Environmental Engineering and Management Journal

January 2022, Vol. 21, No. 1, 163-173
http://www.eemj.icpm.tuiasi.ro/; http://www.eemj.eu

"Gheorghe Asachi" Technical University of Iasi, Romania

DISCOVERY OF AGRICULTURAL DISEASES BY DEEP LEARNING AND OBJECT DETECTION

Mevlüt Karakaya[1], Mehmet Fatih Çelebi[1*], Akın Emrecan Gök[2], Sezgin Ersoy[1]

[1]Mechatronics Engineering Department, Marmara University, Istanbul, Turkey
[2]Environmental Engineering Department, Istanbul University-Cerrahpasa, Istanbul, Turkey

Abstract

In this study deep learning and object detection models for image-based plant disease recognition have been carried. Trained models were tested on pictures and in real-time with a video camera for five different diseases in tomato leaves. Object detection algorithm was implemented from the personal computer, and deep learning models were applied via Google Colab. Real-time object detection was achieved in the developed model with YOLOv5 algorithm with the highest accuracy of 93.38% in validation accuracy and 94.48% in training accuracy with the highest value of 92.96% in precision. Furthermore, it has been observed that YOLOv5 algorithm gives faster and more accurate results than the previous versions of YOLO.

Key words: agricultural disease, deep learning, disease detection, object detection

Received: March, 2021; Revised final: September, 2021; Accepted: October, 2021; Published in final edited form: January, 2022

1. Introduction

In determining the future of agriculture, the whole world is in consensus that it depends entirely on the solutions to be produced to the problems in this field. The most fundamental problems affecting the future of agriculture such as climate change, population growth, usage of resources and food security encourage the use of artificial intelligence in agriculture.

Factors that cannot be controlled with the effect of climate change in the agricultural sector increase crop losses. Traditional methods, taking into account the quality and organic structure of the soil, drainage and irrigation applications, seed selection and sowing method, soil and plant feeding practices, plant protection practices, and harvesting techniques cannot provide professional monitoring. In digital monitoring methods, the information collected from channels such as sensors, cameras, and satellites are stored systematically and based on location, cause and effect relationships can be reported by interpreting using various algorithms, and appropriate data can be transmitted as instant information from mobile phones. Digital technologies that come into our lives with Industry 4.0 act as intermediaries that provide data to decision support platforms (Kirkaya, 2019).

Artificial intelligence applications are expected to take place in the most important agricultural research issues today and in the near future due to their potential to facilitate the applications and to develop solutions to the problems (Dokic et al., 2020; Terzi et al., 2019). Many studies using artificial intelligence techniques in many fields in agriculture have been done by researchers such as crop planning, plant classification, estimating the cost of bring forth, plant disease diagnosis, pest, and weed control, determining the cycle and making application decisions of agricultural robots, providing suitable conditions in the greenhouses, making operational decisions, determination of crop rotation, choosing the most suitable fertilizer and equipment, detection of animal diseases, preparation of suitable feed rations, and issues such as determining animal behaviors (Ali et

* Author to whom all correspondence should be addressed: e-mail: fatih.celebi@marmara.edu.tr; Phone: +90 2167774000; Fax: +90 2167774001

al., 2018; Faridi et al., 2018; Kale and Patil, 2019; Kamilaris and Prenafeta-Boldú, 2018; Kurniasih et al., 2018; Matha et al., 2016; Oppenheim and Shani, 2017). Diseases of agricultural products are one of the most important problems affecting production. Determining the presence of the disease is the most important step for yield estimation, intervention, and treatment. Rapid and accurate diagnosis of the presence of the disease is very important to reduce yield losses (Bock et al., 2010). For this purpose, artificial intelligence can be used to process visual data in disease detection.

Deep learning is a set of algorithms and models working on "Artificial neural networks", which are multi-layered network structures, designed inspired by the structural and functional structures of the brain. *Deep* refers to more than one hidden layer. It uses many layers of nonlinear processing units for deep learning, feature extraction, and conversion. The input/output relation of the layers is serial, i.e., every layer uses the output of the previous layer (Deng and Yu, 2014). The human brain can learn, solves problems by observing and thinking. Neural networks are a class of models inspired by the human brain built with layers (Bre et al., 2017).

As a result of today's developing technology, lots of pictures or video data are generated by cameras or computers. Visual data are transferred to computers in a way that computers can understand. With the development of machine learning methods, rapid steps have been taken in the processing of visual data. The stages in the processing of visual data are pre-processing, object detection, object classification, and object tracking. The most important of these stages is to identify the object. Once the object can be detected, it becomes much easier to track, classify, and other operations. Object detection is generally the detection task of objects of a certain class within an image. It is aimed not only to detect the objects in the images but also to predict the class and position of the objects in the images (Uijlings et al., 2013).

Studies on the use of artificial intelligence in agriculture have been increasing in number recently (Ruiz-Real et al., 2020). Sladojevic et al. (2016) studied plant disease recognition with image classification of leaves with an accuracy rate of 96.3%. Wang et al. (2017) studied plant disease degree estimation with the deep VGG16 model with an accuracy of 90.4%. Wu et al. (2020) studied the detection of the apple flowers using YOLOv4 deep learning algorithm for simultaneous detection comparing different algorithms. Wspanialy and Moussa (2020) studied detecting the nonexclusive diseases of tomato greenhouse plants and severity estimation. Their study shows the relation of different leaf properties and the presence of diseases. Malik et al. (2021) studied disease recognition in sugarcane crops using YOLO and Faster R-CNN with an average accuracy of 58.13% using images collected online.

The aim of our study is to approach the leaf diseases of plants, which are a problem in agriculture, and fires in forests, through machine learning. In this way, diseases in the leaves of the plants and fires in the trees can be detected and intervened with the camera systems of vehicles such as tractors and drones. For this determination, the data set containing the diseased and healthy plants and photographs of fire detection is trained with various models. Detection of objects with the camera is made with the YOLOv5 model and the test results that have obtained are compared with the camera image and instant results are obtained. In this way, disease detection and fire detection can be made.

