applied sciences

MDPI

Article

Internet of Things Meets Computer Vision to Make an Intelligent Pest Monitoring Network

Bruno Cardoso [1,*], Catarina Silva [1,*], Joana Costa [1,2] and Bernardete Ribeiro [1]

[1] Center for Informatics and Systems (CISUC), Department of Informatics Engineering, University of Coimbra, 3004-531 Coimbra, Portugal
[2] Polytechnic Institute of Leiria, School of Technology and Management, 2411-901 Leiria, Portugal
* Correspondence: badscc@student.dei.uc.pt (B.C.); catarina@dei.uc.pt (C.S.)

Abstract: With the increase of smart farming in the agricultural sector, farmers have better control over the entire production cycle, notably in terms of pest monitoring. In fact, pest monitoring has gained significant importance, since the excessive use of pesticides can lead to great damage to crops, substantial environmental impact, and unnecessary costs both in material and manpower. Despite the potential of new technologies, pest monitoring is still done in a traditional way, leading to excessive costs, lack of precision, and excessive use of human labour. In this paper, we present an Internet of Things (IoT) network combined with intelligent Computer Vision (CV) techniques to improve pest monitoring. First, we propose to use low-cost cameras at the edge that capture images of pest traps and send them to the cloud. Second, we use deep neural models, notably R-CNN and YOLO models, to detect the Whitefly (WF) pest in yellow sticky traps. Finally, the predicted number of WF is analysed over time and results are accessible to farmers through a mobile app that allows them to visualise the pest in each specific field. The contribution is to make pest monitoring autonomous, cheaper, data-driven, and precise. Results demonstrate that, by combining IoT, CV technology, and deep models, it is possible to enhance pest monitoring.

Keywords: computer vision; pest monitoring; Internet of Things; smart farming; deep learning

Citation: Cardoso, B.; Silva, C.; Costa, J.; Ribeiro, B. Internet of Things Meets Computer Vision to Make an Intelligent Pest Monitoring Network. *Appl. Sci.* **2022**, *12*, 9397. https://doi.org/10.3390/app12189397

Academic Editors: Paulo Pedreiras, Pedro Gonçalves and António Monteiro

Received: 17 August 2022
Accepted: 13 September 2022
Published: 19 September 2022

1. Introduction

The Internet of Things (IoT) describes a network composed of several interconnected devices to obtain, transmit, and store data. The use of smart farming has grown exponentially with the use of IoT to produce better and cheaper food for a growing world population. In smart farming, data can be used, among other things, to make more efficient use of space, minimise waste, and monitor pests [1]. When data is in the form of images, the IoT can be used together with Computer Vision (CV), which is a field of research that enables machines to process, analyse, and obtain insights from images or videos. In this case, the IoT network obtains, transmits, and stores images, and the CV models process those images to obtain information or insights [2–4].

Smart farming uses IoT for several purposes, e.g., in irrigation systems to minimise water consumption or in animal feeding to optimise feeding [1]. Computer Vision (CV) is also used with different purposes, e.g., in crop monitoring to obtain insights about the best time to harvest or in animal counting [5,6].

Despite the potential of combining IoT and CV, these technologies are still not very commonly used in traditional pest monitoring, which is a visual task and one of the biggest challenges for farmers [7]. In traditional pest monitoring, technicians have to go to the field for the sake of identifying and counting the pests in traps. If the number of pests exceeds a threshold, then they need to use pesticides or biological monitoring. This may not be very precise, as technicians use extrapolation to know the number of pests, i.e., the pests in a small area are counted and used as a sample. It is also expensive, because it requires many hours of specialised work to count the pests and it can also lead to the overuse of pesticides if the visit to the field occurs too late, allowing the pests to propagate [8].

In this article. we present a system that was developed by combining IoT and CV to make an intelligent pest monitoring network. This network contributes to make the pest monitoring:

- **Autonomous:** almost does not need human intervention;
- **Cheaper:** reduces the need for technicians to identify and count the pests;
- **Data-driven:** allows to know the daily pest evolution using images and not only when the technicians make pest detection and counts;
- **Precise:** replaces the sampling by the total count of the pests.

These contributions allow farmers to optimise production quality and increase revenue. With that in mind, the proposed system can be implemented for all types of crops and farms (indoor and outdoor) to allow farmers to make informed decisions in an easy way.

For the implementation, we designed trapezoidal plastic structures to protect the traps and cameras for environmental conditions and to control the obtained images quality. At the base of these structures, we placed yellow sticky traps and, on the top, the cameras used as edge computing. The structures were placed in different crops and the obtained images from the traps were sent to the cloud. The cloud has a CV model trained to detect and count pests. The information resulting from the model is then processed and sent to an application. The application allows farmers to easily make an informed decision, such as when to use pesticides, just by looking at a dashboard with the evolution of pests in each trap. We used low-cost/low-power cameras and solar panels for power, which allows for long-term monitoring in environments without an external power supply. The complex calculations are in the cloud, where all backend tasks/services are centralised.

The rest of the article is structured as follows. Section 2 presents material and methods, describing the types of computation, communication protocols, network types, and CV models. Section 3 describes the proposed approach, including the dataset and CV model used, and the proposed IoT network. In Section 4, we present the conclusions.

2. Material and Methods

In pest monitoring, most approaches focus only on CV techniques [9,10], not considering the process to obtain images automatically and in large numbers, which hampers the application of CV models in real contexts due to lack of information/images. Works that present the two technologies together, CV and IoT, in pest monitoring, are therefore the most promising [11,12], since the images are obtained in a more real context and not under extremely controlled conditions, which could raise issues with the generalisation ability of the CV models.

In the next subsections, we first introduce the most relevant aspects of the IoT, such as computation types, communications protocols, and network topologies; and then the most relevant CV models, such as faster Region-Convolution Neural Network (R-CNN) and You Only Look Once (YOLO).

2.1. Internet of Things

The IoT is a network composed of interconnected nodes/devices that send, receive, and store data. Those nodes can use edge, fog, and cloud computing (see Figure 1). The communication protocols define the communication rules between nodes, and the network topology defines the nodes' distribution and communication links [13].

