A Mobile Application for Detecting and Monitoring the Development Stages of Wild Flowers and Plants

João Videira [1], Pedro D. Gaspar [2,3], Vasco N. G. J. Soares [1,4*] and João M. L. P. Caldeira [1,4]
[1]Polytechnic Institute of Castelo Branco, Av. Pedro Álvares Cabral nº 12, 6000-084 Castelo Branco, Portugal
[2]Department of Electromechanical Engineering, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
[3]C-MAST Center for Mechanical and Aerospace Science and Technologies, University of Beira Interior, 6201-001 Covilhã, Portugal
[4]Instituto de Telecomunicações, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
jvideira@ipcbcampus.pt, dinis@ubi.pt, vasco.g.soares@ipcb.pt, jcaldeira@ipcb.pt
*Corresponding author: g.soares@ipcb.pt

Keywords: wild flowers and plants, development stages, computer vision, convolutional neural networks, YOLOv4, YOLOv4-tiny, mobile app

Received: Januar 7, 2024

Wild flowers and plants appear spontaneously. They form the ecological basis on which life depends. They play a fundamental role in the regeneration of natural life and the balance of ecological systems. However, this irreplaceable natural heritage is at risk of being lost due to human activity and climate change. The work presented in this paper contributes to the conservation effort. It is based on a previous study by the same authors, which identified computer vision as a suitable technological platform for detecting and monitoring the development stages of wild flowers and plants. It describes the process of developing a mobile application that uses YOLOv4 and YOLOv4-tiny convolutional neural networks to detect the stages of development of wild flowers and plants. This application could be used by visitors in a nature park to provide information and raise awareness about the wild flowers and plants they find along the roads and trails.

Povzetek: Raziskava uvaja mobilno aplikacijo z uporabo konvolucijskih nevronskih mrež YOLOv4 za prepoznavanje razvojnih stopenj divjih rastlin, prispevajoč k ohranjanju naravne dediščine.

1 Introduction

Plants are recognized as a vital part of the world's biological diversity and an essential resource for the planet. They play a fundamental role in maintaining basic ecosystem functions and are indispensable for the survival of animal life on our planet [1]. While agricultural plants provide food and basic fibers, many wild plants are of great economic and cultural importance and have enormous potential, serving as food, medicine, fuel, clothing and common shelter [2], [3].

Given the growing number of endogenous and/or wild flowers and plants at risk of extinction and in decline due to climate change and the impact of human action [4], [5], there is an urgent need to contribute technological solutions for their conservation and preservation. Detecting, monitoring and following the development stages of endogenous and/or wild flowers and plants can alert biologists and researchers linked to the various areas of biodiversity to possible problems with the surrounding environment, which can help them make more informed decisions about how to manage and protect natural parks or preserved areas. On the other hand, it can support and help inform tourists and the community in general about wild flowers and plants found along roads and trails. This promotion of awareness about wild flowers and plants is aimed at promoting environmental sustainability, but also the development of economic and cultural activities in regions where these plants grow.

The development stages of wild flowers and plants can be classified as follows [6]: 1) sprout: this stage typically takes place underground, where the plant begins to grow from its seed; 2) seedling: this stage is characterized by the spread of roots and the appearance of the first leaves; 3) vegetative: this stage is identified by the development of stems and foliage; 4) budding: this stage can be identified by the appearance of buds on the plant; 5) flowering: this stage is recognized by the appearance of flowers, which consequently causes pollination and can be accompanied by the appearance of fruit in the early stages; 6) ripening: this stage is identified by the appearance of ripe fruit. The sprout stage cannot be identified by computer vision techniques, as it takes place underground.

This work follows on from the conclusions presented in a previous paper by the same authors [7], which analyzed computer vision as a suitable technological platform for detecting and monitoring the development stages of wild flowers and plants. It presented a survey of the research in this field and applications related to plant identification and plant disease detection. The most

promising computer vision techniques were identified, and open problems and challenges were discussed.

The work presented in this article is one of the stages of an ongoing project which aims to develop a mobile application and system based on computer vision techniques to detect and monitor the stages of development of wild flowers and plants. The application can be used by visitors to a nature park to provide information and raise awareness about the wild flowers and plants they encounter along the roads and trails. The system will allow scientists and biologists, or the merely curious, to remotely monitor and collect information on the stages of development of wild flowers and plants.

Although there are already mobile applications that can identify plant species, such as those available in [8], [9], to the best of the authors' knowledge, at the time of writing this article, there is still no mobile application capable of classifying the developmental stages of wild flowers and plants. Therefore, the main contributions of this article are: 1) the creation of a dataset with the stages of development of a wild plant; 2) a performance evaluation study of the convolutional neural network (CNN) models YOLOv4 and YOLOv4-tiny for detecting the stages of development of this wild plant; 3) the description of the process of developing a mobile application, compatible with the Android platform, which uses these CNNs.

This application is aimed at visitors or workers in nature parks. It works by capturing an image of the plant or flower. Then, using these computer vision techniques, it will be able to identify the species of the wild plant, as well as determine its stage of development, and to provide additional information about them.

The rest of the paper is organized as follows. Section 2 introduces the main concepts of the computer vision techniques used in the context of this work and presents a performance assessment. Section 3 describes the implementation process and the operation of the mobile application developed in the context of this work. Finally, Section 4 concludes the article and presents future work.

1 Computer vision techniques

Computer vision techniques include a variety of algorithms, models and procedures that allow computers to analyze visual data such as photographs and videos [10] and perform tasks such as object detection and classification [11]. Object detection is the task of locating an object in the visual input, while object classification involves assigning a classification to the objects detected in that same input [12]. These concepts are illustrated in Figure 1. Although these tasks differ, deep learning is often used for both.

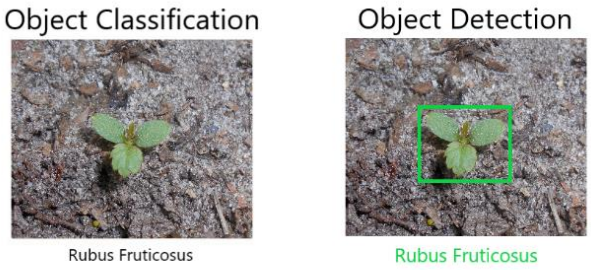

Figure 1: Illustration of object classification and detection concepts.

Deep learning is a subfield of machine learning, which focuses on the creation and training of convolutional neural networks. This training is carried out by learning patterns from a large volume of data [13], such as a dataset of images. This training is possible because CNNs are made up of layers of interconnected nodes, which simulate the behavior of neurons in a human brain [13]. These concepts are illustrated in Figure 2. Training CNNs makes it possible to detect patterns, and consequently to detect and classify objects [14].

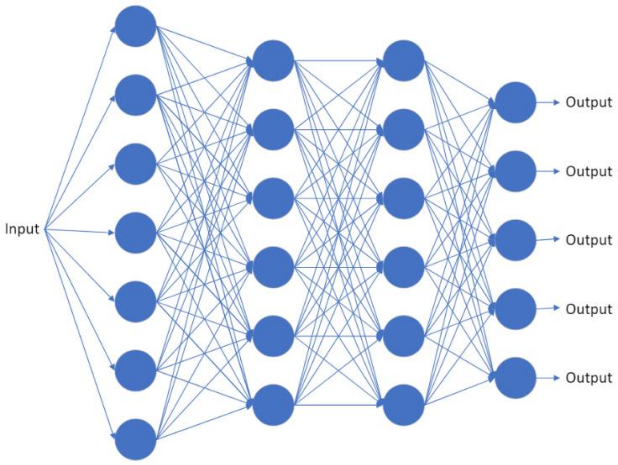

Figure 2: The structure of a convolutional neural network.

