Computer Science and Information Systems 21(1):223–243 https://doi.org/10.2298/CSIS221222073M

Evaluation of Deep Learning Techniques for Plant Disease Detection*

Cedric Marco-Detchart[1], Jaime Andrés Rincon[2], Carlos Carrascosa[1], and Vicente Julian[1,3]

[1] Valencian Research Institute for Artificial Intelligence (VRAIN),
Universitat Politècnica de València (UPV),
Camino de Vera s/n, 46022, Valencia, Spain {cedmarde, carrasco, vjulian}@upv.es
[2] Departamento de Digitalización, Escuela Politécnica Superior,
Universidad de Burgos, 09006, Burgos, Spain
jarincon@ubu.es
[3] Valencian Graduate School and Research Network of Artificial Intelligence (VALGRAI),
Universitat Politècnica de València (UPV),
Camí de Vera s/n, 46022 Valencia, Spain

Abstract. In recent years, several proposals have been based on Artificial Intelligence techniques for automatically detecting the presence of pests and diseases in crops from images usually taken with a camera. By training with pictures of affected crops and healthy crops, artificial intelligence techniques learn to distinguish one from the other. Furthermore, in the long term, it is intended that the tools developed from such approaches will allow the automation and increased frequency of plant analysis, thus increasing the possibility of determining and predicting crop health and potential biotic risks. However, the great diversity of proposed solutions leads us to the need to study them, present possible situations for their improvement, such as image preprocessing, and analyse the robustness of the proposals examined against more realistic pictures than those existing in the datasets typically used. Taking all this into account, this paper embarks on a comprehensive exploration of various AI techniques leveraging leaf images for the autonomous detection of plant diseases. By fostering a deeper understanding of the strengths and limitations of these methodologies, this research contributes to the vanguard of agricultural disease detection, propelling innovation, and fostering the maturation of AI-driven solutions in this critical domain.

Keywords: Plant Disease Detection, Classification, Image Preprocessing, Deep Learning.

1. Introduction

The growing interest of the agri-food sector in the new solutions that digitalisation can provide, and the support given to their development by public administrations and different organisations, is facilitating an increasing number of technological initiatives at the international and national levels.

One factor influencing crop yields is the possible incidence of pests and diseases. To reduce their impact, farmers make recurrent use of phytosanitary products. In this way,

* This paper is an extended version of an ASSIA workshop paper.

the development of potentially dangerous populations can be controlled, thus ensuring production. This group also includes herbicides, which prevent competition for nutrients, water, and the establishment of the main crops with other unwanted plants.

In many cases, given the uncertainty of the time of pest emergence and its virulence, farmers often carry out preventive treatments. Over the last few years, the cost of these treatments has shown a clear upward trend. For example, in Spain, the average price per hectare has gone from 50 euros in 2009 to 88 euros in 2019 [4]. In other words, in just 11 years, this component has undergone an increase of more than 76%.

Within the same EU strategy to reduce the environmental impact of European agriculture, another objective has been set to reduce the use of phytosanitary products by 50% and make more rational use of these products from both an environmental and economic point of view. To achieve this, digitalisation is presented as an essential tool. In this sense, disease detection and pest evolution models can be created so that treatments are only carried out when they pose a risk to production.

According to this, plant disease is nowadays a significant concern in agriculture. There are many involved counterparts, from economic to climatic and social consequences. Early disease detection and control is one of the cornerstones of preventing economic waste and production losses. Plant disease detection is a complex task and has been traditionally done by ocular inspection and through the personal experience of farmers. One of the main concerns is to be as quick as possible to detect the disease before it spreads through the land. Furthermore, the plant is not necessarily affected by a single disease; multiple conditions can show up simultaneously, increasing the detection difficulty. Hence, the wide variety of plants combined with the different diseases nowadays makes experts fail in their tasks. With the significant increase of digital imagery, there is an opportunity to generalise the expert's knowledge and use image processing for automatic plant disease detection.

Soil analysis is traditionally done using satellite and hyperspectral images, which permits the study of wide land extensions and characteristics that humans cannot see. The counterpart of this type of analysis is that the quantity of available images is limited, and that concrete diseases can only be seen at the field level. As for hyperspectral images, the equipment needed is expensive. To make the disease detector system accessible, it must be low-cost so anyone can use it. It should work with images in the visible range, that is, in RGB space.

The current situation in the agricultural sector exacerbates the problems of lack of experts and, therefore, of time available to detect plant diseases. In this sense, the automatic detection of plant diseases plays a crucial role in overcoming these problems and providing an alternative solution to the damage caused by plant diseases.

Automatic identification of plant diseases presents several challenges ranging from problems in the capture process, such as noise or fog over the camera, to unwanted information in the images themselves, such as an inappropriate background, the soil itself, or other plants that interfere with the identification.

Deep learning-based solutions are considered the most suitable for this type of problem. Deep Learning techniques, particularly Convolutional Neural Networks, rely on weight optimisations, searching for maxima in the parameter space. In terms of statistical results, *e.g.* in edge detection, they have achieved exceptional results, even surpassing hu-

[4] https://www.mapa.gob.es/es/estadistica/temas/publicaciones/anuario-de-estadistica/

man performance [30]. As a counterpart, due to their nature, neural networks can undergo with difficulties to adapt to real scenarios when the tested data is considerably different from the training data [28]. Even so, there are still aspects to be taken into account since the proposed models are very dependent on the characteristics of the dataset used. Alternative datasets more adjusted to real situations can easily alter the accuracy of deep learning-based solutions.

This work reviews the most relevant network architectures used in the literature to detect plant disease. The experiments are conducted to classify individual diseases in plant images. Moreover, we analyse and compare their performance and investigate whether preprocessing the images before the training step impacts the correct classification. As it is difficult to say whether preprocessing has a real impact, we evaluate the different image set variants over a real image synthetic dataset, analysing the robustness of each network configuration.

The paper is structured as follows. In Section 2 we review related work in the automatic detection of plant disease. Then, Section 4 presents the use of preprocessing as the first step before training image classification networks along with the network families used. Following this, Section 5 shows the experimental framework devised. Finally, a discussion is presented in Section 5.1 along with some conclusions about the results obtained.

2. Related Work

Automatic plant disease detection is a classification problem done through the analysis of image features, which can be as various as geometric or colour features. In addition, specific indexes such as the NDVI-index, which measure the level of green on images, are commonly used (but in hyperspectral images). As a similar index type, but for visible range images, alternatives are used, such as the VARI index or the vNDVI index [7].

As indicated by Barbedo [3] and Hasan [16], there are a series of challenges when identifying plant diseases and managing crops automatically (as seen in Figure 1). One of the main problems with focussing the interest on the plant or leaf itself is the background removal, which can be soil or other plants. In addition, capture level problems can be an issue as brightness or occlusions. In terms of the disease itself, there might not be a unique disease or different diseases can have similar characteristics or might not be ideally defined.

Some of these challenges might be tackled through a pre-processing step, where an image is treated, so that spurious information is removed, *e.g.* background segmentation or texture removal (smoothing [25, 29]) or even image improvement (*e.g.* contrast enhancement [23]).

Once the images have been processed, they can be classified by two types of techniques. On the one hand, Machine Learning (ML) techniques aim at classifying plant diseases based on different features extracted from images (geometric characteristics, colour information, gradients, etc.). Different classifiers such as Support Vector Machines [6,31], K-means classifier [13], and Random Forest [20] are used to detect different plant diseases. These techniques need a very precise human-made solution (ground truth) and assistance to be performant. They work well when there is a limited amount of data. On the other hand, despite their lack of explainability, Deep Learning (DL) approaches have been

