forests

Article

Forest Vegetation Detection Using Deep Learning Object Detection Models

Paulo A. S. Mendes [1,*], António Paulo Coimbra [1,2] and Aníbal T. de Almeida [1]

1 Institute of Systems and Robotics, University of Coimbra, 3030-290 Coimbra, Portugal; acoimbra@isr.uc.pt (A.P.C.); adealmeida@isr.uc.pt (A.T.d.A.)
2 Department of Electrical and Computer Engineering, University of Coimbra, 3030-290 Coimbra, Portugal
* Correspondence: paulo.mendes@isr.uc.pt

Abstract: Forest fires have become increasingly prevalent and devastating in many regions worldwide, posing significant threats to biodiversity, ecosystems, human settlements, and the economy. The United States (USA) and Portugal are two countries that have experienced recurrent forest fires, raising concerns about the role of forest fuel and vegetation accumulation as contributing factors. One preventive measure which can be adopted to minimize the impact of the forest fires is to cut the amount of forest fuel available to burn, using autonomous Unmanned Ground Vehicles (UGV) that make use of Artificial intelligence (AI) to detect and classify the forest vegetation to keep and the forest fire fuel to be cut. In this paper, an innovative study of forest vegetation detection and classification using ground vehicles' RGB images is presented to support autonomous forest cleaning operations to prevent fires, using an Unmanned Ground Vehicle (UGV). The presented work compares two recent high-performance Deep Learning methodologies, YOLOv5 and YOLOR, to detect and classify forest vegetation in five classes: grass, live vegetation, cut vegetation, dead vegetation, and tree trunks. For the training of the two models, we used a dataset acquired in a nearby forest. A key challenge for autonomous forest vegetation cleaning is the reliable discrimination of obstacles (e.g., tree trunks or stones) that must be avoided, and objects that need to be identified (e.g., dead/dry vegetation) to enable the intended action of the robot. With the obtained results, it is concluded that YOLOv5 presents an overall better performance. Namely, the object detection architecture is faster to train, faster in inference speed (achieved in real time), has a small trained weight file, and attains higher precision, therefore making it highly suitable for forest vegetation detection.

Keywords: object detection; deep learning; computer vision; autonomous vehicle; forest cleaning

check for updates

Citation: Mendes, P.A.S.; Coimbra, A.P.; de Almeida, A.T. Forest Vegetation Detection Using Deep Learning Object Detection Models. *Forests* **2023**, *14*, 1787. https://doi.org/10.3390/f14091787

Academic Editors: David Portugal, João Filipe Ferreira and Francisco J. Yandún

Received: 7 June 2023
Revised: 24 August 2023
Accepted: 30 August 2023
Published: 1 September 2023

1. Introduction

Forest fires continue to pose significant threats to the environment, economy, and human lives in various regions across the globe. The USA and Portugal have both experienced recurrent and devastating forest fires, often exacerbated by the accumulation of forest fuel—dead vegetation that acts as fuel for wildfires. The accumulation of forest fuel is a critical factor contributing to the frequency and intensity of forest fires. Traditional forest management approaches for fuel reduction and wildfire prevention have proven to be insufficient, highlighting the need for innovative and efficient solutions. Autonomous robotics with Artificial Intelligence (AI) have demonstrated remarkable potential in various domains, and its application in forest fuel management could provide a valuable contribution to effectively reduce wildfire risks. An autonomous UGV for forest vegetation cleaning needs to correctly distinguish between the vegetation to remain, the vegetation already cut and the vegetation to be cleaned.

To recognize objects, where they are and how they interact, humans only need to look at a scene. It is a promising objective if an autonomous system could do the same instantaneously with images. The human visual system is fast and accurate, allowing us to perform complex tasks such as the recognition and localization of objects of interest instantaneously. Presently, autonomous vehicles perform complex tasks such as lane [1], pedestrians [2], and road signs (e.g., [3,4]) detection to perform safe autonomous driving. In the agriculture or forestry fields of study, a specialized robot, with this kind of computer vision detection, classification, and localization, could autonomously achieve the harvest of corn [5], weed removal [6], or forest vegetation cleansing.

Object detection is a computer vision technique that allows a computer to detect, classify, and localize objects in images. In recent years, object detection has been greatly improved with the use and development of Deep Learning (DL) techniques. Recent studies indicate that DL provides high accuracy, outperforming existing commonly used image processing techniques [7].

The main goal of this work is to solve the problem of detecting, classifying, and localizing vegetation to be cleaned by an autonomous Unmanned Ground Vehicle (UGV) in forests, to prevent wildland and wildland-urban interface fires (see in Figure 1). To achieve better forest fires fuel cleansing results and help autonomous navigation, a DL model is used to detect, classify, and localize forest vegetation in images, in real time. In this study, we compare two of the most modern DL object detection models to determine the best one for an autonomous UGV in forest environments. The goal is to use this chosen model with a cleansing tool attached to the UGV to prevent forest fires while clearing vegetation. YOLOv5 and YOLOR were chosen because they achieved the best average precision in the MS COCO dataset [8] as object detectors when they were presented to the computer vision community. For the training of the two models, a dataset acquired in a nearby forest was used.

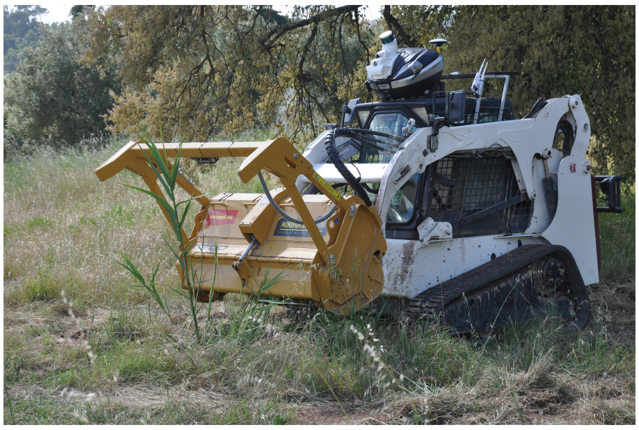

Figure 1. SafeForest Unmanned Ground Vehicle (UGV), showing the sensor system (including LiDAR, RGB, and depth cameras on the top of the vehicle) and the vegetation cleaning attachment at the front.

Zare et al. [9] used vegetation detection for mine detection to minimize false alarms because of the misclassification of vegetation and mines by mine detection algorithms. Some conventional vegetation detection methods are based on normalized difference vegetation index (NDVI) [10–12], which is based on red spectral and near-infrared spectral regions [13]. Zhang et al. [14] provided a comprehensive review of land cover classification and object detection approaches using high-resolution imagery, comparing DL models against traditional approaches. Bulent et al. [15] used DeepLabv3+ to detect the type of vegetation to develop a more accurate digital terrain model. In Torres et al. [16] work it was concluded that DeepLabV3+ performance is worse than Fully Convolutional Network (FCN) in semantically segmenting a single tree species.

The presented work consists of an innovative study in forest vegetation detection, using ground vehicles RGB images, using two of the best computer vision DL object detectors. To the best of the authors' knowledge, this type of study is still not available in the literature and the above-referred publications do not provide a proper solution to our problem.

The main contribution of this work is the detection, classification, and distinction of forest vegetation to be cut and to be preserved by a UGV performing fuel cleaning in forests complex environments.

This paper is organized as follows: a brief state-of-the-art (SOTA) in object detection is presented in Section 2, in Section 3 the YOLO DL network is presented, in Section 4 the YOLOR network is presented, in Section 5 the methodology and results are presented, which are discussed in Section 6, followed by the conclusions.

2. Object Detection State of the Art

2.1. Pioneer Work

Most of the early object detection algorithms were built based on handcrafted features. P. Viola and M. Jones in 2001 achieved real-time detection of human faces, for the first time, without any constraints [17,18]. The Viola and Jones detectors go through all possible scales and locations in an image to verify if any window contains a human face, using sliding windows. The Viola and Jones detector highly improved its detection speed by incorporating three important techniques: detection cascades, Haar-like features, and integral image. Viola Jones algorithm is still used in low-specification devices as it is very fast and efficient in object detection.

N. Dalal and B. Triggs presented the Histogram of Oriented Gradients (HOG) feature descriptor in 2005 [19]. HOG is considered to be an important improvement of the shape context [20] and scale-invariant feature transform [21] of its time. Dalal and Triggs studied the question of feature sets for robust visual object recognition using a linear Support Vector Machine (SVM) and showed experimentally that HOG significantly outperformed existing feature sets for human detection [19]. To balance the feature invariance and nonlinearity such as illumination, translation, or scale, the HOG descriptor is designed to be computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improving accuracy. HOG can be used to detect various object classes but it was motivated, primarily, by the problem of pedestrian detection.

Felzenszwalb et al. presented a system named Deformable Parts Models (DPM) [22] in 2008. DPM uses a sliding window where the classifier is run at evenly spaced localizations over the entire image. DPM achieved higher accuracy than HOG. The DPM detector is defined by a root filter and various part filters. Instead of specifying the configuration of the part-filters (e.g., size and localization) a weakly supervised learning method is used to learn automatically the configuration of the part-filters, as latent variables. Girshick et al. concluded this process as a special case of multi-instance learning [23]. To improve detection accuracy, Girshick et al. applied other techniques named “bounding box regression”, “hard negative mining” and “context priming”. To improve detection speed, Girshick et al., developed a cascade architecture where detection models were compiled into a faster one that achieved over 10 times acceleration without sacrificing any accuracy [24,25].

2.2. Two-Stage Detectors

Object detection could be defined by two genres that are “two-stage detectors” and “one-stage detectors”. A network which has a separate module to generate region proposals is known as a “two-stage detector”. These networks find an arbitrary number of object proposals in an image during the first stage and secondly classify and localize them. These systems have two separate steps and they usually take more time to generate proposals.

