الجمـهوريــة الجزائريــة الديمقراطيــة الشعبيــة

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليـم العالــي و البحــث العلمـي

Ministry Of Higher Education and Scientific Research

Frères Mentouri Constantine 1 University
Faculty of Technology Sciences
Department of Electronic

جامعـة الإخوة منتوري قسنطـينة **1**
كلية علوم التكنولوجية
قسم الإلكترونيك

Order N°: …….. **Series: ….**

A Thesis

Submitted in Partial fulfillment of the Requirements for

The degree of Doctorate Third Cycle

Major : Télécommunications

Option: Signaux et Systèmes de Télécommunications

Theme

Development of image analysis techniques using machine learning

---

By

Salmi Abderrahmane

Committee Members:

President	Faouzi Soltani	Professor Frères Mentouri Constantine 1 University
Supervisor	Said Benierbah	Professor Frères Mentouri Constantine 1 University
Examiner	Meriem Hacini	Assoc-Professor Frères Mentouri Constantine 1 University
Examiner	Makhlouf Derdour	Professor Larbi Ben M'hidi University-Oum El Bouaghi
Examiner	Abdallah Meraoumia	Professor Larbi Tebessi University-Tebessa

Academic year 2021/2022

Defense date: May 31th, 2022

الجمـهوريــة الجزائريــة الديمقراطيــة الشعبيــة

République Algérienne Démocratique et Populaire

وزارة التعليـم العالـي و البحـث العلمـي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des frères Mentouri Constantine 1
Faculté des Sciences de la Technologie
Département d'Electronique

جامعـة الإخوة منتوري قسنطـينة**1**
كلية علوم التكنولوجية
قسم الإلكترونيك

N° d'Ordre: …….. **Série: ….**

Thèse :

Présentée pour Obtenir le Diplôme de

Doctorat Troisième Cycle

Filière : Télécommunications

Spécialité : Signaux et Systèmes de Télécommunications

Développement de techniques d'analyse des images utilisant l'apprentissage automatique

---

Par :

Salmi Abderrahmane

Présentée et soutenue publiquement devant le jury:

Président	Soltani Faouzi	Professeur Université des Frères Mentouri de Constantine 1
Rapporteur	Benierbah Said	Professeur Université des Frères Mentouri de Constantine 1
Examinateur	Hacini Meriem	MCA Université des Frères Mentouri de Constantine 1
Examinateur	Derdour Makhlouf	Professeur Université de Larbi Ben M'hidi -Oum El Bouaghi
Examinateur	Meraoumia Abdallah	Professeur Université de Larbi Tebessi -Tebessa

Année Universitaire 2021/2022

Date de soutenance : 31 Mai 2022

Dedication

I would like to thank my supervisor, Pr. Said Benierbah, for his unwavering support and guidance throughout this thesis.

I would also like to acknowledge and thank Pr. Toufik Laroussi as well as the following people who have supported & helped me, not only during the preparation of this research project but even throughout my Ph.D.

To begin with, I would like to thank fellow colleague, Mehdi Ghazi for his tremendous input and expertise in the idea development, and his supportive advice and help during the testing stages.

In addition, I would like to thank all my close friends and family. Your encouragement has helped me to focus on what has been a rewarding and enriching process. This thesis would not be possible without your collective support.

Acknowledgment

The work presented in this doctoral thesis was developed within the collaboration between the Signals and Communication Systems Laboratory (SISCOM) and Signal Processing LABoratory (SP LAB), Department of Electronics, Frères Mentouri University-Constantine 1. My thanks go first and foremost to Almighty God, for having granted me health, will, and courage. I would like to thank my thesis supervisor Professor Said Benierbah for honoring me with trust and for entrusting me with an excellent research topic. I also thank you for your seriousness, your time, and your fruitful advice Mr. President of Jury: Professor ***Soltani Faouzi*** *Thank you for the honor you do me by agreeing to evaluate this work. Please Sir, find here the assurance of my gratitude and my deep admiration.*

My heartfelt thanks to the members of the jury, The Professors:

- ***Derdour Makhlouf***
- ***Abdallah Meraoumia***
- ***Meriem Hacini***

I would also like to thank

All the professors of the SISCOM & SP Lab Laboratories for their advice, their kindness, and their seriousness.

Finally, I thank all researchers in the field of image processing, computer vision, and telecommunications for their interesting research works.

Abstract

Image analysis and computer vision play a crucial role in many real-world applications such as smart agriculture and smart vehicles. For plant diseases diagnosis, one of the most recent challenges is the improvement of the plant diseases classification on Low-Resolution (LR) images. The farmer is supposed to obtain High-Resolution (HR) images of plant leaves from the field. In this thesis, we propose two contributions. We first propose to fine-tune a low-complex classifier named Dense Convolutional Network (DenseNet) on HR plant leaves images to detect tomato leaves diseases. Because of the small size of plant leaves and other limitations, the obtained HR images can miss some detailed information that results in blurred LR images of leaves with fewer details. As a second contribution, we introduce a novel Super-Resolution (SR) algorithm named Wider-activation for Attention-mechanism based on a Generative Adversarial Network (WAGAN) to improve the diseases classification of the tomato LR images. To evaluate the potential of the proposed SR method in plant diseases recognition, we first recovered SR plant diseases images from LR images using the WAGAN. Next, we compared the performance of the diseases classification using LR, SR, and HR images. The classification results proved the efficacy of the proposed SR method (97.63%) with ×3.6 lower complexity than the state-of-the-art method and very close to the reference HR (97.81%) accuracy. Due to the efficient design of proposed SR model, the WAGAN focuses more on edges, textures, and other valuable information, which are the key information, needed for the DenseNet to recognize the diseases.

Keywords: Image analysis, smart agriculture, Low-Resolution, Super-Resolution, classification.

Résumé

L'analyse d'images et la vision par ordinateur jouent un rôle important dans de nombreuses applications telles que l'agriculture intelligente et les voitures autonomes. Pour diagnostiquer des maladies de plantes, l'un des défis les plus récents est l'amélioration de la détection des maladies des plantes en utilisant des images de Faible Résolution (FR). L'agriculteur est censé obtenir des images des feuilles de plantes à haute résolution (HR) depuis le champ d'agriculture. Notre thèse porte sur deux contributions. Dans un premier temps, nous avons affiné un modèle de classification à faible complexité nommé Dense Convolutional Network (DenseNet) sur des images de feuilles de plantes de HR pour détecter les maladies des feuilles de tomate. En raison de la petite taille des feuilles de plantes et d'autres limitations, les images HR obtenues peuvent manquer certaines informations utiles. Ils seront présentés comme des images FR floues des feuilles avec moins de détails. Pour cela, et dans la deuxième contribution, nous avons proposé un nouvel algorithme de Super-Résolution (SR) nommé Wider-activation for Attention-mechanism basé sur un réseau antagoniste génératif (WAGAN) qui permet d'améliorer la classification des images FR des maladies de la tomate, ce qui est détaillé dans notre deuxième contribution. Afin d'évaluer le potentiel de la méthode SR proposée dans la reconnaissance des maladies des plantes, nous avons d'abord récupéré les images SR des maladies des plantes à partir des images FR à l'aide du WAGAN. Par la suite, nous avons mis en œuvre la classification des maladies à l'aide d'images FR, SR et HR. Les résultats de la classification obtenus ont démontré l'efficacité de notre méthode SR (97,63 %) avec une complexité de ×3.6 inférieure à celle des autres méthodes de l'état de l'art et une précision très proche de l'image référence HR (97,81 %). En raison de sa concentration sur les détails des bordures, les textures et d'autres informations nécessaires des images, le modèle SR que nous avons proposé (WAGAN) s'avère très efficace avec le model DenseNet pour améliorer la reconnaissance des maladies des plantes telle que la tomate.

Mots-clés : analyse d'images, l'agriculture intelligente, Faible Résolution, Super-Résolution, classification.

ملخص

يلعب تحليل الصور ورؤية الكمبيوتر دورًا مهمًا في العديد من المجالات خاصة الزراعة الذكية. يعتبر تشخيص أمراض النبات أحد أكثر التحديات خاصة في مجال تحسين تصنيف صور أمراض أوراق النبات ذات الجودة منخفضة (LR) . من المفترض أن يتحصل المزارع على صور أوراق النبات من الحقل بجودة عالية (HR). في هذه الأطروحة ، نقترح مساهمتين. نقترح أولاً تعديل مصنف صور ذو تعقيد منخفض يسمى الشبكة العصبونية الكثيفة (DenseNet) على صور أوراق نبات الطماطم بجودة عالية لاكتشاف الأمراض. نظرًا لصغر حجم أوراق النبات وعوائق أخرى، يمكن أن تفقد الصور ذات الجودة العالية (التي تم الحصول عليها) بعض المعلومات التفصيلية التي تؤدي إلى عدم وضوح صور الأوراق منخفضة الجودة و بتفاصيل أقل. لذلك و كمساهمة ثانية، قمنا بتصميم خوارزمية تحسين الدقة (SR) تسمى تنشيط أوسع لآلية الانتباهً بواسطة شبكة الخصومة التوليدية (WAGAN)لتحسين تصنيف الصور ذات الجودة منخفضة لأمراض الأوراق. لتقييم إمكانات خوارزمية تحسين الدقة المقترحة في تحسين التعرف على أمراض النبات ، قمنا أولاً باسترجاع صور أمراض النبات SR من صور LR باستخدام WAGAN . بعد ذلك ، قمنا بتصنيف الأمراض باستخدام صور LR و SR و HR. أثبتت نتائج التصنيف فعالية خوارزمية تحسين الدقة المقترحة (97.63٪) مع تعقيد أقل ب × 3.6 من الطرق المقترحة وقريبة جدًا من دقة الصور ذات الجودة العالية المرجعية (97.81٪). نظرًا للتصميم الفعال للخوارزمية المقترحة (تحسين الدقة)، يركز WAGAN بشكل أكبر على الحواف والأنسجة والمعلومات الأخرى ذات قيمة، والتي تعد المعلومات الأساسية اللازمة لشبكة DenseNet للتعرف على الأمراض.

كلمات مفتاحية: تحليل الصور، الزراعة الذكية، الجودة منخفضة ، تحسين الدقة، تصنيف صور.

Content

Content

Content

List of Tables

List of Figures

List of Figures

List of Figures

List of Figures

List of acronyms

SR:	Super Resolution
HR:	High Resolution
LR:	Low Resolution
SISR:	Single Image Super-Resolution
ML:	Machine Learning
DL:	Deep Learning
AI:	Artificial Intelligence
UAV:	Unmanned Aerial Vehicle
DenseNet:	Densely Convolutional Network
ResNet:	Residual Neural Network
TP:	True Positive
FP:	False Positive
FN:	False Negative
TN:	True Negative
CNN:	Convolutional Neural Network
GAN:	Generative Adversarial Network
SVM:	Support Vector Machine
K-NN:	K-Nearest Neighbor
VGG-Net:	Visual Geometry Group Network
MSE:	Mean Square Error
SSIM:	Structural SIMilarity
PSNR:	Peak Signal-to-Noise Ratio

General Introduction

Agriculture has a crucial role in the economic development of every country. Tomato represents the world's 3rd largest vegetable crop after potato and onion. Tomato is known as protective food because of its thick skin, special nutritive value, and widespread production. The world production of tomatoes is 177,118,248 tons per year [1]. Algeria is ranked the 17th with an annual production of 1,280,570 tons. The production of tomatoes is not enough and needs to be increased. For this, most countries focus on the early recovery of damaged plants. This operation can be performed by observing the plants; since most tomato diseases can be first identified on the leaves.

Early recovery of damaged plants may help to increase crop productivity. In past, experts monitored the field with the bare eye to identify plant diseases. Due to the limited number of experts in rural zones, the bare eye showed a lack of accuracy and was time-consuming. Automated image classification [2] is a process that can classify an image based on its visual content. Thus, the use of automated (Machine Learning (ML) and Deep Learning (DL)) classification can detect diseases instantly, without human assistance, and more accurately than traditional methods as well as continuous monitoring of plant health conditions.

The use of ML networks to solve a given problem needs feature extraction identification by an expert and hand-coded according to the domain and data type. For DL, the network retrieves the features directly from a given data during the training phase. However, it requires more data than ML for better feature extraction and learning. For example, a DL network takes directly raw images, sound, or text to perform classification tasks. Thus, we decided to use DL to solve the diseases identification of tomato leaves problem.

No matter what technology is used for field monitoring, producing HR images is not always feasible and is much more expensive due to the limitations of the sensor, power, optics manufacturing technology, or other factors. However, a quality degradation of the input image may have a direct impact on the algorithm performance. Classification usually requires HR images as input. High-Resolution (HR) images contain a high pixel density and more details about the original scene. Thus, HR is highly recommended in agriculture imaging diagnosis, since it is sensitive to regions of interest. Low-Resolution (LR) and zoom-in on that region can give poor results. Super-Resolution (SR) can produce HR from LR images. Adopting an SR algorithm before

the classifier, which is relatively inexpensive, may improve the classifier performance by producing SR images with similar details to the HR ones.

In most communication systems, sending a small amount of data and receiving it without any misses is the optimal case. In our thesis, we used DL with classification and image analysis (SR) to improve the agriculture communication system. Furthermore, we focused on retrieving the missed information (from the sender) of HR images at the farmer workstation (receiver) by introducing a new SR algorithm to improve the classification of LR tomato leaves images.

Fig 1 below represents the scheme communication of the agriculture monitoring system. The machine (robot/ satellite/ unmanned aerial vehicle) starts capturing images from the agriculture field. For further interpretation, the machine sends these images in real-time (or offline) to the Farmer workstation. The Famer workstation is a computer used for plant interpretation and diagnosis. Consequently, the received images might be blurred (after upscaling) or only small parts of the lesion might be present in an image, which can be considered an LR image. The reason for getting LR images is usually because of the sensor limitations (satellite), capturing altitude (UAV), or the movement on uneven soil (robots). For example, the most common issue for UAVs is the airflow generated by the flying UAV, which makes the plants swirl. The UAVs fly at a higher altitude to prevent the generated airflow, whereas the received images may lack to contain more detailed information (tiny size of diseases). Consequently, these images are considered as LR. Concerning the farm robot, the movement on uneven soil, speed of the robot, and other conditions may lack to capture more specific details (considered as LR that needs upscaling) about the diseases, even the light breeze can be a serious factor. Thus, the classifier may lack to classify the diseases. In this case, the unique option is to diagnose the disease based on insufficient information or repeat the process in the hopes of obtaining a better view of the lesion. To tackle these difficulties, we proposed to improve the integration of a DL SR algorithm before the disease classifier, at the farmer workstation, to boost the classification accuracy. Furthermore, the SR algorithm provides the classifier with an upscaled image for a better collection of useful information. Thus, the machine will be able to provide farmers with sufficient image samples of diseases to make an accurate diagnosis.

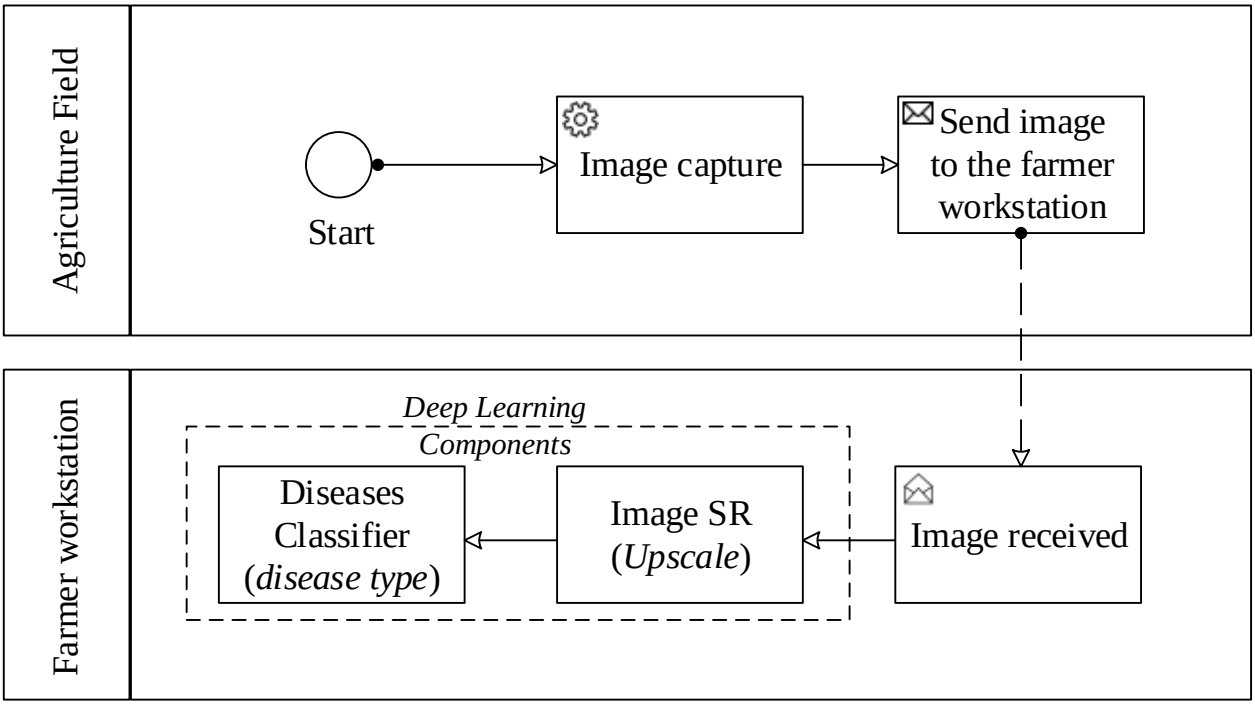


Fig 1 overview of the proposed communication scheme of the agriculture monitoring system.