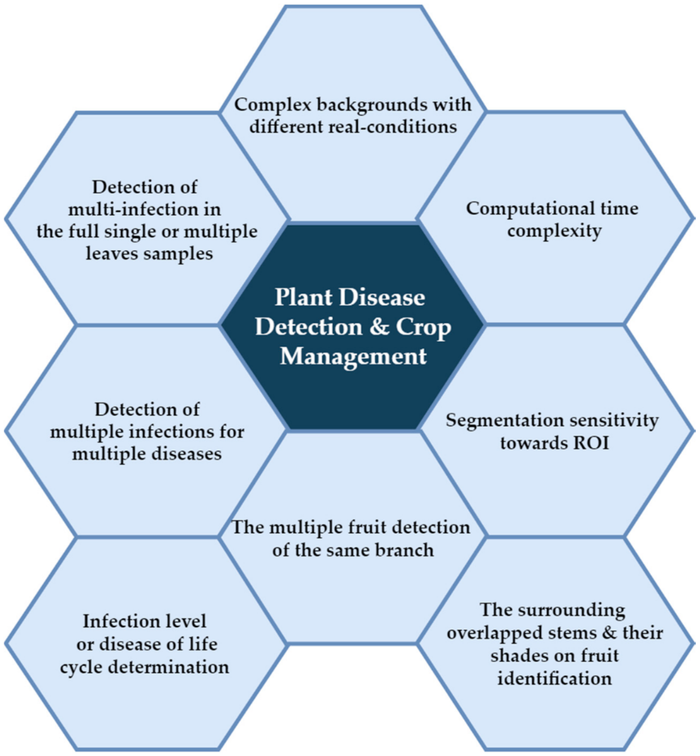

Complex backgrounds with different real-conditions
Detection of multi-infection in the full single or multiple leaves samples
Computational time complexity
Plant Disease Detection & Crop Management
Detection of multiple infections for multiple diseases
Segmentation sensitivity towards ROI
The multiple fruit detection of the same branch
Infection level or disease of life cycle determination
The surrounding overlapped stems & their shades on fruit identification

Fig. 1. Challenges in plant disease detection and crop management [16]

extensively used nowadays and report promising results. In the literature, some recurrent examples of well-known Convolutional Neural Networks (CNNs) are used to detect plant diseases. As is well known, even if there are increasingly more available images to work with, the quantity/quality is limited to learning in a specific task. In those cases, Transfer Learning is used to build a network based on pre-trained information and adapted to the concerned task. These networks are pre-trained on large datasets *e.g*. ImageNet [8]. This process takes the first layer of the trained network and removes the last layers adapting them to the specific task and training only these last steps. In this way, the specific task is not trained from scratch, and the computing time is shortened. The most common and efficient networks in the literature are *Alexnet* [21], *ResNet50* [18], *VGG16* [36], *Inception V3* [37]. *EfficientNet* [38] can also be considered a group of networks, as there are 8 types of subnets.

We can see in [14] that their model, based on *ResNet50* and pre-trained on Imagenet, is robust against field information (soil, background, etc...), obtaining an accuracy of 98.5% with synthetic soil background and 72.03% on real field images. They work with data augmentation and real field images from plants to do so. In addition, as the same disease on different plants is considered a different class, the authors propose to use a species-specific classification as the disease symptoms can be pretty similar. In this way, geometric, colour, and texture features are learnt and then incorporated into the baseline model.

Atila *et al.* propose in [2] the use of EfficientNet in its different variants (B0 to B7). The particularity of this neural network is that, as a first step, it determines the best scaling dimensions and it does not use the well-known Rectifier Linear Unit (ReLU) as an activation function, using the Swish function instead. In this work, the model is pre-trained with ImageNet and adapted by removing the Fully Conected layers with 1000 outputs to fit the 39 possible outputs of the PlantVillage dataset. The results obtained with their proposal get an accuracy beyond 99.6% compared to state-of-the-art neural nets, all pre-trained with ImageNet.

An interesting approach is that of Schwarz Schuler *et al.* [35] which propose an InceptionV3 based architecture where the image processing is separated into two branches where each image channel, in Lab colour space, is treated separately and then fused to better adapt to the inherent characteristics of each channel. The authors test their approach by varying the importance of each channel with the percentage of filters used for each of the branches. The performance of the process in all cases is greater than 99% of accuracy.

An alternative approach to mostly used network architecture is that of Capsule Networks [32] which fixes one of the significant drawbacks of a standard CNN. CNNs do not consider the possible feature hierarchy in an image considering similar images as equal even when they are not. In the work presented by Samin *et al.* [33] the Capsule Network approach is used without using Transfer Learning, obtaining an accuracy of 93.07%.

More recently, a lightweight CNN approach was based on Inception and Residual connection [17]. The proposed approach extracts better features from the input images. This is done by replacing the standard convolution by a depth-wise separable convolution combined with a point-wise convolution, which results in fewer parameters as well as a speed-up in the processing. The resulting performance with the approach presented is 99.39% accuracy.

After studying the state-of-the-art, it can be seen that there are currently a multitude of proposals, most of them based on deep learning techniques that offer promising results from the existing datasets. However, there are specific gaps that we think should be analysed. On the one hand, some works suggest the need for image pre-processing before classification; in our opinion, this aspect should be studied in greater detail as it may allow for an improvement in the classification process. On the other hand, most of the works are evaluated against a so-called ideal dataset. Using more realistic datasets to validate existing models would allow for analysis of their possible robustness. Nevertheless, [5] is an interesting approach, but as they are working with infrared images, they would need to use infrared cameras when applied to the real world, which are expensive to deploy.

In summary, the reviewed works exhibit several shortcomings. First, there is a common challenge in background removal, which includes soil or other plants. This issue can be addressed through pre-processing steps, such as background segmentation and texture removal. Additionally, the variability in disease symptoms and the lack of a unique or well-defined disease pose challenges. To tackle this, the proposed work employs a species-specific classification approach, which considers the similarity of symptoms among diseases within a species. Moreover, most of the existing works rely on ideal datasets for evaluation, which may not reflect real-world conditions. To address this, the proposed work emphasizes the need for more realistic datasets to validate models and analyze their robustness. Additionally, some works suggest image pre-processing before classification, and the proposed work aims to delve deeper into this aspect to potentially improve the classification process. Finally, while the reviewed works primarily focus on deep learning techniques, the proposed work offers a comprehensive comparative study of various approaches, including deep learning and alternative network architectures. This comparative analysis helps identify the strengths and weaknesses of different methods and contributes to a better understanding of plant disease detection techniques. In the following section, the proposed study is presented.

3. Contextualizing Plant Disease Detection

This section provides a comprehensive overview of the problem statement, research objectives, and motivation driving our study on evaluating a collection of different neural network architectures for plant disease detection.

3.1. Problem Statement

The accurate classification of plant diseases is a matter of utmost importance in contemporary agriculture. With the global population rising and increasing pressure on sustainable food production, the need for precise disease detection has never been greater.

Early and precise disease identification facilitates timely intervention, curbing the spread of infections and minimising crop losses. This ensures a stable food supply to meet the demands of growing populations. Accurate classification ensures that infected productions are promptly identified and removed, safeguarding consumer health and maintaining the reputation of agricultural products. Reducing chemical pesticide use and optimising resource allocation in farming can be achieved through targeted disease management, contributing significantly to sustainable agricultural practices.

However, several challenges and implications loom over the issue of plant disease classification. Misclassification can lead to unnecessary treatments or neglect of infected plants, resulting in economic losses for farmers. In addition, it can lead to environmental pollution and harm to non-target species and compromise food security by reducing crop yields and impacting the availability and affordability of agricultural products. Accurate datasets for plant disease classification are essential for advancing research in crop protection and breeding resilient plant varieties.

3.2. Objectives and motivation

Our proposal investigates various deep learning models' effectiveness in the automated detection and classification of plant diseases, rigorously evaluating their performance. Based on the performance measures obtained, we determine which neural network architectures and transfer learning strategies yield the best results for plant disease detection. With the proposed experiments, we bridge the gap between research and practical application by developing a user-friendly tool that can assist farmers in identifying plant diseases quickly and accurately.

Our motivation to embark on this study is grounded in the pressing need to address the challenges posed by plant diseases in agriculture. We aspire to provide a practical and efficient solution to the longstanding problem of plant disease detection, making a tangible impact on the agricultural industry. Ensuring a stable and secure food supply for current and future generations is a driving force behind our work, guarding crop yields against the persistent threat of plant diseases, contributing to reducing environmental impact due to excessive pesticide use and promoting responsible farming practices. By exploring novel approaches for automated disease detection in plants, we seek to advance the fields of computer vision and machine learning, pushing the boundaries of technological innovation.

4. Preprocessing and Classification models

In this Section, we briefly present the preprocessing step that we use as a primary task to train the networks and build specific sets of images with regularisation and sharpening. We also present the different network architectures we put to the test and the different parameters we configure.

4.1. Preprocessing

As the first task in this work, we consider the preprocessing step inspired by the Bezdek Breakdown Structure (BBS) [4] for edge detection. Four steps characterise the process: image conditioning, feature extraction, blending and scaling.

This structure helps to understand edge detection as four independent phases. For our purpose of interpreting plant leaf classification using Convolutional Neural Networks (CNN), in some way, we take the first step of the BBS as the input of our neural system. We do not consider the rest of the steps as they are edge-detection-specific.

