MATHEMATICAL MODELING AND COMPUTING, Vol. 11, No. 4, pp. 1198–1210 (2024)

Advanced YOLO models for real-time detection of tomato leaf diseases

Bellout A.[1], Zarboubi M.[2], Dliou A.[1,3], Latif R.[1], Saddik A.[1,3]

[1] *Laboratory of Systems Engineering and Information Technology LISTI, National School of Applied Sciences, Ibn Zohr University Agadir, Morocco*
[2] *LISAD, National School of Applied Sciences, Ibn Zohr University, Agadir, Morocco*
[3] *Faculty of Applied Sciences, Ibn Zohr University, Ait Melloul, Morocco*

(Received 1 May 2024; Revised 3 December 2024; Accepted 4 December 2024)

The increasing focus on smart agriculture in the last decade can be attributed to various factors, including the adverse effects of climate change, frequent extreme weather events, increasing population, the necessity for food security, and the scarcity of natural resources. The government of Morocco adopts preventative measures to combat plant illnesses, specifically focusing on tomatoes. Tomatoes are widely acknowledged as one of the most important vegetable crops, but they are highly vulnerable to several diseases that significantly decrease their productivity. Deep learning algorithms are increasingly being used to identify tomato leaf diseases. In this study, we thoroughly examine different deep learning methodologies, with a specific emphasis on Convolutional Neural Network (CNN) models. Our study aims at identifying the optimal approach for detecting diseases that impact tomato leaves by combining two publicly accessible datasets, PlantDoc and PlantVillage. We focused on finding a strategy that is effective and efficient in accurately identifying these diseases. This study investigates the feasibility of employing state-of-the-art deep learning methods that are based on YOLO models. We have chosen five models, specifically YOLOv5, YOLOX, YOLOv7, YOLOv8, and YOLO-NAS, which belong to the category of "One-stage detectors." These models are widely recognized for their rapid inference speed and outstanding accuracy. According to the experimental results, YOLOv5 has the highest level of accuracy, reaching a mean average precision (mAP) of 93.1% after adjusting the hyperparameters. The final model is developed as a smartphone application to improve user-friendliness.

Keywords: *precision agriculture; tomato plant disease; deep learning; computer vision; object detection; YOLO.*

2010 MSC: 68T05, 68U10, 97R40 **DOI:** 10.23939/mmc2024.04.1198

1. Introduction

The agriculture sector faces a substantial challenge in satisfying the growing worldwide demand for food. Plant diseases are a major concern that greatly affects the quality and quantity of plant production. While traditional diagnostic approaches for these infections are effective, they may experience delays and require expert assistance, hence increasing the risk of crop loss. At the moment, the majority of Moroccan farmers rely on manual inspection techniques to identify tomato leaf diseases, which leads to considerable time consumption and a high probability of errors. Indeed, at nowadays, there are two main approaches used for disease diagnosis: expert assessments and pathogen analysis. Expert assessments are conducted by plant protection specialists who utilize their significant field and investigative knowledge to examine plant lesions. This strategy relies primarily on the proficiency of the professionals and may be affected by subjective variability and limited accuracy [1]. Pathogen analysis involves the cultivation of pathogens and their observation using a microscope. While this technique provides a high level of accuracy in diagnosing, it requires a significant amount of time and entails intricate procedures that are not practicable for on-site detection [2]. Lately, the progress in machine vision and artificial intelligence has greatly accelerated the incorporation of intelligent engineering across multiple

© 2024 Lviv Polytechnic National University

sectors. This improvement has notably enhanced machine vision applications in industries such as agriculture and manufacturing within complex environments [3, 4]. Addressing the challenges of plant disease detection, techniques that utilize visible light and near-infrared spectroscopic digital imaging have gained traction. These methods, leveraging near-infrared spectroscopic and hyperspectral imaging, offer continuous spectral data and insights into the spatial patterns of plant diseases, making them increasingly favored by researchers [5, 6]. However, the devices needed to capture spectral images are often costly and cumbersome, limiting their widespread adoption. In contrast, acquiring visible light images is simpler and can be accomplished with common electronic devices like digital cameras and smartphones, simplifying the research into visible light image recognition [7].

Recent advancements have also seen visible light image recognition being effectively applied to plant disease detection, facilitated by the need for real-time monitoring and data sharing on crop growth [8–11]. This application typically involves several stages: image segmentation, extraction of disease characteristics from the images, and subsequent classification of the diseases. It is crucial to employ cutting-edge technology like Artificial Intelligence (AI) and Computer Vision (CV) to improve the plant disease identification process. Deep learning has the potential to expedite disease diagnosis and intervention by enabling faster and more efficient processes. High-quality agricultural production relies heavily on precision agriculture. The progress in machine learning and deep neural networks has significantly improved the precision of item identification and recognition systems.

Convolutional neural networks (CNN) have been extensively used in the field of plant disease diagnosis in image processing. These models have demonstrated remarkable precision. Fujita et al. did a study where they proposed a classifier for diagnosing cucumber diseases using Convolutional Neural Networks (CNN) [12]. The classifier had an accuracy rating of 82.3%. Brahimi et al. [13] proposed the utilization of a CNN model as a machine-learning approach to categorize tomato illnesses. Their model achieved an impressive accuracy rate of 99.18%. Rangarajan et al. [14] utilized the AlexNet and VGG16 models to classify six different diseases and a healthy category of tomatoes. Their classification accuracies were 97.49% and 97.23%, respectively, utilizing a dataset consisting of 13 262 photos. Qimei Wang et al. [15] devised methods utilizing deep CNN and object identification models to quickly and accurately identify eleven distinct tomato illnesses. According to the experiment's results, ResNet-101 exhibits the highest detection rate, but it necessitates the lengthiest training and detection durations. MobileNet has the shortest detection time, although it exhibits worse accuracy in comparison to ResNet-101. In their study, Khalid et al. (2023) utilized a deep convolutional neural network (CNN) to accurately identify illnesses in money plant leaves in real time. They employed their trained YOLOv5 model to detect locations on both public and private datasets, reaching a 93% accuracy on a test set.

The objective of this work is to create and implement a deep learning system that utilizes images of both healthy and diseased plants to autonomously detect and classify several types of diseases affecting tomato leaves. The aim of this study is to compare the five most recent and advanced versions of the You Only Look Once models (YOLO): YOLOv5, YOLOX, YOLOv7, YOLOv8, and YOLO-NAS. The models are trained using a combined dataset derived from two publicly available picture datasets, namely PlantDoc and PlantVillage. The completed model will possess the capability to predict the condition of the leaf in real time and provide a prompt and reliable tool for disease prevention, even for individuals without expertise in the field.

