Halyomorpha Halys Detection Using Efficient Neural Networks

Alexandru Dinca1[0000-0003-4752-2183], Nicoleta Angelescu2[0000-0003-4187-6239], Loretta Ichim1[0000-0002-7465-3958], and Dan Popescu1[0000-0002-1883-0091]

1University POLITEHNICA Bucharest, Romania
2University VALAHIA Targoviste, Romania
marius.dinca1411@stud.acs.upb.ro, nicoleta.angelescu@valahia.ro, loretta.ichim@upb.ro,dan.popescu@upb.ro

Abstract. In previous years, there has been a strong increase in working models in the areas of artificial intelligence development based on high-performance hardware and software resources. The agricultural industry was also placed in this area of development, which benefited greatly from the research and tools that emerged. The pest detection area was the main aspect considered and the attachment of image processing modules motivated the adoption of such techniques. After analyzing the existing solutions, the paper implemented two convolutional neural network architectures adapted for the automatic detection of Halyomorpha Halys pests by classifying images taken from orchards. The new version of the Python programming language, its attached libraries, and architectures adapted from EfficientNet and Google Inception V3 was used to define, train, and test the proposed architectures. The good results obtained (accuracy between 0.92 and 0.95) make it possible to implement an efficient system for detecting and monitoring harmful insects in a complex environment such as orchard trees.

Keywords: Image Processing, Neural Networks, Orchard Monitoring, Insect Detection.

1 Introduction

A recent report by Stanford University (Artificial Intelligence Index Report) [1] shows the impact that artificial intelligence has had in a multitude of domains and the fact that it targets highly diversified areas. In the interval 2019-2020, the publications on the artificial intelligence area increased considerably by a percentage of 34.5%, considering the increase of 19.6% from the previous interval 2018-2019. In this sense, the area of artificial intelligence has also had a major impact in the field of agriculture or biology - they are rapidly developing tools for data analysis, complex representations of them, or pest detection.

For agricultural areas, the algorithms of pest detection and monitoring have enhanced automation processes and provided stakeholders or farmland owners with an overview of the areas they are considering. Automatic pest detection and monitoring using remote-controlled imaging systems is one of the key steps in these areas.

Following the evolution of ImageNet systems that provide a graph of performance over time, there has been an upward trend in research areas that focus on increasing the efficiency of the training of image recognition and classification systems. Although the ImageNet – based module is very popular and developed accordingly, existing computer vision systems do not always provide the desired results. In this case, there are a lot of research works that look at the robustness and performance of image classifiers based on hand-built and custom datasets.

The area of implementation and subsequent developments of the convolutional neural networks (CNNs) are limited to the construction of algorithms and their optimization for the architectures capable of classifying or detecting objects in images. Agricultural ecosystems are one of those areas where algorithms have had a massive impact on understanding the processes involved. Careful monitoring and analysis of various aspects are very important in this field, and it has often been desired to integrate IoT systems and remote-controlled devices to meet these needs.

At the same time, the integration of revolutionary technologies has helped agricultural landowners to reduce monitoring costs and carefully introduce non-invasive tools to detect the issues they have in mind. As the authors in [2] pointed out, the monitoring areas were limited to the automatic acquisition and analysis of digital images to identify pests for reference agricultural ecosystems. It has been observed that the integration of computer vision modules and artificial intelligence can meet the need for fast, timely, and accurate detection of areas of interest, compared to manual identification which requires execution time, is prone to errors, and is often done by experts in the field.

Given the automatic acquisition of digital images, research areas were later developed for the implementation of algorithms capable of extracting useful information from these images using deep learning methods based on CNNs. The convolutional neural networks have proven useful in satisfying these aspects of image classification or segmentation, used in practical work as input data sets for specific algorithms. The methodologies and architectures are diverse and present various values of performances. The CPAFNet neural network model developed in [3] was used to identify common pests of agricultural ecosystems. The CNN was modeled accordingly, and the key parameters were gradually transformed based on the data set to obtain a high-performance model. As a classification area, pests were recognized based on a 3- fold validation method. To validate the proposed model, the test experiments were based on VGG16, Inception V3, and ResNet50 architectures.