Image conditioning focuses on removing noise, or additional information that is not needed for the process it is undergoing (in this case, feature extraction in the different layers of the network).

It is a cornerstone task as it removes spurious information on the image due to the capture process or reduces texture information. In our approach, we propose to use The Gravitational Smoothing (GS) process [25] as a conditioning step in the preprocessing. This preprocessing step is done to all the images of the dataset in order to obtain a variation that will be used for training purposes individually without mixing the new images with the original ones.

GS is a content-aware smoothing method that adapts the image regularisation based on local information, as opposed to Gaussian smoothing, whose main problem is the blurring of objects in the image. GS simulates the movement of pixels in a 5D spatial-tonal space, bringing closer or moving away similar/dissimilar pixels. This behaviour allows us to control the blurring effect by removing unwanted information such as textures while preserving abrupt tonal changes (*e.g.* significant colour variation). In this technique, each pixel is considered a particle in the space that exerts an attractive or repulsive force on every surrounding pixel. Then the total force over a pixel is the sum of all individual forces around a pixel. Each force is computed as the product of a gravitational constant G and the inverse of the distance in the spatial-tonal space. Additionally, this force model can be used oppositely, changing the direction of the forces. Using GS in this way enhances the image's characteristics, highlighting the differences, which in our case can be beneficial to detect the diseased regions better. An example of the resulting images obtained can be seen in Figures 2(a)-2(b), where we apply the smoothing and sharpening version of GS to an original leaf image.

4.2. Deep Neuronal Network Description

Different network architectures were evaluated to perform this classification of plant diseases to determine which configuration allowed the best results to be obtained. Different aspects were taken into account, such as transfer learning, data augmentation and the end device: server or edge device. In order to obtain the different networks, a series of hyperparameters were modified to each of the networks. In order to determine which configuration allowed the best results to be obtained.

The hyperparameters presented below were the result of an extensive iterative process. These specific hyperparameters were identified as the ones yielding the most favorable outcomes during the validation phase. Notably, the decision to set the number of training epochs at 7 emerged from a careful consideration of model performance. Extensive experimentation revealed that increasing the number of epochs led to a phenomenon known as overtraining, where the model became overly specialized to the training data, compromising its generalization capability. Conversely, reducing the number of epochs resulted in markedly suboptimal results, indicating the need for a moderate yet effective training duration. This strategic choice of 7 epochs reflects a delicate balance, ensuring that the model captures underlying patterns without succumbing to overfitting, thus optimizing its predictive capacity.

– **Epochs**: 7
– **Network**: InceptionV3, MobilenetV2, NASNetMobile, Efficientnet-B0, EfficientnetV2-Imagenet1k, EfficientNet-Imagenet21k
– **Learning rate**: 0.001
– **Transfer learning** (**Tl**): Yes or No
– **Data augmentation** (**Da**): Yes or No
– **Dataset**: Raw, Mixed, Smoothed, Sharpened

Each of the different networks used has several characteristics that make them stand out from each other. Out of these characteristics we can highlight the ability to be used in systems with ARM architecture. Specifically, if it is possible to run the model efficiently on Edge devices, TPU (Tensor Processing Unit), CPU or GPU. The characteristics of each of the networks used are described below.

Inceptionv3 [37], [26] is an image recognition architecture that has been shown to achieve an accuracy of better than 78.1% on the ImageNet dataset. The Inceptionv3 model comprises symmetric and asymmetric building blocks, including convolutions, mean reduction, maximum clustering, concatenations, dropouts and fully connected layers. Batch normalisation is a widely used technique in Inception v3 and is applied to activation inputs. Inception v3 uses the Softmax function to perform the loss function calculation, which is ideal for non-binary classifications.

The **MobileNetV2** [34] network is an evolution of its predecessor MobileNetV1. These networks belong to a family of neural networks widely used in computer vision. They are general-purpose networks designed especially for use on mobile devices and work very well on devices such as a Raspberry Pi [12]. MobileNetV2 is a significant improvement over MobileNetV1 and presents a breakthrough in visual recognition on low-power devices. It enables devices to perform classification, object detection and semantic segmentation. In the NASNetMobile [27] network, the authors propose to search for an architectural block using a small dataset, in order to then be transferred to a larger dataset. In the NASNet network, they first seek to obtain the best convolutional layer using the CIFAR-10 dataset. Once this layer is obtained, it is applied to the ImageNet dataset, stacking more copies of the obtained convolutional layer. At the same time, the authors of the NASNet network propose a new regularisation technique called ScheduledDropPath, which significantly improves generalisation in NASNet models. One of the advantages of the NASNet network is the reduction of the generated model size.

EfficientNet [38] is a convolutional neural network architecture and scaling method that uniformly scales all depth/width/resolution dimensions using a composite coefficient. Unlike conventional practice that arbitrarily scales these factors, the EfficientNet scaling method uniformly scales the network's width, depth and resolution with a set of fixed scaling coefficients.

EfficientNet-B0 [38] is one of the configurations of the **EfficientNet** family that aims to scale ConvNets, maintaining and even surpassing their efficiency. Scaling is uniformly performed by a given coefficient unlike other methods where it is done arbitrarily. The variant B0 is the basic one and is based on the bottleneck residual blocks from MobileNetV2. **EfficientnetV2-B0** [39] is an evolution of the previous family of neural nets that improves in terms of training times and parameter efficiency. In addition, the authors perform a reduction of a network that is almost 6.8 times smaller. In our experiments, we use two variants of EfficientNetV2-B0, one pre-trained over a subset of Imagenet (EfficientnetV2-Imagenet1k) and the other over the complete Imagenet (EfficientNet-Imagenet21k).

4.3. Discussion

The use of different types of neural network models makes it possible to determine which type of network has the best classification rate. However, the use of convolutional neural networks (CNNs) has been widely used in the classification of plant diseases from RGB images [22]. However, these models do not necessarily focus on the visible parts affected by a plant disease for classification, and can sometimes take into account irrelevant backgrounds or healthy parts of the plant.

However, each of the models you mentioned, InceptionV3, MobileNetV2, EfficientNet and EfficientNetV2-B0, has unique architectural features that may make them suitable for plant disease classification. Here is a brief discussion of each:

1. **InceptionV3:** This model is known for its inception modules, which allow for efficient computation and deep networks through a carefully crafted design1. The inception modules help the network learn useful representations at multiple scales, which can be beneficial for plant disease classification where symptoms can vary in size [10].
2. **MobileNetV2:** This model introduces two new features to the architecture: Linear Bottlenecks and Inverted Residuals [11] [24]. The use of linear bottlenecks is to maintain the representational power. The inverted residuals allow for a reduction in

computational complexity. This makes MobileNetV2 a lightweight model that can be used on devices with limited computational resources.

3. **EfficientNet:** EfficientNet uses a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient. This could potentially capture more complex features in the images, making it suitable for plant disease classification [1].
4. **EfficientNetV2-B0:** This model improves upon EfficientNet by introducing a form of progressive learning mechanism which expands the network topology gradually over the course of training process improving the model's learning capacity. This could be particularly useful in plant disease classification where the dataset might be imbalanced [9].

The choice of a model may depend on several aspects, such as the size and quality of the dataset, the available computing resources and the specific needs of the task. For example, MobileNetV2 may be the best choice if you are working with limited computational resources due to its remarkable efficiency. On the other hand, if you are dealing with a large and intricate dataset, EfficientNet or EfficientNetV2-B0 might be more suitable options given their ability to handle complex features. The choice of a model may depend on several aspects, such as the size and quality of the dataset, the available computing resources and the specific needs of the task. For example, MobileNetV2 may be the best choice when working with limited computing resources due to its remarkable efficiency. On the other hand, when datasets are large and intricate, EfficientNet or EfficientNetV2-B0 may be more suitable options given their ability to handle complex features. On the other hand, it is important to generate instances that reflect real-world scenarios, all in order to improve the performance of the machine learning model. However, it is crucial to understand that not all data augmentation techniques are a one-size-fits-all solution for all datasets. In the experiments conducted, image transformations and rotations were applied, a technique that is usually reserved for image datasets.

5. Experimental Framework

In this Section, we compare the most usual proposals found in the literature for plant disease detection. In Section 5.1, we briefly introduce the dataset used for our experiments as well as the performance measures we used to quantify the results. In Section 5.2, we show the quantitative results obtained in the comparison along with its analysis.