This paper will detail the technique utilized to design our desired system, followed by a presentation of the outcomes and a discussion of their significance.

2. Methodology and results

2.1. Model description

There exist multiple object detection Deep Learning models that differ in their hardware prerequisites, speed, accuracy and capacity to handle various classes. Specifically, we were searching for object detection strategies that utilize convolutional neural network detection models in our particular scenario. These algorithms can be classified into two main categories: the first one uses a two-phase approach

for object detection (such as R-CNN, Fast R-CNN, and Faster R-CNN), whereas the other category employs a single-stage method (such as SSD and YOLO). In this study, we have opted to utilize the single step model, more especially the YOLO family. This model leverages the complete set of image attributes to generate predictions for every bounding box. In addition, it simultaneously predicts the bounding boxes for every class present in an image. This suggests that the network employs global approach, considering the whole image and all its constituent elements. The YOLO design enables seamless training and immediate execution, achieving a consistently high level of accuracy. We primarily focused our work on the latest versions of YOLO Models, considering their numerous variations. Therefore, we performed a comparison examination of five models: Yolov5, YOLOX, Yolov7, YOLOv8, and YOLO-NAS. A succinct overview of the several models is presented.

YOLOv5: Launched in 2020, shortly following the release of YOLOv4, YOLOv5 was developed by Glen Jocher, Ultralytics' CEO. It integrates multiple improvements from YOLOv4, yet it is built using PyTorch instead of Darknet. The YOLOv5 model incorporates the AutoAnchor algorithm, as mentioned in the comprehensive study by Terven et al. (2023) [16]. This pre-training tool analyzes and adjusts anchor boxes that are not suitable for the dataset and training parameters, such as the image dimensions. The starting parameters for a Genetic Evolution (GE) algorithm are derived by employing a k-means model on the labels of the dataset. The GE technique progressively improves the anchors during 1000 generations, utilizing the CIoU loss and Best Possible Recall as the fitness function. The overview of YOLOv5 architecture is illustrated in Figure 1. The backbone is comprised of a modified CSPDarknet53 architecture, which starts with a Stem. The Stem is a convolutional layer with a large window size and stride, specifically designed to reduce memory use and computational costs. Afterwards, the backbone includes convolutional layers that extract important information from the input image. The SPPF layer, in conjunction with the subsequent convolution layers, manages the features at various scales, while the upsample layers improve the resolution of the feature maps. YOLOv5 delivers exceptional accuracy by utilizing a simplified architecture and an extensive augmentation pipeline that incorporates advanced techniques like Mosaic, cutmix, and several others. The product offers a range of models, starting from small and efficient nano versions to more powerful X variants. These models cover a wide spectrum of tradeoffs between throughput and accuracy. YOLOv5 has been widely popular for real-time applications due to its exceptional performance, user-friendly Keras-style APIs, and regular upgrades provided by its active open source community and developers.

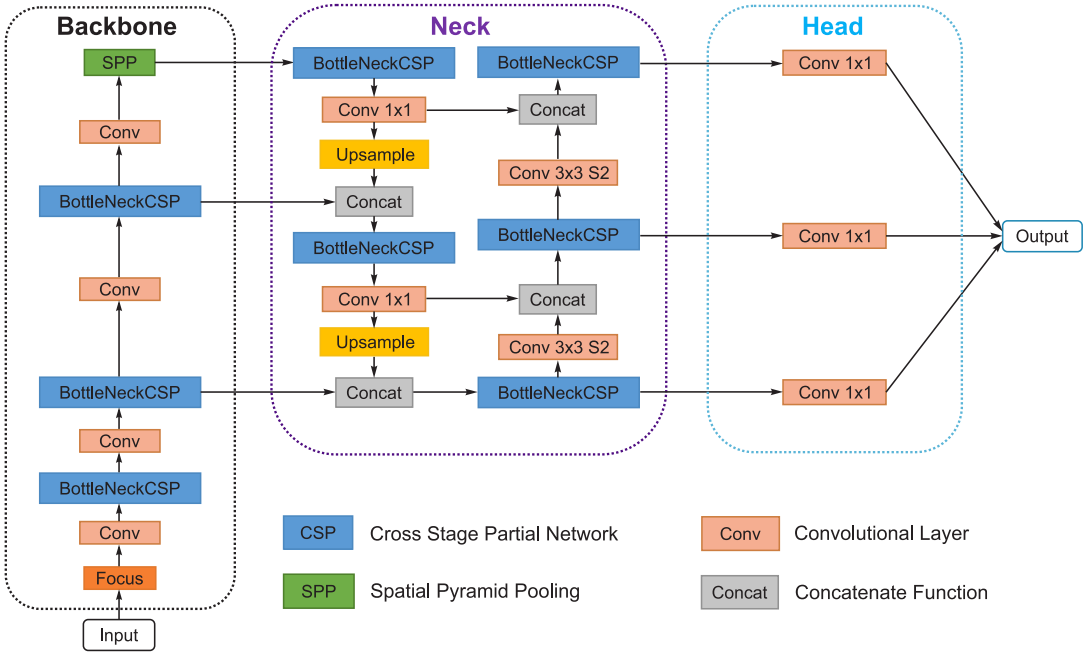

Fig. 1. YOLOv5 Architecture [17]. The structure is comprised of three components: (1) Backbone, (2) Neck, and (3) Head.

YOLOX: created by Megvii Technology and documented in ArXiv in July 2021 [18], has introduced substantial modifications to the YOLOv3 model, resulting in cutting-edge outcomes. The five significant modifications encompass an anchor-free framework that streamlines the process of training and decoding, reverting back to a technique employed prior to YOLOv2. The modification resulted in a 0.9-point rise in the Average Precision (AP). The model implemented a strategy of incorporating several positive examples to address the imbalances resulting from the absence of reference points, resulting in a significant improvement of 2.1 points in average precision (AP). The third modification involved the separation of classification and localization tasks, resulting in a decoupled head. This alteration led to an improvement of 1.1 points in average precision (AP) and accelerated the convergence of the model. In addition, YOLOX resolved uncertainties in assigning ground truth labels by implementing a simplified form of Optimal Transport (OT) known as simOTA, resulting in a 2.3 point increase in average precision (AP). Ultimately, the model employed MixUP and Mosaic augmentations, resulting in a 2.4-point increase in AP and rendering ImageNet pretraining unnecessary. The outcome was an ideal equilibrium between inference speed and precision, achieving 50.1% average precision (AP) at 68.9% frames per second (FPS) on the Tesla V100 [19].