In a previous work [7] by the same authors of this article, it was found that over the years, various CNN models have been considered in the literature for the tasks of plant identification and plant disease detection. Given the results reported in [15], [16] and [17], it was concluded that YOLOv3 or YOLOv4 would be the best option to consider when developing a mobile application to detect the development stages of wild flowers and plants. These models are fast, require relatively little processing capacity and allow results to be obtained in real time. These characteristics fit perfectly with the requirements of the mobile application presented in this paper.

1.1 YOLOv4 e YOLOv4-tiny

The YOLO (You Only Look Once) model [18] analyses images quickly by dividing an image into a grid, predicting the bounding boxes, confidence levels and class probabilities of the objects. The result is a set of objects bounding boxes, with class names and confidence levels [18]. These concepts are illustrated in Figure 3.

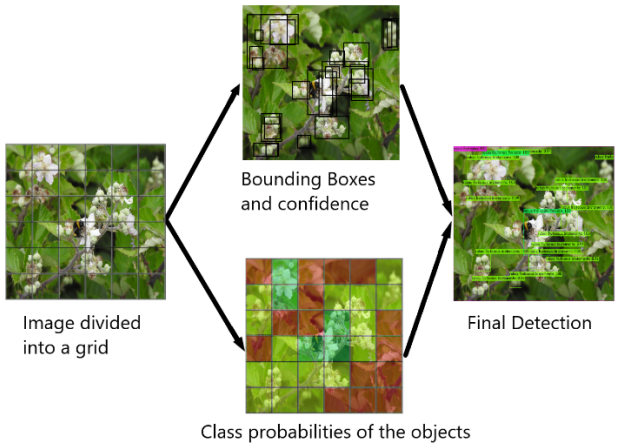

Figure 2: YOLO model detection process.

The YOLOv4 model consists of 3 components: backbone, neck and head. YOLOv4 uses CNN CSPDarkNet53 in the backbone, while YOLOv3 uses DarkNet53 [19]. This component is responsible for feature extraction, which is the process of transforming data into numerical values. In YOLOv4, the neck component uses Path Aggregation Network (PAN) [17] to extract feature maps, while YOLOv3 uses Feature Pyramid Extraction (FPN). Finally, the head component consists of applying anchor boxes to the feature map extracted by PAN. These anchor boxes are used to capture the objects and contain a prediction value [20]. At this stage three heads can be used to identify objects of various sizes, after the feature maps of various scales are joined and subjected to a convolution operation [21].

The YOLOv4-tiny model is a simplified version of YOLOv4 [22]. The first difference in this model is the use of the CSPDarknet53-tiny CNN [23]. The neck component of YOLOv4-tiny uses the Feature Pyramid Network (FPN) structure [23], a design that improves object detection accuracy and increases detection speed [24]. Another difference from YOLOv4 is that YOLOv4-tiny uses only two heads instead of three [23]. This modification could potentially pose challenges when detecting objects at extreme scales, such as very small objects [22]. Despite this limitation, the integration of CSPDarknet53-tiny and FPN into YOLOv4-tiny contributes to its overall performance, allowing it to perform object detection tasks with less computing power and greater speed.

1.2 Performance evaluation

This subsection focuses on evaluating the performance of the YOLOv4 and YOLOv4-tiny CNN models for detecting the developmental stages of wild flowers and plants. To this end, the dataset created as part of this work, the benchmark scenario, the performance metrics considered, and the experimental results are presented next.

1.2.1 Dataset description

To the best of the authors' knowledge, there are no available datasets with images of the developmental stage's wild flowers and plants. Therefore, it was necessary to create a dataset with the developmental stages of a specific wild plant in order to train and test the models. The plant selected for proof of concept and testing was *Rubus Fruticosus*, also known as "bramble" or "blackberry". This choice was because it is a common wild plant and thus with many images available.

The dataset was categorized into 6 different classes, each representing a stage of development of the wild plant. The initial stage, called seedling, marks the appearance of roots and the initial appearance of leaves. This is followed by the vegetative stage, characterized by the development of stems and foliage. The budding stage shows the appearance of buds on the plant, heralding the next stage of flowering. The flowering stage is identified by the presence of flowers, often accompanied by the appearance of the first fruits. The ripening stage is characterized by the appearance of fruit on the plant. Finally, the class "*Rubus Fruticosus*" is used to identify this plant species. All the images used in this dataset were obtained from the INaturalist [25] and biodiversity4all [26], platforms, taking advantage of the large number of images present on them. This dataset can be found on the Kaggle platform in the following link [27]. Figure 4 shows an example of each of the developmental stages described.

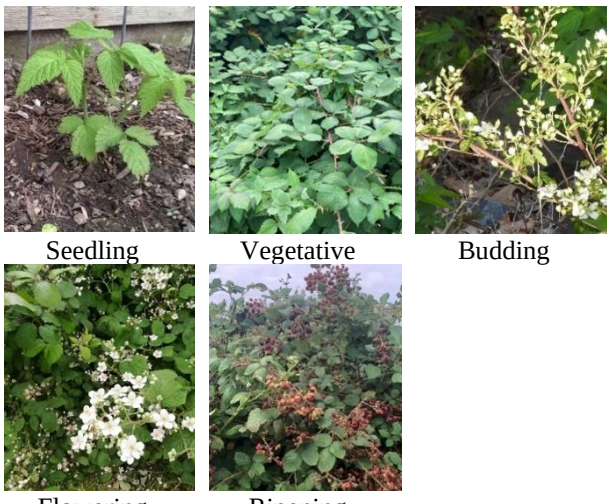

Figure 4: The development stages of the wild plant Rubus Fruticosus.

To train and validate the CNN models, the dataset was separated into train and test. In the training dataset, each development stage category was created with approximately 100 images. An exception was the "Budding" category, which, given the limited availability of images, only contains 80 images for training. It is

important to note that the "*Rubus Fruticosus*" class has 490 images because this is the total number of images of all the development stages for this plant species.

To create the test dataset, 20 images were used for each development stage. This distribution of the number of images for training and testing each development stage guarantees sufficient data for training the models and evaluating their performance. Table 1 shows the number of images available for training and testing for each class (i.e., development stage).

Table 1: Number of images for each class (i.e., development stage) in the training and test dataset.

Class	Train	Test
Seedling	100	20
Vegetative	100	20
Budding	80	20
Flowering	110	20
Ripening	100	20
Rubus Fruticosus	490	100

Next, the Yolo_Label tool [28] was used to create the annotations for each image. This task tells the models the location of the objects and their classification. Then, they can then be trained with this data and their performance can be evaluated. Figure 5 illustrates this task.

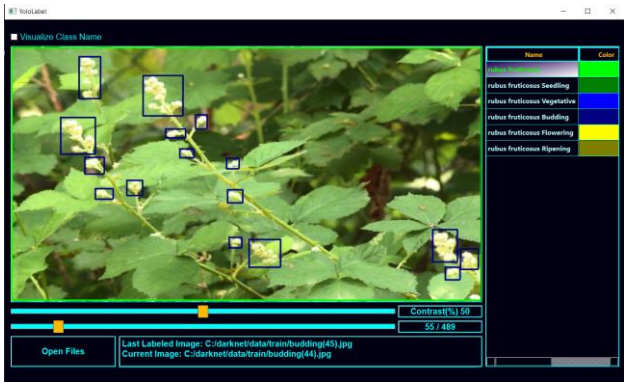

Fig. 3. The process of creating an annotation on a dataset image with the Yolo_Label tool.

