Review

Using Computer Vision to Collect Information on Cycling and Hiking Trails Users

Joaquim Miguel [1], Pedro Mendonça [1], Agnelo Quelhas [2,3], João M. L. P. Caldeira [1,4,*] and Vasco N. G. J. Soares [1,4]

1 Polytechnic Institute of Castelo Branco, Av. Pedro Álvares Cabral n° 12, 6000-084 Castelo Branco, Portugal; joaquim.miguel@ipcbcampus.pt (J.M.); mendonca.pedro@ipcbcampus.pt (P.M.); vasco.g.soares@ipcb.pt (V.N.G.J.S.)
2 Direção Geral da Educação/ERTE, Av. 24 de Julho n.º 140-5.º piso, 1399-025 Lisboa, Portugal; agneloquelhas@ipcb.pt
3 Federação Portuguesa de Ciclismo, Rua de Campolide, 237, 1070-030 Lisboa, Portugal
4 Instituto de Telecomunicações, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
* Correspondence: jcaldeira@ipcb.pt

Abstract: Hiking and cycling have become popular activities for promoting well-being and physical activity. Portugal has been investing in hiking and cycling trail infrastructures to boost sustainable tourism. However, the lack of reliable data on the use of these trails means that the times of greatest affluence or the type of user who makes the most use of them are not recorded. These data are of the utmost importance to the managing bodies, with which they can adjust their actions to improve the management, maintenance, promotion, and use of the infrastructures for which they are responsible. The aim of this work is to present a review study on projects, techniques, and methods that can be used to identify and count the different types of users on these trails. The most promising computer vision techniques are identified and described: YOLOv3-Tiny, MobileNet-SSD V2, and FasterRCNN with ResNet-50. Their performance is evaluated and compared. The results observed can be very useful for proposing future prototypes. The challenges, future directions, and research opportunities are also discussed.

Keywords: hiking trails; cycling trails; computer vision; convolutional neural networks; state of the art; performance evaluation; YOLOv3-Tiny; MobileNet-SSD V2; FasterRCNN with ResNet-50

check for updates

Citation: Miguel, J.; Mendonça, P.; Quelhas, A.; Caldeira, J.M.L.P.; Soares, V.N.G.J. Using Computer Vision to Collect Information on Cycling and Hiking Trails Users. *Future Internet* **2024**, *16*, 104. https://doi.org/10.3390/fi16030104

Academic Editor: Gianluigi Ferrari

Received: 29 January 2024
Revised: 7 March 2024
Accepted: 17 March 2024
Published: 20 March 2024

1. Introduction

In recent years, walking, hiking, and cycling have become one of the most popular activities for promoting physical activity and well-being. These are carried out through trails or routes of varying lengths, classified into three categories depending on the distances involved: Long Routes (GR), routes over 30 km long; Short Routes (PR), less than 30 km long; and Local Routes (PL), where all or more than half of the route is in an urban environment [1]. Each of these trails is duly marked with the signs displayed in Figure 1 along the way. These markings can appear in a wide variety of places, such as rocks, walls, stakes, and electricity pylons, among others [2]. Figure 2 shows some examples of the location of these marks.

Walking, hiking, and cycling trails and routes are created and maintained by various entities, such as federations related to sports, mountaineering, camping, national, regional, or local public administration entities, and even entities linked to the tourism sector [3]. According to the Federation of Camping and Mountaineering of Portugal (FCM) [4], the body responsible for registering and homologating hiking trails, there are 76 small routes and 46 large routes homologated in Portugal, following the Regulation for the Homologation of Walking Routes (RHPP) [1]. This regulation serves to establish specific standards and criteria for a hiking trail to be officially recognized [5]. However, there are many routes that have not been homologated in the National Register of Footpaths (RNPP), managed by the FCM.

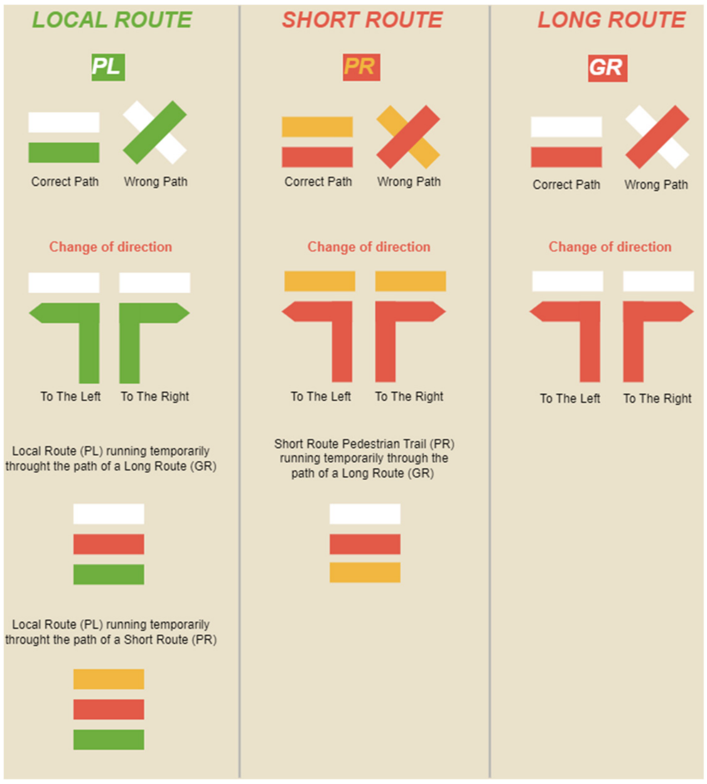

LOCAL ROUTE
PL
Correct Path
Wrong Path
Change of direction
To The Left
To The Right
Local Route (PL) running temporarily throught the path of a Long Route (GR)
Local Route (PL) running temporarily throught the path of a Short Route (PR)
SHORT ROUTE
PR
Correct Path
Wrong Path
Change of direction
To The Left
To The Right
Short Route Pedestrian Trail (PR) running temporarily through the path of a Long Route (GR)
LONG ROUTE
GR
Correct Path
Wrong Path
Change of direction
To The Left
To The Right

Figure 1. Signage used on the routes.

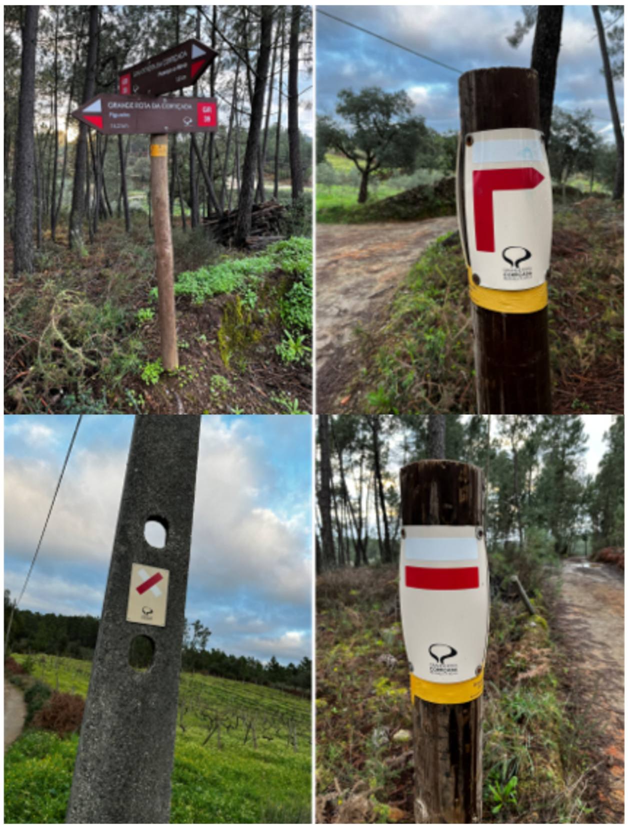


Figure 2. Examples of signposted sites.

In Portugal, the number of approved walking, hiking, and cycling trails and routes has been growing over the years, helping to add value to areas by attracting a growing number of domestic and foreign tourists to practice nature sports and boosting sustainable tourism. However, since these spaces are freely accessible, there is no reliable information available on how often they are used and the types of users that use them most often. Given that, in most cases, municipalities are the driving force behind these infrastructures, it is important to assess, for example, the real tourist impact that these routes bring to the regions. Accurate and reliable data are therefore needed to show how they are used. In this way, identifying the different types of users, counting them, and knowing the times of greatest influx are crucial data for the managers and promoters of these infrastructures. These data can also help gauge the strategic importance of these facilities in tourism development, especially in low-density areas. Another dimension that can be considered when analyzing the data collected is the management of the capacity to use these infrastructures. Given that, along these trails, there are often sensitive areas that have specific preservation regulations (such as natural parks, protected areas, and special protection zones, among others), the collection of data on usage rates is extremely important for managing access to these areas.

Considering the importance of these data in better managing these infrastructures, there is a need to implement solutions that allow these data to be collected, processed, and analyzed. The work presented in this paper represents a first step in an ongoing effort to propose, develop, and evaluate a solution that allows for identifying and quantifying different types of users on walking, hiking, and cycling trails or routes. To achieve this goal, and after a preliminary analysis, it was decided to study solutions that use computer vision techniques, since, as demonstrated in [6,7], these stand out as the main technique for detecting moving objects. Moreover, computer vision demonstrates greater versatility and adaptability to various conditions compared to other technologies like LiDAR [8]. Opting for its use not only contributes to the evolution of this field but also ensures the availability of more data for future utilization [9]. The cost effectiveness of essential hardware, such as cameras and single-board computers, together with the widespread availability of open-source software in this field, further encourages the option of using these technologies. This paper surveys related work on the topic. It analyzes computer vision techniques suitable for solving this problem and evaluates the most promising ones using a new dataset that has been created in the context of this work. The dataset was composed of several images collected by the authors, among others, from two platforms available on the web. All the images were then manually annotated using the Roboflow platform [10].

The rest of the paper is organized as follows. Section 2 presents related work that provides the context for the current research. Section 3 discusses computer vision techniques and convolutional neural network (CNN) architectures, including one-stage detection and two-stage detection, and analyzes YOLOv3-Tiny, MobileNet-SSD V2, and Faster-RCNN with ResNet-50. Section 4 presents a performance evaluation of these models. Finally, Section 5 presents the conclusions and suggests possible future works.

2. Related Work

This section provides context and a comprehensive review of previous work and research related to the topic of this work. The Preferred Reporting Items for Systematic Reviews and Meta-analyses (PRISMA) methodology [11] was used to conduct this research. This methodology consists of a framework for reporting on systematic reviews and meta-analyses. The main objective of PRISMA is to guarantee transparency, quality, and consistency in the conduct and presentation of systematic reviews and meta-analyses, and it can be adapted to carry out a broad search for works related to this project [11]. Thus, the methodology implements the following procedures: (a) research questions, (b) information sources and research strategy, (c) selection process, (d) analysis of studies, and (e) results obtained.

2.1. Research Questions

This procedure encourages questions to be drawn from the purpose of this work. These will help to conduct clear research.

- Question No. 1—How can different types of users (cyclists and pedestrians) on cycling and hiking trails and routes be distinguished?
- Question No. 2—How can different types of users (cyclists and pedestrians) on cycling and hiking trails and routes be counted?
- Question No. 3—What frameworks exist for solutions to detect different types of users on cycling and hiking trails and routes?

2.2. Information Sources and Research Strategy

The articles studied were identified manually and through a search using the b-on platform (Online Knowledge Library) [12]. b-on is a virtual library that provides titles of international scientific periodicals and e-books from the most important content providers. Through the platform, it is possible to access document databases, including Science Direct, IEEE Xplore, and SpringerLink, among others, enabling access to the full text of a wide range of publications [13].

In the advanced search, carried out between November and December 2023, a query was prepared using terms suitable for the search, as follows:

("object detection" OR "people detection ") AND "counting" AND "computer vision".

2.3. Selection Process

To select the studies resulting from the research, it was necessary to apply inclusion filters. Firstly, the search was filtered by documents published between 2019 and 2023 and duplicate studies were eliminated from the search. Once the search was narrowed down, the titles of the remaining articles were analyzed to identify which of them could potentially meet the inclusion criteria. Some titles were promptly excluded due to their clear lack of relevance to the theme of the work, while others, which initially presented a less obvious connection, were kept for a more detailed evaluation in the subsequent stage. Of those selected by title, their abstracts were then analyzed to conclude whether they should be included or excluded. At this stage, those using expensive technologies such as unmanned aerial vehicles (UAV) were also excluded. Articles that lacked clarity in their descriptions, omitting information on the models used and the results obtained, were also excluded. The resulting articles were then selected as part of the systematic review and the full text of all of them was read. Figure 3 summarizes this whole process.

2.4. Analysis of Studies

This section presents an analysis of the projects and works related to the topic covered in this article, which resulted from the previous selection process. Table 1 summarizes the main characteristics considered in this analysis (title of the study, year of publication, model used, dataset used, methodologies used, and results).

Table 1. Summary of the articles studied.

References	Study	Year	Dataset	Methodologies	Results
[14]	Vehicle Counting on Vietnamese Street	2023	Dataset created by the author	YOLOv8, StrongSORT	mAP of 82.9%
[15]	Development of Automated People Counting System using Object Detection and Tracking	2023	MS-COCO	Mask R-CNN, ResNet-50	mAP of 100% and 97.62% in plans with simple funds and 85.73% in complex funds

Table 1. *Cont.*

References	Study	Year	Dataset	Methodologies	Results
[16]	Pedestrian and Object Detection using Image Processing by YOLOv3 and YOLOv2	2023	MS-COCO	YOLOv3	mAP of 57.9%
[17]	Vehicle Counting based on Convolution NeuralNetwork	2023	UA-DETRAC	YOLOv3, SORT	Counting accuracy of 85.45%
[18]	People Detecting and Counting System	2021	ImageNet	ImageNet	N/A
[19]	Intelligent multimodal pedestrian detection using hybrid metaheuristic optimization with deep learning model	2023	UCSD (Ped-1 e Ped-2)	YOLO-v5, RetinaNet	AUC score of 98.86% (Ped-1) and 97.58% (Ped-2)
[20]	Realtime Vehicle Counting Method Using Haar Cascade Classifier Model	2021	3 min video (origin not described)	Haar Cascade Classifier algorithm	mAP of 91.2%
[7]	Real-time Train Wagon Counting and Number Recognition Algorithm	2022	Dataset created by the authors	ResNet-18, ResNet-34, ResNet-50	Accuracy of 99.2% at 36 FPS (ResNet-18), 99.4% at 22 FPS (ResNet-34), and 99.7% at 10 FPS (ResNet-50)
[21]	Improved Person Counting Performance Using Kalman Filter Based on Image Detection and Tracking	2021	N/A	YOLOv3 e Kalman Filter	N/A
[22]	Performance Evaluation of Deep Learning Models on Embedded Platform	2021	Dataset created by the authors	YOLOv4, MobileNet-SSD	mAP of 91% with 7.2 FPS (YOLOv4) and 87.5% with 40 FPS (MobileNet-SSD)
[23]	EmbeddedPigCount: Pig Counting with Video Object Detection and Tracking on an Embedded Board	2022	Hallway, pig pen, people in top view	LightYOLOv4, DeepSORT	mAP of 94.95%
[24]	Counting People and Bicycles in Real Time Using YOLO on Jetson Nano	2022	Dataset created by the authors	Yolov5, V-IOU	mAP of 44.4%

In [14], researchers in Vietnam studied a system for tracking vehicles on roads using a custom dataset and the YOLOv8s model. The custom dataset, containing images of Vietnamese traffic under various conditions, significantly outperformed a generic dataset (COCO) in terms of accuracy (82.9% vs. 15.6%). This suggests that YOLOv8 performs well on data similar to its training data but might be affected by factors like weather and camera angles.