Depending on the purpose, environmental, hardware or communication constraints, IoT networks can use different types of computation, communication protocols, and network topology. Computation types mainly influence computational power, storage capacity, and latency. Communication protocols mainly influence the security and payload size of communications. The network topology mainly influences the reliability and scalability of the network [13].

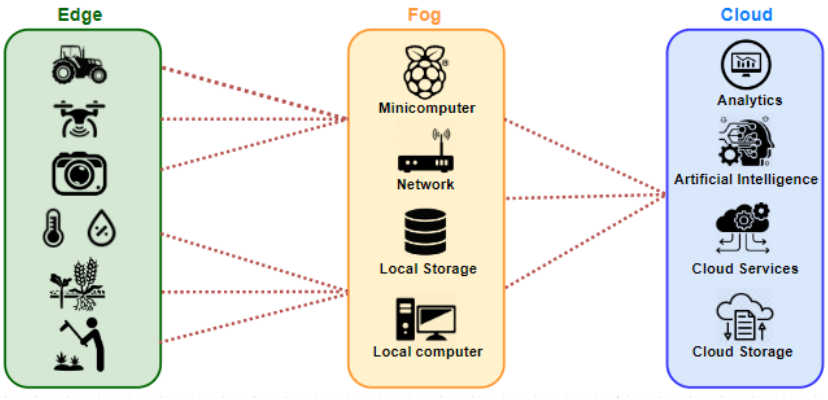

Figure 1. Types of computation (adapted from [14]).

2.1.1. Computation Types

As depicted in Figure 1, edge, fog, and cloud are interconnected computation types that should cooperate to fulfil the final service [14].

Edge computing is the computing performed on devices used to collect the data, i.e., devices that are closer to data sources. The devices used at the edge usually have low computational power and low storage capacity, which makes it difficult to store or process large amounts of data at the edge. Furthermore, the edge has low latency, mobility of its devices (like the fog), and low bandwidth available [13,15]. Examples of edge devices include low-cost cameras or humidity sensors.

Fog computing is the computing used to bring data storage and processing closer to the edge, enhancing the edge ability. The fog can have several devices with different computational and storage resources, which allows it to store or process data locally and almost in real-time. Normally, the fog is a bridge between edge and cloud computing that allows the reduction of the resources used by the cloud, while enhancing edge capabilities [13]. Examples of fog devices include Raspberry Pi and Arduino.

Cloud computing allows us to make more demanding calculations that are difficult at the edge or fog. This type of computation is efficient, scalable, and allows the tailoring of resources as needed [13]. It has disadvantages such as high latency, high bandwidth, and high energy consumption. This type of computing performs the data processing and analysis in the same place, usually in a centralised way. The types of cloud are public, community, hybrid, or private [13,15].

As the IoT is a network composed of several interconnected devices, it is necessary to choose the most appropriate communication protocols to communicate between nodes in each scenario.

2.1.2. Communication Protocols

Communication protocols define the rules to communicate between nodes, allowing them to send data between devices and users in a safe and reliable way. We can divide communication protocols into two types: (1) request–response and (2) publication–subscription.

The most used protocol based on request–response is the Hypertext Transfer Protocol (HTTP). This is one of the most stable and reliable protocols and allows the transmission of large amounts of data. Its disadvantages are high memory and energy consumptions, making it challenging to use at the edge.

The most used protocol based on publication–subscription, is Message Queuing Telemetry Transport (MQTT), which is a lightweight protocol that allows scalability and simplifies communication between devices (despite not having an encryption mechanism). It is suitable for sending small amounts of data in low bandwidth and high latency scenarios typical on the edge. It is also optimised to work on devices with memory constraints and has other advantages, namely in terms of privacy issues. Publishers and subscribers do not need to know each other's existence. One subscriber can receive data from many devices, and the publisher and subscriber do not need to be connected at the same time [16,17].

After choosing the types of computation and communication protocols, it is necessary to define how to arrange the nodes of the IoT network and how to connect these nodes. This arrangement of nodes and how they communicate with each other defines the network topology.

2.1.3. Network Topology

An IoT network is a representation of the distribution of its nodes in space and how they interconnect. The most used topologies are point-to-point, star, and mesh, as depicted in Figure 2 [18].

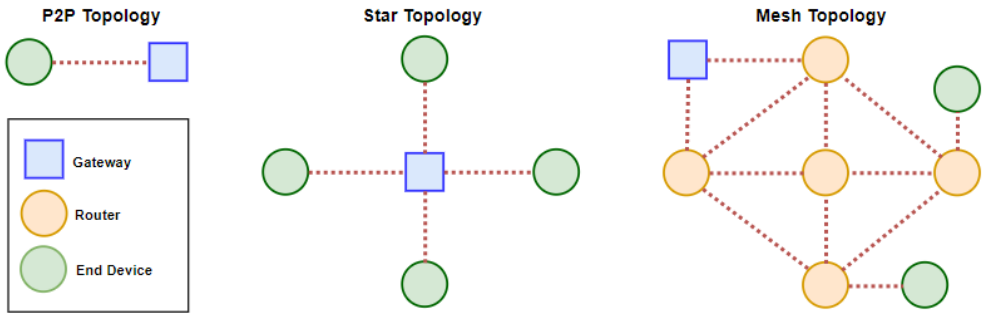

Figure 2. Network topology (adapted from [18]).

Point-to-point (P2P) is the simplest topology and has only two nodes. For this topology to work, none of the elements (the two nodes and the link between them) can fail, which is a tough constraint.

The star topology is one of the most common, since it is scalable and easy to deploy. In this topology, all nodes are connected to a central device that acts as a gateway. The nodes cannot communicate with each other, but only with the gateway. It is possible to add and remove nodes from the network without breaking it. It has the disadvantage of having a single point of failure (the gateway).

The mesh topology is not as common as might be assumed and is more expensive to deploy. All nodes are connected to each other, which causes redundancy in the connections. It also allows the network to work in case of a connection failure (because of the redundancy). Nevertheless, it needs more computational power and more memory than the star topology to move data from one node to the other [19,20].