2. Training algorithm

The training algorithm is shown in Fig. 1. In the beginning, the training model is to be chosen. In step 2, an appropriate dataset is needed. If there exists a suitable one on the internet, it can be downloaded. Else, the dataset can be created by photographing existing known and defined diseased leaves. In step 3, the dataset is having to be checked, and if there any incorrect matches, the wrong data must be deleted. In step 4, the training model (a RetinaNet or YOLO model, etc.) is applied. Note that, even different YOLO versions have different models like YOLOv5 has YOLOv5l, YOLOv5s, etc. If one model takes too long to train, other models can be used. In our study, YOLOv5s is used, since YOLOv5l took too long training time. Also note that, long lasting training models can be used to increase accuracy, if there are good GPU hardware in hand. After the training finishes, the accuracy is having to be controlled. In our study, we set the threshold of successful accuracy threshold to 80%. If the accuracy is below the threshold, another approach must be chosen. After the successful training process, the test results must be controlled again for the possibility of wrong defined dataset errors. If everything is fine in step 6, the training is successful. If there are mistakes, need to go back to step 3 and edit the existing dataset.

3. Materials and methods

Today, the applications of the methods used for object detection can be seen in many areas. With the development of computer science and the emergence of new architectures, object detection architectures are accelerating, and their accuracy rates increase. Today, the most popular algorithm used in simultaneous object detection is YOLO (You Only Look Once) algorithm. YOLO uses convolutional neural networks (CNN). When the YOLO algorithm is run, it detects the objects in pictures or videos and where these objects are in the image at the same time.

The most important feature that distinguishes YOLO from other algorithms is its ability to detect real-time objects. Although there are many algorithms capable of real-time object detection, YOLO was chosen for its high mean average precision (mAP) values. Fig. 2 shows the comparison of YOLOv3 and other algorithms in the COCO dataset where YOLO is much better than its competitors in terms of time and accuracy.

Fig. 1. The training algorithm

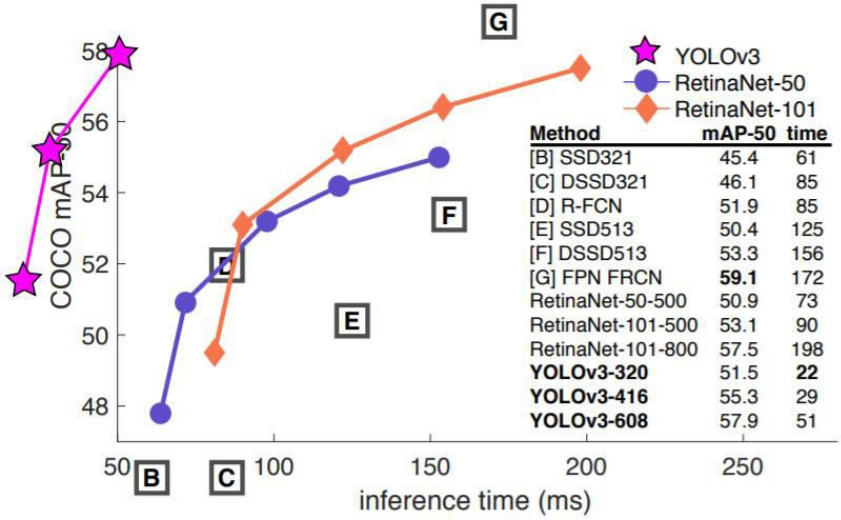

Fig. 2. YOLOv3 vs. other algorithms (Redmon et al., 2016)

CNNs improve the input picture by making it smaller due to its structure. This makes small objects difficult to recognize. As a solution, YOLO used a different method in its architecture. For example, it brings a 28x28x512 layer to 14x14x2048 and adds it to the back of the 14x14x1024 output layer (Redmon et al., 2016).

YOLOv5, unlike other YOLO versions, is implemented using a PyTorch library. CSP backbone and PA-NET neck are used as in YOLOv4. In the head part, the model used in YOLOv4 is used. "Leaky", "ReLU" and "Sigmoid" are used as activation functions. "Leaky" and "ReLU" activation functions are used in hidden layers in YOLOv5. "Sigmoid" activation function is used in the last detection layer. In YOLOv5, the default optimization function “SGD” is used for training (Alan, 2020).

In order to approach leaf diseases of plants and fires in forests through machine learning, firstly, the data set containing photographs of diseased-healthy plants and fire detection was trained with various models, Afterwards, object detection with the camera was made by YOLO and instant results were obtained by comparing the obtained test results with the camera image. In this way, disease detection and fire detection can be made.

The *kaggle.com* platform was used while training the disease detection model in the plant leaf. The model was trained by downloading the ready-made data sets from here. In this data set, there are 1495 healthy and 2084 diseased leaf photographs. These visuals have been turned into a certain standard model while training. 2 labels are used. These have been named "diseased" and "healthy" (Fig. 3).

While creating the data set, the folder extension containing the data was defined first. Next, an empty directory has been defined. All images are then sized to be 128x128 in height and width. Images were brought to 128x128 matrix size and saved in the NumPy array as float32.

Before starting the training, a configuration file was created. This file contains the file extensions of the data sets, the names of the class tags, the ratio of the train and test data, the initial learning rate, batch size, epoch values, the file extension of the model to be saved after the training, the file extension of the Learning Rate Finder graphic, the file extension of the graphic showing the learning history after the model was trained, the extension of the file that shows how many data will be taken mixed from the data set in combination after the model is tested and where it will be saved after it is tested.