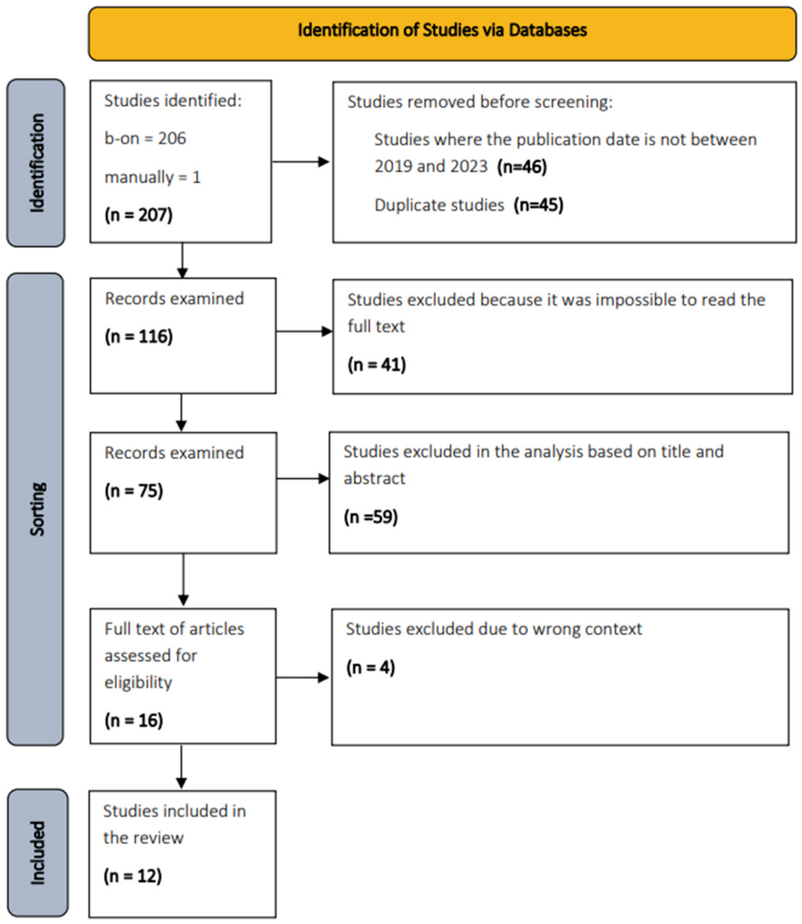

Identification of Studies via Databases
Identification
Sorting
Included
Studies identified:
b-on = 206
manually = 1
(n = 207)
Studies removed before screening:
Studies where the publication date is not between 2019 and 2023 (n=46)
Duplicate studies (n=45)
Records examined
(n = 116)
Studies excluded because it was impossible to read the full text
(n = 41)
Records examined
(n = 75)
Studies excluded in the analysis based on title and abstract
(n =59)
Full text of articles assessed for eligibility
(n = 16)
Studies excluded due to wrong context
(n = 4)
Studies included in the review
(n = 12)

Figure 3. Flowchart of the research phases.

In [15], the paper describes an offline people-counting system using video files. It employs Mask R-CNN for detection, is trained on a large dataset (MS-COCO), and uses background subtraction, blob analysis, and centroid setting for counting. Additionally, it integrates a notification system through ThingSpeak and an old social network (potentially Twitter) to display the count. The system achieved high accuracy (100% and 97.62%) in simple test scenarios and a lower, but still reasonable, accuracy (85.73%) in a more complex scenario.

In [16], this study proposes a real-time object detection system for vehicles, aiming to enhance road safety by identifying pedestrians and other objects and notifying drivers of potential hazards. Utilizing the COCO dataset, the system compares the performance of YOLOv2 and YOLOv3 algorithms. The findings reveal that YOLOv3 achieves superior accuracy (approximately 58% mAP) and operates at a faster refresh rate, rendering it more suitable for real-time applications. Conversely, YOLOv2 exhibits limitations in both object detection and precision, hindering its effectiveness in this context.

In [17], this study presents a method for vehicle detection, classification, and counting using a Virtual Detection Zone (VDZ) aimed at simplifying traffic management applications. The UA-DETRAC dataset was used for evaluation. The system leverages OpenCV, the YOLOv3 model (with Darknet-53 as the feature extractor), and the SORT algorithm for tracking. While minor challenges with duplicate counting were observed in specific scenarios, the system demonstrated a high overall accuracy of approximately 85.45%.

In [18], a people counting system for monitoring entry and exit points is presented in this study. The system leverages a Raspberry Pi for image capture, model execution, and data transmission to a web server. Trained on the ImageNet dataset, the model employs Support Vector Machines alongside HOG and VGG-16 models to classify objects as “person” or “non-person”. Both models exhibit comparable accuracy, although HOG extracts a substantially higher number of features compared to VGG-16.

In [19], a multimodal pedestrian detection and classification system, IPDC-HMODL, is introduced in this work. This deep learning-based model, employing techniques like hybrid metaheuristic optimization and kernel extreme learning machines, demonstrates superior performance on the UCSD Ped-1 and Ped-2 datasets compared to existing methods. IPDC-HMODL achieves exceptional Area Under the Curve (AUC) scores (98.86% and 97.58%) and low error rates (3.12% and 4.35%) on both datasets, surpassing established models like AMDN, MDT, and Social Force. Receiver Operating Characteristic (ROC) analysis further emphasizes IPDC-HMODL’s efficacy, highlighting its potential for real-world applications in pedestrian detection and classification tasks.

In [20], a method was described for the real-time counting of vehicles through the application of computer vision in conjunction with a Single-board Computer (SBC). The main objective was to present an efficient system capable of identifying and counting vehicles entering and leaving certain entrances. The dataset used for evaluation was not explicitly detailed. The hardware used consisted of a Raspberry Pi equipped with a camera module. The software used was the Haar Cascade Classifier algorithm, implemented in the Python programming language. The test results showed that the proposed system achieved an average accuracy of 91.2% in detecting and accurately counting moving vehicles.

In [7], a solution was proposed for train carriage counting and number recognition using deep learning computer vision models. The system processes a video stream into frames and uses three datasets: “Scene Classifier” with 7000 backgrounds, 12,000 intervals, and 11,000 wagon images to train the classifier; “Number detection model” with 3400 wagon images for detecting number edges; and “digit recognition model” with 3400 images for recognizing digits. The first phase validated wagon counting with 99.5% accuracy using ResNet models, while the second phase achieved 98.5% accuracy in recognizing wagon numbers. This approach relies solely on computer vision without additional devices or sensors.

In [21], an approach to people counting in image detection and tracking was introduced. This research introduced an approach to counting people at a bus stop using image detection and tracking. They used NVIDIA Jetson AGX Xavier hardware and NVIDIA DeepStream SDK 5.0 with Python. The method implemented a Kalman filter for counting people, resulting in a notable reduction in counting errors compared to a basic detection approach. Specific details about the dataset were not provided.

In [22], the study analyzed the performance of deep learning models for real-time traffic localization and detection using edge equipment. The dataset, created by the authors, included images of Vietnamese vehicles and license plates. They used Nvidia Jetson Nano as the hardware platform and compared MobileNet-SSD and YOLOv4 models for vehicle counting and license plate detection. Both models showed high accuracy in real-time tasks, with MobileNet-SSD processing at 40 frames per second (FPS) and YOLOv4 demonstrating superior detection of smaller objects with around 91% mean Average Precision (mAP).

In [23], researchers developed an automatic camera-based method to count pigs in a specific area of a large-scale pig farm. They used three datasets, two created manually and one reused from a previous article. The hardware included cameras for video capture, and a deep learning algorithm, specifically a convolutional neural network (CNN), was used for pig identification and localization. The DeepSORT (Deep Simple Online and Real-time Tracking) algorithm tracked the detected pigs in real time, allowing for precise and individualized counting with around 94.95% mean Average Precision (mAP). This system provided an automated solution for accurate and real-time pig counting in agricultural settings, ensuring continuous monitoring and tracking of pig trajectories.

In [24], a system was developed for real-time counting of people and bicycles using a private dataset of 339 videos. The dataset consisted of 4 s videos showing people and bicycles in motion, from which 630 frames were extracted and annotated on the Roboflow platform. The system was implemented on a Nvidia Jetson Nano 2 GB. The study compared various detection models including Faster R-CNN, SSD300, SSD512, YOLOv4, YOLOv4-tiny, YOLOv5n, YOLOv5m, YOLOv5x, YOLOv5n6, YOLOv5m6, and YOLOv5x6, with an average mean Average Precision (mAP) of 44.4%. YOLOv5 showed a higher refresh rate than Faster R-CNN and SSD models and was supported by PyTorch and TensorRT. Object tracking used IOU, V-IOU, and DeepSORT methods, with V-IOU proving superior to IOU for tracking. Tests on YOLOv5 versions with TensorRT showed better results than those with PyTorch. Overall, the system demonstrated efficient detection and tracking, with the V-IOU tracking method outperforming IOU.

2.5. Results Obtained

As described above, a filter was applied by year of publication, from 2019 to 2023, during the search for studies. It should be noted that no studies were found in years 2019 and 2020. Figure 4 shows the percentage of studies per year.

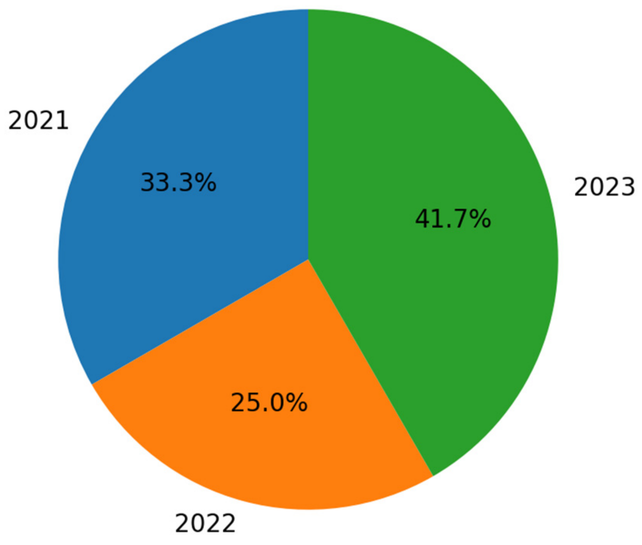


Figure 4. Publication years of the articles.

When analyzing the articles selected for this study, there is a diversity in the implementation of detection models. Each article adopts its own approach, with some also implementing tracking methods. In addition, the datasets are used for different purposes. All in all, the results differ in terms of accuracy. All the articles used convolutional neural networks. The results obtained stand out for their remarkable effectiveness in detecting people, bicycles, and motorcycles. This evidence reinforces the prominent position of these architectures as the most efficient and effective option to identify and count users of cycling and hiking trails.

Regarding the datasets used, there is a wide range of choices for training, validating, and testing the models, as shown in Figure 5. Most of the authors created their own dataset, although the use of MS-COCO was a common practice.

Various detection models were reported in the work surveyed, as shown in Figure 6. The most frequently cited was YOLOv3, which was mentioned in three studies. YOLOv5 and ResNet-50 also stood out, each having been used in two studies.

In addition to detection models, methods have also been used to track objects throughout the sequence of frames in which they are present. Figure 7 shows the various methods used by the authors.

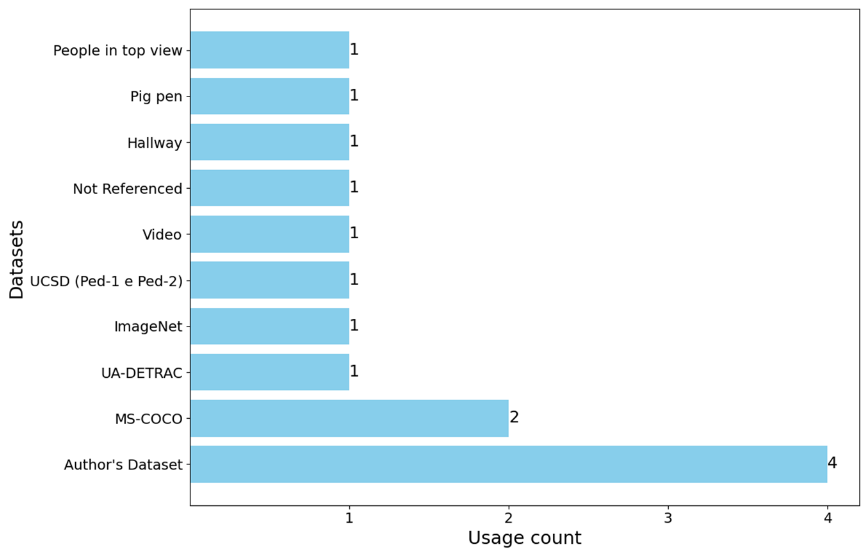


Figure 5. Datasets used.

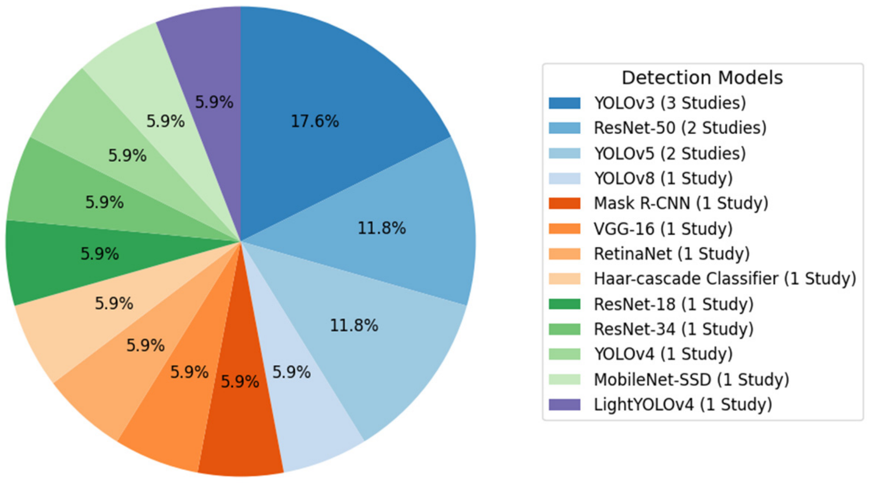


Figure 6. Detection models used.

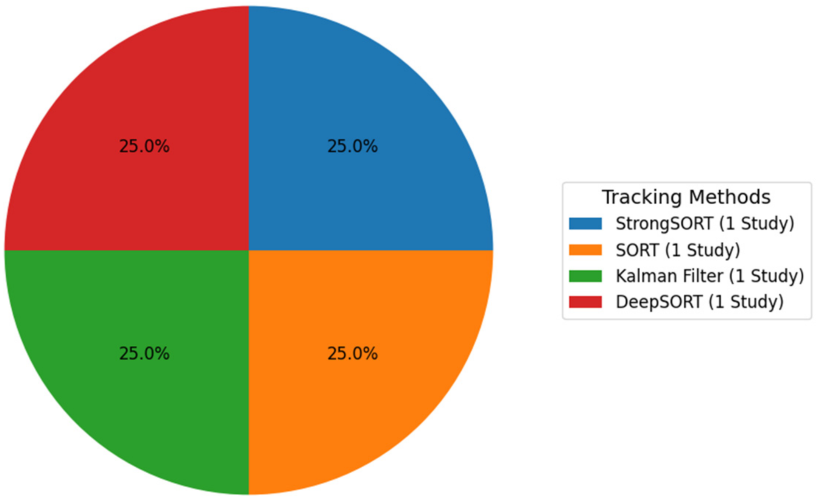


Figure 7. Tracking methods used.