In the first contribution, we propose to fine-tune a low-complex classification model named Dense Convolutional Network (DenseNet) on HR plant leaves to detect tomato leaves diseases. DenseNet is a convolutional neural network that uses Dense Blocks to connect all layers (with matching feature-map sizes) directly to each other, resulting in dense connections between layers. This structure helped the model to be compressed in terms of the number of parameters. Extensive results showed that the DenseNet-121 can identify the tomato leaves diseases with a challenging model size as well as a robust classification accuracy. This helped us to choose DenseNet-121 as the low-complexity classifier for further analysis.

In the second contribution, we supposed that the farmer obtained HR images of plant leaves from the field. Because of the small size of plant leaves and other limitations, the obtained HR images can miss some detailed information resulting in blurred LR images of leaves with fewer details. Extensive results using LR images demonstrated that DenseNet-121 lacked to identify the tomato leaves diseases. To overcome this issue, we introduced a novel SR algorithm named Wider-activation for Attention-mechanism based on a Generative Adversarial Network (WAGAN) to improve the classification of the tomato diseases' LR images. The WAGAN consists of three main parts; the generator network, which has the Wider Activation for Residual Channel Attention Block

(WARCAB) as the principal block, the discriminator network, and the adversarial loss. To evaluate the potential of the proposed method in plant diseases recognition, we first recovered SR plant diseases images from LR images using the WAGAN.

This thesis consists of three chapters and is organized as follows:

The first chapter covers the different precision agriculture technologies used for plant diagnosis. Also, it presents most common tomato leaves diseases. Then, it introduces our contributions. The last section concludes the chapter.

The second chapter discusses the necessary background knowledge for ML, DL, and image analysis (SR).

The third chapter contains sections. Section 1 introduces the proposed method. Section 2 presents a literature review of the classification methods. Section 3 shows the details of the DenseNet-121. Section 4 presents the datasets and training settings. Section 5 discusses the obtained results and the last section concludes the chapter.

The Fourth chapter is organized as follows. Section 1 introduces the proposed work. Section 2 formulate the research problem. Section 3 presents a literature review of the SR and classification methods. Section 4 shows the details of the proposed method. Section 5 performs case studies on the classification of the generated SR images. Section 6 discusses the obtained results and Section 7 concludes the chapter. Section 8 describes the general conclusion and the future work. Section 9 listed the published publications.

Chapter 1

Precision agriculture technologies for plant diagnosis

1. Introduction

Precision agriculture is an agriculture set of cyclic system steps, which consists of localization, data collection, data analysis, decisions management in the field, and field evaluation. Robots and RS technologies are mandatory to perform different precision agriculture tasks. For setting the automation of robots and Remote Sensing (RS) technologies in precision agriculture, the monitoring and the control of environmental parameters are crucial. Most precision agriculture historical data are stored and used for future decision-making, especially for plant protection and harvest purposes. In this chapter, we cover the different precision agriculture technologies used for plant diagnosis. Also, it presents most common tomato leaves diseases. Then, it introduces our contributions.

The rest of the chapter is organized as follows. Section 2 covers the remote sensing technologies for plant diagnosis. Section 3 defines different diseases of tomato leaves. Section 4 introduces our contributions. Section 5 concludes the chapter.

2. Remote sensing technologies for plant diagnosis

The use of RS monitoring in plant diseases and pests is considered a radio-diagnosis of plants. The RS can provide spatial and continuous monitoring of the field (plant diseases and pests). In 1989, Riley *et al.* [3] proved that the interpretation of aerial or satellite images is possible, and they could identify the damaged regions due to the presence of pests and plant diseases.

RS systems can detect plant diseases based on the physiological changes caused by diseases (necrotic tissues or color of the leaves), pests' attacks, or the loss of plant rigidity due to dehydration. Besides, the RS technologies can detect the wide damaged regions in the field and not small areas.

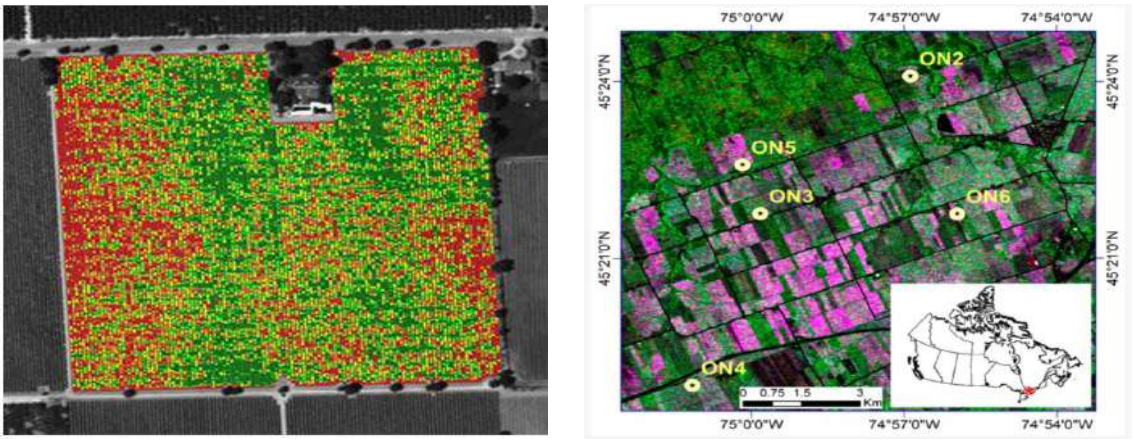
By performing both passive and active radiation, several RS systems are able to detect and monitor plant diseases and pests. More formally, they allow data acquisition ranging from gamma rays to microwaves. RS systems are generally classified into three categories based on the functionality in plant diseases and pests monitoring:

(1) Visible and near-infrared Short Wave InfraRed (VIS-SWIR) spectral systems are available in ground-based, aerial, and satellites. This system helps to determine the damaged plants by the reflectance spectra.

(2) Fluorescence and thermal systems are used for tracking the photosynthetic and respiration processes of plants. More precisely, they permit the presymptomatic monitoring of pests and plant diseases. However, Laser-Induced Fluorescence (LIF) is mostly employed to get the fluorescence response of plants.

(3) Synthetic aperture radar (SAR) and Light Detection and Ranging equipment (Lidar) systems are recently used for plant diseases and pest monitoring. Also, they can be employed to delineate plant environments and their characteristics. The SAR has the ability to retrieve soil characteristics, plant water information, and a few structural parameters of the plant canopy, particularly for tall plants. Fig 1.1 shows an example of captured images via Lidar and SAR.

For the Lidar system, it can obtain detailed canopy morphology. The point cloud data collected by Lidar enables the reconstruction of the 3D plant canopy structure. However, this 3D reconstruction may indicate damaged parts due to pests or diseases.



(a) (b)

Fig 1.1 Lidar (a) and SAR (b) capture structural or morphological changes due to diseases and pests. (b) Soil Monitoring for Agriculture in-situ network in eastern Ontario (Canada) [4].

2.1 Unmanned Aerial Vehicles in precision farming

The most common applications of the UAVs in precision farming are crop health monitoring and pesticide spraying. Nowadays, UAVs are mostly used for agricultural planning, productive data analysis, and related spatial information collection. Fig. 1.2 illustrates a UAV performing field mapping and crop monitoring. On daily monitoring, the UAV flies on a specified trajectory, set by the farmer, above the field. It used the cameras to collect the data like monitoring the crops or the most common diseases and field mapping using advanced computer vision and image analysis. However, crop information like nutrients requirements, crop diseases, and water stress are estimated based on some specific indices. Some indices like vegetation helped to differentiate between weeds, healthy, and unhealthy plants. Such indices are based on the crop's image spectrum, where the spectrum helps to specify the crop health conditions. The UAV may help to detect a specific region that contains weeds or damaged plants, but cannot specify the crop diseases easily as a farm robot does. Usually, the UAVs are mostly used in monitoring crop health and help to take future actions for preventing crop spoiling.

In agriculture, some common commercial UAVs (such as DJI Phantom 4 PRO, AGCO Solo, Sentera Omni Ag, SenseFly eXom, AgBot, and InDago AG) are used for some specific tasks like Nutrition, crop stress considering local field needs, spot pesticide, spraying, monitoring of small fields, crop height, health condition of crop estimation, soil and field analysis, and water stress calculation. To perform effective crop monitoring, choosing the UAV sensor type for a specific task is necessary such as nutrients detection, disease detection, water status identification, etc.

(a) (b)

Fig. 1.2 UAV application in agricultural field (a) Field mapping (b) Crop monitoring [5].

2.2 Robots in precision farming for plant diagnosis

Technological development in computer vision, laser, global positioning systems (GPS), and mechatronics allowed the implementation of more accurate robotic systems, especially in smart agriculture.

Fig 1.3 example of farm robots, Ladybird is on the left, SMP S4 in the middle, Mantis on the right. [6]

Agricultural robotics is divided into three principal categories: weed control, field scouting, and harvesting. The weed control consists of a platform of wheeled mounting sprays and sensors for chemical control, fertilizer applications, and weed identification. For instance, AgBot II is a lightweight farm vehicle implemented by the Queensland University of Technology for weed identification, removing weeds mechanically, and spot-spraying. In the same manner, the Swiss Ecorobotix company funded the wheeled autonomous robots named AVO and ARA, which are powered by solar panels. For the second category (field scouting), the agricultural robots are categorized by robotic arms and mounting the dedicated environmental sensors for field samples collection or precision maneuvers. These samples contain useful information (2D images or 3D reconstruction images) about plant health, which are mainly used for plant diagnosis and protection. As an example, GRAPE is a ground robot developed by the European Union. This robot was designed for monitoring and vineyard health protection. In another development, the European program funded the Trimbot2020 for automatic bush trimming, which has 6 DOFs manipulators on a stabilized platform. Finally, the harvesting robots particularly have additional cameras integrated into them to improve the vision sensors for better manipulation of the end-effector. On a unique DOF mobile platform, they mounted a robotic arm for a better workspace adaptation. To realize such a robot, the National Technology Research and Development Program of China implemented a mobile harvest robot with lifting support for tomato harvesting.

Fig 1.4 Field data collection system "TerraSentia Robot" in the agricultural field [6]

Field scouting and data collection robots: The field scouting robots discipline has different challenges such as providing reliable data and measurements for precision agriculture processing. Some scouting robots have a mission to retrieve the inherent physical and biological variability of plants. Therefore, these robots should be flexible, multipurpose, and affordable, which yields a reduction in production cost, improves quality, increases productivity, and allows accurate plant and crop treatments. The use of advanced sensors in the scouting robots for data collection is necessary to produce valuable information in many tasks like obstacle avoidance, navigation and manipulator control, and 3D environment reconstructions. For instance, different multi-spectral imaging and LiDAR sensors are integrated into some robots to automate the monitoring and the reconstruction of 3D images of plants.

Some of the recent robotic technology for data collection and automated field scouting are illustrated in Fig 1.3. Ladybird is an autonomous multifunction robot for plant monitoring, mapping, classification, and detection of various types of vegetables. SMP S4 is a monitoring robot used for controlling birds and pests (developed by SMP Robotics). Mantis is a flexible robot, which is mostly used for data collection and plant health monitoring. This robot is equipped with RADAR, LiDAR, stereovision, thermal cameras, and panospheric.

TerraSentia is an autonomous farm robot. With its small size, this robot has the ability to move between crop rows smoothly to measure the plant's traits using multiple cameras as well as an array of sensors. Fig 1.4 shows the TerraSentia robot in the agriculture field. TerraSentia transmits the plant's data and location in real-time to its operator’s mobile controller or a computer.

TerraSentia can be trained to identify and detect common plant diseases, and it can also have some additional features like measuring the plant's height, biomass, and leaf area. Thus, the gathered data may help plant breeders to diagnose genetic lineages and to make decisions for protecting the damaged plants. Robotic technology also eliminates the possibility of human error, allowing plants to be measured accurately and non-subjectively

Field scouting robots have several challenges, which help to minimize the possibility of human error. In addition, it allowed an accurate measurement in tight areas. Commercial robots do not concentrate on improving the identification of common diseases for a specific crop. However, they try to build an identification system for the most common diseases for all crops (for commercial purposes, like the TerraSentia robot), which sometimes cannot be accurate. In our thesis, due to the data limitation where most companies do not share the collected data using farm robots from the field. Thus, we focus on improving the classification task of tomato leaf diseases with the available datasets that may help the farmer in the diagnosis process.

To sum up, the choice of the monitoring system depends directly on the application. For wide field monitoring, the use of satellite data is the best choice. However, the small regions can be monitored by the UAV for more precision or even for spraying. In order to detect weed or plant diseases, farm robots can provide more accurate data as well as perform more actions on the field. There are some limitations that the UAV and farm robot may face while monitoring the field to identify plant diseases. When it comes to UAVs, the most common issue is the airflow generated by the flying UAV, which makes the plants swirl. It also complicates the capturing and identification process due to the presence of blurred images. Concerning the farm robot, the movement on uneven soil and the speed of the robot, and other conditions may lack to capture more specific details about the diseases, even the light breeze can be a serious factor.

3. Most common tomato leaf diseases

Producing tomatoes for commercial or even for personal use is not easy. During any particular growing season, several disorders, diseases, insects, and pests can harm the crop, which results in poor quality of the crop. Fig 1.5 shows some leaves diseases. A variety of pathogens (disease-causing organisms) can infect tomatoes and cause disease. Various fungi, bacteria, and viruses are

Fig 1.5. Example of leaves diseases. Bacterial spot, Bacterial speck, Early Blight, Mite Damage. [7]

responsible for many of the most frequent tomato diseases in the world. We describe below the most common diseases of tomatoes that more commonly occur in the field.

3.1. Target Spot

The target spot consists of fungus that may infect the whole part of the tomato plant or even the whole canopy. Fig 1.6. Shows the target spot symptoms on tomato and tomato leaf. Such disease can occur just before and during fruiting. In the early stages, the foliar symptoms look pinpoint-sized on the upper leaf region (Fig 1.6(a)). The spots evolve into smaller sizes and necrotic

(a) (b) (c)

Fig 1.6. Target spot symptoms begin as small brown lesions (a). The lesion enlarges to form light brown lesions with concentric pattern and a yellow halo around it in the transplants (b). Fruit lesions while starting as small lesions can enlarge to form large lesions with cracked centers (c). [8]

(a) (b) (c)

Fig 1.7. Early blight disease: the leaves turn brown and fall off (b), or dead (a). On stems, the lesion appears with a dark border and gray center. The lesion can encompass and girdle steam causing the death of lower canopy leaves (c). [8]

lesions have dark margins and light brown centers. The described symptoms are similar to bacterial spot symptoms. However, the lesions grow in terms of size and get a gray to brown in the center with an outer circular shape. As the lesions get larger, the target spot becomes darker concentric circles (Fig 1.6 (c)) with, sometimes, yellow edges surrounding the lesions. After this stage, the leaves might become infected or can drop definitely. Thus, the earlier symptoms cannot be observed by the farmer, because this lesion starts on the lower leaves of the plant canopy, which makes the early identification difficult.

3.2. Early Blight

Early blight (Alternaria tomatophiland) disease initially occurred on small plants with a brown color, which is mainly caused by a specific fungus. Fig 1.7. Illustrates the early blight lesion. However, the spots of leaves get enlarged with an initial small circle shape. It can be located in the diseased center. At later stages, this tissue around the spots can change its color to yellow. Some conditions may be the reason for leaves dropping like the high temperature and humidity. Then, the lesions get a wider size and may infect the entire tomato with concentric rings. The fungus lives and exists in the soil. At the seedling, it affects the tomato plant.

Fig 1.8. Leaf Mold disease: it is observed on older leaves near the soil where air movement is poor and humidity is comfortable for the fungus. [8]

3.3. Leaf Mold

Fungi are the main reason for leaf mold disease. Older leaves at the lowest canopy part are the most infected, where the airflow is not enough and the humidity is suitable for the fungus. Fig 1.8 shows an example of lead mold disease. The early symptoms appear on the leaf as lighter green spots or yellowish, which become larger with a distinctive yellow later. Moreover, the spots turn gray in the presence of humidity. In a case of a clear severity infection, the leaves fall immediately, and occasionally the tomato gets infected. The fungus survives in the soil and on the tomato plant residuals. The disease development is surely related to the fungus conditions such as high humidity and temperature.