In the same trend, the authors in [4] proposed a solution for automatic pest detection based on images taken from mustard and bean crops. Regions of interest are properly extracted using Wavelet transformation and image fusion techniques. At the same time, the research area allowed the integration of the proposed solution with real-time monitoring networks based on IoT or wireless sensor networks.

Because of its characteristics of very high mobility and increased reproductive potential, Halyomorpha Halys (HH) is a harmful pest, spreading today throughout the European continent. They are causing significant damage to agricultural crops and especially to orchards. In this context, the population feeds on fruits and seeds, bringing significant mutations to plant products, being almost impossible to market them. Although solutions have been adopted, strongly based on the fight against HH using insecticides, they do not bring satisfactory results, such solutions being unfriendly to the environment and affecting the credibility or trust of the production staff. This paper aims to introduce a less invasive automation solution that will allow the acquisition of data from the orchards and then the identification and recognition of the target population for this type of pest. Thus, we proposed a solution for detecting harmful insects like HH in images using performant convolutional neural networks (CNNs) and, especially, a modified Single Shot Detector (SSD) [5]. The paper's goal was to implement and compare two efficient convolutional neural network architectures modified for the automatic detection of HH pests by classifying images taken from orchards. The new version of the Python programming language, its attached libraries, and architectures adapted from EfficientNet and Google Inception V3, was used to define, train, and test the proposed architectures. The CNNs used were training and tested on a new dataset created by the authors.

2 Materials and Methods

2.1 Convolutional Neural Networks Used

As is well known, a convolutional neural network is structured using several convolution layers, pooling (mean-pooling or max-pooling) layers, fully connected layers, and normalization layers [6]. The convolutional network type is a key element for many computer vision algorithms. In this case, the process shows the existence of a small matrix (called filter) which is passed over the reference image and transformed based on the filter values. We chose, modified, and tested two performant neural networks, EfficientNet and Inception V3, to detect HH in an orchard context.

EfficientNet [7] represented the network model that rethought the scaling mode of the architecture to optimize the performances. In general, the three scaling dimensions are represented by depth, width, and resolution. The authors of this paper provided a systematic research model for concrete balancing in their scaling to achieve notable performances. At the same time, the authors in [8] introduced new methods of network optimization through the efficient use of computing resources, increasing the number of input data, and the use of graphics processing units for training.

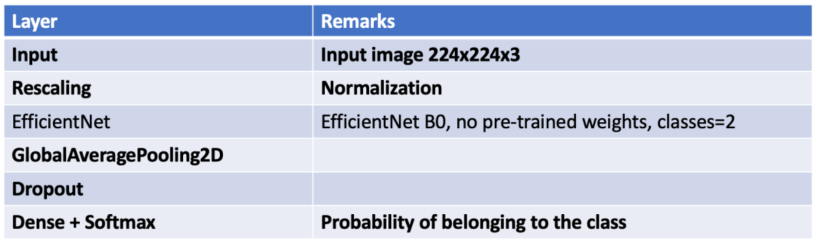
On the other hand, we discuss an alternative presented by Google, called Inception, that was proposed in the 2014 ImageNet Visual Recognition Challenge [9]. The Inception architecture brought to the forefront a revolutionary, high-performance technique for image recognition and detection algorithms. The complex architecture of the CNN Inception network features various techniques to increase performance in terms of both speed and model accuracy. Over time, it has seen steady growth, which has been reduced to several versions, V1 [10], V2-V3 (with quite similar structures) [11], and V4 [12], each new version showing considerable improvements compared to the previous one.

To increase the performance and relevance of a convolutional network, the techniques usually involve adding extra layers and computing areas that will ultimately increase the so-called depth of the proposed network [13]. In the case of the Inception architectures, the number of layers practically does not increase but goes wider, implementing several convolution areas of different sizes in the same layer. Therefore, choosing a standard kernel size is obviously difficult. For such variable dimensions, the need is introduced to create kernels of different sizes for the analysis of the distribution of areas of interest, because in a set of images the area occupied by the object of interest can be considerably different. The larger core is preferred for globally distributed areas of interest, and the smaller core is chosen for locally distributed information in the image.