Girshick et al. presented a Region-based Convolutional Neural Network (R-CNN) in 2014. R-CNN uses regional proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification using an SVM, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and re-score the boxes based on other objects in the scene [26]. These complex modules are hard to optimize because each single component must be trained separately resulting in slow processing.

Krizhevsky et al. presented deep ConvNets [27], in 2012, which improved object detection and image classification accuracy. Object detection is a challenging task that requires complex architectures, because of this complexity some approaches from the past (e.g., [28–31]) train models in multi-stage pipelines that haves a slow processing.

In 2014, K. He et al. presented Spatial Pyramid Pooling Networks (SPP-Net) [29]. SPP-Net is more than 20 times faster than R-CNN without sacrificing detection accuracy. K. He et al. presented work that eliminated the requirement in Deep Convolutional Neural Networks (DCNNs) to have a fixed-size input image of 224 $\times$ 224 pixels, mentioning that this requirement could reduce the recognition accuracy [29]. The main contribution of SPP-Net is the introduction of a pooling strategy, the Spatial Pyramid Pooling (SPP) layer, that enables a Convolutional Neural Network (CNN) to generate a fixed-length representation regardless of the size of the image/region of interest without re-scaling it. Pyramid pooling is also robust to object deformations. When using SPP-Net for object detection, the feature maps are calculated from the entire image only once. Then, fixed-length representations of arbitrary regions are generated to train the detectors, which avoids repeatedly calculating the convolutional features.

In 2015, Girshick et al. presented the Fast Region-based Convolutional Neural Network (Fast R-CNN) [32]. The biggest problem with R-CNN and SPP-Net is the need to train multiple systems separately. By contrast, Fast R-CNN enables simultaneously training a bounding box regressor and a detector. Fast R-CNN was developed with several innovations to improve testing speed, training, and detection accuracy. "Fast R-CNN trains the very deep VGG16 network 9$\times$ faster that R-CNN, is 213$\times$ faster at test-time, and achieves a higher mAP on PASCAL VOC 2012" [32]. Fast R-CNN without the region proposal network attains near real-time inference with considerable accuracy. Admitting the presented method integrates the improvements of SPP-Net and R-CNN, the detection speed is limited by that region proposal network.

In 2015, Ren et al. presented the Faster R-CNN [33]. Faster R-CNN is the first end-to-end, and the first near real-time DL detector achieving 5 frames per second on a GPU (Graphics Processing Unit). Ren et al. introduced a Region Proposal Network (RPN) that is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position of the input image. RPN enables nearly cost-free region proposals for the input image. It could be said that Faster R-CNN is the Fast R-CNN architecture with RPN integrated as a region proposal module. Faster R-CNN RPN improves region proposal quality resulting in higher object detection accuracy.

In 2016, Dai et al. presented R-FCN (Region-based Fully Convolutional Network) [34] for accurate and efficient object detection. "In contrast to previous region-based detectors such as Fast/Faster R-CNN that apply a costly per-region subnetwork hundreds of times, our region-based detector is fully convolutional with almost all computation shared on the entire image" [34]. Dai et al. proposed an innovative method to solve the problem of translation-variance in object detection and translation-invariance in image classification. R-FCN achieved a test-time speed of 170 ms per image, about 2.5$\times$ to 20$\times$ faster than Faster R-CNN [34]. To achieve a faster detector with higher accuracy, R-FCN combines Faster R-CNN and the Fully Convolutional Network (FCN) [35].

In 2017, Lin et al. presented Feature Pyramid Networks (FPN) [36], a two-stage detector based on Faster R-CNN. "This architecture, called a Feature Pyramid Network (FPN), shows significant improvement as a generic feature extractor in several applications. Using a basic Faster R-CNN system, our method achieves state-of-the-art single-model results on the COCO detection benchmark . . . " [36]. FPN shows great advances in detecting objects with a wide variety of scales.

In 2017, He et al. presented Mask R-CNN, ". . . a conceptually simple, flexible, and general framework for object instance segmentation" [37]. He et al.'s method efficiently detects objects while simultaneously generating a high-quality segmentation mask for each instance in an image. Mask R-CNN is based on Faster R-CNN by integrating a module for predicting an object mask in parallel with the existing module for bounding box recognition. Mask R-CNN is simple to train, runs at 5 frames per second, and is easy to generalize to other tasks such as pose estimation.

2.3. Single-Stage Detectors

Single-stage detectors classify and localize objects in a single time. They use predefined bounding boxes of various scales and aspect ratios to localize objects. Single-stage detectors have a simpler design, attain real-time performance, and surpass two-stage detectors.

In 2016, Redmon et al. presented "You Only Look Once" (YOLO) [38], it was the first single-stage detector in the DL era. This network divides the image into regions and predicts bounding boxes and probabilities for each region simultaneously. Later, Redmon et al. made a series of improvements to YOLO and proposed its second and third versions [39,40], which further improve the detection accuracy while keeping a very high detection speed.

In 2016, Liu et al. presented Single Shot MultiBox Detector (SSD) [41]. It was the second single-stage detector in the DL era. "Our approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape" [41]. SSD does not use proposal generation and feature re-sampling. It combines predictions from multiple feature maps with different resolutions to easily detect objects of various sizes. The contributions of SSD, multi-reference, and multi-resolution detection techniques significantly improve the detection accuracy of single-stage detectors. Furthermore, SSD was significantly faster and more accurate than both SOTA networks like Faster R-CNN and YOLO but it had difficulty in detecting small objects.

In 2017, Redmon et al. presented YOLOv2 and YOLO9000 [39] as an improvement to the first version of YOLO [38]. YOLOv2 offered an easy trade-off between speed and accuracy while the YOLO9000 model could predict 9000 object classes in real time. Redmon et al. replaced the backbone architecture of GoogLeNet [42] with DarkNet-19 [43].

In 2017, RetinaNet was presented by Lin et al. [44]. RetinaNet predicts objects by the dense sampling of the input image in aspect ratio, location, and scale. A new loss function named "focal loss" has been introduced in RetinaNet by reshaping the standard cross entropy loss so that the RetinaNet detector will put more focus on hard and misclassified examples during training. Focal Loss makes it possible for the single-stage detectors to achieve comparable accuracy of two-stage detectors while keeping a very high detection speed.

In 2018, Redmon et al. presented YOLOv3 [40], an improvement on YOLO [38], YOLOv2 and YOLO9000 [39]. Redmon et al. replaced the feature extractor network with a larger Darknet-53 network [43]. They also incorporated numerous techniques like batch normalization, data augmentation, and multiscale training, among others. In YOLOv3, the Softmax classifier layer was replaced by a logistical classifier.

In 2019, EfficientDet was presented by Tan et al. [45]. Tan et al. studied network architecture design choices for efficient object detection and proposed a weighted bidirectional feature network and a customized compound scaling method, in order to improve the efficiency and accuracy of the detector. Based on these optimizations and better backbones, Tan et al. developed a new family of detectors that achieved better efficiency and accuracy than the SOTA architectures, named EfficientDet [45].

In 2020, Bochkovskiy et al. presented the YOLOv4 [46] model by efficiently scaling the network design and scale, surpassing the previous SOTA EfficientDet [46]. The backbone of the YOLOv4 network is the CSPDarkNet53, a feature extraction network that calculates the feature maps from the source image. The YOLOv4 architecture neck is composed of an SPP module and a Path Aggregation Network (PANet). The architecture head processes the aggregated features and predicts the bounding boxes, objectness scores, and classification scores. The YOLOv4 network uses the YOLO layer (same as YOLOv3 [40]) as the architecture head.

Finally, YOLO is the fastest general-purpose object detector in the literature and YOLO pushes the SOTA in real-time object detection [47]. YOLO also generalizes well to new domains, making it ideal for applications that rely on fast and accurate object detection. In 2021, YOLOR was presented to the computer vision community and achieved comparable object detection accuracy as YOLOv4, while the inference speed was increased by 88% [48]. Some examples of custom DL techniques developed for the end of object detection could be seen in [49–53].

In the next two sections, the DL YOLOv5 and YOLOR object detectors are presented in more detail.

3. YOLOv5

The YOLO architecture was presented by Joseph Redmon et al. in 2016 [38]. The YOLO architecture is defined by a single neural network trained end-to-end with an image input and predicts bounding boxes and class labels. YOLO operates at 45 frames per second and can operate up to 155 frames per second for a speed-optimized version of the model [38]. This architecture works by splitting the input image into a grid of cells and each cell is responsible for predicting a bounding box only if the center of a bounding box falls within the cell. Each grid cell predicts a bounding box involving the x and y coordinates, the width, height, and the confidence of the prediction. A class prediction is based on each cell. The bounding boxes with confidences and the class probability map are then combined into a final set of bounding boxes and class labels.

The YOLO architecture was updated by Joseph Redmon and Ali Farhadi in an effort to further improve the model performance [39]. Various training and architectural changes were made to the YOLO model, such as batch normalization and the use of high-resolution input images. YOLOv2 architecture makes use of anchor boxes like Faster R-CNN and predefined bounding boxes with useful sizes and shapes that are tailored during training. The choice of bounding boxes for the image is pre-processed using a k-means analysis on the training dataset. Importantly, the predicted representation of the bounding boxes is changed so that small changes have less effect on the predictions, resulting in a stronger architecture. Rather than predicting position and size directly, for reshaping and moving the predefined anchor boxes, offsets are predicted relative to a grid cell and dampened by a logistic function.