5.1. Dataset and Quantification of the Results

For our experiments, we have put to the test our proposal with the PlanVillage dataset [19] which contains 54303 images of healthy and diseased leaf plants classified by species and disease, having a total of 38 classes.

In addition, the data set has three different types of images available. The main raw image is an RGB colour space image taken on a uniform background. In addition, there is a grey-scale version of the previous image. And finally, a segmented version, where the background has been removed and only contains the leaf information.

From the original dataset, we prepare four different derivatives:

- **RAW**: It consists of images of the original dataset with colour and background.
- **Mixed**: Contains a mixture of RAW images, rotated and with background removed.
- **Smoo**: This dataset contains images of RAW where the background has been removed, and the leaf has been smoothed with the gravitational algorithm.
- **Sharp**: The images in this dataset are similar to those in Smoo but inverting the sense of the force over each pixel and sharpening the tonal variations.

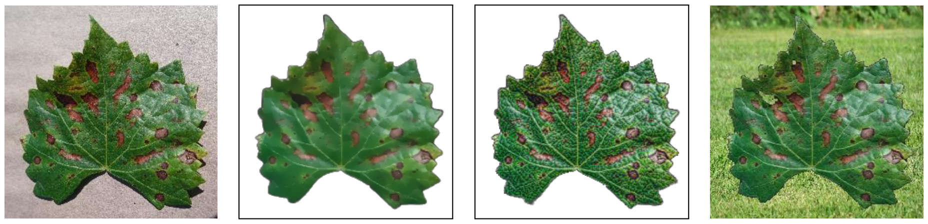


Fig. 2. Example images in PlantVillage dataset [19] from class *Grape Esca (Black Measles)* with the different preprocessings and from the PlantVillage synthetic dataset. (a) shows the original raw image, (b) is the preprocessed image applying smoothing, (c) is the preprocessed image using sharpening and (d) is the generated image adding grass background

The dataset and its derivatives were divided into three subsets with an 80/20 ratio, 80% for training, 10% for testing and the remaining 10% for validation.

Furthermore, to verify the robustness of each of the tested configurations, we trained them with a modified version of the PlantVillage dataset, where the uniform background was removed and replaced with a field image. In this way, more realistic images are simulated. This modified dataset introduced in [15] is called synthetic PlantVillage dataset (Synth-PV).

To interpret the results obtained in the confusion matrices, where True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) are extracted, we quantify the results using the following well-known Precision ($Prec$), Recall (Rec) and F_β measures:

$$Prec = \frac{TP}{TP + FP},\ Rec = \frac{TP}{TP + FN},\ F_\beta = (1 + \beta^2)\frac{Prec \cdot Rec}{\beta^2 \cdot Prec + Rec}.$$

We select the values of $\beta = 0.5$ and $\beta = 1$ as the most commonly used in the literature. In particular, The F_1 score balances precision and recall, making it valuable when both measures are important. Taking the harmonic mean of precision and recall provides a more comprehensive assessment of a model's performance, especially in tasks where false positives and false negatives have distinct consequences, such as medical diagnosis or plant disease detection. In addition, we use Accuracy and Loss metrics for the training and validation phases.

Combining all the six selected networks, we are to analyse along with the possible parameters that we configure (transfer learning and data augmentation) with two possibilities each one and the four different sets generated from the original PlantVillage dataset, we have 96 different configurations. To facilitate analysis and visualisation of the results, we separate the 96 experiments into two blocks separated by whether they are mobile-orientated network architectures (Inception v3, MobileNetV2, NASNetMobile), shown in Table 1, or high-performance-computer-orientated (EfficientNet-based), presented in Table 2.

5.2. Discussion

In this subsection, we expose and analyse the results obtained with the training and testing with the different network models over the original PlantVillage dataset with the proposed preprocessing step in the first phase. We then show the test results obtained with the previously trained models in the synthetic PlantVillage dataset.

We show the different results of the train, validation, and test in Tables 1-2. We can see that the accuracy obtained during the training phase is very similar and relatively high for all the experiments, going from 0.850 to 0.993. This great value decays as validation is performed. In this case, we obtain values between 0.578 and 0.992. The results indicate that all those networks prepared for mobile devices get lower results, which is the expected behaviour, as they are optimised for limited-power devices. Despite their optimized architecture designed for limited devices, these models perform quite well. For instance, NasNetMobile achieves an accuracy validation score of 0.968, which is not too far behind the top performer, EfficientNetV2, with a validation accuracy of 0.992. As a general overview, we can extract from the training results, on the one hand, that fine-tuning (Ft) and data augmentation are not beneficial if we look at the training accuracy but permits us to retain it in the validation step for the mobile-oriented Nets. On the other hand, when viewing EfficientNet experiments, we can see that data augmentation does not benefit the training and validation accuracy, as the best results are obtained only by fine-tuning. During the training and validation phase, the best performers are E_{63} and E_{64}, that is, EfficientNet-B0 and EfficientNetV2-B0 using fine-tuning but not data augmentation.

If we observe more locally at each network type, for Inception v3, the best performance in terms of validation accuracy is E_2, using the Raw data set and without fine-tuning or data augmentation. The same behaviour happens with MobileNetV2, but in the case of NASNetMobile, the best performer in this terms is also obtained in the Raw dataset but using fine-tuning and data augmentation. Regarding EfficientNet results, the best performer configuration using EfficientNet-B0 uses fine-tuning and data augmentation over the Raw dataset, just the opposite to mobile-orientated designs. When observing both EfficientNetV2-B0 with transfer learning from a subset of Imagenet and the complete one, we can see that we have to use fine-tuning but not data augmentation to obtain the best result.

Once the models are trained, we evaluate them during the test phase obtaining the quantitative results in Tables 3-4 with the measures indicated in Section 5.1. We tested our trained models with the corresponding test partition images of each model, and to analyse its robustness to real field images, we also tested each model with the Synthetic-PV dataset. As can be seen, the best performers with the original dataset usually decay with the synthetic one. For example, in the case of experiments with mobile-oriented nets ($E_1 - E_{48}$), the best performer in terms of original images is E_{21}, but its performance decays by approximately 76%, and the best one in terms of real synthetic images is E_{17}

maintaining its performance with a decay 24 %. This tells us it is more robust to interference from the leaf context. The same behaviour occurs in the case of EfficientNet-based experiments. The best performer is not robust and loses 76% of its performance. Instead, the best performance in the Synthetic dataset has a decay of 25 %.

In a more detailed analysis of the test results, looking at ∇F_1 by network type, we can see that with Inception v3 and MobileNetV2, excluding E_{14} due to its low results in both the original and synthetic data sets, the most stable configurations are E_2 and $E_1 8$, which match the results of the training phase. In the case of NASNetMobile, the stables experiment is E_{42}, which is not among the best performers in the training phase. Additionally, the best performer using NASNetMobile during training decays 46%.

On the other side, as for EfficientNet-B0 experiments, the more robust configuration is E_{75}, which has been trained over the Mixed dataset. However, close to the most stable experiment, we find those who have been trained with Smoo and Sharp preprocessing, such as E_{81} or E_{60}, respectively. In the case of EfficientNetV2-B0 with transfer learning with the Imagenet subset, we obtain slightly less stable results, being the best E_{61}, that is, using the Mixed dataset. With this network family, all other configurations have more than 50% performance variation. Finally, looking at the EfficientNetV2-B0 with the complete Imagenet transfer learning, the most stable configuration is E_{74}, trained with the Mixed dataset. Then, With a nearly a 30% decay in the F_1 measure, we have configurations trained with Smoo preprocessing (E_{56} and E_{80}).

As a result of the database validation process, a series of metrics were extracted for each of the experiments conducted to determine which of the experiments and which network architecture is the best for classifying plant diseases. Figure 3 shows a plot of the Nightingale rose, which is a plot on a polar coordinate grid. This radial graph divided each of the 48 experiments into equal segments. The distance of each segment from the centre of the polar axis depends on the value of each score it represents. In this way, each of the rings from the centre of the polar grid represents a higher value of each score. Four colours will be used to differentiate each of the sockets in the following graphs, the blue colour represents the F_1 score of the models, the yellow colour represents the $F_0.5$, the green colour represents the $Recall$, and the red colour represents the $Precision$.

Figure 3 shows the scores obtained from the 96 experiments with the different networks. Figure 3 (a) shows the scores for the networks launched on Edge devices: MobileNet, Inception v3 and NASNetMobile. Figure 3 (b) shows the scores obtained for EfficientNet networks, which can be executed on devices with higher-performance computers. As shown in Figure 3 (a), the scores obtained are very unstable compared to those in Figure 3 (b).