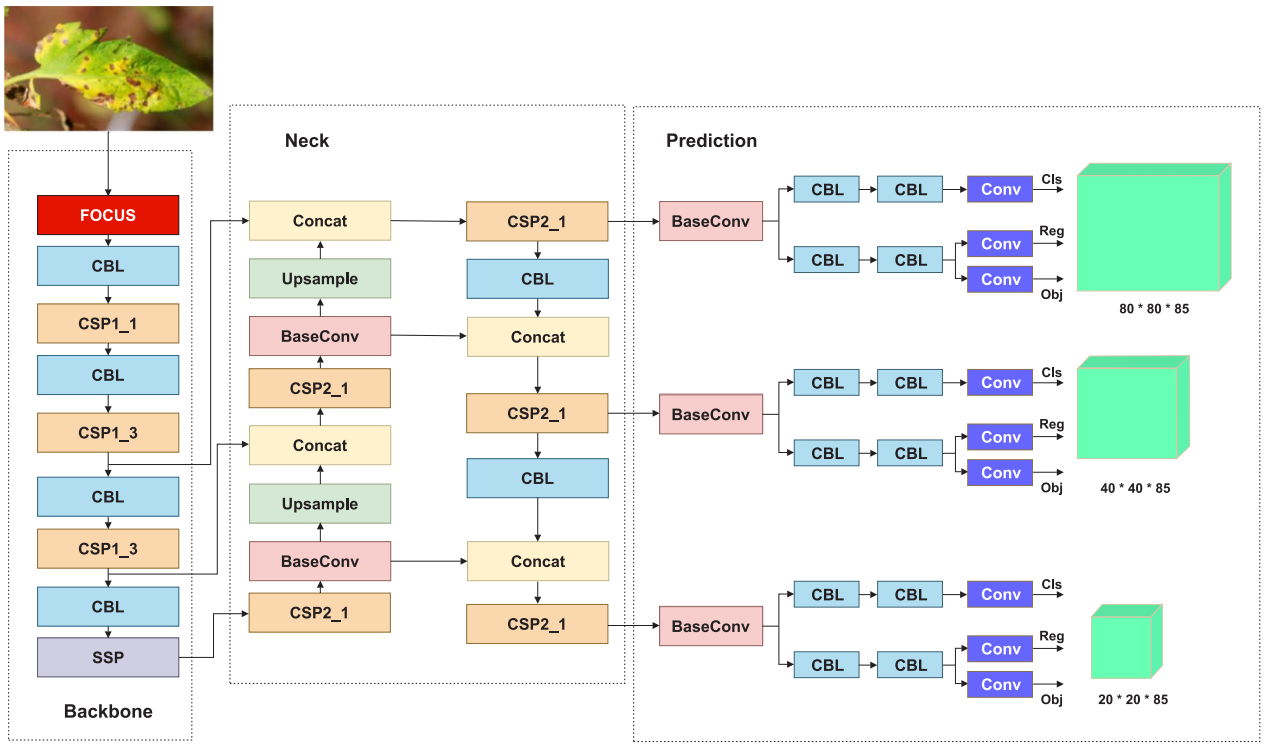

Fig. 2. YOLOX Architecture [19].

YOLOv7: is a model created by the creators of YOLOv4 and YOLOR. It was released in ArXiv in July 2022 [20]. YOLOv7 incorporates many architectural modifications and a set of bag-of-freebies techniques that improve accuracy without compromising inference speed. The model included an Extended Efficient Layer Aggregation Network (E-ELAN) which facilitated efficient learning and convergence in models with an unlimited number of stacked computing units. In addition, it proposed a new approach to scaling models for concatenation-based models that uniformly scales both the width and depth of the blocks, while maintaining the model's desired structure.

YOLOv7's bag-of-freebies incorporated a technique called planned re-parameterized convolution (RepConvN). This technique was inspired by re-parameterized convolutions, but it did not contain the identity connection found in ResNet or the concatenation seen in DenseNet, as these connections can disturb the residual and dense blocks, respectively. Additionally, it implemented a method of assigning rough labels to the auxiliary head and detailed labels to the lead head, which enhanced the efficiency of training. At the inference step, batch normalization was incorporated into the bias and weight of the

convolutional layer. Additional enhancements encompassed the utilization of tacit knowledge derived from YOLOR and the implementation of an exponential moving average as the ultimate inference model.

YOLOv7 surpassed all existing object detectors in terms of both speed and accuracy, achieving a range of 5 FPS to 160 FPS at the time. It was trained exclusively on the MS COCO dataset without utilizing pre-trained backbones.

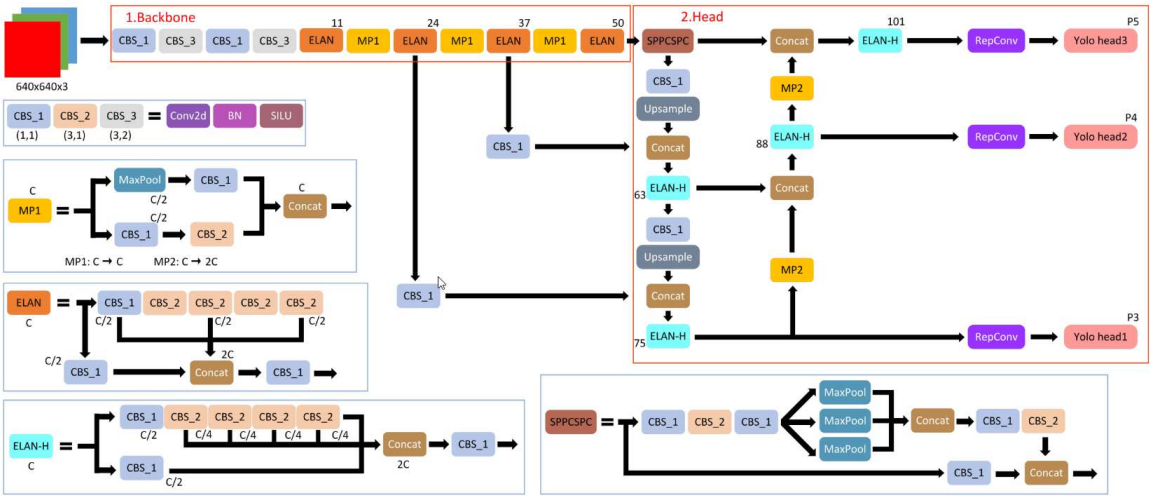

Fig. 3. YOLOv7 Structure.