The proposed Inception architecture has been tested on color images with a resolution of 299×299 pixels. Each Inception module performs 4 types of operations: convolution with kernels of size 1×1, 3×3, and 5×5, followed by a max-pooling layer to reduce the resolution of the feature map. The usefulness of 1×1 convolution reduces the depth, and the results of the operations are sent to form the block called FilterConcatenation. The mentioned global characteristics are captured by the 5×5 convolution layer, and the distributed ones are captured by the 3×3 layer.

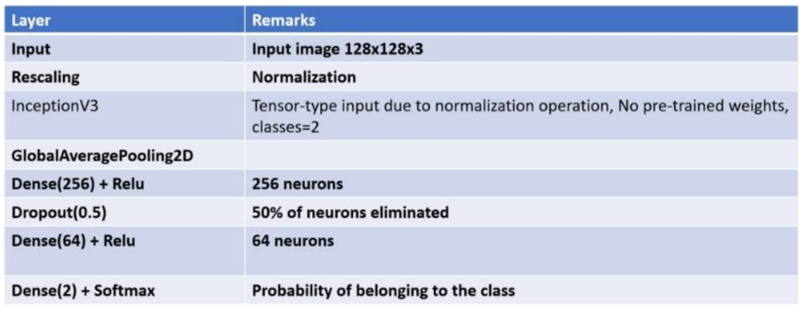
Compared to the EfficientNet network architecture, the drive time is similar for both architectures [13].

As first objective of the paper was to propose and implement a neural network modified from theEfficientNet architecture (CNN-1) and a modified architecture that includes the Inception V3 architecture (CNN-2) have been in this paper. Because the number of available images and the number of classes are small, modified architectures are necessary. For the CNN-1 network, we introduced new layers that are shown in Fig. 1. The new layers for the network CNN-2, modified and based on Inception V3, are shown in Fig.2. Using bold text, the changes and techniques brought to create the architectures for this work are shown. The total number of parameters for each network used is attached below each figure.



Layer	Remarks
Input	**Input image 224x224x3**
Rescaling	**Normalization**
EfficientNet	EfficientNet B0, no pre-trained weights, classes=2
GlobalAveragePooling2D	
Dropout	
Dense + Softmax	**Probability of belonging to the class**

Fig. 1. CNN-1 EfficientNet modified.



Layer	Remarks
Input	**Input image 128x128x3**
Rescaling	**Normalization**
InceptionV3	Tensor-type input due to normalization operation, No pre-trained weights, classes=2
GlobalAveragePooling2D	
Dense(256) + Relu	**256 neurons**
Dropout(0.5)	**50% of neurons eliminated**
Dense(64) + Relu	**64 neurons**
Dense(2) + Softmax	**Probability of belonging to the class**

Fig. 2. CNN-2 Inception V3 modified.

2.2 Dataset Used

Images with HH and without HH were hand-cut into patches (sub-images) of 128×128 size. These were made on different orchards, and the images were generated manually by tracking and photographing the individual insects, especially HH. The data set with such images were created by the authors. Examples of such patches for the learning phase are given in Fig.3.