2.6. Critical Analysis

Throughout this study, several limitations were noted in the works analyzed, such as the absence of methodologies and datasets used to implement the solutions. In addition, some results are not explicitly presented. However, it was concluded that there are similarities in implementation between studies.

After analyzing all the articles obtained, only 12 of them fall within the scope of this work. It should be noted that the choice of the CNN model to be used for detecting users of cycling and hiking trails and routes is very important. Not all detection models are equally efficient in terms of resource consumption, making it important to select based on the implementation environment. In the scenario of this work, where the system will be implemented in a remote location, it is anticipated that the hardware used may present computational limitations. It is therefore necessary to strike a balance between the accuracy of the model and the hardware's ability to respond to that model.

Based on the analysis of the related studies, three models were identified that could potentially have the best results in the development of the solution to be proposed, for the reasons described below. YOLOv3 was the most widely used model in the studies analyzed and is therefore included in this selection. In the future, its Tiny version will be used, which is recognized for being lighter, more efficient, and less computationally demanding. The choice of the MobileNet-SSD model was based on its use in embedded boards with few computing resources. This selection was based on the model's ability to offer satisfactory performance (87.5% mAP [19]). Finally, the ResNet-50 model was also selected due to its remarkable accuracy performance in [7,15], oscillating between 85% and 100% in various test conditions. It is also highlighted as the second most used model in the related works studied, demonstrating the relevance and efficiency of this model in different contexts. Considering that the system will be placed in remote locations (such as pedestrian routes), access to operational resources will probably be limited. Therefore, the option for a system with low power consumption should be considered.

The adaptation of the dataset to the context is also a point of analysis. In the existing studies, no dataset has been identified that is perfectly aligned with the requirements of the work described in this paper. It is therefore essential to build one that encompasses people and bicycles and considers the possibility of including motorcycles. It is also important that the images that make up the dataset are captured on footpaths or in similar scenarios, such as dirt tracks. In this way, the development of the dataset plays a key role in training and testing the model to achieve more accurate detection.

Finally, based on the analysis of the previous studies, it is now possible to answer the research questions.

Question No. 1—How can different types of users (cyclists and pedestrians) on cycling and hiking trails and routes be distinguished?

The classification of different types of users on cycling and pedestrian routes can be carried out using computer vision and machine learning (ML) techniques. A computer vision system can be trained to recognize distinctive features between cyclists and pedestrians, using convolutional neural networks to learn patterns in the data received from real-time video frames, thus distinguishing pedestrians from cyclists. All the articles studied present a detection model based on CNNs.

Question No. 2—How can different types of users (cyclists and pedestrians) on cycling and hiking trails and routes be counted?

Quantifying different types of users on cycling and pedestrian routes involves counting the number of cyclists and pedestrians present in a scene, which must first be detected. Object detection techniques can therefore be used to identify and locate cyclists and pedestrians in a video frame in real time. Counting is then carried out based on the detections. This can be completed using the inputs and outputs of an ROI, as in [14,21].

Question No. 3—What frameworks exist for solutions to detect different types of users on cycling and hiking trails and routes?

Tensorflow, TensorRT, Keras, and PyTorch are examples of ML and deep learning frameworks. In the article [23], TensorRT was used and in [24], a comparison of accuracy between TensorRT and PyTorch was carried out. Tensorflow was the framework chosen in [18,20].

The next section presents the definition of computer vision, as well as CNN architectures, highlighting the one-stage detection and two-stage detection approaches. As expected, the characteristics of the following CNN models will be presented: YOLOv3-Tiny, MobileNet-SSD V2, and ResNet-50.

3. Computer Vision Techniques

Computer vision is a technology that makes it possible to develop systems capable of understanding and interpreting visual information and thus describing it accurately and efficiently, automatically. Computer vision applications use artificial intelligence and machine learning. They use a vast number of images and videos captured by different devices, such as smartphones, surveillance cameras, and traffic systems, among others. These data are processed for object identification and facial recognition, as well as classification, recommendation, monitoring, and detection. In the future, various computational performance tests will be carried out to determine the best hardware solution for developing the proposed solution.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans (i.e., learn by example), by creating and training convolutional neural networks. Training is carried out based on datasets, such as a dataset of images [25,26]. This field has evolved considerably in recent decades thanks to advances in image processing capacity and the increased availability of more data [27]. In this way, computer vision plays an important role in the context of the work presented in this paper, since the aim is to study the use of cameras to capture visual information from pedestrians, cyclists, or other users on walking, hiking, and cycling trails and routes. Figure 8 represents the concept described.

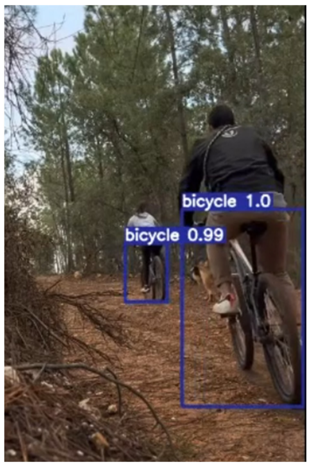


Figure 8. Example of cyclist classification and detection.

Throughout this section, convolutional neural networks and their layered architecture will be explored. In addition, one-stage and two-stage detection strategies will be discussed, providing examples of models that implement these approaches. Finally, provides details regarding the most promising models to be considered for implementation in the future prototype.

3.1. CNN Architectures

Most computer vision algorithms that detect objects in real time use CNNs, as they are generally the ideal choice [28], since they are especially effective at extracting features from images, identifying relevant patterns, and locating objects within a scene [29], efficiently and quickly. Convolutional neural networks are deep learning networks, influenced by the functions and organization of the visual cortex. They are designed to resemble the behavior of neurons in a human brain [30].

Figure 9 shows the structure of convolutional neural networks. They are made up of convolutional, pooling, and fully connected layers, each of which has a specific function in propagating the input image. The convolutional layers are responsible for extracting features from the previous layers or from the input image if it is the first layer of the network. The main function of the pooling layers is to simplify the information at the output of the convolutional layer [31], reducing the size of the data, as well as helping to make the representation constant in small translations of the input. The deep learning used with convolutional neural networks is characterized by the repetition of the convolutional and pooling layers [32]. Finally, the fully connected layer is responsible for propagating the signal through point-to-point multiplication and the use of an activation function. The last layer outputs the probability of an input image belonging to one of the classes for which the network was trained, depending on the activation function used [33].

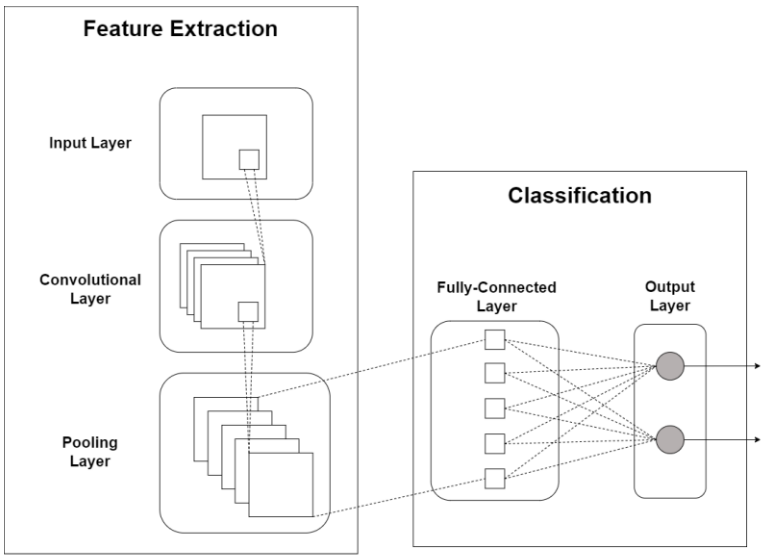


Figure 9. Simplified architecture of CNNs.

Object detection consists of two operations: object localization, which detects where an object is present, and object classification, which determines the class to which the object belongs. However, finding the location of objects using CNNs is challenging due to differences in image views, sizes, postures, and lighting conditions [34]. Thus, the models that can be used for these tasks fall into two approaches: one-stage detection and two-stage detection. To summarize, two-stage detection models tend to achieve greater precision in object detection [35], while one-stage detection focuses more on speed [36]. Each of these approaches will be presented below. Figure 10 shows three examples of models for each of them.

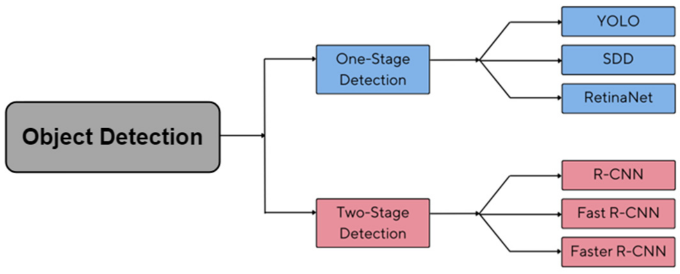


Figure 10. Classification of models in one-stage detection and two-stage detection.

One-Stage Detection

One-stage detection models simultaneously localize and classify objects, as illustrated in Figure 11. Initially, the images are analyzed by a feature extractor using CNN. The extracted features are then used directly for classification and regression of the bounding box coordinates [37].

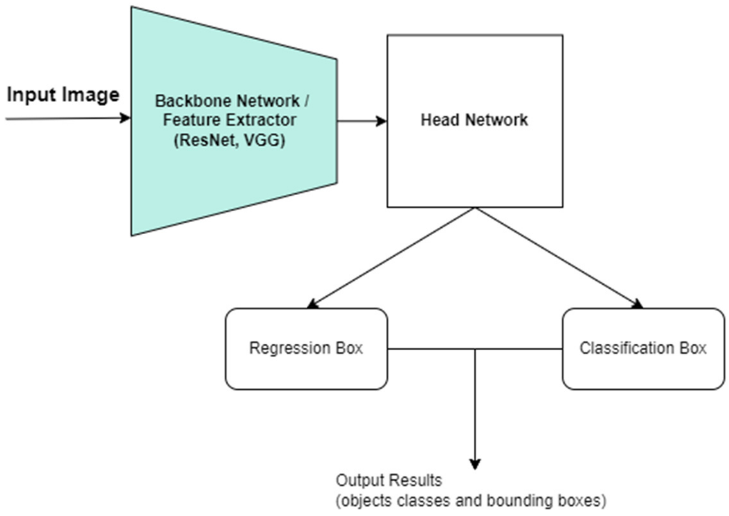

Input Image
Backbone Network / Feature Extractor (ResNet, VGG)
Head Network
Regression Box
Classification Box
Output Results
(objects classes and bounding boxes)

Figure 11. One-stage detection model architecture.

Compared to two-stage detection, these models are much faster and strike a balance between speed and detection accuracy [36]. Because of their speed, they can be used for real-time object detection.

Two-Stage Detection

Two-stage detection solves the problem of detection in two phases, as illustrated in Figure 12. The first stage involves creating an ROI and extracting features using CNNs. The second phase consists of putting the results identified in the ROI into the SVM or CNN-based classifier to classify the objects and then correcting the object positions using bounding box regression [36].

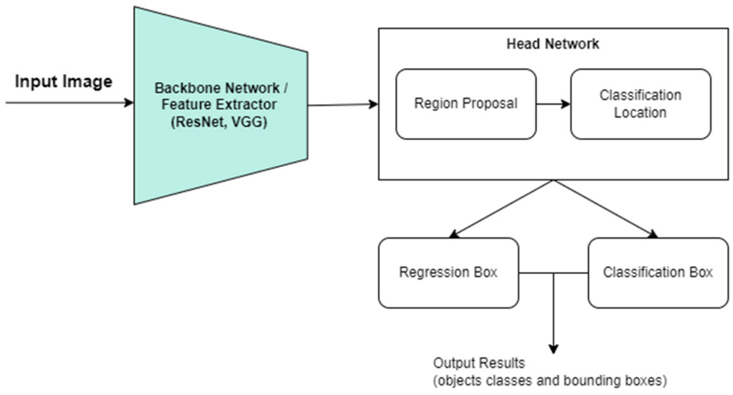


Figure 12. Two-stage detection model architecture.

3.2. CNN Model Analysis

In this subsection, the possible models to be used for the future implementation of the prototype for detecting and counting different users of cycling and hiking trails and routes will be analyzed.

3.2.1. YOLOv3-Tiny

YOLOv3 [38] is a real-time object detection model. It performs one-stage detection, so it processes the entire image only once to try to find objects. It divides the image into a grid of 3 × 3 cells, and each cell is responsible for predicting whether or not there is an object in its area, as shown in Figure 13. For each cell, the model makes a prediction consisting of a bounding box representing the area of the object in the image, a probability indicating the presence of an object within that box, and the class of the object, where applicable. This process offers high precision and efficiency in object detection [39].

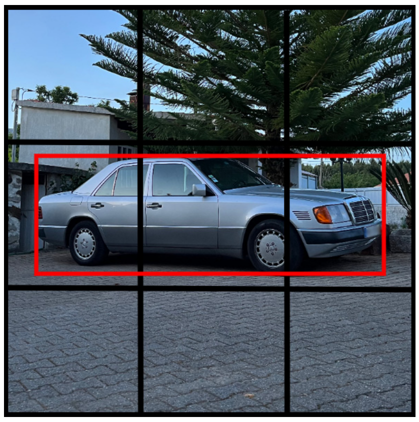


Figure 13. 3 × 3 frame division.

YOLOv3-tiny [40] is a reduced version, designed to be more efficient in terms of computational resources. It uses 13 convolutional layers, a much smaller number than the 106 layers present in the standard version (53 convolutional layers added to the 53 layers of the Darknet-53 feature extractor used by YOLOv3) [41]. As this architecture has fewer filters, it reduces the size of the model and the training time. Figure 14 shows the architecture of the YOLOv3-Tiny model. It is an excellent option for applications that require high computational efficiency, such as mobile devices or embedded systems [38].

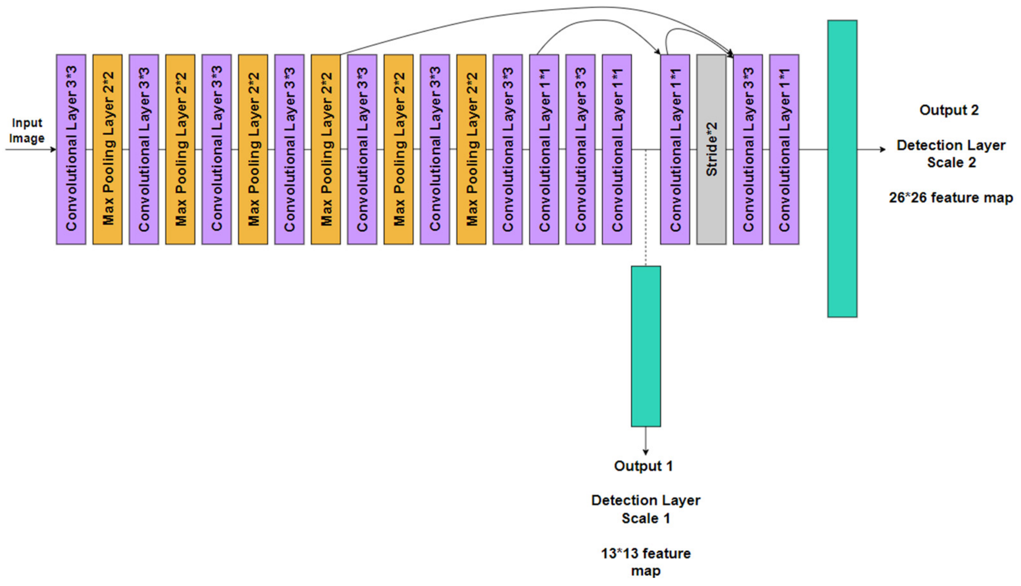


Figure 14. YOLOv3-Tiny architecture.