YOLOv8: Ultralytics, the corporation responsible for the development of YOLOv5, released YOLOv8 in January 2023 [21]. The YOLOv8 model provides five distinct versions with varied scales: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. YOLOv8 has the ability to carry out a range of vision tasks such as object identification, segmentation, pose estimation, tracking, and classification. The YOLOv8 architecture is based on the same foundation as YOLOv5, but it includes changes to the CSPLayer, which is now called the C2f module. This module, comprising of two convolutions, combines high-level characteristics with contextual information to improve the accuracy of detection. The YOLOv8 model employs an anchor-free design, where a separate head is used to independently perform tasks related to objectness, classification, and regression. This strategy enables the division of tasks among different branches and improves the overall precision of the model. The output layer of YOLOv8 employs the sigmoid function as the activation function to calculate the objectness score. This value denotes the likelihood of an object's existence within the bounding box. The Softmax function is used to represent the probabilities of the items belonging to each potential class.

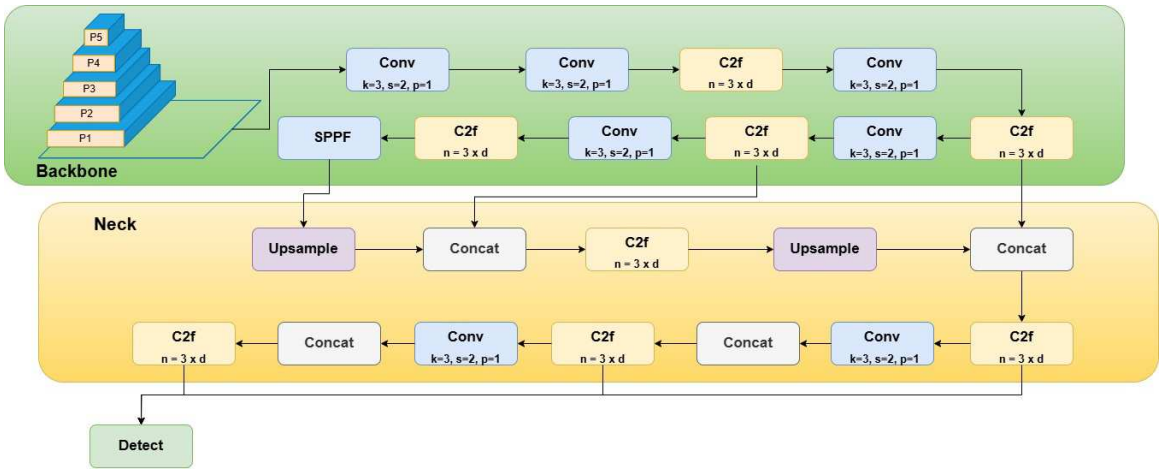

Fig. 4. YOLOv8 Architecture.

YOLO-NAS: Introduced by Deci in May 2023 [22], represents a groundbreaking advancement in neural architecture search (NAS) applied to object detection. Specifically tailored for real-time applications on edge devices, YOLO-NAS excels in detecting small objects and enhancing localization accuracy, crucial for dynamic environments. This model leverages innovative quantization-aware modules, QSP and QCI, which permit 8-bit quantization while minimizing accuracy loss, a notable feature in post-training optimization. AutoNAC, Deci's proprietary NAS technology, drives the automatic design of the model architecture, ensuring an optimal balance between latency and accuracy through a hybrid quantization approach. Furthermore, YOLO-NAS utilizes a robust pre-training regimen that includes automatically labeled data and self-distillation with large datasets, enhancing its performance. The architecture supports multiple configurations "small (S), medium (M), and large (L)" tailored to various operational needs and performance goals, as showcased in its diverse model options YOLO-NASS, YOLO-NASM, and YOLO-NASL. Available for research through its open-source framework, YOLO-NAS is not only a technical achievement but also a versatile tool adaptable to diverse data and environmental conditions, thereby setting a new standard in NAS for object detection.

