IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# Design, Development and Evaluation of an Intelligent Animal Repelling System for Crop Protection based on Embedded Edge-AI

**DAVIDE ADAMI[1], MIKE O. OJO[2], (MEMBER, IEEE) AND STEFANO GIORDANO[3], (Senior Member, IEEE)**

[1]CNIT Research Unit, Department of Information Engineering, University of Pisa, Italy
[2]Department of Veterinary Sciences, University of Turin, Grugliasco, TO, Italy
[3]Department of Information Engineering, University of Pisa, Via G. Caruso 16, 56122, Pisa, Italy

Corresponding author: Mike O. Ojo (e-mail: mikeoluwatayo.ojo@unito.it)

This work was supported by Regione Toscana, Italy, under the program POR FESR 2014-2020 (ULTRADEFENDER and ULTRREP projects) and by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

**ABSTRACT** In recent years, edge computing has become an essential technology for real-time application development by moving processing and storage capabilities close to end devices, thereby reducing latency, improving response time and ensuring secure data exchange. In this work, we focus on a Smart Agriculture application that aims to protect crops from ungulate attacks, and therefore to significantly reduce production losses, through the creation of virtual fences that take advantage of computer vision and ultrasound emission. Starting with an innovative device capable of generating ultrasound to drive away ungulates and thus protect crops from their attack, this work provides a comprehensive description of the design, development and assessment of an intelligent animal repulsion system that allows to detect and recognize the ungulates as well as generate ultrasonic signals tailored to each species of the ungulate. Taking into account the constraints coming from the rural environment in terms of energy supply and network connectivity, the proposed system is based on IoT platforms that provide a satisfactory compromise between performance, cost and energy consumption. More specifically, in this work, we deployed and evaluated various edge computing devices (Raspberry Pi, with or without a neural compute stick, and NVIDIA Jetson Nano) running real-time object detector (YOLO and Tiny-YOLO) with custom-trained models to identify the most suitable animal recognition HW/SW platform to be integrated with the ultrasound generator. Experimental results show the feasibility of the intelligent animal repelling system through the deployment of the animal detectors on power efficient edge computing devices without compromising the mean average precision and also satisfying real-time requirements. In addition, for each HW/SW platform, the experimental study provides a cost/performance analysis, as well as measurements of the average and peak CPU temperature. Best practices are also discussed and lastly, this article discusses how the combined technology used can help farmers and agronomists in their decision making and management process.

**INDEX TERMS** Ungulates, Internet of Things, Smart Agriculture, Edge Computing, Real-time Embedded Systems, Object Detection

## I. INTRODUCTION

IN the Agriculture 4.0 era, cutting-edge technologies such as the Internet of Things (IoT), Big Data, Blockchain, Edge/Cloud computing, and Artificial Intelligence (AI) are increasingly used to enable innovative applications that have the potential to revolutionize our daily lives [1]–[3].

Agriculture automation has been on the rise leveraging, among others, Deep Neural Networks (DNN) and IoT for the development and deployment of many controlling, monitoring and tracking applications at a fine grained level [4].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



For instance, the automation of agricultural production has enabled the continuous monitoring of crop growth and the intelligent management of weeds [5]. This helps to provide accurate and efficient solutions to support agricultural activities compared to the traditional methods, which are performed manually, with processes that are time consuming, tedious, increase production costs, and are prone to errors.

In this rapidly evolving scenario, managing the relationship with the elements external to the agriculture ecosystem, such as wildlife, is a relevant open issue. One of the main concerns of today's farmers is protecting crops from wild animals attacks. In the last three decades, indeed, the amount of damages caused by predatory wild animals have significantly increased all over Europe. For instance, the percentage of losses in top wine production due to ungulates attacks amounts to 75% of total losses. In the region of Tuscany, Italy, wild boars, roe deers and fallow deers (see Figure 1) are the most common ungulates causing crop damage [6], [7]. There are different traditional approaches to address this problem which can be lethal (e.g., shooting, trapping) and non-lethal (e.g., scarecrow, chemical repellents, organic substances, mesh, or electric fences). Nevertheless, some of the traditional methods have environmental pollution effects on both humans and ungulates, while others are very expensive with high maintenance costs, with limited reliability and limited effectiveness.

As shown in [8], [9], the authors of this paper have recently contributed to the design and development of an innovative ad-hoc system capable of detecting the presence of animals[1] by means of a PIR sensor, and repelling them through the generation of ultrasounds, which has recently been proven as an alternative, effective method for protecting crops against ungulates. Animals generally have a sound sensitive threshold that is far higher than humans. They can hear sounds having lower frequencies with respect to the human ear. For instance, while the audible range for humans is from $64Hz$ – $23KHz$, the corresponding range of goats, sheep, domestic pigs, dogs and cats is $78Hz$ – $37KHz$, $10Hz$ – $30KHz$, $42Hz$ – $40.5KHz$ $67Hz$ – $45KHz$ and $45Hz$ – $64KHz$ respectively [10], [11]. Researchers have shown that generating ultrasounds within the critical perceptible range causes animals to be disturbed, thus making them move away from the sound source [12]. At the same time, these ultrasounds are not problems to the human ear even when the frequency range is beyond the human ear. The human eardrum has a far lower specific resonant frequency than animals and cannot vibrate at ultrasound frequency. In addition, such solution is non-lethal and has no effect of environmental pollution, no impact on the landscape, and no limitation on tourists' and bikers movements, which is very relevant especially in tourist areas.

In this paper, we present an upgraded version of the system, that combines AI Computer Vision for detecting animal species, and specific ultrasound emission (i.e., different for each species) for repelling them. The new system requires communication, computation and storage capabilities, and therefore we designed and developed an infrastructure that integrates ad-hoc IoT devices, Edge and Cloud Computing.

Within the IoT world, the need for edge computing devices that utilize complex AI algorithms in the consumer market are skyrocketing [13]–[15]. This helps to bring computation and data storage onto the device by addressing latency, security and load balancing issues [16], [17] but may suffer from poor performance and energy efficiency [18]. This is because many AI algorithms require computational power that greatly outweighs the capacity of resource-and energy constrained IoT devices. However, this has inspired the industry and academia to look for new computing and storage solutions that can enable both dense and energy efficient architectures for embedded devices. In this work, we explore the case of embedded artificial intelligence, in which the data analysis runs directly on the embedded system itself.

As our system is deployed in rural areas, where energy efficiency is essential, we evaluated some HW and SW architectural alternatives for the design and implementation of the AI embedded Edge computing components to be added to our system. More specifically, in this work, as regards HW, we considered two low-power consumption edge computing devices: Raspberry Pi Model 3 B+ (hereinafter, RPi 3B+) with or without Intel Movidius Neural Compute Stick (NCS) and NVIDIA Jetson Nano, whereas animal recognition programs were developed on two DNN-based real-time object detectors including YOLOv3 and Tiny-YOLOv3. These constitute state-of-the-art real-time detectors with an acceptable trade-off between accuracy and efficiency that enables their deployment on hardware platforms with limited computational capabilities. Moreover, as the coverage of telecommunication infrastructures in rural areas is generally poor or absent, and troubleshooting or maintenance operations are very difficult, we identified LoRa and LoRaWAN as the most suitable connectivity solutions, especially for monitoring and control operations as demonstrated in [19], [20]. This ensures robustness, low power consumption and ability to cover wide areas.

The main contributions of this article are:

1) We provide a thorough, complete description of the design, deployment and assessment of an intelligent smart agriculture repelling and monitoring IoT system based on embedded edge AI, to detect and recognize the ungulates, as well as generate ultrasonic signals tailored to each species of the ungulate.
2) We provide a methodology for deploying the system.
3) We deploy and evaluate various edge computing devices (Raspberry Pi, with or without a neural compute stick, and NVIDIA Jetson Nano) running real-time object detector (YOLO and Tiny-YOLO) with custom-trained models to identify the most suitable animal recognition HW/SW platform to be integrated with the ultrasound generator.