3.2.2. MobileNet-SSD V2

MobileNet-SSD V2 (MobileNet Single Shot Detector) is a model specially designed for mobile devices such as smartphones, tablets, and embedded systems with low computing power. This Single Shot Detector (SSD) object detection model uses MobileNet as a backbone and can perform fast object detection [42,43]. Compared to the previous version (V1), there was a significant reduction in the complexity, cost, and size of the model to make it possible to use it on devices with low computing power. The MobileNet-SSD V2 model has an architecture divided into two distinct parts. In the first phase, the MobileNetV2 base network plays the role of extracting characteristics. Two types of block are incorporated, a residual block with a stride of one (processing the image information pixel by pixel) and others with a stride of two (every two pixels). The purpose of these blocks is to reduce the number of parameters and improve the network's performance, contributing to efficient visual feature extraction. In the second phase, the SSD layer uses the features extracted by MobileNetV2 to detect and classify objects in the image.

MobileNet-SSD V2 was thus designed to achieve competitive accuracy with few parameters and minimal computational complexity, with the advantage of having a faster inference time compared to the V1 version. It thus proves to be effective in image classification, object detection, and image segmentation [44]. Figure 15 shows the architecture of this model.

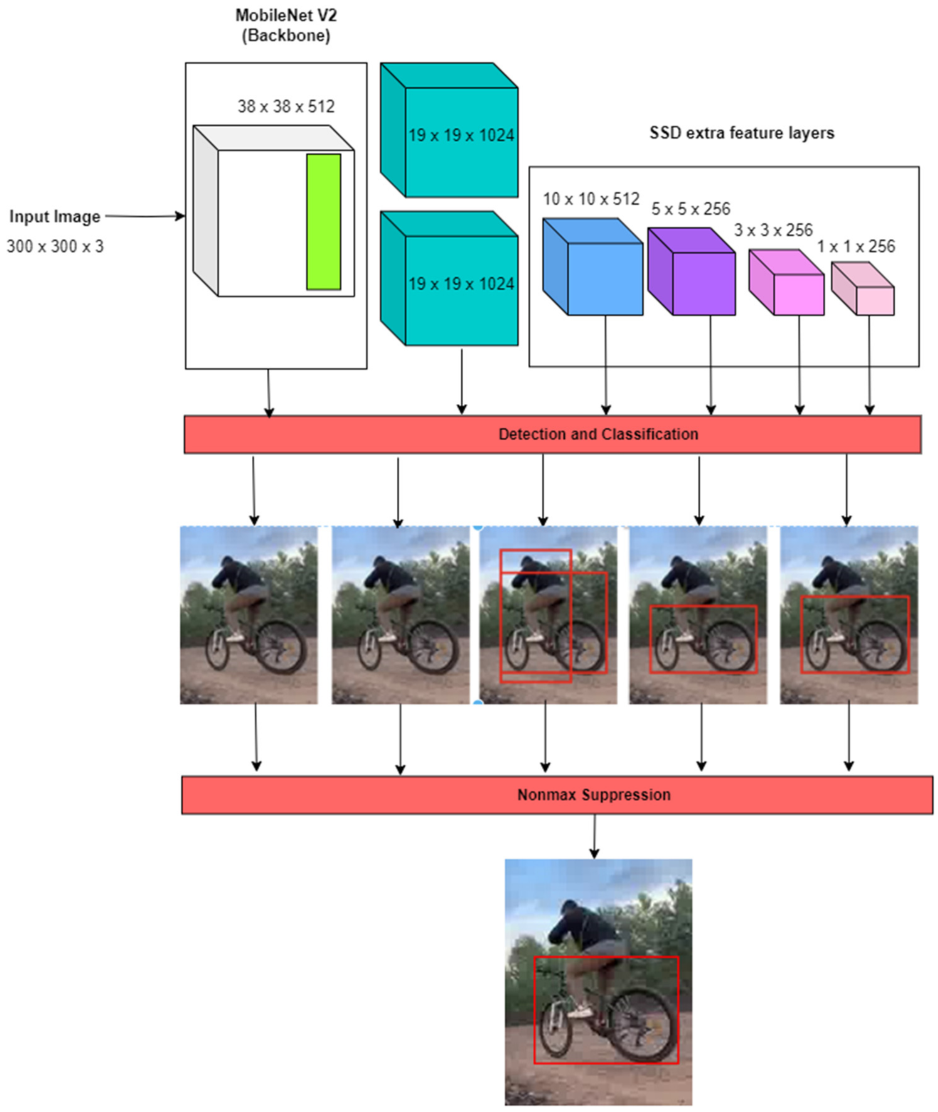


Figure 15. MobileNet-SSD V2 architecture.

3.2.3. FasterRCNN and ResNet-50

Faster R-CNN with ResNet-50 is an object detection implementation technique that integrates these two technologies [45]. The term Faster R-CNN refers to an approach that divides the image into regions and then uses a CNN to process these regions and identify objects. ResNet-50 is a deep neural network architecture known as a "residual network", which aims to overcome training challenges in these networks [46]. As can be seen in Figure 16, by joining these two technologies, one benefits from ResNet-50's ability to extract complex features from images, while Faster R-CNN handles the task of object detection [47].

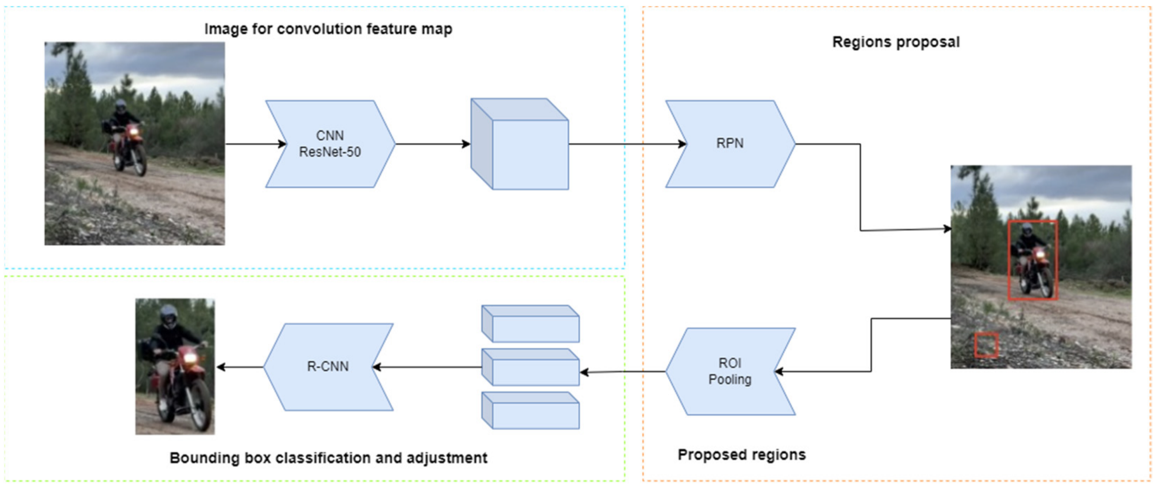


Image for convolution feature map
CNN ResNet-50
Regions proposal
RPN
ROI Pooling
Proposed regions
R-CNN
Bounding box classification and adjustment

Figure 16. FasterRCNN and ResNet-50 architecture.

4. Performance Evaluation

This section presents a performance comparison between the most promising models identified above. Thus, YOLOv3-Tiny, MobileNet-SSD V2, and FasterRCNN with ResNet-50 models will be evaluated for detecting pedestrians, cyclists, or other users on walking, hiking, and cycling trails and routes. Firstly, the dataset created for this work is presented. Secondly, the benchmark scenario and the performance metrics are described. And thirdly, the results are presented, analyzed, and discussed.

4.1. Dataset Description

After extensive research, it was decided that we create a new dataset of images that allowed joint and individual detection of people, motorcycles, and bicycles. This dataset will be used to train and validate the models under analysis. Various images were selected from [48,49]. For both websites, there is no need for special licenses to use their images. In addition to these images, some proprietary images were also captured in an environment that resembles the real context of this work. These images were captured with an iPhone 13 in a vertical orientation. They were taken in two different positions, some at ground level and others about 1 m above the ground. The final dataset is available in [50]. It is categorized into three different classes: persons, bicycles, and motorcycles. It consists of 440 images, organized into three subsets: 70% for training, consisting of 309 images; 20% for validation, comprising 89 images; and 10% for testing, with 42 images. The discrepancy in the number of images between the person class and the other two classes is due to the variety of accessories that people can carry with them, such as backpacks, hiking poles, overcoats, and raincoats. For the object detection model to be able to recognize these accessories, a greater number of images of this class were required. Table 2 shows the number of images per set and per class in the dataset.

Table 2. Number of images per set and class in the dataset.

	No. of Training Images	No. of Validation Images	No. of Test Images
Persons	180	53	25
Bicycles	63	17	9
Motorcycles	68	19	8
Total	311	89	42

The Roboflow tool [10] was used for the labeling process [51], where annotations are created for each image. The goal of this phase is to indicate to the model the location of the objects in the images, as well as identify their class. Figure 17 illustrates the process of creating image annotations with Roboflow, while Figure 18 shows the images already annotated.

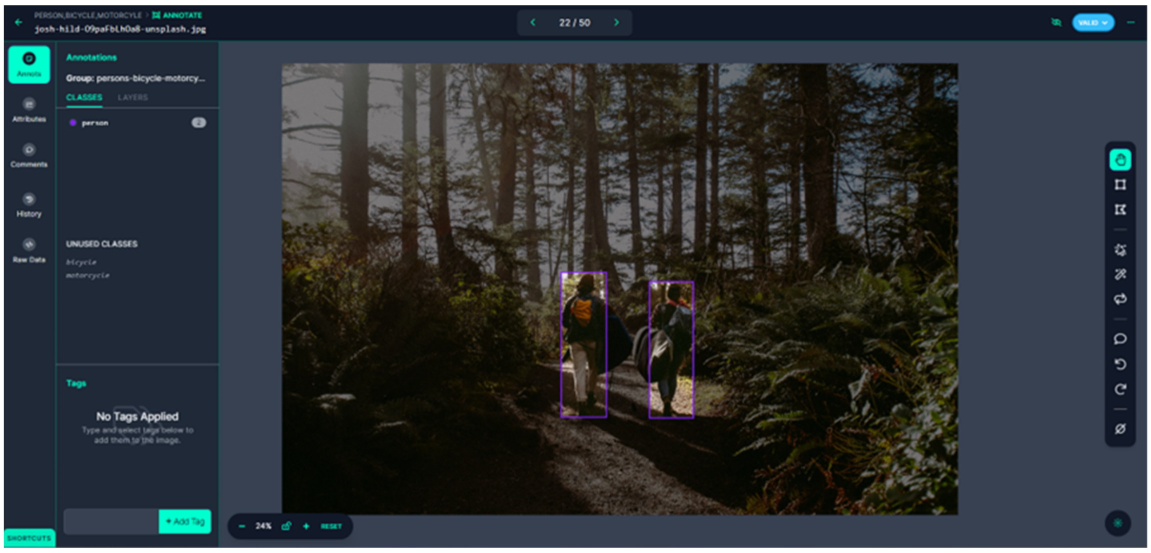

PERSON,BICYCLE,MOTORCYLE > ANNOTATE
josh-hild-O9paFbLh0a8-unsplash.jpg
22 / 50
Annotations
Group: persons-bicycle-motorcy...
CLASSES
LAYERS
person
UNUSED CLASSES
bicycle
motorcycle
Attributes
Comments
History
Raw Data
Tags
No Tags Applied
Type and select tags below to add them to the image.
+ Add Tag
SHORTCUTS
24%
RESET

Figure 17. Image annotation on the Roboflow platform.

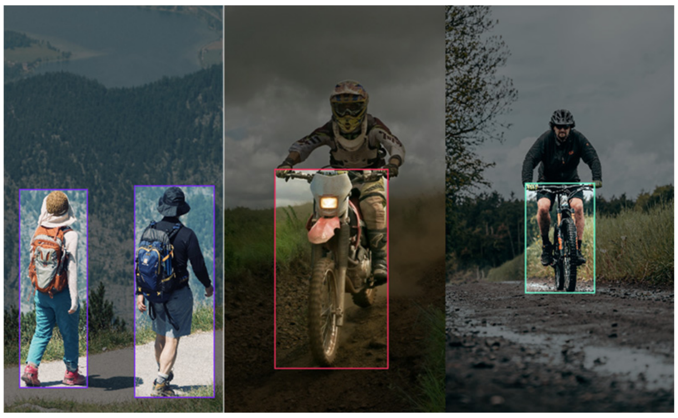


Figure 18. Annotation of three images from the dataset with their respective classes.

Images captured in adverse weather and lighting conditions were annotated and incorporated into the dataset, as shown in Figure 19. These images enrich the diversity of the dataset, giving the detection model the ability to deal effectively with different real-world scenarios.

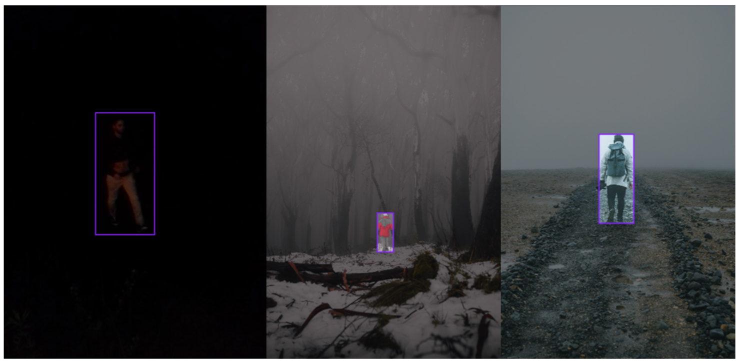
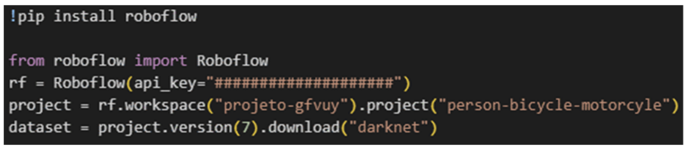


Figure 19. Images affected by atmospheric and lighting conditions.

4.2. Benchmark Scenario

The Google Colab platform [52], which provides computing resources, was used to train the models. To train the three models, a machine with the following characteristics was allocated: an NVIDIA T4 graphics card with 16 GB of VRAM and 13 GB of RAM. With the use of a free Google Colab subscription, this platform is limited in the amount of time it can be used. As the graphics card is the most important piece of hardware in the model training process [53], this subscription only allows for approximately 6 h of use. This number of hours is not exact, nor is it disclosed by Google, as these usage limitations are dynamic and vary according to how the platform is used [54]. Therefore, the model was saved in Google Drive [55] at the end of each training epoch. It was also necessary to use another Google account to split up the YOLOv3-Tiny model training process.