1.2.2 Benchmark scenario

Both models were trained and tested on the same device. To do this, the darknet code [29] was downloaded to carry out the necessary training and tests. The test environment was hosted on a device with an AMD Ryzen 5 4600H CPU, 16 GB of RAM, and an NVIDIA GeForce GTX 1650 GPU which increased the computing power required for deep learning operations.

This test environment allowed an unbiased comparison of the accuracy, processing speed and efficiency of the YOLOv4 and YOLOv4-tiny models.

1.2.3 Performance metrics

To assess the performance of YOLOv4 and YOLOv4-tiny in the task of object detection and classification, the models were trained with 12,000 iterations with batches of 64 images, following the instructions in [29]. This number of iterations is equivalent to approximately 1567 epochs, according to the formula show in (1).

$$Epochs = \frac{number\ of\ iterations}{\frac{number\ of\ images\ in\ train}{batch}} \quad (1)$$

Once the training was complete, the trained models with the best average precision (mAP) were selected. This metric is calculated according to the formula shown in (2) and considers the accuracy of each class (APk) and the number of classes (n). The mAP is the metric commonly used to compare the performance of CNN models such as YOLO.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} APk \quad (2)$$

It is also interesting to analyze in CNN models, the average loss obtained by calculating the average of the total loss of several batches in a dataset, and the test time required to determine the mAP.

Overfitting [30] is a challenge when training CNNs. It occurs when a model becomes excessively specialized in the training data it has been exposed to, to the point of memorizing the details of the data. As a result, the model performs remarkably well on the training data, but fails when confronted with new and unseen data [30]. To try to solve this problem, it is essential to find a balance between accurately capturing significant patterns and avoiding an overly complex model that adapts too much to the training data. Figure 6 illustrates the concept of overfitting.

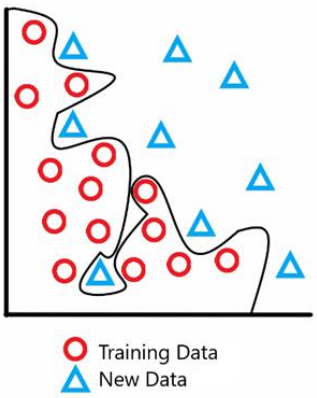

Figure 4: Illustration of the concept of overfitting.

The solution found to avoid overfitting was the Early Stopping [31]. This solution consists of monitoring the model's performance with a validation dataset while it is being trained. As training is carried out, signs of performance degradation or stagnation are looked for. When it is detected that the model's performance on the validation dataset is no longer improving or is getting worse, which indicates a greater number of errors, the training is stopped. This prevents the model from specializing on the training dataset and helps to ensure that the model maintains good performance against new data

[32]. With this solution, the training of both models was stopped when they showed a degradation or stagnation in performance. Figure 7 illustrates this concept.

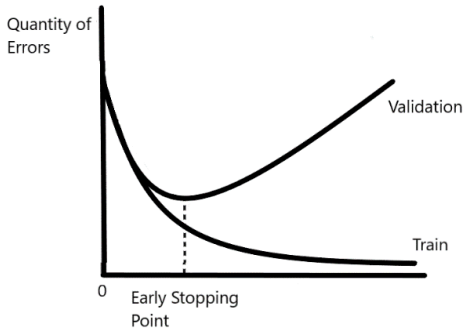

Figure 5: Illustration of the early stopping approach.

1.2.4 Results and discussion

The YOLOv4 and YOLOv4-tiny models went through 12000 training iterations. The model weights were then selected based on the maximum mAP value achieved, for use in the mobile application presented in the next section.

Figure 8 shows the results of the YOLOv4 model training process. The final mAP and average loss values were 74,83% and 1,4794, respectively. It was also concluded that the model achieved its best performance at around 7600 iterations with an accuracy of 77,18%. This performance demonstrates the model's effectiveness in identifying and classifying objects.

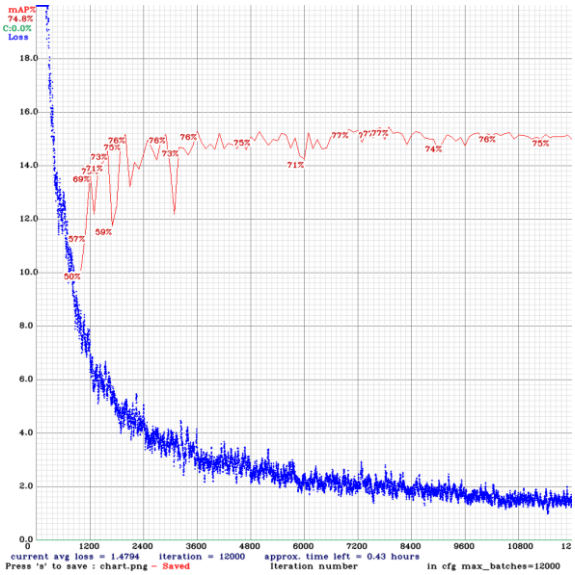

Figure 6: Results of the YOLOv4 training process.

To further analyze the performance of the YOLOv4 model, the accuracy of each class (i.e., development stage) was evaluated. Table 2 shows the accuracy results recorded for each class in the YOLOv4 model. Although each class exceeded the average accuracy, it was observed that the "Ripening" class had a lower average accuracy with a value of 26,24%. This suggests possible areas of future improvement for object recognition in this specific class.

Table 2: Accuracy results for each class (i.e., development stage) in the YOLOv4 and YOLOv4-tiny models.

	mAP	
Class	**YOLOv4**	**YOLOv4-tiny**
Rubus Fruticosus	92,52%	86,03%
Seedling	82,13%	69,86%
Vegetative	100%	72,40%
Budding	82,71%	74,99%
Flowering	79,44%	83,51%
Ripening	26,24%	31,50%

Figure 9 shows the results of the training process for the YOLOv4-tiny model. The model finished its training with a mAP value of 65,43% and an average loss of 0,3566. However, the model achieved its best performance at around 7000 iterations with an accuracy of 69,72%. Therefore, it can be concluded that this model is very effective at classifying and detecting objects, even though it is a simplified version of the YOLOv4 model.

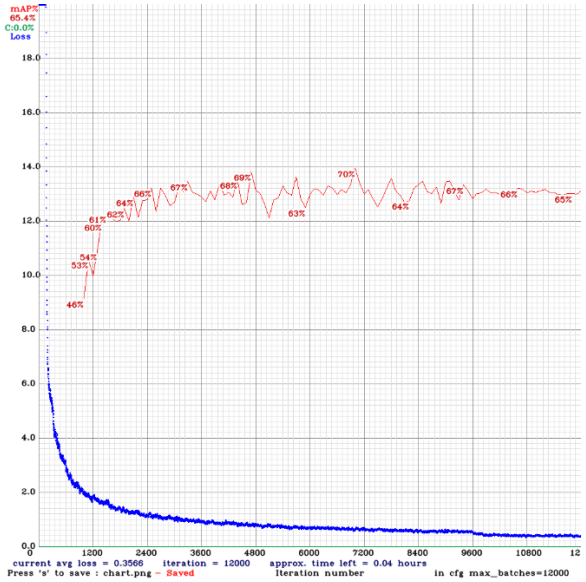

Figure 7: Results of the YOLOv4-tiny training process.

The performance of the YOLOv4-tiny model was also analyzed in depth. Table 2 also shows the accuracy results recorded for each class in this model. It was found that, although most of the classes exceeded the average accuracy, the "Ripening" development stage again showed a considerably lower average accuracy of 31,50%. This result reinforces the conclusion that both models have greater difficulty in recognizing this stage of development.