[1] We use the terms "animals" and "ungulates" interchangeably

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



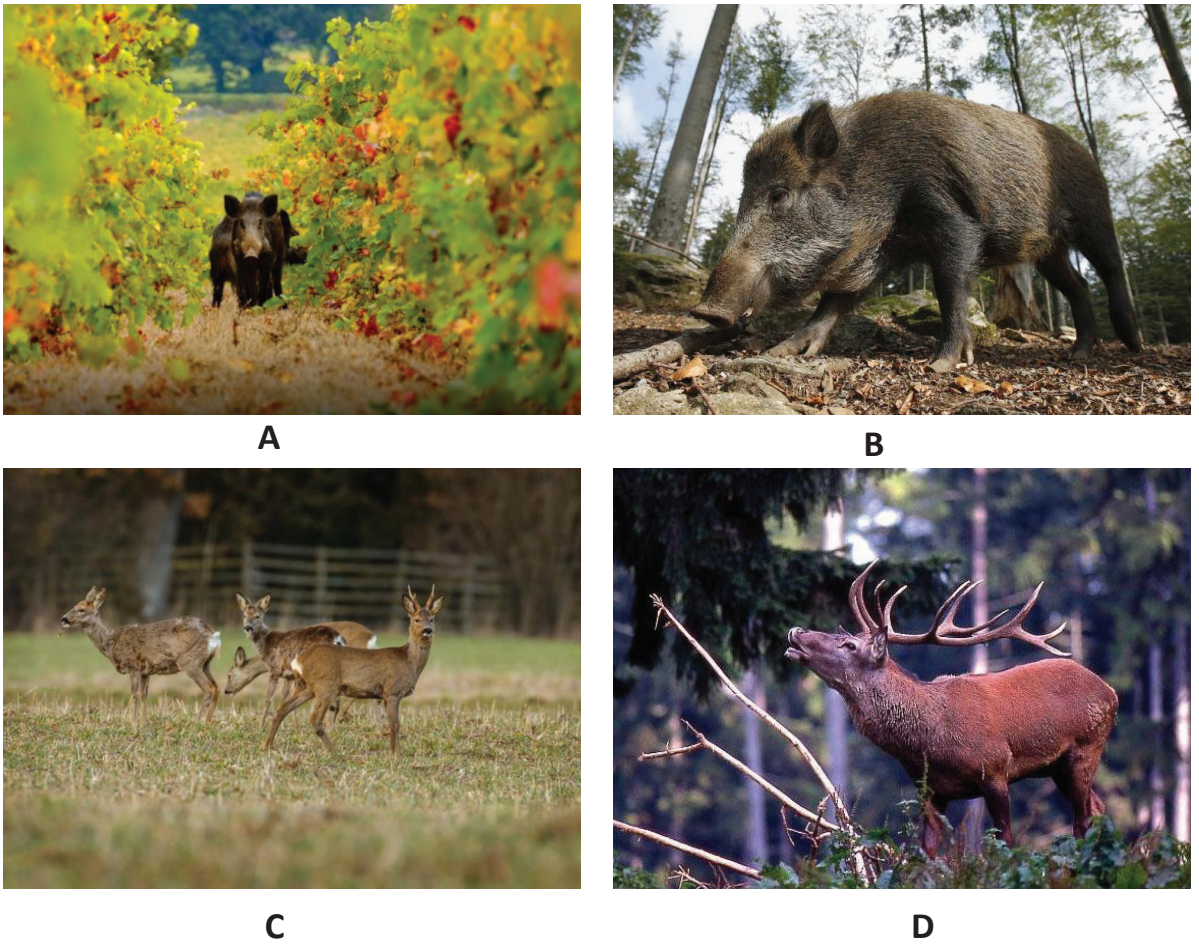



Figure 1: (A) wild boar in a vineyard, (B) wild boar (C) roe deer (D) fallow deer

4) For each HW/SW platform, we provide an experimental study that provides a cost/performance analysis, as well as measurements of the average and peak CPU temperature.
5) We provide how the combined technology used can help farmers and agronomists in their decision making and management process.

The rest of this paper is organized as follows: Section II presents the related work and the background. Edge-AI is also described in this section. The overall system architecture is described in Section III, with emphasis on the hardware description. Section IV introduces the innovative application focusing on data set description, image pre-processing and the DNN based animal detectors. In Section V, we present the experimental setup, where the experimental results discussing the model performance and the embedded implementation are presented Lastly, Section VI concludes the paper with some final remarks.

## II. BACKGROUND AND RELATED WORKS

In this section, we provide the background about the technologies used in this work focusing on real-time detection and edge-AI. Next, we present the related works.

### A. REAL-TIME DETECTION BACKGROUND

Running a real time object detection algorithm involves computational demanding tasks, which involves powerful devices where these algorithms are implemented on. This is essential to achieve good results in less time. However, due to the nature of our problem and its environs, device portability plays a major role in realizing a real-time animal detection. This is challenging because of the additional constraints in terms of memory footprint and power consumption, which generally conflict with latency and accuracy requirements. This paper will promote the use of embedded devices with low form factor.

As earlier described, DNN need high performance computing resources (i.e. devices with graphics processing units (GPU)) for real time computations. This makes it difficult to deploy DNN on embedded systems with low computation resources, due to its extensive computation tasks. However, there are some condensed versions of some DNNs that work on low computation embedded systems efficiently, as well as some embedded systems with commercial hardware

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



accelerators such as NCS and Google Coral [21], where DNN computations can run efficiently in real-time bringing the concepts of edge-AI. Such hardware accelerators feature optimized hardware architectures that allow to realize inferences of DNN models with low latency and reduced power consumption.

There are several platforms available in order to run the object detection algorithms. OpenCV and Tensorflow are considered proven platforms for machine learning purposes [22]. OpenCV is a mature and stable computer vision library, with an impressive user community. Using OpenCV results in more utilization of time and resources in image processing and less in interpreting, thus resulting in faster execution. On the other hand, tensorflow is a framework that enables a developer to create his/her algorithms, in addition to the ones it has implemented. Tensor is less mature than OpenCV, and on this note, OpenCV is selected as the benchmark in our system.

In the literature, several deep learning approaches, including Region-Based Convolutional Neural Networks (R-CNN), You Only Look Once (YOLO), and Single Shot multibox Detector (SSD) have been applied extensively to agriculture-related applications. Faster R-CNN [23] uses the region proposal network (RPN) method to detect a region of interest (RoI) in the image. Then a classifier is used to classify bounding boxes, and fine tuning is used to process the bounding boxes, enabling a speed increase over R-CNN, while the target can be detected accurately. He et al. introduced Mask-RCNN [24], which is an extension to Faster R-CNN for instance segmentation (i.e., location of exact pixels followed by masks for each object inside the bounding box).

While Faster R-CNN and Mask-RCNN were 2-stage methods (region proposal stage followed by classification stage), YOLO was developed as a one stage method. In YOLO, the object classification (the category of the bounding box) and object positioning (the position of the bounding box) can be predicted through a single convolutional network in one step. The 'single shot detector' SSD implemented prior boxes (subsets of fixed sized anchor boxes) at different resolutions of feature maps at different levels inside the network for multi-scale training, making it a very fast (faster than Faster R-CNN) yet accurate framework for object detection. The YOLOv3 (the third version of the YOLO), being one of the state-of-the-art detectors was chosen because of its stability and its ability to achieve good performances with high frame rate. YOLOv3 has high detection accuracy and speed and it also performs well with detecting small targets. Another strength in YOLOv3 is its tiny version, which is lighter, smaller (less layer) and faster (execution time). Tiny-YOLOv3 has less accuracy than other CNNs, but it is very convenient for constrained environments, which is the main focus of our problem statement. Therefore, YOLOv3 and Tiny-YOLOv3 are the object detection algorithms proposed for our testbed and use case. A list of abbreviations used in this article is provided in Table 1.

Table 1: List of Abbreviations

| Symbols | Description |
|---|---|
| AI | Artificial Intelligence |
| AP | Average Precision |
| API | Application Programming Interface |
| ARM | Advanced RISC Machines |
| ASIC | Application Specific Integrated Circuits |
| CNN | Convolutional Neural Networks |
| CPU | Central Processing Unit |
| CUDA | Compute Unified Device Architecture |
| DNN | Deep Neural Networks |
| EUI | Extended Unique Identifier |
| FLOPS | Floating Point Operations per Second |
| FN | False Negative |
| FPS | Frames per Second |
| GPU | Graphics Processing Unit |
| HW | Hardware |
| IoT | Internet of Things |
| IoU | Intersection over Union |
| LiPo | Lithium-ion Polymer |
| LoRaWAN | Long Range Wide Area Network |
| LPWAN | Low-Power Wide-Area Network |
| mAP | Mean Average Precision |
| MPPT | Maximum Power Point Tracker |
| NCS | Neural Compute Stick |
| OTAA | Over-the-Air Activation |
| PIR | Passive Infra-red |
| PWM | Pulse Width Modulation |
| R-CNN | Region-Based Convolutional Neural Networks |
| RAM | Random-Access Memory |
| RoI | Region of Interest |
| RPi | Raspberry Pi |
| RPN | Region Proposal Network |
| SSD | Single Shot Multibox Detector |
| SoC | System on Chip |
| SW | Software |
| TTN | The Things Network |
| TP | True Positive |
| TPU | Tensor Processing Unit |
| UAV | Unmanned Aerial Vehicle |
| USB | Universal Serial Bus |
| YOLO | You Only Look Once |

### B. EDGE-AI

Artificial intelligence (AI) on edge will be part of the next generation of the Internet of Things. Moreover, the need to bring more essential computation onto the devices has been on the increase, which undoubtedly serve as the concept for edge-AI. This has to do with performing computations locally on an embedded system to deal with applications with low latency situations and real-time responses by boosting operational speed, reducing decision latency, network congestion and power consumption. Edge-AI computing mechanism for agricultural applications is very vital for the entire world to solve most of the relevant issues at global level. The challenge in meeting the implementation requirements for edge-AI is to ensure high output accuracy of algorithms while consuming low power. Nevertheless, the innovation in hardware options, involving central processing units (CPUs), GPUs, application-specific integrated circuits (ASICs), and system-on-a-chip (SoC) accelerators, has made edge-AI possible.

Qualcomm, Intel and NVIDIA are the leading market

brands which are contributing enormously to the development of AI at the edge. Some of the devices are Intel's Movidius Neural Computing Stick (NCS), NVIDIA's Jetson variants such as TX1, TX2, Nano, AGX Xavier which are considerably cheap, with low form factor, light weight and works efficiently to implement computationally extensive algorithms with multiple layers of CNN. Another importance of NVIDIA Jetson variants is the introduction of CPU-GPU architecture [25], where CPU boots up the firmware and the CUDA-capable GPU come with potential to accelerate complex machine-learning tasks.