3.4. Septoria Leaf Spot

Septoria lycopersici is the fungus that causes the septoria leaf disease. Once some of the fruits occur, the infection appears on the lowest leaves of the canopy. Fig 1.9 demonstrates the septoria leaf spot on tomato leaves. Several tiny circular dashes (spots with dark edges) and a beige at the center occur on some leaves. At the center of the spots, some small black specks can be observed, which represent the spore-producing bodies. With time, the spot becomes yellowish and at the severe stage, the leaves get completely damaged and fall off the plant. The fungus can survive in high temperatures and humidity conditions. Thus, the fruits are exposed directly to the sunscald due to the high Defoliation. The decaying vegetation, the residual of the damaged plants, the wild hosts related to the tomato, and other conditions may help the fungus to damage the plant.

Fig 1.9. Septoria Leaf Spot: Infection usually occurs on the lower leaves near the soil, after the plant begins to set fruits. Numerous small, circular spots with dark borders surrounding a beige-colored center appear on the first leaves. [8]

3.5. Late Blight

The Phytophthora infestans are one of the most serious fungi, which cause the Late blight tomato disease. This disease usually appears in low temperatures or rainy weather. In addition, this fungus has the ability to affect all the tomato parts. Fig 1.10 illustrates the late blight disease on the

Fig 1.10. Late Blight disease: is especially damaging during low temperatures and wet weather. The fungus can affect the whole plant parts. In the Young leaf, lesions are small and appear as dark, water-soaked spots. [8]

Fig 1.11 Tomato Yellow Leaf Curl disease: Whiteflies may bring the disease in ton the farm area from infected weeds nearby, such as various nightshades and jimsonweed. [8]

tomato leaf. For the new leaf, lesions are tiny and occur with dark dashes. In the water-soaked, the leaf spots become more active and a white mold occurs at the borders of the lesions at the bottom of the leaves. This fungus spreads quickly in the field, especially in rainy weather or heavy wind.

3.6. Tomato Yellow Leaf Curl

Tomato Yellow Leaf Curl does not come from the seed-borne but it is infected by the whiteflies. Whiteflies may transmit the disease into the crop region from the infected nearby weeds (jimsonweed). Fig 1.11 shows an example of the tomato Yellow Leaf Curl. The observed symptoms in tomato plants are the upward leaves curling, yellow color on the leaf borders, tiny leaves than ordinary size, flower drops, and plant stunting. Once the plants are infected by the whiteflies (yellow leaf curl) at early stages, the plant won't be able to form the tomatoes.

Fig 1.12. Tomato mosaic virus disease: Mottled light and dark green on leaves. If plants are infected early, they may appear yellow and stunted overall. Leaves may be curled, malformed, or reduced in size. [8]

3.7 Tomato mosaic virus

This disease is one of the most common plant viruses. It can be known by the dark green and mottled light on tomato leaves. When the plants are infected at early stages, the yellow and stunted appearance occurs on the edge of the leaf. Fig 1.12 shows tomato mosaic virus disease. More precisely, the leaf can look malformed, curled, or reduced in size. The dead leaf tissue can be more visible by the bare eye and with clear cultivars at high temperatures. However, the tomato can ripen unevenly with reduced size and low productivity. Yellowish circles can be formed once the tomato ripens in warm weather. The yielded tomatoes may contain internal browning areas (just under the skin). The infected seedlings may not exhibit symptoms only in a warm environment.

3.8 Tomato Two-spotted spider mite

The only cause of this disease is the mites (tiny pests), which feed on the leaves' undersides. To remove the sap from the leaf, this pest uses its sucking mouthparts, which gives the upper leaf side a speckled appearance. Fig 1.13 illustrates an example of a tomato's two-spotted spider mite disease. Most leaves damaged by the mite may become yellow and dry up. In addition, the plants can lose their vigor and fall off once the infestations are severe. The undersides of affected leaves may have a crusty texture and look yellow or tan. Heavy infestations of this disease produce fine webbing and cover all plant parts. Mites are hard to identify, which is possible only with a hand lens. Severely infected leaves are often unmarketable, which yields in bad tomatoes.

Fig 1.13. Tomato two-spotted spider mite disease: The mites use their sucking mouthparts to remove sap from plants, giving the upper leaf surface a speckled or mottled appearance. [8]

Fig 1.14. Tomato Bacterial spot disease: circular spots surrounded by a yellow halo. The center of the leaf spots often falls out resulting in small holes. [8]

3.9 Tomato Bacterial spot

Signs and symptoms Tomato leaves have small brown spots surrounded by a yellow color. Fig 1.14 demonstrates the tomato bacterial spots in the tomato leaves. The leaf spots center usually falls out and leaves empty holes. Tiny brown circular spots can also appear on the tomatoes. However, the tomato usually has a white halo surrounding the tomato spot. The tomato bacterial spots disease is comfortable with High temperatures, high humidity, overhead irrigation, and rainfall.

4. Contributions

The machine (robot/ satellite/ unmanned aerial vehicle) starts capturing images from the agriculture field. Fig 1.15 represents our proposed communication of the agriculture monitoring system scheme. For further data interpretation, the machine sends these images in real-time (or offline) to the Farmer workstation. The Famer workstation is a computer used for plant interpretation and diagnosis. Consequently, the received images might be blurred (after upscaling) or only small parts of the lesion might be present in an image, which can be considered an LR image. The reason for getting LR images is usually because of the sensor limitations (satellite), capturing altitude (UAV), or the movement on uneven soil (robots). For example, the most common issue for UAVs is the airflow generated by the flying UAV, which makes the plants swirl.

The UAVs fly at a higher altitude to prevent the generated airflow, whereas the received images may lack to contain more detailed information (tiny size of diseases). Consequently, these images are considered as LR. Concerning the farm robot, the movement on uneven soil, speed of the robot, and other conditions may lack to capture more specific details (considered as LR that needs upscaling) about the diseases, even the light breeze can be a serious factor. Thus, the classifier may lack to classify the diseases. In this case, the unique option is to diagnose the disease based on insufficient information or repeat the process in the hopes of obtaining a better view of the lesion. To tackle these difficulties, we proposed to improve the integration of a DL SR algorithm before the disease classifier, at the farmer workstation, to boost the classification accuracy. Furthermore, the SR algorithm provides the classifier with an upscaled image for a better collection of useful information. Thus, the machine will be able to provide farmers with sufficient image samples of diseases to make an accurate diagnosis.

Different studies has been adopted ML in the agriculture field to automate the process of as tomato leaves diagnosis [9]–[11]. However, the obtained accuracy was not sufficient to build a robust diagnosis system. For instance, Using a histogram matching approach to recognize leaves on the tomato PlantVillage dataset, Hlaing *et al.*[12] used a SIFT texture feature SVM-based for image distribution to detect diseases of tomato leaves. The authors obtained just 85.1% classification accuracy.

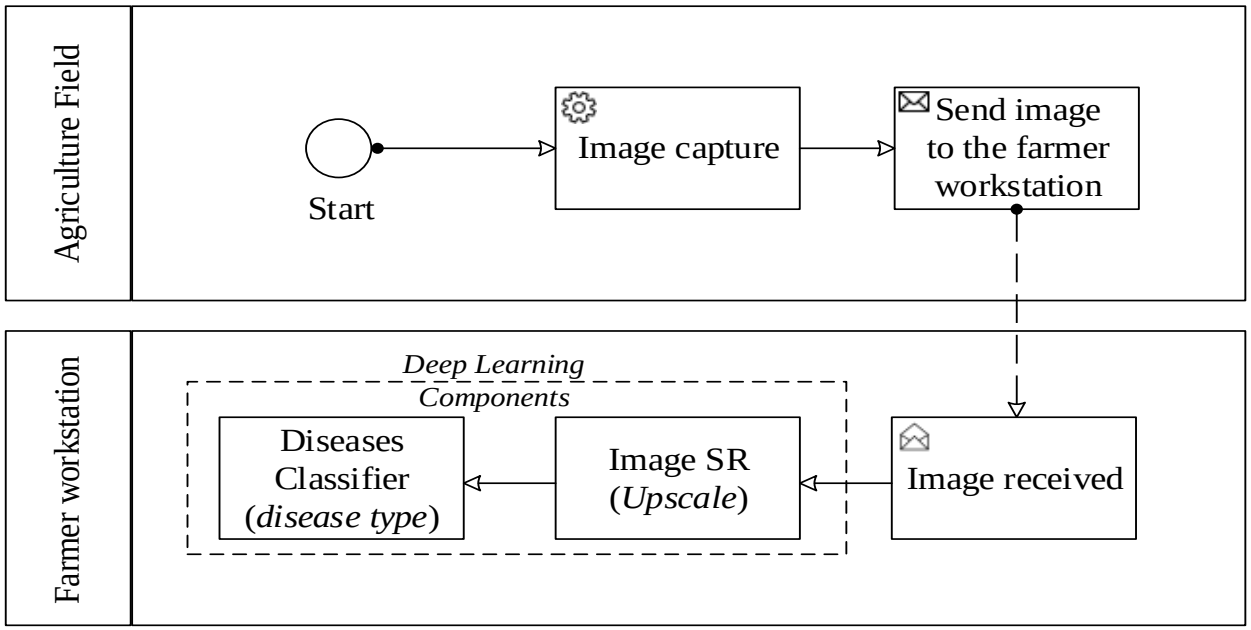


Fig 1.15 overview of our proposed communication scheme for the agriculture monitoring system.

Following the success of DL, many researchers in the agricultural field have adopted DL to overcome several issues of plant diagnosis [13]–[17]. For instance, the authors of [18] used a DL network (AlexNet [19]) to improve the diagnosis of tomato leaves, and they achieved an accuracy of 91.67%. Also, the study of [20] proved that the use of SR (using DL) not only may enhance the quality of the LR images, but can also improve the accuracy of diagnosis. That is, we decided to use DL to improve the tomato leaves diagnosis.

In this thesis, we propose two contributions. In the first contribution, we propose to fine-tune a low-complex classification model named Dense Convolutional Network (DenseNet) on HR plant leaves to detect tomato leaves diseases. DenseNet is a convolutional neural network that uses Dense Blocks to connect all layers (with matching feature-map sizes) directly to each other, resulting in dense connections between layers. This structure helped the model to be compressed in terms of the number of parameters. Extensive results showed that the DenseNet-121 can identify the tomato leaves diseases with a challenging model size as well as a robust classification accuracy. This helped us to choose DenseNet-121 as the low-complexity classifier for further analysis.

In the second contribution, we supposed that the farmer obtained HR images of plant leaves from the field. Because of the small size of plant leaves and other limitations, the obtained HR images can miss some detailed information resulting in blurred LR images of leaves with fewer details. Extensive results using LR images demonstrated that DenseNet-121 lacked to identify the tomato leaves diseases. To overcome this issue, we introduced a novel SR algorithm named Wider-activation for Attention-mechanism based on a Generative Adversarial Network (WAGAN) to improve the classification of the tomato diseases' LR images. The WAGAN consists of three main parts; the generator network, which has the Wider Activation for Residual Channel Attention Block (WARCAB) as the principal block, the discriminator network, and the adversarial loss. To evaluate the potential of the proposed method in plant diseases recognition, we first recovered SR plant diseases images from LR images using the WAGAN.

5. Conclusion

Precision agriculture is a farming management approach. It consists of collecting data from the field, and analysis them to take future management decisions. Different precision agriculture technologies used for plant diagnosis such as SAR, Lidar, UAV, and robots. Also, most farming machines may have some limitations of capturing. Thus, we propose to overcome these limitations using classification and SR to improve the diagnosis.

In the next chapter, we cover the principle background knowledge of ML, DL, image classification, and SR that will be used in our contributions.

Chapter 2
Background knowledge

1. Introduction

In our work, we will perform classification. In computer vision, image classification [2] is a process that can classify an image based on its visual content. In addition, we will use SR to improve the classification. SR [21] aims to produce HR images from LR ones. The need for HR images is common in computer vision applications, where it may improve the performance of pattern recognition and images analysis. This chapter discusses the background knowledge of ML, SR, as well as classification.

2. Background

Before diving into the details of classification, SR literature, and the proposed method, it is helpful to explain some background knowledge needed for a better understanding of this study.

2.1 Machine Learning

ML [22] is a branch of Artificial Intelligence that uses data to teach computers to solve different problems without being explicitly programmed. In the recent decade, ML has improved self-driving cars, image recognition, effective web search, and understanding of human genetic data.

ML is divided into three types namely:

• *Supervised Learning:*

In supervised learning [23], the algorithm uses labeled data to predict a label given some features, the dataset contains training and target attributes, the supervised learning algorithm learns the relationship between training examples and their associated target variables and applies that learned relationship to classify entirely new inputs.

• *Unsupervised Learning:*

In unsupervised learning [24], the algorithm would identify patterns, anomalies, and similarities in the data to make the data more readable and organized.

• *Reinforcement Learning:*

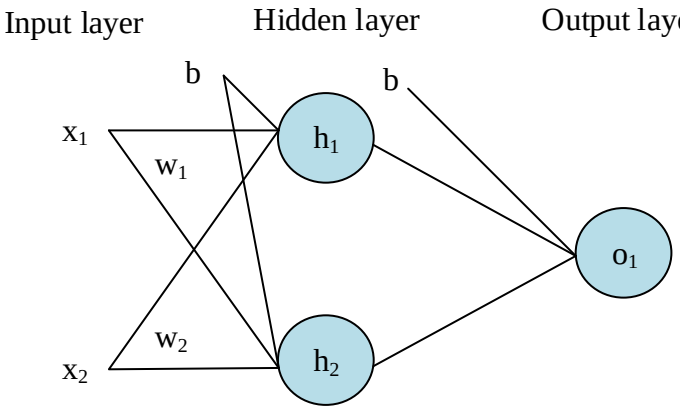


Fig 2.1. Overview of the Neural Network architecture. [26]

In reinforcement learning [25], the algorithm allows software agents and machines to automatically determine the ideal behavior within a specific context, to maximize its performance.

In this thesis, we focus only on supervised learning. Several supervised algorithms have been developed such as Neural Networks, Decision Tree, Random Forest, K-Nearest Neighbor, Linear Regression, Logistic Regression, and Support Vector Machine. Neural networks are among the most common algorithms used in supervised learning.

Artificial Neural Networks: Artificial Neural Networks (ANN) [26] are complex algorithms that are performed in an organized manner to extract labels for a given data set.

NNs are composed of many layers. Fig 2.1 demonstrates the different components of the NN architecture. The first layer (input layer) receives the input data where the NN tries to process and recognize the data. The next layers, (just before the last layer) named hidden layers, consist of other artificial neurons sets. The hidden layers re-process the input layer x_1 *and* x_2 with a neuron function *h* (h_1 *and* h_2). The last layer (the output layer) produces the computation results from hidden layers using the activation function. The NN output is represented as follows:

$$o_1 = \sum_{i=1}^{2} h_i \, , with \; h_i = w_i x_i + b, \tag{2.1}$$

where, o_l, h_i represent the output and the neuron function, respectively. w_i the weight, b the bias, and x_i the network input. Simple Artificial Neural Networks (ANN) are represented by a single input layer, a hidden layer, and an output layer of artificial neurons. Deep Neural Networks (DNN) have a multilayers (many hidden layers) ANN architecture to learn more complex features.

2.2 Deep learning

DL [27] is a sub-field of ML that uses big data to teach computers how to do some operations only humans were capable of before (like generating art pieces, understanding human language, and interpreting complex images). DL uses deep neural networks to learn hierarchical and complex representations from raw data.

2.2.1 Deep learning optimization process

Most DL algorithms try to optimize the cost function in order to update the weights during the learning process. After a feedforward process, the cost function is calculated in order to perform the backpropagation algorithm. The goal of optimization is to let the DL algorithm finds the global minimum of the learning function, which yields in the best case of learning.

a) Cost function

The cost function measures the machine and DL model performance for given data. This function quantifies the error between the predicted values and the expected values, which yielded in a single real number (result). Depending on the result of the cost function, the algorithm updates the weights of the network. The cost function purpose is to be either:

Minimization: the function aims to find the values of the model parameters and returns a small value (as small as possible).

Maximized: the function yields a reward, which aims to find model parameters values and returns a large value (as large as possible).

b) Forward Propagation

The forward pass is the process of propagating values from the first layer (the input layer) through the entire network until the network outputs a value.

c) Gradient Descent

The optimization algorithm Gradient Descent is used to train the network. The optimization algorithm is based on a convex function and updates its parameters iteratively to minimize the cost function to reach the local minimum.

d) Learning Rate

The learning rate is a hyperparameter of the network. Each time the model updates the weights, it controls the model changes (steps) as a function of the cost function. The choice of the learning rate is related to the training process. If the value is too small, the training process is slow and takes a long time, as well as may get stuck. However, as the value is too large, it can result in an unstable training process.

e) Back Propagation

The feedforward algorithm propagates to get the output. Then, it compares the output value with the real value in order to calculate the cost function. The backpropagation algorithm is set to update every weight in the network starting from the output until the input. The backpropagation algorithm tries to improve the actual output and make it closer to the real output. At the same time, it minimizes the error for every layer's weights.

f) Batch Normalization layer

For each mini-batch, this layer standardized the inputs before getting into a hidden layer. In addition, it helps the network to converge faster. Batch Normalization has a normalization step (shifting inputs to zero-mean and unit variance), where it makes the inputs of every trainable layer comparable along with all features. Thus, this layer allows the activation functions (like ReLU, Tanh, and Sigmoid) to prevent the saturation mode.