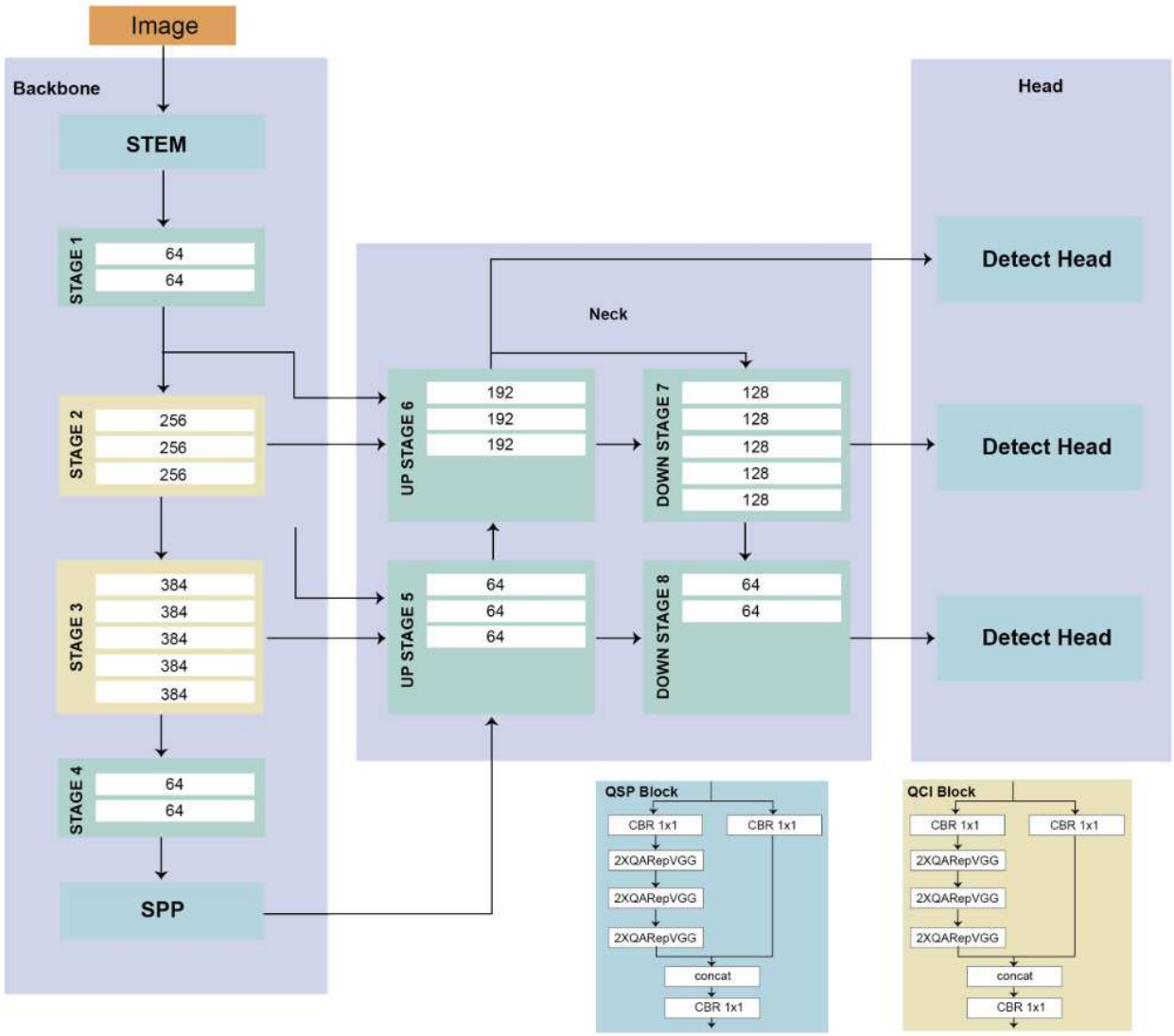

Fig. 5. YOLO-NAS Architecture [16].

2.2. Datasets

Deep learning training processes heavily depend on the availability of large datasets. In the context of our current project, there is a specific need for an extensive array of images depicting both healthy and

diseased tomato leaves to efficiently train our object detection models. Considering the labor-intensive nature of collecting such photos in natural settings, we have opted to incorporate images from two widely recognized public databases, **PlantVillage** and **PlantDoc**.

PlantVillage features an extensive collection of 54 305 images representing individual leaves from 14 different crop species, organized into 38 distinct classes labeled as either species-disease or species-healthy. Each leaf was carefully removed from the plant and placed against a controlled backdrop of gray or black, then photographed outdoors using a high-quality digital camera. However, the images in the PlantVillage database lack pre-existing labels, necessitating a manual annotation process to render them usable for object detection purposes. This crucial step involves detailed labeling of each image to ensure accurate identification and categorization for training the models.

Similarly, the **PlantDoc** dataset serves as a vital resource for plant disease detection, containing 2 598 images that cover 13 different plant species and 17 disease classes. These images were meticulously curated from various online sources and required approximately 300 hours of human labor to annotate thoroughly. This dataset not only enriches the diversity of our training data but also enhances the robustness of our object detection models by providing a wide range of disease manifestations across multiple species.

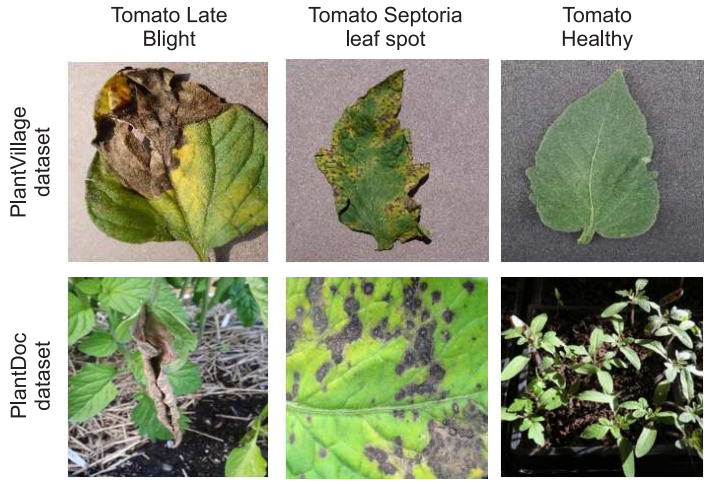

Fig. 6. Plant disease and healthy image examples from PlantVillage and Plandoc.

Fig. 7. Image dataset repartition by class.

The image in Figure 6 depicts some plant disease and healthy examples obtained from two datasets.

In our research, we successfully collected high-quality photos of two different disease categories, specifically Tomato Late Blight and Tomato Septoria leaf spot. In addition, we obtained a distinct set of photos illustrating healthy tomato leaves. This decision was made since we need to annotate three thousand PlantVillage photos. The object detection database will consist of a total of 3325 images, including 3000 images obtained from PlantVillage and 325 images extracted from PlantDoc Figure 7.

The combined count of images for training, testing, and validation is shown in Table 1. Each image may have numerous bounding boxes, which correspond to different health conditions of the leaves. In total, there are 4025 bounding boxes, which serve as annotations. The ultimate dataset was partitioned into three subsets: the training set (70%), the validation set (20%), and the testing set (10%).

Table 1. Image counts for training, validation, and testing.

Training Sample	Validation Sample	Testing Sample	Total Sample
2327	665	333	3325

2.3. Methods

The proposed models were evaluated by the use of methodologies and a comparison of the results. Figure 8 illustrates a comprehensive procedure for identifying and classifying plant illnesses. Furthermore, once the picture dataset containing three classes (Tomato Septoria leaf spot, Tomato leaf late blight, and Healthy Tomato leaf) was collected from both PlantDoc and PlantVillage, the images obtained from the PlantVillage dataset were annotated using the Roboflow annotation web platform. The annotation procedure entails defining bounding boxes around the areas of the leaves that exhibit signs of unhealthiness. It is possible for a single image to have multiple bounding boxes, depending on the extent of the leaves' health status.