### C. RELATED WORKS

With the rapid increase of machine learning in various sectors, deep learning technology has been extensively used in agriculture-related applications [26]. Deep learning can be used for animal identification, crop classification among others. In recent years, the development of DNN achieved good results in crop detection such as strawberry detection [27], mango detection [28] and apple detection [29], in weed classification [5], in face recognition [30], [31], in behavioural recognition [32], handwritten character recognition [33], license plate recognition [34].

Shifting to animal identification, animal detection has been researched by many researchers for the purposes of estimation, localization among others. The work in [35] examined how YOLOv3 was able to locate sheep in drone footage. The work was able to detect sheep as a superclass, with a resolution of 832 $\times$ 832 pixels, and a confidence ratio of 0.1 yielding a reliability of 99%. In [36], the authors introduced FLYOLOv3, a deep learning filter layer YOLOv3 to detect the key parts of dairy cows in complex scenes. Wang, Juan, *et.al* The work in [37] presented a YOLOv3 model to detect the behaviour of egg breeders. In addition, transfer learning was also implemented to expand the DNN model in realizing the behaviour recognition of egg breeders with low stock density. The authors in [38] also followed the same approach by proposing an automated broiler digestive disease detector based on DNN model to classify fine-grained abnormal broiler droppings images as normal and abnormal.

The work in [39] introduced a drone-based approach for cattle detection using SSD and YOLOv3 as the object detectors. However, very few images were captured for training the dataset, which reflected on the results. The work in [40] provided a holistic approach in using different DNN object detector methods including YOLOv3 to create bird detection models using aerial photographs captured by Unmanned aerial vehicle (UAV). The work presented the model performance, but lacked the implementation of the model on a platform. [41] introduced YOLO Fish, an effective fish detector using YOLO on nordic fish species. Moreover, in [42], the use of animal behavior analysis software was also used to evaluate the behavior of chickens such as Pocket Observer Version 3 (Noldus Information Technology; Wageningen, the Netherlands).

With the idea of implementing deep learning algorithms on embedded platform, we briefly describe some of the works that utilizes deep learning on embedded platforms related to Agriculture. The use of RPi has also been demonstrated by [43] by using devices to repel birds in optimizing crop production in Agriculture. RPi and NVIDIA Jetson TX2 were used in [44] to monitor and detect Asian citrus psyllid pests in orchards using convolutional neural networks. The use of RPi with Intel Movidius has been demonstrated in [45] to perform seeds recognition, and germination detection through the images processing, where 97% accuracy and 83% of IoU was achieved. The work in [46] presented a Neural Network based prediction model that acts as a data collector for sensor nodes to measure air temperature inside a greenhouse with embedded devices such as RPI. Livestock detection algorithm using modified versions of U-net and Google Inception-v4 net was presented in [47]. In this work, we made use of Tiny-YOLOv3 and YOLOv3 as our object detector.

However, despite these studies being successful in terms of animal detection, most of them focused on the model performance rather than providing both the model performance and its implementation on a platform. In addition, the number of images used in these studies was relatively small. Therefore, these methods show some limitations in terms of their ability to detect animals across various environments. This work presents a real-time monitoring solution based on IoT technology to address the problems of crop damages against ungulates. This work, an embedded Edge-AI application, focuses on the model performance of YOLOv3 and Tiny-YOLOv3 as well as implementing it in real-time, running on various embedded systems such as RPi 3B+ with or without Intel Movidius NCS and NVIDIA Jetson Nano.

Relating to ultrasound in animals, researchers have shown that the use of Ultrasound emission is an excellent system to repel ungulates, without simultaneously disturbing humans. Extraneous sound usage has been demonstrated in [48] to be stressful to animals and said to have profound behavioural, physiological and even anatomical effects. In the literature, ultrasound usage has been shown to be effective in repelling bats in [49], rodents in [50] and even insects in [51]. Our work is based on the repelling of ungulates especially wild boar, deer and wolves with the use of ultrasounds.

## III. INTELLIGENT ANIMAL REPELLING SYSTEM ARCHITECTURE

In this section, we present the overall system architecture (see Figure 2), and we analyze in detail the characteristics of the Intelligent Animal Repelling Devices and of the network infrastructure.

### A. INTELLIGENT ANIMAL REPELLING DEVICE

As mentioned in the introduction section, the system is based on Intelligent Animal Repelling Devices, which enable real-time animal recognition and repelling. To this aim, a new version of the Animal Repelling Device has been integrated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



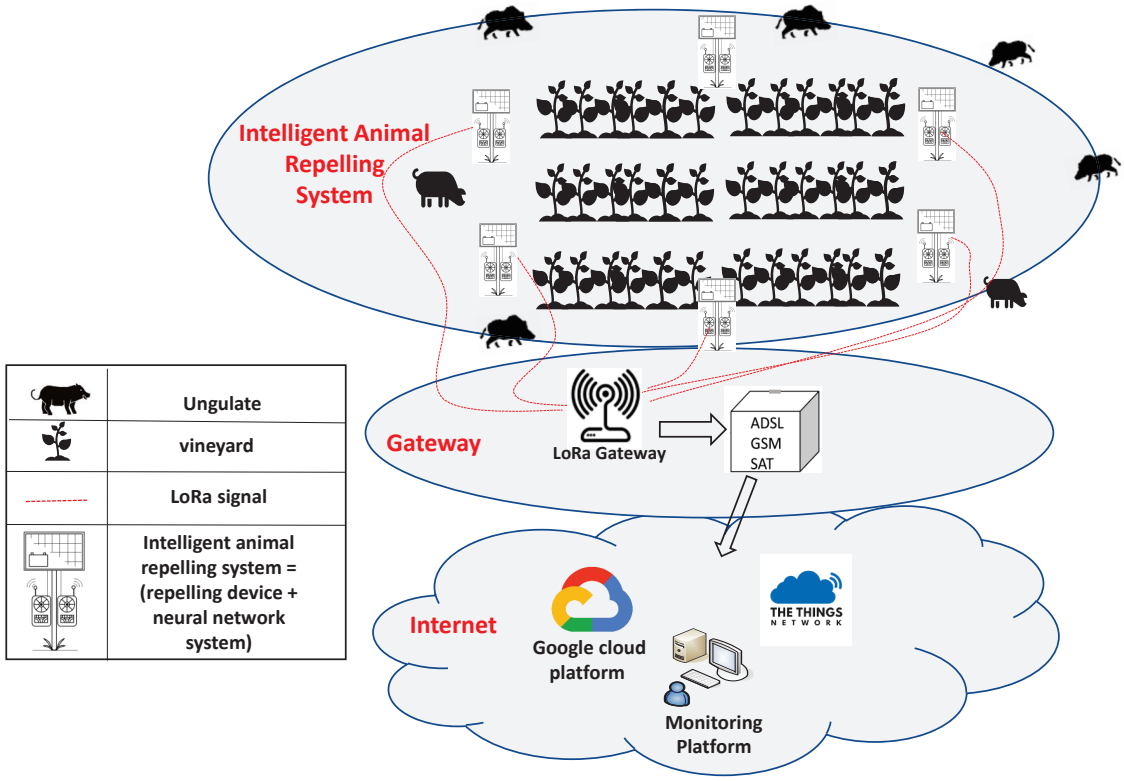



Figure 2: System Architecture

with a tiny and powerful edge computing device running DNN software.

### 1) Detection and ultrasound generation

The board of the new Animal Repelling Device is still based on the ATSAMD21G18A 32-bit ARM Cortex® M0+ core architecture clocked at 48MHz, with 32KB of RAM and 512 KB of flash memory. It also features a LoRa module RN2483A and xbee radio module supported by the LoRaWAN and IEEE 802.15.4 standard respectively. The device uses a solar panel and LiPo batteries charged with the use of Maximum Power Point Tracker (MPPT). Moreover, the device is equipped with a PIR sensor to detect targets and activate animal recognition function. The power of the ultrasound produced by the tweeter is $110dB$ at about $\sim$1m in a wide band of $18kHz-27kHz$. Frequencies can be tuned according to the animal to repel.

### 2) Real-time animal recognition

For the implementation of the animal recognition model, and to improve real-time performance, different edge computing devices have been considered: RPi 3B+ with or without Intel Movidius NCS and NVIDIA Jetson Nano. Figure 3 shows the hardware platforms considered in this work, while Table 2 provides a comparison between their main specifications.

