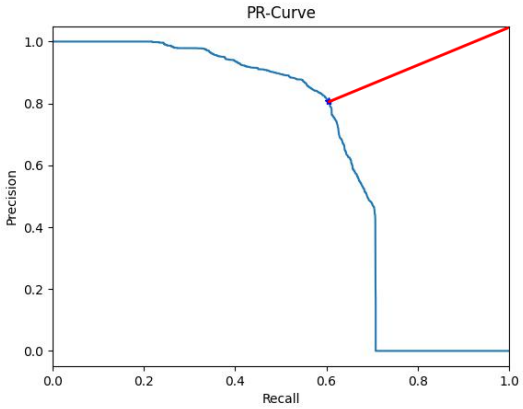

Figure 4.1: Precision-Recall curve for the Deep Feature Flow model.

The optimal confidence threshold for the DFF model was 0.39. Note that the model's precision decreases rapidly when we lower the threshold. This happens typically, given that the higher the confidence threshold, the fewer predictions are considered and thus, making it more precise. On the other hand, if we lower the threshold, the model will consider predictions with less confidence and increase the number of false positives. The average precision for this model was also calculated considering this; refer to figure 4.2 which illustrates the results obtained for each class.

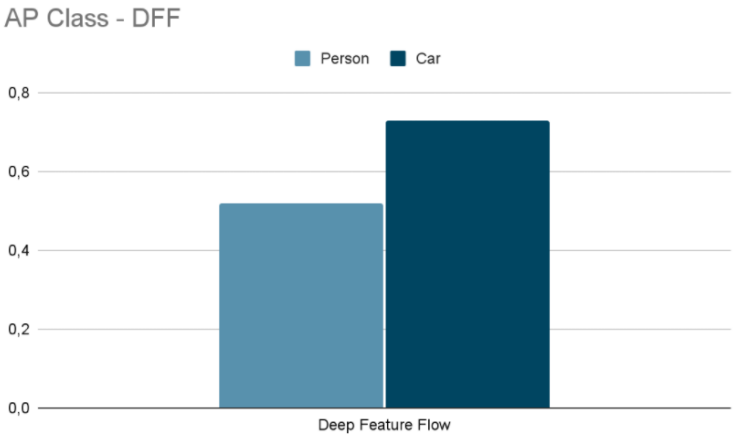

Figure 4.2: DFF model's average precision for each class.

Considering everything said up to this point, the results shown above make sense from the start. The standard supposition is that it is expected that the model is miss detecting people because they are small. Furthermore, it is easier to detect cars since they are considerably larger than people when seen from above. However, since the beginning, it has been mentioned that we would know where the model was failing the most by performing segmentation to outline the different terrains and thus, instead of taking these results for granted and only explaining them through mere suppositions, it is crucial to understand the reason for the miss predictions and where they are happening. For this process, we calculated the number of true positives, false positives, true negatives, and false negatives according to their terrain. Afterwards, we can calculate other metrics such as specificity with these values, illustrating

which type of terrain the model is failing. The following figure, 4.3, illustrates both sensitivity and specificity of the model for both classes but categorized by terrain.

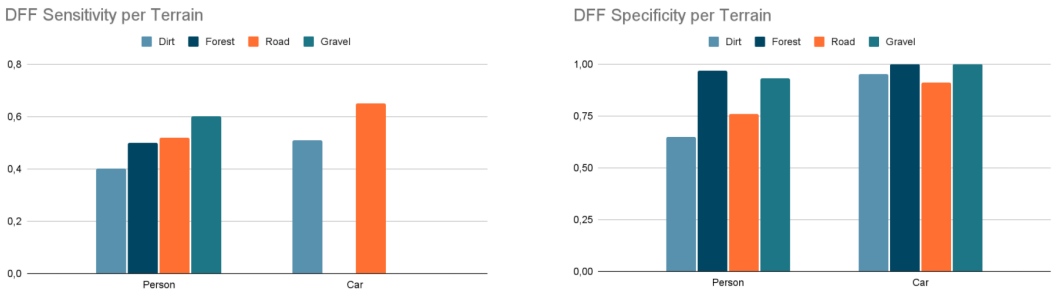

Figure 4.3: Sensitivity and specificity for the DFF model for each class per terrain.

The previous plots provide much valuable information but first, note that the class person has values for all terrains given that people could appear in every terrain. However, cars only appeared on the tar and dirt roads, meaning that only for those terrains the model would detect, and indeed, it is what happens. By analyzing the left plot, which showcases the sensitivity per terrain, it is noticeable that even though the model has a good precision in detecting cars, there are still cars that it does not detect, which happens mostly on dirt roads. This happens because there are only a handful of recordings where cars are actually in the dirt; thus, the model is not robust enough in this terrain. On the other hand, the model finds most cars on tar roads which is a sign. When it comes to the person's class, even though the model is not that good at finding all individuals in every terrain, this performance is worse in the dirt. This enables us to conclude that the model lacks object detail to find the best features to detect them successfully.

This is even more noticeable in the specificity plot, which is shown on the right and illustrates the model's performance in identifying the negative cases when they are negative (*i.e.,* detecting that there is not a person when in fact, there is not). By analyzing this plot, we can conclude that the model is correctly not detecting cars, with only a drop of specificity for both terrains (dirt and road). Furthermore, when analyzing the person's class side of the plot, it is clear what is going wrong with the model. It predicts people where they are not, and the terrain where this happens the most is dirt and gravel. The key takeaway from this plot is that the model is outputting false predictions for people in the dirt and gravel for some reason. By knowing that these models learn to find the best features to represent an object, it is clear that there must be some scenarios where the model considers a few places to have the same characteristics as a person. In order to completely understand why this happens, it is crucial to analyze an example of when this happens. Consider the following figure, 4.4, which is a frame of a clip given to the model as part of the test set.

Figure 4.4: Frame example of two people walking on a dirt road.

The figure above is one of the many examples where it is very noticeable that we humans can look at it and perceive that two people are walking by. However, a model must analyze that specific region and extract valuable features, which can be somewhat hard in these specific scenarios. Take into consideration the following image, which represents the model's output for two people walking, figure 4.5.

Figure 4.5: Model's correct predictions for two people.

Even though there are almost no details, the model still finds the best features to detect people, and it can do it somewhat accurately, showcasing how well these models can find those features. However, this leads to a particular problem, and it is the factor that most of the time decreases these models' accuracy overall for this problem's type. Previously we noticed that the model was outputting wrong predictions in the dirt terrain, and by analyzing the following figure, 4.6, it should be clear why.

Figure 4.6: Model's correct predictions for two people but falsely detecting a person in the dirt.

The previous figure is a frame of the same clip as the previous one, but the model predicted another person, only a simple hole on the ground. The fascinating factor here is that, as said from the beginning, expected from the beginning, the hole on the dirt road is very similar to a person when seen from such a height, making it understandable that the model confuses it as a person.

Figure 4.7: Region that the model confuses with a person.

The previous figure shows a close-up of the previously mentioned scenario to showcase further how detail is essential in these tasks. We can see how the model outlines this hole almost perfectly with a high confidence level on the right side. This corroborates the first parts of this document which stated that it was not that easy to achieve the same results on aerial images with state-of-the-art object detectors, which were not correctly designed for scenarios like this.

4.2.2 Flow Guided Feature Aggregation Evaluation

The experiments for this model were performed using the same parameters as the previous one, making it possible to analyze how they perform under the same conditions. Furthermore, the same parameters that were used to train the previous model, shown in table 4.1, were also the most appropriate for this model. In terms of performance, this model appears to be better in terms of average precision, especially in the class person, when compared to the previous video object detector. The following figure, 4.8 shows the precision-recall curve for this model.

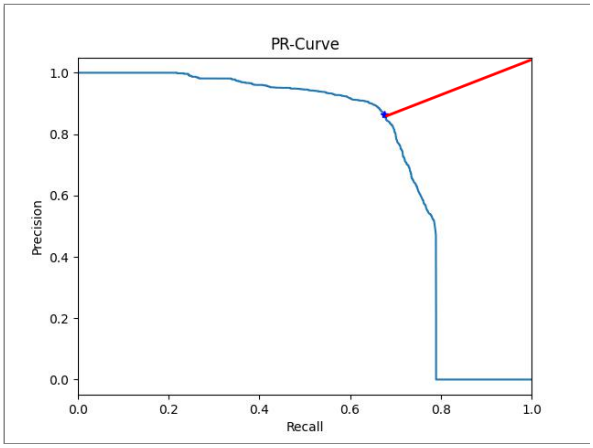

Figure 4.8: Precision-Recall curve for the FGFA model.

The model has better precision on the class person and performs the same as the previous one when detecting cars, shown in figure 4.9. This improvement in the performance makes sense also considering the models' architectures. Since the DFF model propagates feature maps' information to the following, the FGFA model will consider the features of both the previous and following frames.