Although both networks used the same dataset, the internal architectures of each network make a difference.

To sum up, in this work we have analysed different network architectures and configurations for the automatic detection of plant disease. We have also tested our trained models, evaluating their robustness and behaviour with different image alterations and synthetic images that simulate a real environment.

The results obtained have shown in a primal analysis that the best performers during the training phase tend to have a significant decay in the test phase on real-context simulated images. Therefore, models with near-the-top best performers but slightly more restrained results are more stable when used in real scenarios. We have also noticed that

#	Network	Ft	Da	Data	Accuracy T	Accuracy V	Loss T	Loss V
E_1	InceptionV3	✗	✗	Mixed	.967	.858	.122	.500
E_2	InceptionV3	✗	✗	Raw	.940	.924	.212	.279
E_3	InceptionV3	✗	✗	Sharp	.917	.905	.277	.325
E_4	InceptionV3	✗	✗	Smoo	.939	.915	.213	.297
E_5	InceptionV3	✓	✗	Mixed	.975	.825	.285	.962
E_6	InceptionV3	✓	✗	Raw	.979	.578	.300	.909
E_7	InceptionV3	✓	✗	Sharp	.973	.844	.327	.889
E_8	InceptionV3	✓	✗	Smoo	.971	.902	.323	.569
E_9	InceptionV3	✗	✓	Mixed	.922	.811	.261	.630
E_{10}	InceptionV3	✗	✓	Raw	.867	.884	.435	.389
E_{11}	InceptionV3	✗	✓	Sharp	.850	.872	.494	.418
E_{12}	InceptionV3	✗	✓	Smoo	.876	.912	.414	.300
E_{13}	InceptionV3	✓	✓	Mixed	.963	.790	.328	.247
E_{14}	InceptionV3	✓	✓	Raw	.968	.735	.341	.368
E_{15}	InceptionV3	✓	✓	Sharp	.957	.679	.383	.951
E_{16}	InceptionV3	✓	✓	Smoo	.955	.909	.383	.549
E_{17}	MobileNetV2	✗	✗	Mixed	**.988**	.908	**.062**	.315
E_{18}	MobileNetV2	✗	✗	Raw	.980	.966	.095	.138
E_{19}	MobileNetV2	✗	✗	Sharp	.966	.941	.136	.220
E_{20}	MobileNetV2	✗	✗	Smoo	.974	.953	.109	.176
E_{21}	MobileNetV2	✓	✗	Mixed	.980	.878	.178	.533
E_{22}	MobileNetV2	✓	✗	Raw	.979	.940	.196	.360
E_{23}	MobileNetV2	✓	✗	Sharp	.974	.958	.210	.279
E_{24}	MobileNetV2	✓	✗	Smoo	.976	.953	.207	.326
E_{25}	MobileNetV2	✗	✓	Mixed	.961	.891	.144	.354
E_{26}	MobileNetV2	✗	✓	Raw	.938	.939	.220	.221
E_{27}	MobileNetV2	✗	✓	Sharp	.915	.926	.291	.271
E_{28}	MobileNetV2	✗	✓	Smoo	.930	.942	.242	.206
E_{29}	MobileNetV2	✓	✓	Mixed	.969	.853	.217	.655
E_{30}	MobileNetV2	✓	✓	Raw	.972	.959	.229	.257
E_{31}	MobileNetV2	✓	✓	Sharp	.964	.797	.247	**.057**
E_{32}	MobileNetV2	✓	✓	Smoo	.965	.933	.252	.377
E_{33}	NasNetMobile	✗	✗	Mixed	.972	.871	.122	.451
E_{34}	NasNetMobile	✗	✗	Raw	.948	.930	.203	.260
E_{35}	NasNetMobile	✗	✗	Sharp	.928	.915	.264	.309
E_{36}	NasNetMobile	✗	✗	Smoo	.949	.939	.206	.241
E_{37}	NasNetMobile	✓	✗	Mixed	.984	.903	.323	.625
E_{38}	NasNetMobile	✓	✗	Raw	.985	.943	.336	.503
E_{39}	NasNetMobile	✓	✗	Sharp	.980	.957	.360	.430
E_{40}	NasNetMobile	✓	✗	Smoo	.983	.952	.350	.477
E_{41}	NasNetMobile	✗	✓	Mixed	.931	.815	.240	.621
E_{42}	NasNetMobile	✗	✓	Raw	.898	.880	.363	.421
E_{43}	NasNetMobile	✗	✓	Sharp	.874	.893	.429	.385
E_{44}	NasNetMobile	✗	✓	Smoo	.900	.925	.357	.284
E_{45}	NasNetMobile	✓	✓	Mixed	.978	.870	.351	.729
E_{46}	NasNetMobile	✓	✓	Raw	.979	**.968**	.364	.396
E_{47}	NasNetMobile	✓	✓	Sharp	.971	.935	.389	.503
E_{48}	NasNetMobile	✓	✓	Smoo	.971	.954	.391	.446

Table 1. PlantVillage experiments with 7 Epochs, feature vector model using InceptionV3, MobileNetV2 and NasNetMobile networks. Fine-tuning (Ft) and Data augmentation (Da) use are indicated with a cross (✗) and a tick (✓), differentiating the train (T) and validation (V) values. The maximum values for Accuracy and minimum values for Loss are shown in bold

#	Network	Ft	Da	Data	Accuracy T	Accuracy V	Loss T	Loss V
E_{49}	EfficientnetV2-B0-imagenet1k	✗	✗	Mixed	.989	.926	.065	.254
E_{50}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✗	Mixed	.991	.947	**.045**	.186
E_{51}	Efficientnet-B0	✗	✗	Mixed	**.993**	.946	.052	.193
E_{52}	EfficientnetV2-B0-imagenet1k	✗	✗	Raw	.984	.976	.087	.114
E_{53}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✗	Raw	.987	.980	.064	.091
E_{54}	Efficientnet-B0	✗	✗	Raw	.990	.983	.071	.091
E_{55}	EfficientnetV2-B0-imagenet1k	✗	✗	Smoo	.975	.967	.124	.146
E_{56}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✗	Smoo	.983	.976	.079	.103
E_{57}	Efficientnet-B0	✗	✗	Smoo	.980	.978	.102	.119
E_{58}	EfficientnetV2-B0-imagenet1k	✗	✗	Sharp	.973	.958	.127	.174
E_{59}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✗	Sharp	.978	.971	.097	.127
E_{60}	Efficientnet-B0	✗	✗	Sharp	.977	.966	.115	.138
E_{61}	EfficientnetV2-B0-imagenet1k	✓	✗	Mixed	.990	.946	.048	.197
E_{62}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✗	Mixed	.989	.958	.053	.154
E_{63}	Efficientnet-B0	✓	✗	Mixed	**.993**	.960	.156	.286
E_{64}	EfficientnetV2-B0-imagenet1k	✓	✗	Raw	.990	**.992**	.047	**.042**
E_{65}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✗	Raw	.989	.991	.052	.043
E_{66}	Efficientnet-B0	✓	✗	Raw	.989	.986	.173	.180
E_{67}	EfficientnetV2-B0-imagenet1k	✓	✗	Smoo	.988	.982	.058	.078
E_{68}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✗	Smoo	.987	.981	.057	.081
E_{69}	Efficientnet-B0	✓	✗	Smoo	.986	.976	.183	.221
E_{70}	EfficientnetV2-B0-imagenet1k	✓	✗	Sharp	.990	.981	.049	.090
E_{71}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✗	Sharp	.987	.969	.058	.141
E_{72}	Efficientnet-B0	✓	✗	Sharp	.988	.975	.182	.216
E_{73}	EfficientnetV2-B0-imagenet1k	✗	✓	Mixed	.970	.920	.122	.264
E_{74}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✓	Mixed	.978	.923	.090	.267
E_{75}	Efficientnet-B0	✗	✓	Mixed	.975	.915	.106	.298
E_{76}	EfficientnetV2-B0-imagenet1k	✗	✓	Raw	.962	.961	.160	.166
E_{77}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✓	Raw	.969	.964	.123	.138
E_{78}	Efficientnet-B0	✗	✓	Raw	.969	.963	.134	.148
E_{79}	EfficientnetV2-B0-imagenet1k	✗	✓	Smoo	.943	.949	.218	.198
E_{80}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✓	Smoo	.958	.965	.155	.136
E_{81}	Efficientnet-B0	✗	✓	Smoo	.957	.965	.176	.150
E_{82}	EfficientnetV2-B0-imagenet1k	✗	✓	Sharp	.942	.939	.222	.229
E_{83}	EfficientnetV2-B0-imagenet21k-ft1k	✗	✓	Sharp	.950	.942	.176	.202
E_{84}	Efficientnet-B0	✗	✓	Sharp	.947	.954	.208	.174
E_{85}	EfficientnetV2-B0-imagenet1k	✓	✓	Mixed	.984	.938	.069	.220
E_{86}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✓	Mixed	.982	.921	.078	.286
E_{87}	Efficientnet-B0	✓	✓	Mixed	.981	.959	.198	.286
E_{88}	EfficientnetV2-B0-imagenet1k	✓	✓	Raw	.986	.981	.061	.079
E_{89}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✓	Raw	.984	.979	.068	.096
E_{90}	Efficientnet-B0	✓	✓	Raw	.984	.988	.192	.180
E_{91}	EfficientnetV2-B0-imagenet1k	✓	✓	Smoo	.980	.980	.081	.081
E_{92}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✓	Smoo	.979	.981	.085	.087
E_{93}	Efficientnet-B0	✓	✓	Smoo	.979	.978	.213	.216
E_{94}	EfficientnetV2-B0-imagenet1k	✓	✓	Sharp	.978	.952	.086	.166
E_{95}	EfficientnetV2-B0-imagenet21k-ft1k	✓	✓	Sharp	.977	.940	.092	.206
E_{96}	Efficientnet-B0	✓	✓	Sharp	.978	.957	.215	.297