Intel Movidius NCS is the first generation of the neural compute sticks, an embedded AI platform designed by Intel

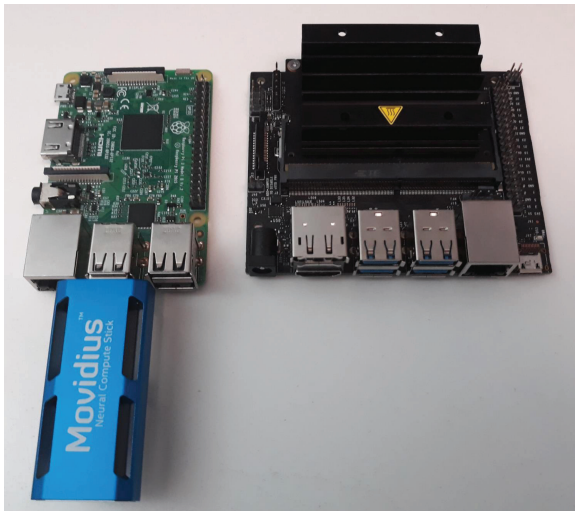



Figure 3: RPi 3B+ with Intel Movidius NCS on the *left* and NVIDIA Jetson Nano on the *right*

Movidius [52]. It is a USB hardware accelerator designed for low power devices to achieve high frame rates. At the core of this device is the Myriad 2 Visual Processing Unit (VPU) processor, an AI-optimized chip for accelerating vision computing based on CNN. Intel Movidius NCS provides a USB 3.0 interface and can be easily attached to edge devices such

Table 2: Main specifications of the platforms investigated in this study

| Features | Intel Movidius NCS | NVIDIA Jetson Nano | RPi 3B+ |
|---|---|---|---|
| Size | 73 × 26 mm | 73 × 45 mm | 85.6 × 56.5 mm |
| HW Accelerator | Myriad 2 VPU | 128-core NVIDIA Maxwell GPU | N.A |
| CPU | N.A | Quad-core ARM A57 @ 1.43GHz | Quad-core Cortex-A53 |
| Memory | 4GB LPDDR3 | 4GB LPDDR4 | 1GB LPDDR2 |
| Nominal Power | 1W | 5/10W | 2W |
| Weight | 18g | 140g | 50g |
| Peak Performance | 100GFLOPs | 472GFLOPs | 6GFLOPS |

as RPi.

RPi 3B+ is one of the most used embedded platforms with a large variety of usage. RPi 3B+ is a low-cost, small size, powerful computer board with onboard electronics supporting a large number of input and output peripherals [53]

NVIDIA Jetson Nano is one of the recent Jetson platforms presented by NVIDIA. NVIDIA Jetson Nano is a small powerful embedded computer with a dedicated GPU for hardware acceleration. It was presented in June 2019 and allow to run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. It features a 128-core NVIDIA Maxwell® GPU with a AI peak performance of 472GFLOPs. [54] It runs multiple neural networks in parallel and processes several high-resolution sensors simultaneously, providing a high-performance computing at just 5W to 10W.

#### 3) Integration between animal recognition and animal repelling functions

When a movement is detected through the PIR sensor, the SAMD21 microprocessor sends an "activity detection" message to the edge computing device through the xbee radio interface. Note that, the edge computing device is also equipped with the xbee radio, which makes it to be embedded with IEEE 802.15.4 capabilities. The edge computing device activates the camera, then executes its DNN software to identify the target, and if an animal is detected, it sends back a message to the Animal Repelling Device including the type of ultrasound to be generated according to the category of the animal. The "activity" message is also transmitted from the repeller device via LoRa to the LoRa gateway, which then forwards the packet to the TTN server.

### B. NETWORK INFRASTRUCTURES

A LoRa/LoRaWAN network has been set-up leveraging The Things Network (TTN)[2]. The network consists of the following parts:

- ***Intelligent Animal Repelling Devices***: Type A LoRa end-nodes with firmware supporting LoRaWAN protocol and OTAA, i.e., communication with the LoRa network server to perform the activation process, called JOIN procedure. In TTN a new application has been created (AppID, AppEui, Description, Handler) and devices have been registered with their configuration parameters (Device ID, Device EUI, AppEUI, AppKey, AppEUI).
- ***Gateway***: a LoRa gateway has been set-up and configured to enable the coverage of the area where the LoRa devices are deployed. Also, the gateway has been registered in TTN (Gateway ID, Frequency Plan, Location, Indoor or Outdoor Placement, Description). The gateway, which scans the frequency spectrum and forwards all packets to the network server, is connected to the Internet through a satellite backhaul network [3].
- ***Network Server***: it understands the LoRa protocol and interprets data sent by the end-devices, routing messages to the right application and back. The network server processes data, but does not store it anywhere. TTN provides network servers and offers a data API to get the data out of the network. Therefore, it is possible to store it on servers or forward it to a cloud service.
- ***Application***: according to TTN definition, whatever devices communicate with on the Internet. In our case, the monitoring application is a visual flow developed using Node-RED[4]. As previously highlighted, before communicating with the end-devices, it is necessary to add a new application to the TTN and register devices to it.

## IV. METHODOLOGY AND SERVICES FOR ANIMAL RECOGNITION

### A. DATA SET DESCRIPTION

In this work, we consider two popular ungulates (wild boar and deer) in the Tuscany region of Italy. Fallow deer and Roe deer are classified as deer in our experimental work. Images were acquired through the use of digital camera with 20 megapixels under variable lighting conditions during the different times and days of June to August. The images were collected with cloudy and sunny weather conditions. 1000 images of wild boar and deer were collected during the

[2]https://www.thethingsnetwork.org/

[3]https://www.skydsl.eu

[4]https://www.nodered.org/

different times highlighted above. These 1000 images were expanded to 10000 images using data augmentation methods, yielding the training set. The training data set is used to train the animal recognition model.

### B. IMAGE PRE-PROCESSING AND IMAGE SETS

All object recognition methods re-size the input image to a specific resolution ('network resolution') for training. The feature extractors used by deep learning frameworks usually require a square input resolution. In its original configuration, YOLO resizes the input image such that the shorter dimension of height or width is resized to 416 x416 pixels, while keeping the aspect ratio unchanged. Network resolution can be increased to accept larger input images (e.g., 2048 $\times$ 2048 pixels), but at the cost of increased memory and computation requirements. Before training the image, data augmentation was necessary to increase the image diversity for better representation of the images. In order to achieve better recognition accuracy and robustness provided by effective data augmentation, it was necessary to pre-process the images with the augmentation methods such as jitter, image rotation, flipping, cropping, multi-scale transformation, hue, saturation, Gaussian noise and intensity.

Training object detection models require labelled data, i.e., the class-label and the coordinates of all ground truth bounding boxes in training images. Manual labelling is tedious, intensive and prone to user bias especially for images having a large number of objects and clusters [55], annotation ((drawing of ground truth bounding boxes) on the other hand was easier on tiles than on full images. The task of labeling objects in images has become easier due to availability of freely graphical annotation tools. The graphical image annotation tool YOLO Annotation tool[5] was used. The annotated information was saved in YOLO Darknet format, which includes the class of the annotated object, the coordinates of the rectangular box among others.

### C. YOLOV3 AND TINY-YOLOV3 NETWORK STRUCTURE

YOLO algorithm proposed by Redmon et.al [56] is a neural network capable of detecting objects, people and animals present in an image and their position in only one step and also obtains the position and category of the object, animal etc. directly at the output layer. The YOLO algorithm is based on convolutional design, where an input image is divided into a grid system and each grid cell represents a candidate region of detected objects. The main innovation brought by YOLO is to be able to perform the surveys in one go, so for this reason it is quite fast and performing. Moreover, another advantage of YOLO is the ability to predict a large fixed number of objects and sets a threshold to eliminate predictions with low probabilities. YOLO introduces improved versions namely YOLOv2 [57] and YOLOv3 [58]. The main difference between YOLO (the first version of YOLO) and YOLOv2 (the second version of YOLO) is the addition of batch normalization layers after all convolutional layers and the removal of the last fully connected layers. The addition of anchor boxes introduced in YOLOv2 helps in improved accuracy in predicting the object location. Darknet-19 (a 19-layer convolutional network) with 5 max-pooling layers is the backbone to the YOLOv2 network.

#### 1) YOLOv3

YOLOv3 has made significant improvements to the YOLO network, and has been demonstrated in [58] in terms of better accuracy and efficiency. The choice of feature extractor is crucial as the number of parameters and types of layers directly affect memory, speed, and performance of the detector. The backbone feature extractor for YOLOv3 is Darknet-53. Darknet-53 (a 53-layer convolutional network) is an improvement on Darknet-19 by the addition of successive 3 $\times$ 3 and 1 $\times$ 1 convolutional layers with skip connections similar to ResNet [59] as shown in Figure 4. Moreover, in YOLOv3, the softmax loss in the YOLOv2 structure is replaced by a logistic loss, which has obvious advantages in small object detection. YOLOv3 works by performing a logistic regression, which indicates the bounding boxes and the probabilities of belonging to a class for each of them, with a single network passage.

#### 2) Tiny-YOLOv3

Tiny-YOLOv3 is a tiny version of YOLOv3 that unifies object detection and classification into a single regression problem. This tiny version is based on a reduction of the number of hidden layers. Tiny-YOLOv3 is suitable for constrained environments where its resource consumption leads to an increase in speed. Though, it leads to an increase in speed, but comes at the cost of its accuracy of the neural network, which is just able to predict objects at two scales. Image size of 416 $\times$ 416 is used as the input in Tiny-YOLOv3, and two detectors consisting of 3 sub-detectors of Yolo output 13 $\times$ 13 and 26 $\times$ 26 grids. Tiny-YOLOv3 is able to meet with real-time requirements, but with low detection precision. Table 3 describe the settings of Tiny-YOLOv3 network structure.

### D. BATCH PROCESSING PATTERN

The parameters used in the experiment described in the next section are as follows:

1) The batch parameter, which indicates the batch size used during the training stage, was set to 64. This means that 64 images from the dataset were used in one iteration to update the parameters of the neural network.
2) The subdivision parameter, which controls a fraction of the batch size at one time to be processed by the GPU was set to 2. This prevented the overload of the GPU memory while the DNN was trained.