In view of these results, an explanation was sought as to why the "Ripening" class had a manifestly lower performance when compared to the other classes in both models. The visual appearance of the fruits of this wild plant, which can be red or black depending on the stage

they are at, is how this development stage is detected. This variation in the appearance of the fruit adds complexity and may certainly play an important role in the lower average accuracy value recorded.

Therefore, it is believed that compiling a dataset with a greater number and variety of photographs that capture the various stages of fruit development can help resolve this constraint and contribute to improving model performance. This will improve the ability to recognize and generalize patterns associated with the different appearances of this fruit, resulting in greater accuracy in detecting this stage.

The performance evaluation also showed that the YOLOv4 model completed the test to determine the mAP in 5 seconds, i.e., it detected all the test data in 5 seconds, while the YOLOv4-tiny model completed the same test in 1 second. Figure 10 shows the time and mAP results of the tests carried out.

```
 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.771755, or 77.18 %
Total Detection Time: 5 Seconds
```

```
 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.697157, or 69.72 %
Total Detection Time: 1 Seconds
```

Figure 10: Accuracy results and total detection and classification times for YOLOv4 and YOLOv4-tiny respectively.

This difference in test time results suggests that the YOLOv4-tiny model requires less computing power than YOLOv4. YOLOv4-tiny may sacrifice some detection accuracy in favor of processing speed. But the faster processing of the test dataset demonstrates its suitability for scenarios where fast response and real-time performance are key, and/or there are limited computing resources.

2 Mobile application

This section describes the development of the mobile application for Android called "MontanhaVivaApp". It can be used by visitors in a nature park to interactively promote knowledge about the development stages of different wild flowers and plants found along the roads and trails, contributing to their conservation and preservation.

2.1 Methodology

To develop this application, the User-Centred Design and Iterative Development methodologies were adopted. User-Centered Design [33] gives priority to end users throughout the development process, to understand their needs and preferences. This involves techniques such as usability testing to ensure that the resulting product is in line with user expectations. Iterative Development [34] emphasizes continuous improvement, through repeated cycles of design, implementation, and evaluation.

The User-Centered Design methodology was applied to the creation of a prototype in Adobe XD [35]. It enabled usability tests to be carried out on the mobile application's interface, to find and correct shortcomings, and to determine missing features. The usability testing phase is described in subsection 3.4.

The Iterative Development methodology was applied in the development phase. The project was divided into several reasonably sized tasks, each focusing on a different set of functionalities. The continuous integration of new code and testing of new features ensured that the development went smoothly. This development phase is explained in subsection 3.5.

2.2 Requirement analysis

User requirements refer to the needs and expectations of end users, determining the desired features and interactions [36]. Requirements can be divided into two categories: functional requirements and non-functional requirements.

Functional requirements specify the precise tasks that an application must perform, i.e., the main functionalities and user interactions. To assist in the process of identifying functional requirements, a use case diagram was drawn, shown in Figure 11 in Unified Modeling Language (UML) notation [37]. Use cases represent the interactions between users and the application. They provide a means of capturing the system's requirements and identify the functionalities to which the user has access.

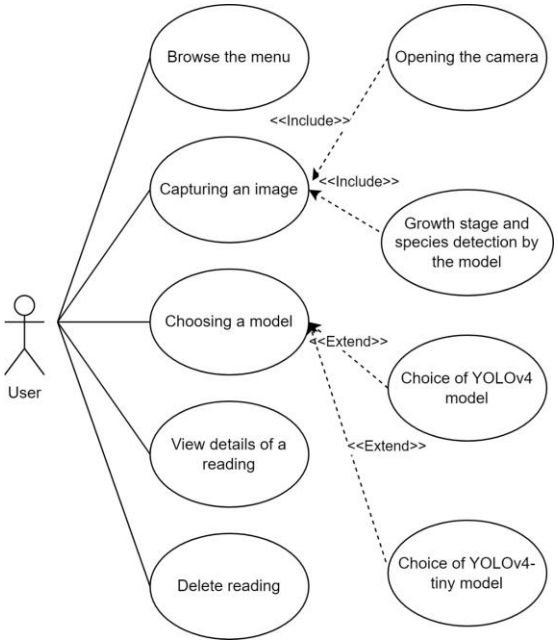

Figure 8: The use case diagram of the application.

Next, the functional requirements are described with the name of the interaction, a description, its preconditions, and outputs. Table 3 summarizes this information.

1) Navigation through the main menu, allowing the user to view previous readings. Its precondition is that the application is initialized. As an output of the requirement, the user can use all the functions found in the main menu,

such as taking a photo to take a reading or checking the details of a previous reading.

2) Capturing a photo to be analyzed by the CNN model. The precondition for this requirement is that the user clicks on the button available in the main menu to take a photo. As an output of the requirement, the image is processed by the chosen model.

3) The choice of CNN template, which can be selected from the top section of the main menu. The precondition for this requirement is that the user is in the main menu. The output of the requirement is that the user can use all the functions found in the main menu.

4) The use of the YOLOv4 model. The precondition for this requirement is that the user has selected this CNN model in the main menu and has taken a photograph. As an output of the requirement, the user returns to the main menu where the result of the new reading is now available.

5) The use of the YOLOv4-tiny template. The precondition for this requirement is that the user has selected this CNN model from the main menu and captured a photo. As an output of the requirement, the user returns to the main menu where the result of the new reading is now available.

6) Seeing all the details of a reading. The precondition for this requirement is that the user has selected one of the readings in the main menu. To exit the requirement, the user can delete the reading or return to the main menu.

7) Delete a specific reading using a button on the details page of a reading. This requirement has the precondition of being on the details page of a reading. As an exit from the requirement, the user returns to the main menu.

Table 3: Summary of functional requirements.

Name	Precondition	Exit
Browse the menu	Initialize the application.	Use of any functionality found in the menu.
Capturing an image	Click on the button to capture a photo.	Return to the menu.
Choosing a model	Find yourself in the menu.	Use of any functionality found in the menu.
Choice of YOLOv4 model	Find yourself in the menu.	Use of any functionality found in the menu.
Choice of YOLOv4-tiny model	Find yourself in the menu.	Use of any functionality found in the menu.
View details of a reading	Click on one of the readings.	Delete reading. Return to the menu.
Delete reading	Finding yourself on the details page of a reading.	Return to the menu.

The application also has non-functional requirements, which include attributes such as performance, feedback on errors and ease of use.

An intuitive interface refers to the ease of use and interaction of the user interface. It denotes the system's ability to facilitate user involvement and navigation, without the need for extensive training or guidance. It contributes to the application's ease of use by minimizing the learning curve, allowing users to quickly understand the interface's functionalities and access them effortlessly.

Performance is the application's ability to perform tasks effectively and efficiently, even under varying conditions and workloads. It includes factors such as responsiveness, speed and resource utilization. A well-performing application meets user expectations by providing quick responses, fast data processing and smooth functionality across different devices and usage scenarios.

Error feedback refers to the application's ability to give clear and informative answers to users when errors or exceptions occur during its operation. This requirement underlines the importance of maintaining a user-friendly environment, even in the presence of unforeseen problems. Effective error feedback provides users with concise and understandable explanations of the problems encountered, suggests potential solutions and guides them towards troubleshooting or making informed decisions. By ensuring informative error feedback, the application facilitates user understanding, minimizes frustration and promotes a positive user experience.

A scalable database refers to the importance of a structured and adaptable database architecture to which information about other species of wild flowers and plants and their development stages can be added, while maintaining its efficiency and the organization of the data.

These requirements combined form a framework for developing an application that aligns with user expectations, provides the desired functionalities and meets quality expectations.