2.2.2 Convolutional Neural Network

Computer vision [28] is a sub-field of Artificial Intelligence that aims to give computers the ability to extract meaningful information from visual data (digital images, videos, and other visual inputs), and take actions according to the extracted information.

Computer vision also makes sense (interpret) of visual data (images or videos) in the same manner that humans do. The computer vision concept is based on creating an algorithm that can teach computers to process an image at a pixel level, like understanding and interpreting the content of an image or video (detecting objects in an image). Practically, machines seek to retrieve visual information as well as interpret results through classification and other algorithms.

	Actually Positive	Actually Negative
Predicted Positive	True Positives (TP)	False Positive (FP)
Predicted Negative	False Negatives (FN)	True Negatives (TN)

Fig 2.2. Confusion matrix for measuring classification performance. [28]

Image Classification:

Image Classification is the task of associating one or more labels to a given image, this association will help algorithms (such as linear regression, SVM (Support Vector Machine), decision tree, K-Nearest Neighbor (KNN), and Convolutional Neural Network (CNN)) to predict or classify the specific object class in the image. Thus, the image classification method accurately identifies the features in an image.

A common and well-known method for image classification is CNN. CNN is similar to an ANN concept but it contains convolution layers instead of neuron layers.

Confusion matrix:

Fig 2.2 shows an overview of the confusion matrix. The confusion matrix defines the performance of the classification network, which evaluates four classification entities namely True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. The TP means that the classifier identifies the actual positive class as positive, TN indicates the predicted negative class as an actual negative, FP classifies the predicted positive class as a false positive, and FN indicates the actual positive class as a negative predicted.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) consist of several successive convolution layers operations. In past years, CNNs are mostly used in computer vision tasks. CNNs are feedforward hierarchical networks, where several different layers are stacked to form the basic CNNs architecture, including the convolutional layer, activation layer, pooling layer, and fully connected

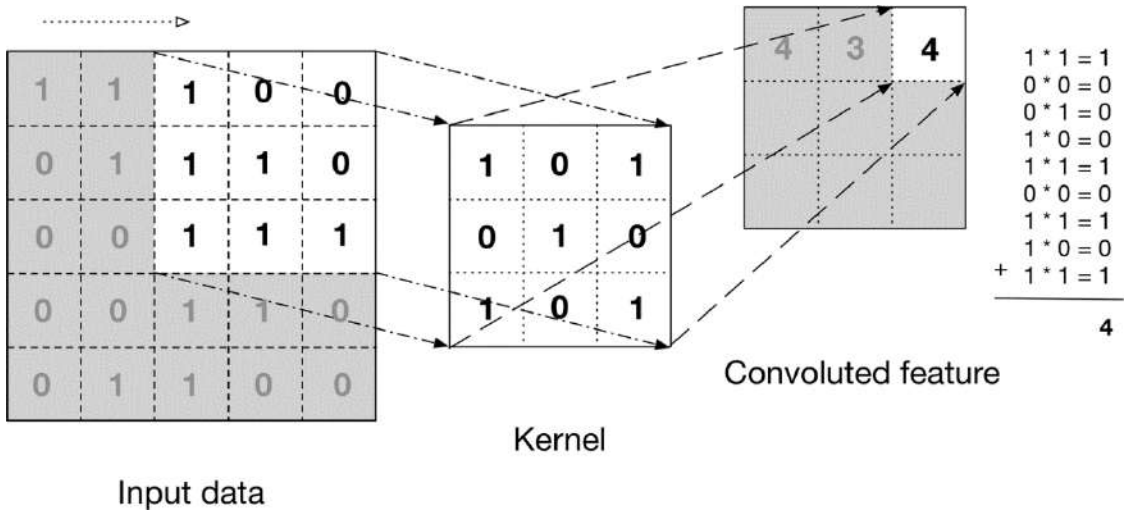


Fig 2.3. Summary of the convolution layer operation. [28]

layer. The convolution kernels in the convolutional layer have the main role to extract valuable features from local image data.

a. Convolution layer: A convolution layer is a filter function. Fig 2.3 demonstrates an example of a convolution operation. The purpose of this operation is to extract the high-level features such as edges, from the input image.

b. Weight kernel: The Weights controlled the data between two neurons in NN. In convolutional layers, the weights are represented as the multiplicative factors of the kernels (filters). In 2D images, the matrix elements (in the convolution filter) are the weights that are trained at every iteration. Initially, the weights start with random values. During the training process, the CNN learns more about the input data at every iteration. Simultaneously, the network updates the values of the weights.

c. Bias: The bias is set to allow the activation function to shift by adding a constant value to the input.

d. Pooling layer: The pooling layer is set to progressively reduce the spatial size of the image. In addition, it is also used to reduce the number of parameters and computations in CNN. The pooling layer operates on each feature map independently.

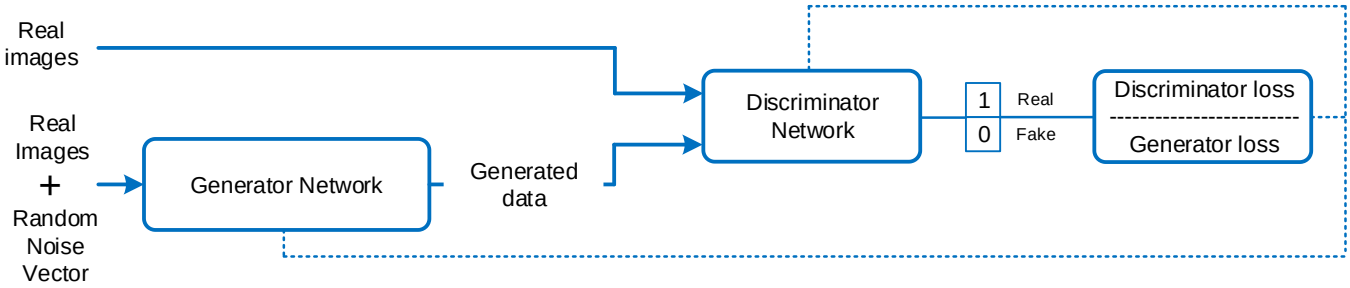


Fig 2.4. Generative Adversarial Network Architecture [29].

e. Fully connected layer: Fully Connected Layers (consist of typical ANNs) represent the last layers in the network. The fully connected layer input is the final Pooling or Convolutional layers output, which is flattened before feeding it into the fully connected layer.

f. Activation layers: The primary role of the activation layers is to introduce nonlinearity. They transform the summed weighted input from the layer into an output value, which is fed to the next hidden layer or as the output. There are many different activation functions, namely: Tanh, ReLU, Sigmoid, Softmax, Leaky ReLU,… etc.

Sigmoid: The sigmoid function is bounded between 0 and 1 and results in output as a probability

*ReLU***:** Rectified Linear Unit allows positive values to pass and turns the negative input values into zero.

Leaky ReLU: Leaky ReLU has a small slope for negative values instead of a flat slope.

2.2.3 Generative Adversarial Network

GANs (Generative Adversarial Networks) [29] are min-max models where two algorithms compete: the generator is the one that generates data, and the discriminator differentiates between fake and real data. The generator's goal is to maximize the error of the discriminator. Otherwise, the discriminator aims to minimize the error. However, the GAN is an iterative algorithm. Once the discriminator minimizes the error, the algorithm finishes training. A GAN can be described as a "Cat and Mouse Game" between a money and a cop counterfeiter, where the counterfeiter (the generator) seeks to deceive the cop (the discriminator), resulting in an unending loop where the two players improve by a highly competitive process. A basic GAN system is depicted in Fig 2.4.

As previously described, it consistently requires a source of "creativity" to produce fake images. This case comes from a random noise vector (seed). However, a database of genuine images is

needed to distinguish between real and fraudulent images. Thus, the discriminator can inform the generator of which generated images are incorrect. Consequently, the objective function of the GAN is:

$$min_G\ max_D\ V\ (D,G) = E_{x\sim P_{data}(x)}[logD(x)] + E_{z\sim P_z(z)}\left[log\left(1 - D\big(G(z)\big)\right)\right] \quad 2.2$$

where V denotes a value of a function for both (the discriminator D and the generator G). The goal of this function is to minimize the generator (G) loss while increasing the discriminator (D) loss. However, the predicted total log-likelihood for real and produced data is the value V. The outputs of the discriminator for real or generated images are (likelihoods) probabilities. Before taking the *log*, the output of the discriminator (of a generated image) is subtracted from 1. Maximizing the generated values may help the discriminator settings to be optimized. It also allows it to learn more to accurately differentiate between real and produced data.

The equation symbols are described as follows:

• P_{data} is the distribution of real data.

• P_z denotes the noise distribution (usually a Gaussian) from the generator.

• *x* and *z* represent the samples from each corresponding space.

• *Ex* and *Ez* are the expected log-likelihood from different outputs of both real as well as fake images.

• *D* is the output (represented by a real number) between 0 and 1, which is the prediction of getting real (1) or fake (0).

Additionally, the output is connected with the network to feed the output errors in order to update the training process of the generator as well as the discriminator. The loss functions are utilized to propagate the errors as gradients. The discriminator update function is:

$$D_{update}(D,G) = \nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[logD\left(x^{(i)}\right)\right] + \left[log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right] \quad 2.3$$

The generator update is represented as:

$$G_{update}(D,G) = \nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m} log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \quad 2.4$$

where *m* denotes the total number of samples in the batch used for updating both networks. θ_d and θ_g are the weights of the network.

GAN's Iterative Steps

The training process steps of GAN are listed below. For every epoch, all real samples are split into batches. The errors from the disparity between the discriminator and generator outputs (as well as their predicted values) are back-propagated to the networks after each iteration. At each batch, the GAN computes the following steps:

1. The batch of real images is firstly fed into both the generator and discriminator.

2. The batch of fake images is generated by combining noise vectors with the real images.

3. The generator output is then fed into the discriminator.

4. The output of the discriminator should be 1 as these images are all real, and 0 for fake. The discriminator calculates the error for every batch.

5. The errors calculated by the discriminator are now injected back into the networks.

6. Steps number 2 and 3 are repeated at every iteration.

7. Once the discriminator output is 1. Thus, the generator had perfectly imitated the real images.

9. The generator tries to do the same process with the rest of the batches in order to continue fooling the discriminator.

In the following chapter, we demonstrated the general idea of the GAN process. In chapter 3, we will describe the use of SR with GAN in detail, we even introduce our proposed network, which is GAN-based.

2.2.4 Transfer learning

Transfer learning [30] is a crucial task in DL and ML. It is used to solve the common problem of applications that have small amounts of training data. The learning process of transfer learning is illustrated in Fig. 2.5. Transfer learning allows the share of knowledge from the source to the target application. In addition, the test and the training data must be identically distributed and independent. Many applications may face some difficulties to enhance the learning accuracy with

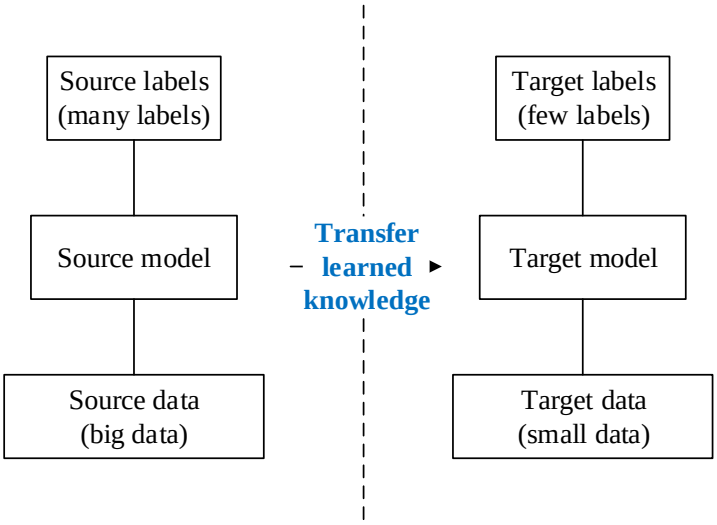


Fig 2.5. Learning process of transfer learning [30]

limited training data, such as medical imaging, agriculture, remote sensing, self-driving cars, etc…. Therefore, transfer learning can help to get significant results in a short period. However, the commonly used technique in transfer learning is Fine-tuning, which means of taking weights of a trained NN and use it as initialization for a new model that have been trained on data from the same domain.

Transfer learning has several advantages and limitations, we can cite them as follows:

Advantages:

• Transfer learning does not require a big dataset.

• It saves training time because the network uses the pre-trained weights of the model, which has been already trained on a big dataset to solve a problem that is similar to the existing problem.

Limitations:

• Transfer learning suffered from negative transfer problems. Where it only performs well when the initial and target problems are similar enough.

• In transfer learning, we cannot remove layers to reduce the number of parameters. Thus, the flexibility does not apply to the network.

2.3 Classification using Machine Learning

2.3.1 Support Vector Machine:

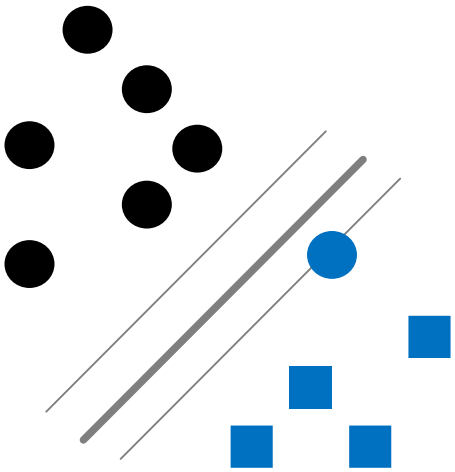


Fig 2.6. Support Vector Machine, two classes that are linear [31].

The Support Vector Machine (SVM) [31] is a supervised ML technique that represents different classes in a multidimensional space hyperplane. The SVM tries to find a maximum marginal hyperplane by dividing the datasets into classes. It creates a hyperplane (or a series of hyperplanes) in a high-dimensional space. Thus, the SVM achieves good separation between the two classes, when the hyperplane has the largest distance from the nearest training data samples of any class. The kernel function is employed to determine the algorithm's efficiency. This model can be used to classify and predict data. Fig 2.6 illustrates the SVM algorithm separating the data into two classes using a linear separating hyperplane.

The black dots in Fig. 2.6 represents one class, while the blue squares represent the other class. These two classes are separated by a hyperplane in the middle. It's critical to identify the best hyperplane for this separation. Depending on the dataset and application situation, SVM usually employs kernels, and even multiple kernels can be used. The kernel is a strategy for solving nonlinear problems that are based on a linear classifier. An example of SVM is shown in Fig 2.7.

Below, we mention the different advantages and limitations that the SVM may have.

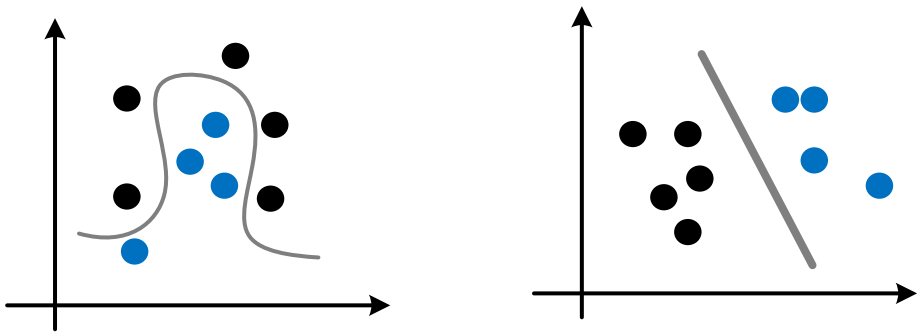


Fig 2.7. (From left) Non-linear separable data and separable data. [31]

Advantages:

• The SVM has the ability to handle both categorical and continuous variables.

• SVM may be employed for both linear and non-linear problems.

• The SVM has different powerful kernels such as the linear kernel, polynomial kernel, and Gaussian kernel.

• It has a simple way to identify the correct results.

• It works accurately even with noisy training images.

• SVM does not suffer from the problem of overfitting.

Limitations:

• Time-consuming and the training process is slow.

• It can perform well only in binary classification tasks.

2.3.2 K-Nearest Neighbor:

K-Nearest Neighbor (K-NN) [32] is a non-parametric algorithm. The algorithm classifies unknown data points by identifying the most frequent class among the k-closest examples as well as the distance between feature vectors.