2.2. Computer Vision

Computer Vision (CV) is an interdisciplinary research area that processes digital images or videos to extract information [21]. Currently, CV is strongly based on artificial intelligence, namely, on machine learning and deep learning models that replace human-visual tasks, such as animal counting and crop or pest monitoring. In the specific context of this work, we can divide CV into image classification and object detection. Typically, image classification is used to classify an image, but it can be adapted to classify multiple objects in the same image. For that, we have to segment the objects in the image, extract their most relevant features, and then apply the classification model to each object. Object detection models incorporate all those steps into a single model and allow for the classification of multiple objects in the same image.

In pest monitoring, the goal is usually to detect and count the pests in each trap, and not to holistically classify the trap; hence, object detection models are usually more suitable. There are two main types of object detection models: (1) two-phase models, such as the Faster Region-Convolution Neural Network (Faster R-CNN) and (2) one-phase models, such as You Only Look Once (YOLO) [22].

2.2.1. Faster R-CNN

Faster R-CNN is a two-phase model because it includes two phases: first, it looks for the Regions of Interest (RoI) to predict if an object exists; then, using that RoI, it predicts the object's class and coordinates. This model has three main steps. The backbone is where a Feature Pyramid Network (FPN) obtains the feature maps of the images. The Region Proposal Network (RPN) predicts the RoI, that is, the regions that have objects and their coordinates. Lastly, there are the RoI classifier and the bounding box regressor. The first classifies each object in the bounding boxes (regions that have an object). The latter improves predictions for the dimensions of those boxes [23] (see Figure 3).

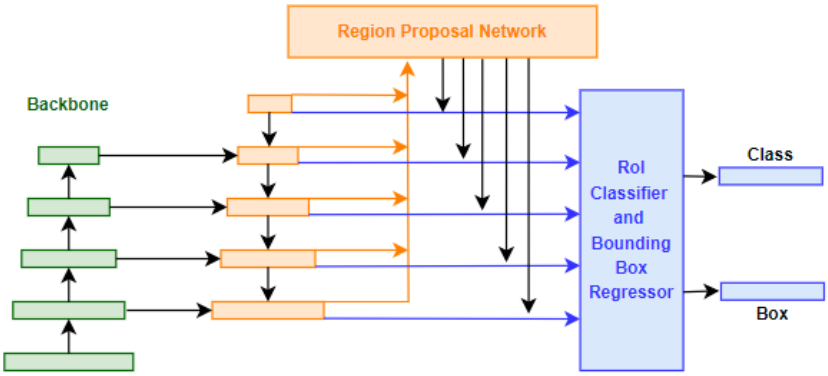

Figure 3. Faster R-CNN general architecture (adapted from [24]).

2.2.2. YOLOv5

YOLOv5 is one of the most recent YOLO models and like the other YOLO models, is a one-phase model because it only takes one look at the images, i.e., this model does not first look for the RoI and then process that area. Instead, it makes the classification and prediction all at once. It has three main steps. First, there is a backbone where a Cross Stage Partial Network (CSPN) extracts the most relevant features. Then, a neck generates feature pyramids using a Path Aggregation Network (PANet) to obtain new and better feature maps, i.e., features maps that combine the same information with different scales. The final step is the head (see Figure 4), which is used to acquire the bounding boxes from the feature maps, classify them, and predict their coordinates [25].

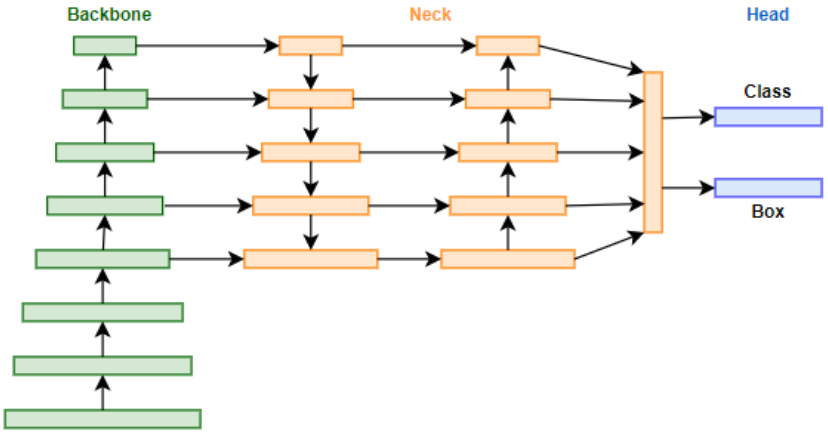

Figure 4. YOLOv5 general architecture (adapted from [26]).

3. Proposed Approach

The main goal of this work is to propose and develop an IoT network that uses CV for pest monitoring. We start by constructing and validating the use of CV models for pest detection. Then, we define the type of devices and structures to capture images at the edge of the network. Next, we use the cloud to centralise the backend of the IoT

network. This is where we store and process data and connect the edge and users of the network. Then, we develop and test a mobile application for users to easily monitor the pest's evolution. Finally, we connect all network nodes (edge, cloud, and application) and place the intelligent traps in different crops to test the whole proposed approach.

3.1. Computer Vision Model

We have developed CV models to detect the Whitefly (WF) pest. This type of pest is one of the most common; thus, it is one of the most relevant to fight, especially in tomato crops. The life cycle of WF starts as an egg, then passes on to nymph and ends up as an adult propagating to the entire crop. There are several species of whiteflies that are visually similar, such as *Trialeurodes* vaporariorum and *Bemisia tabaci*. The next subsections describe the dataset used to train several CV models (object detection models), the obtained results, and the chosen CV model for the IoT network.

3.1.1. Dataset

To train the computer vision models, we used a public dataset [27] that contains 284 images (5184 × 3456) of yellow sticky traps with WF and other insects (Figure 5). The dataset contains 4940 annotated WF instances and, as described by the dataset authors [27], some unannotated or improperly annotated instances. This affects the precision of CV models, as the detection of an unannotated instance is classified as a False Positive detection. To improve the dataset, we contacted technicians and, with their help, we annotated the missing WF instances. From that process, the total annotations passed from 4940 to 5863 WF instances. Finally, we divided the whole dataset into a training (80%) and a testing (20%) dataset.

Figure 5. Part of a dataset image with WFs.