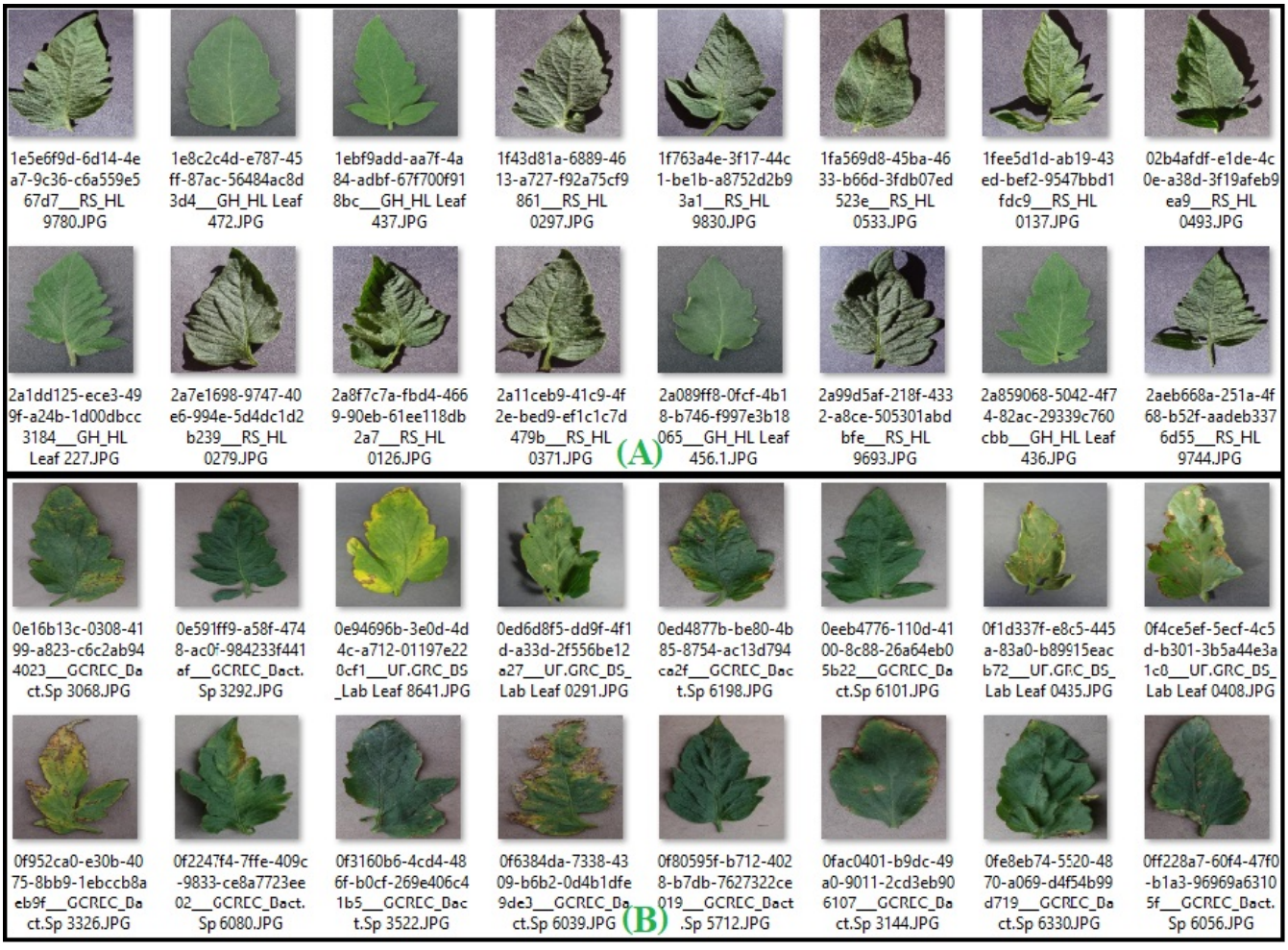

Fig. 3. A visual of the data set containing (A) healthy leaves and (B) diseased leaves

4. Implementation and experiments

4.1. Creating the model

Keras library is used to create a CNN model. The CNN model we used has been developed by creating differences on the previously prepared LeNet model. LeNet model is one of the first published CNN models. The behavior of this model was first demonstrated by Facebook artificial intelligence director Yann Lecun. It was then used by researchers working at AT&T Bell Laboratories to identify handwritten numbers. This model consists of six layers: input, two convolutions, two pooling, hidden, and output (Britz, 2015).

To create the LeNet model, Dense, Conv2D, MaxPooling2D, Flatten, Dropout, Convalution2D functions, and Sequential model among the layers defined in the Keras library were used. It is aimed to change the parameters on an existing LeNet CNN model to increase the performance.

- In the first convolution layer, the number of filters was chosen as 16.
- In the first convolution layer, the activation function used is Rectified linear unit (ReLU).
- In the convolution layer 32 filters used following the first pooling layer.
- In the activation function preceding the second pooling layer, ReLU is again chosen.

In the summary of the model, the size of each layer and the parameter count to be trained are written. The number of parameters to be trained in this model is 4213317.

4.2. Training the model

The previously prepared data sets for the training of the model have been divided to two parts in certain proportions: a training set, and a validation set. This ratio is divided into 0.25 validation and 0.75 training set. The disease detection data set consists of 3579 images in total. In this data set, 25% verification corresponds to 894 numerical values. The remaining 75% corresponds numerically to 2685 images.

After the training process of the model was completed, it was saved in the configuration file. The training works were carried out on the platform named Colab on Google. Before performing the training, all necessary codes and data sets were uploaded to the cloud. In this way, faster training was achieved. The training period lasted approximately 15 minutes in the fire detection data set, and approximately 1.5 hours in the disease detection data set. The reason why the detection of the disease takes so long is that there is more data.

4.3. Testing the model

A python code was written to test the model whose tutorial was completed. In this code, the model is tested using the data sets used while training. While creating the test file, random samples are taken from this data set. These examples are used on the model that has been trained and recorded, and the visual event is detected. After completing the test on the visual of the model, it is printed in color on the selected visual in the upper left part (Fig. 4). There were also some incorrect detection results as can be seen in red frames.

4.4. Model results

This project has been carried out to detect diseases and fire in plants using artificial neural networks. The data set used in disease detection was found ready on the internet. The fire detection data set was also available online. The data sets used in our project were obtained by developing the LeNet model. In the model we have created, performance improvement is made according to the LeNet model. The number of visuals in the data sets we have used are given in Table 1.