Further improvements to the model were proposed by Joseph Redmon and Ali Farhadi in YOLOv3 [40]. The improvements were reasonably minor, including minor representational changes and a deeper feature detector network.

Bochkovskiy et al. presented the YOLOv4 [46] architecture in 2020. YOLOv4 has been developed by efficiently scaling the network scale and design, surpassing the previous SOTA EfficientDet [45]. Bochkovskiy et al.'s developments of the YOLO model surpass prior benchmarks from previous object detection architectures on the speed versus accuracy frontier [46]. "We offer a state-of-the-art detector which is faster (FPS) and more accurate (MS COCO $AP_{50...95}$ and AP_{50}) than all available alternative detectors" [46]. In general, the authors of YOLOv4 presented a few scaling concepts in balance as they developed the model (e.g., image size, number of layers, and number of channels), while optimizing for inference speed and model performance. To detect large objects in large images, Bochkovskiy et al. concluded that it is important to increase the depth and number of stages in the CNN backbone and neck. This allowed them to first scale up the input size and number of stages, and dynamically adjust width and depth according to real-time inference speed requirements.

YOLOv5 [54] was released in 2020, very shortly after YOLOv4. This implementation shares the same design and provides higher performance than YOLOv4. YOLOv5 is fully written in the PyTorch framework, different from the past versions of YOLO that use the Darknet framework. YOLOv5 is significantly smaller, faster to train, and more accessible to use in a wider range of development environments (Ultralytics YOLOv5 GitHub repository (last seen 12 December 2022): https://github.com/ultralytics/yolov5). A performance comparison between EfficientDet and YOLOv5 can be seen in Figure 2.

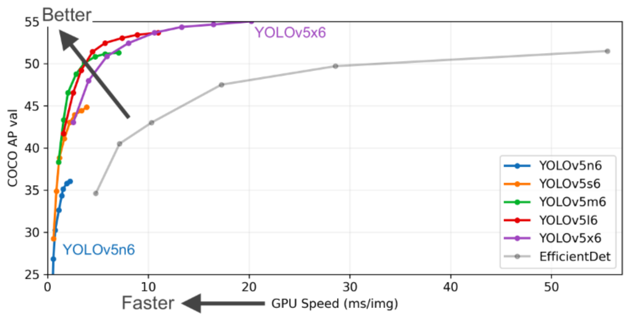

Figure 2. Performance of YOLOv5 and EfficientDet (information taken from Ultralytics YOLOv5 GitHub page).

The main differences between YOLOv3, YOLOv4, and YOLOv5 architectures are that YOLOv3 uses Darknet-53 backbone, YOLOv4 architecture uses CSPdarknet53 backbone, and YOLOv5 uses focus structure with CSPdarknet53 as a backbone. The focus layer is first introduced in YOLOv5. The CSPDarknet53 backbone solves the repetitive gradient information in large backbones and integrates gradient change into a feature map that reduces the inference speed, increases accuracy, and reduces the model size by decreasing the parameters [55]. YOLOv5 uses PANet as a neck to boost the information flow. PANet adopts a new FPN that includes several bottom-up and top-down layers, improving the propagation of low-level features in the model. PANet improves the localization in lower layers, which enhances the localization accuracy of the object. In addition, the head in YOLOv5 is the same as YOLOv4 and YOLOv3, the YOLO layer, which generates three different outputs of feature maps to achieve multiscale prediction. It also helps to enhance the prediction of small to large objects efficiently in the model. The image is fed to CSPDarknet53 for feature extraction and again fed to PANET for feature fusion. Finally, the YOLO layer (YOLOv5 head) generates the results (class, score, location, and size). A YOLOv5 structure diagram can be seen in Figure 3.

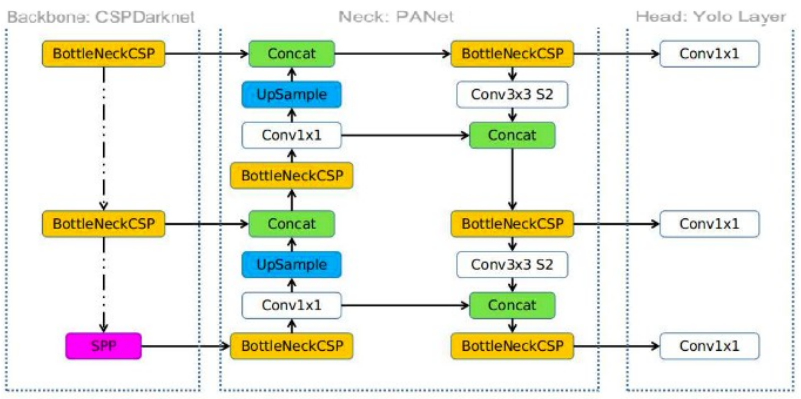

Figure 3. The structure diagram of the YOLOv5 network [56]. It consists of three parts: the Backbone that is the CSPDarknet53, the neck that is PANet and the Head that is the YOLO Layer.

4. YOLOR

In 2021, Wang et al. presented YOLOR [48], a SOTA DL algorithm for object detection. YOLOR stands for "You Only Learn One Representation" and it was presented as a "unified network to encode implicit knowledge and explicit knowledge together". The presented YOLOR concept can be seen in Figure 4.

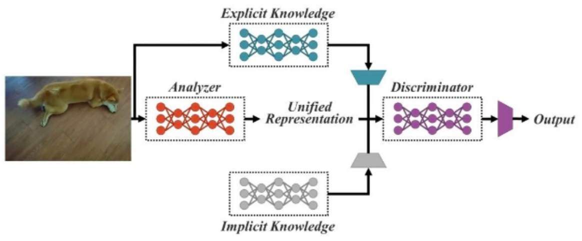

Figure 4. YOLOR concept with implicit and explicit knowledge-based multitask learning [48].

Wang et al. YOLOR paper describes an approach to combine explicit knowledge, defined as learning based on given data and input, with implicit knowledge learned subconsciously. "YOLOR unified network can generate a unified representation to simultaneously serve various tasks" [48]. YOLOR architecture applies prediction refinement, kernel space alignment, and multitask learning in a CNN and according to the results achieved, when implicit knowledge is introduced to the neural network that was already trained with explicit knowledge, the network benefits the performance of various tasks [48]. YOLOR achieved comparable object detection accuracy as the YOLOv4, while the inference speed was increased by 88% [48]. This made YOLOR one of the fastest object detection architectures in modern computer vision. A performance evaluation between YOLOv4, YOLOR, EfficientDet, and others, tested in the MS COCO dataset [8], is presented in Figure 5 (Information taken from Wong Kin-Yiu GitHub repository (last seen 12 December 2022): https://github.com/WongKinYiu/yolor). Tests made in the MS COCO dataset show that the mean average precision of YOLOR is 3.8% higher compared to the PP-YOLOv2, at the same inference speed [48]. Wang et al. adopted the MS COCO dataset because it provides rich annotation content with the Ground Truth (GT) for many different tasks including object detection, instance segmentation, and multi-image classification.

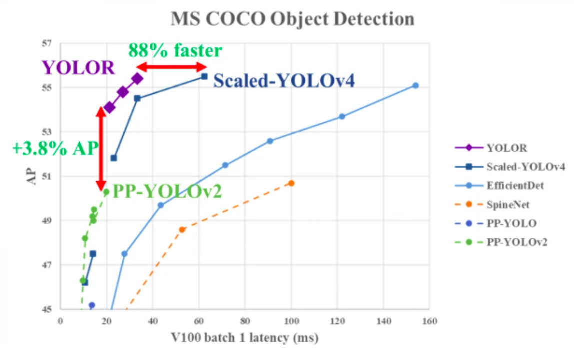

Figure 5. Performance evaluation of YOLOR vs. YOLOv4 vs. EfficientDet vs. others tested in the MS COCO dataset [8].

Wang et al. mention in their paper that YOLOR architecture is based on the architecture of YOLOv4 [48]. "YOLOR-P6 has same architecture as YOLOv4-P6-light, we replace all Mish activation in YOLOv4-P6-light by SiLU activation" [48]. "YOLOR-W6 is wider YOLOR-P6, base channels are set as {128, 256, 512, 768, 1024}" [48]. "YOLOR-E6 expands the width of YOLOR-W6, the width scaling factor is set as 1.25, and all of the convolution down-sampling modules are replaced by CSP convolution" [48]. The YOLOv4-P5, YOLOv4-P6 (same architecture as YOLOR-P6) and YOLOv4-P7 architecture diagrams can be seen in Figure 6.

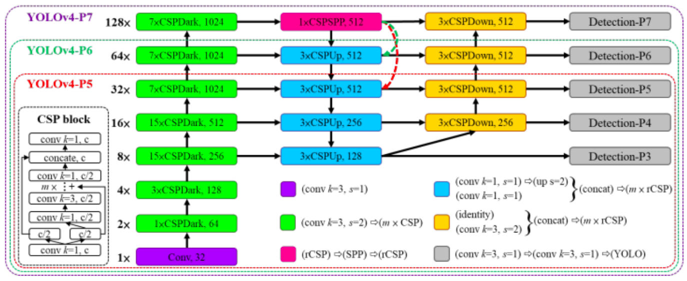

Figure 6. Architecture of YOLOv4-large, including YOLOv4-P5, YOLOv4-P6 (same architecture as YOLOR-P6) and YOLOv4-P7 [57].

5. Methodology and Results

5.1. YOLOv5 and YOLOR Training

Using Google Colab (Google Colab website (last seen 22 February 2022): https://colab.research.google.com/) a YOLOv5 network was trained for vegetation detection in a forest environment for a thousand epochs and a YOLOR network was trained for five hundred epochs. The training was made using an NVIDIA Tesla T4 GPU with 16 GB of memory for both models. YOLOv5 and YOLOR used base models that were pre-trained in COCO Dataset [8].