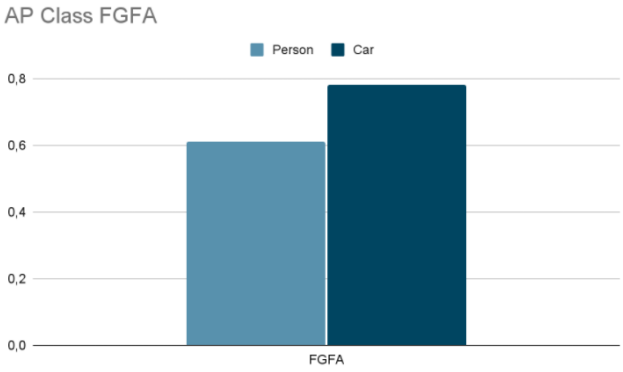

Figure 4.9: FGFA model's average precision for each class.

Even though there was an improvement in precision, this model shares the same problem that the previous one did, which leads to miss detections on certain terrains. The following figure, 4.10, shows this model's sensitivity and specificity graphs.

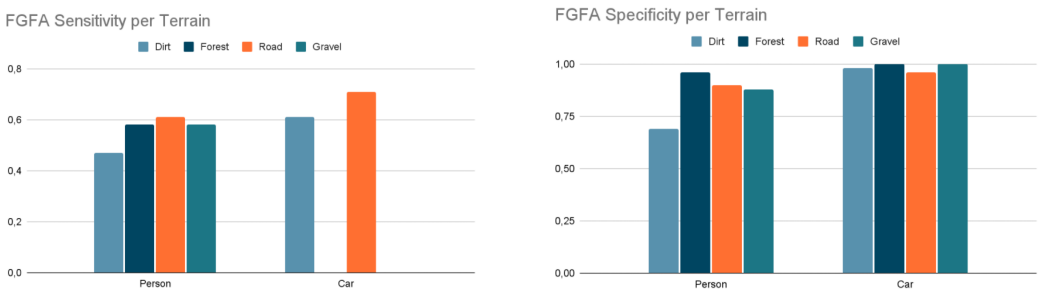

Figure 4.10: Sensitivity and specificity for the FGFA model for each class per terrain.

With these results, we can notice that the model has better sensitivity in the dirt and forest

terrains, which is an improvement, meaning that the model can correctly detect more people in those terrains. However, if we look at the specificity plot, this model also suffers from the same problem as the previous one. It is detecting people where there are none in the same terrains. Since that, in the previous model, we analyzed an example of the model miss detecting a person on a dirt road; in this model, we will analyze another terrain to perceive why this is happening. The main cause for these miss detections is that the model does not find the best features given the object's detail when seen above at such heights. This makes it logical that the model confuses people with holes in the dirt terrain, and in the gravel terrain, the fact that there are many rocks makes it hard for the model to distinguish. In the forest terrain, there are also false positives since, in these scenarios, shadows are another important factor leading to the model's confusion. Consider the following figure 4.11.

Figure 4.11: Frame example of the test set.

In the previous figure, shadows affect many zones due to the sun's position at that time. Considering this and the previously mentioned statements about analyzing features, the following figure, 4.12 represents the model's output to this frame.

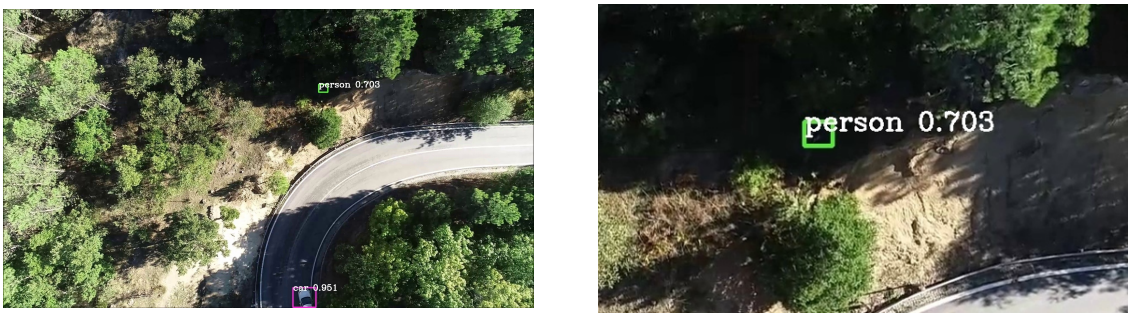

Figure 4.12: Model's output for the previous frame containing a miss detection and a correct detection.

Even though the model made the correct prediction for the car, it also predicted a person in a zone affected by a shadow with very high confidence. Once again, this is what affects the model's accuracy and precision overall. These techniques might provide an excellent way to detect objects from aerial images with some post-processing methods.

4.2.3 Static Camera-based Background Subtraction

Following the experiments described in section 4.2, and based on the problems observed there, this section aims to solve them by using the additional information obtained by the different background subtraction algorithms. The main idea is to use the result of the background modellers to determine if the detector's prediction is valid.

Primarily, it is crucial to analyze how the different types of algorithms perform in the different scenarios of this work (*i.e.,* roads, dirt, and forest). In the first example, we use the the static-camera background subtraction algorithm mentioned in section 2.6.1, *Gaussian Mixture-based Algorithm*. Figure 4.13 illustrates one example of the dataset's frames and the same frame after being passed through the background modeller.

Figure 4.13: Static-camera background subtraction algorithm output for an example frame containing a person in dirt and a vehicle on a tar road.

It can be noticed that the model fails to model the person and the car. Additionally, this occurs in most of the scenarios on our dataset, providing a clear illustration that the algorithms which work well for static camera scenarios fail to model the background in scenarios where the camera is moving. Another example is shown in figure 4.14, which illustrates how the algorithm performs in a different scenario.

Figure 4.14: Static-camera background subtraction algorithm output for an example frame containing two people on a dirt road.

In this case, it can also be noticed that the algorithm also does not model people on dirt roads either. Moreover, given that most of the dataset consists of scenarios like these, it is clear that this algorithm is not the most suitable to use as support to the object detector.

4.2.4 Dynamic Camera-based Background Subtraction

The next phase of this experiment was to evaluate how the *JA-POLS* algorithm, mentioned in 2.6.2, which was designed especially for moving cameras, would perform in the same scenarios. Figure 4.15 and figure 4.16 illustrate the same scenarios only when applied with this algorithm.

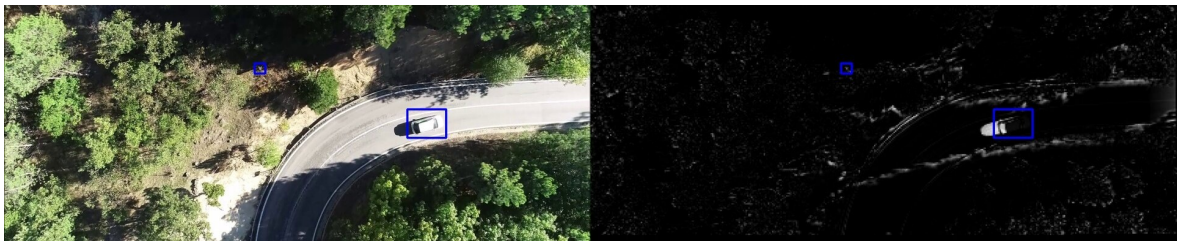

Figure 4.15: Moving-camera background subtraction algorithm output for an example frame containing a person in dirt and a vehicle on a tar road.

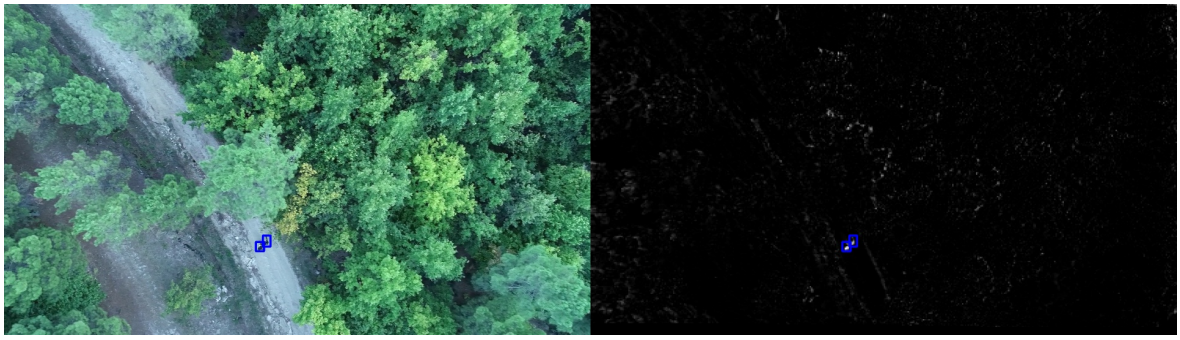

Figure 4.16: Moving-camera background subtraction algorithm output for an example frame containing two people on a dirt road.

The previous figure clearly shows that this algorithm can successfully model the person on the frame and partially the car on the top image. Additionally, the same occurs for the second example, where the algorithm can also successfully model the two people on the dirt road, which is a significant improvement from the static-camera algorithm. Another terrain where this is verified is the gravel terrain, as shown in figure 4.17.