To implement the training of the YOLOv3-Tiny model, an existing notebook on Google Colab created by Roboflow was used, which was adapted from YOLOv4 to YOLOv3-Tiny [56]. On the machine provided, the NVIDIA Cuda Compiler Driver [57] was installed to associate the graphics processing unit (GPU) hardware with the execution. It also installed the Darknet framework, which is an open-source neural network architecture implemented in C and CUDA, recognized for its speed, ease of installation, and support for computing on the central processing unit (CPU) and GPU [58]. Next, the YOLOv3-Tiny base weights [59] were downloaded in YOLO Darknet format. With the help of the Roboflow library for Python [60], it was easy to download and prepare the dataset for the framework, as shown in Figure 20.

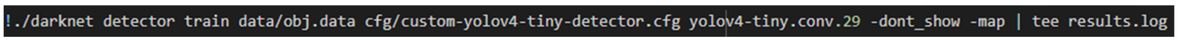


```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="###################")
project = rf.workspace("projeto-gfvuy").project("person-bicycle-motorcyle")
dataset = project.version(7).download("darknet")
```

Figure 20. Use of the Roboflow library.

After this stage, the images and labels were prepared in the correct directories. A training configuration file adjusted to the dataset used was also built. The changes made to this code were the number of epochs (*max_batches*), where the value was changed to 6000. The value recommended by the notebook follows the instructions in the darknet repository, which states that *max_batches* should be defined as the number of classes multiplied by 2000 while ensuring that it is not less than the number of training images and not less than 6000 [61].

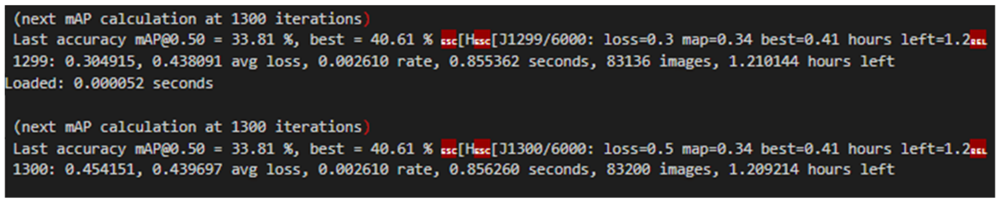
In the next phase, the training was carried out using the command shown in Figure 21, where Darknet took control of the process.



```
!./darknet detector train data/obj.data cfg/custom-yolov4-tiny-detector.cfg yolov4-tiny.conv.29 -dont_show -map | tee results.log
```

Figure 21. Command to perform the training.

Once the training was complete, it was possible to analyze the first 1000 epochs, followed by analysis at each increment of 100 subsequent epochs. These analyses were extracted to a *results.log* file where information was displayed, highlighting the current training epoch number, the average loss in training (loss), average detection accuracy (mAP), the best average accuracy achieved so far (best), time remaining to complete training based on current progress, details of the average loss (avg loss), and the total number of images processed so far (images). Figure 22 illustrates this analysis in the file, at iterations 1299 and 1300 of the training.

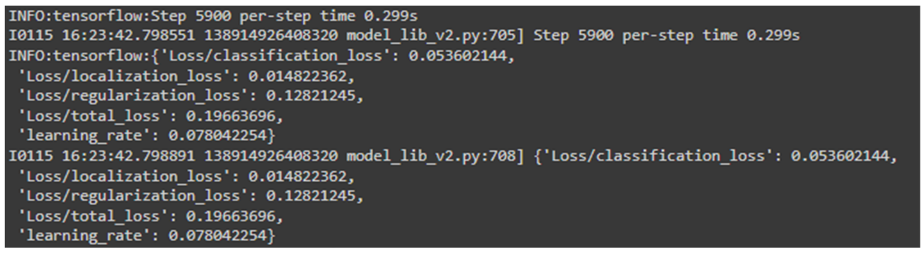


```
 (next mAP calculation at 1300 iterations)
 Last accuracy mAP@0.50 = 33.81 %, best = 40.61 % ESC[HESC[J1299/6000: loss=0.3 map=0.34 best=0.41 hours left=1.2BEL
 1299: 0.304915, 0.438091 avg loss, 0.002610 rate, 0.855362 seconds, 83136 images, 1.210144 hours left
Loaded: 0.000052 seconds

 (next mAP calculation at 1300 iterations)
 Last accuracy mAP@0.50 = 33.81 %, best = 40.61 % ESC[HESC[J1300/6000: loss=0.5 map=0.34 best=0.41 hours left=1.2BEL
 1300: 0.454151, 0.439697 avg loss, 0.002610 rate, 0.856260 seconds, 83200 images, 1.209214 hours left
```

Figure 22. Log file with analysis of iterations 1299 and 1300.

The MobileNet-SSD V2 model was implemented using the TensorFlow framework in a Google Colab notebook [62]. It was later copied and adapted to our case [63]. Initially, the dependencies for TensorFlow object detection were installed, guaranteeing the availability of the libraries needed for the process. Next, to prepare the training data, the dataset was downloaded in Pascal VOC format and TFRecords format from the Roboflow platform. The purpose of this notebook is to convert the images and their XML files, in Pascal VOC format, into TFRecords format, to be used by TensorFlow during training. In addition, tests were carried out to evaluate the performance of the trained model, also using test images extracted from the dataset in Pascal VOC format. The use of the Roboflow platform considerably simplified this task.

In the training configuration, the MobileNet-SSD V2 model to be used was selected from the TensorFlow 2 Object Detection Model Zoo [64]. At this stage, hyperparameters were also specified for training the model, such as the number of epochs (*num_steps*) and *batch_size*, which represent the number of images to be used per training step. In this case, 6000 epochs and 16 images were selected, respectively. Every 100 epochs, the time taken and metrics such as classification_loss, localization_loss, regularization_loss, total_loss, and learning_rate were displayed. Figure 23 provides a detailed view of the outputs, focusing particularly on the 5900th iteration epoch.

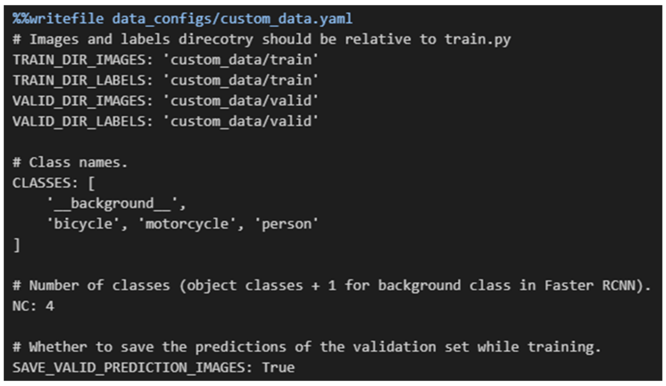


```
INFO:tensorflow:Step 5900 per-step time 0.299s
I0115 16:23:42.798551 138914926408320 model_lib_v2.py:705] Step 5900 per-step time 0.299s
INFO:tensorflow:{'Loss/classification_loss': 0.053602144,
 'Loss/localization_loss': 0.014822362,
 'Loss/regularization_loss': 0.12821245,
 'Loss/total_loss': 0.19663696,
 'learning_rate': 0.078042254}
I0115 16:23:42.798891 138914926408320 model_lib_v2.py:708] {'Loss/classification_loss': 0.053602144,
 'Loss/localization_loss': 0.014822362,
 'Loss/regularization_loss': 0.12821245,
 'Loss/total_loss': 0.19663696,
 'learning_rate': 0.078042254}
```

Figure 23. Runtime output and loss metrics at the iteration epoch 5900.

Subsequently, the model resulting from the training was converted into the TensorFlow Lite format, a version optimized for devices with low computing power. Still in the notebook, the model was tested on 10 test images, before the mAP was calculated to assess the model's effectiveness in terms of average detection accuracy.

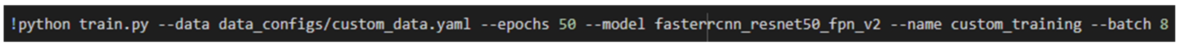
ResNet-50 was implemented in a notebook based on [65]. The framework used to train this model was PyTorch [66]. A clone was made of a repository created in [65]. After this step, the Roboflow library [60] was used to download and format the dataset in Pascal VOC format. Next, the training configuration file was prepared to assign the paths of the training and test images and labels. This file, shown in Figure 24, also defines the classes present in the dataset and their ID.



```
%%writefile data_configs/custom_data.yaml
# Images and labels direcotry should be relative to train.py
TRAIN_DIR_IMAGES: 'custom_data/train'
TRAIN_DIR_LABELS: 'custom_data/train'
VALID_DIR_IMAGES: 'custom_data/valid'
VALID_DIR_LABELS: 'custom_data/valid'

# Class names.
CLASSES: [
    '__background__',
    'bicycle', 'motorcycle', 'person'
]

# Number of classes (object classes + 1 for background class in Faster RCNN).
NC: 4

# Whether to save the predictions of the validation set while training.
SAVE_VALID_PREDICTION_IMAGES: True
```

Figure 24. ResNet-50 training configuration file.

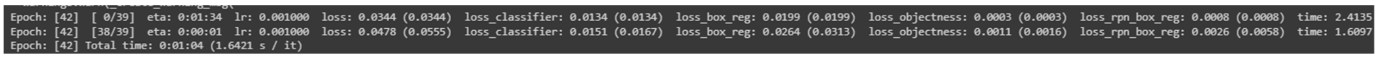
Next, the "train.py" file needs to be executed to start training the model. As can be seen in Figure 25, input parameters need to be set: the location of the training configuration file, the number of epochs to be trained, the model to be used, the name of the output of the trained model, and the batch size.



```
!python train.py --data data_configs/custom_data.yaml --epochs 50 --model fasterrcnn_resnet50_fpn_v2 --name custom_training --batch 8
```

Figure 25. Command to run ResNet-50 training.

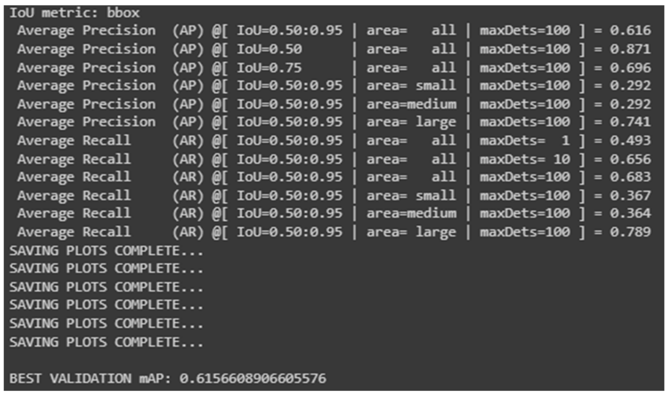
The model selected was "*fasterrcnn_resnet_50_fpn_v2*", the output name was "*custom_training*", and the batch size value was 8. For each training epoch, the metrics loss, loss_classifier, loss_box_reg, loss_objectness, loss_rpn_box_reg, and execution time are displayed, as shown in Figure 26.



```
Epoch: [42]  [ 0/39]  eta: 0:01:34  lr: 0.001000  loss: 0.0344 (0.0344)  loss_classifier: 0.0134 (0.0134)  loss_box_reg: 0.0199 (0.0199)  loss_objectness: 0.0003 (0.0003)  loss_rpn_box_reg: 0.0008 (0.0008)  time: 2.4135
Epoch: [42]  [38/39]  eta: 0:00:01  lr: 0.001000  loss: 0.0478 (0.0555)  loss_classifier: 0.0151 (0.0167)  loss_box_reg: 0.0264 (0.0313)  loss_objectness: 0.0011 (0.0016)  loss_rpn_box_reg: 0.0026 (0.0058)  time: 1.6097
Epoch: [42] Total time: 0:01:04 (1.6421 s / it)
```

Figure 26. Epoch 42 training metrics.

At the end of each epoch, tests were also carried out on the model, shown in Figure 27, resulting in a summary of validation metrics, such as Average Precision and Average Recall. The mAP is also displayed if these results are better than those obtained previously.



```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.616
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.871
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.696
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.292
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.292
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.741
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.493
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.656
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.683
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.367
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.364
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.789
SAVING PLOTS COMPLETE...
SAVING PLOTS COMPLETE...
SAVING PLOTS COMPLETE...
SAVING PLOTS COMPLETE...
SAVING PLOTS COMPLETE...
SAVING PLOTS COMPLETE...

BEST VALIDATION mAP: 0.6156608906605576
```

Figure 27. Epoch 42 validation metrics.

The test environment was then carried out on a device with an Intel-Core I7-11370H CPU, 16 GB of RAM, and an NVIDIA RTX 3050 GPU. All the models were tested on it, allowing an equal comparison between models in terms of accuracy, processing speed, and model efficiency.

4.3. Performance Metrics

In order to assess the effectiveness of models in detecting and classifying objects, it is essential to understand the metrics associated with these methods. AP (average precision) is a commonly used metric in binary classification and information retrieval. It serves to summarize the precision–recall curve and provides a single numerical value that characterizes the quality of retrieval results for a given class or category. This is particularly relevant in tasks like object detection [67]. It is calculated according to Equation (1).

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

One of the most important and widely used metrics is the mAP. It is calculated as the average of the accuracies at different recall levels (evaluation of the model's effectiveness in detection). The mAP is an important metric because it is insensitive to the size of the objects, allowing effective comparison of models. It is calculated using Equation (2), using the average of the APs of all the classes considered.

$$mAP = \frac{1}{k}\sum_{i}^{k} AP_i \tag{2}$$

The F1-Score is a metric commonly employed to evaluate a model's precision. It combines both precision and recall into a unique measure. Consequently, it offers an assessment of a model's performance across varying levels of precision and recall. It is mainly calculated using true positives, false positives, and false negatives, as delineated in Equation (3).

$$F1\ score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3}$$

Intersection over union (IoU) is also used to assess the accuracy of a model. It is calculated as the ratio of the area of intersection of the detection and the area of union of the detection and the reference rectangle. It is also a metric that is independent of the size of the objects, which makes it useful for comparing models of different sizes. The loss metric represents the collective error in the model's predictions, calculated as the difference between the model's output and the desired value. The loss_classifier measures the discrepancy between the model's predictions and the actual classes of the objects in the image, quantifying how far off the model's predictions are from the actual classes. This encourages the model to adjust the parameters to minimize this discrepancy. The loss_box refers to the loss associated with the location or bounding box, assessing the discrepancy between the coordinates of the bounding box predicted by the model and the actual coordinates of the object's bounding box in the image. The loss_rpn_box specifies the loss associated with adjusting the bounding boxes generated by the RPN (Region Proposal Network). The loss_object metric is used to assess the overall accuracy of a model, looking at the accuracy of the location of objects in the image and the correct assignment of categories to objects. It is a weighted combination of these two components, reflecting the overall accuracy of the model in detecting and classifying objects.

Overfitting [68] represents a significant challenge when training CNNs. It is manifested when a model is overtrained, going so far as to memorize specific details of that data. As a result, the model demonstrates outstanding performance on the training data, but fails when dealing with new data. To mitigate this problem, a balance must be struck in accurately capturing meaningful patterns, while avoiding building an overly complex model that adapts too much to the training data. Figure 28 provides an illustration of the concept of overfitting.

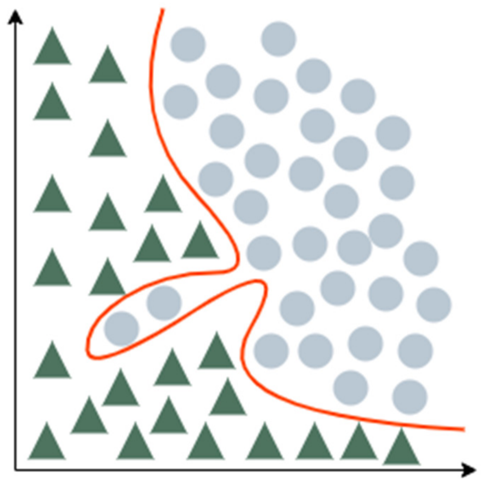


Figure 28. Illustration of the concept of overfitting.