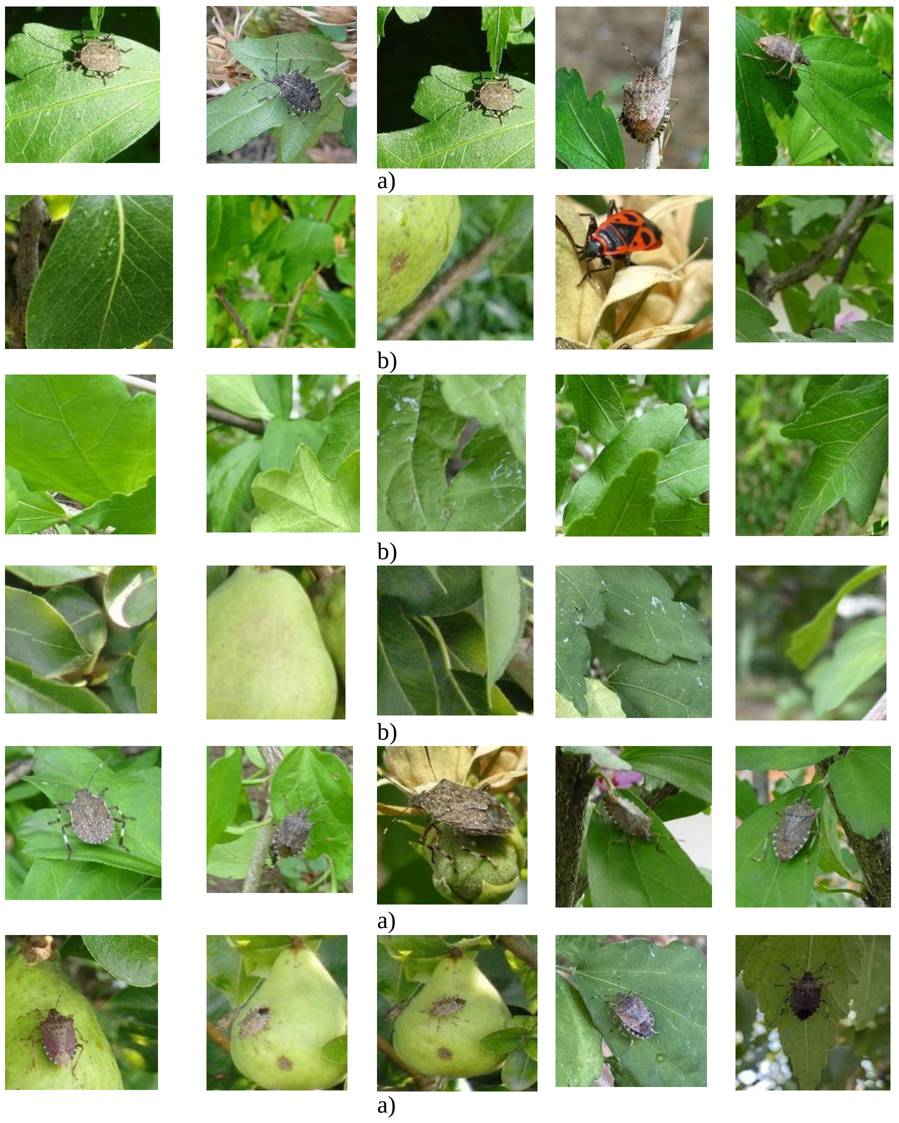


Fig. 3. Examples of learning images (patches): a) images containing HH, b) images without HH.

At the testing phase, each CNN receives 128×128-pixel color RGB patches at the input. These patches are generated by a sliding box algorithm. For both networks analyzed, the original image set was divided into two sets: training and testing. The patches were resized in the network training/testing process. The original image set (without augmentation) includes the following sets:

- Training set: 160 images with HH, 262 images without HH (nonHH).
- Testing set: 72 images with HH, 60 images without HH.

To increase the robustness of the network, the number of images in the original set for the training and validation stage was increased by augmentation operations [14].

The operations selected for image preprocessing are adaptive histogram equalization (CLAHE - Contrast Limited Adaptive Histogram Equalization), Gaussian noise generation, median filter, optical distortion, blur, etc.

The second objective of this paper was to describe and implement a robust, well-structured data set that would become a solid entry point for classification algorithms. In this regard, images of the HH were taken manually and integrated with other types of images with insects belonging to other families to give the algorithms a parallel path in the classification operation. In this way, the classification classes, denoted HH for the appearance of the harmful stink bug in the image and nonHH for its absence, for the images without HH or insects of any kind or not, were constructed theoretically. Furthermore, these classes have been transposed and implemented practically at the software level. The dataset is an original one and no other insect databases were used for this work. Furthermore, the data set was annotated manually, with increased care, using the popular bounding-box structures and then subsequently prepared for the training area.