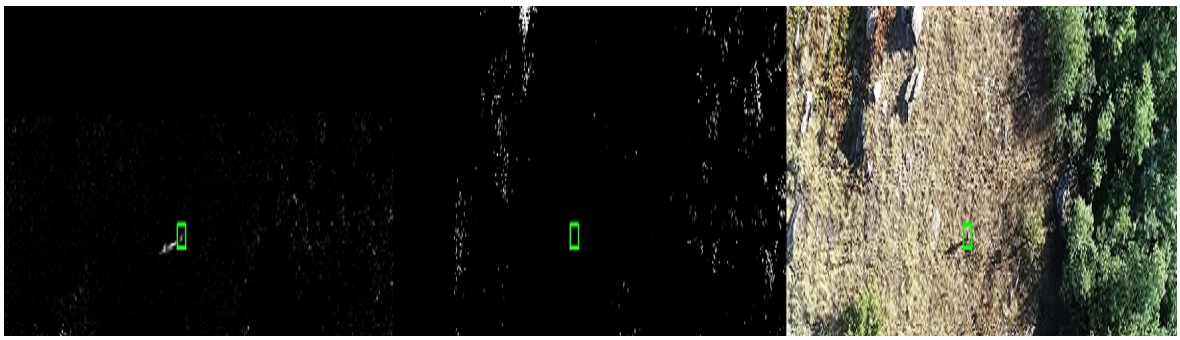

Figure 4.17: Illustration of the original example frame, shown on the right side, with the static-camera algorithm's output, shown on the center image, and the dynamic-camera algorithm's output shown in the left side.

In most of the scenarios of this project's dataset, as shown by the previous figures, the dynamic-moving camera algorithm can model the people and the vehicles particularly well. This sets up a good baseline for the hypothesis that by using this algorithm, we can, as a post-process method, define valid regions in which we can consider the object detector's predictions valid.

4.2.5 Evaluation with Background and Static-based Background Subtraction

As previously explained, the main idea for using the background modeller is to determine possible regions of interest (*i.e.,* zones where objects appear to be moving) and limit the detections of the model to those zones only. This would prevent scenarios such as those shown in figure 4.6, where the model detects a person when it is a hole in the ground. If we consider, for example, the output of the background modellers for that specific frame, we obtain the following results, figure 4.18.

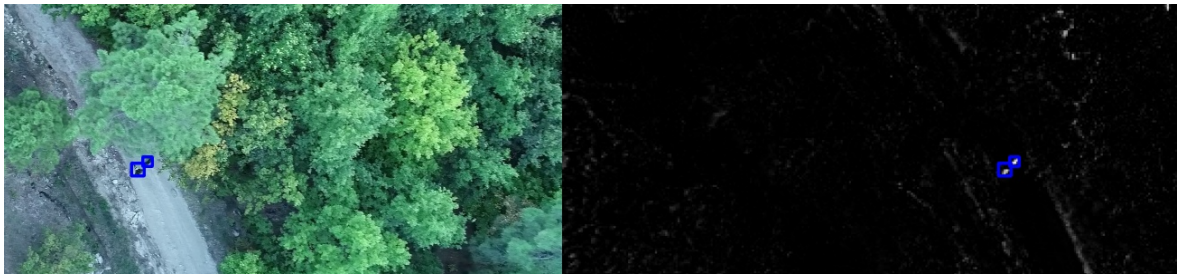

Figure 4.18: Illustration of the original frame where the video object detector model miss detected a hole with a person, side-by-side with the output of the dynamic-camera algorithm background modeller's output for that same frame.

It can be noticed that, in the background algorithm's output, the two persons are substantially highlighted. At the same time, there is no obvious highlight in the ground hole zone which was previously classified as a person. With this being said, in the following experiments, we will only consider the output detections of the model valid if they match a substantial part of the background model's output.

By delineating this boundary between valid and invalid detections, it is possible only to evaluate the detections considered valid and obtain the same metrics for these new detections to compare the model's performance without background removal, with the static-camera algorithm, and with the moving-camera algorithm. The first set of results is shown in the graph represented by figure 4.19.

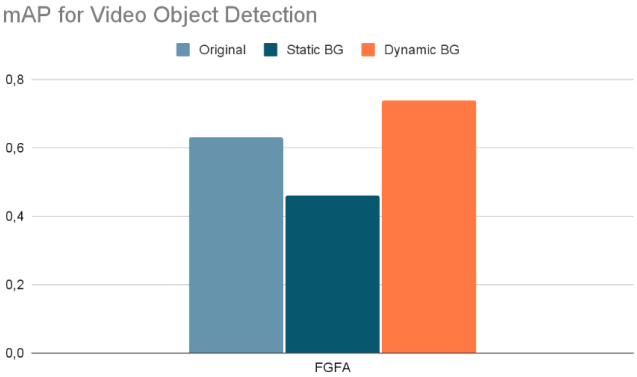

Figure 4.19: Mean Average-Precision for the FGFA model using none of the background-modellers, the static-camera based algorithm and at last, the moving-camera background modeller as valid and non valid region identifiers.

By analyzing the previous graph, we can observe that we obtain even worse results by considering the static-camera algorithm and using it to separate the valid and invalid detections than without any algorithm. This happens because, as shown in figure 4.17, since this algorithm cannot successfully model neither the persons nor vehicles, there will be detections that contain an object within it, but since the background modeller does not model any of the objects, they are considered. However, we obtain much better results when using the moving-camera background modeller to perform the separation. In order to better perceive in which specific scenarios the performance is increasing, a new set of terrain metrics were obtained, shown in figure 4.20.

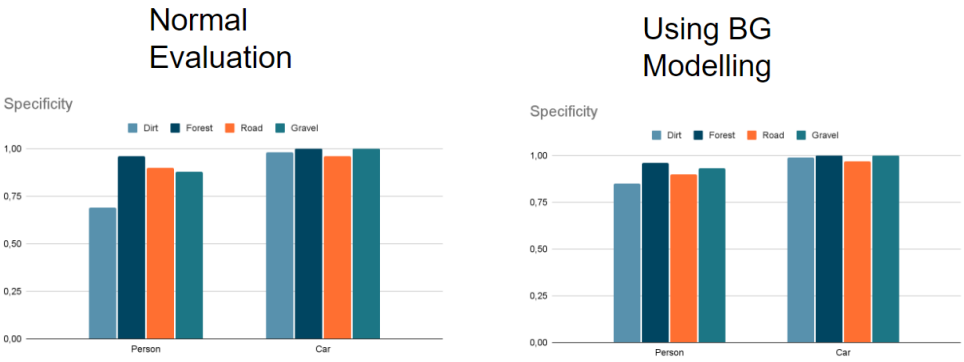

Figure 4.20: Region that the model confuses with a person.

The main problem with the video object detectors analyzed in this work was that they would often mistake variables that appear in this specific scenario (*e.g.*, ground holes and shadows) with people and thus, affecting its evaluation performance and leading to false positives. These holes and shadows would often appear on dirt roads, the same terrain where people appear more in the dataset. With this being said, there is a significant improvement in the previous graph, especially in the dirt terrain. Based on examples such as the one illustrated in 4.18, it is clear that by using only the detections in a valid region, we remove most of the false positives that often occurred while using these video object detector models.

4.3 Self-Supervised Learning Experiments

As mentioned previously in this document, another main topic in this work is the analysis of state-of-the-art self-supervised methods when applied to this dataset. The main idea is to adapt the architecture shown in 4.3 to our problem. Figure 4.21 depicts the proposed architecture.

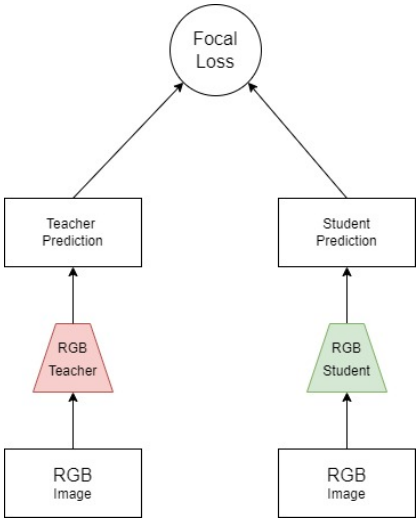

Figure 4.21: Self-supervised architecture proposed for this work.

There is no need for depth, audio, or thermal modalities in this situation; thus, the only modality used in the proposed architecture is the RGB one. The idea is that the teacher model provides the detections, which are then considered to be the ground truth and are further used to calculate the focal loss, according to 2.1, with the student's detections.
This way, we can have a part of the dataset that is labelled and is used to train the teacher, and then we can use the non-labelled part of the dataset to train the student, as shown in figure 4.22.

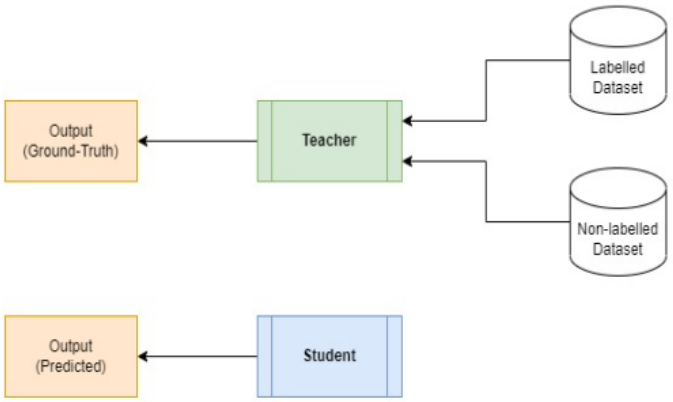

Figure 4.22: Illustration of the two different data sources for the self-supervised architecture.