3.1.2. Model

We used the train dataset to train the Faster R-CNN, Scaled-YOLOv4, YOLOv5 (Small), and YOLOv5 (XLarge) models. The Faster R-CNN was published in 2017 [28], the YOLOv5 models were made public in June 2020 [29], and the Scaled-YOLOv4 (Large) in November 2020 [30]. YOLO models allow testing of different depths of the same architecture, from Tiny to XLarge (changing only the number of layers and neurons). This allows testing the state of the art of object detection and evaluating the trade-off between performance and depth of the network. The R-CNN allows the testing of a two-stage model and compares performance with one stage models.

To evaluate those object detection models, we considered three metrics. Mean Average Precision (mAP) is one of the most used metrics to assess the precision of object detection models [31]. We also considered the metrics memory consumption and average detection time for an image (using a NVIDIA Tesla T4 GPU).

As described in Table 1, YOLOv5 (XLarge) presented the best performance (best mAP) in the test dataset and therefore was the model deployed in the IoT network. The memory consumption was not an issue because we used the cloud, and average time for each detection was not relevant because we did not have the goal of performing real-time detection.

Table 1. Computer Vision models performance.

Metric	Faster R-CNN	YOLOv5 (Small)	YOLOv5 (XLarge)	Scaled-YOLOv4 (Large)
mAP (%)	76.15	81.20	**89.70**	82.50
Memory (MB)	244	**14**	167	101
Average Time (S)	1.5	**0.01**	0.13	0.03

3.2. IoT Network

As depicted in Figure 6, we developed a star topology network with cameras (at the edge) placed in crops, the backend centralised in a private cloud, a pest monitoring application for the users obtain insights, and all communications using the HTTP protocol. The chosen architecture (star topology) allowed the development of a very efficient and inexpensive network. Furthermore, this type of topology allows us to easily scale the network to other crops just by adding new edge devices. The cameras used at the edge (ESP32-CAM), although low cost, allow the obtaining of images with enough quality to not influence the performance of the CV models.

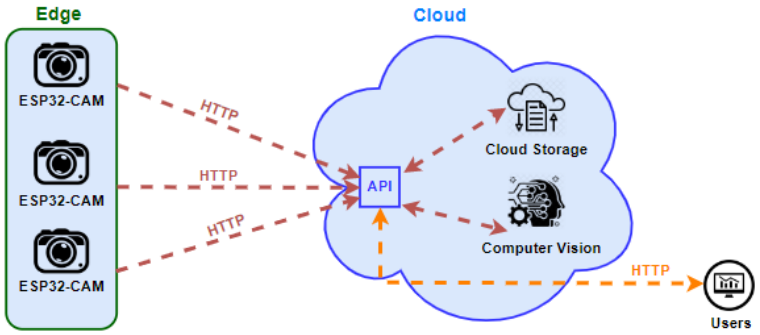

Figure 6. Proposed IoT network.

If we had used the fog layer, the cost would have increased significantly; each crop would need a fog device and solar panels to power them. The goal of this system is not to obtain images in real-time, so the lower latency of the fog (over the cloud) was not a decision factor. Another advantage of fog that was not important to us was the mobility of its devices, as we wanted to leave the devices in the same location for a long time (the most autonomous possible). However, introducing a fog layer could be a good solution for cultures without internet access or if the backend is on a public cloud.

The chosen communication protocol (HTTP) allowed us to send the images to the cloud without the payload size restriction that MQTT sometimes has when dealing with images. HTTP power consumption was not an issue because each edge node sent only a daily image to the cloud to perform daily monitoring). HTTP memory consumption was not a constraint, as we did not send high resolution images. However, the use of MQTT would result in a lower memory and energy consumption, but as this protocol was developed to send small amounts of data, the size of the images would have to be reduced to have viable transfers. As MQTT was designed to be lightweight, it does not use a security mechanism for data transmission. In that case, we would have to implement, for instance, Transport Layer Security (TLS), to guarantee the encryption of the transmitted images.

Using the application for automatic monitoring, the user only needs to visualise the graph with the daily evolution of the pest in order to choose which intervention to take. Nevertheless, this limits the monitoring to available nodes at the edge. With that in mind, we introduced a feature for the user to use their phone's camera to take a picture of a trap. With that, he can point the camera at any yellow sticky trap and is not limited to the ones

at the edge. However, this changes how images are obtained and introduces the possibility of human error, which can influence the performance of CV models.

The next subsections describe the edge and cloud of the IoT network and show part of the application.

3.2.1. Edge

The number of edge nodes to use on each farm mainly depends on the size of the farm. With more nodes, it is possible to know more quickly and reliably how the number of pests is evolving. However, as the pests are attracted by the traps inside each structure, just one node is enough to see the daily evolution of the pests.

For each farm, we used one ESP32 microcontroller with a camera attached to obtain the trap images. This is a low-cost, low-power microcontroller suitable for when there is no external power source. It has only 520 kilobytes (KB) of memory, with 320 KB of DRAM (for storage) and 192 KB of IRAM (for instruction execution); the rest is RTC memory (which persists when the device is on standby) [32].

We attached the OV2640 camera to the ESP32 to create the ESP32-CAM. This camera allows the obtaining of images of different qualities (jpeg quality) and resolutions (320 × 240, 352 × 288, 640 × 480, 800 × 600, 1024 × 768, 1280 × 1024 and 1600 × 1200).Finally, we programmed the ESP32-CAM to send daily images (1280 × 1024) to the cloud through the HTTP method [33].

To test if the ESP-Cam had the capacity to detect pests, we counted the WF in the testing dataset (we counted 1371 WF) and then resized those pictures to different dimensions and sizes that are allowed by the Esp32-Cam. Next, we used the YOLOv5 (XLarge) model to make detections on those pictures, as shown in Figure 7.