In addition to the user requirements, the platform requirements must also be addressed. As this application was developed using Android Studio [38], it requires an Android operating system. More specifically, as minSdkVersion 24 was used, the application works on Android versions 7 and later. Installation requires approximately 80 MB of storage, and at least 2 GB of RAM is recommended for optimum performance. A working camera is also essential for capturing photographs. In addition to these requirements, users who choose to use YOLOv4 need an Internet connection to use it.

2.3 Technologies and architecture of the mobile app

The "MontanhaVivaApp" mobile application was implemented using the Android Studio integrated development environment (IDE) (electric eel 2022.1.1) [38] and the Java programming language. The application uses an SQLite database [39] and the Structured Query Language (SQL). The database was designed to hold a

large amount of data on the different species of wild flowers and plants and their development stages.

The application uses the YOLOv4 and YOLOv4-tiny trained models to detect and classify images of wild flowers and plants in order to identify the species and their development stage. The YOLOv4-tiny model was implemented locally in the application, as it requires less computing power, allowing it to be used by a mobile device. The YOLOv4 model, which requires more computing power, was implemented remotely and is used via an application programming interface (API). To create this API, the Flask platform in version 2.3.3 [40] was used, together with Python language. Figure 12 shows the diagram of the architecture described.

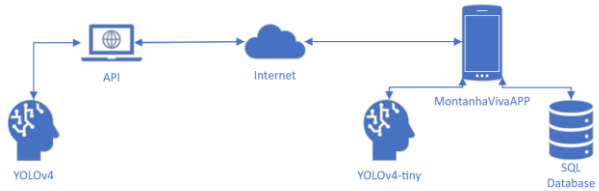

Figure 9: The MontanhaVivaAPP application architecture.

Since the application will be used by visitors in nature parks, they may be in remote locations where Internet access may be scarce or non-existent. Therefore, it was decided to implement the YOLOv4-tiny model and the database locally. This way, even if a user is in an area without network coverage and captures a photograph, the application will be able to identify the wild plant and its development stage, providing a reading with the plant's information. Figure 13 shows a sequence diagram describing the process of creating a new reading locally, using the YOLOv4-tiny model.

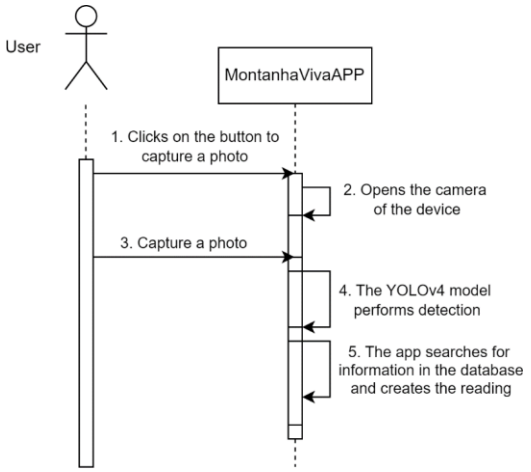

Fig. 10. The sequence diagram illustrating the process of creating a new reading locally using the YOLOv4-tiny model.

In addition, the application offers another option for detection and classification using the YOLOv4 model, for users who have access to the Internet. The user may prefer to use this model because it is more accurate, as discussed in a subsection above. This model is available to the user via an API, which will receive a photograph and use the YOLOv4 model to perform the detection and classification, returning the results to the application (i.e., species and development stage). Then, the application will create a new reading from this data. Figure 14 shows a sequence diagram illustrating the process of creating a new reading locally, using the YOLOv4 model.

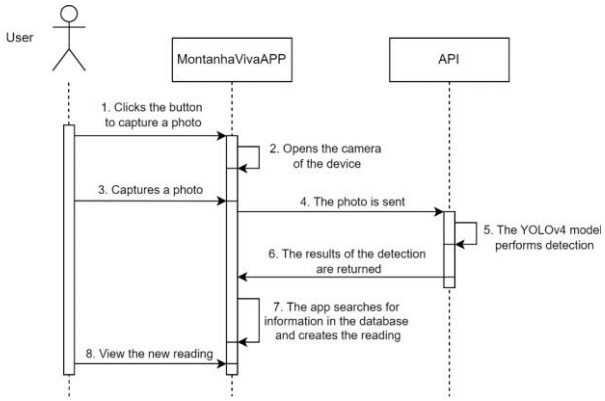

Figure 11: The sequence diagram illustrating the process of creating a new reading using the API with the YOLOv4 model.

2.4 Usability tests

Before starting the application development process, an Adobe XD prototype was developed to simulate the application's functionalities. This prototype is available at [41]. The prototype made it possible to analyze how users would interact with the application, as well as to identify missing features to improve the application.

The initial interface of the prototype is the main menu, which serves as the entry point to the application. This menu contains all the readings of wild plants previously taken by the user. These readings are separated into cells. Each cell displays a summary set of information, including the scientific name, the common name, the stage of development, the period of that stage and the date of the reading. The main menu also has a camera button in the bottom right-hand corner, which simulates the process of capturing a photograph of a wild plant for detection and classification. After successfully capturing the photo, the user returns to the main menu interface, where the newly generated reading allows further exploration and management. Figure 15 shows the process of creating a new reading and shows these described interfaces.

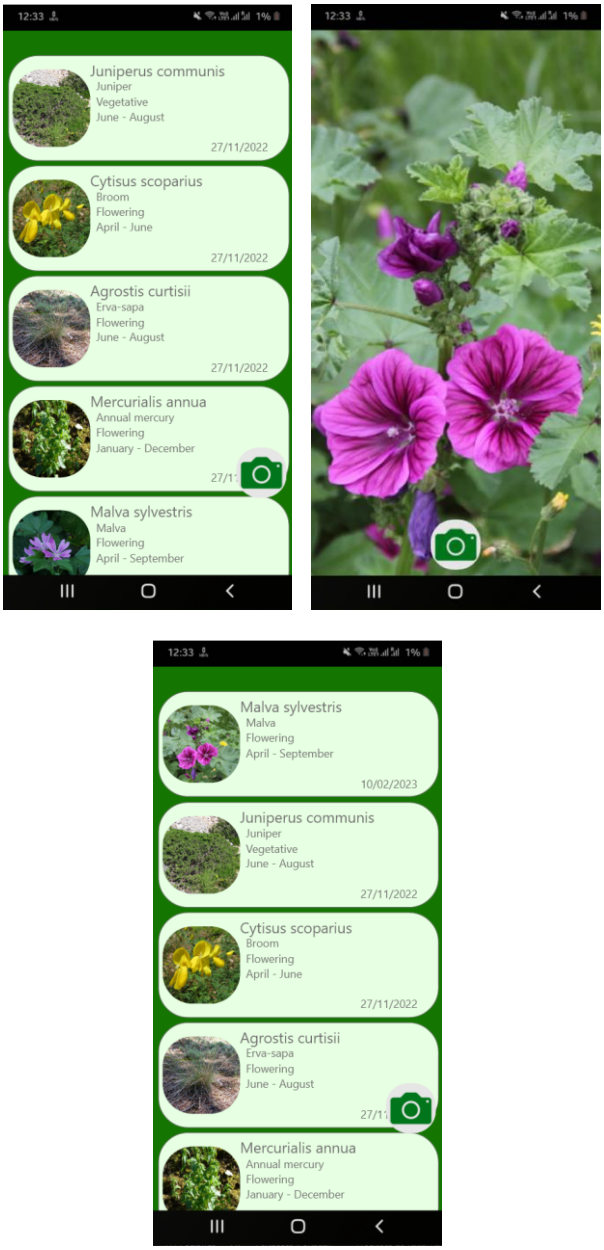

Figure 12: The process of creating a new reading in the prototype.