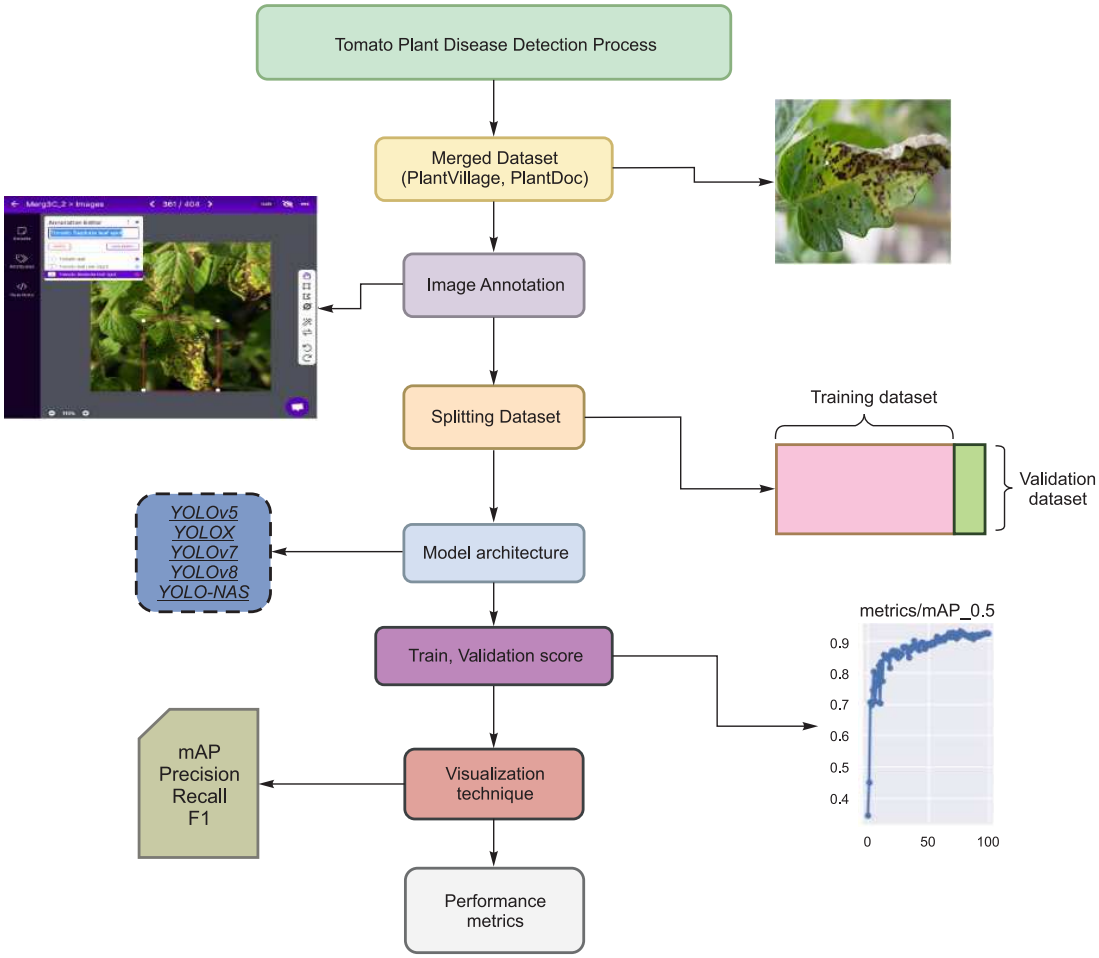

Fig. 8. Tomato leaf disease detection process.

The ultimate dataset was combined and subsequently split into three portions, with 70% allocated for training, 20% for validation, and 10% for testing. Afterwards, the five deep-learning (DL) models were trained on the Google Colab platform using an NVIDIA Tesla P100 GPU with 16 GB of memory. The training process involved either starting from the initial state or using a specific learning approach. The resulting training plots were obtained to assess the importance of the model. The next phase involved classifying the pictures using performance matrices, while the last step involved locating the images using visualization methods. The procedure of identifying unhealthy plant leaf sections involved several steps, as shown in Figure 8.

2.4. Training process

Each of the five models being evaluated in this study consists of many sizes. The lowest model has a minimal parameter number resulting in lower accuracy but faster performance. As the size of the model increases, the number of parameters also increases, resulting in improved accuracy. However,

this also leads to longer inference times. Using YOLOv5 as an illustration, there exist five distinct model sizes: YOLOv5n (the smallest), YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x (the largest). The Figure 9 illustrates the size, accuracy, and inference speed of each model.

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	Params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Fig. 9. Different YOLOv5 models sizes,

In our research, we have selected the YOLOv5s model and similar variants from other model architectures due to their ability to perform rapid inference while maintaining a high level of accuracy. This choice aligns with our objective to develop a model capable of delivering real-time results on devices with constrained computational resources, such as smartphones, IoT devices, and Raspberry Pi systems. The decision to use these specific models is underscored by their efficiency in processing and speed, which are critical for applications requiring immediate feedback in resource-limited settings. The accompanying Table 2 details the weight sizes for each model and outlines the parameters employed during the training phase, providing a clear framework for understanding the configuration and optimization strategies that were implemented to achieve effective performance under the specified hardware limitations.

Table 2. Models training parameters.

Model	Image Size	Epochs	Batch size	Params (M)
YOLOv5s	416	100	16	7.2
YOLOXs	416	1000	16	9.0
YOLOv7 tiny	416	100	16	6.2
YOLOv8s	416	200	16	11.2
YOLO-NAS S	416	200	16	19.0

3. Results and discussions

3.1. Obtained results

After training several models using our finalized dataset, we have documented the results in Table 3. Among the models evaluated, the YOLOv5 stands out with its exceptional performance, achieving a mean Average Precision (mAP0.5) score of 92.7%. It is closely followed by the YOLOv8s, which scored 92.2%, and the YOLOX-s, which registered a score of 89.3%. On the other hand, the YOLO-NAS-S and YOLOv7-Tiny models show the least effective performance, with a score of 88.8% and 87.6% respectively. This array of results highlights the varying capabilities of each model, reflecting how each handles the complexities of the task within the parameters set by our dataset. The scoring indicates that while all models are competent, some are more suited to achieving higher accuracy in our specific application, particularly in scenarios demanding real-time processing on limited-resource devices.

Table 3. Outcomes from training the three models.