The training dataset is composed of three hundred images. Two-hundred ten for training, 45 for validation, and 45 for testing. These training images were labeled with five classes (using RoboFlow label tool (RoboFlow website (last seen 22 February 2022): https://roboflow.com/)): "Live Vegetation", "Grass", "Cut Vegetation", "Tree Trunk" and "Dead Vegetation". The class "Live Vegetation" can be defined as green vegetation in an upright position, the class "Grass" can be defined as small live and green vegetation low on the ground, the class "Cut Vegetation" can be defined as cut down vegetation and also minor tree branches on the ground. The class "Tree Trunk" can be defined as tree trunks in an upright position. Lastly, the class "Dead Vegetation" can be defined as dead/dry vegetation that was not cut down and is in an upright position. The dataset was acquired in a forest, near the Department of Electrical and Computer Engineering of the University of Coimbra. The vegetation in and around Coimbra includes a mix of Mediterranean and Atlantic flora. The dataset images were acquired with a JAI FS-1600D-10GE (JAI cameras website (last seen 21 July 2023): https://www.jai.com/products/fs-1600d-10ge) and with an Intel RealSense D455 (Intel Realsense cameras website (last seen 21 July 2023): https://www.intelrealsense.com/depth-camera-d455/).

The YOLOv5 model was trained with the default parameter settings (with default hyperparameters). The used batch size for training was 16 and the image size was 640 × 640 pixels. The YOLOR-P6 model version was trained with the default parameter settings (with default hyperparameters). The used batch size for training was 16 and the image size was 640 × 640 pixels.

Figure 7 shows the methodology used, as a diagram, for the model training and assessment and dataset development.

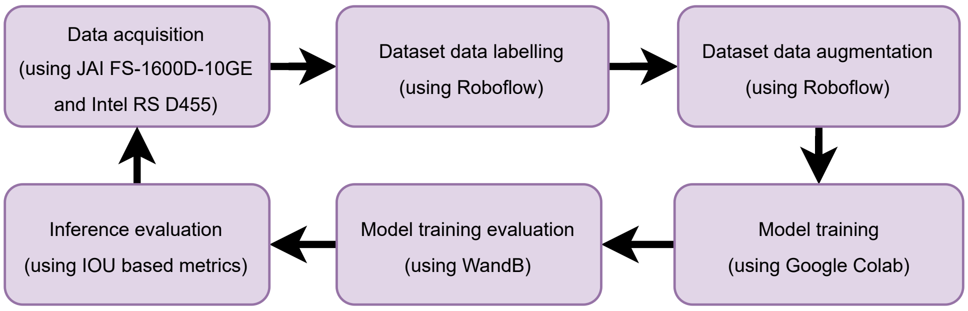

Figure 7. Diagram of the methodology used for the models training and dataset development.

5.1.1. Computation Times for Training

Table 1 presents the computation times of the YOLOv5 and YOLOR models training with images with 1440 × 1080 pixels, in hours. These training computation times were obtained with an NVIDIA Tesla T4 GPU with 16 GB of memory. The processing was made using the GPU only.

Table 1. Processing time of YOLOR and YOLOv5 models for training with images with 1440 × 1080 pixels in a NVIDIA Tesla T4 GPU in hours.

	Training for 500 Epochs	**Training for 1000 Epochs**
YOLOv5 (h)	0.417	1.121
YOLOR (h)	2.383	4.466

As it can be concluded from Table 1, the YOLOv5 training time takes 17.5% of the YOLOR training time when training for 500 epochs. Also, YOLOv5 training time takes 25.1% of the YOLOR training time when it is trained for 1000 epochs. YOLOv5 is faster to train, which results in faster deployment of models for object detection in any system.

5.1.2. Size of the Trained Weight File

Table 2 presents the trained weight file size of the YOLOv5 and YOLOR models. The weight file is used to load the DL network to make inferences about objects in images.

Table 2. Size of the YOLOR and YOLOv5 best epoch trained weight file in megabytes (MB).

	Size in Megabytes
YOLOv5	14.8 MB
YOLOR	295.7 MB

YOLOv5 weight file is 95% smaller than YOLOR, which results in an easier deployment of projects for object detection in embedded systems with low memory.

Also, YOLOv5 allows exporting the trained weight file to eleven different formats such as TorchScript, ONNX, TensorFlow Lite, or TensorRT. The YOLOv5 exported weight file in the TensorFlow Lite format allows object detection project development for Android smartphones. YOLOR only allows users to save the trained weight file as a .pt weight file to be used in any given system with a GPU.

5.2. Vegetation Detection with the YOLOv5 Trained Model

Next, we present the obtained results of the vegetation detection using the YOLOv5-trained network.

Figures 8–11 present the mean Average Precision (mAP) at 0.5 and from 0.5 to 0.95 Intersection Over Union (IOU), precision, recall, box loss, classification loss, and objectness loss that resulted from the YOLOv5 training for a thousand epochs. These metrics are obtained over the training and validation sets from the training dataset.

From the resulting training performance metrics, it is possible to conclude that the maximum mAP achieved at 0.5 IOU of all classes is 70%. The maximum mAP from 0.5 to 0.95 IOU is 30%. The maximum precision achieved is 82%, which means that the true positives detected in the images are much higher than the false positives detected. The maximum achieved recall is 64%, which means there are more true positives than false negatives. The training losses, box loss, classification loss, and objectness loss tend to be zero, as intended.

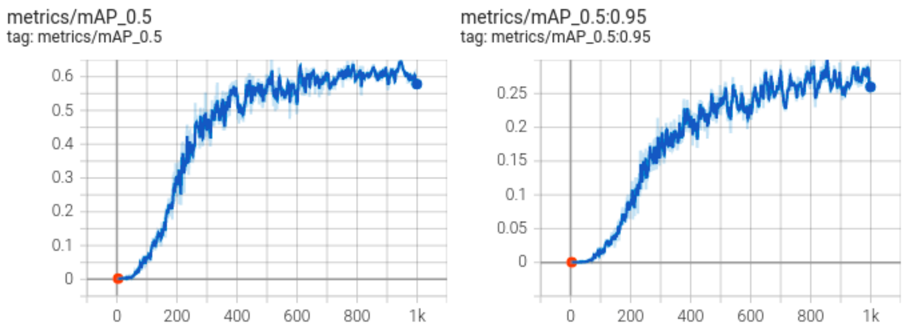

Figure 8. YOLOv5 model training performance metrics mean Average Precision (mAP) at 0.5 Intersection Over Union (IOU) (**left**) and from 0.5 to 0.95 IOU (**right**) during the 1000 epochs.

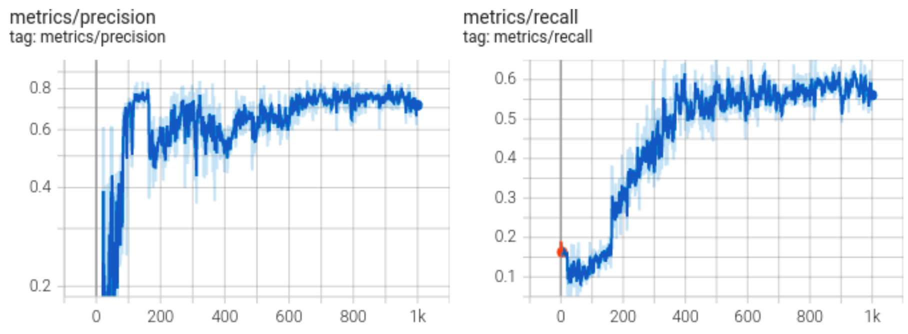

Figure 9. YOLOv5 model training performance metrics precision (**left**) and recall (**right**) during the 1000 epochs.

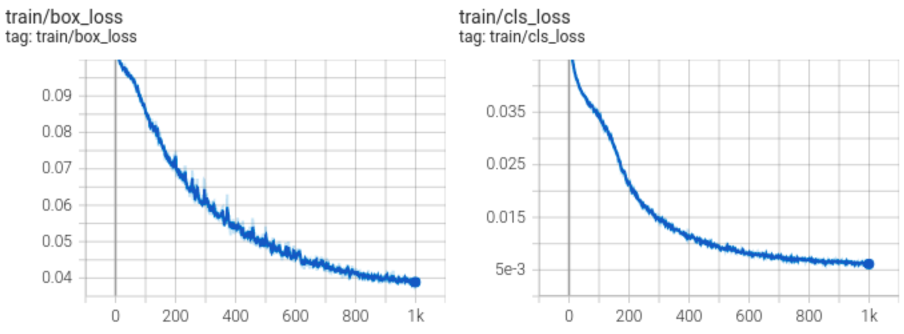

Figure 10. YOLOv5 model training box loss (**left**) and the classification loss (**right**) during the 1000 epochs.

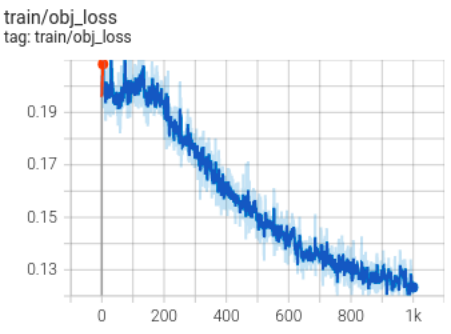

Figure 11. YOLOv5 model objectness loss evolution during the 1000 epochs training.

The box loss represents how well the algorithm can locate the center of an object and how well the predicted bounding box covers an object. Objectness is essentially a measure of the probability that an object exists in a proposed region of interest. If the objectivity is high, this means that the image window is likely to contain an object. Classification loss indicates how well the algorithm can predict the correct class of an object [49]. The box loss, objectness loss, and classification loss are indicators of how well an algorithm predicts an object [58].