An interface is also available that allows users to obtain more detailed information about a particular wild plant reading. As it can be seen in Figure 16, it displays information such as the common name, the scientific name, the development stage, the period of the development stage, the habitat specifications, and a full description of the species.

Figure 13: The interface details of a reading in the prototype.

Usability tests were carried out with real users to obtain information about interaction with the application's interface and the user experience. The concept and purpose of the mobile application were explained, and users were asked to create a new reading and consult its details. The difficulty of performing these tasks was assessed. Users were then asked two questions: Question 1 "Would you use this application?"; Question 2 "Would you add any functionality or information?". Table 4 summarizes the results of the usability tests.

The feedback from the usability tests allowed assessing possible changes to be made to the application. In view of what was reported, a "Delete reading" button was incorporated, giving users control over their readings. The category in which the plant is classified by the International Union for Conservation of Nature (IUCN), was also added to the detailed information, which reflects its degree of risk of extinction.

It was also decided to give autonomy to the user to choose the CNN model to be used for detection and classification, unlike the previous approach which decided automatically based on the existence of network coverage. These adjustments based on user experience were aimed at improving the usability of the application.

Table 4: Usability test results

ID of the user	Problems with the interface	Question 1 Would you use this application?	Question 2 Would you add any functionality or information?
1	None observed.	Yes, I like hiking and it would be good to identify plants and what stage they're at.	I would add a button to choose whether I want to use local detection, as I may not want to waste mobile data.
2	None observed.	No, I'm not in the habit of visiting nature parks.	No.
3	None observed.	Yes, because I have a garden at home.	Add a description of the plant's growth stage.
4	None observed.	Yes, it looks like something I would use on a visit to a nature park.	Add information about the state of danger the plant is in.
5	None observed.	No, I'm not interested in plants.	Add a button to delete a reading.
6	None observed.	Yes, if I went to a nature park.	The option to choose the model so as not to use mobile data.
7	None observed.	Yes, I thought it was an interesting idea.	No.
8	None observed.	Yes, if you visit a nature park.	No.
9	None observed.	No, I'm not in the habit of visiting nature parks.	No.
10	None observed.	Yes, I'm in the habit of hiking and I think it's an interesting thing to use.	Add a description of the growth stage.

2.5 Development

The Iterative Development methodology was used to develop the mobile application. Thus, the process was divided into several tasks. Initially, all the interfaces and menus were created and the navigation between them was tested, ensuring that the new code was implemented in this iteration without any problems.

Next, the SQLite database implemented in the application was created. At this stage, the application only returns results for the *Rubus Fruticosus* species, due to the difficulties when creating the dataset described in a subsection above. However, both the application and the database have been designed to store and display information on the development stages of various plant species.

The database stores the information that will be shown to the user. It consists of 6 tables, shown in the entity-relationship (ER) model in Figure 17, which store information such as: the common name of the species, the scientific name of the species, the development stage, the time period of the development stage, the IUCN code, the habitats in which it can be found, and descriptions of the species and stage. After planning and implementing the database, its operation was tested.

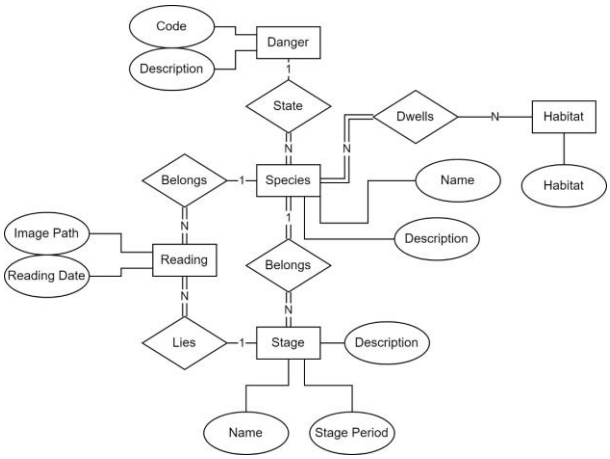

Figure 14: The ER model of the database.

After implementing the database, the YOLOv4-tiny model was implemented. To do this, it was necessary to convert the model trained in darknet to tensorflow lite [42]. This approach allowed the model to be implemented locally in the application. After this implementation, the model was rigorously tested with all the development stages to ensure that any errors were detected and corrected.

Next, an API was implemented with the Flask framework to allow the YOLOv4 model to be executed remotely. To do this, the application checks if there is access to the API using a mobile Internet connection (3G or higher). If successful, it sends the photo for detection and classification. The results are returned in JSON format, and include the species and development stage, to form the reading that is then displayed in the application. The API was subjected to functional tests to detect and correct errors. Figure 18 illustrates how this works.

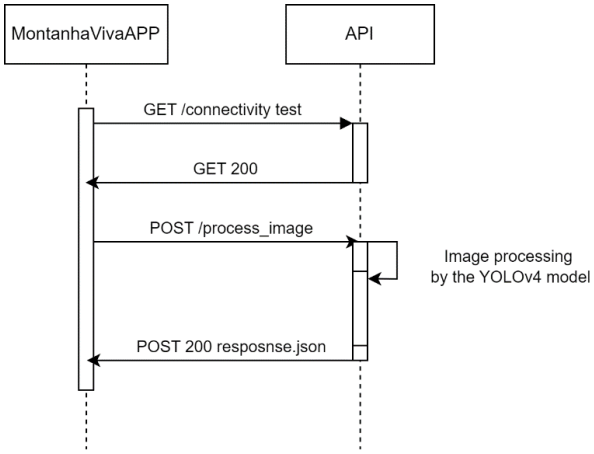

Figure 15: The sequence diagram illustrating the interaction between the application and the API.

After these iterations, the functional prototype of the MontanhaVivaApp application was developed and is available at [43]. Figure 19 shows the modules of the mobile application, including the Java files and menus, respecting the Java language nomenclature [44].

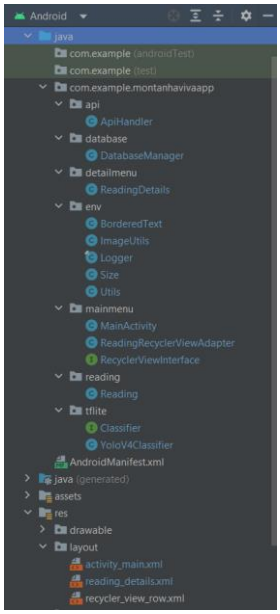

Figure 16: The modules of the MontanhaVivaApp mobile application.

2.6 Structure of the application and evaluation

This subsection describes the features of the MontanhaVivaApp application and the main operations that can be performed on it.

The main menu, shown in Figure 20, serves as the entry point to the application. It provides a view of the previously recorded wild flowers and plants readings. These readings are listed in cells in the main menu, each containing summary information about the scientific name, the common name, the development stage, the period of the stage and the date of the reading.

Figure 17: The main menu interface.

Users can consult more detailed information about a particular reading. All they need to do is to select one of the readings listed in the main menu and the interface shown in Figure 21 will appear. This screen provides a range of information including the common name of the plant, the scientific name, the development stage, the period of that stage, the category assigned by the IUCN, the common habitat, and a detailed description of the species. In the top right-hand corner, there is a button that allows the user to delete this reading. If this button is clicked, once the reading has been removed, the message "Reading successfully deleted" is displayed, as shown in Figure 21. This user-initiated deletion process simplifies data management and improves the application's functionality.