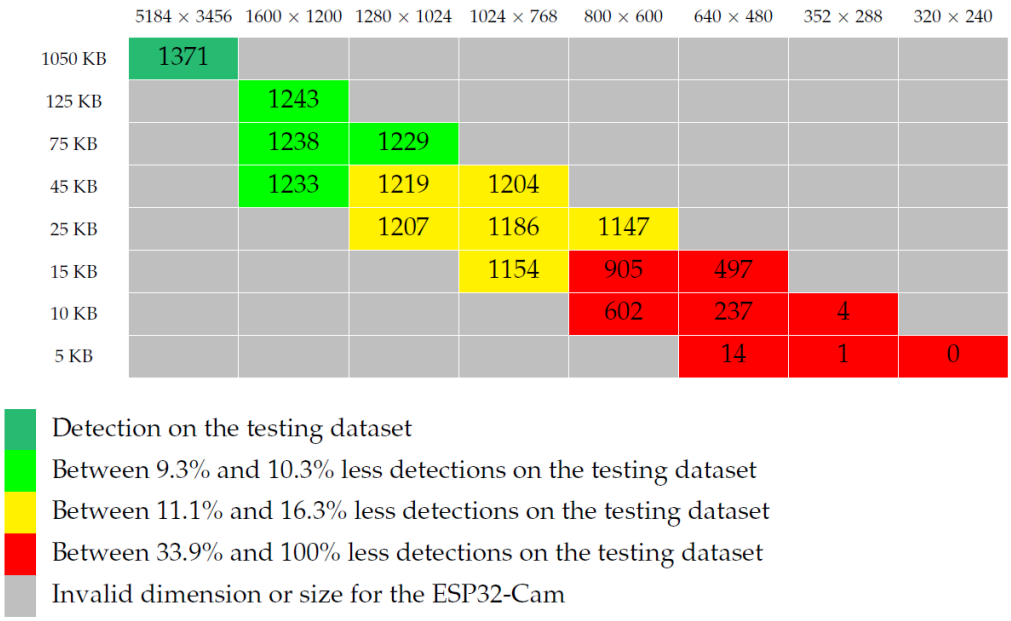

Figure 7. YOLOv5 (XLarge) performance (number of whitefly detections) for different image dimensions and sizes.

The previous table shows that it is possible to use the Esp32-Cam with 1280 × 1024 images for pest detection without losing significant performance in YOLOv5 (XLarge) (between 10.3% and 16.3% less detections).

To power the ESP32-CAM, we used two 18650 batteries and also solar panels to extend the battery life (see Figure 8). The daily power consumption of each edge node is 24 mA and the used batteries have 3350 mAh of power capacity, providing more than four months of duration for each battery. The solar panels with 150 mA and an average of 5 h of direct

sun exposure during the year (in Coimbra, where these edge nodes are placed) allow the edge nodes to run without the need to change batteries.

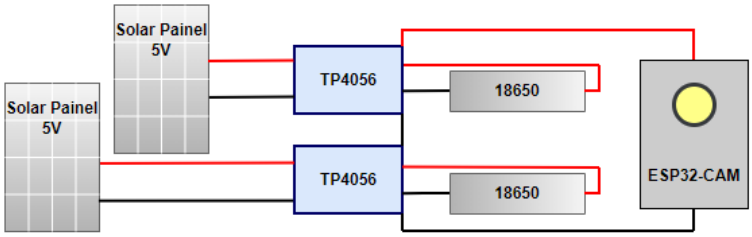

Figure 8. Diagram of ESP32-CAM powered by solar panels.

In crops, the edge is subject to adverse conditions that can affect the durability of the devices and the quality of the images. With that in mind, we designed a structure (see Figure 9 in this paper and Figure A1 in the Appendix A) to protect the ESP32-CAM from atmospheric conditions and to better control the image quality (such as brightness). At the base of this structure, we placed the yellow sticky trap, and the ESP32-CAM at the top.

Figure 9. Trap structures at the edge.

3.2.2. Cloud

The backend part of the IoT network is in the cloud, which is available through an Application Programming Interface (API) that makes the images available to the CV model (YOLOv5 (XLarge)) and stores them. When an image arrives at the cloud, the model detects the pests and counts them (see Figure 10 in this paper and Figure A2 in the Appendix A). These discoveries, the metadata of the images, and plantation sites are stored in a relational database. The pest monitoring application, through the same API, can access the CV models, the database, and the stored images in the cloud.

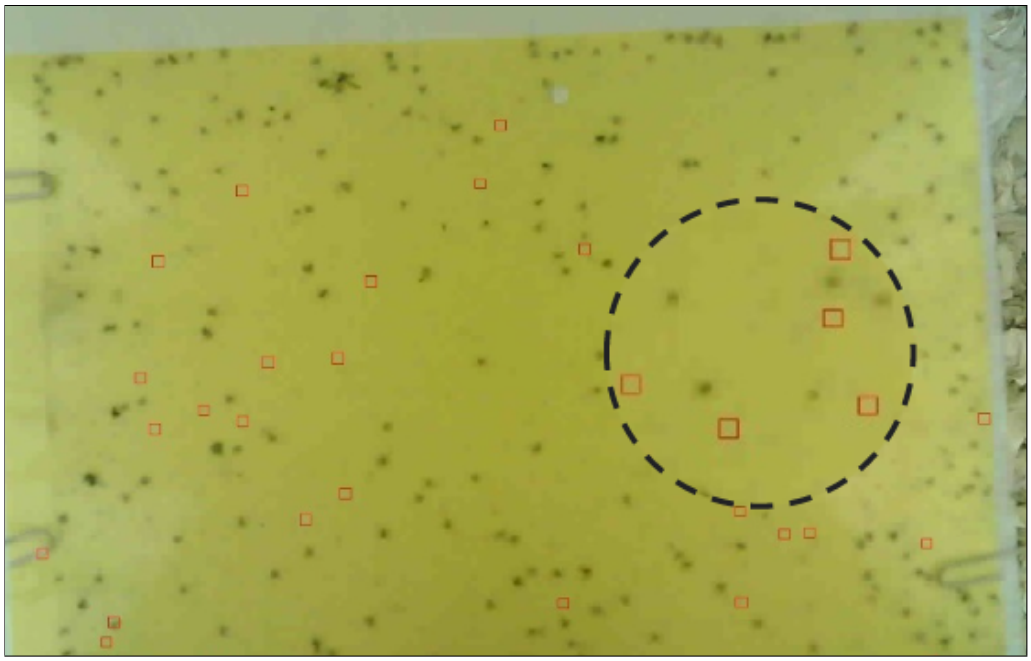

Figure 10. Example of a daily pest detection by the CV model.