In the development of this work, the currently available data, both in the labelled and unlabelled sets, are the following:

- **Labelled Dataset** – 21.225 frames;

- **Unlabelled Dataset** – 71.352 frames.

With this being said, the experiments will consist of training the teacher model with the labelled dataset and then using this trained model to be the provider of ground-truth detections for training the student. In order to analyze how the student model performs, the experiments were performed by increasing the amount of data used in the student's training, **U**, based on the original size of the labelled dataset.

4.3.1 *EfficientDet* Evaluation

In order to perform experiments with the self-supervised architecture proposed, we must first train the teacher model with the labelled dataset. Table 4.2 shows the parameters used to train this network.

Table 4.2: Parameters used to train the *EfficientDet* model.

Parameter	Value
Optimizer	Adam
Learning Rate	1e-6
Batch Size	6
Epochs	25
Anchor Scales	[0.5, 2, 4]
Anchor Ratios	[(0.9, 1.1), (1.0, 1.0), (1.1, 0.9)]

After training the network with the previous parameters, we obtain results such as the model's mean average precision, which we can compare with the same metric obtained for the previously mentioned video object detectors, illustrated in figure 4.23.

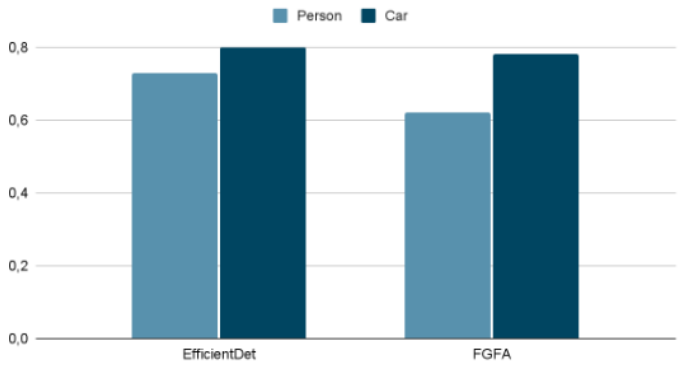

Figure 4.23: Average precision evaluation for each class by the *EfficientDet* model, shown on the left side of the graph, and for the FGFA model, shown on the right side of the graph.

The previous graph shows a noticeable improvement, especially in the class person, and even though there are these improvements in the class person, there is still a downside to using this model over the FGFA one. The following figure, 4.24 illustrates why the video object detectors work particularly well in tracking objects.

Figure 4.24: Illustration of the same example frame's output from the different models. The figure on the left, shows the output from the *EfficientDet* model which shows no detections. The figure on the right illustrates the output from the *FGFA* model with at least one correct detection.

By analyzing the image, we notice that the single-frame-based detector does not detect any object in a particular situation when another object partially blocks the object. In contrast, the video object detector can successfully identify the object even when the tree partially blocks it. In order to obtain a more profound insight into how the model perceives these objects and what types of features it is extracting, we obtained the features maps of the different fusion levels, as illustrated in 2.8. For illustration purposes, we obtained these feature maps for the two types of classes involved in this scenario, the first, for the class car, shown in figure 4.25, and the latter, for the person class, shown in figure 4.26.

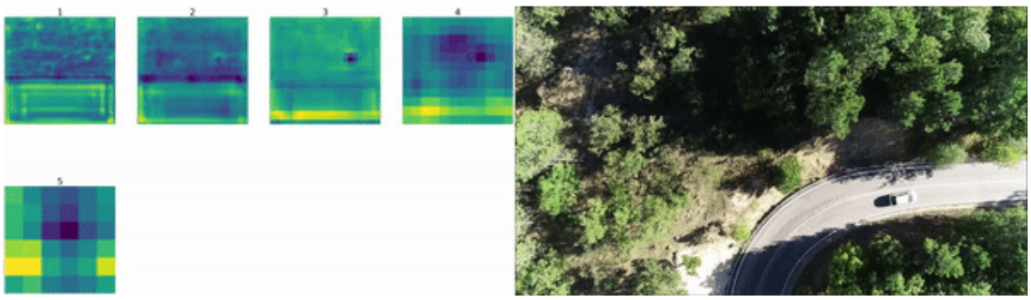

Figure 4.25: Illustration of the different levels of feature maps extracted for an example of a frame from the test set containing a vehicle.

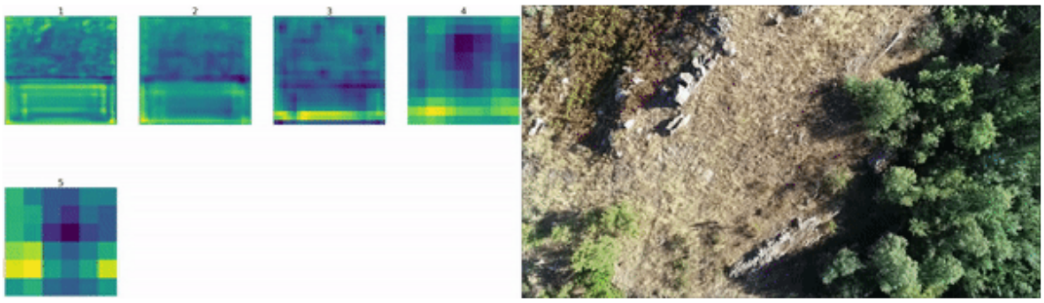

Figure 4.26: Illustration of the different levels of feature maps extracted for an example of a frame from the test set containing a person.

The first figure, 4.25, provides a visualization of the different levels of feature that the *EfficientDet* model extracts from the corresponding frame shown on the right side. On the one hand, it is very noticeable that the model can extract crucial features through its learned filters that successfully portray the car, thus showcasing why this model is remarkably accurate when detecting cars on dirt and tar roads. On the other hand, the second figure, 4.26, also clearly illustrates why these models' worst problem is precisely detecting people. Due to the size of people in these types of images, it is challenging in some scenarios for the model to extract meaningful features that will lead to the detection of the object. Additionally, it is essential to note that the *EffcientDet* model was designed to combine multiple levels of feature maps to obtain an even better representation of classes than other models. Even with these models, which are designed to perform multi-level feature fusion, having a hard time extracting meaningful features to characterize a person in an aerial image, it is also clear why models such as the video object detectors perform even worse on detecting people. Furthermore, given that this model performs better in average precision than the video object detectors, it is interesting to analyze in which terrains this improvement occurs. The following figure, 4.27 illustrates the graph for specificity corresponding to the *EfficientDet* model related to the different terrains.

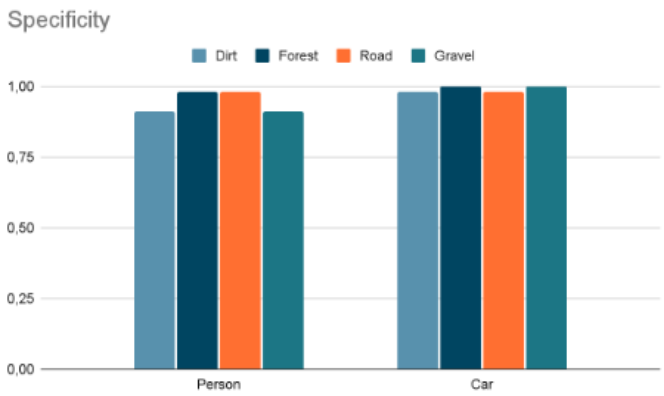

Figure 4.27: Graph representing the terrain specificity results for the *EfficientDet* model.

The main improvement when comparing the previous graph to the one shown in figure 4.10 occurs in the class person and the dirt terrain. In the evaluation of video object detectors, the main problem was the output of false positives due to the reasons specified in subsection 4.2.1 and subsection 4.2.2. These occurrences do not happen when evaluating the *EfficientDet* model, as shown in the following figure 4.28.

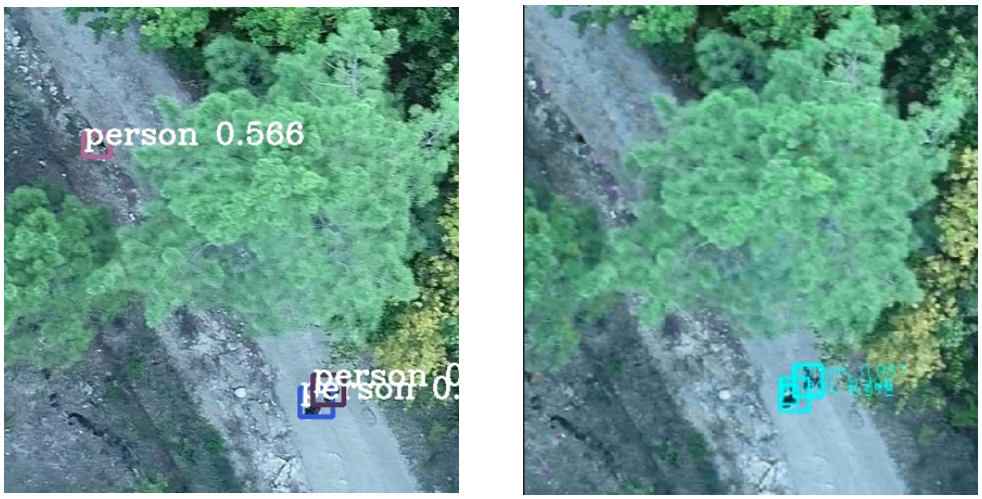

Figure 4.28: Illustration of the different models' outputs for the same frame. On the left, the output of the *FGFA* model, including a miss detection. And, on the right, the output from the *EfficientDet* model, with only valid predictions.