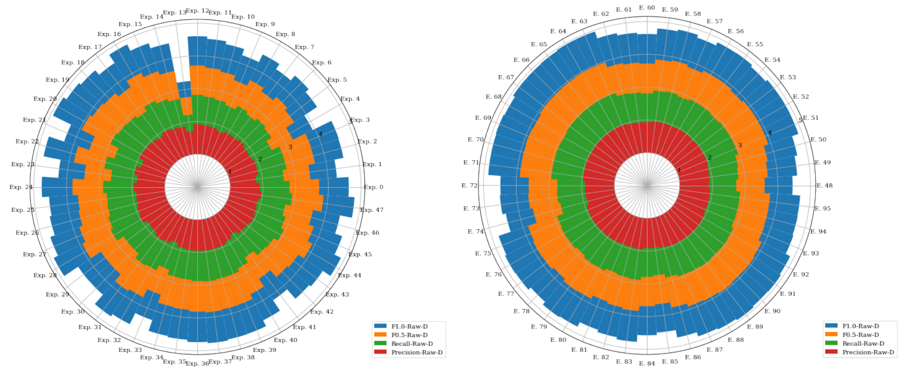
Table 2. PlantVillage experiments with 7 Epochs, feature vector model using EfficientNet-B0, EfficientNetV2-B0 pre-trained with imagenet-ilsvrc-2012-cls and EfficientNetV2-B0 pre-trained with full ImageNet and fine-tuned with imagenet1k networks. Fine-tuning (Ft) and Data augmentation (Da) use are indicated with a cross (✗) and a tick (✓), differentiating the train (T) and validation (V) values. The maximum values for Accuracy and minimum values for Loss are shown in bold

#	$Prec.$	$Prec.'$	$\nabla Prec.$	$Rec.$	$Rec.'$	$\nabla Rec.$	$F_{0.5}$	$F'_{0.5}$	$\nabla F_{0.5}$	F_1	F'_1	∇F_1
E_1	.901	.643	.258	.917	.538	.379	.902	.552	.350	.905	.519	.386
E_2	.807	.733	.074	.843	**.691**	.152	.806	**.699**	**.107**	.812	**.681**	.131
E_3	.893	.652	.241	.884	.479	.405	.888	.516	.372	.884	.461	.423
E_4	.907	.612	.295	.879	.496	.383	.893	.520	.373	.883	.476	.407
E_5	.849	.579	.270	.681	.341	.340	.724	.356	.368	.673	.304	.369
E_6	.855	.502	.353	.860	.296	.564	.843	.285	.558	.838	.245	.593
E_7	.911	.493	.418	.857	.268	.589	.887	.281	.606	.866	.237	.629
E_8	.862	.527	.334	.817	.291	.526	.833	.294	.538	.812	.247	.565
E_9	.878	.645	.232	.881	.495	.386	.876	.525	.351	.875	.475	.400
E_{10}	.813	.694	.119	.827	.568	.259	.805	.586	.219	.802	.549	.253
E_{11}	.878	.654	.223	.844	.495	.349	.866	.541	.324	.854	.484	.370
E_{12}	.890	.647	.243	.876	.505	.371	.884	.529	.355	.879	.486	.393
E_{13}	.915	.314	.601	.894	.209	.685	.901	.175	.726	.890	.164	.726
E_{14}	.709	.637	**.071**	.504	.399	**.104**	.547	.426	.121	.479	.371	**.107**
E_{15}	.883	.549	.333	.862	.215	.647	.866	.264	.602	.856	.192	.663
E_{16}	.915	.318	.597	.915	.181	.734	.902	.122	.780	.899	.109	.790
E_{17}	.958	.709	.249	.968	.574	.394	.960	.592	.368	.963	.552	.410
E_{18}	.881	**.744**	.137	.904	.665	.239	.882	.669	.212	.886	.644	.242
E_{19}	.933	.692	.241	.939	.475	.463	.932	.489	.443	.932	.442	.490
E_{20}	.957	.612	.345	.947	.496	.450	.954	.490	.463	.951	.457	.493
E_{21}	**.980**	.429	.550	**.982**	.255	.727	**.980**	.264	.716	**.980**	.221	.759
E_{22}	.829	.616	.212	.852	.465	.386	.813	.456	.356	.807	.423	.384
E_{23}	.962	.428	.534	.934	.305	.629	.952	.257	.695	.942	.235	.707
E_{24}	.851	.338	.512	.802	.147	.655	.814	.143	.670	.792	.117	.675
E_{25}	.940	.696	.244	.939	.503	.435	.939	.567	.372	.938	.507	.430
E_{26}	.881	.719	.162	.897	.650	.247	.881	.643	.238	.883	.623	.260
E_{27}	.904	.641	.263	.899	.406	.493	.900	.486	.414	.897	.415	.482
E_{28}	.940	.611	.328	.935	.510	.425	.937	.494	.443	.935	.468	.467
E_{29}	.974	.481	.493	.974	.323	.651	.973	.266	.707	.973	.249	.724
E_{30}	.852	.658	.193	.862	.375	.487	.836	.419	.417	.831	.358	.473
E_{31}	.868	.422	.446	.846	.222	.624	.839	.214	.625	.823	.187	.635
E_{32}	.952	.196	.756	.926	.124	.802	.941	.105	.836	.932	.087	.845
E_{33}	.915	.570	.345	.922	.451	.471	.915	.446	.469	.917	.410	.507
E_{34}	.813	.630	.182	.851	.553	.297	.812	.558	.254	.816	.531	.284
E_{35}	.912	.540	.372	.909	.395	.514	.909	.363	.546	.907	.341	.566
E_{36}	.931	.578	.353	.905	.398	.507	.922	.369	.553	.913	.339	.574
E_{37}	.946	.671	.274	.927	.404	.523	.936	.440	.496	.927	.377	.550
E_{38}	.951	.632	.318	.943	.343	.599	.944	.368	.576	.940	.307	.633
E_{39}	.951	.430	.520	.922	.243	.679	.941	.241	.700	.930	.210	.720
E_{40}	.944	.138	.805	.928	.098	.830	.936	.059	.877	.929	.051	.878
E_{41}	.882	.542	.339	.891	.391	.500	.882	.374	.508	.884	.336	.548
E_{42}	.787	.632	.155	.810	.535	.275	.780	.527	.253	.779	.504	.275
E_{43}	.892	.538	.354	.880	.416	.464	.886	.397	.489	.881	.376	.505
E_{44}	.914	.590	.324	.891	.451	.440	.906	.428	.478	.896	.403	.493
E_{45}	.968	.417	.550	.978	.290	.688	.969	.289	.679	.972	.255	.717
E_{46}	.916	.672	.244	.927	.490	.437	.913	.492	.421	.914	.449	.465
E_{47}	.886	.445	.441	.853	.213	.640	.861	.228	.633	.841	.188	.653
E_{48}	.965	.541	.423	.938	.244	.694	.958	.256	.702	.949	.221	.728