Table 1. Amount of the disease data set images

Disease detection data set	***Total number***
Diseased leaf images	1495
Healthy leaf images	2084

The disease detection model took approximately 1.5 hours. These pieces of training were also carried out on Google Colab. The error and accuracy rates obtained on the validation data set are 0.1643 and 0.9338, respectively. The model was tested using the test data set after completing the training. The training results of the disease detection model can be seen in Fig. 5. The blue lines in Fig. 5 show the variation of (a) error and (b) accuracy values according to the number of epoch (iteration) on the verification data set. Red lines show the variation of error and accuracy values according to the epoch number on the training data set.

As can be seen in Fig. 5, the education error in the 22nd Epoch decreased below 0.14 and the verification error fell below 0.17. The verification error did not descend to a lower level after this point. Even as the training error decreases to lower values after this point, the verification error starts to rise in contrast to it. For this reason, no important Over Fitting was seen until the 22nd epoch, when the best learning performance occurred. After the 22nd epoch, the model started to memorize which means it has lost its ability to classify images that are not in the data set, in other words, the ability to give correct results for images that the model has not seen before.

4.5. Comparison of training models

The performance of deep learning algorithms has been demonstrated by using deep learning against many methods used for the classification of plant leaves.

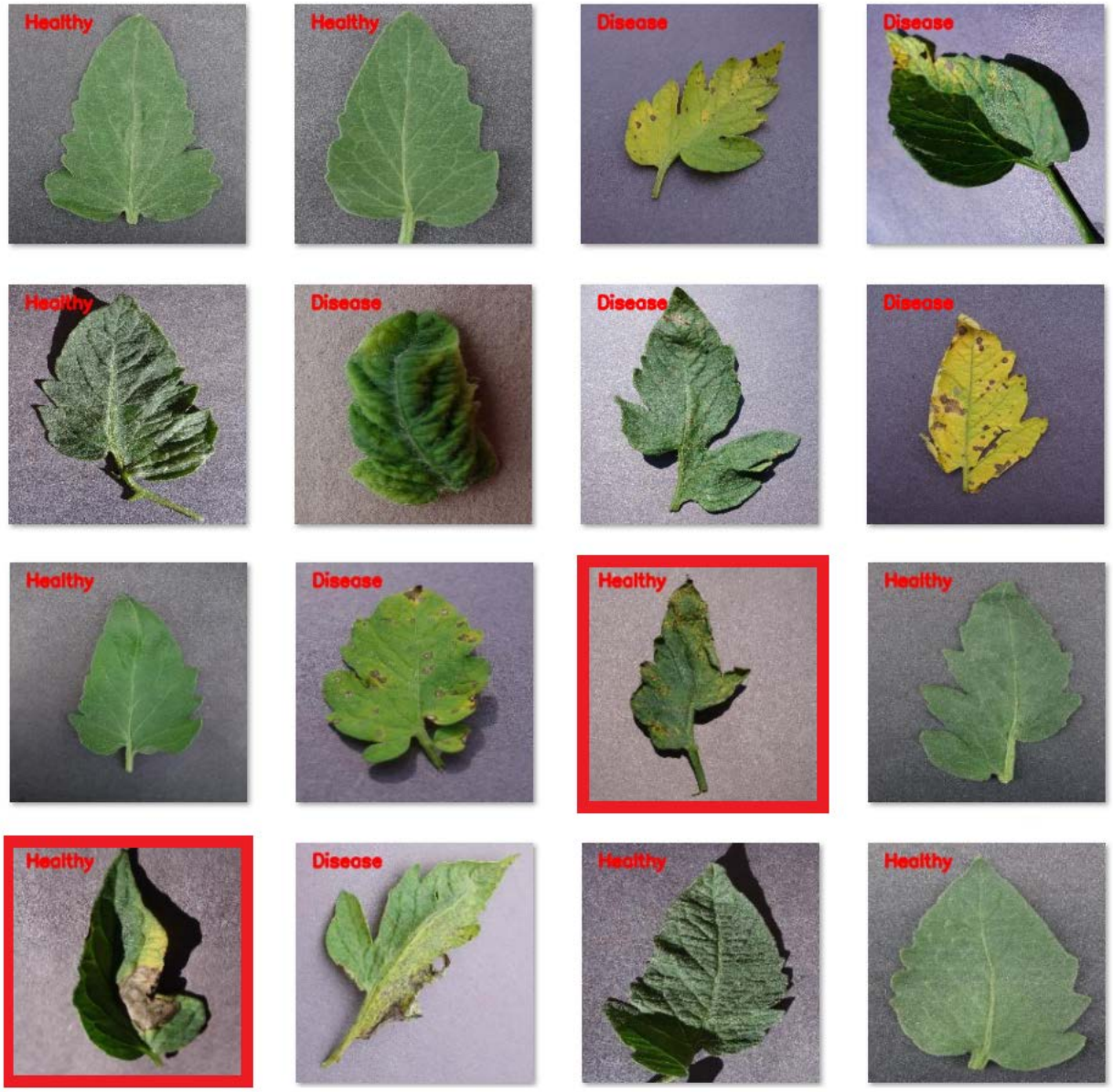

Fig. 4. Visual test results of the disease detection model (red frames show the incorrectly detected images)

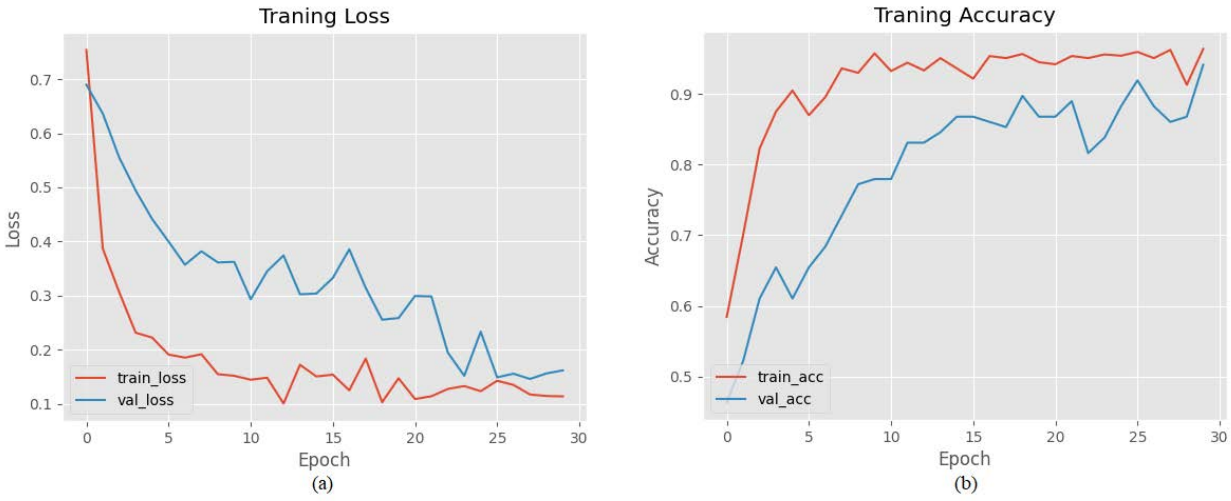

Fig. 5. Change graph of (a) error and (b) accuracy dependent on epoch number of the developed model