3.2.3. Mobile Application

The mobile application was developed using React Native, and allows the user to monitor pests through a mobile device (see Figure 11). For that, the user can choose a plantation site and a specific edge node (one of the ESP32-CAMs) on a map. For that node, the user can see the evolution of the detected pests and the images obtained for each day. The middle image shows a pest evolution where the traps were changed almost every day. To complement this approach, it is also possible to take a photo of a trap, which is sent to the backend part of the IoT network. Then, the CV model makes the detection and returns results to the application.

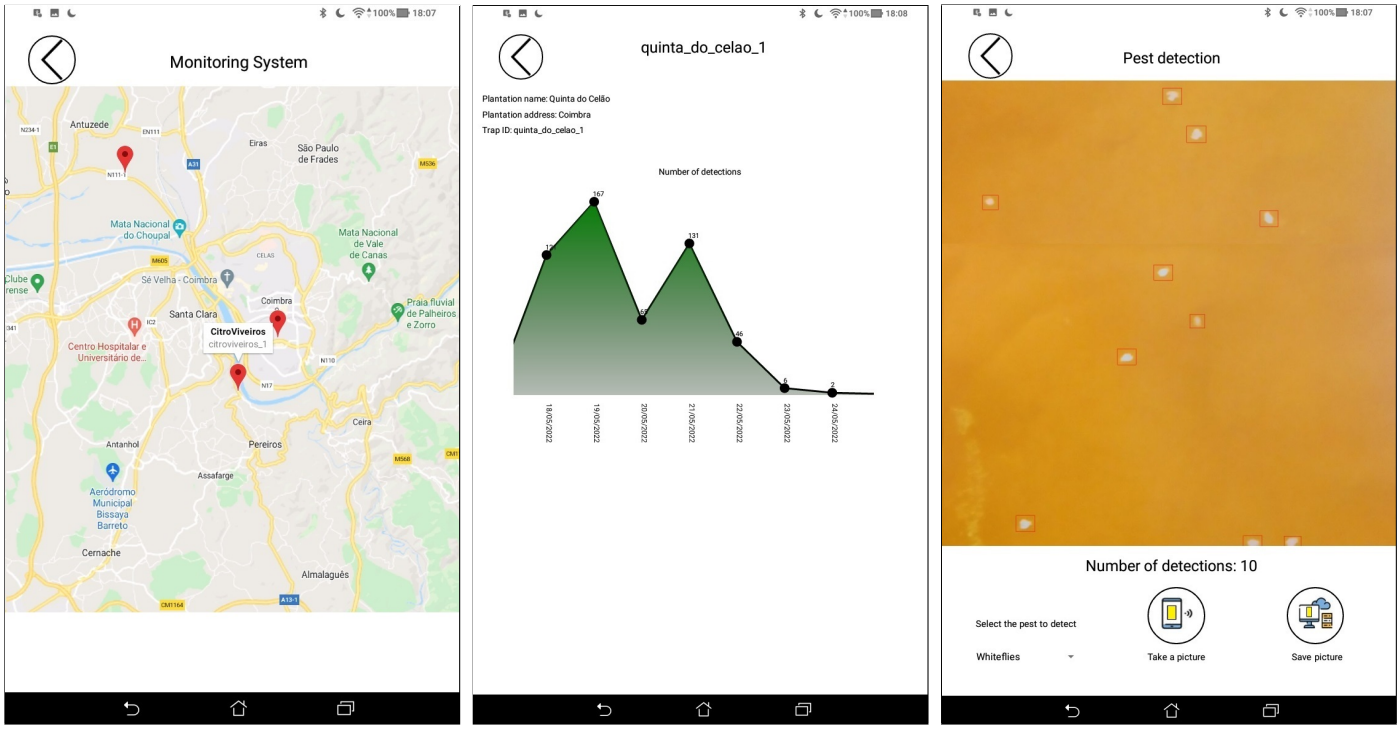

Figure 11. Part of the mobile app that allows users to monitor pests.

4. Conclusions and Future Work

The presented approach aims to contribute to more sustainable and efficient farming. We presented an IoT network to automate pest monitoring and to enhance part of the technicians work with new technologies. The star topology of the network allows its easy extension to new plantations and other pest types. This network is composed of edge devices, a private cloud, and a mobile application, and uses HTTP to communicate between nodes. The low-cost devices (such as the ESP32-Cam) are associated with low-power consumption, allowing the use of the network in crops with external power constraints. It is possible to run the edge nodes for four months using 18650 batteries, and without the need to change those batteries with the addition of solar panels. Furthermore, the use of low-cost devices at the edge does not significantly decrease the performance of CV models (between 10.3% and 16.3% less detections).

We tested four CV models (Faster R-CNN, Scaled-YOLOv4, YOLOv5 (Small), and YOLOv5 (XLarge)) and used three metrics for their evaluation: mAP, memory consumption, and average time for each detection. The model with the best mAP was YOLOv5 (XLarge), with a mAP of 89.70%. The model with the lowest memory consumption and average time for each detection was YOLOv5 (Small), with 15 MB and 0.01S, respectively.

As the model runs in the cloud and the goal is to monitor pests daily, we used the model with the highest mAP (YOLOv5 (XLarge)) for the monitoring system. For a real-time monitoring system, the model with the lowest memory consumption (to be closer to the edge) and the lowest average time for each detection was determined to be the best option, in this case YOLOv5 (Small). For a monitoring system with a CV model in the fog, the Scaled-YOLOv4 constitutes the best choice, as it is more balanced.

The mobile application allows users to easily monitor the pest's evolution, thus supporting more informed decisions. The application was presented to potential users that have shown interest in such a pest monitoring application, namely to integrate in their own current monitoring system.

Most approaches on pest monitoring only address CV models. Advances in IoT, such as reducing hardware costs and increasing computational power, make it especially relevant to use IoT networks for the deployment of those CV models, giving them a purpose and a use. Additionally, most proposed approaches use images obtained in controlled conditions, e.g., always the same light or without noise, making it difficult to develop CV models with good performance outside the lab. This work shows that the use of an IoT network allows the acquiring of images in real contexts and in large quantities, which can then contribute to improve the real performance of CV models.