Table 3. Resulting test performance of the configurations trained with the parameters in Table 1 over the original data set and its pre-processed variants along with those obtained with the Synth-PV dataset and its difference. The maximum values for $Prec.$, Rec and F_β values and minimum values for ∇ are shown in bold

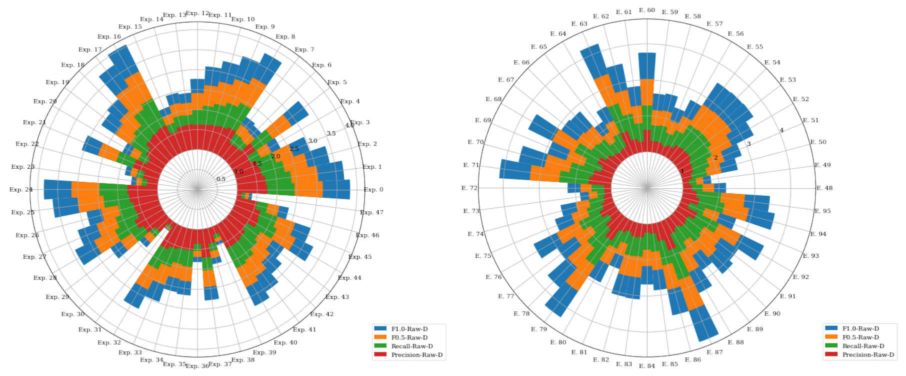
#	$Prec.$	$Prec.'$	$\nabla Prec.$	$Rec.$	$Rec.'$	$\nabla Rec.$	$F_{0.5}$	$F'_{0.5}$	$\nabla F_{0.5}$	F_1	F'_1	∇F_1
E_{49}	.893	.591	.302	.835	.408	.427	.872	.412	.460	.851	.369	.482
E_{50}	.912	.803	**.108**	.887	.666	.220	.904	.699	**.205**	.895	.659	.236
E_{51}	.931	.733	.198	.909	.619	.290	.924	.608	.316	.917	.578	.339
E_{52}	.973	.667	.305	.969	.408	.560	.972	.424	.548	.971	.373	.598
E_{53}	.971	**.814**	.157	.974	.709	.265	.971	.707	.264	.972	.685	.286
E_{54}	.976	.700	.276	.968	.600	.368	.973	.607	.366	.970	.575	.395
E_{55}	.956	.651	.304	.953	.472	.481	.955	.443	.512	.954	.416	.538
E_{56}	.971	.777	.193	.968	.677	.290	.970	.695	.275	.969	.659	.309
E_{57}	.970	.708	.262	.965	.574	.391	.968	.559	.408	.966	.532	.433
E_{58}	.949	.639	.309	.933	.416	.517	.943	.388	.554	.937	.365	.572
E_{59}	.965	.776	.188	.960	.541	.418	.963	.614	.349	.961	.542	.418
E_{60}	.955	.678	.276	.944	.558	.385	.951	.535	.415	.947	.512	.434
E_{61}	.926	.779	.147	.886	.573	.313	.911	.646	.265	.897	.583	.314
E_{62}	.931	.558	.373	.909	.288	.621	.918	.341	.577	.910	.282	.628
E_{63}	.940	.734	.205	.931	.566	.365	.935	.608	.327	.931	.549	.382
E_{64}	.990	.700	.290	**.990**	.426	.564	.990	.462	.528	**.990**	.404	.586
E_{65}	**.991**	.472	.519	.989	.266	.723	**.991**	.269	.722	**.990**	.222	.768
E_{66}	.986	.586	.400	.984	.313	.671	.985	.318	.667	.984	.285	.699
E_{67}	.982	.382	.600	.982	.146	.836	.981	.177	.804	.981	.141	.840
E_{68}	.985	.556	.428	.979	.258	.721	.983	.273	.710	.981	.220	.761
E_{69}	.969	.551	.417	.960	.328	.631	.966	.381	.585	.963	.320	.643
E_{70}	.976	.384	.592	.967	.238	.729	.973	.260	.713	.969	.223	.746
E_{71}	.967	.494	.473	.969	.142	.827	.966	.151	.815	.966	.105	.861
E_{72}	.965	.707	.258	.969	.434	.534	.965	.481	.484	.966	.420	.546
E_{73}	.889	.592	.297	.853	.443	.410	.877	.423	.454	.864	.392	.472
E_{74}	.910	.788	.122	.849	.654	**.194**	.885	.679	**.205**	.863	.643	**.219**
E_{75}	.879	.712	.167	.826	.566	.260	.859	.554	.304	.839	.524	.314
E_{76}	.952	.642	.309	.945	.473	.472	.949	.461	.487	.946	.422	.524
E_{77}	.960	.804	.155	.962	**.717**	.245	.959	**.734**	.224	.959	**.704**	.255
E_{78}	.961	.758	.202	.951	.608	.342	.958	.626	.331	.954	.588	.366
E_{79}	.949	.575	.374	.914	.432	.482	.937	.381	.556	.925	.359	.566
E_{80}	.958	.756	.201	.960	.646	.313	.958	.652	.305	.957	.617	.339
E_{81}	.954	.687	.266	.949	.539	.409	.952	.569	.383	.949	.526	.422
E_{82}	.933	.527	.406	.896	.374	.522	.917	.360	.557	.905	.339	.566
E_{83}	.939	.722	.216	.927	.495	.432	.931	.568	.363	.925	.501	.424
E_{84}	.951	.638	.312	.941	.462	.478	.948	.482	.465	.944	.449	.494
E_{85}	.909	.618	.291	.882	.391	.491	.896	.423	.473	.885	.374	.511
E_{86}	.909	.598	.311	.845	.355	.490	.884	.379	.505	.860	.332	.528
E_{87}	.940	.798	.141	.926	.433	.493	.935	.536	.399	.929	.449	.480
E_{88}	.972	.758	.213	.964	.461	.502	.965	.485	.480	.962	.425	.536
E_{89}	.976	.564	.412	.975	.408	.567	.975	.410	.565	.974	.362	.612
E_{90}	.984	.661	.322	**.990**	.401	.589	.984	.455	.528	.986	.400	.586
E_{91}	.979	.245	.734	.973	.144	.829	.977	.120	.857	.975	.096	.879
E_{92}	.982	.349	.633	.979	.168	.810	.981	.156	.825	.980	.141	.839
E_{93}	.972	.467	.504	.969	.241	.728	.970	.259	.711	.969	.218	.751
E_{94}	.929	.650	.279	.941	.279	.661	.927	.330	.597	.928	.281	.647
E_{95}	.930	.606	.324	.930	.310	.620	.921	.348	.573	.918	.290	.628
E_{96}	.930	.574	.356	.921	.259	.662	.922	.298	.624	.917	.245	.672

Table 4. Resulting test performance of the configurations trained with the parameters in Table 2 over the original data set and its pre-processed variants along with those obtained with the Synth-PV dataset and its difference. The maximum values for $Prec.$, Rec and F_β values and minimum values for ∇ are shown in bold



(a) Edge-Architecture (b) Higher Performance Computer

Fig. 3. Performance results obtained by each of the 96 experiment configurations with the PlantVillage datasets and its pre-processed variants



(a) Edge-Architecture (b) Higher Performance Computer

Fig. 4. Performance results obtained by each of the 96 experiment configurations with the synthetic PlantVillage dataset

those configurations trained over preprocessed dataset tend to be more stable and robust during the validation process with the simulated data.

6. Conclusions

In this work, we have analyzed different network architectures and configurations for the automatic detection of plant disease. We have shed light on several important aspects while acknowledging its inherent limitations. We have recognized potential biases in the dataset used, stemming from factors like geographical and temporal distribution of diseases, which can influence model performance. Furthermore, the complex real-world context of plant disease management introduces confounding factors such as weather conditions, soil quality, and agricultural practices, which must be considered when interpreting our results.

Throughout our research, we have grappled with the trade-offs between precision and recall, recognizing that achieving high precision may entail the risk of missing some cases while emphasizing recall can lead to more false alarms. These trade-offs underscore the need for a nuanced approach that can be tailored to specific agricultural scenarios and disease management strategies.