The model improved significantly in terms of precision, recall, and mAP after 500 epochs and became stable after 600 epochs, which means that stopping the model early would give almost the same results in 40% to 50% less training time.

Also, it is possible to achieve even better precision, mAP, and recall in detecting forest vegetation for YOLOv5 using other pre-trained models (pre-trained on the MS COCO dataset [8]) such as YOLOv5m, YOLOv5l or YOLOv5x models (YOLOv5 Ultralytics website (last seen 10 October 2022): https://github.com/ultralytics/yolov5) instead of YOLOv5s.

After the model was trained, for model inference, it was fed with unseen images, not belonging to the dataset, with a confidence threshold of 0.4. Figure 12 shows the RGB source images (at left), the resulting object detection using the trained YOLOv5 model (at right), and the GT annotated images in the center. The color code for the GT Bounding Boxes (BBs) are as follows: red for the class "Live Vegetation", cyan for the class "Grass", blue for the class "Tree Trunk", purple for the class "Dead Vegetation" and orange for the class "Cut Vegetation".

The objects intended to be detected in the source images are detected with YOLOv5 inference from the trained model (e.g., grass, cut vegetation, tree trunks, dead vegetation, and live vegetation), as the resulting inference follows the labeled data for YOLOv5 network training.

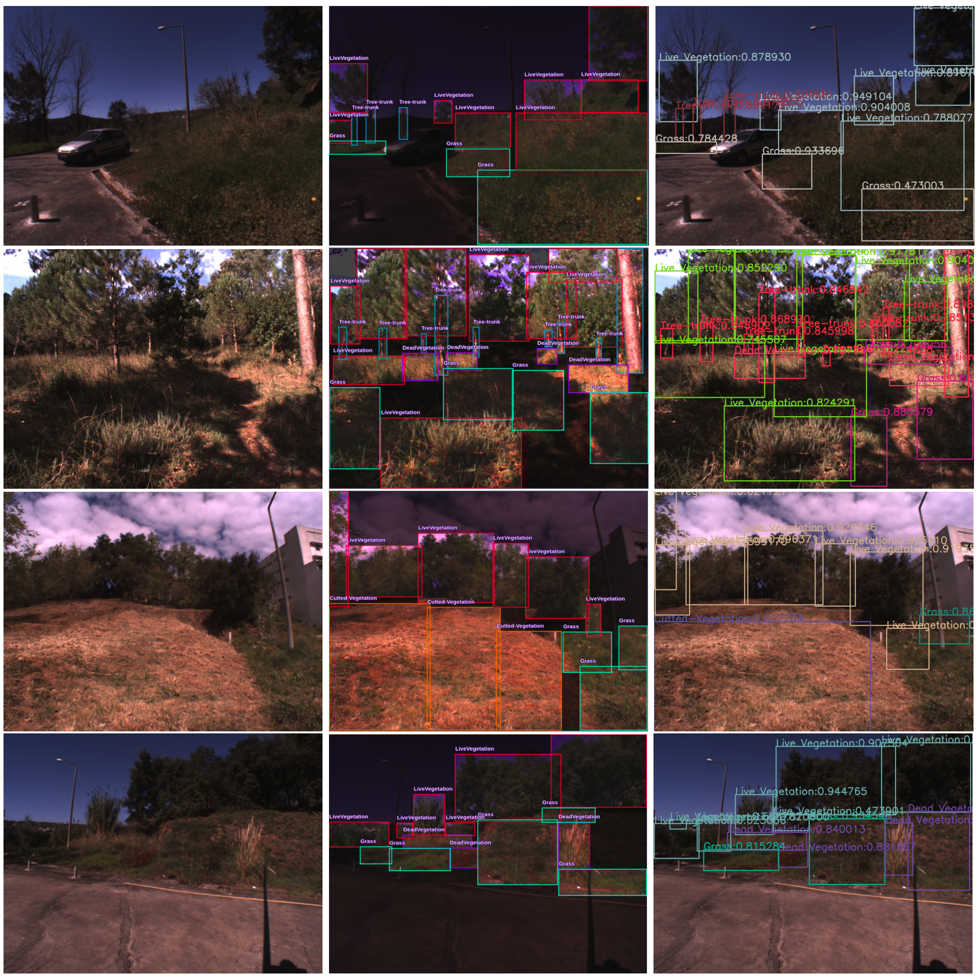

Figure 12. Object detection using the trained YOLOv5 model. Source RGB images at left, inference results at right and in the center the Ground Truth (GT).

5.3. Vegetation Detection with the YOLOR Trained Model

Next, the obtained results of the vegetation detection using the YOLOR trained model are presented.

Figures 13–16 present the mAP at 0.5 and from 0.5 to 0.95 IOU, precision, recall, box loss, classification loss, and objectness loss that resulted from the YOLOR training for five hundred epochs. These metrics are obtained over the training and validation sets from the training dataset.

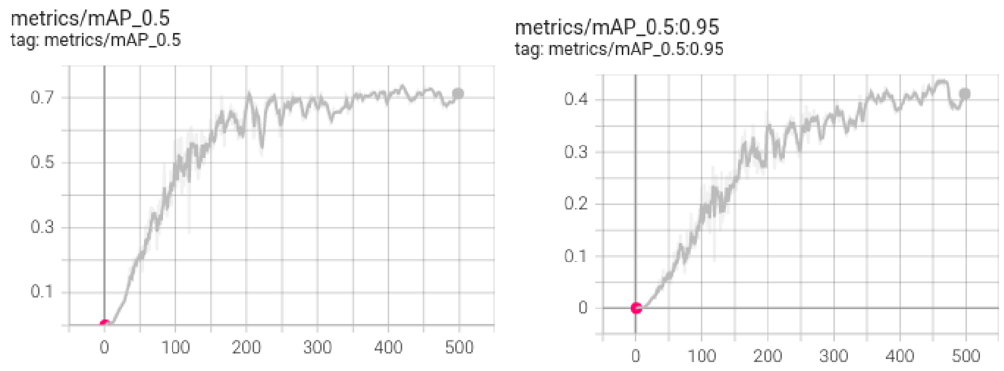

Figure 13. YOLOR model training performance metrics mAP at 0.5 IOU (**left**) and from 0.5 to 0.95 IOU (**right**) during the 500 epochs.

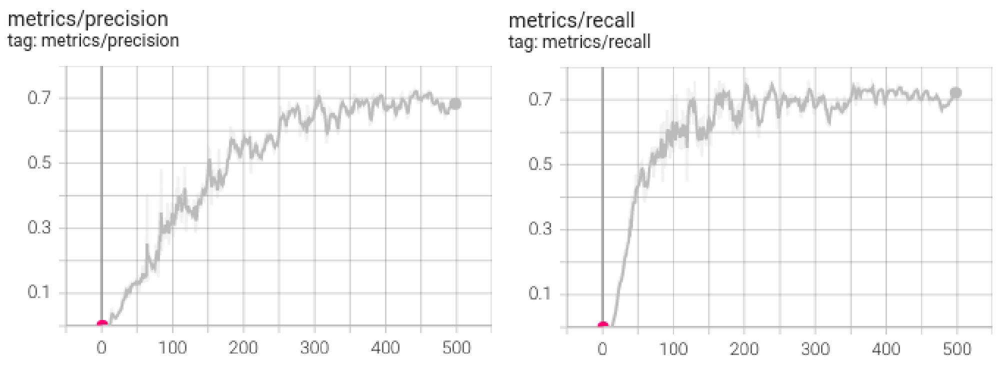

Figure 14. YOLOR model training performance metrics precision (**left**) and recall (**right**) during the 500 epochs.

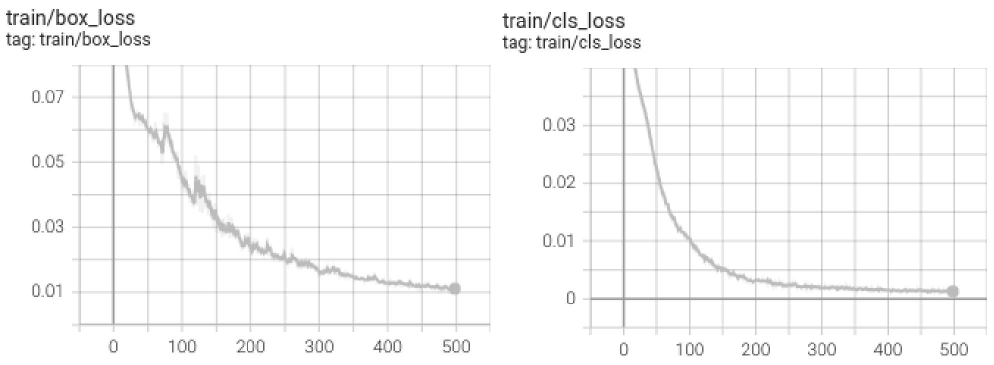

Figure 15. YOLOR model training box loss (**left**) and the classification loss (**right**) during the 500 epochs.

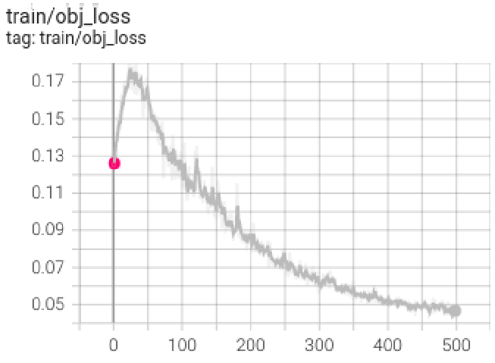

Figure 16. YOLOR model objectness loss evolution during the 500 epochs training.