Plant diseases encountered in agriculture are an important issue and the solution to the problem can be facilitated by artificial intelligence. In classification processes made with image processing methods, results can be obtained through pre-process feature extraction and classification method. There is no need for such processes in deep learning methods. In deep learning methods, stages such as pre-processing and feature extraction are performed with Convolutional Neural Networks.

In our study, the training of CNN models of diseases that consist of 4 classes, including late blight, early blight, bacterial spot, and leaf mold, and the performance of these models were compared. 7355 photographs were used for training of the models and 1838 photographs were used for testing. In deep learning models, GoogleNet (InceptionV3), ResNet50 and VGGNet (16,19) are used.

5. Results and discussion

YOLOv5 is an algorithm capable of real-time object detection. It is also one of the object detection methods that gives high accuracy. YOLOv5 was published on the GitHub platform (Nelson and Solawetz, 2021). The data must be arranged in accordance with the algorithm. As with deep learning, deep data must first be organized and brought into a form that the algorithm can understand. Unlike deep learning, the coordinates and classes of the objects on the pictures used for YOLOv5 training must be written into the "txt" file. A variety of programs are used for this. The program named "LabelImg" was used for the data of this study. The disease detection model was applied on tomato leaves for the study. 5 classes in total as shown in Table 3 have been used for this purpose. Although it is divided into classes, it is difficult to train leaf data in terms of leaf geometry and thousands of data are needed. For this reason, a total of 11791 images were used with the help of the "LabelImg" program.

However, it will take a lot of time to classify 11791 images. A method was used to solve this problem. Each picture in the data allocated to classes contained only the picture of that class. It was aimed that each picture was named and recorded with the class number and an average coordinate value. An algorithm has been written for this. Before the algorithm was implemented, all pictures were processed in the *LabelImg* program. Then txt files created for classes and label for YOLOv5 were put into folders. While doing this, it is necessary to separate both pictures and labels into two parts as "train" and "validation". Thus, the data in the study was organized in a way that YOLOv5 can understand. After this step, the path of the train and validation data, the class number, and the names of the classes should be written into the "*yaml*" file. After the data and "*yaml*" file was prepared, the data training phase was completed.

At this stage, the train.py library in YOLOv5 was used. In the settings made with the editor program, the number of epoch, batch size, image size of the pictures in the training data, the file path of the "*yaml*" file, and the YOLOv5 model must be set separately. YOLOv5 model selection is one of the most important parameters that should be selected correctly, as it directly affects training time and accuracy. Fig. 6 shows that as the accuracy rate increases, the training time increases. Despite the sufficient data count, YOLOv5I model takes much more time compared to the YOLOv5s model, thus YOLOv5s model was chosen for the study. Settings that have been chosen for the training are given in Table 4.

Table 2. Error and accuracy values of fire and disease detection data sets

Model	***Training Time***	***Training Loss***	***Training Accuracy***	***Validation Loss***	***Validation Accuracy***
Disease Detection	90 min	13.64%	94.48%	16.43%	93.38%

Table 3. Data class and number of images

Disease type	***Train data set images***	***Validation data set images***
Healthy leaf	1926	481
Yellow leaf	1961	490
Septoria leaf spot	1745	436
Tomato leaf mold	1882	470
Early blight	1920	480
Total data count	9434	2357

Table 4. Training settings

Number of epoch	16
Batch size	8
Image size	256 x 256
Image count	Train 9434 -Validation 2357
Model	YOLOv5s
Total training time (hours)	16.4

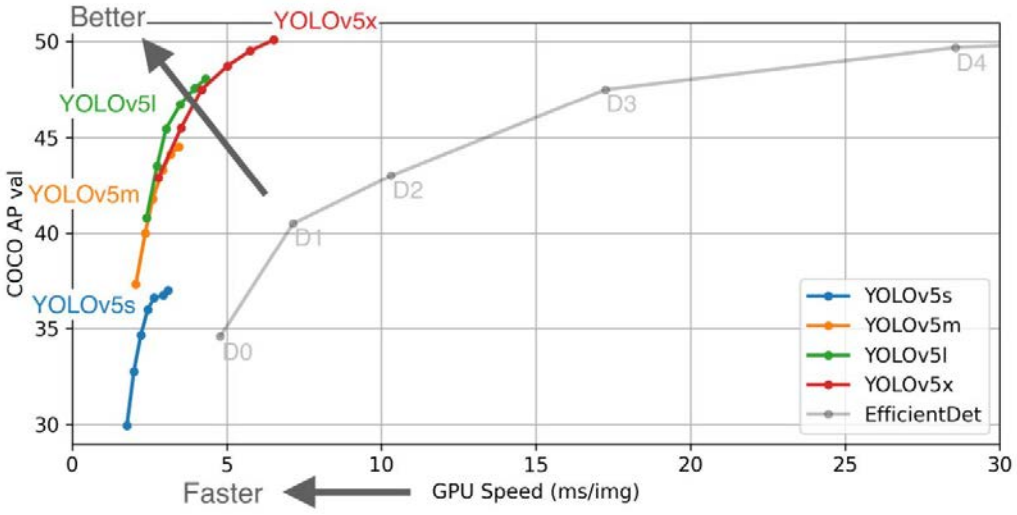

Fig. 6. Model comparison on YOLOv5 Coco data

Total training time took 16.4 hours with the computer which has an i7-7500U CPU but can be faster on computers with higher CPU models, and epoch time was approximately 1 hour. During the training, the training was interrupted 2 times due to meaningless errors.