The scenario shown in the two previous figures illustrates the main reason for the *EfficientDet*'s performance being better than the *FGFA*'s. Furthermore, these false positives mainly occurred in the dirt due to its natural characteristics, often being confused as a person.

4.3.2 Self-Supervised Proposed Method Evaluation

Posterior to training the *EfficientDet* model, we can then use this model as the teacher in the self-supervised learning architecture and evaluate how it performs in different amounts of data used to train the teacher. In the first phase of these experiments, the ratio of data used in the teacher and the student's training is illustrated in the following figure, 4.29.

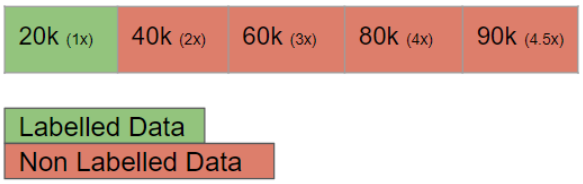

Figure 4.29: Distribution of labelled and unlabelled data related to second phase of self-supervised experiments, with a baseline of 20k labelled set.

As shown by the previous image, this experiment will be performed by training the teacher with 20k labelled frames and then using that teacher in the self-supervised architecture. The parameters used in the training of the student model are shown in the following table, 4.3.

Table 4.3: Parameters used to train the student model in the self-supervised architecture.

Parameter	Value
Optimizer	Adam
Learning Rate	1e-8
Batch Size	6
Epochs	50
Patience	10
Anchor Scales	[0.5, 2, 4]
Anchor Ratios	[(0.9, 1.1), (1.0, 1.0), (1.1, 0.9)]

This experiment's fundamental objective is to analyze how the student's model performs with the different unlabelled amounts of data and if, at any time, it can surpass the teacher's baseline performance. This would mean that the student model would get more robust, and its ability to generalize to a broader range of scenarios would improve. The experiments were performed by setting a fixed number of epochs, 50. However, in most training runs, the model stopped around the 30th epoch due to the patience setting, which was set to 10 epochs (*i.e.,* early stopping the model's training once it stops learning). The following graph, 4.30, provides the evolution of the student's performance and the amount of data used in its training.

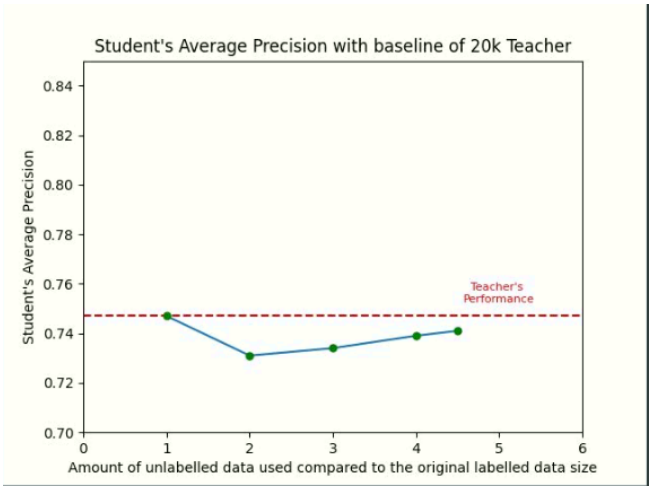

Figure 4.30: Graph illustrating the first experiment student's average precision evolution along with the number of unlabelled data used, with the teacher's baseline when trained with 20 thousand frames.

The first observation we can get after analyzing the previous graph is that, after the first iteration with the baseline data, the student's performance starts lower than the teacher's model. This is because we are introducing new scenarios to the dataset and thus, worsening the model's performance on the test set. However, the main idea was that at the beginning, the model would worsen, but while increasing the amount of unlabelled data, it would be able to adapt and be more robust. This happens after the second iteration, where the student's performance increases slowly towards the baseline performance. In this experiment, the last iteration used five times the baseline data (100k frames), and the tendency of the graph was

still in an upwards direction which led to the belief that the performance would reach the teacher's performance and, in theory, surpass it.
After this experiment and the graph showed an upwards trend, it led to the belief that if there were a more significant amount of data, the experiments would show a continuous performance increase. However, the amount of data gathered for this project was insufficient for more experiments. With this being said, the other manner of performing this experiment but with more ratios of labelled and unlabelled data would be to train the teacher model with fewer data. The following figure, 4.31, illustrates the data ratio used in the subsequent experiments with a baseline of 10k frames.

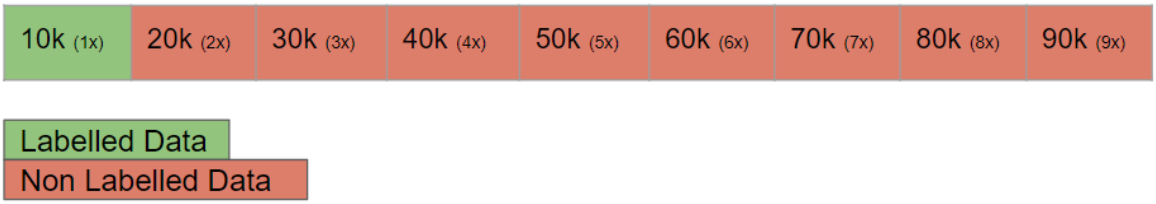

Figure 4.31: Distribution of labelled and unlabelled data related to second phase of self-supervised experiments, with a baseline of 10k labelled set.

By using the distribution previously shown, we could perform the same experiment but now up to 10 times the baseline amount of data that was used to train the teacher. As expected, the baseline average precision also lowers because we are using a smaller amount of data to train the teacher than in the previous experiments. With this being said, the following graph, 4.32, illustrates the results of the different experiments performed in this phase for the different amounts of data used in the student's training.

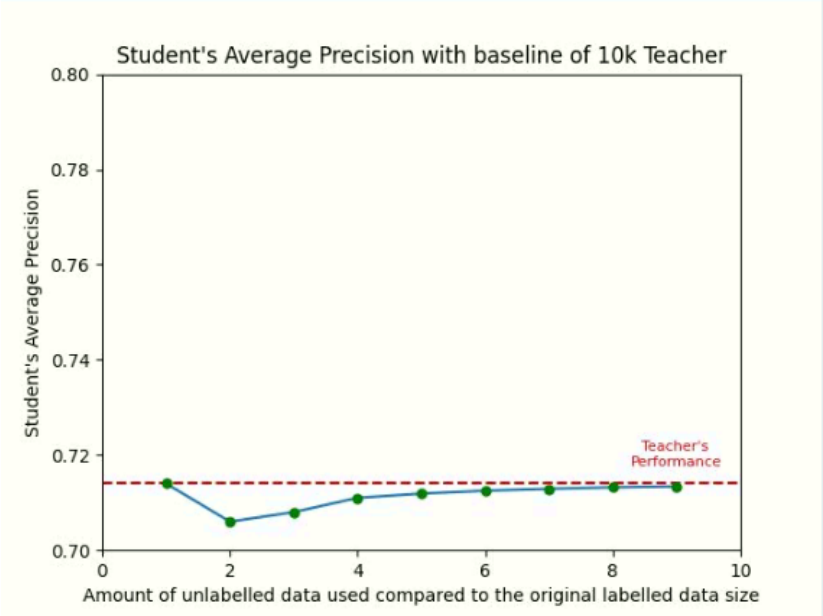

Figure 4.32: Graph illustrating the first experiment student's average precision evolution along with the number of unlabelled data used, with the teacher's baseline when trained with 10 thousand frames.

In this experiment, we can verify that the precision drops after the first iteration, as it was verified in graph 4.30. Additionally, perhaps the most interesting fact from this graph is that

the student's performance is close to the teacher's baseline performance. In contrast to previous experiments where the student's performance tended to increase, there was not enough data to analyze how it would develop; in this experiment, with more iterations, we can almost achieve the teacher's performance with a ratio of 10 times the labelled data. Considering this, achieving the teacher's performance was not the primary goal of this experiment, but instead, analyzing how the student's performance would evolve while providing various amounts of unlabelled data. This experiment clarifies that, by using an amount of unlabelled data 10 times higher than the original labelled dataset, we can achieve almost the same results as if we were only to train the model with that original labelled data. Even though this might not seem relevant for this scenario, not that in most real-life scenarios, models need to be trained with hundreds of thousands, if not millions, of frames and thus, making the possibility for having a ratio of unlabelled data up to 50 times or more. Note that these experiments were done with the dataset gathered explicitly for this work. Even though it has many good characteristics, it is in no way complete or has enough amount of data needed to perform more extensive experiments.

To analyze how the architecture would perform when given more sets of unlabelled data, we further decreased the data in which the teacher was trained to five thousand labelled frames. The ratio of data that we obtained is illustrated in the following figure, 4.33.