[5]https://github.com/ManivannanMurugavel/YOLO-Annotation-Tool : accessed 03/05/2021

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



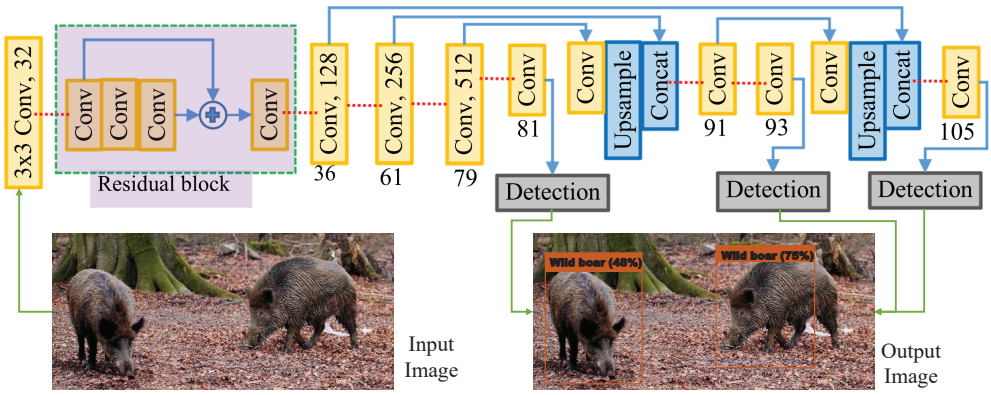



Figure 4: Block diagram of YOLOv3 architecture. 'Conv' refers to 'convolution layer' and'concat' to 'concatenation layer'. Output image displays bounding box on region of interest (ROI), classification result for wild-boar and probability score

Table 3: Tiny-YOLOv3 original architecture with input images of dimension 416 × 416

| Layer | Type | Filters | Size/Stride | Input | Output |
|---|---|---|---|---|---|
| 0 | Convolutional | 16 | 3 x 3/1 | 416 x 416 x 3 | 416 x 416 x 3 |
| 1 | Maxpool | | 2 x 2/2 | 416 x 416 x 16 | 208 x 208 x 16 |
| 2 | Convolutional | 32 | 3 x 3/1 | 208 x 208 x 16 | 208 x 208 x 32 |
| 3 | Maxpool | | 2 x 2/2 | 208 x 208 x 32 | 104 x 104 x 32 |
| 4 | Convolutional | 64 | 3 x 3/1 | 104 x 104 x 32 | 104 x 104 x 64 |
| 5 | Maxpool | | 2 x 2/2 | 104 x 104 x 64 | 52 x 52 x 64 |
| 6 | Convolutional | 128 | 3 x 3/1 | 52 x 52 x 64 | 52 x 52 x 128 |
| 7 | Maxpool | | 2 x 2/2 | 52 x 52 x 128 | 26 x 26 x 128 |
| 8 | Convolutional | 256 | 3 x 3/1 | 26 x 26 x 128 | 26 x 26 x 256 |
| 9 | Maxpool | | 2 x 2/2 | 26 x 26 x 256 | 13 x 13 x 256 |
| 10 | Convolutional | 512 | 3 x 3/1 | 13 x 13 x 256 | 13 x 13 x 512 |
| 11 | Maxpool | | 2 x 2/1 | 13 x 13 x 512 | 13 x 13 x 512 |
| 12 | Convolutional | 1024 | 3 x 3/1 | 13 x 13 x 512 | 13 x 13 x 1024 |
| 13 | Convolutional | 256 | 1 x 1/1 | 13 x 13 x 1024 | 13 x 13 x 256 |
| 14 | Convolutional | 512 | 3 x 3/1 | 13 x 13 x 256 | 13 x 13 x 512 |
| 15 | Convolutional | 255 | 1 x 1/1 | 13 x 13 x 512 | 13 x 13 x 255 |
| 16 | YOLO | | | | |
| 17 | **Route 13** | | | | |
| 18 | Convolutional | 128 | 1 x 1/1 | 13 x 13 x 256 | 13 x 13 x 128 |
| 19 | Up-sampling | | 2 x 2/1 | 13 x 13 x 128 | 26 x 26 x 128 |
| 20 | **Route 19 8** | | | | |
| 21 | Convolutional | 256 | 3 x 3/1 | 13 x 13 x 384 | 13 x 13 x 256 |
| 22 | Convolutional | 255 | 1 x 1/1 | 13 x 13 x 256 | 13 x 13 x 256 |
| 23 | YOLO | | | | |

### *E. TRAINING PLATFORM*

Edge-AI consists of performing computations locally on an embedded system in real-time. Since the training process requires a lot more computational power as compared to the inference process, it cannot be performed on the embedded platform in a reasonable amount of time. It requires a powerful machine with integrated GPUs, or TPUs.

In our research, we avoided the use of a physical machine and relied on a virtual machine provided by Google Cloud Platform (GCP) to perform our model training. The virtual machine is made up of 16 vCPU 2, 64GB of memory, with a single NVIDIA® Tesla® T4 GPU running Ubuntu 18.04 LTS. A T4 GPU has 16GB GDDR6 of GPU memory on-board, and also includes NVIDIA Tensor Cores for faster training and RTX hardware acceleration for faster ray tracing. T4 is offered as a passively cooled board that requires system air

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



flow to operate the card within its thermal limits. This GPU is designed to accelerate inference, or predictions generated by deep learning models, for low latency or high throughput.

## V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, firstly some technical details of the training process of the neural network models used by YOLOv3 and Tiny-YOLOv3 are presented, then experimental evaluations are discussed for both the models and their deployment on the selected embedded platforms. Quantitative and qualitative results are reported with a performance comparison between the different platforms. Finally, the deployment methodology is described.

### A. MODEL PERFORMANCE

In this section, we analyze the performance of YOLOv3 and Tiny-YOLOv3 models after training their DNNs for ungulates recognition. As highlighted in Section IV-C, both YOLO versions are regarded as state-of-the-art object detection approaches and can be used as real-time detectors based on performance requirements. The training was carried out on the Google Cloud Platform as demonstrated in Section IV-E. To evaluate the models performance, we adopted three metrics: Recall, AP (Average Precision), mAP (mean Average Precision), which are computed using the formulas (1), (2) and (3) respectively. These metrics are essential in evaluating the performance of the neural networks models for object detection:

$$Recall(P) = \frac{TP}{TP + FN} \tag{1}$$

$$AP = \int_0^1 p(r)dr \tag{2}$$

$$mAP = \frac{\sum_{i=1}^{C} AP_i}{C} \tag{3}$$

As regards Recall, $TP$ (True Positive) denotes the number of detected ungulates, and $FN$ (False Negative) denotes the number of undetected ungulates, which are marked as missing predictions. Therefore, Recall evaluates how good the model is in detecting ungulates: low values of Recall mean that in many cases, although present in the images, ungulates are not detected. Precision measures the percentage of correct predictions with respect to overall predictions for each class of ungulates, denoting the accuracy of the detector. $mAP$ is the mean of $AP$ computed over all classes where $C$ denotes the number of classes.

Computing the values of Precision and Recall, it is possible to get the precision/recall curve. The graph is then usually smoothed in order to get a monotonically decreasing precision curve by setting $p(r) = \max_{r' \geq r} p(r')$. T). The average precision of the model is computed as the area under the obtained curve and it is always a number between 0 and 1 as shown in (2). By increasing recall, precision decreases and the downward slope of the curve depends on the setting of the detection threshold. High detection threshold increases precision at the cost of low recall, on the contrary, low detection threshold decreases precision and increases recall. An average precision equal to 1 means that the detector is able to reach a perfect precision $(100\%)$ for all the values of recall, while an average precision of 0 means that the mode can not detect any object correctly. The testing results are presented in Table 4.

The best performance was achieved by YOLOv3, reaching 82.5% $mAP$ and 64% Recall on the average. The Tiny-YOLOv3 achieved 62.4% $mAP$ and an average Recall of 49%. All experiments were for 180k iterations. Figure 5 shows some test images during the training phase. It can be seen that YOLOv3 is able to achieve a better detection accuracy compared to Tiny-YOLOv3 at the expense of detection speed.

### B. EMBEDDED IMPLEMENTATION

After the training, we discuss the deployment of the models on the hardware platforms introduced in Section III-A2. In our experimental setup, we run Tiny-YOLOv3 and YOLOv3 in three different configurations. The first configuration consists of a NVIDIA Jetson Nano as a host machine, with a camera module attached to it. The second configuration comprises of a RPi 3B+ as a host machine, and Intel Movidius NCS, connected to it via USB 2.0 interface. A camera module is attached to the RPi 3B+ for detecting images. Intel NCS takes the input image, executes the neural network and sends the output back to RPi 3B+. The third configuration features a RPi 3B+ with a camera module attached to it. The corresponding schemes are shown in Figure 6.

At first, we demonstrated the performance of the NVIDIA Jetson Nano, RPi with NCS, RPi only while running the YOLOv3 and the Tiny-YOLOv3 in terms of the frame rate and the average power consumption at different modes (maximum absorbed power), which are shown in Table 5. Various parameters are considered as highlighted below

- FPS denotes the number of frames processed per time unit during the execution of the detector
- Power consumption denotes the average power consumption during the execution of the detector
- Power Efficiency denotes the FPS processed per watt.