The model improved significantly in terms of precision, recall, and mAP after 300 epochs and reached a plateau after 350 epochs, which means that the model training could be stopped after 350 epochs. The model would give almost the same results with 70% of the 500 training epochs (with less training time).

From the resulting training performance metrics, it is possible to conclude that the maximum mAP at 0.5 IOU achieved is 73%. The maximum mAP from 0.5 to 0.95 IOU is 48%. The maximum precision achieved is 72%. The maximum achieved recall is 74%. The training losses, box loss, classification loss, and objectness loss tend to be zero, as intended.

After the model was trained, for the model inference, it has been fed unseen images, not belonging to the dataset, with a confidence threshold of 0.4. Figure 17 shows the unseen (for YOLOR model) RGB images (at left), the resulting object detection using the trained YOLOR model (at right), and the GT annotated images in the center. The color codes for the GT BBs are as follows: red for the class "Live Vegetation", cyan for the class "Grass", blue for the class "Tree Trunk", purple for the class "Dead Vegetation" and orange for the class "Cut Vegetation". The inference BB's color is random.

When compared with YOLOv5 resulting inference images (from the third row the right image from Figure 12) it is possible to conclude from the right image from the third row from Figure 17 that the dead vegetation is not detected for YOLOR inference and it is detected for YOLOv5 inference. When comparing YOLOR and YOLOv5 inference results it is possible to conclude that YOLOv5 makes inference for more true positives than YOLOR, as mentioned in the last example, even when trained with the same dataset. From the training performance metrics obtained from YOLOv5 and YOLOR, the precision achieved by YOLOR is lower than with YOLOv5, concluding that fewer true positives are predicted for YOLOR. In the next subsection, a quantitative study assessing and comparing YOLOv5 and YOLOR results is presented.

5.4. Inference Assessment with the GT

Table 3 presents the mean IOU of the inference BBs (shown in Figures 12 and 17 at the right column) with the GT BBs (seen in Figures 12 and 17 at the center column), number of False Negatives (FN), False Positives (FP), True Positives (TP), precision, recall and F1 score for each one of the source images for YOLOv5 and YOLOR. The F1 score is a single value that takes into account both precision and recall, making it a useful metric for tasks with imbalanced classes, such as object detection. A high F1 score indicates that the object detector performs well in both detecting relevant objects and minimizing false positives [59]. As can be concluded from Table 3, the mean IOU is higher for YOLOV5, the number of FP predicted is the same for YOLOv5 and YOLOR, the number of FN predicted is higher for YOLOR, and the number of TP is higher for YOLOv5.

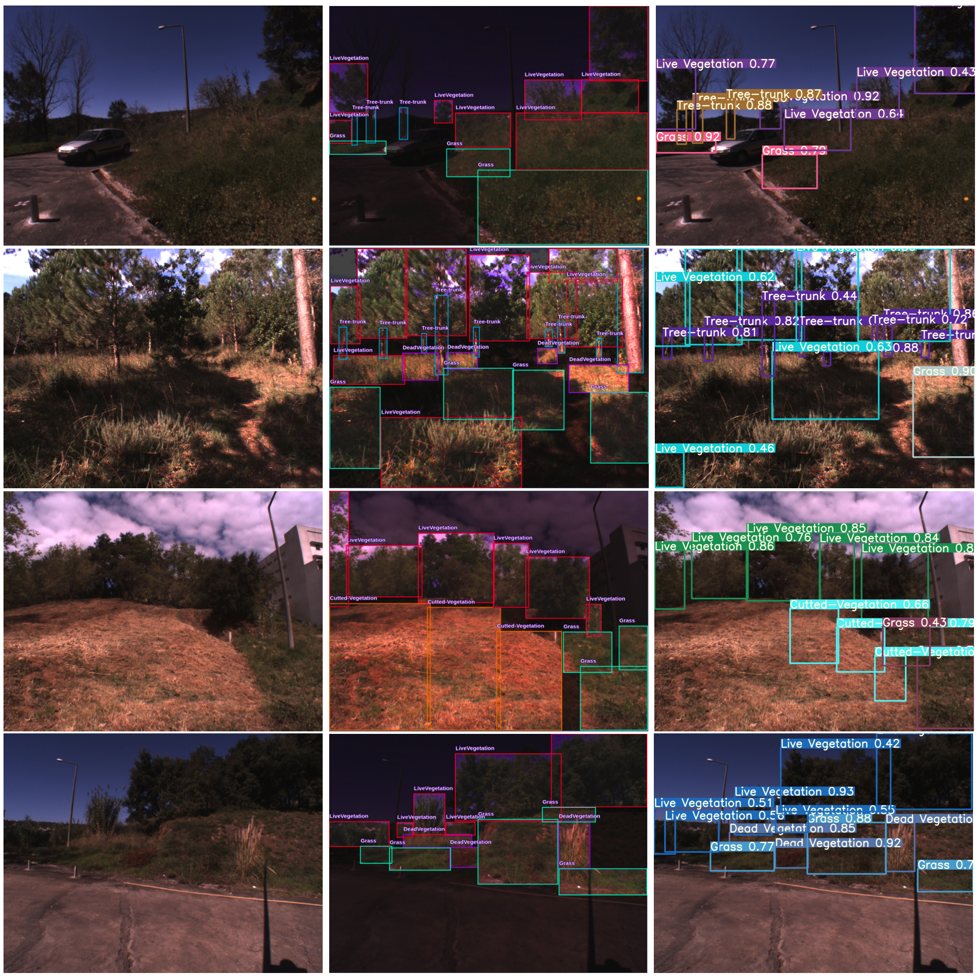

Figure 17. Object detection using the trained YOLOR model. Source RGB images at left, inference results at right and in the center the GT.

Table 3. Inference assessment with the Ground Truth (GT) for the four images shown in Figures 12 and 17.

	Image 1	**Image 2**	**Image 3**	**Image 4**
YOLOv5 inference mean IOU	0.607	0.625	0.676	0.682
YOLOR inference mean IOU	0.551	0.628	0.532	0.681
YOLOv5 inference FP	0	1	0	1
YOLOR inference FP	0	2	0	0
YOLOv5 inference FN	1	4	1	3
YOLOR inference FN	4	12	3	3
YOLOv5 inference TP	13	20	9	13
YOLOR inference TP	10	14	10	12
YOLOv5 inference precision	1	0.952	1	0.929
YOLOR inference precision	1	0.875	1	1
YOLOv5 inference recall	0.929	0.833	0.9	0.813
YOLOR inference recall	0.714	0.539	0.769	0.8
YOLOv5 inference F1 score	0.963	0.889	0.947	0.867
YOLOR inference F1 score	0.833	0.667	0.869	0.889

5.5. Inference Assessment with a Test Set

To evaluate the results achieved with the trained model in the inference of unseen images (in the training), it was made inference in a different dataset, with the test set composed of 946 labeled images. This dataset is composed of images with the same types of forest vegetation, labeled as defined in Section 5.1. These images were acquired in distinct regions from the training images, with different illumination conditions. This test set contains images with motion and lighting noise, and also, images acquired in wildland-urban interface areas. Table 4 presents, for the 946 images of the used test set, the precision, recall, F1 score, and the mean Average Precision at 0.5 IOU (mAP@0.5) and the mean Average Precision from 0.5 to 0.95 IOU (mAP@0.5:0.95) for YOLOv5 and YOLOR.

Table 4. Inference assessment for a 946 images test set.

Model	Precision	Recall	F1 Score	mAP@0.5	mAP@0.5:0.95
YOLOv5	0.466	0.116	0.186	0.485	0.299
YOLOR	0.406	0.101	0.162	0.236	0.119

As can be concluded from Table 4, the precision, recall, F1 score, and mAP are higher for YOLOV5. The test performance metrics presented in Table 4 show good results in precision, recall, F1 score, mAP at 0.5 IOU, and mAP at 0.5 to 0.95 IOU in a test set composed of 946 images, when using the trained YOLOv5 model trained in a 300 images only dataset. The higher results in the precision, recall, and F1 score for YOLOv5 means fewer False Positives (FP), fewer False Negatives (FN), and a higher number of True Positives (TP) in the model inference. The higher mAP achieved means that the YOLOv5 object detection model is effectively finding and accurately localizing a higher number of objects in the dataset when compared with YOLOR.

5.6. Inference Computation Times

Table 5 presents the computation times of the YOLOv5 and YOLOR models for vegetation detection. The four source images shown in Figures 12 and 17 have a 1440 $\times$ 1080 pixels size. The computation times were obtained with an NVIDIA Tesla T4 GPU with 16 GB of memory. The processing was made using the GPU only.

Table 5. Inference computation times of YOLOv5 and YOLOR models for images with 1440 $\times$ 1080 pixels.

	Image 1	Image 2	Image 3	Image 4
YOLOv5 (s)	0.0536	0.0160	0.0150	0.0154
YOLOR (s)	0.228	0.197	0.231	0.223

From Table 5, it is possible to conclude that YOLOv5 inference is 88.6% faster than YOLOR inference (on average). On average, YOLOv5 inference speed takes 11.4% of YOLOR inference speed using a NVIDIA Tesla T4 GPU. YOLOv5 inference computation time, on average, takes 25 milliseconds per image. YOLOv5 inference can run at more than 40 frames per second (in real time). YOLOR, at maximum, can run at 5 frames per second (using an NVIDIA Tesla T4 GPU).

In embedded systems using YOLOv5, the system needs lower specifications than using YOLOR because of the size of the weight file and speed of inference (using the GPU). YOLOv5 makes it possible to make inferences using the CPU, which can be used in systems without a GPU. YOLOR can only be used in systems with a GPU.