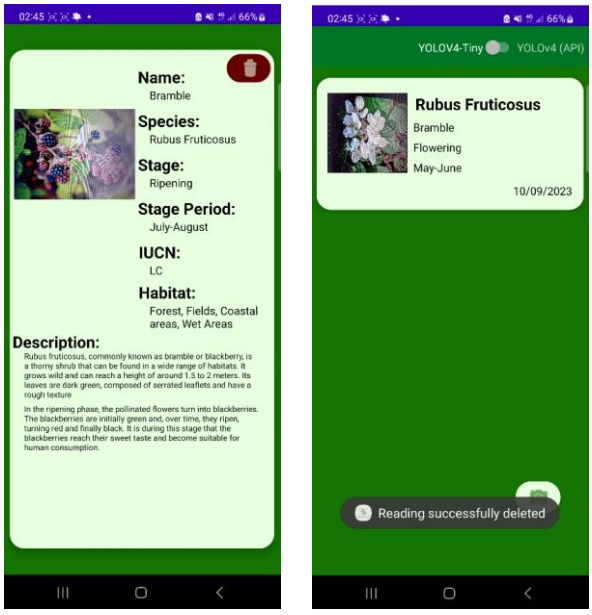

Figure 18: The interfaces showing the details and the message with the deletion notification.

In the top corner of the main menu (Figure 20), the user can choose the CNN model they want to use to detect and classify plants: YOLOv4 or YOLOv4-tiny. In the bottom right-hand corner, there is a button that activates the mobile device's camera to take a photo. This image is then transmitted to the model selected for analysis. Figure 22 shows the process of capturing an image (i.e., photo) to be analyzed.

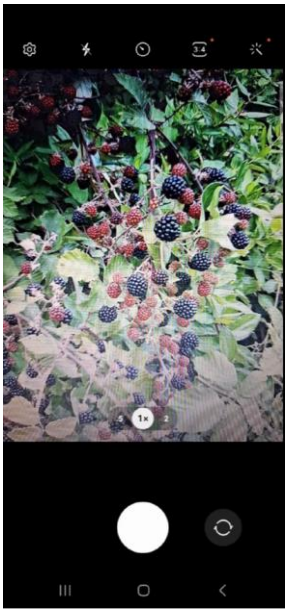

Figure 19: The process of capturing an image.

If the user has selected the YOLOv4 model and there is no Internet connectivity to the API, he will receive the message "No connection to the API, use the local model", shown in Figure 23. This information allows the user to decide whether to use YOLOv4-tiny for local processing in the main menu (Figure 20).

In cases where the chosen model cannot detect the plant and its development stage, the user will receive the message "Bad reading, please try again", shown in Figure 23. These error messages are enlightening in that they suggest possible causes to help the user.

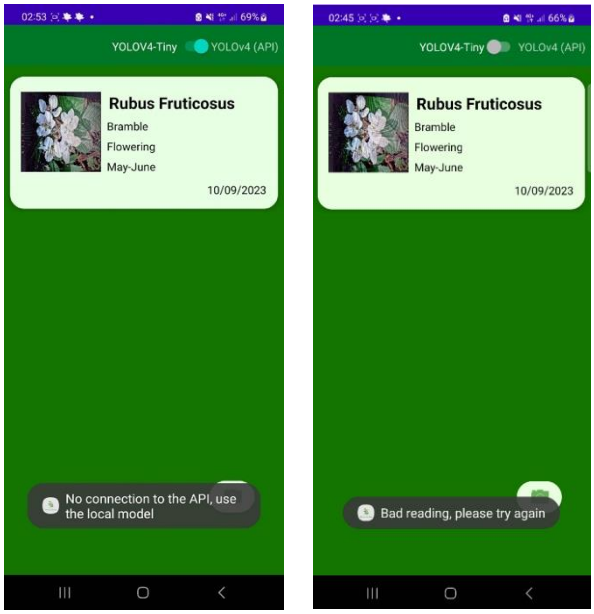

Figure 20: The error messages notification.

3 Conclusion

Wild flowers and plants are a vital part of biological diversity and an essential resource for the planet. However, we are seeing an increase in the number of wild flowers and plants at risk of extinction and in decline due to climate change and the impact of human action. Therefore, there is an urgent need to contribute technological solutions for their conservation and preservation.

The work presented in this article is one of the stages of an ongoing research project, which aims to develop a mobile application and a system based on computer vision techniques to detect and monitor the development stages of wild flowers and plants.

In summary, the main contributions resulting from this article are: 1) the creation of a dataset with the stages of development of a wild plant; 2) a comparative performance analysis of the YOLOv4 and YOLOv4-tiny convolutional neural network models for detecting the development stages of this wild plant; 3) a description of the process of developing a mobile application, using YOLOv4 and YOLOv4-tiny, as a proof of concept.

This mobile application can be used by visitors to a nature park to provide information and raise awareness about the development stages of the wild flowers and plants they encounter along the roads and trails. The application is currently in the testing phase of its prototype version.

Several points remain open for future work, including: 1) creating a dataset with support for a wide range of wild flowers and plants species, with a large number of images for each plant species and development stage; 2) testing and evaluating other convolutional neural network models; 3) continuing the process of validating the application with a wide range of real users; 4) using the feedback from these users to add features to the application that enrich and facilitate its use.

Acknowledgments

J.M.L.P.C. and V.N.G.J.S. acknowledge that this work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020. P.D.G. thanks the support provided by the Center for Mechanical and Aerospace Science and Technologies (C-MAST) under project UIDB/00151/2020.

This is within the activities of project Montanha Viva – An intelligent prediction system for decision support in sustainability, project PD21-00009, promoted by PROMOVE program funded by Fundação La Caixa and supported by Fundação para a Ciência e a Tecnologia and BPI.

Declarations

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

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

References

[1] Agence France-Presse, "Chain-reaction extinctions will cascade through nature: Study | Daily Sabah." https://www.dailysabah.com/life/environment/chain-reaction-extinctions-will-cascade-through-nature-study (accessed Jan. 29, 2023).

[2] L. E. Grivetti and B. M. Ogle, "Value of traditional foods in meeting macro- and micronutrient needs: the wild plant connection," *Nutr Res Rev*, vol. 13, no. 1, pp. 31–46, Jun. 2000, doi: 10.1079/095442200108728990.

[3] E. Christaki and P. Florou-Paneri, "Aloe vera: A plant for many uses," *J Food Agric Environ*, vol. 8, pp. 245–249, 2010.

[4] Trevor Dines, "Plantlife - A Voice for Wildflowers - Ark Wildlife UK." https://www.arkwildlife.co.uk/blog/plantlife-a-voice-for-wildflowers/ (accessed Jan. 29, 2023).

[5] X. Chi *et al.*, "Threatened medicinal plants in China: Distributions and conservation priorities," *Biol Conserv*, vol. 210, Part A, pp. 89–95, Jun. 2017, doi: 10.1016/J.BIOCON.2017.04.015.

[6] Woodstream, "Learn The Six Plant Growth Stages." https://www.saferbrand.com/articles/plant-growth-stages (accessed Sep. 04, 2023).

[7] João Videira, Pedro D. Gaspar, Vasco N. G. J. Soares, and João M. L. P. Caldeira, "Detecting and Monitoring the Development Stages of Wild Flowers and Plants using Computer Vision: Approaches, Challenges and Opportunities (in press)," *International Journal of Advances in Intelligent Informatics (IJAIN)*, 2023.

[8] PEAT GmbH, "Plantix - seu médico agrícola – Apps no Google Play." https://play.google.com/store/apps/details?id=com.peat.GartenBank&hl=pt_PT&gl=US (accessed Oct. 06, 2022).