Despite the advantages of an automatic monitoring system, there are some disadvantages that must be taken into account. The lack of precision of the CV model or the lack of the quality of the images can lead to false positives and/or false negatives, both of which can be tackled by large annotated datasets throughout the pest's different development stages obtained under real conditions.

Future work will mostly focus on overcoming these disadvantages. For that, we will focus on the expansion of the IoT network to obtain more images and the creation of a dataset with ESP32-Cam images to improve CV models' performance under uncontrolled conditions. Lastly, we will evaluate performance of the monitoring system during multiple crop seasons.

Author Contributions: Conceptualisation, C.S., J.C. and B.R.; methodology, C.S., J.C. and B.R.; software, B.C.; validation, C.S., J.C., B.R. and B.C.; formal analysis, C.S., J.C., B.R. and B.C.; investigation, C.S., J.C., B.R. and B.C.; resources, C.S., J.C., B.R. and B.C.; data curation, C.S., J.C., B.R. and B.C.; writing—original draft preparation, C.S., J.C., B.R. and B.C.; writing—review and editing, C.S., J.C., B.R. and B.C.; visualisation, C.S., J.C., B.R. and B.C.; supervision, C.S., J.C. and B.R.; project administration, C.S., J.C. and B.R.; funding acquisition, C.S., J.C. and B.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the FCT—Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit—UIDB/00326/2020 or project code UIDP/00326/2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to train the CV models was obtained from https://research.wur.nl/en/publications/detection-and-classification-of-insects-on-stick-traps-in-a-tomat/datasets/ (accessed on 19 July 2020) and was made public by [27].

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

Abbreviations

The following abbreviations are used in this manuscript:

AP	Average Precision
API	Application Programming Interface
CSPN	Cross Stage Partial Network
DRAM	Dynamic Random Access Memory
FPN	Feature Pyramid Network
HTTP	Hypertext Transfer Protocol
IRAM	Instruction Random Access Memory
KB	Kilobyte
mAP	mean Average Precision
MQTT	Message Queuing Telemetry Transport
P2P	Point-to-Point
R-CNN	Region-Convolution Neural Network
RoI	Regions of Interest
RPN	Region Proposal Network
RTC	Real-Time Clock
TLS	Transport Layer Security
WF	Whitefly
YOLO	You Only Look Once

Appendix A

Figure A1. More examples of trap structures at the edge.

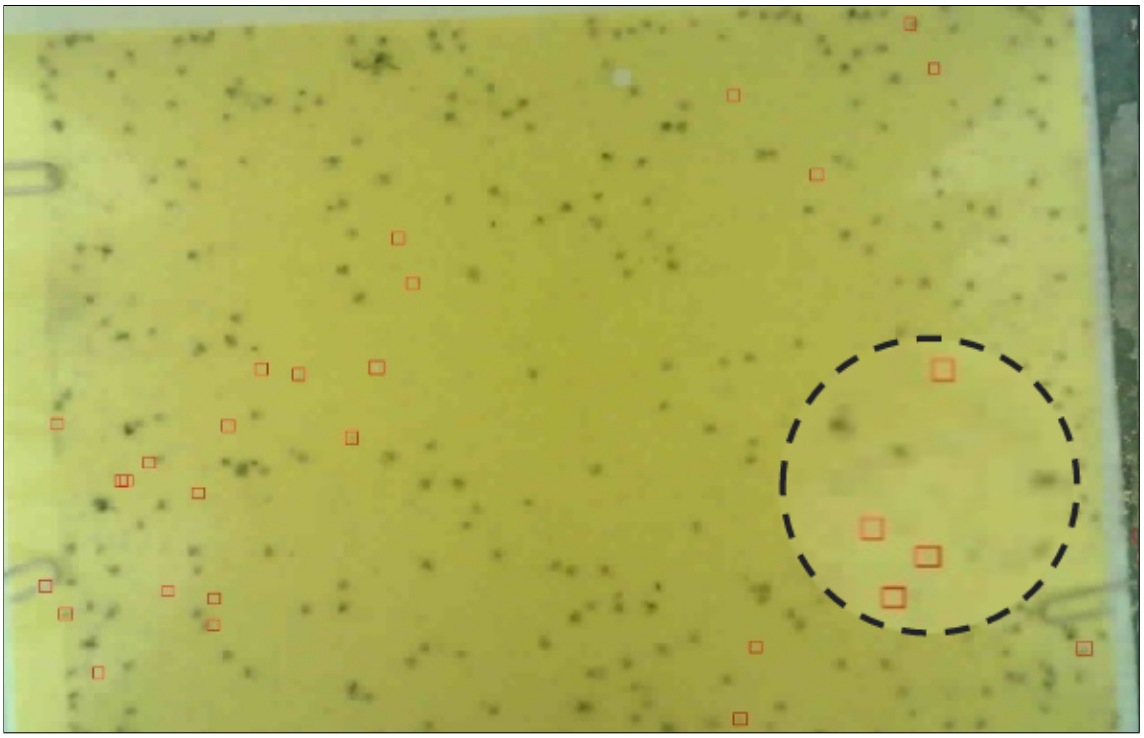

Figure A2. Another example of a daily pest detection by the CV model.