Model	Total Loss	mAP 0.5
YOLOv5s	0.02	92.7 %
YOLOX-s	0.59	89.3 %
YOLOv7 Tiny	0.042	87.6 %
YOLOv8s	1.45	92.2 %
YOLO-NAS S	1.37	88.8 %

We have chosen the YOLOv5 model for the hyperparameter tuning phase, with the goal of identifying the most effective settings to enhance the model's accuracy and refine its performance. The

YOLOv5 model utilizes a method known as "Hyperparameter evolution," which involves using a Genetic Algorithm (GA) to fine-tune the hyperparameters. This optimization process aims to improve training results by systematically adjusting hyperparameters to maximize the model's effectiveness.

In particular, YOLOv5 focuses on optimizing a fitness value calculated from the Intersection over Union (IoU) between the predicted bounding boxes and the actual ground truth, as well as from the mean average precision metric. This mean average precision is calculated by averaging the precision across all classes.

However, conducting hyperparameter evolution is both resource-intensive and time-consuming. In our project, the evolution process began with the weights obtained at the 30th epoch. After adjusting the hyperparameters, we observed an increase in the mAP0.5 scores from previous benchmarks, improving by 0.4% to reach 93.1%. Additionally, the mAP0.5:0.95 score increased by 3.6% to 81.55%. These enhancements underscore the value of thorough hyperparameter optimization in achieving superior model performance.

After finalizing the model weights, we decided to deploy our model within an Android application designed to detect various tomato diseases in the field. Consequently, it is essential to convert the trained model from its original PyTorch format, indicated by the file extension .pt, to the Tensorflow Lite format, marked by the extension .tflite. This conversion is crucial as the Tensorflow Lite format is optimized for mobile devices, making it suitable for real-time applications on smartphones. The converted file will then be integrated into the Android application, allowing it to be installed and utilized effectively on mobile devices. This step ensures that the powerful capabilities of our deep learning model are accessible in a portable format, facilitating immediate and practical use in agricultural settings.

3.2. Model deployment

We deployed our application on a Samsung Note 10 Plus smartphone, which is equipped with a Qualcomm Snapdragon 855 processor (running at 2840 MHz) and a Qualcomm Adreno 640 GPU card with a frequency of 672 MHz. The performance of the application can be seen in the screenshots displayed in Figure 10, taken directly from the smartphone. These images illustrate the application's high accuracy and confidence in determining whether a tomato plant is diseased, specifying the type of disease, or confirming the plant's health.

A significant performance insight gained from this deployment is the comparison of inference speeds between the GPU and CPU on the device. The GPU demonstrates a markedly faster inference speed, averaging about 70 milliseconds, compared to the CPU's 350 milliseconds. This results in GPU processing being approximately seven times quicker than its CPU counterpart. Such a disparity underscores the efficiency and suitability of using GPU acceleration for real-time applications like ours, where speed is critical for performance.

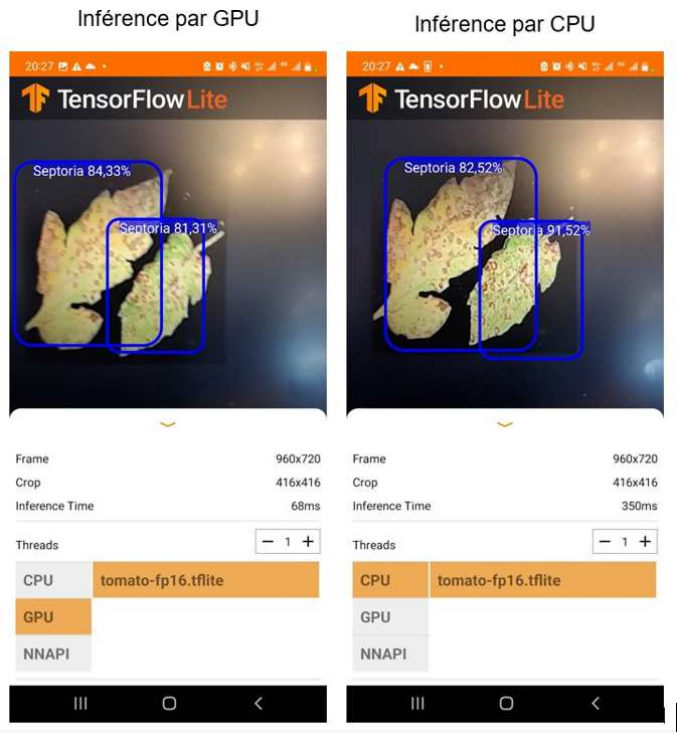

Fig. 10. Tomato Septoria spot disease inference on smartphone (GPU and CPU).

4. Conclusions

The primary goal of this study was to develop a highly effective and precise method for the early detection of tomato diseases using cutting-edge deep-learning techniques. We utilized images of tomato leaves from two databases, PlantDoc and PlantVillage, merging them to create a diverse and comprehensive dataset for training our object detection models. We conducted a detailed comparative analysis using five versions of the YOLO (You Only Look Once) object detection framework: YOLOv5, YOLOX, YOLOv7, YOLOv8, and YOLO-NAS, with YOLOv5 demonstrating the highest accuracy at 92.7%. Despite this achievement, we pursued further improvements in our model's performance. We implemented hyperparameter evolution techniques to optimize the model's settings, which significantly improved the accuracy to 93.1%. Our study highlights the effectiveness of the YOLO model variants in accurately detecting tomato diseases from leaf images and demonstrates the impact of hyperparameter optimization in refining model performance. Furthermore, to increase the usability and reach of our solution, we developed a mobile application, enhancing the practical application of our research. This integration into a mobile platform allows for real-time, on-site disease diagnosis, making our system a valuable tool in the field of agricultural technology. Ultimately, our work improves tomato crop management by enabling early disease detection, which can help maximize yields and reduce losses due to disease.

[1] Lin G., Tang Y., Zou X., Xiong J., Fang Y. Color-, depth-, and shape-based 3D fruit detection. Precision Agriculture. **21** (1), 1–17 (2020).

[2] Buja I., Sabella E., Monteduro A. G., ChiriacΓΙ M. S., De Bellis L., Luvisi A., Maruccio G. Advances in Plant Disease Detection and Monitoring: From Traditional Assays to In-Field Diagnostics. Sensors. **21** (6), 2129 (2021).

[3] Wang C., Tang Y., Zou X., Luo L., Chen X. Recognition and Matching of Clustered Mature Litchi Fruits Using Binocular Charge-Coupled Device (CCD) Color Cameras. Sensors. **17** (11), 2564 (2017).