2.3 Software Used

From the software point of view, the CNN network used to classify HH and nonHH was implemented in Python version 3.9, using the TensorFlow version 2.7 library developed by Google [15]. CNN's implementation used the Keras module in Tensorflow [15].

The number of network parameters is small, which reduces the complexity of the calculation in the network drive phase, respectively in the prediction phase. As mentioned earlier, the network input layer is a 128×128 pixels RGB color image, each color level is represented by 8 bits. Fig. 4 shows the organigram that is representative of the implementation of augmentation operations. Through the augmentation operation, a larger number of images is obtained: 2110 images as a training set and 500 images as a test set.

In the proposed network the convolution layers have a kernel size of 3×3 for each plane. The max-pooling operation operates on a 2×2 size window. The image is partitioned into 2×2 blocks, and each block is replaced by a pixel whose value is the maximum in the block (max-pooling operation). The first Conv2D layer generates 8 feature maps. Equation (1) shows the number of parameters (*No*) to be learned.

$$No = 8 \times (3 \times 3 \times 3 + 1) = 224 \tag{1}$$

For the first feature map, we learn a matrix of weights measuring 3×3×3 + bias. The third parameter in the multiplication operation is the number of planes in the image. Before entering, the pixels in each color plane are normalized in the range [0,1] by dividing by 255.

From the mathematical point of view, a feature map is the result of the activation function *f* applied to the convolution operation between the RGB image and the convolution kernel K to which we add the bias b to each pixel resulting from the convolution operation.

It is important to note that if the RGB image has P planes, the result of the convolution operation will be an image with the same number of planes. If we add the same bias b to each pixel, the result is a plane P image. The activation function reduces the number of planes to 1 and we get a feature map. The feature map will become the entry for the next layer of the network.

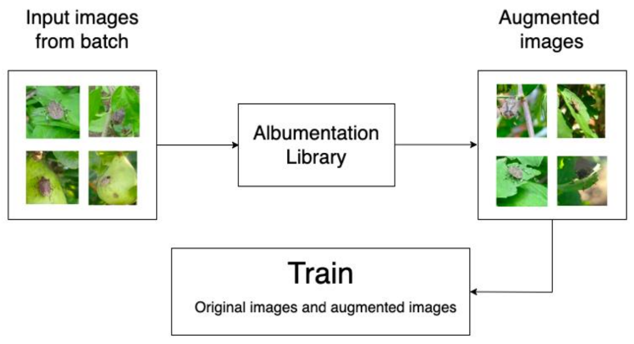
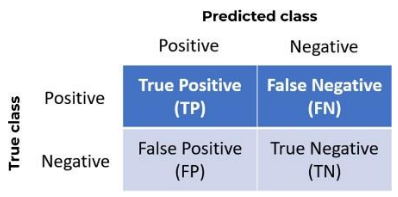


Fig. 4. The organigram for the augmentation process.

To reduce the number of calculation operations for the convolution operation the MaxPooling 2D operation was used. This reduced the resolution of the feature map. It can be observed the increase of the number of feature maps as the resolution decreases, which leads to a controlled increase in the number of network parameters. The Flatten operation turns a feature map into a vector. Then we apply through the Dense type of operation, and we build a fully connected layer of multilayer perceptron type (Multilayer Perceptron - abbreviation MLP). This MLP network will be used for classification. The activation function used is SoftMax. Because we have two classes (HH and nonHH), the function selected for optimization is binary cross-entropy. The lower its value, the better the classification. The number of epochs for the convolutional network is 25 epochs. The optimization algorithm is Adam, with a learning rate of 0.0001.