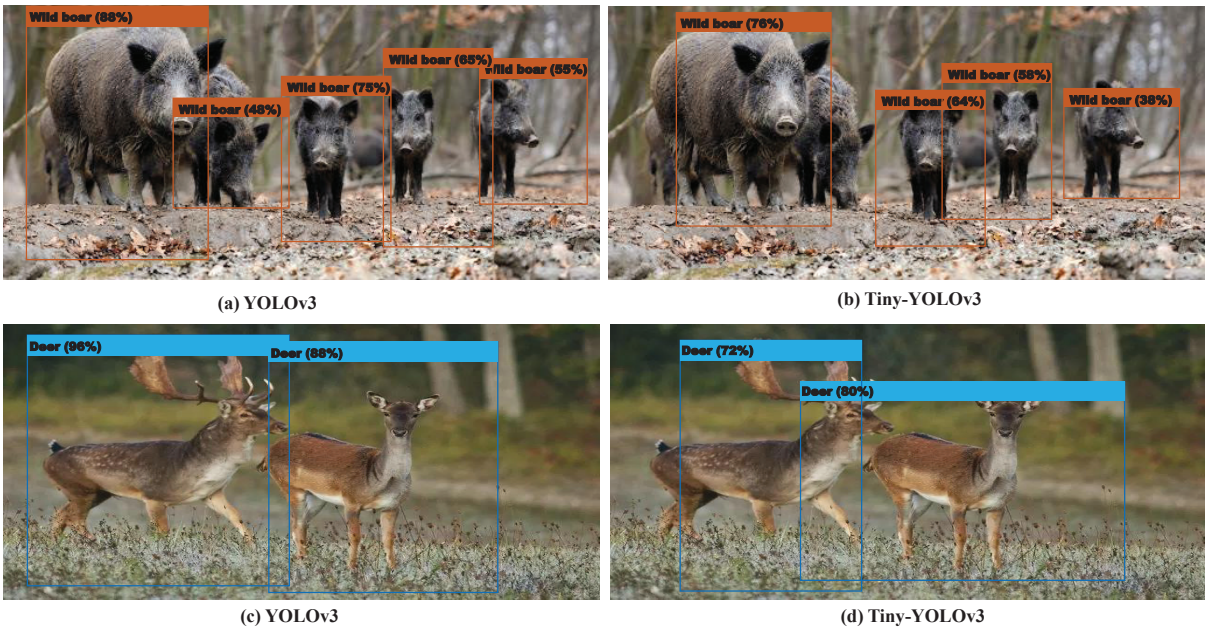
We measured the power consumption of the NVIDIA Jetson Nano executing both YOLOv3 and Tiny-YOLOv3 object detectors. Since NVIDIA Jetson Nano works at different modes at the expense of the computational capabilities, we tested all of them. We observed that, while operating at $10W$, the NVIDIA Jetson Nano switched off after a brief period of time. This is because $10W$ is insufficient when peripherals such as keyboard, mouse, camera are added to the NVIDIA Jetson Nano. However, we were able to collect the FPS value before the device switched off. We also observed that running NVIDIA Jetson Nano at other higher operational modes such

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



Table 4: A comparison of the performance of YOLO-V3 and Tiny-YOLOv3 in this study

| Recognition Models | | YOLOV3 | | Tiny-YOLOV3 | |
|---|---|---|---|---|---|
| | | Recall | AP | Recall | AP |
| Classes | Wild boar | 0.52 | 0.786 | 0.38 | 0.583 |
| | Deer | 0.76 | 0.864 | 0.6 | 0.6652 |
| Mean | | 0.64 | 0.825 | 0.49 | 0.6241 |




(a) YOLOv3 (b) Tiny-YOLOv3

(c) YOLOv3 (d) Tiny-YOLOv3

Figure 5: Qualitative results of some test images acquired during the data set training

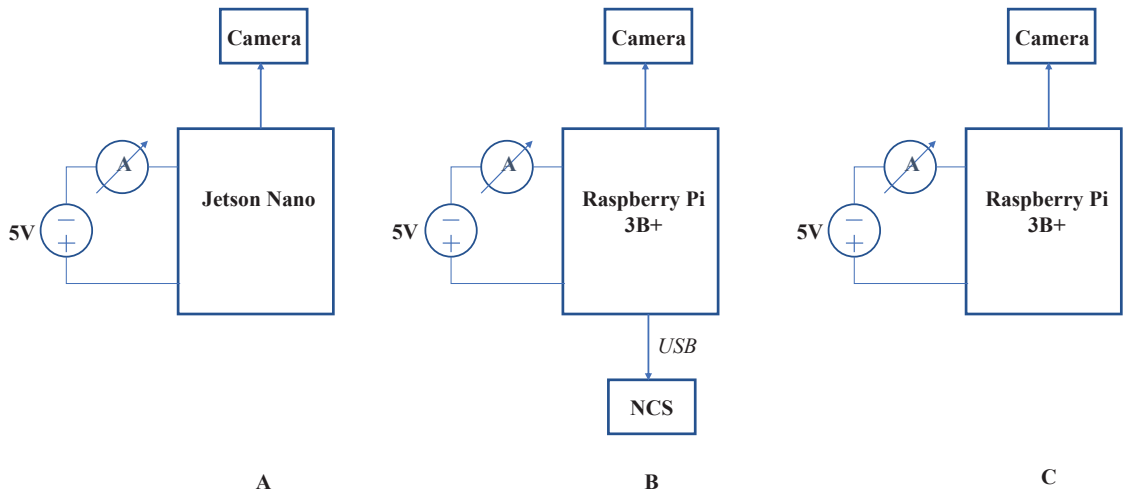



Figure 6: Detection System Schematic (A) NVIDIA Jetson Nano (B) RPi 3B+ and NCS (C) RPi 3B+

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3114503, IEEE Access



Table 5: Comparison between devices power consumption and performances with YOLOv3 and Tiny-YOLOv3

| Object Detector | Platform | Mode | P[W] | FPS | Power Efficiency, FPS/W |
|---|---|---|---|---|---|
| YOLOv3 | NVIDIA Jetson Nano | 20W | 17.32 | 4 | 0.231 |
| YOLOv3 | NVIDIA Jetson Nano | 15W | 12.94 | 3.2 | 0.247 |
| YOLOv3 | NVIDIA Jetson Nano | 10W | 8.5 | 3.2 | 0.376 |
| YOLOv3 | RPi 3B+ | at 5W | – | – | – |
| YOLOv3 | RPi 3B+ and NCS | 5W | – | – | – |
| Tiny-YOLOv3 | NVIDIA Jetson Nano | 20W | 16.70 | 14.8 | 0.832 |
| Tiny-YOLOv3 | NVIDIA Jetson Nano | 15W | 12.68 | 12.8 | 1.01 |
| Tiny-YOLOv3 | NVIDIA Jetson Nano | 10W | 8.3 | 10 | 1.205 |
| Tiny-YOLOv3 | RPi 3 B+ | 5W | 3.2 | 0.8 | 0.25 |
| Tiny-YOLOv3 | RPI 3 B+ and NCS | 5W | 3.2 | 4 | 1.25 |

as full power supply of 20W (5V/4A) through a barrel jack socket and 15W (5V/3A) works fine.

Results show that NVIDIA Jetson Nano running Tiny-YOLOv3 is the most performing solution, being able to reach 15FPS in the 20W operational mode. Notwithstanding, NVIDIA Jetson Nano in the 10W and 15W operational modes running the Tiny-YOLOv3 are also suitable for real time ungulates detection. With the NVIDIA Jetson Nano in different modalities running YOLOv3, the frame rate dropped to an average of 3FPS, which is acceptable in soft-real time contexts, such as ungulates detection and repelling.

With the RPi 3B+, running Tiny-YOLOv3, the performance is the worst reaching less than 1FPS, which is not suitable for real time applications. Adding the Intel Movidius NCS, we observe that the speed increases to 4FPS, but one can still arguably discuss if it is suitable for soft-real time applications or not. Despite the flexibility of the accelerators, the USB accelerator can not go beyond 5FPS in the best case.

We also compared the $HW$ platforms in terms of Cost/FPS ratio as shown in Table 6. In real-time application, cost is a major factor. Based on the previous results, Tiny-YOLOv3 was claimed to provide acceptable detection accuracy and detection speed, thus in Table 6 FPS values refer to Tiny-YOLOv3. The NVIDIA Jetson Nano appears to be the best choice taking into account the balance between cost and performance. Indeed, Intel NCS is a hardware USB accelerator, and can not be used as a stand-alone device for real time object detection. This indicates that an additional embedded device [53] must be purchased for the Intel NCS, thus increasing its final cost.

Moreover, we considered the CPU temperature of the embedded platform, – to determine how safely we can execute the ungulates detectors for an extended period of time in order to avoid overheating. The analysis of the CPU temperature while using NVIDIA Jetson Nano, RPi 3B+ & NCS are reported in Table 7. We compare the CPU temperature with the light (i.e. Tiny-YOLOv3) and the heavy (YOLOv3) detectors. From the Table 7, while executing YOLOv3 and Tiny-YOLOv3 on the out-of-the-box NVIDIA Jetson Nano, its CPU temperature reaches a peak of 51°C and an average of 38°C for the Tiny-YOLOv3 and a peak of 58°C and and an average of 46°C for YOLOv3 after hours of execution of the detector. Moreover, the performance remains stable and the CPU temperature does not reach its thermal limits, thus not operating in the thermal throttling mode. However, executing Tiny-YOLOv3 on RPi 3B+ with the NCS, its CPU reaches its critical limit, having a peak temperature of 93°C and an average of 70°C. This means that if we continue over a long period of time, we will face the problem of performance reduction, increased power consumption and CPU quality reduction in the worst case situation. Against thermal issues affecting the RPi 3B+, we attached a PWN fan, and we obtained an improvement in the CPU temperature leading to 35°C average with a peak of 60°C.