5.1. Training results

The study results are shown in Fig. 7 step by step according to the epoch number. YOLOv5 ensures the security of the data by uploading the training data to the site called "wanb.ai". In the training results, there are graphs of loss functions of "train" and "validation" values, precision, recall, 0.5 average sensitivity (mAP@0.5) and 0.5-0.95 average sensitivity (mAP@0.5:0.95). Overall, better results were obtained in each step. From this inference, better results can be obtained with more epochs.

Mean Average Precision (mAP) is a system where values such as sensitivity and precision come together. Considering the precision rates, it is seen that the third epoch reaches almost the maximum result. Higher results resulted in a mean average precision of 0.5 (mAP@0.5). The maximum result was 99.54%. The precision between 0.5-0.95 (mAP@0.5:0.95) was a maximum of 77.59%. Based on these results, as the mean average precision value increased, the obtained value decreased.

Recall is the value that shows how much of the transactions that need to be estimated correctly are correctly predicted. As can be seen in Fig. 7, we can say that a high value has been obtained. It roams around 99% after the third epoch happens and the highest value is 99.96%.

Precision shows how many of the values we correctly predicted are actually correct. In Fig. 7, it is seen that it increases gradually, although not continuously. It was steadily increasing between the third epoch and the thirteenth epoch, while it tended to decrease or increase in subsequent steps. It has the highest value of 92.96%.

Loss functions were created separately as "train" and "validation". Looking at the loss functions, it is seen that the "train" loss functions are constantly decreasing, while on the "validation" side, it has not decreased as smoothly as the "train". The reason for this is that the "train" side learns by questioning the "validation" side during training. We can explain this phenomenon as follows: Considering that the "Train" data set is correct; the algorithm is determined. In the "Validation" section, the applied model is tried to be improved. “Box” and “val Box” graphs are GIuO (Generalized Intersection over Union) which is the improved version of IuO (Intersection over Union) (Rezatofighi et al., 2019). When the graph is examined, it is seen that the "train" side gets a lower value than the "validation" data set. The GIuO loss function value of the "train" data set is at least 0.01168 while it is 0.01045 in the "validation" data set.

"Objectness" and "val Objectness" represent the objectivity loss function. This function is constantly decreased in the "train" data set. However, there was a constant change in the "validation" data set. It took the value 0.04629 as the lowest value in the "Train" data set. "Classification" and "val Classification" show the classification loss function. Unlike other loss functions, the classification loss function shows a continuous decrease in two data sets. The smallest classification loss function value of the "train" data set is 0.01121. In the "validation" data set, it has the smallest value of 0.001681.

5.2. Test

After the training is completed, even if the information can be obtained through numbers and graphics, it cannot be understood whether it is successful or not. Testing is done via "detect.py" in the YOLOv5 file.

The first thing to do in the "detect.py" is to write the file path of the weights of the training. As can be seen below, weights as the result of training, two weights have come out as a result of the training.