For overfitting prevention Early Stopping [69], Figure 29 illustrates the concept, showing the training error in blue and the validation error in red as a function of the training epochs. If the model continues learning after a certain point, the validation error will increase while the training error will continue to decrease. The aim is to find the right moment to stop training, avoiding both underfitting and overfitting.

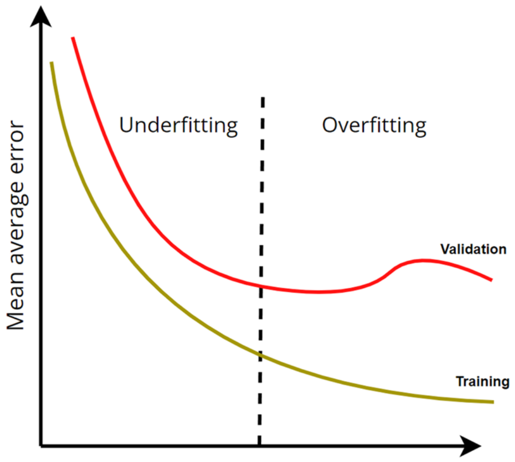


Figure 29. Validation error versus training error.

4.4. Results and Discussion

To evaluate the results obtained with each of the models, initially, each of the models was trained and the respective training results were obtained. Then, various tests were carried out using real videos to assess the ability of each model to detect the various classes (persons, bicycles, and motorcycles).

4.4.1. Train

YOLOv3-Tiny training lasted a total of 2 h and 30 min. For each iteration, a graph was generated, as shown in Figure 30, in which the mAP values are shown in red, while the training loss values are shown in blue. The graph shows the upward trend of the mAP over the epochs, indicating a continuous improvement in detection accuracy. The model ended with a mAP of 68.7%, which is an acceptable performance. In addition, the graph showed the exponential decrease in training loss, initially sharp and stabilizing at values close to 0.1.

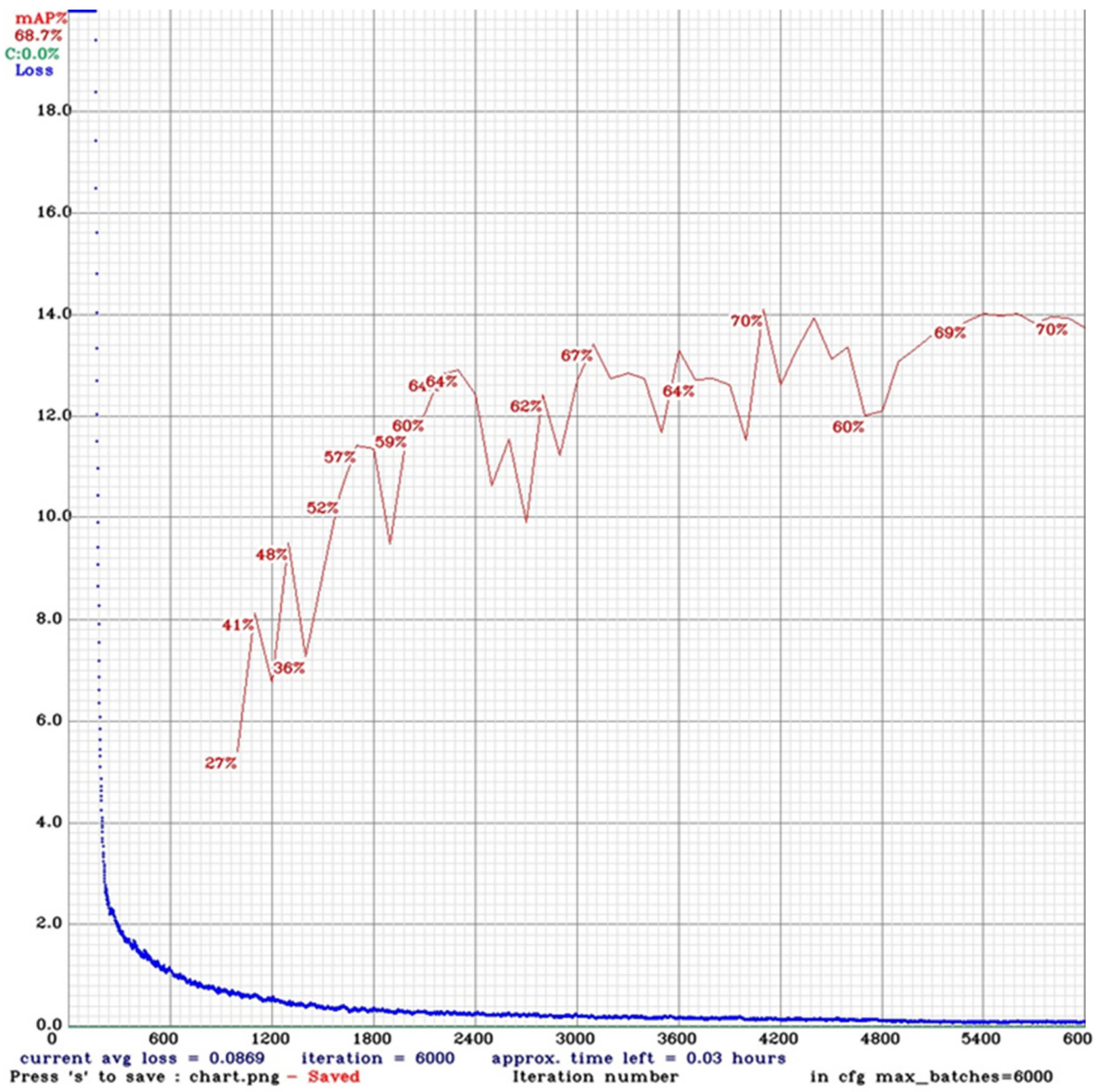


Figure 30. Training loss and mAP in YOLOv3-Tiny.

Figure 31 shows the AP of each class, where bicycles obtained 65.27%, motorcycles 78.71%, and persons 62.02%. In addition to this information, a precision of 71%, a recall of 57%, and an F1-score of 63% were obtained.

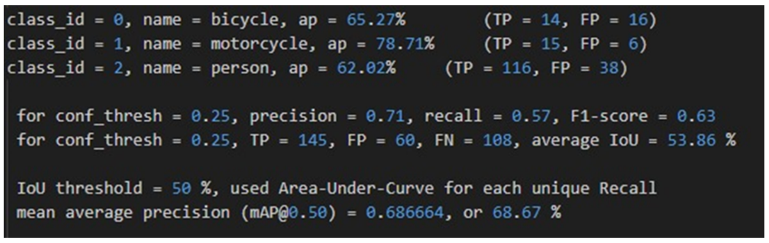
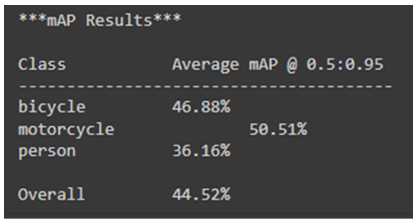

```
class_id = 0, name = bicycle, ap = 65.27%        (TP = 14, FP = 16)
class_id = 1, name = motorcycle, ap = 78.71%     (TP = 15, FP = 6)
class_id = 2, name = person, ap = 62.02%     (TP = 116, FP = 38)

 for conf_thresh = 0.25, precision = 0.71, recall = 0.57, F1-score = 0.63
 for conf_thresh = 0.25, TP = 145, FP = 60, FN = 108, average IoU = 53.86 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.686664, or 68.67 %
```

Figure 31. YOLOv3-Tiny training results.

Training the MobileNet-SSD V2 model was faster than YOLOv3-Tiny, taking approximately 31 min. The mAP calculated for this model shown in Figure 32 was 44.52%, which is considered a relatively low value. This lower value may be a consequence of training speed that may impact accuracy.



```
***mAP Results***

Class           Average mAP @ 0.5:0.95
----------------------------------------
bicycle         46.88%
motorcycle              50.51%
person          36.16%

Overall         44.52%
```

Figure 32. MobileNet-SSD V2 training results.

A TensorBoard session was started to monitor the training progress of the MobileNet-SSD V2 model [70]. Figure 33 shows the results of the loss metrics, where there is a variation in values over the epochs in the classification loss, localization loss, regularization loss, and total loss metrics. Figure 34, on the other hand, shows the learning rate metric, where it can be concluded that, close to 1500 epochs, the model no longer learns.

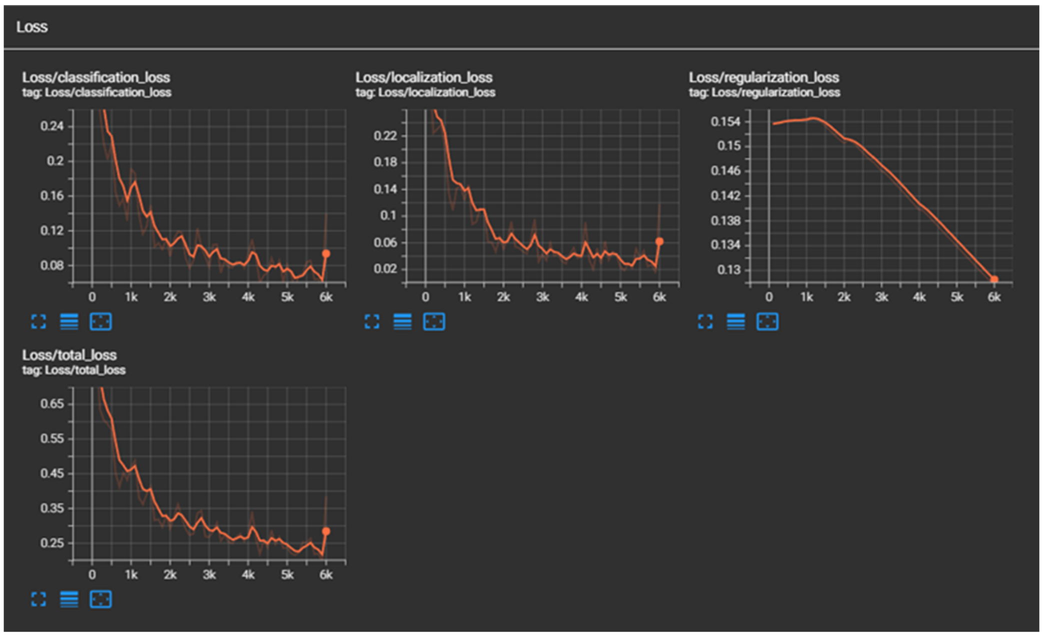


Figure 33. MobileNet-SSD V2 loss metrics.

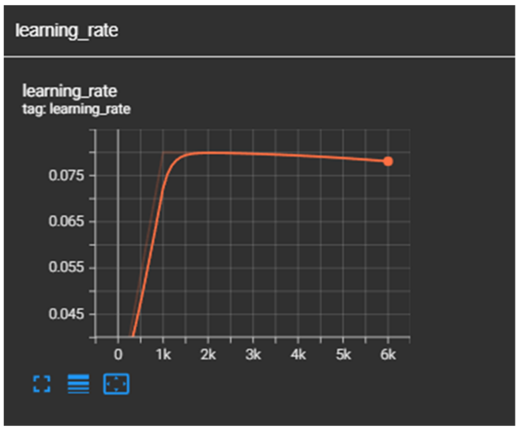


Figure 34. MobileNet-SSD V2 learning rate metric.

The ResNet-50 model took 3 h and 50 min to train, making it the longest. A mAP of 61.5% was obtained in epoch 43. Figure 35 shows the evolution of the mAP over the epoch. The model's evolution in terms of accuracy does not improve significantly from epoch 30 onwards. So, there is no need for exhaustive training of the model.

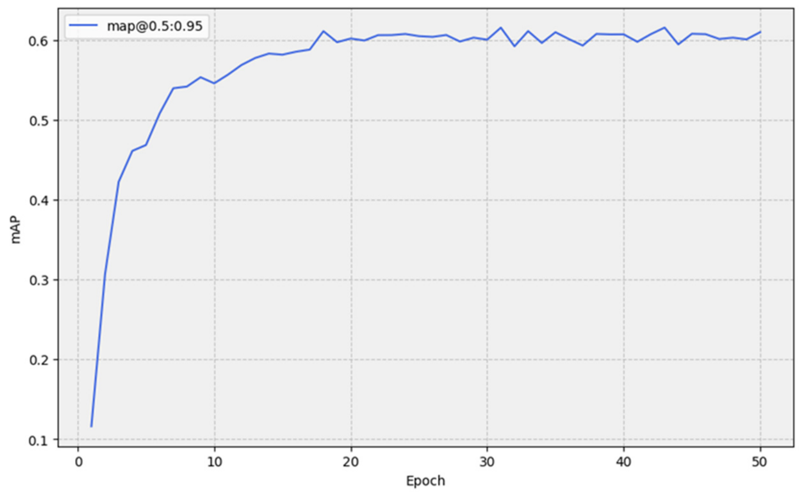


Figure 35. ResNet-50 mAP.

Figure 36 shows the metrics associated with ResNet-50 training, focusing on the various losses. Analysis of this figure reveals an evolution over the seasons, manifested by the constant decrease in all the values represented in it. In the last season considered, the results obtained were as follows: loss 0.0261, loss_classifier 0.0166, loss_box 0.0313, loss_rpn_box 0.0058, and loss_object 0.0016. These values demonstrate a remarkable ability to recognize and identify objects. The consistent reduction in losses over the epochs suggests that the training is evolving towards an optimal solution.

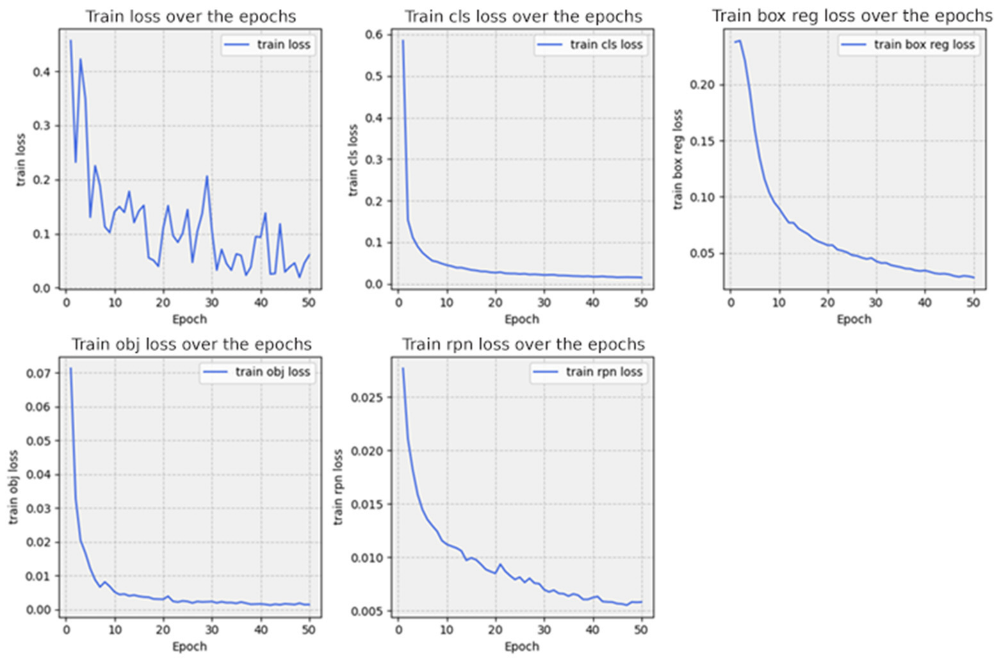
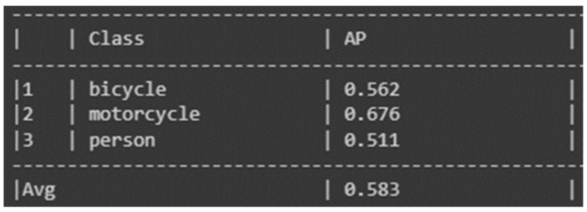


Figure 36. ResNet-50 training metrics.