K-Nearest Neighbors is a supervised learning algorithm that is used to tackle regression and classification issues. Class membership is the output in the case of a classification problem. As a result, an object's classification is determined by the votes of its neighbors. If k = 1, for example, the object is allocated to the class that is closest to the neighbor. If the problem is a regression problem, the output is the object's property value. The object's value is the average of its k closest neighbors' values.

The KNN algorithm has many advantages and limitations. We ordered them as follows:

Advantages:

• KNN has a simple implementation.

• It does not require training for small datasets.

Limitations:

• The quality of the data has a direct impact on the resulting accuracy.

• The testing process is time-consuming and slow.

• It is expensive in terms of memory allocation, which requires high memory to store all of the training data.

2.3.3 Random Forest Algorithm

The Random Forests Algorithm [33] is a supervised algorithm that classifies data by constructing a set of classifiers to improve classification accuracy. Random Forest approaches are used to analyze a data set by building the trees and joining the individuals to predict the class label. Classifying a large amount of dataset with a single classifier is not recommended and even can result in lower accuracy.

The Random Forest algorithm does not require cross-validation, and it prevents overfitting. To build many classifiers, the Random forest utilized Bootstrapping and Adaboost algorithms.

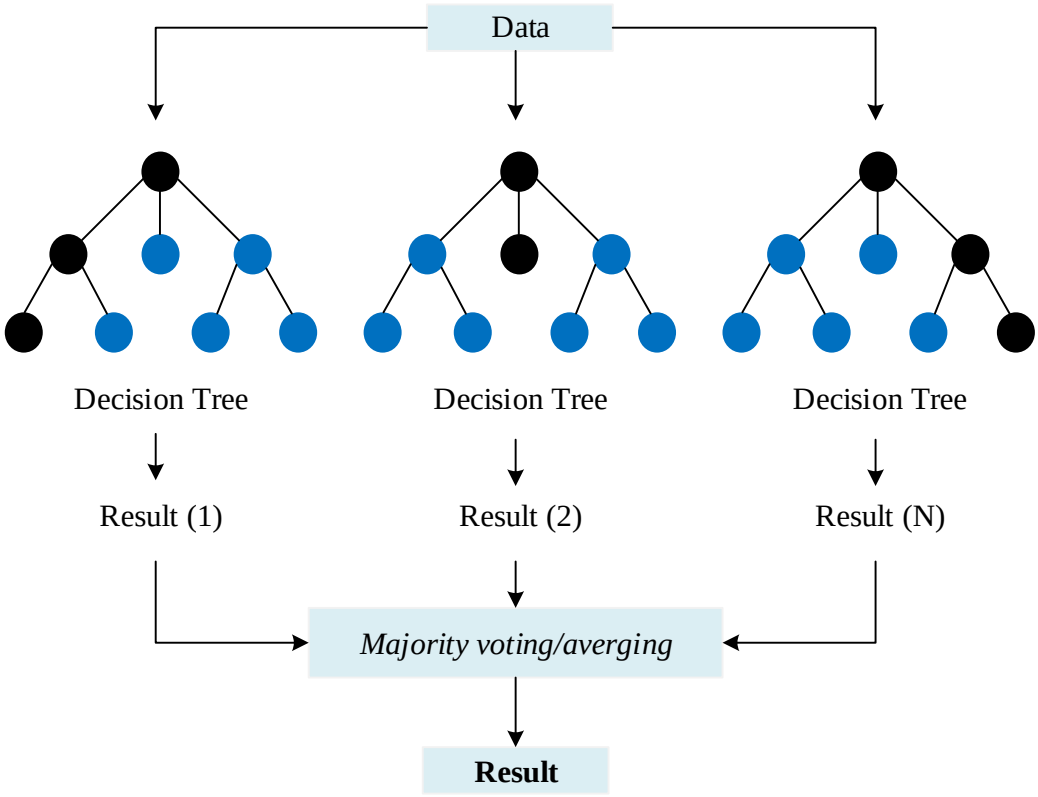


Fig 2.8. Overview of the Random Forest Algorithm. [33]

Fig 2.8 describes the functionality of the Random Forest algorithm. The Random Forest Algorithm is summarized in the following steps:

• Let *N* be the number of training data instances in the samples. Take *M* as the number of attributes in a given input dataset.

• *m* (where *m* is smaller than *M*) represents the number of parameters of the input, which determines the following attribute to be chosen at every tree node.

• The samples of training are taken, and a tree is built for every sample with replacement.

• For tree nodes, it arbitrarily selects *m* attributes in that specific node.

• The best split is computed based on the *m* sample dataset input attributes.

• Every tree is growing without pruning.

As the Random Forest algorithm has several advantages, it also has some limitations. First, we start with the advantages:

• The algorithm operates even with missing data and keeps accuracy in case large parts of the data are missing.

• Random Forest may be employed for both classification and regression applications.

• It can operate with non-parametric training data.

• This algorithm is simple to implement.

For the Limitations:

• The creation of a Random Forest algorithm needs several trees and combines their outputs, which makes it complex in terms of structure.

• The training phase is time-consuming and slow.

• The control of the algorithm is limited.

2.4 Deep learning classification using CNN

For image classification, CNNs [34] are a widely used approach. Many different CNNs architectures have been introduced and even different designs, which aim to improve the

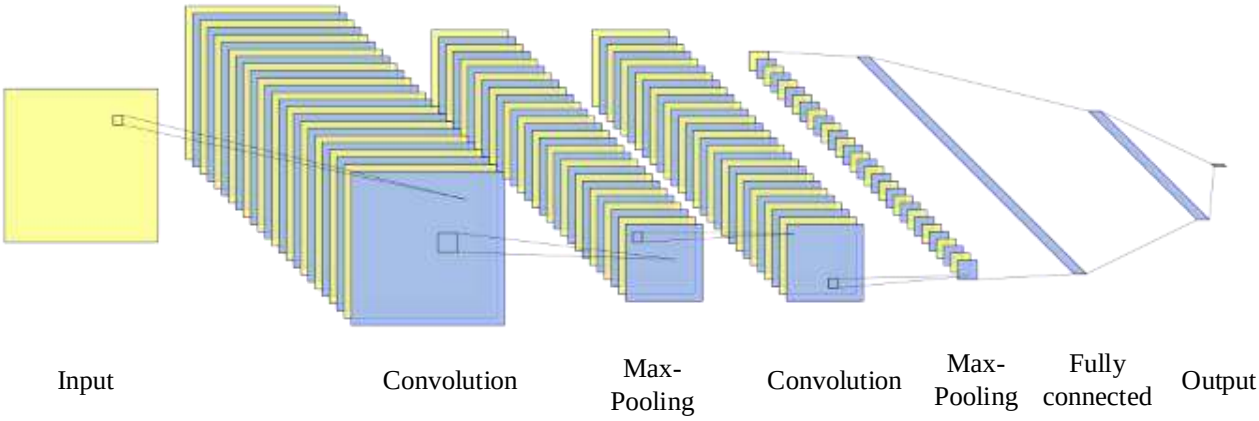


Fig 2.9. Convolutional Neural Network Architecture [34]

classification accuracy. Fig 2.9 illustrates an example of a CNN architecture. All CNNs have the same fundamental structure. Convolutional, pooling, fully connected layers, and output layers are the four types of layers that are mostly used in CNNs design. The convolutional layers have many filters (kernels), each one is used for calculating the visual features representations. On the other side, a pooling layer is set to decrease the representation's (feature map) resolution. In most architecture, every convolutional layer is followed by a pooling layer. More precisely, the first convolutional layer is set for low-level features, where it detects edges in an image. Next, the following convolutional layers are utilized for extracting more detailed (deep) features. Following that, the fully connected layers are mostly placed after the two first types of layers. The fully connected layers are used to exhibit a communication channel between neurons. The output layer is placed at the end of the CNN. This layer is used to establish the class label winner.

The CNN method has many advantages and limitations, we mention some of them:

Advantages

• CNN is the most accurate architecture (compared to neurons), widely employed, and a powerful model for image classification applications.

• This approach gives the possibility to share pre-trained weights.

• CNNs do not require feature extraction.

Limitations

• CNNs require a large number of training samples.

• The training is time-consuming and slow.

Many known classification CNN models are largely used in literature. such as LeNet, AlexNet, Visual Geometry Group Network (VGG-Net), Residual Neural Network (ResNet), DenseNet, and GoogleLeNet. As an example, we will present VGG-Net and ResNet here, and we will use them for comparison in the next chapter. The DenseNet model is described in detail in Chapter 3.

2.4.1 Visual Geometry Group network

VGG-Net [35] is a CNN model that was the winner in the ILSVRC-2014 challenge. A 224×224 RGB image is used as the input to the VGG-based CNN. The normalization is applied as a preprocessing layer. The normalization consists of subtracting the mean image values. These values are derived for the entire ImageNet training dataset, from the RGB images.

After preprocessing, the input images are transmitted to a stack of convolution layers that are used to process the training images with many conventional kernels. Fig 2.10 shows the structure of the VGGNet. VGG16 architecture consists of 13 convolutional layers and 3 fully connected layers. Contrary to other network designs, VGG uses a fixed size of the convolutional kernels (3×3), but with a different number of kernels.

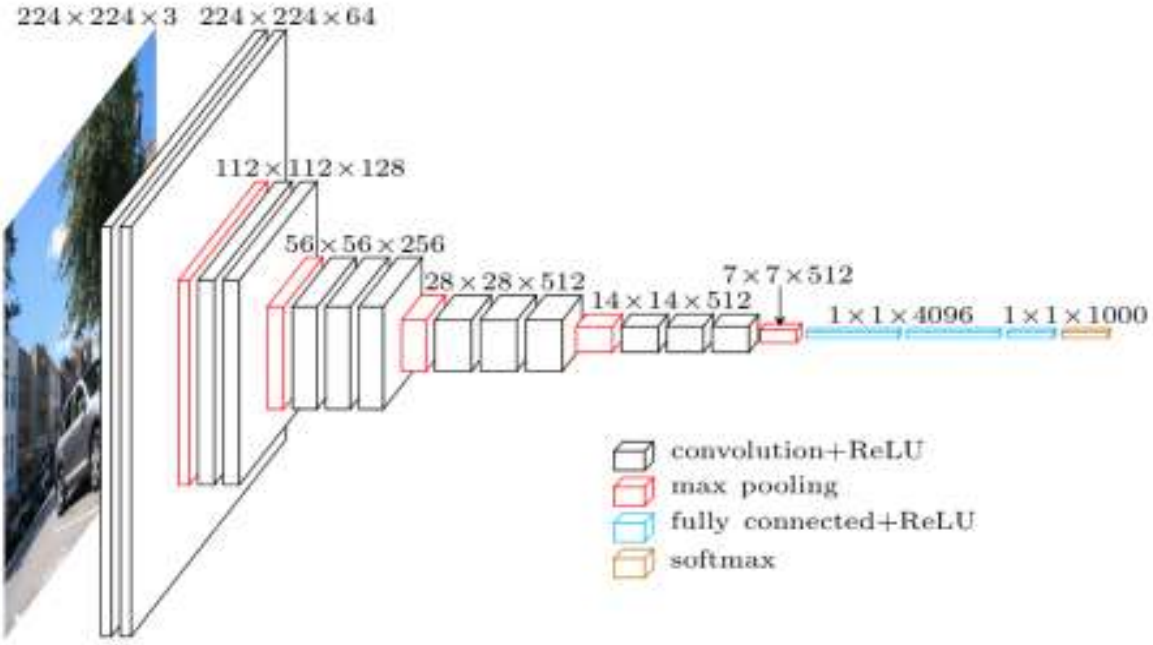


Fig 2.10. VGGNet architecture overview. [35]

The second version of VGG (VGG19) has 19 layers, with 16 convolutional, 3 fully connected, and the same 5 pooling layers as VGG16. VGGNet has a fully connected layer with 4096 neurons to convert it into a 1000 vector of classes as an output. The details of the VGGNet architecture are organized as follows

- The first two convolutional layers have 3×3 kernel size with 64 filters, which yields 224×224×64 shape. The 3×3 filters are set with a stride of 1.
- To reduce the feature map size from 224×224×64 to 112×112×64, a max-pool pooling layer with a 2×2 size and stride of 2 is used.
- The next two convolution layers have 128 filters to output a dimension of 112×112×128.
- The output shape is then shrunk to 56×56×128 after passing through a pooling layer.
- Two convolution layers that take 56×56×128 with 256 filters followed by a pooling layer that downsizes it to 28×28×256.
- Another max-pool layer is placed between two convolution blocks, each one of them has 3 layers.
- The shape of 7×7×512 is then flattened to pass through a fully connected layer of 4096 neurons and a Softmax function to output the 1000 labels.

2.4.2 Residual Neural Network

The only difference between the ResNet and the other traditional network designs is the identity connections. Fig 2.11 illustrates the main difference between the traditional designs and the residual one. According to the research of [36], one or multiple layers can be summarized as a non-linear mapping function H(x) of input data *x*.

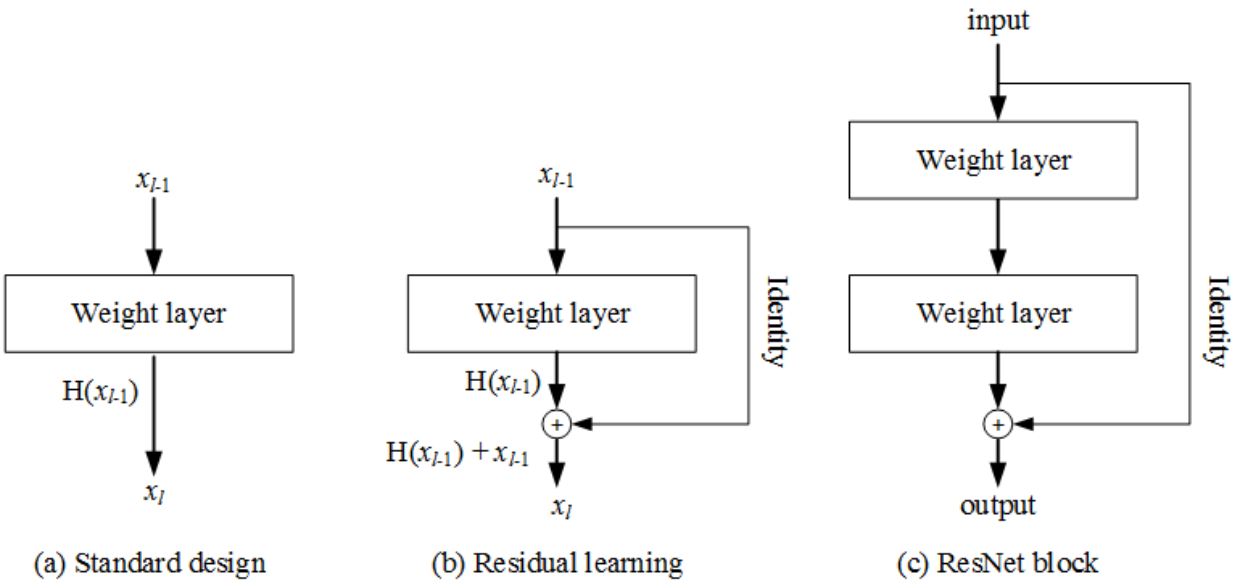


Fig 2.11 overview of the residual learning versus the standard design [36]

Let's l be an index of every layer in the network. Considering x_l as a layer output, which is represented by $x_l = H(x_{l-1})$. During the network feedforward process, residual learning is the setting of identity/skip connections (or path) to the network by bypassing one weight or multiple layers. In order to accelerate the training and prevent the vanishing gradient, these connections allowed a smooth process for the gradient to easily backpropagate through them. Assuming that only the identity connection skipped only one layer, the layer l output is then described by:

$$x_l = H(x_{l-1}) + x_{l-1} \tag{2.5}$$

Note that the dimensions of x_l and x_{l-1} should be the same to perform the addition operation. In terms of parameters, the identity connections do not increase the number of parameters or even the complexity of the network. The main idea of residual learning is to save the residual values (approaches zero). Where the residual values are learned by passing them through the identity mapping. As a result, all the layers produce more detailed feature maps. More formally, these feature maps contain both high frequency and residual values, which are the main necessary features for more accurate image classification.

ResNet achieved a remarkable generalization performance in the recognition tasks and obtained first place in the ImageNet 2015 competitions. ResNet has many different architectures with the same concept of residual learning but with a different number of layers, namely ResNet-18,

ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, ResNet-1202 etc. Fig 2.12 illustrates an example of the ResNet-34 design. The ResNet architecture obtained significant and competitive results as well as outperformed other networks, which used stacked convolution layers design. The identity paths allowed an efficient optimization and benefited from being deeper in the network architecture.

Huang *et al.* [37] introduced a Densely connected CNN (DenseNet) architecture. Instead of using the residual connections [36], all layers are connected directly with each other in the network to ensure the maximum flow of information between layers. Feature maps of all previous layers are fed as inputs for each layer, and their feature maps are then fed as inputs into all next layers. DenseNet improved the feedforward process compared to [36], especially the problem vanishing gradient, and remarkably reduced the number of parameters.