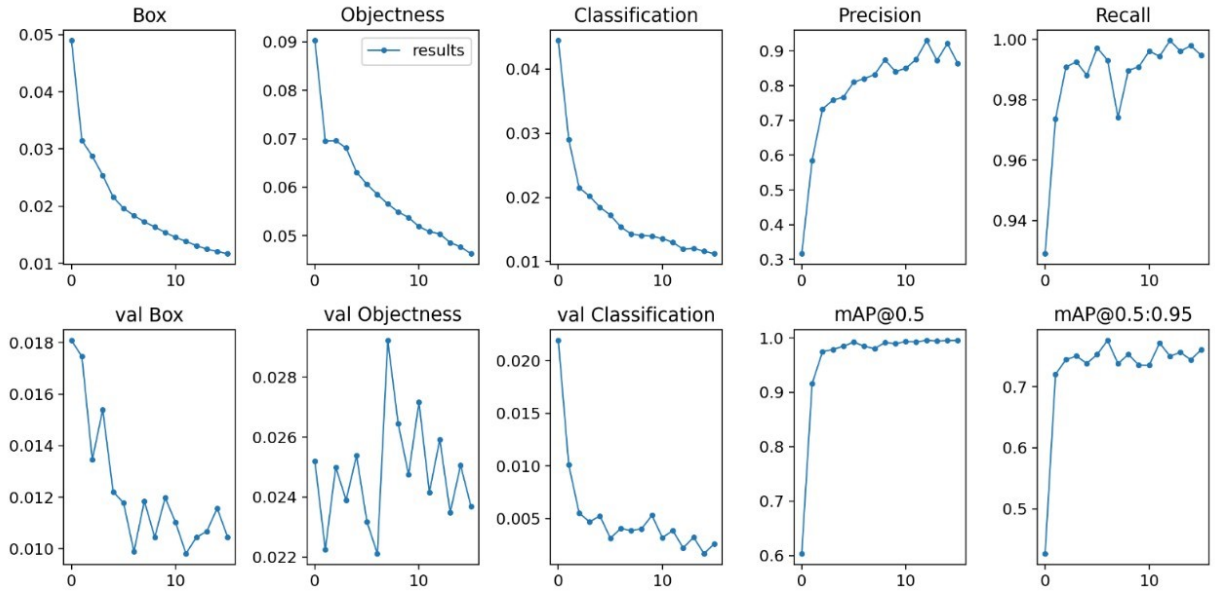

Fig. 7. Training results

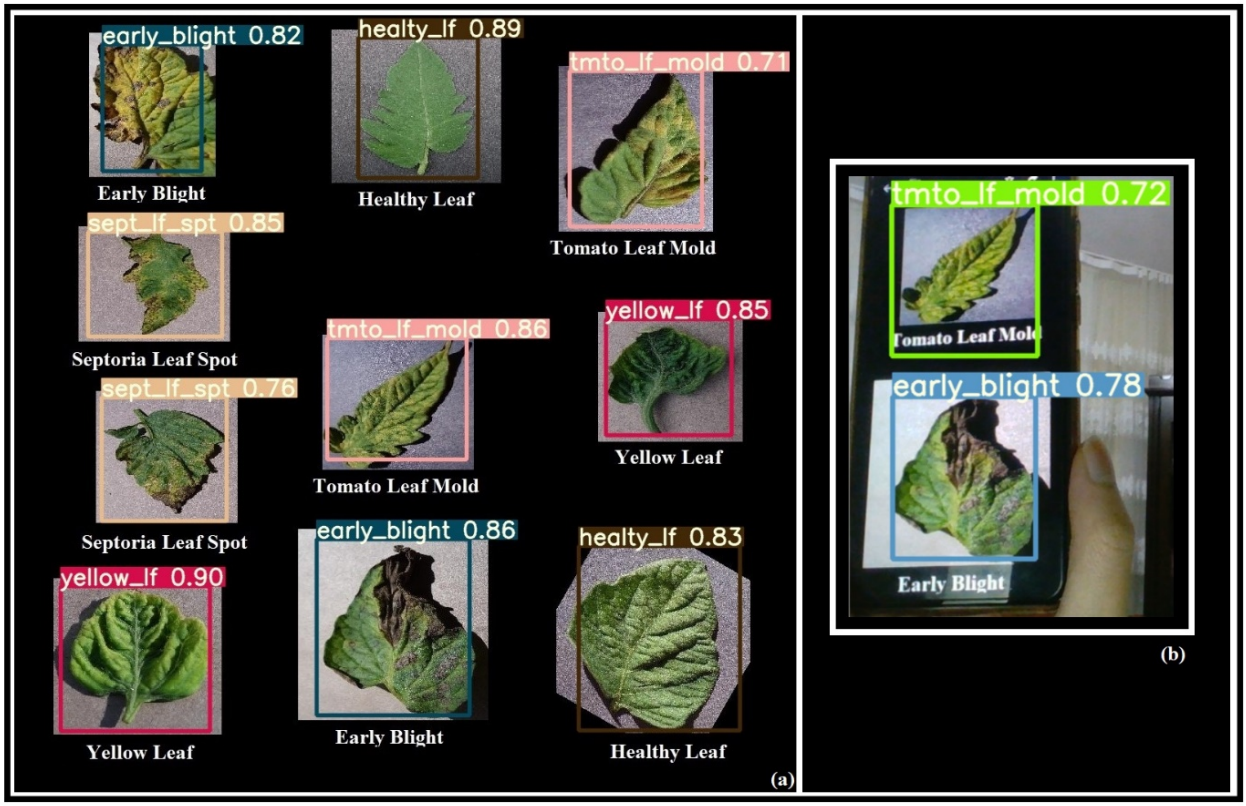

Fig. 8. (a) Image test and (b) Real-time testing

- Optimizer stripped from runs\exp32_wm\weights\last.pt, 14.8MB
- Optimizer stripped from runs\exp32_wm\weights\best.pt, 14.8MB
- 16 epochs completed in 16.427 hours.

The first of these weights is the weight of the last epoch, and the other is the weight of the epoch where the best results are obtained. In this study, the weight named "*best.pt*" with the best results was chosen. Visual data are required for testing. For this, a picture can be used as well as test data consisting of many pictures such as a video. For this, the file path is written in the "source" section where the test images are located. YOLOv5 can test on a laptop as it allows real-time testing. For this, it is sufficient to write "0" instead of the file path. The next thing to do is to write the size of the visual data to be tested. This test algorithm has many options such as processor

selection. However, in this study, other options were used by default.

In the study, 10 pictures in total, 2 from each class, were tested. The pictures were placed in random places by changing the image size, and as a result of the test, Fig. 8 (a) was successfully obtained. YOLOv5 specifies the names of the previously specified classes and how many percent of the objects are correct when outputting. It also helps to distinguish classes by giving each object class a corresponding color.

As can be seen in Fig. 8 (b), a real-time testing was also carried out successfully. Many factors affect performance in real-time testing. In particular, the power of the system used directly affects the number of images per second (FPS). The quality of the camera is important in detecting objects. Because many things such as the light coming into the camera and the image quality make the image look different than it is. These situations can cause objects to be detected incorrectly or not detected at all while testing.

5.3. Comparison

In this section, the comparison of our results with the study of Sladojevic et al. (2016) is given in Table 5. The results show that YOLOv5 can learn with less epochs with less training, using its more complex neural networks.

Table 5. Comparison with Sladojevic et. al. (2016)

	Sladojevic et. al. (2016)	***Our study***
Algorithm	CaffeNet CNN	YOLOv5sI
Number of epoches	30	10
Training accuracy	95.8%	94.48%
Dataset size	30000	11500

6. Conclusions

The primary goal in agricultural production is to ensure sustainable, efficient and economical management in production. Nowadays, the use of more effective technologies facilitating agricultural operations needed.

The use of artificial intelligence in agriculture can create a more profitable, efficient and sustainable agricultural industry, and can greatly reduce the consumption of additional natural resources, unnecessary consumption of clean water, soil and water pollution caused by excessive use of agricultural chemicals, and waste of energy in agricultural vehicles and workers.

Plant diseases are an inevitable part of nature, and therefore of agriculture. Despite all the advanced crop protection culture and technologies, plant diseases are the leading problems that harm crop production. The length of time spent to detect the disease or pest that damages the plant is one of the factors that carry plant losses to a critical point. With artificial intelligence in agriculture, it is possible to shorten the time between the detection of the problem and the beginning of the control and to achieve high efficiency with rapid intervention. In addition to the time, efficiency and financial benefits gained by disease detection and diagnosis, which can be performed within a few seconds, minimum damage is provided in the specified environmental problems.

YOLOv5 was released shortly after YOLOv4. YOLOv5 was launched on GitHub in June 2020. YOLOv5, unlike other YOLO versions, is implemented using a PyTorch library. CSP backbone and PA-NET neck are used as in YOLOv4. In the head part, the model used in YOLOv4 is used. "Leaky", "ReLu" and "Sigmoid" are used as activation functions. "Leaky", "ReLu" activation functions are used in hidden layers in YOLOv5. "Sigmoid" activation function is used in the last detection layer. In YOLO v5, the default optimization function “SGD” is used for training.