5.7. Integration in the UGV Platform

The used UGV platform, from SafeForest (SafeForest project website (last seen 7 December 2022): https://safeforest.ingeniarius.pt/) and SEMFIRE (SEMFIRE project website (last seen 7 December 2022): http://semfire.ingeniarius.pt/) projects, navigates in wildland (forests) and wildland-urban interfaces areas [60–68]. Presently, the main challenge to be addressed is the reliable discrimination between obstacles (mainly trees) and forest vegetation to be cleaned by the UGV cleaning tool. YOLOv5 detects, classifies, and localizes in 2*D* images various types of forest vegetation (such as live vegetation, grass, cut vegetation, tree trunks, and dead vegetation). The UGV has a vegetation cleaning tool and depth cameras attached in the front of the vehicle (as shown in Figure 1). The trained YOLOv5 object detection model will be used, with the UGV frontal depth camera, to make inferences in the RGB images retrieved by the depth camera. The retrieved depth information is obtained as a 2*D* image with the depth information per pixel, and a 2*D* depth image with the same characteristics as the RGB image from the same camera. Since YOLOv5 object detection has the output of all BB characteristics (x, y, width, and height) this information is used to calculate the same BB in the depth image. These BBs define the regions of interest where the object is detected in the image. After obtaining the BB in the depth image, it is possible to calculate the 3*D* coordinates in the real world of the object detected that results from the 2*D* bounding box.

6. Discussion

Object detection using YOLOv5 and YOLOR was developed and evaluated for the purpose of forest vegetation detection, classification, and localization for an autonomous UGV to clean forest fuel and to prevent forest fires. A specialized dataset comprising forest images near Coimbra University was created specifically for training DL object detector models for forest vegetation detection.

As shown in Table 1, the YOLOv5 training time is 17.5% of the YOLOR training time when training for 500 epochs. Additionally, YOLOv5 takes 25.1% of the YOLOR training time when it is trained for 1000 epochs. YOLOv5 is faster to train, which results in faster deployment of models for object detection. As shown in Table 2, the YOLOv5 best epoch trained weight file is 95% smaller than YOLOR's best epoch-trained weight file which results in an easier deployment of projects for object detection in embedded devices with low memory. Also, YOLOv5 allows exporting of the trained weight file in a wider variety of formats such as TorchScript, ONNX, TensorFlow Lite, or TensorRT. When comparing YOLOv5 and YOLOR for inference in images it is concluded that YOLOv5 achieves the detection of more true positives than YOLOR. In Table 5, on average, YOLOv5 inference speed is 11.4% of the YOLOR inference speed using a NVIDIA Tesla T4 GPU. YOLOv5 inference can run in real time with more than 40 frames per second, while YOLOR can only run at 4 frames per second.

From the resulting training performance metrics, YOLOR achieves maximum mAP at 0.5 and from 0.5 to 0.95 IOU (73% and 48%, respectively, higher mAP when compared with 70% and 30% from YOLOv5). YOLOv5 achieves the maximum precision of 83% and YOLOR achieves 72%. YOLOR achieves a higher recall of 74% when compared with the 64% of YOLOv5. The training losses, box loss, classification loss, and objectness loss tend to be zero, for YOLOv5 and YOLOR, as intended. It is also possible to achieve better performance (e.g., precision, mAP, and recall) from YOLOv5 using other pre-trained models such as YOLOv5m, YOLOv5l or YOLOv5x instead of YOLOv5s.

The quantitative metrics of Table 3 show that the mean IOU is higher for YOLOV5, the number of false positives predicted is the same for YOLOv5 and YOLOR, the number of false negatives predicted is higher for YOLOR and the number of true positives is higher for YOLOv5. The precision, recall, and F1 score shown in Tables 3 and 4 lead us to conclude that in the source images used, the YOLOv5 object detector usually performs better in detecting positive instances (true positives), the detection of false instances (false positives) is reduced, and also, the false negatives detection is reduced. In Table 4, the achieved inference mAP in the used test set (for evaluation of the models) is higher for YOLOv5, which means that YOLOv5 object detection model is effectively finding and localizing a higher number of objects in the dataset.

The main finding of this work is the identification of a real-time object detector (YOLOv5) capable of accurately detecting various types of vegetation in forest ground vehicle images. Also, it can be used by a UGV to identify where to clean forest fires fuel, in real time. The main error sources that lead to negative outcomes are the few training images, and both lighting and motion noise in the test dataset.

Some of the drawbacks of DL object detectors are the rectangular BB detection (different from instance or semantic segmentation), time-consuming and expensive dataset labeling, the need for medium/high-resolution images (for higher inference accuracy), and the need for the dataset to have thousands of images in order to let the trained model be used in distinct types of forests, different weather and lightening conditions, and time of the year. On the other hand, one of the advantages is the accuracy of the DL object detector (YOLOv5) with the real-time capability to detect forest ground vegetation to be cut, using ground vehicle RGB images. Also, the work presented can be used for the UGV local navigation and path planning. The hardware required to attain the demonstrated outcomes is readily accessible and cost-effective. Moreover, to obtain better inference results, the training dataset should be iteratively improved with different types of vegetation and lighting conditions.

In future work, the result of this research will be integrated into the SafeForest UGV ROS software through vegetation 3*D* localization using the UGV depth camera outputs.

7. Conclusions

The work addressed was a study of forest vegetation detection and classification using ground vehicles' RGB images to support autonomous forest cleaning operations to prevent fires, using an UGV. The presented work compared two recent high-performance DL methodologies, YOLOv5 and YOLOR, to detect and classify forest vegetation in five classes: grass, live vegetation, cut vegetation, dead vegetation, and tree trunks.

From the presented work, it is concluded that forest vegetation detection in ground vehicles forest images can be achieved with recent DL object detectors. Advancements in technology and data processing techniques continue to improve the accuracy and reliability of these methods, surpassing problems such as dense vegetation, lighting conditions, occlusion, and complex backgrounds.

For the task of forest vegetation detection, YOLOv5 stands out as the superior DL object detection model due to its exceptional inference speed, precision, low training time, higher number of TP, lower number of FP, lower number of FN, and smaller size of the trained weight file when compared to YOLOR. YOLOv5 is better to use in low-specification embedded systems when compared with YOLOR object detection because of its inference speed and size of the trained weight file. YOLOv5 makes it possible to make inferences using the CPU, which can be used in embedded systems without GPU. YOLOR can only be used in systems with a GPU. Additionally, the resulting inference computation times show real-time processing capabilities for YOLOv5 object detection, as YOLOv5 can run at more than 40 frames per second in a NVIDIA Tesla T4 GPU, which is a step forward to the task of forest fuel cleansing for cluttered forest environments using autonomous robots.

In future work, the 3*D* localization and mapping of the detected vegetation using YOLOv5 and a depth camera will be developed. Also, more advanced DL object detectors will be trained in our custom dataset and compared with YOLOv5. Also, instance segmentation for one class detection (forest fire fuel) detection, classification, and localization in forest images will be developed.

Author Contributions: P.A.S.M. is the first author of this work and contributed for the conceptualization, methodology, software, investigation, data curation, visualization and writing—original draft preparation. A.P.C. and A.T.d.A. contributed for the validation, formal analysis, resources, supervision, project administration, funding acquisition and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Programa Operacional Regional do Centro, Portugal 2020, European Union through the FEDER Fund, through the project with the Operation Code: CENTRO-01-0247-FEDER-045931.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
BB	Bounding Box
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DL	Deep Learning
DPM	Deformable Parts Models
FN	False Negative
FP	False Positive
Fast R-CNN	Fast Region-based Convolutional Neural Network
FCN	Fully Convolutional Network
FPN	Feature Pyramid Network
GPU	Graphics Processing Unit
GT	Ground Truth
HOG	Histogram of Oriented Gradients
IOU	Intersection Over Union
mAP	mean Average Precision
MS COCO	Microsoft Common Objects in Context
PANet	Path Aggregation Network
R-CNN	Region-based Convolutional Neural Network
R-FCN	Region-based Fully Convolutional Network
RPN	Region Proposal Network
SOTA	State-Of-The-Art
SPP-Net	Spatial Pyramid Pooling Networks
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
TP	True Positive
UGV	Unmanned Ground Vehicle
YOLO	You Only Look Once
YOLOR	You Only Learn One Representation

References

1. Tang, J.; Li, S.; Liu, P. A review of lane detection methods based on deep learning. *Pattern Recognit.* **2021**, *111*, 107623. [CrossRef]
2. Dollar, P.; Wojek, C.; Schiele, B.; Perona, P. Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *34*, 743–761. [CrossRef] [PubMed]
3. Fang, C.Y.; Chen, S.W.; Fuh, C.S. Road-sign detection and tracking. *IEEE Trans. Veh. Technol.* **2003**, *52*, 1329–1341. [CrossRef]
4. Maldonado-Bascón, S.; Lafuente-Arroyo, S.; Gil-Jimenez, P.; Gómez-Moreno, H.; López-Ferreras, F. Road-sign detection and recognition based on support vector machines. *IEEE Trans. Intell. Transp. Syst.* **2007**, *8*, 264–278. [CrossRef]
5. Liu, Z.; Wang, S. Broken corn detection based on an adjusted YOLO with focal loss. *IEEE Access* **2019**, *7*, 68281–68289. [CrossRef]
6. Su, W.H. Advanced Machine Learning in Point Spectroscopy, RGB-and hyperspectral-imaging for automatic discriminations of crops and weeds: A review. *Smart Cities* **2020**, *3*, 767–792. [CrossRef]
7. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]

8. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. *arXiv* **2014**, arXiv:1405.0312.
9. Zare, A.; Bolton, J.; Gader, P.; Schatten, M. Vegetation mapping for landmine detection using long-wave hyperspectral imagery. *IEEE Trans. Geosci. Remote Sens.* **2007**, *46*, 172–178. [CrossRef]
10. Gandhi, G.M.; Parthiban, S.; Thummalu, N.; Christy, A. Ndvi: Vegetation Change Detection Using Remote Sensing and Gis—A Case Study of Vellore District. *Procedia Comput. Sci.* **2015**, *57*, 1199–1210. [CrossRef]
11. Bhandari, A.; Kumar, A.; Singh, G. Feature Extraction using Normalized Difference Vegetation Index (NDVI): A Case Study of Jabalpur City. *Procedia Technol.* **2012**, *6*, 612–621. [CrossRef]
12. Zhou, H.; Van Rompaey, A.; Wang, J. Detecting the impact of the "Grain for Green" program on the mean annual vegetation cover in the Shaanxi province, China using SPOT-VGT NDVI data. *Land Use Policy* **2009**, *26*, 954–960. [CrossRef]
13. Salamati, N.; Larlus, D.; Csurka, G.; Süsstrunk, S. Incorporating near-infrared information into semantic image segmentation. *arXiv* **2014**, arXiv:1406.6147.
14. Zhang, X.; Han, L.; Han, L.; Zhu, L. How Well Do Deep Learning-Based Methods for Land Cover Classification and Object Detection Perform on High Resolution Remote Sensing Imagery? *Remote Sens.* **2020**, *12*, 417. [CrossRef]
15. Ayhan, B.; Kwan, C.; Larkin, J.; Kwan, L.; Skarlatos, D.; Vlachos, M. Deep learning model for accurate vegetation classification using RGB image only. In Proceedings of the Geospatial Informatics X, SPIE, Online, 21 April 2020; Volume 11398, pp. 130–143.
16. Lobo Torres, D.; Queiroz Feitosa, R.; Nigri Happ, P.; Elena Cué La Rosa, L.; Marcato Junior, J.; Martins, J.; Olã Bressan, P.; Gonçalves, W.N.; Liesenberg, V. Applying Fully Convolutional Architectures for Semantic Segmentation of a Single Tree Species in Urban Environment on High Resolution UAV Optical Imagery. *Sensors* **2020**, *20*, 563. [CrossRef]
17. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, HI, USA, 8–14 December 2001; Volume 1, p. I-I. [CrossRef]
18. Viola, P.; Jones, M. Robust Real-Time Face Detection. *Int. J. Comput. Vis.* **2004**, *57*, 137–154. [CrossRef]
19. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893. [CrossRef]
20. Belongie, S.; Malik, J.; Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 509–522. [CrossRef]
21. Lowe, D. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157. [CrossRef]
22. Felzenszwalb, P.; McAllester, D.; Ramanan, D. A discriminatively trained, multiscale, deformable part model. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
23. Andrews, S.; Tsochantaridis, I.; Hofmann, T. Support vector machines for multiple-instance learning. *Adv. Neural Inf. Process. Syst.* **2002**, *15*.
24. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D. Cascade object detection with deformable part models. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2241–2248.
25. Girshick, R.B. *From Rigid Templates to Grammars: Object Detection with Structured Models*; The University of Chicago: Chicago, IL, USA, 2012.
26. Van de Sande, K.E.; Uijlings, J.R.; Gevers, T.; Smeulders, A.W. Segmentation as selective search for object recognition. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 1879–1886.
27. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*. [CrossRef]
28. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 23–28 June 2014; pp. 580–587.
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef]
30. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv* **2013**, arXiv:1312.6229.
31. Zhu, Y.; Urtasun, R.; Salakhutdinov, R.; Fidler, S. segdeepm: Exploiting segmentation and context in deep neural networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4703–4711.
32. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
33. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef] [PubMed]

34. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.
35. Zaidi, S.S.A.; Ansari, M.S.; Aslam, A.; Kanwal, N.; Asghar, M.; Lee, B. A survey of modern deep learning based object detection models. *Digit. Signal Process.* **2022**, *126*, 103514. [CrossRef]
36. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
37. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
38. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
39. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
40. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
41. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 21–37.
42. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
43. Redmon, J. Darknet: Open Source Neural Networks in C. 2013–2016. Available online: http://pjreddie.com/darknet/ (accessed on 6 June 2023).
44. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
45. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
46. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
47. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.
48. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. You only learn one representation: Unified network for multiple tasks. *arXiv* **2021**, arXiv:2105.04206.
49. Kasper-Eulaers, M.; Hahn, N.; Berger, S.; Sebulonsen, T.; Myrland, Ø.; Kummervold, P.E. Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5. *Algorithms* **2021**, *14*, 114. [CrossRef]
50. Hussain, A.; Barua, B.; Osman, A.; Abozariba, R.; Asyhari, A.T. Low Latency and Non-Intrusive Accurate Object Detection in Forests. In Proceedings of the 2021 IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA, 5–7 December 2021; pp. 1–6.
51. Chang, C.Y.; Hsieh, M.H.; Hsu, S.M. Localization of Fresh and Old Fracture in Spine CT Images Using YOLOR. In Proceedings of the 2022 IEEE International Conference on Consumer Electronics-Taiwan, Taipei, Taiwan, 6–8 July 2022; pp. 253–254.
52. Yu, J.; Zhang, W. Face mask wearing detection algorithm based on improved YOLO-v4. *Sensors* **2021**, *21*, 3263. [CrossRef]
53. Tran, V.T.; To, T.S.; Nguyen, T.N.; Tran, T.D. Safety Helmet Detection at Construction Sites Using YOLOv5 and YOLOR. In *Proceedings of the International Conference on Intelligence of Things*; Springer: Cham, Switzerland, 2022; pp. 339–347.
54. Jocher, G. Ultralytics/yolov5: v3.1—Bug Fixes and Performance Improvements. Available online: https://github.com/ultralytics/yolov5 (accessed on 6 June 2023).
55. Nepal, U.; Eslamiat, H. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors* **2022**, *22*, 464. [CrossRef]
56. Xu, R.; Lin, H.; Lu, K.; Cao, L.; Liu, Y. A Forest Fire Detection System Based on Ensemble Learning. *Forests* **2021**, *12*, 217. [CrossRef]
57. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. Scaled-yolov4: Scaling cross stage partial network. In Proceedings of the IEEE/cvf Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 13029–13038.
58. Jung, H.K.; Choi, G.S. Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions. *Appl. Sci.* **2022**, *12*, 7255. [CrossRef]
59. Goutte, C.; Gaussier, E. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In *Proceedings of the Advances in Information Retrieval*; Losada, D.E., Fernández-Luna, J.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 345–359.
60. Couceiro, M.; Portugal, D.; Ferreira, J.F.; Rocha, R. SEMFIRE: Towards a new generation of forestry maintenance multi-robot systems. In Proceedings of the 2019 IEEE/SICE International Symposium on System Integration (SII), Paris, France, 14–16 January 2019. [CrossRef]
61. Portugal, D.; Ferreira, J.F.; Couceiro, M.S. Requirements Specification and Integration Architecture for Perception in a Cooperative Team of Forestry Robots. In *Proceedings of the Towards Autonomous Robotic Systems*; Mohammad, A., Dong, X., Russo, M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 329–344.

62. Nasir, A.K.; Araújo, A.G.; Couceiro, M.S. Localization and navigation assessment of a heavy-duty field robot. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), Workshop on Perception, Planning and Mobility in Forestry Robotics (WPPMFR 2020), Las Vegas, NV, USA, 25–29 October 2020; pp. 25–29.
63. Andrada, M.; De Castro Cardoso Ferreira, J.; Portugal, D.; Couceiro, M. Testing different CNN architectures for semantic segmentation for landscaping with forestry robotics. In Proceedings of the IROS 2020 Workshop on Perception, Planning and Mobility in Forestry Robotics, Las Vegas, NV, USA, 29 October 2020.
64. Lourenço, D.; De Castro Cardoso Ferreira, J.; Portugal, D. 3D local planning for a forestry UGV based on terrain gradient and mechanical effort. In Proceedings of the IROS 2020 Workshop on Perception, Planning and Mobility in Forestry Robotics (WPPMFR 2020), Las Vegas, NV, USA, 29 October 2020.
65. Portugal, D.; Andrada, M.E.; Araújo, A.G.; Couceiro, M.S.; Ferreira, J.F. Ros integration of an instrumented bobcat t190 for the semfire project. In *Robot Operating System (ROS)*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 87–119.
66. Andrada, M.E.; Ferreira, J.F.; Portugal, D.; Couceiro, M.S. Integration of an Artificial Perception System for Identification of Live Flammable Material in Forestry Robotics. In Proceedings of the 2022 IEEE/SICE International Symposium on System Integration (SII), Narvik, Norway, 9–12 January 2022; pp. 103–108.
67. Andrada, M.E.; Ferreira, J.F.; Kantor, G.; Portugal, D.; Antunes, C.H. Model Pruning in Depth Completion CNNs for Forestry Robotics with Simulated Annealing. In Proceedings of the Innovation in Forestry Robotics: Research and Industry Adoption Workshop—IEEE Conference on Robotics and Automation (ICRA 2022), Philadelphia, PA, USA, 23–27 May 2022.
68. Bittner, D.; Ferreira, J.F.; Andrada, M.E.; Bird, J.J.; Portugal, D. Generating Synthetic Multispectral Images for Semantic Segmentation in Forestry Applications. In Proceedings of the ICRA 2022 Workshop in Innovation in Forestry Robotics: Research and Industry Adoption, Philadelphia, PA, USA, 23–27 May 2022.