While subject to these limitations, our findings hold significant promise for practical applications in agriculture and plant disease management. The deep learning models and strategies identified can be valuable tools in real-world scenarios. They can facilitate early disease detection, providing farmers with timely information for proactive intervention. Moreover, the analyzed models can contribute to precision farming by pinpointing affected areas within fields, thus reducing the indiscriminate use of pesticides and minimizing environmental impact. This aligns with the global push toward sustainable agriculture, where minimizing chemical inputs is a key goal.

Looking ahead, our research opens doors to further exploration. Addressing dataset biases through more extensive and diverse data collection efforts can result in more robust models with broader applicability. Investigating the generalization of our models to different geographical regions and agricultural settings is another promising avenue. Additionally, interdisciplinary collaboration with plant pathology, agriculture, and environmental science experts can enhance the practicality and real-world relevance of our disease detection models. Developing interactive systems that involve farmers and agricultural experts in the decision-making process is another avenue to explore, allowing for more context-aware disease management.

Acknowledgments. The authors gratefully acknowledge the financial support of Consellería d'Innovació, Universitats, Ciencia i Societat Digital from Comunitat Valenciana (APOSTD/2021/227), the European Social Fund (Investing In Your Future), grant PID2021-123673OB-C31 funded by MCIN/AEI/ 10.13039/501100011033 and by "ERDF A way of making Europe" and grant from the Research Services of Universitat Politècnica de València (PAID-PD-22).

References

1. Atila, Ü., Uçar, M., Akyol, K., Uçar, E.: Plant leaf disease classification using efficientnet deep learning model. Ecological Informatics 61, 101182 (2021)
2. Atila, U., Uçar, M., Akyol, K., Uçar, E.: Plant leaf disease classification using EfficientNet deep learning model. Ecological Informatics 61, 101182 (2021)
3. Barbedo, J.G.A.: A review on the main challenges in automatic plant disease identification based on visible range images. Biosystems engineering 144, 52–60 (2016)
4. Bezdek, J.C., Chandrasekhar, R., Attikouzel, Y.: A geometric approach to edge detection. IEEE Transactions on Fuzzy Systems 6(1), 52–75 (1998)
5. Bhakta, I., Phadikar, S., Majumder, K., Mukherjee, H., Sau, A.: A novel plant disease prediction model based on thermal images using modified deep convolutional neural network. Precision Agriculture pp. 1–17 (2022)
6. Camargo, A., Smith, J.: Image pattern classification for the identification of disease causing agents in plants. Computers and Electronics in Agriculture 66(2), 121–125 (2009)
7. Costa, L., Nunes, L., Ampatzidis, Y.: A new visible band index (vndvi) for estimating ndvi values on rgb images utilizing genetic algorithms. Computers and Electronics in Agriculture 172, 105334 (2020)

8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
9. Devi, R., Kumar, V., Sivakumar, P.: Efficientnetv2 model for plant disease classification and pest recognition. Computer Systems Science & Engineering 45(2) (2023)
10. Dong, Z.: Image-Based Plant Leaf Disease Recognition with InceptionV3 Network. Ph.D. thesis, The Ohio State University (2021)
11. Elfatimi, E., Eryigit, R., Elfatimi, L.: Beans leaf diseases classification using mobilenet models. IEEE Access 10, 9471–9482 (2022)
12. Glegoła, W., Karpus, A., Przybyłek, A.: Mobilenet family tailored for raspberry pi. Procedia Computer Science 192, 2249–2258 (2021)
13. Gueye, Y., Mbaye, M.: Kmeans kernel-learning based ai-iot framework for plant leaf disease detection. In: International Conference on Service-Oriented Computing. pp. 549–563. Springer (2020)
14. Gui, P., Dang, W., Zhu, F., Zhao, Q.: Towards automatic field plant disease recognition. Computers and Electronics in Agriculture 191, 106523 (2021)
15. Gui, P., Dang, W., Zhu, F., Zhao, Q.: Towards automatic field plant disease recognition. Computers and Electronics in Agriculture 191, 106523 (2021)
16. Hasan, R.I., Yusuf, S.M., Alzubaidi, L.: Review of the state of the art of deep learning for plant diseases: A broad analysis and discussion. Plants 9(10), 1302 (2020)
17. Hassan, S.M., Maji, A.K.: Plant disease identification using a novel convolutional neural network. IEEE Access 10, 5390–5401 (2022)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
19. Hughes, D., Salathé, M., et al.: An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060 (2015)
20. Khan, S., Narvekar, M.: Novel fusion of color balancing and superpixel based approach for detection of tomato plant diseases in natural complex environment. Journal of King Saud University - Computer and Information Sciences (2020)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems 25 (2012)
22. Lee, S.H., Goëau, H., Bonnet, P., Joly, A.: Attention-based recurrent neural network for plant disease classification. Frontiers in Plant Science 11, 601250 (2020)
23. Madrid, N., Lopez-Molina, C., Hurtik, P.: Non-linear scale-space based on fuzzy contrast enhancement: Theoretical results. Fuzzy Sets and Systems 421, 133–157 (2021)
24. Mahesh, T., Sivakami, R., Manimozhi, I., Krishnamoorthy, N., Swapna, B., et al.: Early predictive model for detection of plant leaf diseases using mobilenetv2 architecture. International Journal of Intelligent Systems and Applications in Engineering 11(2), 46–54 (2023)
25. Marco-Detchart, C., Lopez-Molina, C., Fernandez, J., Bustince, H.: A gravitational approach to image smoothing. In: Advances in Fuzzy Logic and Technology 2017, pp. 468–479. Springer (2017)
26. Mohd Noor, F.N., Mohd Isa, W.H., Khairuddin, I.M., Mohd Razman, M.A., Musa, R.M., PP Abdul Majeed, A., et al.: The diagnosis of diabetic retinopathy: An evaluation of different classifiers with the inception v3 model as a feature extractor. In: International Conference on Robot Intelligence Technology and Applications. pp. 392–397. Springer (2022)
27. Ngo, H., Fang, H., Wang, H.: Beamforming and scalable image processing in vehicle-to-vehicle networks. Journal of Signal Processing Systems pp. 1–10 (2022)
28. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 427–436 (2015)
29. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on pattern analysis and machine intelligence 12(7), 629–639 (1990)

30. Poma, X.S., Riba, E., Sappa, A.: Dense extreme inception network: Towards a robust cnn model for edge detection. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1923–1932 (2020)
31. Rumpf, T., Mahlein, A.K., Steiner, U., Oerke, E.C., Dehne, H.W., Plümer, L.: Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance. Computers and electronics in agriculture 74(1), 91–99 (2010)
32. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. Advances in neural information processing systems 30 (2017)
33. Samin, O.B., Omar, M., Mansoor, M.: CapPlant: a capsule network based framework for plant disease classification. PeerJ Computer Science 7, e752 (2021)
34. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
35. Schuler, J., Romaní, S., Abdel-nasser, M., Rashwan, H., Puig, D.: Reliable Deep Learning Plant Leaf Disease Classification Based on Light-Chroma Separated Branches, pp. 375–381. IOS Press (10 2021)
36. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
37. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
38. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
39. Tan, M., Le, Q.: Efficientnetv2: Smaller models and faster training. In: International Conference on Machine Learning. pp. 10096–10106. PMLR (2021)

Cedric Marco-Detchart received his PhD on Artificial Intelligence from Universidad Pública de Navarra, and is a researcher at the Valencian Research Institute for Artificial Intelligence (VRAIN), focusing on enhancing cognitive assistance systems through comparison measures, with a particular interest in computer vision, advances in psychology and neurosciences.

Jaime Andrés Rincón Arango has a PhD in Computer Science from the Universitat Politècnica de València; he is currently an assistant professor at the University of Burgos, where he teaches. Some of the research lines in which he works are multi-agent systems, IoT, IoMT, Robotics, Edge AI, Machine Learning, Virtual Reality and Mixed Reality.

Carlos Carrascosa Casamayor holds a PhD in Computer Science from the Universitat Politècnica de València, where he has been teaching and researching since 1996, currently as an Associate Professor. Some of the research lines he is working in are: multi-agent systems, virtual organisations, agreement technologies, adaptive systems, federated learning, consensus algorithms, intelligent virtual environments, affective computing, real-time systems and the so-called ”serious games”.

Vicente Julian is Full Professor of the Department of Computer Systems and Computing at the Polytechnic University of Valencia (UPV) since 2017. He is also Deputy Director of Research of the Department of Computer Systems and Computation and Coordinator of the Doctorate Program in Computer Science of the UPV.

Received: December 22, 2022; Accepted: October 10, 2023.