The ResNet-50 model was validated in order to check its accuracy. The images used for this scenario were the validation images in the dataset described above. Figure 37 shows the results obtained at this stage, with an AP of 56.2% in the bicycle class, an AP of 67.6% in the motorcycle class, and an AP of 51.1% in the person class. These results gave a mAP of 58.3%, suggesting that the model performed reasonably well.



```
-------------------------------------------------------
|    | Class                    | AP                  |
-------------------------------------------------------
|1   | bicycle                  | 0.562               |
|2   | motorcycle               | 0.676               |
|3   | person                   | 0.511               |
-------------------------------------------------------
|Avg                            | 0.583               |
```

Figure 37. ResNet-50 validation results.

4.4.2. Tests

To validate the results obtained, several comparative tests were carried out on the three trained models, using various videos recorded in real context. For these tests, the error/accuracy of the different models was taken into account as well as the consumption of computational resources using the FPS comparison. According to [71], video input is more demanding in terms of computing power than direct input from a webcam or similar device. The confidence parameter was set to 93%, i.e., only bounding boxes appear for the object classes that the model has identified with a confidence equal to or greater than this threshold.

In a future prototype, it is intended to place the camera in strategic locations, such as on a pole about a meter off the ground (a topic that will be studied in the second part of this work). In addition, cameras with a high field of view (FOV) will be used, allowing objects to appear in the scene for longer. The initial option of carrying out tests with the camera positioned on the ground aims to provide an effective preliminary evaluation of the algorithms under controlled conditions, allowing for initial optimizations before transitioning to the final configuration.

For the first test, a video containing a bicycle moving toward the camera was used to see how each of the models identified this element. In Figure 38, we can see that all the models correctly identify the bicycle class. Interestingly, ResNet-50 also classifies a dog in the scenario as a person, which is an error.

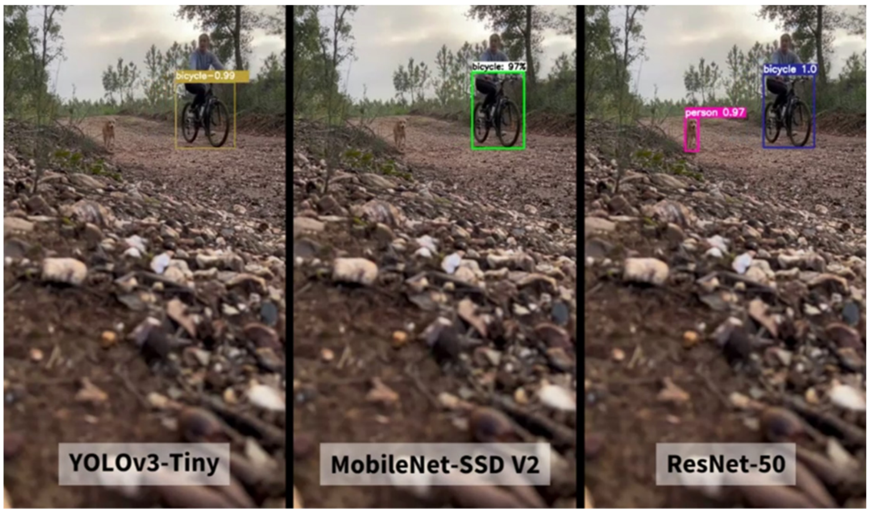


Figure 38. Examples of bicycle detection results for the three models (moving towards the camera).

Figure 39 shows that both YOLOv3-Tiny and ResNet-50 models correctly identify a bicycle even when it moves in the opposite direction to the camera. In contrast, MobileNet-SSD V2 identifies a bicycle as a person. It can also be seen that ResNet-50 wrongly identifies a dog as a person. For this test, it can be concluded that YOLOv3-Tiny performs better than the other models.

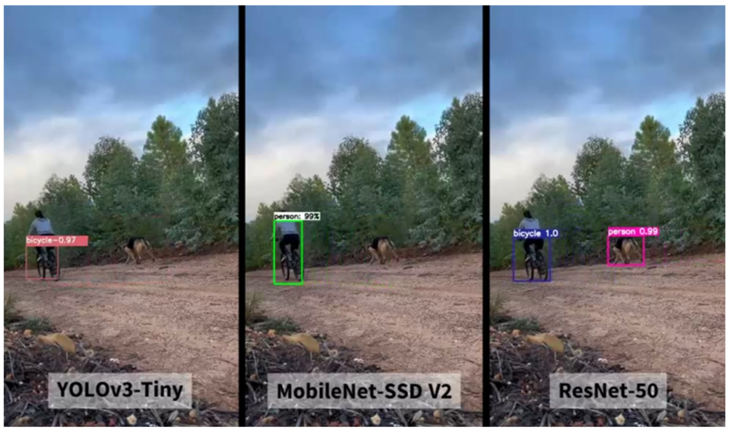


Figure 39. Examples of bicycle detection results for the three models (moving in the opposite direction to the camera).

For the second test, videos containing people were used to see how each of the models identified this element. Figure 40 shows that all the models correctly identified the people present, classifying them as a person. It is worth noting that ResNet-50 again identified wrongly a dog as a person. So, YOLOv3-Tiny and MobileNet-SSD V2 models perform better in this test.

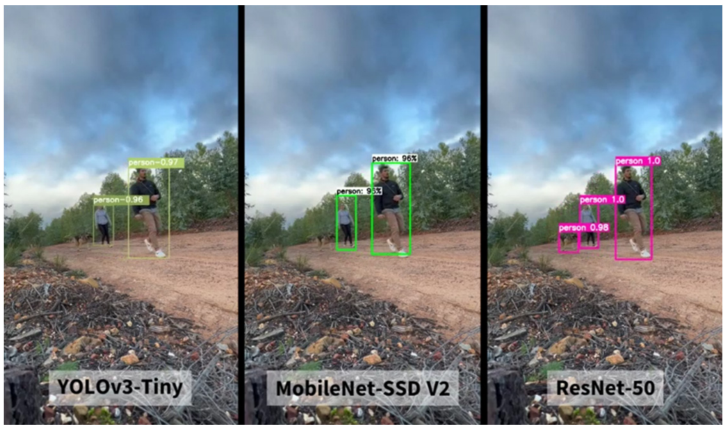


Figure 40. Examples of person detection results for the three models.

For the last test, videos containing motorcycles were used to assess their identification by each of the models. Figure 41 shows motorcycles moving in the opposite direction to the camera. YOLOv3-Tiny and ResNet-50 models correctly detect and classify motorcycles, whereas MobileNet-SSD V2 incorrectly mistakes the motorcycle with a person. Therefore, for this test, it can be concluded that YOLOv3-Tiny and ResNet-50 perform better.

Figure 42 shows a motorcycle moving towards the camera. Both the YOLOv3-Tiny and MobileNet-SSD V2 models fail to classify the motorcycle, wrongly classifying it as a person. ResNet-50 correctly identifies the motorcycle. Therefore, it performs better in this case. Figure 43 shows that all the models were able to correctly classify the motorcycle when it was near the camera.

As far as computing resources are concerned, after the video test was completed, it was observed that the most demanding model was the ResNet-50, reaching only 4.15 FPS. On the other hand, the model that proved to be the least demanding was the MobileNet-SSD V2, reaching 27.85 FPS. Finally, the YOLOv3-Tiny achieved a rate of 16.58 FPS. From these results, it was concluded that ResNet-50 could be too demanding for the scenario considered in the context of this work. Table 3 summarizes the results of the performance metrics of the models under evaluation.

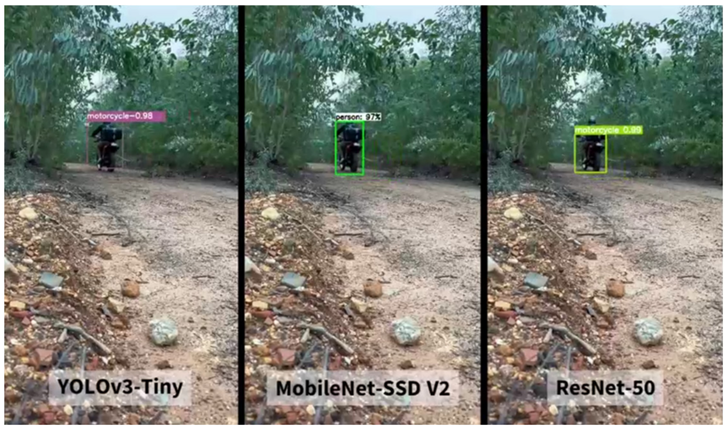


Figure 41. Examples of motorcycle detection results for the three models (moving in the opposite direction to the camera).

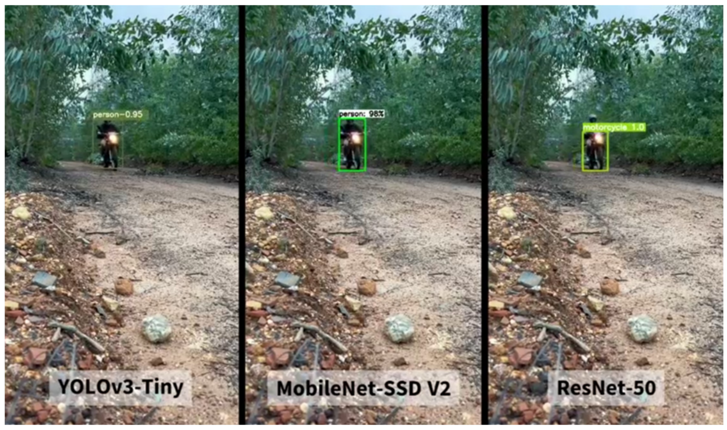


Figure 42. Examples of motorcycle detection results for the three models (moving towards the camera).

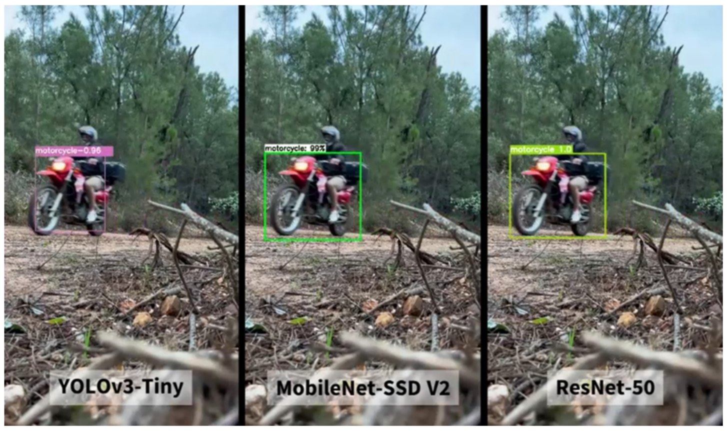


Figure 43. Examples of motorcycle detection results for the three models.

It was concluded that YOLOv3-Tiny achieved a mAP of 68.7%, while MobileNet-SSD V2 only achieved 44.52%. These results corroborate the results obtained in the validation of the models, indicating that YOLOv3-Tiny has superior accuracy. Although MobileNet-SSD V2 offers a higher FPS rate, its lower accuracy suggests a limitation in the reliability of the data obtained. Based on this analysis, YOLOv3-Tiny is considered the most promising model for the solution to be developed in future work.

Table 3. Comparative table of performance metrics of the models tested.

	AP	mAP	FPS
YOLOv3-Tiny	Bicycle: 65.27% Motorcycle: 78.71% Person: 62.02%	68.7%	16.58
MobileNet-SSD V2	Bicycle: 46.88% Motorcycle: 50.51% Person: 36.16%	44.52%	27.85
FasterRCNN and ResNet-50	Bicycle: 56.2% Motorcycle: 67.6% Person: 51.1%	58.3%	4.15

5. Conclusions

This work aims to present a response to the identified challenge of the lack of accurate and reliable data on the actual use of walking, hiking, and cycling trails and routes. The absence of detailed information on the frequency of use and the types of users of these infrastructures hinders the assessment of the tourist impact on the regions, as well as management, maintenance, and access control itself.

Initially, a comprehensive study was carried out of various projects, techniques, and methods applied to the identification and quantification of different types of users. For this study, an analysis of various articles was conducted using the PRISMA methodology. As a result of this analysis, 12 articles were selected that fit the theme of this work. Considering these articles, a critical analysis was carried out of the most prevalent and most promising computer vision techniques. YOLOv3-Tiny, MobileNet-SSD V2, and ResNet-50 were identified for further study. Next, fundamental concepts of computer vision, CNN architectures, as well as one-stage and two-stage detection approaches were covered, followed by an analysis of the specific features of YOLOv3-Tiny, MobileNet-SSD V2, and ResNet-50.

Then, a performance assessment was carried out of these models. A new dataset was developed for this purpose. The benchmark scenario and the training environment for each model were detailed. The performance metrics used to evaluate the results obtained for each of the models were also explained. The training and validation results for each model were then analyzed. In addition, to validate the training results, a set of tests were carried out using videos captured in real context. Due to the hardware limitations of the prototype to be built in future work, the results analysis also considered the computing power required for each model, measuring their performance in terms of frames per second. In conclusion, it should be noted that, although YOLOv3-Tiny did not register the best FPS rate, it did achieve the best mAP. Thus, the choice between real-time performance and accuracy leads to the conclusion that YOLOv3-Tiny is the most promising model to be used in the development of the future prototype.

It should be noted that the work presented in this paper represents the first step in an ongoing effort to develop a prototype to be applied, tested, and demonstrated in real-world scenarios. Therefore, as for future work, it is necessary to consider technological challenges, as well as opportunities to improve it. In this way, challenges are faced in remote environments, subject to adverse conditions such as rain, fog, extreme temperatures, and the risk of malicious activities such as theft. To guarantee the system's operability in the face of these challenges, it is essential to implement suitable protections. In addition, the issue of energy is crucial, considering that setting up a network infrastructure for the site can be unfeasible. In this context, the implementation of alternative energy sources, such as solar panels [72], is a viable solution. Mobile network integration is also indispensable for the efficient transmission of collected data to a centralized database. Although it is possible to schedule the periodic transmission of data, optimizing resource consumption, the absence of mobile network infrastructure can be circumvented by involving route users. An effective strategy in this regard involves the use of a dedicated mobile application installed on users' devices, turning them into "bridges" for data transfer. In addition to these challenges, it is necessary to substantially enrich the dataset, and carry out additional tests to optimize the mAP in the models evaluated.

Author Contributions: Conceptualization, J.M., P.M. and A.Q.; methodology, J.M. and P.M.; validation, J.M.L.P.C. and V.N.G.J.S.; formal analysis, J.M.L.P.C. and V.N.G.J.S.; investigation, J.M. and P.M.; writing—original draft preparation, J.M. and P.M.; writing—review and editing, A.Q., J.M.L.P.C. and V.N.G.J.S.; supervision, J.M.L.P.C. and V.N.G.J.S.; funding acquisition, J.M.L.P.C. and V.N.G.J.S. All authors have read and agreed to the published version of the manuscript.