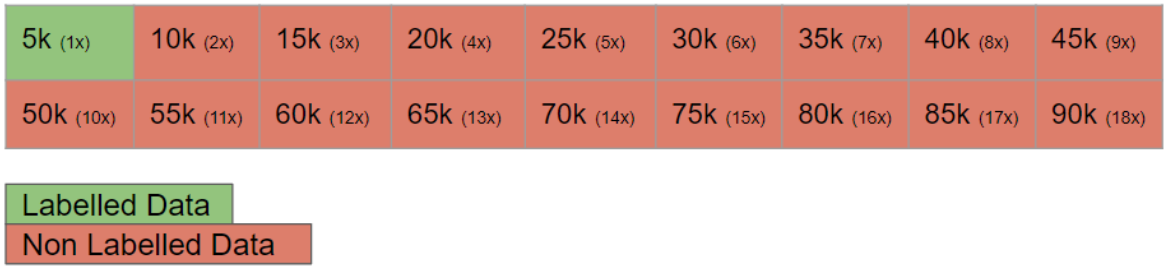

Figure 4.33: Distribution of labelled and unlabelled data related to second phase of self-supervised experiments, with a baseline of 5k labelled set.

By using the previous sets of labelled and unlabelled data, we could perform many more iterations compared to the previous experiments. The precision of the teacher model decreased, just like in the other experiments. However, it allows us to understand better if, at any moment, we can obtain a student model that can outperform the teacher's performance. With this being said, the following graph, figure 4.34, illustrates the evolution of the student's precision by increasing the amount of unlabelled data gradually.

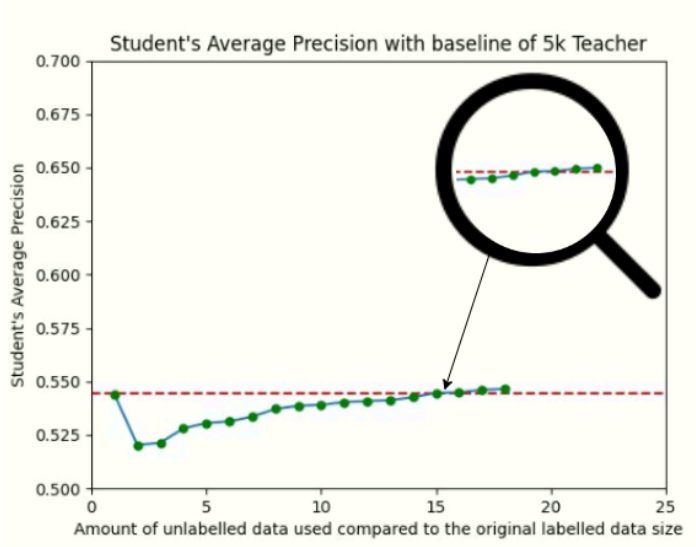

Figure 4.34: Graph illustrating the first experiment student's average precision evolution along with the number of unlabelled data used, with the teacher's baseline when trained with five thousand frames.

From analyzing the experiments illustrated by the previous graph, we notice that at a ratio of 16 times the original data, we indeed obtain a student model that outperforms the teacher model. Additionally, it can also be noticed that since we were able to perform many more iterations, it is noticeable that the student's precision is still increasing with the increase of unlabelled data provided. Although the precision value, in this case, is not very significant, in this experiment, we were able to illustrate that the student's model can outperform the teacher's model past a specific ratio of labelled and unlabelled data. This sets up the precedent for further experiments where we have much more available data; for example, in a situation where we have 400k available frames, we can start by labelling 20k frames and then use the rest in a self-supervised matter to improve the teacher.

Finally, after performing these experiments, to further analyze how the student's performance would evolve with more iterations of unlabelled data, there would be the need to reduce the set of labelled data used to train the teacher's model even more. However, as seen before, the teacher's precision drops by using fewer data to train it. This happened when we trained the teacher with 10 thousand and five thousand frames instead of the original 20 thousand. The main problem with performing these experiments with this limited amount of data is that, for us to use the teacher model in the self-supervised architecture, we need to ensure that the model can generalize well for the unlabelled data. If we use a model that does not perform well on new data, then there is no logic in training a student's model in a self-supervised manner to achieve even better results. In past experiments, even when training the teacher model with 10 thousand frames, its performance was good enough to generalize to the new unlabelled data; however, when we start training the teacher with fewer data, like 1k, then it starts to become clear that its generalization capability starts decaying, as shown in figure 4.35, making it non-fitting for the self-supervised architecture anymore.

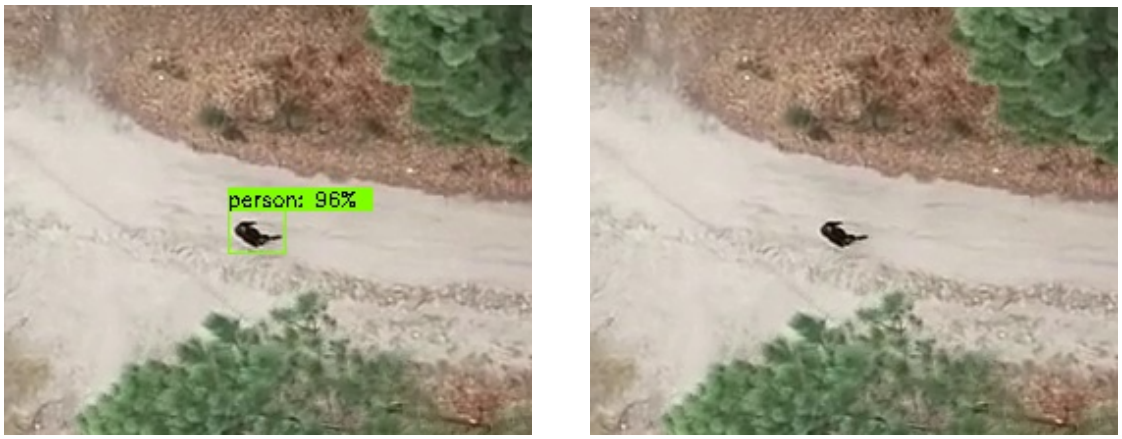

Figure 4.35: The left figure illustrates the output of the teacher's model, trained with 10 thousand frames for an unlabelled frame. The figure on the left illustrates the output of the teacher's model but then trained with a thousand frames for that same frame.

For this reason, the main objective of performing more extensive experiments would be to obtain more unlabelled data and simultaneously build a more robust teacher capable of performing better in unseen scenarios. Making it so that, when inserted into the self-supervised method, it would be able to provide valid and quality ground-truth predictions when training the student model.

4.4 Conclusions

The primary objective of the first experiments was to analyze how video object detectors performed on the custom dataset gathered for this work. Based on the results that were discussed in section 4.2, we can conclude that, for most scenarios, both video object detectors, DFF and FGFA, can detect people and vehicles. Even though the FGFA model was the best out of the video object detectors, both these models tend to perform worse when detecting people, especially on dirt terrain, as illustrated in figure 4.3 and figure 4.10. After analyzing specific scenarios in which these miss-detections occurred, it was noticed that particular environment variables were often confused with people (*i.e.,* holes in the dirt terrain and shadows). The analysis of these results enables us to conclude that applying models that work incredibly well on state-of-the-art datasets but that, when applied to aerial images, suffer from these problems that surge from the problem's nature.

Afterwards, the next step was analyzing how a single-frame-based object detector would perform in the same dataset. As shown in subsection 4.3.1, it became clear that even though this model obtains better performance in average precision, especially when detecting people, it also has its downsides. In this model, we obtain more accurate detections, meaning that the problem's nature causes fewer miss detections. This is because the *EfficientDet* performs multi-level feature fusion and, thus, can extract more meaningful features from frames where the object appears on a tiny scale, as it occurs in this work.
Subsequently, two methods were proposed to solve the previous methods' downsides after analyzing the results and showing the downsides of both the single-frame-based and video

object detectors in aerial images. The first method uses background removal algorithms to identify which detections are valid. In sub section 4.2.4, it is clear that the best algorithm to use is, logically, the one designed for moving-camera problems. We can increase both video object detectors' performance by using this method to define regions where we can consider the detection valid. The second method uses a self-supervised method to utilize unlabelled data to increase the supervised model's performance. In subsection 4.3.2 we explore how the self-supervised architecture behaves with different ratios of labelled and unlabelled data. The first observation was that the performance of the student model kept increasing the more iterations; however, the ratio of labelled and unlabelled data only allowed for the execution of five different iterations. Afterwards, we decreased the data used to train the teacher and then performed the same experiments. We observed that, even with more iterations, the student's performance improved further, but it was not enough to surpass the teacher. Finally, we performed a final experiment by only training the teacher with five thousand training frames and then using the rest of the data in the self-supervised architecture. In these final experiments, we could surpass the teacher's performance at a ratio of 16x the original data; this indicates that, if we have enough unlabelled data, we can develop a more robust model using this architecture.

Furthermore, we tried decreasing the training set for the teacher even more; however, the decrease in performance began affecting the performance of the self-supervised method; since the teacher is not able to generalize well for unseen data, there is never going to be an advantage on using this method given that is mainly based on the teacher's performance on unlabelled data. The main improvements in this experiment are to gather more data and: train the teacher with more labelled data to improve generalization and, in the self-supervised method, to experiment with more iterations of unlabelled data.