[9] AIBY Inc., "Plantum - Identificar plantas – Apps no Google Play." https://play.google.com/store/apps/details?id=plant.identification.flower.tree.leaf.identifier.identify.cat.dog.breed.nature&hl=pt_PT&gl=US (accessed Oct. 06, 2022).

[10] N. Buch, S. A. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920–939, Sep. 2011, doi: 10.1109/TITS.2011.2119372.

[11] S. Xu, J. Wang, W. Shou, T. Ngo, A. M. Sadick, and X. Wang, "Computer Vision Techniques in Construction: A Critical Review," *Archives of Computational Methods in Engineering 2020 28:5*, vol. 28, no. 5, pp. 3383–3397, Oct. 2020, doi: 10.1007/S11831-020-09504-3.

[12] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," in *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37.*, 20015, pp. 13–27. doi: 10.1109/CVPR.2011.5995330.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[14] The MathWorks Inc., "What Is Object Detection? - MATLAB & Simulink." https://www.mathworks.com/discovery/object-detection.html?s_tid=srchtitle_object%20detection_1 (accessed Dec. 26, 2022).

[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv, 2020. doi: 10.48550/ARXIV.2004.10934.

[16] G. Li, X. Huang, J. Ai, Z. Yi, and W. Xie, "Lemon-YOLO: An efficient object detection method for lemons in the natural environment," *IET Image*

Process, vol. 15, no. 9, pp. 1998–2009, Mar. 2021, doi: 10.1049/ipr2.12171.

[17] A. Shill and M. A. Rahman, "Plant disease detection based on YOLOv3 and YOLOv4," *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0, ACMI 2021*, pp. 1–6, Jul. 2021, doi: 10.1109/ACMI53878.2021.9528179.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. doi: 10.1109/cvpr.2016.91.

[19] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018, Accessed: Aug. 21, 2023. [Online]. Available: https://arxiv.org/abs/1804.02767v1

[20] The MathWorks Inc., "Anchor Boxes for Object Detection - MATLAB & Simulink." https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html (accessed Dec. 26, 2022).

[21] Q. Chen and Q. Xiong, "Garbage Classification Detection Based on Improved YOLOV4," *Journal of Computer and Communications*, vol. 8, pp. 285–294, 2020, doi: 10.4236/jcc.2020.812023.

[22] Z. Jiang, L. Zhao, S. Li, Y. Jia, and Z. Liquan, "Real-time object detection method based on improved YOLOv4-tiny," *Journal of Network Intelligence*, vol. 7, no. 1, Nov. 2022, Accessed: Aug. 23, 2023. [Online]. Available: https://arxiv.org/abs/2011.04244v2

[23] L. Song *et al.*, "Object detection based on Yolov4-Tiny and Improved Bidirectional feature pyramid network," *Journal of Physics: Conference Series 2021 International Conference on Electronic Communication, Computer Science and Technology 07/01/2022-09/01/2022 Nanchang*, vol. 2209, no. 1, Feb. 2022, doi: 10.1088/1742-6596/2209/1/012023.

[24] W. Zhang *et al.*, "Airborne infrared aircraft target detection algorithm based on YOLOv4-tiny," *Journal of Physics: Conference Series 2021 International Conference on Advances in Optics and Computational Sciences (ICAOCS) 2021 21-23 January 2021, Ottawa, Canada*, vol. 1865, no. 4, Apr. 2021, doi: 10.1088/1742-6596/1865/4/042007.

[25] Ken-ichi Ueda, Nate Agrin, and Jessica Kline, "Uma comunidade para naturalistas · iNaturalist." https://www.inaturalist.org/ (accessed Aug. 21, 2023).

[26] Ken-ichi Ueda, Nate Agrin, and Jessica Kline, "Uma comunidade para naturalistas · BioDiversity4All." https://www.biodiversity4all.org/ (accessed Aug. 23, 2023).

[27] João Videira, Pedro D. Gaspar, Vasco N. G. J. Soares, and João M. L. P. Caldeira, "MontanhaVivaApp Dataset | Kaggle." https://www.kaggle.com/datasets/krosskrosis/montanhavivaapp-dataset (accessed Sep. 05, 2023).

[28] Yonghye Kwon, "GitHub - developer0hye/Yolo_Label: GUI for marking bounded boxes of objects in images for training neural network YOLO." https://github.com/developer0hye/Yolo_Label (accessed Aug. 21, 2023).

[29] Alexey Bochkovskiy, "GitHub - AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)." https://github.com/AlexeyAB/darknet (accessed Aug. 21, 2023).

[30] J. A. Cook and J. Ranstam, "Overfitting," *British Journal of Surgery*, vol. 103, no. 13, p. 1814, Dec. 2016, doi: 10.1002/bjs.10244.

[31] M. Decuyper, M. Stockhoff, S. Vandenberghe, al -, and X. Ying, "An Overview of Overfitting and its Solutions," *J Phys Conf Ser*, vol. 1168, no. 2, p. 022022, Feb. 2019, doi: 10.1088/1742-6596/1168/2/022022.

[32] L. Prechelt, "Early Stopping - But When?," pp. 55–69, 1998, doi: 10.1007/3-540-49430-8_3.

[33] The Interaction Design Foundation, "What is User Centered Design?" https://www.interaction-design.org/literature/topics/user-centered-design (accessed Sep. 08, 2023).

[34] Eastern Peak, "Iterative Development." https://easternpeak.com/definition/iterative-development/ (accessed Sep. 08, 2023).

[35] Matt Rae, "What is Adobe XD and What is it Used for?" https://www.adobe.com/products/xd/learn/get-started/what-is-adobe-xd-used-for.html (accessed Sep. 08, 2023).

[36] Pavel Gorbachenko, "Functional vs Non-Functional Requirements | Enkonix." https://enkonix.com/blog/functional-requirements-vs-non-functional/ (accessed Sep. 08, 2023).

[37] IBM, "Use-case diagrams - IBM Documentation." https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case (accessed Sep. 08, 2023).

[38] Google and JetBrains, "Android Studio & App Tools - Android Developers." https://developer.android.com/studio (accessed Sep. 08, 2023).

[39] D. Richard Hipp, "SQLite." https://www.sqlite.org/index.html (accessed Sep. 08, 2023).

[40] Armin Ronacher, "Welcome to Flask." https://flask.palletsprojects.com/en/2.3.x/ (accessed Sep. 08, 2023).

[41] João Videira, Pedro D. Gaspar, Vasco N. G. J. Soares, and João M. L. P. Caldeira, "MontanhaVivaApp – Google Drive." https://drive.google.com/drive/u/2/folders/1FX6pwvDgV2lN9u3EwtH66DhPunIPJ3AT (accessed Aug. 24, 2023).

[42] Việt Hùng, "GitHub - hunglc007/tensorflow-yolov4-tflite: YOLOv4, YOLOv4-tiny, YOLOv3, YOLOv3-tiny Implemented in Tensorflow 2.0, Android. Convert YOLO v4 .weights tensorflow, tensorrt and tflite." https://github.com/hunglc007/tensorflow-yolov4-tflite (accessed Aug. 21, 2023).

[43] João Videira, Pedro D. Gaspar, Vasco N. G. J. Soares, and João M. L. P. Caldeira, "videira202011/MontanhaVivaApp." https://github.com/videira202011/MontanhaVivaApp (accessed Sep. 04, 2023).

[44] Oracle, "Creating and Using Packages (The Java™ Tutorials > Learning the Java Language > Packages)." https://docs.oracle.com/javase/tutorial/java/package/packages.html (accessed Sep. 05, 2023).