[4] Li J., Tang Y., Zou X., Lin G., Wang H. Detection of Fruit-Bearing Branches and Localization of Litchi Clusters for Vision-Based Harvesting Robots. IEEE Access. **8**, 117746–117758 (2020).

[5] Gui J., Fei J., Wu Z., Fu X., Diakite A. Grading method of soybean mosaic disease based on hyperspectral imaging technology. Information Processing in Agriculture. **8** (3), 380–385 (2021).

[6] Appeltans S., Pieters J. G., Mouazen A. M. Detection of leek white tip disease under field conditions using hyperspectral proximal sensing and supervised machine learning. Computers and Electronics in Agriculture. **190**, 106453 (2021).

[7] Phadikar S., Sil J., Das A. K. Rice diseases classification using feature selection and rule generation techniques. Computers and Electronics in Agriculture. **90**, 76–85 (2013).

[8] Singh S., Gupta S., Tanta A., Gupta R. Extraction of Multiple Diseases in Apple Leaf Using Machine Learning. International Journal of Image and Graphics. **22** (03), 2140009 (2022).

[9] Almadhor A., Rauf H. T., Lali M. I. U., Damaševičius R., Alouffi B., Alharbi A. AI-Driven Framework for Recognition of Guava Plant Diseases through Machine Learning from DSLR Camera Sensor Based High Resolution Imagery. Sensors. **21** (11), 3830 (2021).

[10] Bellout A., Dliou A., Latif R., Saddik A. Deep Learning technique for predicting tomato leaf disease. 2023 IEEE International Conference on Advances in Data-Driven Analytics and Intelligent Systems (ADACIS). 1–6 (2023).

[11] Zarboubi M., Chabaa S., Dliou A. Advancing Precision Agriculture with Deep Learning and IoT Integration for Effective Tomato Pest Management. 2023 IEEE International Conference on Advances in Data-Driven Analytics And Intelligent Systems (ADACIS). 1–6 (2023).

[12] Fujita E., Kawasaki Y., Uga H., Kagiwada S., Iyatomi H. Basic Investigation on a Robust and Practical Plant Diagnostic System. 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). 989–992 (2016).

[13] Brahimi M., Boukhalfa K., Moussaoui A. Deep Learning for Tomato Diseases: Classification and Symptoms Visualization. Applied Artificial Intelligence. **31** (4), 299–315 (2017).

[14] Rangarajan A. K., Purushothaman R., Ramesh A. Tomato crop disease classification using pre-trained deep learning algorithm. Procedia Computer Science. **133**, 1040–1047 (2018).

[15] Wang Q., Qi F., Sun M., Qu J., Xue J. Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques. Computational Intelligence and Neuroscience. **2019**, 9142753 (2019).

[16] Terven J., Cordova-Esparza D. A Comprehensive Review of YOLO: From YOLOv1 and Beyond. Preprint ArXiv:2304.00501 (2023).

[17] Huang Y., Kruyer A., Syed S., Kayasandik C. B., Papadakis M., Labate D. Automated detection of GFAP-labeled astrocytes in micrographs using YOLOv5. Scientific Reports. **12** (1), 22263 (2022).

[18] Ge Z., Liu S., Wang F., Li Z., Sun J. YOLOX: Exceeding YOLO Series in 2021. Preprint ArXiv:2107.08430 (2021).

[19] Peng H., Tan X. Improved YOLOX's Anchor-Free SAR Image Ship Target Detection. IEEE Access. **10**, 7001–70015 (2022).

[20] Wang C. Y., Bochkovskiy A., Liao H. Y. M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. Preprint ArXiv:2207.02696 (2022).

[21] Jocher G., Chaurasia A., Qiu J. YOLO by Ultralytics. `https://github.com/ultralytics/ultralytics` Version 8.0.0; AGPL-3.0 (January 2023).

[22] Research Team. YOLO-NAS by Deci Achieves State-of-the-Art Performance on Object Detection Using Neural Architecture Search. 2023.
`https://deci.ai/blog/yolo-nas-object-detection-foundation-model/` (accessed on 12 May 2023).

Удосконалені моделі YOLO для виявлення захворювань листя томатів у реальному часі

Беллаут А.[1], Зарбубі М.[2], Дліу А.[1,3], Латіф Р.[1], Саддік А.[1,3]

[1] *Лабораторія системної інженерії та інформаційних технологій LISTI, Національна школа прикладних наук, Університет Ібн Зохр Агадір, Марокко*
[2] *LISAD, Національна школа прикладних наук, Університет Ібн Зор, Агадір, Марокко*
[3] *Факультет прикладних наук, Університет Ібн Зохр, Айт Меллул, Марокко*

Збільшення уваги до розумного сільського господарства за останнє десятиліття можна пояснити різними факторами, включаючи несприятливі наслідки зміни клімату, часті екстремальні погодні явища, збільшення населення, необхідність продовольчої безпеки та дефіцит природних ресурсів. Уряд Марокко приймає профілактичні заходи для боротьби з хворобами рослин, особливо зосередившись на томатах. Помідори широко визнані однією з найважливіших овочевих культур, але вони дуже вразливі до ряду хвороб, які значно знижують їх урожайність. Алгоритми глибокого навчання все частіше використовуються для виявлення хвороб листя томатів. Це дослідження передбачає комплексне вивчення різних методологій глибокого навчання з особливим акцентом на моделях згорткової нейронної мережі (CNN). Мета нашого дослідження полягала в тому, щоб визначити оптимальний підхід для виявлення хвороб, які впливають на листя томатів, шляхом поєднання двох загальнодоступних наборів даних PlantDoc та PlantVillage. Наша увага була зосереджена на пошуку стратегії, яка була б ефективною та дієвою для точного визначення цих захворювань. У цій статті досліджується доцільність використання найсучасніших методів глибокого навчання, які базуються на моделях YOLO. Обрано п'ять моделей, а саме: YOLOv5, YOLOX, YOLOv7, YOLOv8 та YOLO-NAS, які відносяться до категорії "одноступеневі сповіщувачі". Ці моделі широко відомі своєю високою швидкістю логічного висновку та винятковою точністю. Відповідно до експериментальних результатів, YOLOv5 має найвищий рівень точності, досягаючи середньої точності (mAP) 93.1% після налаштування гіперпараметрів. Остаточна модель розроблена як додаток для смартфона, щоб покращити зручність використання.

Ключові слова: *точне землеробство; хвороба томатів; глибоке навчання; комп'ютерний зір; виявлення об'єктів; YOLO.*