References

1. Sharma, A.; Jain, A.; Gupta, P.; Chowdary, V. Machine Learning Applications for Precision Agriculture: A Comprehensive Review. *IEEE Access* **2021**, *9*, 4843–4873. [CrossRef]
2. Arshad, B.; Ogie, R.; Barthelemy, J.; Pradhan, B.; Verstaevel, N.; Perez, P. Computer Vision and IoT-Based Sensors in Flood Monitoring and Mapping: A Systematic Review. *Sensors* **2019**, *19*, 5012. [CrossRef] [PubMed]
3. Heidari, A.; Jabraeil Jamali, M.A.; Jafari Navimipour, N.; Akbarpour, S. Deep Q-Learning Technique for Offloading Offline/Online Computation in Blockchain-Enabled Green IoT-Edge Scenarios. *Appl. Sci.* **2022**, *12*, 8232. [CrossRef]
4. Ullah, U.; Bhatti, F.A.; Maud, A.R.; Asim, M.I.; Khurshid, K.; Maqsood, M. IoT-enabled computer vision-based parts inspection system for SME 4.0. *Microprocess. Microsyst.* **2021**, *87*, 104354. [CrossRef]
5. Patrício, D.; Rieder, R. Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review. *Comput. Electron. Agric.* **2018**, *153*, 69–81. [CrossRef]
6. Norouzzadeh, M.S.; Morris, D.; Beery, S.; Joshi, N.; Jojic, N.; Clune, J. A deep active learning system for species identification and counting in camera trap images. *arXiv* **2019**, arXiv:1910.09716.
7. Barbedo, J. Detecting and Classifying Pests in Crops Using Proximal Images and Machine Learning: A Review. *Atificial Intell.* **2020**, *1*, 312–328. [CrossRef]
8. Preti, M.; Verheggen, F.; Angeli, S. Insect pest monitoring with camera-equipped traps: Strengths and limitations. *J. Pest Sci.* **2020**, *94*, 203–217. [CrossRef]
9. Sukju, H. Automatic pest counting from pheromone trap images using deep learning object detectors for matsucoccus thunbergianae monitoring. *Insects* **2021**, *12*, 342.
10. Liu, L.; Wang, R.; Xie, C.; Yang, P. PestNet: An end-to-end deep learning approach for large-scale multi-class pest detection and classification. *IEEE Access* **2019**, *7*, 45301–45312. [CrossRef]
11. He, Y.; Zeng, H.; Fan, Y.; Ji, S.; Wu, J. Application of Deep Learning in Integrated Pest Management: A Real-Time System for Detection and Diagnosis of Oilseed Rape Pests. *Mob. Inf. Syst.* **2019**, *2019*, 1–14. [CrossRef]
12. Partel, V.; Nunes, L.; Stansly, P.; Ampatzidis, Y. Automated vision-based system for monitoring Asian citrus psyllid in orchards utilizing artificial intelligence. *Comput. Electron. Agric.* **2019**, *162*, 328–336. [CrossRef]
13. Motlagh, N.H.; Mohammadrezaei, M.; Hunt, J.; Zakeri, B. Internet of Things (IoT) and the Energy Sector. *Energies* **2020**, *13*, 494. [CrossRef]
14. Cao, H.; Wachowicz, M.; Renso, C.; Carlini, E. Analytics everywhere: Generating insights from the Internet of Things. *IEEE Access* **2019**, *7*, 71749–71769. [CrossRef]
15. Farooq M.S.; Riaz S.; Abid A.; Umer T.; Zikria Y.B. Role of iot technology in agriculture: A systematic literature review. *Eletronics* **2020**, *9*, 319. [CrossRef]
16. Balaji, S.; Nathani, K.; Santhakumar, R. IoT technology, applications and challenges: A contemporary survey. *Wirel. Pers. Commun.* **2019**, *108*, 363–388. [CrossRef]

17. Babun, L.; Denney, K.; Celik, Z.B.; McDaniel, P.; Uluagac, A.S. A survey on IoT platforms: Communication, security, and privacy perspectives. *Comput. Netw.* **2021**, *192*, 108040. [CrossRef]
18. Ekanayake, J.C.; Hedley, C.B. Advances in information provision from wireless sensor networks for irrigated crops. *Wirel. Sens. Netw.* **2018**, *10*, 71–92. [CrossRef]
19. Mukherjee, A.; Misra, S.; Sukrutha, A.; Raghuwanshi, N.S. Distributed aerial processing for IoT-based edge UAV swarms in smart farming. *Comput. Netw.* **2020**, *167*, 107038. [CrossRef]
20. Callebaut, G.; Van der Perre, L. Characterization of LoRa Point-to-Point Path Loss: Measurement Campaigns and Modeling Considering Censored Data. *IEEE Internet Things J.* **2020**, *7*, 1910–1918. [CrossRef]
21. Forsyth, D.A.; Ponce, J. Computer Vision: A Modern Approach. *Prentice Hall Professional Technical Reference*. 2002. Available online: https://dl.acm.org/doi/abs/10.5555/580035 (accessed on 16 August 2022).
22. Wu, Y.; Chen, Y.; Yuan, L.; Liu, Z.; Wang, L.; Li, H.; Fu, Y. Rethinking classification and localization for object detection. *arXiv* **2019**, arXiv:1904.06493.
23. Du, J. Understanding of object detection based on CNN family and YOLO. *J. Phys. Conf. Ser.* **2018**, *1004*, 012029. [CrossRef]
24. Guo, P.; Xue, Z.; Long, L.R.; Antani, S. Cross-Dataset Evaluation of Deep Learning Networks for Uterine Cervix Segmentation. *Diagnostics* **2020**, *10*, 44. [CrossRef]
25. 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.
26. Nepal, U.; Eslamiat, H. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors* **2022**, *22*, 464. [CrossRef] [PubMed]
27. Nieuwenhuizen, A.; Hemming, J.; Suh, H. Detection and classification of insects on stick-traps in a tomato crop using Faster R-CNN. In Proceedings of the The Netherlands Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2018.
28. He, K.; Gkioxari, G.; Doll´ar, P.; Girshick, R. Mask R-CNN. In Proceedings of the International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
29. Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; NanoCode012. *ultralytics/yolov5*, v6.1; TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference; Zenodo: Geneva, Switzerland, 22 February 2022. [CrossRef]
30. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y. ScaledYOLOv4: Scaling Cross Stage Partial Network. *arXiv* **2020**, arXiv:2011.08036.
31. Padilla, R.; Netto, S.L.; Silva, E.A.B. A Survey on Performance Metrics for Object-Detection Algorithms. In Proceedings of the International Conference on Systems, Signals and Image Processing, Rio de Janeiro, Brazil, 1–3 July 2020; pp. 237–242.
32. Babiuch, M.; Foltýnek, P.; Smutný, P. Using the ESP32 Microcontroller for Data Processing. In Proceedings of the 20th International Carpathian Control Conference, Hotel Turówka, Poland, 26–29 May 2019; pp. 1–6.
33. Dokic, K. Microcontrollers on the edge–is esp32 with camera ready for machine learning? In Proceedings of the International Conference on Image and Signal Processing, Marrakech, Morocco, 4–6 June 2020; pp. 213–220.