## VI. CONCLUSIONS

In this paper, we introduced a new application to defend crops from ungulate attacks that takes advantage of the latest technological developments in different ICT areas, such as AI, Edge Computing, IoT and LPWAN. The implementation of the application required the design and development of a complex system for intelligent animal repulsion, which integrates newly developed HW and SW components and allows to recognize the presence and species of ungulates in real time and also to avoid crop damages caused by the ungulates. The system has been designed taking into account application requirements (response time, accuracy, precision) , resource limitations (computational power, memory) and constraints (network coverage, energy consumption coming from the rural environment and the impact of all these factors on the HW and SW components, so as to achieve a satisfactory trade-off between requirements and performance.

Concerning animal recognition, YOLOv3 and Tiny-YOLOv3 have been evaluated and adopted as detectors for their ability to work in real-time at high performance as well as to adapt, after training their neural network models, to different edge computing platforms, such as RPi (with or without NCS) and Jetson Nano.

The first lesson learnt from this work is about the methodology to follow for the design and development of the HW and SW components enabling animal recognition and repulsion

Table 6: Cost/Performance analysis of the different embedded platforms.

| **Device** | **Cost($) Starting at** | **FPS$_{max}$** | **Cost/FPS** |
|---|---|---|---|
| NVIDIA Jetson Nano | 99 | 15 | 6.6 |
| RPi 3B+ with Intel NCS | 105 | 4 | 26.3 |
| RPi 3B+ | 35 | 1 | 35 |

Table 7: CPU temperature analysis of the different embedded platforms.

| Object Detector | Device | FAN | Peak Rating (°C) | Average Rating (°C) |
|---|---|---|---|---|
| YOLOv3 | NVIDIA Jetson Nano | NO | 58 | 46 |
| YOLOv3 | NVIDIA Jetson Nano | YES | 38 | 30 |
| Tiny-YOLOv3 | NVIDIA Jetson Nano | NO | 51 | 38 |
| Tiny-YOLOv3 | NVIDIA Jetson Nano | YES | 30 | 24 |
| Tiny-YOLOv3 | RPI 3B+ & NCS | NO | 93 | 70 |
| Tiny-YOLOv3 | RPI 3B+ & NCS | YES | 60 | 35 |

in real-time. One decade ago, Langendoen et al [60] wrote a foundational paper listing everything that went wrong in a precision agricultural deployment similar to our use case, which includes board failure, batteries running out, etc. What went right this time, is that the field of low-power devices for IoT applications has substantially evolved and has radically changed in that decade. However, although IoT technology has successfully transitioned from the academic to the commercial world, when uncommon use cases are considered, the development and deployment of applications still require accurate customization and complex integration activities.

The second lesson learnt concerns the key role played by the experiments to identify the best HW/SW configuration for this kind of application. Edge computing and IoT are the right technologies for AI-based applications, and using them makes a huge monetary difference. Moreover, as demonstrated throughout this work, Edge-AI technology is ready for use, but the deployment of innovative applications is still far from being easy and straightforward.

The third lesson learnt was about the architectural design of the Intelligent Animal Repelling System and, more specifically, the reliability of the activity detection mechanism. In some instances, when the PIR sensor misses detecting the movement of an ungulate, the device is configured in such a way that the camera acts as a backup in initiating the detection of the ungulate. In this case, the PIR sensor is not used and the real-time animal detection system on the embedded system is executed when the camera detects a target. Based on the category of the ungulate detected, the edge computing device executes its DNN software to identify the target, and if an animal is detected, it sends back a message to the Animal Repelling Device including the type of ultrasound to be generated according to the category of the animal. The "activity" message is also transmitted from the repeller device via LoRa to the LoRa gateway, which then forwards the packet to the TTN server.

Finally, this work delivers an innovative, perfectly working system that could be integrated with other HW devices and SW modules, thus opening the way to the introduction of unprecedented applications and services, not only in Smart Agriculture, that take advantage of both open hardware and cognitive approaches.

## ACKNOWLEDGMENT

The authors thank Alessandro Vaselli for his assistance in the preparation of the software used in this work.

## References

[1] M. De Clercq, A. Vats, and A. Biel, "Agriculture 4.0: The Future of Farming Technology," *Proceedings of the World Government Summit, Dubai, UAE*, pp. 11–13, 2018.

[2] Y. Liu, X. Ma, L. Shu, G. P. Hancke, and A. M. Abu-Mahfouz, "From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges," *IEEE Transactions on Industrial Informatics*, 2020.

[3] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A survey on the role of IoT in Agriculture for the Implementation of Smart Farming," *IEEE Access*, vol. 7, pp. 156 237–156 271, 2019.

[4] K. Kirkpatrick, "Technologizing Agriculture," *Communications of the ACM*, vol. 62, no. 2, pp. 14–16, 2019.

[5] A. Farooq, J. Hu, and X. Jia, "Analysis of Spectral Bands and Spatial Resolutions for Weed Classification via Deep Convolutional Neural Network," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 2, pp. 183–187, 2018.

[6] M. Apollonio, S. Ciuti, L. Pedrotti, and P. Banti, "Ungulates and their Management in Italy," *European ungulates and their management in the 21th century. Cambridge University Press, Cambridge*, pp. 475–505, 2010.

[7] A. Amici, F. Serrani, C. M. Rossi, and R. Primi, "Increase in Crop Damage Caused by Wild Boar (Sus scrofa l.): The "Refuge Effect"," *Agronomy for sustainable development*, vol. 32, no. 3, pp. 683–692, 2012.

[8] S. Giordano, I. Seitanidis, M. Ojo, D. Adami, and F. Vignoli, "IoT Solutions for Crop Protection Against Wild Animal Attacks," in *2018 IEEE International Conference on Environmental Engineering (EE)*. IEEE, 2018, pp. 1–5.

[9] M. O. Ojo, D. Adami, and S. Giordano, "Network Performance Evaluation of a LoRa-Based IoT System for Crop Protection Against Ungulates," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–6.

[10] H. HEEFNER and R. Heffner, "Auditory Perception," *Behaviour*, vol. 24, p. 81, 1992.

[11] H. E. Heffner and R. S. Heffner, "Hearing Ranges of Laboratory Animals," *Journal of the American Association for Laboratory Animal Science*, vol. 46, no. 1, pp. 20–22, 2007.

[12] N. Rantanen and R. Ewing III, "Principles of Ultrasound Application in Animals," *Veterinary Radiology*, vol. 22, no. 5, pp. 196–203, 1981.

[13] A. Levisse, M. Rios, W.-A. Simon, P.-E. Gaillardon, and D. Atienza, "Functionality Enhanced Memories for Edge-AI Embedded Systems," in *2019 19th Non-Volatile Memory Technology Symposium (NVMTS)*. IEEE, 2019, pp. 1–4.

[14] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying Machine Learning Techniques for Caching in Next-Generation Edge Networks: A Comprehensive Survey," *Journal of Network and Computer Applications*, p. 103005, 2021.

[15] W. Dai, H. Nishi, V. Vyatkin, V. Huang, Y. Shi, and X. Guan, "Industrial Edge Computing: Enabling Embedded Intelligence," *IEEE Industrial Electronics Magazine*, vol. 13, no. 4, pp. 48–56, 2019.

[16] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand Accelerating Deep Neural Network Inference via Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[17] E. Ahmed, A. Ahmed, I. Yaqoob, J. Shuja, A. Gani, M. Imran, and M. Shoaib, "Bringing Computation Closer Toward the User Network: Is Edge Computing the Solution?" *IEEE Communications Magazine*, vol. 55, no. 11, pp. 138–144, 2017.

[18] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[19] G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari, "LoRaFarM: a LoRaWAN-Based Smart Farming Modular IoT Architecture," *Sensors*, vol. 20, no. 7, p. 2028, 2020.

[20] M. O. Ojo, D. Adami, and S. Giordano, "Experimental Evaluation of a LoRa Wildlife Monitoring Network in a Forest Vegetation Area," *Future Internet*, vol. 13, no. 5, p. 115, 2021.

[21] G. Coral, "Datasheet: https://coral.withgoogle.com/tutorials/accelerator-datasheet/," *Accessed on: May 3, 2021*.

[22] I. Martinez-Alpiste, P. Casaseca-de-la Higuera, J. Alcaraz-Calero, C. Grecos, and Q. Wang, "Benchmarking Machine-Learning-Based Object Detection on a UAV and Mobile Platform," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv preprint*, arXiv:1506.01497, 2015.

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[25] S. Mittal and J. S. Vetter, "A Survey of CPU-GPU Heterogeneous Computing Techniques," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, pp. 1–35, 2015.

[26] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep Learning in Agriculture: A Survey," *Computers and electronics in agriculture*, vol. 147, pp. 70–90, 2018.