Funding: J.M.L.P.C. and V.N.G.J.S. acknowledge that this work is funded by FCT/MCTES through national funds and, when applicable, co-funded EU funds under the project UIDB/50008/2020.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Federação de Campismo e Montanhismo de Portugal Regulamento de Homologação De Percursos. Lisboa, Portugal, 2006. Available online: https://cm-nisa.pt/images/documentos/areas_atividade/desporto/regulamentopercursospedestres.pdf (accessed on 9 December 2023).
2. Sinalização. Available online: http://www.solasrotas.org/2008/09/sinalizao.html (accessed on 9 December 2023).
3. Carvalho, P. *Pedestrianismo e Percursos Pedestres*; Cadernos de Geograia: Coimbra, Portugal, 2009.
4. Federação de Campismo e Montanhismo de Portugal Site Oficial Da FCMP. Available online: https://www.fcmportugal.com/ (accessed on 16 January 2024).
5. Federação de Campismo e Montanhismo de Portugal Site Oficial Da FCMP—Percursos Pedestres. Available online: https://www.fcmportugal.com/percursos-pedestres/ (accessed on 9 December 2023).
6. Zhao, W.; Li, J. A Survey of Object Detection Methods in Inclement Weather Conditions. In Proceedings of the 2023 IEEE International Conference on Unmanned Systems (ICUS), Hefei, China, 13–15 October 2023; IEEE: New York, NY, USA, 2023; pp. 1405–1412.
7. Vavilin, A.; Lomov, A.; Roman, T. Real-Time Train Wagon Counting and Number Recognition Algorithm. In Proceedings of the 2022 International Workshop on Intelligent Systems (IWIS), Ulsan, Republic of Korea, 17–19 August 2022; pp. 1–4.
8. Gideon Why Vision Is Better than LiDAR. Available online: https://www.gideon.ai/resources/why-is-vision-better-than-lidar-for-logistics-robots/ (accessed on 27 February 2024).
9. Cardoso, O. Visão Computacional: Desafios e Avanços Recentes Na Área | Vigeversa. Available online: https://vigeversa.com/inteligencia-artificial/visao-computacional/ (accessed on 27 February 2024).
10. Roboflow Inc. Roboflow Website. Available online: https://roboflow.com/ (accessed on 22 November 2023).
11. Prisma PRISMA Statement. Available online: http://www.prisma-statement.org/ (accessed on 6 November 2023).
12. FCCN Biblioteca Do Conhecimento Online (b-On). Available online: https://www.b-on.pt/ (accessed on 7 November 2023).
13. ISCTE, B. *Guia de Apoio Ao Utilizador (b-On)*; Lisboa, Portugal, 2013; Volume 3. Available online: https://www.iscte-iul.pt/assets/files/2017/01/30/1485777979520_Guia_b_on_MOD_SID_AU_003_4.pdf (accessed on 7 November 2023).
14. Minh, K.T.; Dinh, Q.-V.; Nguyen, T.-D.; Nhut, T.N. Vehicle Counting on Vietnamese Street. In Proceedings of the 2023 IEEE Statistical Signal Processing Workshop (SSP), Hanoi, Vietnam, 2–5 July 2023; pp. 160–164.
15. Hong, C.J.; Mazlan, M.H. Development of Automated People Counting System Using Object Detection and Tracking. *Int. J. Online Biomed. Eng.* **2023**, *19*, 18–30. [CrossRef]
16. Chatrasi, A.L.V.S.S.; Batchu, A.G.; Kommareddy, L.S.; Garikipati, J. Pedestrian and Object Detection Using Image Processing by YOLOv3 and YOLOv2. In Proceedings of the 7th International Conference on Trends in Electronics and Informatics, ICOEI 2023—Proceedings, Tirunelveli, India, 11–13 April 2023; pp. 1667–1672.
17. Anil, J.M.; Mathews, L.; Renji, R.; Jose, R.M.; Thomas, S. Vehicle Counting Based on Convolution Neural Network. In Proceedings of the 7th International Conference on Intelligent Computing and Control Systems, ICICCS, Madurai, India, 17–19 May 2023; pp. 695–699.
18. Myint, E.P.; Sein, M.M. People Detecting and Counting System. In *LifeTech 2021, Proceedings of the 2021 IEEE 3rd Global Conference on Life Sciences and Technologies, Nara, Japan, 9–11 March 2021*; IEEE: Nara, Japan, 2021; pp. 289–290.
19. Kolluri, J.; Das, R. Intelligent Multimodal Pedestrian Detection Using Hybrid Metaheuristic Optimization with Deep Learning Model. *Image Vis. Comput.* **2023**, *131*, 104268. [CrossRef]
20. Mimboro, P.; Heryadi, Y.; Lukas; Suparta, W.; Wibowo, A. Realtime Vehicle Counting Method Using Haar Cascade Classifier Model. In Proceedings of the 2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation, ICAMIMIA, Surabaya, Indonesia, 8–9 December 2021; IEEE: Surabaya, Indonesia, 2021; pp. 229–233.

21. Vignarca, D.; Prakash, J.; Vignati, M.; Sabbioni, E. Improved Person Counting Performance Using Kalman Filter Based on Image Detection and Tracking. In Proceedings of the 2021 AEIT International Conference on Electrical and Electronic Technologies for Automotive, AEIT AUTOMOTIVE, Torino, Italy, 17–19 November 2021; IEEE: Torino, Italy, 2021; pp. 1–6.
22. Minh, H.T.; Mai, L.; Minh, T.V. Performance Evaluation of Deep Learning Models on Embedded Platform for Edge AI-Based Real Time Traffic Tracking and Detecting Applications. In Proceedings of the 2021 15th International Conference on Advanced Computing and Applications, ACOMP, Ho Chi Minh City, Vietnam, 24–26 November 2021; pp. 128–135.
23. Kim, J.; Suh, Y.; Lee, J.; Chae, H.; Ahn, H.; Chung, Y.; Park, D. EmbeddedPigCount: Pig Counting with Video Object Detection and Tracking on an Embedded Board. *Sensors* **2022**, *22*, 2689. [CrossRef] [PubMed]
24. Gomes, H.; Redinha, N.; Lavado, N.; Mendes, M. Counting People and Bicycles in Real Time Using YOLO on Jetson Nano. *Energies* **2022**, *15*, 8816. [CrossRef]
25. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
26. Morera, Á.; Sánchez, Á.; Moreno, A.B.; Sappa, Á.D.; Vélez, J.F. SSD vs. YOLO for Detection of Outdoor Urban Advertising Panels under Multiple Variabilities. *Sensors* **2020**, *20*, 4587. [CrossRef] [PubMed]
27. Amazon Web Services O Que é Visão Computacional?—Explicação de IA/ML de Reconhecimento de Imagem—AWS. Available online: https://aws.amazon.com/pt/what-is/computer-vision/ (accessed on 30 October 2023).
28. IBM O Que São Redes Neurais? IBM. Available online: https://www.ibm.com/br-pt/topics/neural-networks (accessed on 17 October 2023).
29. Nunes dos Santos, V. Reconhecimento de Objetos Em Uma Cena Utilizando Redes Neurais Convolucionais. Bachelor's Thesis, Universidade Tecnológica Federal do Paraná, Curitiba, Brazil, 2018.
30. Bhatt, D.; Patel, C.; Talsania, H.; Patel, J.; Vaghela, R.; Pandya, S.; Modi, K.; Ghayvat, H. CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics* **2021**, *10*, 2470. [CrossRef]
31. Deep Learning Book Capítulo 43—Camadas de Pooling Em Redes Neurais Convolucionais. Available online: https://www.deeplearningbook.com.br/camadas-de-pooling-em-redes-neurais-convolucionais/ (accessed on 5 November 2023).
32. Barbosa, G.; Bezerra, G.M.; de Medeiros, D.S.; Andreoni Lopez, M.; Mattos, D. Segurança Em Redes 5G: Oportunidades e Desafios Em Detecção de Anomalias e Predição de Tráfego Baseadas Em Aprendizado de Máquina. In Proceedings of the Minicursos do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, Online, Belém, 4–7 October 2021; pp. 164–165. [CrossRef]
33. Poloni, K. Redes Neurais Convolucionais. Available online: https://medium.com/itau-data/redes-neurais-convolucionais-2206a089c715 (accessed on 1 November 2023).
34. Archana, V.; Kalaiselvi, S.; Thamaraiselvi, D.; Gomathi, V.; Sowmiya, R. A Novel Object Detection Framework Using Convolutional Neural Networks (CNN) and RetinaNet. In Proceedings of the International Conference on Automation, Computing and Renewable Systems, ICACRS, Pudukkottai, India, 13–15 December 2022; pp. 1070–1074.
35. Carranza-García, M.; Torres-Mateo, J.; Lara-Benítez, P.; García-Gutiérrez, J. On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. *Remote Sens.* **2020**, *13*, 89. [CrossRef]
36. Zhou, L.; Lin, T.; Knoll, A. Fast and Accurate Object Detection on Asymmetrical Receptive Field. *Comput. Vis. Pattern Recognit. Arxiv* **2023**, arXiv:2303.08995. [CrossRef]
37. Thakur, N. A Detailed Introduction to Two Stage Object Detectors. Available online: https://namrata-thakur893.medium.com/a-detailed-introduction-to-two-stage-object-detectors-d4ba0c06b14e (accessed on 23 December 2023).
38. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *Comput. Vis. Pattern Recognit.* **2018**, arXiv:1804.02767. [CrossRef]
39. Bajaj, V. The YOLO Algorithm—Deep Learning Specialization—Coursera. Available online: https://vikram-bajaj.gitbook.io/deep-learning-specialization-coursera/convolutional-neural-networks/object-detection/the-yolo-algorithm (accessed on 28 January 2024).
40. Adarsh, P.; Rathi, P.; Kumar, M. YOLO V3-Tiny: Object Detection and Recognition Using Stage Improved Model. In Proceedings of the 2020 6th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 6–7 March 2020; pp. 687–694.
41. Li, T.; Ma, Y.; Endoh, T. A Systematic Study of Tiny YOLO3 Inference: Toward Compact Brainware Processor With Less Memory and Logic Gate. *IEEE Access* **2020**, *8*, 142931–142955. [CrossRef]
42. Cochard, D. MobilenetSSD: A Machine Learning Model for Fast Object Detection. Available online: https://medium.com/axinc-ai/mobilenetssd-a-machine-learning-model-for-fast-object-detection-37352ce6da7d (accessed on 16 January 2024).
43. Sabina, N.; Aneesa, M.P.; Haseena, P.V. Object Detection Using YOLO And Mobilenet SSD: A Comparative Study. *Int. J. Eng. Res. Technol.* **2022**, *11*, 136–138.
44. Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. *Comput. Vis. Pattern Recognit.* **2019**, arXiv:1905.02244. [CrossRef]
45. Sovit Rath, R. Object Detection Using PyTorch Faster RCNN ResNet50 FPN V2. Available online: https://debuggercafe.com/object-detection-using-pytorch-faster-rcnn-resnet50-fpn-v2/ (accessed on 26 February 2024).
46. ResNet-50: The Basics and a Quick Tutorial. Available online: https://datagen.tech/guides/computer-vision/resnet-50/ (accessed on 26 February 2024).
47. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]

48. Pixabay Pixabay Website. Available online: https://pixabay.com/ (accessed on 29 November 2023).
49. Unsplash Unsplash Website. Available online: https://unsplash.com/pt-br (accessed on 29 November 2023).
50. Miguel, J.; Mendonça, P. Person, Bicycle and Motorcyle Dataset. Available online: https://universe.roboflow.com/projeto-gfvuy/person-bicycle-motorcyle/model/7 (accessed on 26 January 2024).
51. Nelson, J. How to Label Image Data for Computer Vision Models. Available online: https://blog.roboflow.com/tips-for-how-to-label-images/ (accessed on 15 January 2024).
52. Google Colab. Available online: https://colab.google/ (accessed on 22 December 2023).
53. Weka Why GPUs for Machine Learning? A Complete Explanation—WEKA. Available online: https://www.weka.io/learn/ai-ml/gpus-for-machine-learning/ (accessed on 23 December 2023).
54. Google Colab—FAQ. Available online: https://research.google.com/colaboratory/faq.html#resource-limits (accessed on 15 January 2024).
55. Armazenamento Na Nuvem Pessoal e Plataforma de Partilha de Ficheiros—Google. Available online: https://www.google.com/intl/pt-PT/drive/ (accessed on 23 December 2023).
56. Roboflow Notebook—Train Yolov4 Tiny Object Detection On Custom Data. Available online: https://github.com/roboflow/notebooks/blob/main/notebooks/train-yolov4-tiny-object-detection-on-custom-data.ipynb (accessed on 23 December 2023).
57. NVIDIA. NVIDIA CUDA Compiler Driver. Available online: https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html (accessed on 23 December 2023).
58. Darknet Darknet: Open Source Neural Networks in C. Available online: https://pjreddie.com/darknet/ (accessed on 23 December 2023).
59. yaming116 Darknet—Yolov3-Tiny Weights. Available online: https://github.com/smarthomefans/darknet-test/blob/master/yolov3-tiny.weights (accessed on 23 December 2023).
60. Traore, M. Roboflow's Python Pip Package For Computer Vision. Available online: https://blog.roboflow.com/pip-install-roboflow/ (accessed on 23 December 2023).
61. Charette, S. Programming Comments—Darknet FAQ. Available online: https://www.ccoderun.ca/programming/darknet_faq/ (accessed on 28 January 2024).
62. Juras, E.; Technology Consultants, E. Notebook—Train TFLite2 Object Detection Model. Available online: https://colab.research.google.com/github/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Train_TFLite2_Object_Detction_Model.ipynb (accessed on 15 January 2024).
63. Juras, E.; Technology Consultants, E.; Miguel, J.; Mendonça, P. Notebook Adaptado—Train TFLite2 Object Detection Model. Available online: https://colab.research.google.com/drive/1rK3GNbJA_i_rupahuWyWgvrEHHvvp44i?authuser=1#scrollTo=fF8ysCfYKgTP (accessed on 15 January 2024).
64. TensorFlow; vighneshbirodkar; TF Object Detection Team TensorFlow 2 Detection Model Zoo. Available online: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (accessed on 15 January 2024).
65. sovit-123 FasterRCNN Pytorch Training Pipeline: PyTorch Faster R-CNN Object Detection on Custom Dataset. Available online: https://github.com/sovit-123/fasterrcnn-pytorch-training-pipeline (accessed on 15 January 2024).
66. PyTorch PyTorch. Available online: https://pytorch.org/ (accessed on 24 December 2023).
67. Lakera Average Precision. Available online: https://www.lakera.ai/ml-glossary/average-precision (accessed on 17 January 2024).
68. Cook, J.A.; Ranstam, J. Overfitting. *Br. J. Surg.* **2016**, *103*, 1814. [CrossRef] [PubMed]
69. Ying, X. An Overview of Overfitting and Its Solutions. *J. Phys. Conf. Ser.* **2019**, *1168*, 022022. [CrossRef]
70. TensorFlow TensorBoard. Available online: https://www.tensorflow.org/tensorboard?hl=pt-br (accessed on 16 January 2024).
71. Rosebrock, A. YOLO and Tiny-YOLO Object Detection on the Raspberry Pi and Movidius NCS—PyImageSearch. Available online: https://pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/ (accessed on 17 January 2024).
72. Serra, R. Como Funcionam Os Painéis Solares Para Casa? Available online: https://www.doutorfinancas.pt/energia/como-funcionam-os-paineis-solares-para-casa/ (accessed on 29 November 2023).