Finally, these experiments mainly illustrated that, due to the problem's nature, there are specific scenarios in which these models fail to detect any object. These miss detections occur mainly on people due to their tiny scale on the frame. We have shown that, on the one hand, video object detectors show a better capacity for tracking persons and vehicles but tend to confuse some environmental variables. On the other hand, single-frame-based methods such as *EfficientDet* do not confuse these environmental variables due to their better feature extraction architecture. However, they often fail to track the object throughout the object's course.

Chapter 5

Conclusions and Further Work

The main focus at the start of this work was to gather a specialized dataset for the problem in the matter, annotate it, and then use it to perform different experiments with different types of state-of-the-art object detectors. After performing such experiments, we analyzed the results, shown in chapter 4. We observed that when applying these models to a non-conventional aerial image dataset, the problems mentioned at the beginning of this document occur. While these models are considered exceptional at detecting objects in conventional datasets, their performance is not as good when facing aerial data, mainly due to the problem's natural characteristics mentioned in section 3.1. With the experiments performed in section 4.2 we could thoroughly analyze the models' outputs and determine precisely which terrains and classes the models had worse performance. The worst terrain was dirt, and the worst class was the person. These results precisely showcase the assumptions made in the beginning about what problems would arise from these models. The scale of the class person and its lack of detail, when seen from such height, makes it difficult for these models to find the optimal features for that object. One evidence of this occurrence is illustrated in figure 4.26, where we see that even the *EfficientDet* model, which combines different level feature maps to obtain a better representation of an object, is barely able to model the person in contrast to vehicles. We also show in figure 4.6 that environment variables such as ground holes can be easily mistaken for a person.

We conclude that both models have their downsides; however, in this work's scenario, it is better to have a model that sometimes outputs false positives but can track the object for its entire course, even if it is slightly obstructed. Note that, even though these miss detections are not very harmful in this context, we could not guarantee that this would not be a recurring factor in the system's deployment and would make it ineffective or exploitable. Thus, we proposed using background subtraction to filter the model's detections and determine the valid ones. As shown in section 4.20, we reduced the number of miss detections to the point where the performance of the video object detectors using the background removal as a post-processing method matches the single-frame-based method results in terms of average precision.

The last objective of this project was to use a self-supervised architecture to analyze how valuable unlabelled data could be in creating a more robust alternative to the self-supervised model. In most real-life scenarios, gathering data is often time-consuming; however, annotating all that data is even more time-consuming and requires intensive labour. The main idea would be not to need to annotate all the data but instead annotate a partition of the whole set and then use the labelled and unlabelled sets to train an even more accurate and robust model. We concluded that with the available data, the self-supervised method could

only achieve the same results as the supervised method, as described in subsection 4.3.2. Moreover, we show that the tendency of the student model increases the more unlabelled data used. We consider that the results support the possibility that, in scenarios where the amount of unlabelled data is much higher, we can obtain a self-supervised model that outperforms the supervised teacher model. In a more advanced stage of this work, it would be of high value to present a graph which illustrated the relation between the number of labelled data and the number of unlabelled data that would be needed to improve the supervised method, as shown by the example graph in figure 5.1.

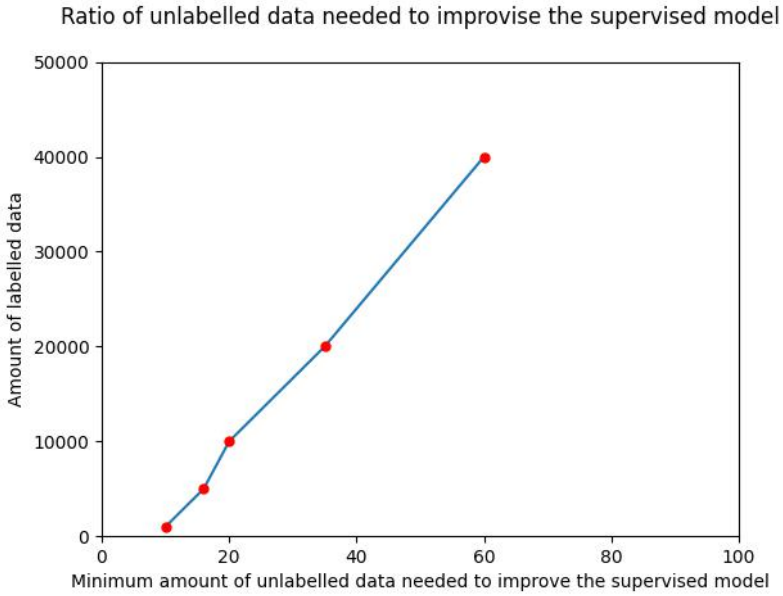

Figure 5.1: Example of the graph we could obtain by performing experiments with more data. In this graph, we would illustrate the minimum amount of unlabelled data that it would require to surpass the performance of the supervised model, based on the original labelled data size.

Suppose we performed all these experiments. We could then analyze how the self-supervised architecture would behave with the different amounts of original labelled data and the various amounts of unlabelled data and design a graph such as the previous one. By having a graph like this, we would be able to, at the start of any project, consider the size of our dataset, decide to only label it partially according to the graph, and then use a self-supervised architecture to improve the supervised model results. We could not perform such experiments due to the lack of available data and because performing such experiments without dedicated servers takes an immense amount of time; however, further plans for this work could include a much more rigorous and intensive data gathering process, so there would be the possibility to perform all the experiments.

Finally, further development of this work can focus on integrating single-frame-based and video object detectors in a self-supervised matter. At the current stage of this work, the self-supervised experiments were performed using the *EfficientDet* model as in the original architecture. However, it would be interesting to use an additional RGB video object detector teacher instead of a single RGB teacher for the possibility of considering the detections of

these two models, as shown in figure 5.2.

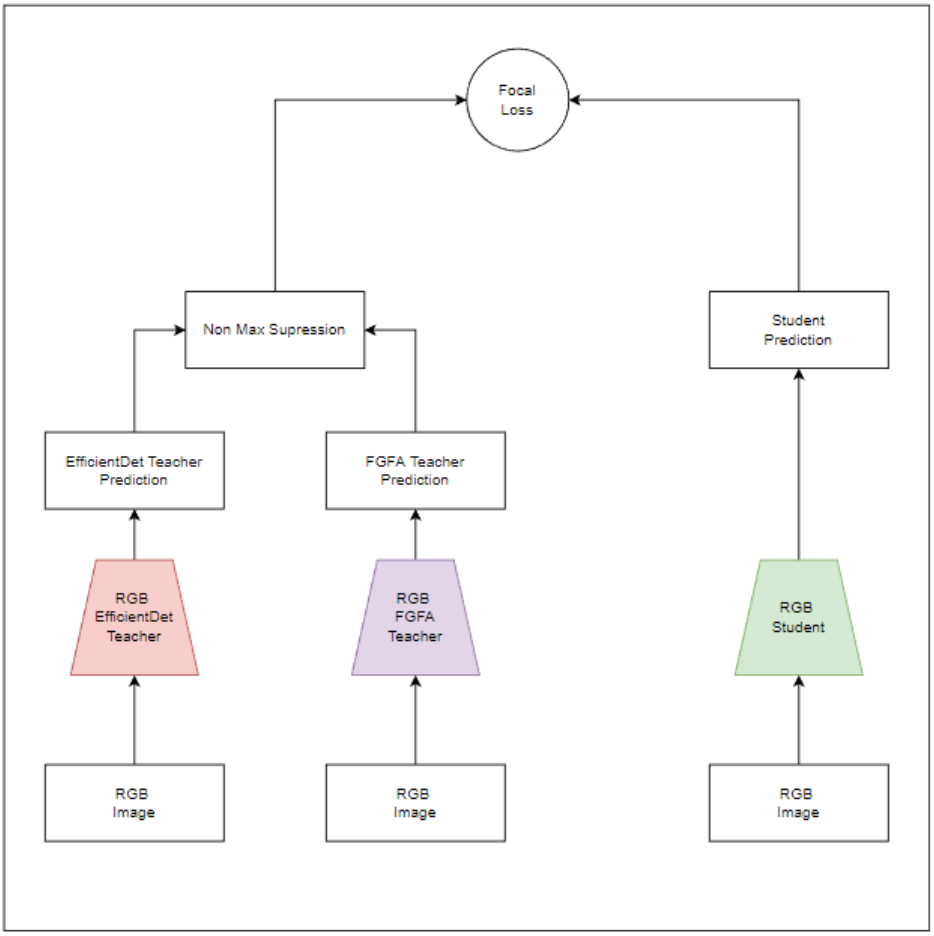

Figure 5.2: Illustration of the new proposed self-supervised architecture.

Primarily, we would obtain precise predictions from the *EfficientDet* and, additionally, we would also only obtain precise predictions from the *FGFA* model by applying background subtraction post-processing but also, as it has been evidenced, we can obtain predictions where objects are partially hidden or obstructed, which single-frame-based methods tend to miss.

Furthermore, gathering and annotating more data in wild forests would be essential to building a complete dataset covering a wide range of scenarios, which would be used to build more robust models. Regarding this, there would also be the need to gather a more significant amount of unlabelled data that would then be used to perform more extensive self-supervised experiments.