To assess the algorithm performance, the aim was to calculate the representative indices using the confusion matrix. This is a visual indicator of performance and provides an overview of the errors and performance of the classification algorithm [17]. A test dataset with the expected results is required to calculate the confusion matrix (Fig. 5). Then a prediction is made for each row in the dataset. Subsequently, the correct predictions for each class and the number of incorrect predictions, organized by the predicted class, are noted. Each row in the matrix corresponds to an existing class and each column to a predicted class. Finally, the number of correct and incorrect classifications is completed, with better visualization and interpretation of the data.



		Predicted class	
		Positive	Negative
True class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Fig. 5. The confusion matrix.

3 Experimental Results and Discussions

The evolution of the accuracy function for CNN-1 EfficientNet is shown in Fig. 6. a. and for CNN-2 Inception V3 in Fig. 6. b. The evolution of loss function for CNN-1 EfficientNet is shown in Fig. 6. c. and for CNN-2 Inception V3 in Fig. 6. d.

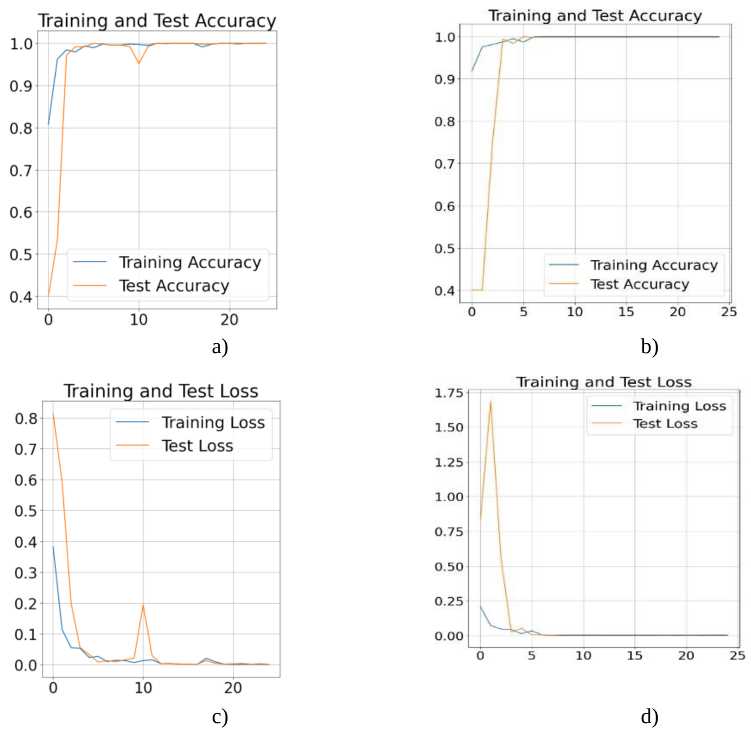


Fig. 6. Performance indicators for CNNs. a) Accuracy graph for CNN-1, b) Accuracy graph for CNN-2, c) Loss function graph for CNN-1, d) Loss function graph for CNN-2.

For each figure, the blue line shows the evolution of the functions in the training area, and the orange one reflects the evolution of the functions in the testing phase.

Examples from the testing dataset are shown in Fig. 7.