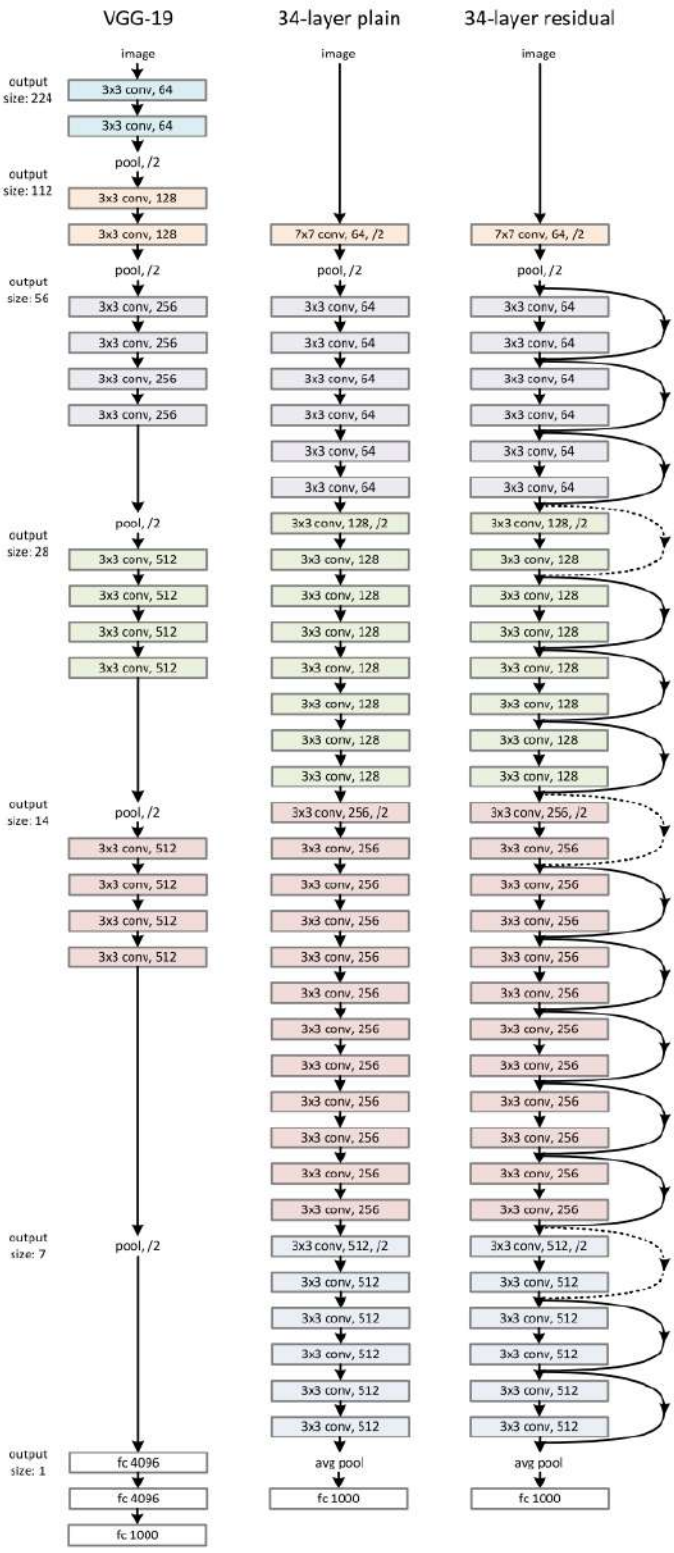


Fig 2.12. Overview of the ResNet design [36].

2.5 Image Analysis

Image analysis [38] is the extraction of meaningful information from images; mainly from digital images using image processing techniques. Image analysis tasks can be simple as reading bar-coded tags or sophisticated applications such as denoising a person's face from a noisy face image. Image analysis covers several techniques such as Image segmentation, Image denoising, Image SR, etc…

Super-Resolution

To increase the image size, many interpolation approaches have been studied in detail. However, the loss of information is unrecoverable, and there is no specific algorithm that can be used to recover the lost pixel values. While a distribution function on the image may deliver various estimations of the missing values. A single frame interpolation method is insufficient to restore the lost high-frequency pixels that are lost during the downsampling procedure. Because the prediction of the specific lost pixels is essential, some information from the same image should be acquired and fused. Even so, the exploration of image restoration and interpolation algorithms helps SR to recover some details that may improve the image quality.

The basic idea of SR is to utilize one or a sequence of LR images of a specific scene to produce a higher spatial resolution image with more valuable information presented as high-frequency pixels in SR images. Because of the limitations of imaging equipment, it's common to have a natural loss of spatial resolution and some form of blur and noise effects presented while recording. Some of these blur regions can be negligible after recovering some high-frequency information using some methods like nearest-neighbor interpolation, Bicubic, bilinear,... etc.

2.5.1 Image Nearest neighbor interpolation

One of the easiest methods to interpolate pixels from the LR images is to use nearest-neighbor interpolation [39]. Each interpolated output pixel in this method is referenced to the nearest sample pixel in the input image. Fig 2.13 demonstrates the process of the nearest neighbor interpolation algorithm. In Fig 2.13, known pixel values are presented as *(i,j)*, *(i+1,j)*, *(i,j+1)*, *(i+1,j+1)*, and an unknown pixel is denoted by *P* that will be interpolated.

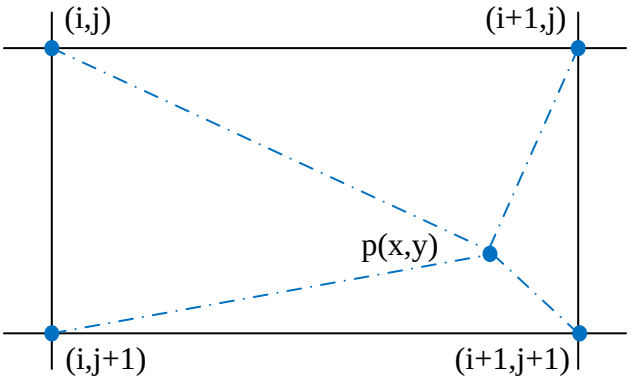


Fig 2.13. Overview of the nearest neighbor interpolation SR. [39]

First, the algorithm calculates the distance between each point of the known pixel and the unknown point *p(x,y)*. Then, it chooses the pixel values that are near to each other. More precisely, the minimum distance is selected to an unknown point as expressed below:

$$D_{min} = min\{D[(x,y),(i,j)], D[(x,y),(i,j+1)], D[(x,y),(i+1,j)], D[(x;y);(i+1,j+1)]\} \quad 2.6$$

where D is the distance between points. Consequently, the value of the unknown point *p(x,y)* is $(i+1, j+1)$. This interpolation approach used a limited spatial kernel in order to estimate neighbor pixels. The nearest-neighbor interpolation kernel is written as:

$$y(x) = \begin{cases} 0 & |x| > 0 \\ 1 & x < 0 \end{cases} \quad 2.7$$

Fig 2.14 shows the resultant SR image of the interpolation method compared to reference HR. The SR image may contain a lot of artifacts, and the quality is poor. The blurring and aliasing usually appear after applying the kernel. In terms of quality, the improvement is quite modest. Compared to the other approaches, nearest-neighbor interpolation is less efficient.

Fig.2.14. Original picture (HR) on the left, and nearest neighbor interpolation on right.

Single Image SR (SISR) and multi-frame SR are the main two SR approaches. SISR received a single LR image as an input, and the SR algorithm generated the SR image by recovering the missing pixels. However, multi-frame SR takes several LR images as input and generates multi SR images. There are numerous methods to predict those missing pixels. We can classify SISR algorithms into three types: Interpolation Methods, ML-SISR based, and DL SISR methods.

a) **Interpolation Methods:** Interpolation methods are commonly based on arbitrary functions, such as nearest-neighbor, bilinear, and Bicubic interpolations.

Example: Bicubic Interpolation is commonly used in most SR applications. Bicubic takes the closest 4x4 neighborhood of known pixels (a total of 16 pixels). Since these pixels are located at different distances from the unknown pixel, closer pixels get the highest weight in the calculation. Bicubic generates sharper and smoother images than the other interpolation methods, and possibly it has a reasonable combination in terms of quality and processing time.

b) ML-SISR based methods: Machine-learning-based SR techniques try to overcome the weakness of interpolation-based. The SR algorithm predicted the missing pixels using a learning dictionary, for instance: Neighbor embedding and sparse coding algorithms.

c) Deep learning SISR methods: Most of these algorithms are example-based approaches that use prior information from LR and HR. The most common SR models are based on deep CNN, such as SR Convolutional Neural Networks (SRCNN).

2.5.2 Machine Learning SISR methods

a) Sparse Dictionary learning

The main idea of the SR dictionary learning is to create a set of atoms, which are represented as pairs of LR patches and their HR equivalent [40]. In the sparse dictionary learning method, various patches are extracted from the training-set images. In the beginning, the algorithm receives an LR image, then it just splits it into LR patches to find a representation using the LR dictionary atoms. Finally, the algorithm reconstructs the SR images using the matching HR atoms. In the last decades, various SR methods for constructing dictionaries have been introduced.

To illustrate the idea of sparse dictionary learning, the authors [40] proposed to utilize a full sparse dictionary of LR as well as HR patches (around 100, 000) to generate the SR images. A given LR patch y_i is represented as a sparse linear combination of LR dictionary elements αD_l and an HR patch xi. More formally, xi is reconstructed using the corresponding HR elements $D_h\alpha$.

$$y_i = D_l\alpha_i \tag{2.8}$$

$$x_i = D_h\alpha_i \tag{2.9}$$

For every patch y_i in an image, an optimal α_i should be found to minimize this equation:

$$\mathrm{T} = \underset{\alpha_i}{\operatorname{argmin}} ||y_i - D_l\alpha_i||_2^2 + \lambda||\alpha_i||_1 \tag{2.10}$$

where, λ is used to balance the sparsity of α_i.

In addition, the authors added a constraint on the top and left border region of the recovered HR patches *(PDhαi)*. That is, to ensure its compatibility with the previously recovered images overlapping the region,

$$S = ||w - D_h\alpha_i||_2^2 \tag{2.11}$$

Put all together with 1.8, we get:

$$\mathrm{Q} = \underset{\alpha_i}{\operatorname{argmin}} ||\widetilde{y_i} - \widetilde{D_l}\alpha_i||_2^2 + \lambda||\alpha_i||_1 \tag{2.12}$$

The constraints are included in $\widetilde{y_i}$ and $\widetilde{D}$. In the optimization phase, the L_1 norm constraint, on the sparse code α, is used to ensure the sparsity. Also, a global reconstruction constraint based on the

back-projection method is used to ensure that the HR image is consistent with the model. To create more compact dictionaries from the extracted patch pairs, the authors used an iterative sparse coding optimization method.

To conduct SR, the authors suggested using two learning dictionaries at the same time, one in the LR space and the other one in the HR [40]. They extracted the patches from LR (Bicubic) and HR images in order to learn the dictionaries. Fig 2.15 shows an example of the position-patch principle is used to construct a dictionary of connected LR and HR patches. Also, they respected the restriction of the LR and HR patches linearity reconstruction, which are recovered from their particular dictionaries using the same sparse vector. Thus, this method alternates between the initial dictionary (with a fixed sparse code) as well as the searching for sparse codes while optimizing the process (with a fixed joint dictionary).

b) Sparse coding learning dictionary

A dictionary database of paired LR and HR patches is shown in Fig 2.16. The LR patches are projected onto a low-dimensional subspace that captures the majority of the patch's information [41]. Using the Principal Component Analysis (PCA) algorithm, the low-dimensional Eigen-space of every patch position is calculated to find the training dictionary's set of the arranged LR patches. Using a linear combination of the arranged HR patches dictionary, the HR patches are then reconstructed. More formally, It is important to note that every patch feeds to a particular iris

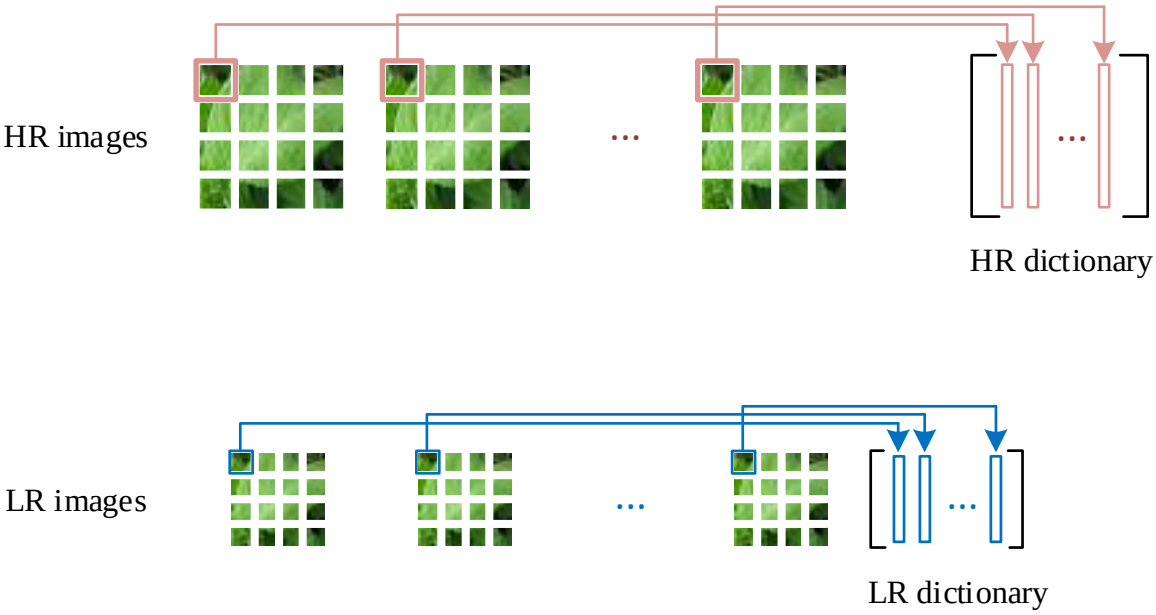


Fig 2.15. The position-patch principle is used to build a dictionary of connected LR and HR patches. [40]

Fig 2.16. Example of a learning dictionary. [40]

region, and it has its special set of coupled dictionaries. In this way, every input patch is ensured to have its particular reconstruction weights, which is resulting in an optimized local solution. Fig 2.17 demonstrates an example of a Super-resolved plant leaf image using the sparse coding dictionary compared to the Bicubic one. For better recovery of the local texture features, it's preferable to reconstruct every patch individually with its ideal weights.

Fig 2.17 Qualitative results of a Bicubic and Sparse learning. Bicubic (left) vs Sparse learning

c) Extensions and improvements

Many studies have proposed to improve the sparse coding learning dictionary method. For instance, the authors of [42] introduced sub-dictionary learning for both SR as well as deblurring. Instead of using a single learning dictionary, they applied the K-means algorithm to split the high-pass filtered image space. Then, the dictionary learned each cluster using an L1 constraint on the sparse code, which is resolved with an iterative shrinkage algorithm. In another method [43], the authors have fine-tuned the dictionary with the HR images. Nevertheless, they encoded the LR sparse representation with the K-SVD algorithm and reconstructed the HR using a pseudo inverse. Instead of overcoming the LASSO optimization issue with constraints on the sparse code, the sparse coding is optimized with the Orthogonal Matching Pursuit (OMP) algorithm. In addition, they set the dictionary with the size of 1000 atoms for the LR representation.

The sparse vector inference is introduced by implementing it with a NN model [44]. More precisely, the method additionally had a forward pass rather than just using search optimization. That is, the NN offered a novel sparse code coupling, which is presented in a simple linear relation between the HR and LR sparse representations. So, it's preferable to use a coupled dictionary of LR patches and HR sparse codes rather than using the same sparse representation as previous works.

2.5.3 Deep learning SISR methods

Commonly, the LR image *lr* is expressed as a degradation output as follows

$$Ir = D(hr; \alpha), \tag{2.13}$$

where D is a function of degradation, hr denotes the corresponding HR image. The α represents the degradation parameters (the scaling factor).

Usually, the process of degradation (i.e., D and α) is unidentified. Besides, only LR images are available for all SISR. In the present case, the SR algorithm is required to generate an approximation (hr') of the ground truth HR image from the LR image. We can express this operation as

$$hr' = F(Ir; \theta), \tag{2.14}$$

where F denotes the SR algorithm, and θ is the parameters of F.