[27] Y. Yu, K. Zhang, D. Zhang, L. Yang, and T. Cui, "Optimized Faster R-CNN for Fruit Detection of Strawberry Harvesting Robot," in *2019 ASABE Annual International Meeting*. American Society of Agricultural and Biological Engineers, 2019, p. 1.

[28] R. Shi, T. Li, and Y. Yamaguchi, "An Attribution-Based Pruning Method for Real-Time Mango Detection with Yolo **N**etwork," *Computers and Electronics in Agriculture*, vol. 169, p. 105214, 2020.

[29] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple Detection during Different Growth Stages in Orchards using the Improved YOLO-V3 model," *Computers and electronics in agriculture*, vol. 157, pp. 417–426, 2019.

[30] N. P. Ramaiah, E. P. Ijjina, and C. K. Mohan, "Illumination Invariant Face Recognition using Convolutional Neural Networks," in *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*. IEEE, 2015, pp. 1–4.

[31] R. Singh and H. Om, "Newborn Face Recognition using Deep Convolutional Neural Network," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 19 005–19 015, 2017.

[32] T. Dobhal, V. Shitole, G. Thomas, and G. Navada, "Human Activity Recognition using Binary Motion Image and Deep Learning," *Procedia computer science*, vol. 58, pp. 178–185, 2015.

[33] H. A. Alwzwazy, H. M. Albehadili, Y. S. Alwan, and N. E. Islam, "Handwritten Digit Recognition using Convolutional Neural Networks," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 2, pp. 1101–1106, 2016.

[34] R.-C. Chen *et al.*, "Automatic License Plate Recognition via Sliding-Window Darknet-Yolo Deep Learning," *Image and Vision Computing*, vol. 87, pp. 47–56, 2019.

[35] J. H. Muribø, "Locating Sheep with YOLOv3," Master's thesis, NTNU, 2019.

[36] B. Jiang, Q. Wu, X. Yin, D. Wu, H. Song, and D. He, "FLYOLOv3 Deep Learning for Key Parts of Dairy Cow Body Detection," *Computers and Electronics in Agriculture*, vol. 166, p. 104982, 2019.

[37] J. Wang, N. Wang, L. Li, and Z. Ren, "Real-Time Behavior Detection and Judgment of Egg Breeders Based on YOLOv3," *Neural Computing and Applications*, vol. 32, no. 10, pp. 5471–5481, 2020.

[38] J. Wang, M. Shen, L. Liu, Y. Xu, and C. Okinda, "Recognition and Classification of Broiler Droppings Based on Deep Convolutional Neural Network," *Journal of Sensors*, vol. 2019, 2019.

[39] R. Y. Aburasain, E. A. Edirisinghe, and A. Albatay, "Drone-Based Cattle Detection using Deep Neural Networks," in *Intelligent Systems and Applications*, K. Arai, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2021, pp. 598–611.

[40] S.-J. Hong, Y. Han, S.-Y. Kim, A.-Y. Lee, and G. Kim, "Application of Deep-Learning Methods to Bird Detection using Unmanned Aerial Vehicle imagery," *Sensors*, vol. 19, no. 7, p. 1651, 2019.

[41] E. S. Kalhagen and Ø. L. Olsen, "Hierarchical Fish Species Detection in Real-Time Video using YOLO," Master's thesis, University of Agder, 2020.

[42] M. E. Hunniford and T. M. Widowski, "Nest Alternatives: Adding a Wire Partition to the Scratch Area Affects Nest Use and Nesting Behaviour of Laying Hens in Furnished Cages," *Applied Animal Behaviour Science*, vol. 186, pp. 29–34, 2017.

[43] A. Roihan, M. Hasanudin, and E. Sunandar, "Evaluation Methods of Bird Repellent Devices in Optimizing Crop Production in Agriculture," in *Journal of Physics: Conference Series*, vol. 1477, 2020, p. 032012.

[44] V. Partel, L. Nunes, P. Stansly, and Y. Ampatzidis, "Automated Vision-Based System for Monitoring Asian Citrus Psyllid in Orchards Utilizing Artificial Intelligence," *Computers and Electronics in Agriculture*, vol. 162, pp. 328–336, 2019.

[45] D. Shadrin, A. Menshchikov, D. Ermilov, and A. Somov, "Designing Future Precision Agriculture: Detection of Seeds Germination Using Artificial Intelligence on a Low-Power Embedded System," *IEEE Sensors Journal*, vol. 19, no. 23, pp. 11 573–11 582, 2019.

[46] G. Codeluppi, L. Davoli, and G. Ferrari, "Forecasting Air Temperature on Edge Devices with Embedded AI," *Sensors*, vol. 21, no. 12, p. 3973, 2021.

[47] L. Han, P. Tao, and R. R. Martin, "Livestock Detection in Aerial Images using a Fully Convolutional Network," *Computational Visual Media*, vol. 5, no. 2, pp. 221–228, 2019.

[48] G. Sales, K. Wilson, K. Spencer, and S. Milligan, "Environmental Ultrasound in Laboratories and Animal Houses: a Possible Cause for Concern in the Welfare and use of Laboratory Animals," *Laboratory Animals*, vol. 22, no. 4, pp. 369–375, 1988.

[49] E. B. Arnett, C. D. Hein, M. R. Schirmacher, M. M. Huso, and J. M. Szewczak, "Evaluating the Effectiveness of an Ultrasonic Acoustic Deterrent for Reducing Bat Fatalities at Wind Turbines," *PloS one*, vol. 8, no. 6, p. e65794, 2013.

[50] S. A. Schumake, "Electronic Rodent Repellent Devices: A Review of Efficacy Test Protocols and Regulatory Actions," in *National Wildlife Research Center Repellents Conference 1995*, 1995, p. 34.

[51] J. Ahmad, S. Mahamad, and N. S. M. Fadzil, "Integrated Mosquitoes Repellent with Ultrasound and Frequency," in *2018 IEEE Conference on Open Systems (ICOS)*. IEEE, 2018, pp. 42–46.

[52] Intel Software, "Intel movidius neural compute [online], available: https://software.intel.com/en-us/movidius-ncs," *Accessed on: May 3, 2021*.

[53] M. O. Ojo, S. Giordano, G. Procissi, and I. N. Seitanidis, "A Review of Low-end, Middle-end, and High-end IoT Devices," *IEEE Access*, vol. 6, pp. 70 528–70 554, 2018.

[54] NVIDIA Developer, "NVIDIA Jetson—Hardware for Every Situation [online], available: https://developer.nvidia.com/embedded/develop/hardware," *Accessed on: May 3, 2021*.

[55] S. Bargoti and J. Underwood, "Deep Fruit Detection in Orchards," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3626–3633.

[56] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[57] J. Redmon and A. Farhadi, "Yolo9000: Better, Faster, Stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[58] ——, "Yolov3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[60] K. Langendoen, A. Baggio, and O. Visser, "Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture," in *Proceedings 20th IEEE international parallel & distributed processing symposium*. IEEE, 2006, pp. 8–pp.

DAVIDE ADAMI received the degree in electronic engineering from the University of Pisa, Italy, in 1992. From 1993 to 1997, he was with Consorzio Pisa Ricerche, taking part in a lot of research projects funded by the EU, such as the MAESTRO ACTS Project. In 1997, he joined CNIT, where he is a Senior Researcher in the field of telecommunication networks. He has conducted research for several research projects funded by ASI, ESA, EU (FP6 RINGRID, FP7 DORII, FP7 OFELIA, FP7 Fed4FIRE, FP7 SCOUT), and the Italian MIUR. He has authored many papers in scientific journals and international conference proceedings. His research interests mainly concern software defined networking and network function virtualization in cloud data centers, the design and development of new innovative solutions for the integration of Cloud applications and networks architectures providing QoS support.

MIKE O. OJO (Member, IEEE) received the M.Sc degree in Telecommunications Engineering from Politecnico di Milano, Milan, Italy in 2014, and a Ph.D degree in Information Engineering from University of Pisa, Pisa, Italy in 2019. He is currently a Postdoctoral Research Fellow with the University of Turin, Italy. His research interests include design, optimization and performance evaluation of Industrial Internet of things networks, low power wide area networks, virtualization and cloud computing.

STEFANO GIORDANO (M'89, SM'10) received the Laurea (cum laude) degree in electronics engineering and the Ph.D. degree in information engineering from the University of Pisa in 1990 and 1994, respectively. He is currently a Full Professor with the Department of Information Engineering, University of Pisa where he is responsible for the telecommunication networks laboratories. He is the Representative of the University of Pisa in the Scientific Committee of CNIT (the Italian National Consortium for Telecommunications) and the University of Pisa in the GTTI (Group for Telecommunications and Information Theory). He has authored over 400 papers in international conferences and journals. He is a member of IFIP WG 6.3, Internet Society since its foundation in 1992, and a member of the Board of Directors of its Italian charter. He is the former Chair of the Communication Systems Integration and Modelling Technical Committee. He founded Juniper Networks, the first European Juniper Networks Higher Learning Center. He is chairing a Standardization Research Group of Comsoc on IoT Communications and Networking Infrastructure. He has been a reviewer for the NSF in the U.S., the EU, the Italian Ministry of Industry, and the Italian Ministry of Research (member of the albo of experts of the ministry). He is an associate editor for several journals. He has been the general chair, the TPC chair, and a TPC member for many international conferences. He was co-founder of three start-ups (Nextworks, Netresults, Natech) and co-founder of the Cubit Consortium where at present is president of the Scientific and Technical Committee).