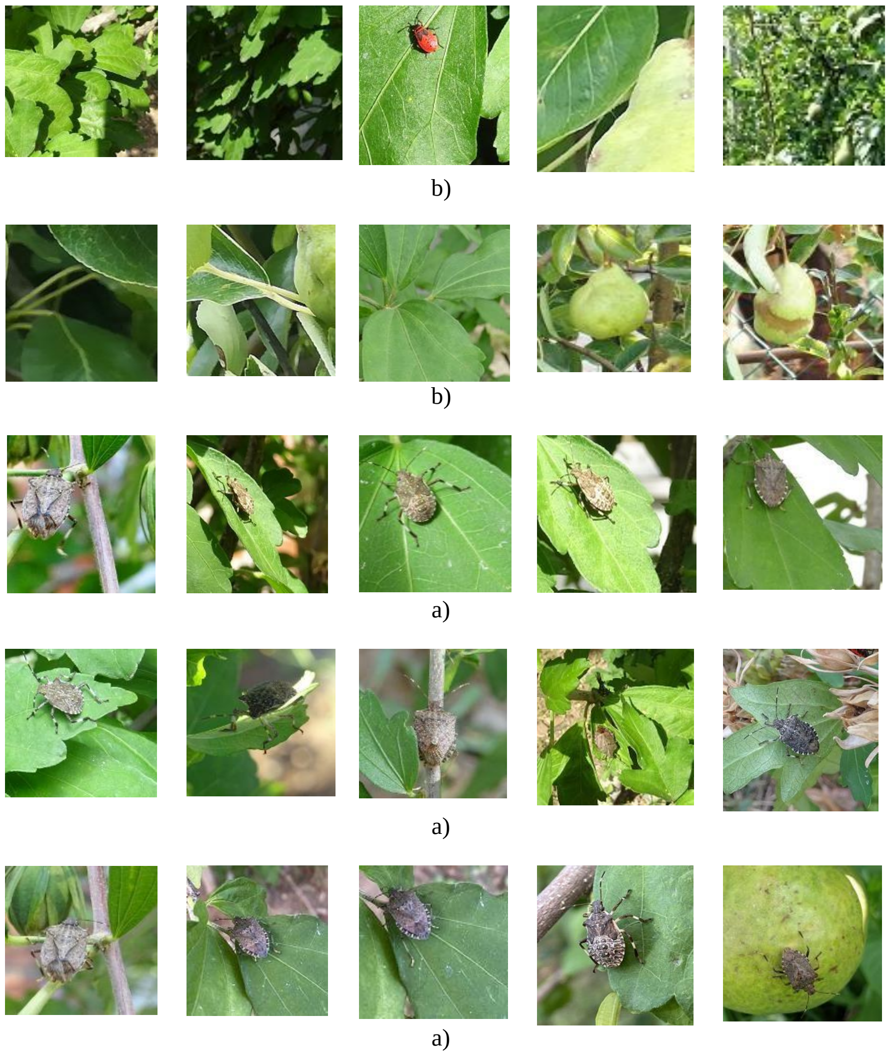


Fig. 7. Examples of testing images: a) images with HH, b) images without HH.

For the efficient visualization of the data and the performances, the implementation of the confusion matrix for each network was used. For CNN-1 EfficientNet, the non-normalized confusion matrix is attached in Fig.8. a. For CNN-2 Inception V3, the non-normalized confusion matrix is attached in Fig.8. b. For their implementation, 32 images with bugs (HH) were considered, respectively 61 images without HH from the data set.

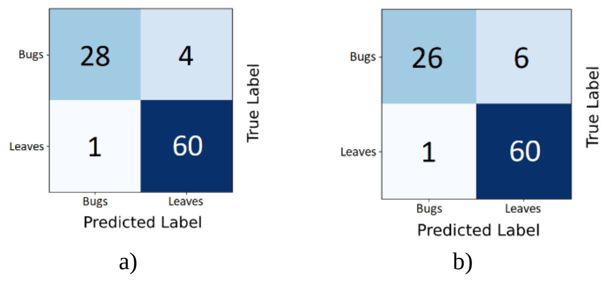


Fig. 8. Confusion matrices: a) CNN-1 EfficientNet, b) CNN-2 Inception V3.

Based on the confusion matrix, the performance evaluation was limited to the calculation of its representative indices: precision, sensitivity, specificity, accuracy, and F1 score. These indicators are presented for each network in Table 1. The first disadvantage of the proposed implementation for this paper is the small to medium size of the data set with images from orchards. For the development of an impact solution, a considerable data set of representative images for the population of HH may be entered, described, and annotated accordingly. Secondly, the images in the major set to be introduced may exemplify enough hypostases for the reference stink bug. The discussion and this implementation want to propose a set of images of various sizes, scaled or rotated, and with a considerable arrangement of the referenced bug. Very important that the acquisition of images is not perfect many times, the object of interest in the image may be small, cropped, or with reduced clarity.

Table 1. Performance evaluation.