Although the process of degradation is unknown and can be impacted by several factors (e.g., anisotropic degradations, speckle noise, sensor noise, and compression artifacts). Most research work modeled the degradation mapping directly as a single downsampling operation:

$$D(hr; \alpha) = (hr) \downarrow s, \{s\} \subset \alpha, \tag{2.15}$$

where $\downarrow s$ represents the process of downsampling with the scaling factor s. Generally, most research used Bicubic interpolation with antialiasing as a downsampling operation. However, other works expressed the degradation as a combination of several operations:

$$D(hr; \delta) = (hr \otimes \kappa) \downarrow s + n\varsigma, \{\kappa, s, \varsigma\} \subset \alpha, \tag{2.16}$$

where $hr \otimes \kappa$ denotes the convolution operation between a blur kernel κ and the HR. In addition, $n\varsigma$ is an additive white Gaussian noise with a standard deviation ς. Consciously, the SR objective is given by

$$\theta' = arg\ min\ \theta L(hr', hr) + \lambda \Phi(\theta), \tag{2.17}$$

where $L(hr', hr)$ denotes the loss function between the ground truth image $hr$and the generated SR image hr'. The $\Phi(\theta)$ represents the regularization term, and λ is the parameter of tradeoff.

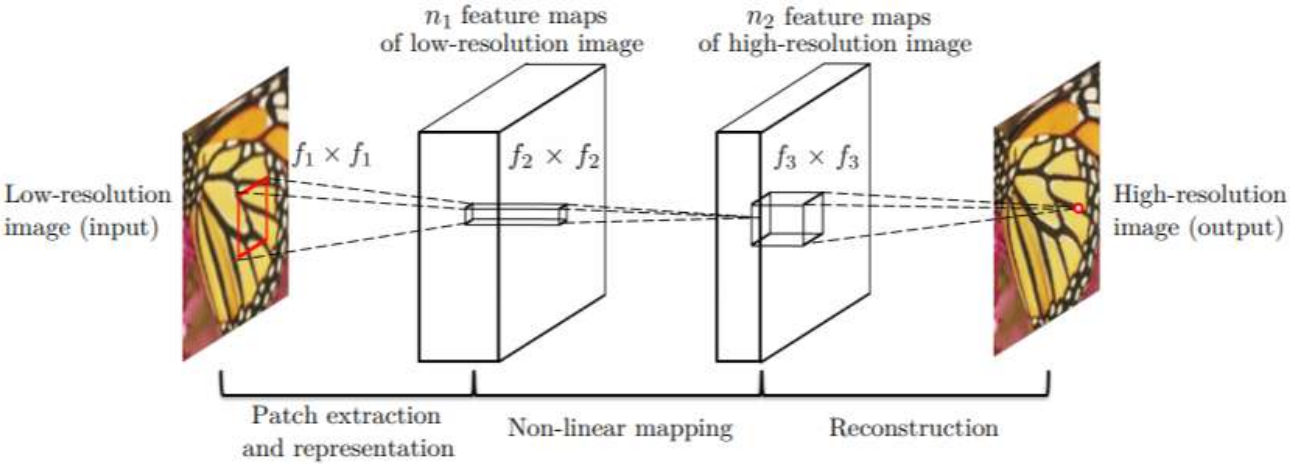


Fig 2.18. SRCNN architecture [44].

a) Super-Resolution Convolutional Neural Networks

The cornerstone of the convolutional neural network for SISR is called Learning a Deep Convolutional Network for Image SR (SRCNN), which is introduced by C. Dong *et al*, in 2014 [44]. The author applied a CNN for an end-to-end mapping between the LR and the HR. Instead of taking each feature component one by one in the dictionaries like traditional sparse coding methods, SRCNN optimized all layers at one time. As a result of this process, faster and better image quality is obtained with the SRCNN. Fig 2.18 demonstrates the three main steps of the SRCNN image enhancement process.

Patch extraction: The first layer is defined as a function set F1 as it's shown in equation 1.19. These functions are used to extract image patches by convolving the image.

$$F_1(Y) = max(0, W_1 * Y + B_1), \tag{2.18}$$

where, Y represents the input LR image, $W1$ are filters and $B1$ are biases. The size of $W1$ is $c \times f1 \times f1 \times n1$ where c represents the number of image channels, $f1$ is the size of the filter, and n1 is the number of convolution filters. In addition, the ReLU is applied to the filter output.

End-to-end non-linear mapping

In this process, one high-dimensional vector is mapped onto another vector. Each non-linearly mapped vector represents an HR image patch. Equation 1.20 defines the second layer as:

$$F_2(Y) = max(0, W_2 * F_1Y + B_2); \tag{2.19}$$

where $B2$ represents the n2-dimensional vector, and $W2$ is filters with $n1 \times 1 \times 1 \times n2$ size. Each $n2$ sized vector represents an HR image patch. Then these vectors are used to reconstruct the final HR image.

Reconstruction: In this layer, the generated overlapping HR patches are averaged to create the final HR image. The construction step is defined with a convolution layer which is presented as:

$$F(Y) = W_3 * F_2Y + B_3; \quad (2.20)$$

where $B3$ is a vector with c-dimensional, and the size of $W3$ is $n2 \times f3 \times f3 \times c$.

The process of training the SR network is similar to supervised methods. During the training phase, the sub-images are randomly clipped from the training images. By "sub-images," these samples are considered as small images instead of patches, in the sense that "patches" overlap and necessitate some post-processing averaging, whereas "sub-images" do not. Once the network is trained, a quality evaluation should be conducted to verify the resulting images.

b) Generative adversarial network for SISR (SRGAN):

The first model to obtain visually realistic results for 4× SISR was SRGAN [45]. Instead of using the MSE per-pixel error, the authors used an adversarial approach, an optimized content loss, and an adversarial loss. This model is capable of generating incredibly realistic images, unlike the literature approaches. Moreover, its performance, in terms of PSNR and SSIM values, indicated that it did not outperform the state-of-the-art SR models. The generator and discriminator architectures are shown in Fig 2.19. The ResNet architecture [36] inspired the generator, which has residual blocks and skip-connections. In the end, the discriminator is a CNN with fully connected layers.

Loss functions

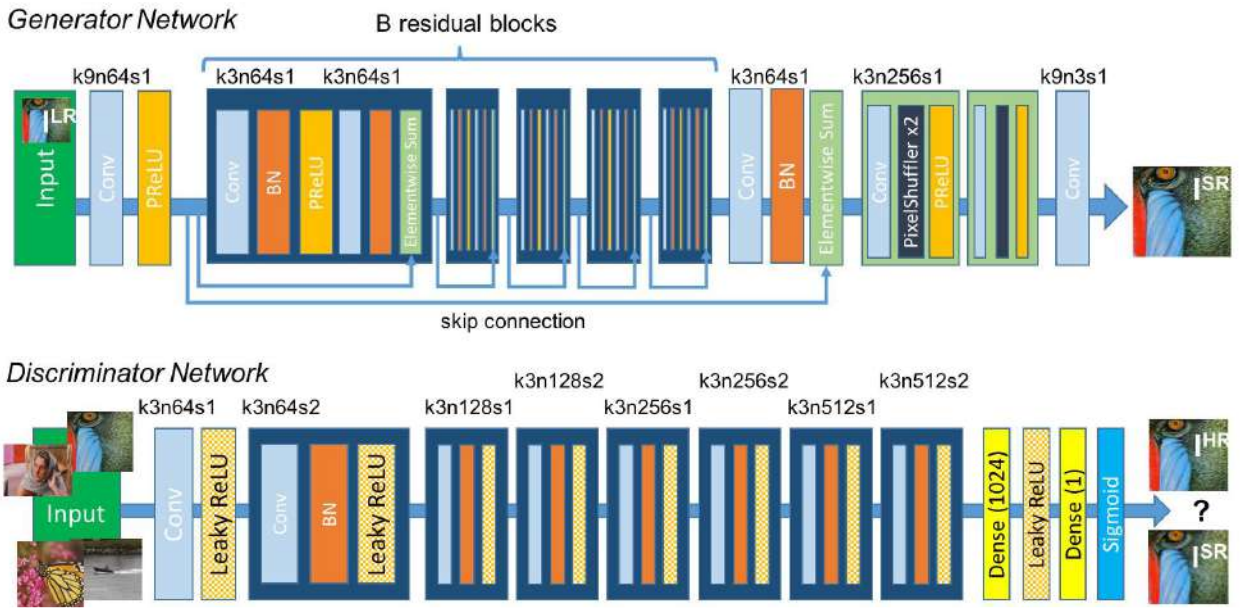


Fig. 2.19 SRGAN Generator and discriminator architecture [45].

Initially, the loss l^{SR} was based on the MSE loss function (Dong *et al.* [44]). The lack of this function is leading the model to obtain too smoothed images, which means that high frequency information is lost. Later, the l^{SR} loss function was upgraded to a more sophisticated version, where it took the perceptual similarity into account. Ledig *et al.* [45] developed the l^{SR} perceptual loss model and used it to evaluate the SRGAN model in the learning phase. Because the SRGAN model employed an adversarial architecture, which consists of two parts: a content loss I_{VGG}^{SR} and an adversarial loss $10^{-3}I_{Gen}^{SR}$. The perceptual loss function is calculated as follows

$$I^{SR} = \underbrace{I_{VGG}^{SR}} + \underbrace{10^{-3}I_{Gen}^{SR}} \tag{2.21}$$

b.1 Content Loss (VGG loss)

The content loss is also referred to as VGG loss I_{VGG}^{SR}. It is based on the features extraction part of the VGG19 network that was proposed in [35]. Within the VGG19 network, the variable denotes the feature maps acquired by the 2nd convolutional layer (after the ReLU activation) before the 2nd max pooling layer, which are the feature maps that get the best outcomes according to [45]. Inspired by [46], our content loss is defined as the Euclidean distance between the feature representation of a reconstructed image $G_{\theta G}(I^{LR})$ and the reference image I^{HR}.

In particular, this loss used the MSE metric. So, the content loss is expressed by

$$I_{VGG}^{SR} = \frac{1}{WH}\sum_{x=1}^{W}\sum_{y=1}^{H}(\varphi(\mathrm{I}^{HR})_{x,y} - \varphi\left(\hat{I}^{HR}\right)_{x,y})^2 \quad (2.22)$$

where the feature maps acquired by the second convolutional layer are denoted by φ. Within the VGG network, W and H denote the dimensions (width and height) of the respective feature maps (after the second convolutional layer and before the second max-pooling layer).

b.2 Adversarial Loss:

The adversarial loss consists of using the discriminator as a learned loss function rather than a fixed function that has been built for a particular purpose. This is especially relevant for picture synthesis applications when the loss functions are difficult to express theoretically. It's difficult to draw out a mathematical formula that analyzes how well an image matches a human face in high resolution in the SR task. Because each input may have multiple valid outputs and training set samples cannot cover all scenarios, it is inadvisable to focus only on reducing the distance between synthetic and ground-truth images. Indeed, rather than memorizing training samples, it tried to push the generator to learn the data distribution.

Rather than focusing on raw pixel level (e.g. content loss), the model can construct feature-based loss functions that retain features consistency. However, such loss functions are bound by pre-trained image classification models. However, calculating the content loss with another pre-trained model back to which specific layers to select.

On the other hand, the discriminator does not need to define the loss explicitly, because it only learns how to evaluate the generated SR images. Given enough training data, the discriminator can develop a better loss function. Lower adversarial loss function values indicate that the generator is better at producing SR samples similar to I^{HR} that may fool the discriminator.

The generative loss I_{Gen}^{SR} based on the discriminator probabilities $D_{\theta D}$ overall training samples, which is described by

$$I_{Gen}^{SR} = \sum_{n=1}^{N} -log\, D_{\theta D}\left(G_{\theta G}(I^{LR})\right) \quad (2.23)$$

where $D_{\theta D}(G_{\theta G}(I^{LR}))$ is the probability that the reconstructed images by the generator $G_{\theta G}(I^{LR})$ are real or fake. This follows the basic idea of the min-max game of GANs, but the authors choose to define it as $-\log D_{\theta D}(G_{\theta G}(I^{LR}))$ instead of $\log[1 - D_{\theta D}(G_{\theta G}(I^{LR}))]$ in order to improve the gradient behavior during minimization.

2.5.4. Image Quality Evaluation

To evaluate the SR image quality, various image quality approaches are presented in the literature. The most common techniques in SISR are Peak Signal-to-Noise Ratio (PSNR), Mean Square Error (MSE), and Structural SIMilarity (SSIM) because of their simplicity and efficiency.

PSNR: is commonly used to compare image compression quality. As a first step, MSE needs to be calculated before finding the PSNR result, which is defined as

$$MSE(f, \tilde{f}) = \frac{1}{n}\sum_{i=1}^{n}(fi - \tilde{f}i)^2 , \tag{2.24}$$

where n denotes the input signal size, $f = (f1, \ldots, fn)$ and $\tilde{f} = (\tilde{f}1, \ldots, \tilde{f}n)$. The distortion is minimized once the MSE is nearby zero.

The MSE is also used to calculate the PSNR expression as follow

$$PSNR(f, \tilde{f}) = 10\, log_{10} \frac{(2^b - 1)^2}{MSE(f, \tilde{f})}, \tag{2.25}$$

where b represents the bits per pixel value. The PSNR is expressed in deciBels (dB) and based on the MSE between the output and the input signals. High value of PSNR implies a high image quality, in the case of PSNR approaches to the infinity, and MSE approaches to zero. At the opposite extreme, a small value of PSNR implies high numerical differences between images.

SSIM: [47] is another image quality evaluation technique that is mostly used. This method focuses on the structural similarities between two images. These features are luminance, contrast, and structures. The SSIM used a combination of these features to obtain a better assessment of the similarity between the two images. The SSIM is measured between two windows *x* and *y* of an image with a size of N×N. the SSIM is expressed as

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{2.26}$$

where $\mu_x \; and \; \mu_y$ stand for the average of *x* and *y*, respectively. σ_x^2 is the variance of *x*, and σ_y^2 of *y*. σ_{xy} the covariance of *x* and *y*. $c_1 \; and \; c_2$ are two variables to stabilize the division. The range of the SSIM output is $0 \leq SSIM \leq 1$, and as the image quality is high, the SSIM approaches to 1.

3. Conclusion

Image classification is where a computer can analyze an image and identify the labels the image falls under. For SR, the algorithm tries to upscale images to recover the ground truth size from an LR image.

ML and DL methods can be used to improve Image analysis and Computer vision performance to achieve state-of-the-art results in challenging applications such as image classification and SR.

In this chapter, we covered the principle background knowledge of ML, image classification, and SR that will be used in the next two chapters.

Chapter 3

Diseases classification of plant leaves using DenseNet-121

1. Introduction

In this chapter, we focus on improving the detection and the recognition accuracy of diseases using leaves of the tomato plants by fine-tuning the DenseNet-121 model and comparing it with the state-of-the-art. DenseNet is a form of CNN that uses Dense Blocks to connect all layers (with matching feature maps sizes) directly to each other, resulting in dense connections between layers. Each layer takes extra inputs from all preceding levels and passes on its own feature maps to all next layers to maintain the feed-forward process. We compared with some state-of-the-art deep learning models that focused on detecting and classifying the diseases of tomato plant leaves, and our contributions are listed below:

1) We applied transfer learning with the tomato leaves dataset.

2) We present the DenseNet structure, the dense block, as well as the DenseNet parameters reduction philosophy.

3) We test the fine-tuned DenseNet-121 on PlantVillage and PlantDoc datasets. Thus, the DenseNet-121 obtains superior classification results with a low-complex structure than the state-of-the-art networks.

The rest of the chapter is organized as follows. Section 2 presents a literature review of the classification methods. Section 3 shows the details of the state-of-the-art models and the DenseNet-121. Section 4 presents the datasets and training settings. Section 5 discusses the obtained results and Section 6 concludes the chapter.

2. Related work:

Not long ago, image classification is a very active research area with various challenges in the identification of different types of crop diseases [48], [49], such as in sugar [50], cucumber [51], Mango [52], weed [53], and tomato [54]–[58]. In this section, we report different researches conducted on tomato diseases identification using Machine learning (Supervised as well as Unsupervised) and Deep Learning techniques (custom deep CNN and fine-tuned models).

2.1 Machine learning for diseases classification of tomato leaves

This section covers the diseases classification of tomato leaves using different Machine-Learning Methods.

Support Vector Machine (SVM) based ML for diseases classification of tomato leaves: Support Vector Machine (SVM) has been adopted in the agriculture field to automate many processes such as disease identification [9]–[11]. Mokhtar *et al*. [59] introduced the SVM to detect the diseases of tomato leaves via different kernel types (eg. Cauchy, Laplacian, and Invmult). The proposed method achieved a classification accuracy of 99.5%. Also, the authors of [60] proposed a framework to identify leaves disease on the rough tomato diseases dataset using the SVM approach. Nevertheless, the selected parameters were not well optimized. Using a histogram matching approach to recognize leaves on the tomato PlantVillage dataset, Hlaing *et al*.[12] used a SIFT texture feature SVM-based for image distribution and obtained 85.1% classification accuracy. Comparatively, the authors of [61] discussed the use of the Model-based statistical features with the SVM architecture as texture extractor, where the classification results decreased by 1.1% on the PlantVillage dataset compared to the study of [12].