Bibliography

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. xv, 5, 6

[2] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083 xv, 7

[3] 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. [Online]. Available: http://arxiv.org/abs/1506.01497 xv, 8

[4] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946 xv, 8, 9

[5] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144 xv, 9, 10

[6] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *CoRR*, vol. abs/1911.09070, 2019. [Online]. Available: http://arxiv.org/abs/1911.09070 xv, 9, 10

[7] G. Xia, X. Bai, J. Ding, Z. Zhu, S. J. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "DOTA: A large-scale dataset for object detection in aerial images," *CoRR*, vol. abs/1711.10398, 2017. [Online]. Available: http://arxiv.org/abs/1711.10398 xv, xvi, xix, 11, 29, 30

[8] J. Han, J. Ding, N. Xue, and G.-S. Xia, "Redet: A rotation-equivariant detector for aerial object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 2786–2795. xv, 12, 13

[9] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *CoRR*, vol. abs/1504.06852, 2015. [Online]. Available: http://arxiv.org/abs/1504.06852 xv, 13, 14

[10] H. Zhu, H. Wei, B. Li, X. Yuan, and N. Kehtarnavaz, "A review of video object detection: Datasets, metrics and methods of video object detection; deep learning-based video object detection," *Applied Sciences*, vol. 10, p. 7834, 11 2020. xv, 15

[11] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," *CoRR*, vol. abs/1611.07715, 2016. [Online]. Available: http://arxiv.org/abs/1611.07715 xv, 14, 15

[12] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," *CoRR*, vol. abs/1703.10025, 2017. [Online]. Available: http://arxiv.org/abs/1703.10025 xv, 16

[13] D.-Y. Kang, H. Duong, and J.-C. Park, "Application of deep learning in dentistry and implantology," *The Korean Academy of Oral and Maxillofacial Implantology*, vol. 24, pp. 148–181, 09 2020. xv, 17

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597 xv, 18, 19

[15] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-resolution representations for labeling pixels and regions," *CoRR*, vol. abs/1904.04514, 2019. [Online]. Available: http://arxiv.org/abs/1904.04514 xv, 19

[16] Sarvasv, "The all-in-one workspace for your notes, tasks, wikis, and databases." [Online]. Available: https://www.notion.so/A-Brief-Introduction-To-GANs-397de071301f4e56b4907a65d93cef7b xv, 20

[17] A. Chaudhary, "The illustrated simclr framework," Mar 2020. [Online]. Available: https://amitness.com/2020/03/illustrated-simclr/ xv, 21

[18] P. Ganesh, "Knowledge distillationnbsp;: Simplified," Oct 2019. [Online]. Available: https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764 xv, 22

[19] F. R. Valverde, J. V. Hurtado, and A. Valada, "There is more than meets the eye: Self-supervised multi-object detection and tracking with sound by distilling multimodal knowledge," *CoRR*, vol. abs/2103.01353, 2021. [Online]. Available: https://arxiv.org/abs/2103.01353 xv, 22, 23

[20] D. Das and S. Saharia, "Implementation and performance evaluation of background subtraction algorithms," *International Journal on Computational Science Applications*, vol. 4, 05 2014. xv, 25

[21] I. Chelly, V. Winter, D. Litvak, D. M. Rosen, and O. Freifeld, "Ja-pols: A moving-camera background model via joint alignment and partially-overlapping local subspaces," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 582–12 591, 2020. xv, 25, 26, 27

[22] W. Mcculloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943. 5

[23] Nielsen, Michael A. (2019) How the backpropagation algorithm works. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap2.html 5

[24] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 826–834, 1983. 5

[25] C. Zitnick and P. Dollar, "Edge boxes : Locating object proposals from edges," vol. 8693, 09 2014. 8

[26] G. Ghiasi, T. Lin, R. Pang, and Q. V. Le, "NAS-FPN: learning scalable feature pyramid architecture for object detection," *CoRR*, vol. abs/1904.07392, 2019. [Online]. Available: http://arxiv.org/abs/1904.07392 10

[27] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010. 11, 23

[28] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312 11, 23

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. 11, 23, 26

[30] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242 11

[31] 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. [Online]. Available: http://arxiv.org/abs/1506.01497 11

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325 11

[33] D. Liang, Z. Wei, D. Zhang, Q. Geng, L. Zhang, H. Sun, H. Zhou, M. Wei, and P. Gao, "Learning calibrated-guidance for object detection in aerial images," *CoRR*, vol. abs/2103.11399, 2021. [Online]. Available: https://arxiv.org/abs/2103.11399 12

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 13

[35] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 13

[36] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370281900242 13

[37] D. Rosenbaum, D. Zoran, and Y. Weiss, "Learning the local statistics of optical flow," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/4a213d37242bdcad8e7300e202e7caa4-Paper.pdf 13

[38] G. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Computer Vision, ECCV 2010 - 11th European Conference on Computer Vision, Proceedings*, no. PART 6. Springer Verlag, 2010, pp. 140–153. 13

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385 19

[40] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870 19

[41] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561 19

[42] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805 20

[43] A. Haghighi and D. Klein, "Prototype-driven learning for sequence models," in *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA: Association for Computational Linguistics, Jun. 2006, pp. 320–327. [Online]. Available: https://aclanthology.org/N06-1041 20

[44] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *J. Mach. Learn. Res.*, vol. 6, pp. 1817–1853, 2005. 20

[45] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): A survey," *IEEE Access*, vol. 7, pp. 36 322–36 333, 2019. 20

[46] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018. [Online]. Available: http://arxiv.org/abs/1807.03748 20

[47] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *CoRR*, vol. abs/2002.05709, 2020. [Online]. Available: https://arxiv.org/abs/2002.05709 21

[48] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014. [Online]. Available: http://arxiv.org/abs/1403.6382 21

[49] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *CoRR*, vol. abs/2006.10029, 2020. [Online]. Available: https://arxiv.org/abs/2006.10029 21

[50] Z. Dai, B. Cai, Y. Lin, and J. Chen, "Up-detr: Unsupervised pre-training for object detection with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 1601–1610. 22

[51] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li, "Dense contrastive learning for self-supervised visual pre-training," *CoRR*, vol. abs/2011.09157, 2020. [Online]. Available: https://arxiv.org/abs/2011.09157 22

[52] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/7dcd340d84f762eba80aa538b0c527f7-Paper.pdf 22

[53] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 23

[54] 23

[55] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 2, pp. 246–252 Vol. 2, 1999. 24

[56] L. Balzano, R. D. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," *CoRR*, vol. abs/1006.4046, 2010. [Online]. Available: http://arxiv.org/abs/1006.4046 24

[57] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1568–1575. 24

[58] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, 2004, pp. 28–31 Vol.2. 25

[59] A. Viswanath, R. K. Behera, V. Senthamilarasu, and K. Kutty, "Background modelling from a moving camera," *Procedia Computer Science*, vol. 58, pp. 289–296, 2015, second International Symposium on Computer Vision and the Internet

(VisionNet'15). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915021341 25

[60] ——, "Background modelling from a moving camera," *Procedia Computer Science*, vol. 58, pp. 289–296, 2015, second International Symposium on Computer Vision and the Internet (VisionNet'15). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915021341 25

[61] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard, "Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group," *The International Journal of Robotics Research*, vol. 38, 12 2016. 26

[62] 26

[63] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842 26

[64] G. Chau and P. Rodriguez, "Panning and jitter invariant incremental principal component pursuit for video background modeling," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017. 27

[65] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 597–610, 2013. 27

[66] B. E. Moore, C. Gao, and R. R. Nadakuditi, "Panoramic robust pca for foreground–background separation on noisy, free-motion camera video," *IEEE Transactions on Computational Imaging*, vol. 5, no. 2, pp. 195–211, 2019. 27

[67] H. Guo, C. Qiu, and N. Vaswani, "Practical reprocs for separating sparse and low-dimensional signal sequences from their sum," *CoRR*, vol. abs/1310.4261, 2013. [Online]. Available: http://arxiv.org/abs/1310.4261 27

[68] G. Cheng, J. Han, P. Zhou, and L. Guo, "Multi-class geospatial object detection and geographic image classification based on collection of part detectors," *Isprs Journal of Photogrammetry and Remote Sensing*, vol. 98, pp. 119–132, 2014. 30

[69] C. Benedek, X. Descombes, and J. Zerubia, "Building development monitoring in multitemporal remotely sensed image pairs with stochastic birth-death dynamics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 33–50, 01 2012. 30

[70] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *ECCV*, 2008. 30

[71] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, "A large contextual dataset for classification, detection and counting of cars with deep learning," *CoRR*, vol. abs/1609.04453, 2016. [Online]. Available: http://arxiv.org/abs/1609.04453 30

[72] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery : A small target detection benchmark," *J. Vis. Commun. Image Represent.*, vol. 34, pp. 187–203, 2016. 30

[73] H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, "Orientation robust object detection in aerial images using deep convolutional neural network," *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 3735–3739, 2015. 30

[74] Z. Liu, H. Wang, L. Weng, and Y. Yang, "Ship rotated bounding box space for ship extraction from high-resolution optical satellite images with complex backgrounds," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 8, pp. 1074–1078, 2016. 30

[75] K. Liu and G. Máttyus, "Fast multiclass vehicle detection on aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, pp. 1938–1942, 2015. 30

Declaração de Integridade

Eu, Pedro Jorge Franco Brito, que abaixo assino, estudante com o número de inscrição 10815 de Engenharia Informática da Faculdade de Engenharia declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 29 /06 /2022