Network	Precision	Sensitivity	Specificity	Accuracy	F1
CNN-1 EfficientNet	1.00	0.87	1.00	0.95	0.93
CNN-2 Inceptionv3	0.96	0.81	0.98	0.92	0.87

4 Conclusions

Following the experimental results described in this paper and considering the two proposed networks, a comparative study shows that the CNN-1 EfficientNet network performs better than the CNN-2 Inception V3 network, due to the complexity of the associated structure. The increased performance of the CNN-1 EfficientNet network is also visible in the associated confusion matrix. Sensitivity and accuracy are higher (so better) with CNN-1 EfficientNet. At the same time, considerable information and results werepresented, which allows the development of this work in the future by enlarging the setof images and by implementing more robust classification solutions. Given these strategies, for the present paper, it is possible to considerably improve the entry point and the learning model for the presented algorithms and architectures, the subject of the paper being one of interest. However, we can see from the data presented in this paper, that the advantages of the existing data set are the fact that the implemented architectures presented a good classification score. Also in this sense, the increase of the data set would imply the further development of the classification solutions. Finally, it was studied how the introduction of a convolutional network with considerable dimensions improved the metrics and values of performance indicators. As further work we intend to use the individual selected neural networks as subjective classifiers inside a combined multi-network system (as a collective intelligence) to better detection of HH in different hypostases inside the trees.

Acknowledgment

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI – UEFISCDI, project number 202/2020, within PNCDI III.

References

1. Zhang, D., Mishra, S., Brynjolfsson, E., Etchemendy, J., Ganguli, D., Grosz, B., Lyons, J., Manyika, T., Niebles J. C., Sellitto, M., Shoham Y., Clark J., Perrault, R.: The AI Index 2021 Annual Report. AI Index Steering Committee, Human-Centered AI Institute, Stanford University, Stanford, CA, (2021).
2. De Cesaro Jr, T., Rieder R.: Automatic identification of insects from digital images: A survey. Computers and Electronics in Agriculture, (178), 105784, 1-7 (2020).
3. Wang, J., Li, Y., Feng, H., Ren,.L. Du, X., Wu, J.: Common pests image recognition based on deep convolutional neural network. Computers and Electronics in Agriculture, (179), 105834, 1-9 (2020).
4. Nagar H., Sharma, R. S.: Pest Detection on Leaf using Image Processing. 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-5, (2021).
5. Trufelea, R., Dimoiu, M., Ichim, L. Popescu, D.: Detection of Harmful Insects for Orchard Using Convolutional Neural Networks, U.P.B. Sci. Bull., Series C, Vol. 83, Iss. 4, 2021, pp 85-96 (2021).
6. Escontrela, A.: Convolutional Neural Networks from the ground up. Jun. 17, 2018. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1 (2018).
7. Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International conference on machine learning. PMLR, 2019.

8. Krizhevsky, A., Sutskever I., Hinton G. E.: ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6 (June 2017), 84–90 (2017).
9. Raj, B.: A Simple Guide to the Versions of the Inception Network. May 29, 2018. [Online]. Available: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
10. Szegedy, C. et al: Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-9, (2015)
11. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818-2826 (2016).
12. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-ResNet and the impact of residual connections on learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17). AAAI Press, pp.4278–4284 (2017).
13. Anwar, A.: Difference between AlexNet, VGGNet, ResNet, and Inception. Jun. 7, 2019. [Online]. Available: https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and- inception-7baaaecccc96 (2019).
14. Albumentations Library, [Online]. Accessed: Jan. 2020. Available: https://albumentations.ai/(2020).
15. Abadi M. et. al.: TensorFlow: Large-scale machine learning on heterogeneous systems". 2015. [Online]. Software is available from tensorflow.org. Accessed Jan. 2020.
16. Chollet, F. et. al.: (2015). Keras. GitHub. [Online]. Retrieved from https://github.com/fchol- let/keras (2020).
17. Orellana, E.: Breakdown Confusion Matrix. Oct. 15, 2020. [Online]. Available: https://emilia-orellana44.medium.com/breakdown-confusion-matrix-2cf25842f1ae (2020).