More reliable results were obtained with this developed artificial neural network model. Real-time object detection was achieved with YOLOv5, which is used in object detection. The reason for choosing the YOLOv5 algorithm in the developed model is that YOLOv5 gives results with very high accuracy.

The model developed using artificial neural networks was designed as a solution to one of the biggest problems in agriculture, the agricultural disease issue. As a result of the researches and applied models, an automatic diagnosis system was tried to be developed for this problem. Some errors were encountered in the created applications. As a result of these errors, changes were made to the model and the best working model was developed.

References

Alan M., (2020), *Biosignal Classification and Disease Prediction with Deep Learning (in Turkish),* MSc Thesis, Marmara University, Istanbul, Turkey.

Ali R.B., Bouadila S., Mami A., (2018), Development of a Fuzzy Logic Controller applied to an agricultural greenhouse experimentally validated, *Applied Thermal Engineering*, **141**, 798-810.

Bock C.H., Poole G.H., Parker P.E., Gottwald T.R., (2010), Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging, *Critical Reviews in Plant Sciences*, **29**, 59-107.

Bre F., Gimenez J., Fachinotti V., (2017), Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks, *Energy and Buildings*, **158,** 1428-1441, https://doi.org/10.1016/j.enbuild.2017.11.045.

Britz D., (2015), *Understanding Convolutional Neural Networks for NLP*, Online at: http://Www.Wildml.Com/2015/11/Understanding-Convolutional-Neuralnetworks-for-Nlp/ (Visited on 11/02/2021).

Deng L., Yu D., (2014), Deep learning: methods and applications, *Foundations and Trends in Signal Processing*, **7**, 197-387.

Dokic K., Blaskovic L., Mandusic D., (2020), *From Machine Learning to Deep Learning in Agriculture-The Quantitative Review of Trends*, IOP Conference Series: Earth and Environmental Science, Changchun, China, **614**, 12138, http://dx.doi.org/10.1088/1755-

1315/614/1/012138/.

Faridi M., Verma S., Mukherjee S., (2018), *Integration of GIS, Spatial Data Mining, and Fuzzy Logic for Agricultural Intelligence*, In: *Soft Computing: Theories and Applications*, Springer, 171-183.

Kale S.S., Patil P.S., (2019), *Data Mining Technology with Fuzzy Logic, Neural Networks and Machine Learning For Agriculture*, In: *Data Management, Analytics and Innovation*, Sharma N., Chakrabarti A., Balas V.E. (Eds.), Springer, Singapore, 79-87.

Kamilaris A., Prenafeta-Boldú F.X., (2018), A review of the use of convolutional neural networks in agriculture, *The Journal of Agricultural Science*, **156**, 312-322.

Kirkaya A., (2020), Smart farming- precision agriculture technologies and practices, *Journal of Scientific Perspectives*, **4**, 123-136.

Kurniasih D., Jasmi K.A., Basiron B., Huda M., Maseleno A., (2018), The uses of fuzzy logic method for finding agriculture and livestock value of potential village, *International Journal of Engineering & Technology*, **7**, 1091-1095.

Malik H.S., Dwivedi M., Omkar S.N., Javed T., Bakey A., Pala M.R., Chakravarthy A., (2021), *Disease Recognition in Sugarcane Crop Using Deep Learning BT*, In: *Advances in Artificial Intelligence and Data Engineering*, Chiplunkar N.N., Fukao T. (Eds.), Springer, Singapore, 189-206.

Matha T.W., Jati A.N., Azmi F., (2016), Fuzzy logic as a method of decision making in automatic watering plants, *Journal of Measurements, Electronics, Communications, and Systems*, **2**, 15-21.

Nelson J., Solawetz J., (2021), *Responding to the Controversy about YOLOv5*, Online at: Https://Blog.Roboflow.Com/Yolov4-versus-Yolov5/. https://blog.roboflow.com/yolov4-versus-yolov5/

Oppenheim D., Shani G., (2017), Potato disease classification using convolution neural networks, *Advances in Animal Biosciences*, **8**, 244.

Redmon J., Divvala S., Girshick R., Farhadi A., (2016), You only look once: Unified, real-time object detection, Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 779-788.

Rezatofighi H., Tsoi N., Gwak J., Sadeghian A., Reid I., Savarese S., (2019), *Generalized Intersection over Union: A Metric and a Loss for Bounding Box Regression*, Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 658-666.

Ruiz-Real J.L., Uribe-Toril J., Torres Arriaza J.A., de Pablo Valenciano J., (2020), A look at the past, present and future research trends of artificial intelligence in agriculture, *Agronomy*, **10**, https://doi.org/10.3390/agronomy10111839.

Sladojevic S., Arsenovic M., Anderla A., Culibrk D., Stefanovic D., (2016), Deep neural networks based recognition of plant diseases by leaf image classification, *Computational Intelligence and Neuroscience*, **2016**, 3289801, https://doi.org/10.1155/2016/3289801.

Terzi İ., Ozguven M., Altaş Z., Uygun T., (2019), *Using Artificial Intelligence in Agriculture* (in Turkish), International Erciyes Agriculture, Animal & Food Sciences Conference Book, Kayseri, Turkey, 245-255.

Uijlings J.R.R., Van De Sande K.E.A., Gevers T., Smeulders A.W.M., (2013), *Selective search for object recognition*, *International Journal of Computer Vision*, **104**, 154-171.

Wang G., Sun Y., Wang J., (2017), Automatic image-based plant disease severity estimation using deep learning, *Computational Intelligence and Neuroscience*, **2017**, 2917536, https://doi.org/10.1155/2017/2917536.

Wspanialy P., Moussa M., (2020), A detection and severity estimation system for generic diseases of tomato greenhouse plants, C*omputers and Electronics in Agriculture*, **178**, 105701, https://doi.org/10.1016/j.compag.2020.105701.

Wu D., Lv S., Jiang M., Song H., (2020), Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments, *Computers and Electronics in Agriculture*, **178**, 105742.