Other ML algorithms for tomato leaf disease classification: Some researcher has proposed to tackle the tomato identification problem using K-Nearest Neighbors (KNN) and Random Forest methods [11], [12], [59]–[62]. To identify the diseases of tomato leaves' maturity level, Indriani *et al.* [9] applied the KNN algorithm. The authors combined the Gray-Level Co-Occurrence Matrix (GLCM) and HSV algorithms to extract features before the KNN classifier. They achieved remarkable results but with a limited amount of data. Comparatively, Xie *et al.* [10] proposed the Hyperspectral-Imaging method to classify the healthy and the Mold tomato leaves using a private dataset. On the collected tomato diseases dataset from the internet, Jakjoud *et al.* [62] made a combination of SVM and KNN as two sub-classifiers with a fuzzy-decision maker rule. Each classifier obtained a classification accuracy of more than 80%, and the combination of the sub-classifier attained more than 98%. On the other hand, the proposed work needs to be improved and tested on a public tomato dataset. Using Decision-Tree Tree and Random Random-Forest algorithms, Basavaiah *et al.* [11] proposed a Multiple Feature Extraction approach to identify diseases of tomato leaves by reducing the computing time and improving the identification accuracy. The authors used a set of essential features extraction using Hu Moments, Color histograms, Local Binary Pattern, and Haralick approaches. Thus, these extracted features are

employed for training and testing. They obtained a classification accuracy of 90% and 94% for Random Forest algorithms and Decision Tree, respectively.

Unsupervised Machine Machine-Learning Methods for Tomato Diseases Detection: Unsupervised Machine Learning Spectral and Clustering approaches have been used to classify the diseases of tomato leaves [63]–[65]. For instance, Tian *et al.* [63] improved an adaptive clustering algorithm K-means-based for diseases of tomato leaves segmentation. They obtained significant results on the greenhouse tomato leaves dataset. In the same way, the authors of [64] proposed an android application for diseases identification of tomato leaves. The identification algorithm is based on the Fuzzy C-means clustering algorithm. With minimal error, experimental results proved the efficacy of the algorithm on greenhouse diseases of tomato leaves. Park *et al.* [65] enhanced the k-means clustering algorithm to analyze the tomato leaves images and detect the infected area in the leaves. The edge-tracking and detection methods were used to identify the disease type of the extracted infected areas. Comparatively, Wang *et al*. [66] introduced an Outlier Removal Auxiliary Classifier Generative Adversarial Nets (OR-AC-GAN) to identify the diseases of tomato leaves conducted on a collected dataset from the internet. To identify miner leave tomato diseases, Xu *et al.* [67] employed a spectroscopy algorithm based on near-infrared using a collected dataset from the Greenhouse. The experiment resulted in a slight improvement in classification accuracy (60%). In contrast, Lu *et al*. [68] utilized hyperspectral images to identify a specific disease (yellow tomato leaves) using a plant leaf diseases dataset. They achieved a classification accuracy of 100%.

2.2 Deep learning for diseases classification of tomato leaves

Classification using Convolutional Neural Network: Following the success of CNNs in machine learning and computer vision, LeNet5 [69], was the first network used for document recognition. This network consists of only 5 convolutional layers. However, AlexNet [19] was the first deep CNN network in the ImageNet 2012 challenge. In the image classification task, AlexNet outperformed all the previous shallow networks. The authors achieved a classification accuracy of 84.60%. The design of AlexNet has 61M network parameters. In similar work, Oxford's Visual Geometry Group (VGG) introduced two different designs, VGG16 and VGG19 [70]. VGG16 obtained a Top-5 accuracy of 91.90% in ImageNet-2014. It has 138M network parameters. The number of successive convolution layers is the only difference between VGG16 and VGG19, where VGG19 has 19 convolution layers. Thus, the number of parameters of the VGG19 is 143M.

These extra layers gave the model more ability to learn more complex architecture. In another research, Google researchers proposed a new design named the GoogLeNet (also known as Inception-v1) [71]. With a top 5 accuracy rate of 92.2%, the Inception-v1 won the ImageNet-2014. Following the same concept of InceptionV1, the authors developed other novel versions InceptionV2 [72] and InceptionV3 [73]. The Inception block is the central idea of Inception networks. To capture several features from the images, this block was set to let the information go wider and not deeper only. The Inception-v1, Inception-v2, and inception-v3 have 23M, 11.2M, and 25M parameters, respectively (Inception-v1 is larger than Inception-v2). In an impressive work, He *et al.* [36] introduced the Residual Network (ResNet). The network has 25M parameters (and ResNet50 has 23M), which was the winner in the ImageNet-2015 with a 94.29% Top-5 accuracy. Unlike the previous designs, ResNet has 152 layers, which makes the network very deep. ResNet is based on residual connections. A residual connection is set on a stack of convolution layers, where it links between the input of the first convolutional layer and the output of the last layer. In a recent approach, Huang *et al.* [37] introduced a novel CNN design named Dense Convolutional Network (DenseNet). They proposed straight connections between two layers that have the same size feature maps. Even though demonstrating no optimization difficulties, the authors proved that this network has the ability to scale to hundreds of layers. Experimental results yielded reliable improvement in the classification accuracy without any degradation in the performance (Overfitting). Under a different structure, DenseNet obtained state-of-the-art results using several competitive datasets, and with fewer parameters (7.1M). Following the inception-v3 architecture, Chollet [74] introduced eXtreme inception (Xception) network. The author replaced the inception module with a depth-wise separable convolution followed by a point-wise separable convolution. This design consists of 71 layers deep with 22.9M parameters. The Xception yielded a Top-5 accuracy of 94.50% in the ImageNet challenge.

Classification of tomato crop leaves diseases using shallow networks: Recently, many researchers in the agricultural field have adopted deep learning to overcome several issues including plant diseases classification [13]–[17]. For instance, Ashqar *et al.* [17] employed a shallow CNN to detect healthy tomato leaves and five plant diseases. The authors tested the CNN on the collected tomato leaves dataset and obtained a classification accuracy of 99.84%. Using the greenhouse tomato diseases dataset, Cevallos *et al.* [75] applied data augmentation preprocessing to increase the classification accuracy of their CNN. Experiments yielded an overall classification

accuracy of 86.57%. In another work, Brahimi *et al*. [76] employed a CNN to classify several types of leaves diseases. The authors retrieved the extracted features automatically from raw images. Using an old dataset of tomato leaves diseases, they obtained a classification accuracy of 99.18%. To detect tomato diseases at the greenhouse of Agricultural University, the authors of [77] proposed the Feature Pyramid Network (FPN) algorithm CNN-based. The FPN helped to extract multi-scale features, which improved the mean average precision (mAP) from 90.7% to 99.5%.

Deep network for tomato crop leaves diseases classification: Some recent research works have proposed to use deep CNN in order to tackle the tomato diseases classification problem [78]–[80]. To detect the diseases of tomato leaves on the tomato PlantVillage dataset, the authors of [81] built a deep CNN. The proposed method yielded an overall accuracy of 96.34%. Likewise, Karthik *et al.* [82] proposed an attention-based deep residual network for detecting only three diseases in tomato leaves of the PlantVillage dataset [8]. They obtained an accuracy of 98% compared to other existing networks. Whereas, Elhassouny *et al.* [79] introduced a new Deep CNN to extract features from several types of diseases of tomato leaves images. This deep CNN obtained a classification accuracy of 90%. In another different approach, Wu *et al.* [80] developed a Deep Convolutional Generative Adversarial Networks (DCGAN) classification framework to identify diseases of tomato leaves. The framework yielded an average test accuracy of 94.33%. Similarly, Emebo *et al.* [81] implemented a Deep CNN on a Raspberry Pi to detect two types of diseases from a collected dataset. The experimental results demonstrated that the proposed method has the ability to detect diseases with an accuracy of 99.01%.

Transfer learning for tomato crop leaves diseases classification: The transfer learning approach has demonstrated the ability to solve complex problems of classification, especially in diseases of tomato leaves [83]–[85]. For example, the authors of [18] achieved a classification accuracy of 91.67% after pretrained the AlexNet [19] and the F-RCNN framework to detect tomato leaf diseases. In other work, Zhang *et al.* [86] improved the time and the accuracy (98%) for detecting four diseases of tomato using a Fast Regional Convolutional Neural Networks (Fast RCNN) conducted on an AIChallenger crop dataset. In similar research, Liu *et al.* [87] used the pretrained MobileNetv2 and YOLOv3 models to improve the identification of gray tomato leaves spot diseases using an embedded camera. Whereas, Ahmad *et al*. [88] discussed the pretrained VGG-16, VGG-19, Inception-v3, and ResNet, and the classification performance of three types of

diseases of tomato leaves. The inception-v3 outperformed the other designs by 1.7% in the classification results.

Some researchers have tried to tune the parameter of the pretrained model in order to improve the classification performance [89]–[91]. Similarly, Suryawati *et al.* [92] tested different Deep networks such as AlexNet, GoogLeNet, and VGGNet on the PlantVillage dataset to identify diseases of tomato leaves. Experimental results showed that the VGGNet (95.24%) has the ability to identify diseases of tomato leaves better than other networks. The study of [93] employed the AlexNet and SqueezeNet to identify diseases of tomato leaves using the PlantVillage dataset and a collected dataset. The AlexNet obtained an accuracy of 95.65% and outperformed the SqueezeNet. In a different study, Mkonyi *et al.* [94] conducted their experiments on a collected dataset to recognize one plant leaf disease (Tuta absoluta) using VGG16, VGG19, and ResNet50. The VGG16 (91.1%) outperformed the two other pretrained networks in the experimental results. Following the same approach, Shijie *et al.* [95] combined the pretrained VGG16 with the SVM to classify the diseases of tomato leaves. The proposed method achieved an overall classification accuracy of 89%. To identify three specific types of plant leaves diseases, Jiang *et al.* [96] pretrained the ResNet-50 on the PlantVillage dataset. The results showed that the pretrained ResNet-50 (98.0%) can perform well on this task. Similarly, Zaki *et al.* [97] used MobileNet-v2 to identify three types of tomato plant leaves diseases as well as healthy ones. The authors evaluated the MobileNet-v2 on a PlantVillage dataset and achieved a classification accuracy of 95.62%. The study of [98] proposed to apply transfer learning using SqueezeNet to recognize six different diseases of tomato plant leaves as well as healthy leaves. The proposed method obtained 86.92% classification accuracy. Whereas, the authors of [99] proposed to classify the diseases of tomato leaves using a pretrained ResNet after the data augmentation process. Extended results were applied to the collected dataset and achieved a classification accuracy of 97.01%. On the other hand, Aversano *et al.* [86] tested three different models (VGG19, Xception, and ResNet-50) to identify the diseases of tomato leaves. Experimental results were conducted on PlantVillage, and the VGG19 (97%) outperformed the Xception (95%) as well as the ResNet-50 (60%) in terms of classification accuracy. Likewise, the authors of [100] discussed the classification accuracy of different pretrained models like VGG16, DensNet121, and DensNet161 to classify six different types of diseases of tomato leaves from a collected dataset. The DensNet161 surpassed the other pretrained networks with an overall accuracy of 95.65%. Using the same dataset, Too *et al.* [101]

used fine-tuned CNN models for diseases classification of 14 different crop plants, comprising Visual Geometry Group VGG16, Inception V4, and ResNet with 50, 101, and 152 layers, as well as DenseNet with 121 layers (DenseNet-121). The DenseNet-121 beat all the previous networks that obtained a test accuracy of 99.75%. Likewise, Chakraborty *et al.* [102] employed DenseNet101 to classify 13 different plant leaves diseases using PlantDoc [7]. With the DenseNet-121, the authors obtained a classification accuracy of 92.5%. Regardless of the source of the experimental images, all the previously mentioned studies were conducted on different plants or specific tomato diseases. However, choosing the low-complex classifier with a significant accuracy for most diseases of tomato leaves was not taken into consideration. As investigated in [101], [102] and due to the best performance of the pretrained DenseNet-121, we decided to fine-tune the DenseNet-121 on tomato diseases of the PlantVillage [8] and fine-tuned it on the PlantDoc datasets [7]. In addition, we compare DenseNet-121 with all state-of-the-art approaches [12], [61], [78], [92], [93], [99] that focused on the classification of all the diseases of tomato leaves types (9 diseases as well as healthy ones) using the PlantVillage [8] as well as the PlantDoc datasets [7].

3. Dense Convolutional Network

The DenseNet [37] design has the ability to achieve state-of-the-art accuracy with a fewer number of parameters, compared to other networks [36], [70]. The most explicit difference between the insight of dense connectivity and the residual connection is the combination between the layer's identity and output. In the ResNet design, identity mapping is employed to help the propagation of the gradient. Whereas, the element-wise addition is set to combine the output with the input via residual blocks. Fig 3.1 demonstrates the layout of DenseNet. The DenseNet exploits the concept of identity path in the whole network and not only in blocks. By a simple connection, DenseNet employed an identity path to each layer and at different stages (short connections and long connections). More formally, DenseNet consists of a dense structure, which is designed by a dense connection between convolution layers, as shown in Fig 3.1. Thus, each layer input is the

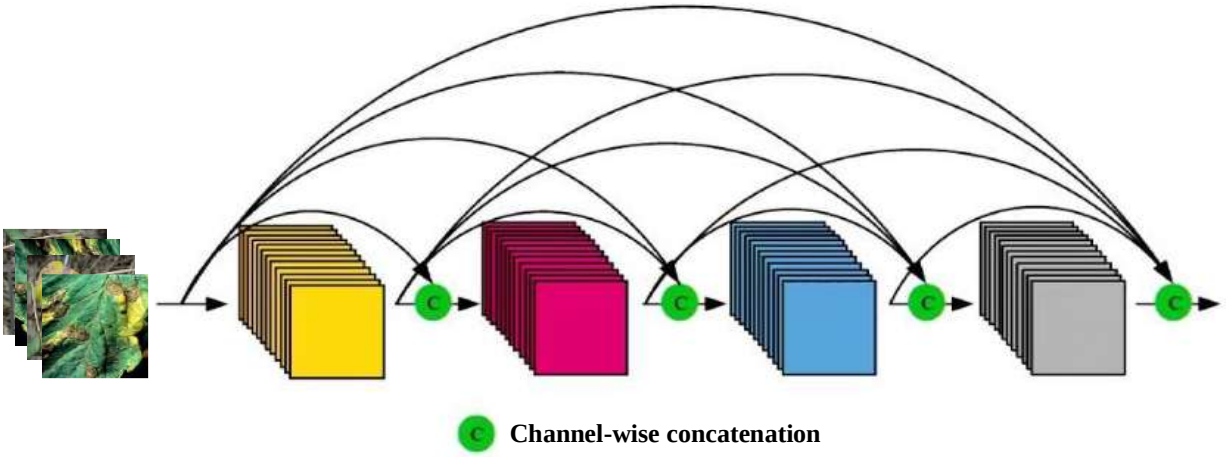


Fig 3.1. Structure of the dense block components: Dense connections (short and long connections), and the channel wise concatenation operator. [37]

concatenation of the entire preceding layers output, which is the structure of "collective knowledge". In this manner, each layer can collect information from all prior layers.

3. 1 Feature maps in dense connections

In the feedforward computation, l denotes the layer index. Consider that each layer produces a feature maps, denoted by $x_0, x_1, x_2, \ldots\ldots, x_l$. The layer l input is composed of the feature maps concatenation that is already produced by the layers 0 up to $l-1$, and $[x_0, x_1, x_2, \ldots\ldots, x_{l-1}]$ represents its tensor. In addition, the layer output is then passed through the non-linear function $H(\cdot)$ and is described as

$$x_l = H([x_0, x_1, x_2, \ldots\ldots, x_{l-1}]) \quad 3.1$$

Each layer is able to retrieve the feature maps from the preceding layers. This design has the advantage of avoiding redundant feature maps, which were present in residual connections. The DenseNet utilized three operations to compute the weight layers. For regularization, batch normalization is set after each convolution layer. In addition, the mini-batch mean and variance of the network input are employed for shifting and scaling. This normalization may enhance network optimization and training. The ReLU function follows the batch normalization process and the 3 × 3 convolutional layer.

The design pattern of DenseNet consists of a convolution layer, batch normalization (BN), and ReLU. In addition, skip connections are introduced to each individual layer. Fig 3.2 illustrates a detailed example of DenseNet structure. The network is composed of 4 densely connected layers. The 6 channels represent the input data x_0 . This input is also connected to all next layers due to dense connectivity. Next, the layer H_1 produces 4 feature maps from the inputx_0 . Thus, the same concept is applied to all next layers. The transition layer is set at the end of the network to receive the concatenation of all produced feature maps $[x_0, x_1, x_2, x_3, x_4]$. Since each layer aggregates the entire 4 feature maps in the concatenation tensor, the output would have a depth of 22. The number of feature maps added by each layer is considered as a hyperparameter named "growth rate", which has a direct impact on the properties of the network. Following the notation presented in [37], we denote *k* as the parameter of growth rate. Hence, the number of feature maps increases from the input until the end of the network. If an input layer generates *k* feature maps, the next layer *l* may have $k_0 + k \times (l-1)$input feature maps, where x_0 denotes the channels of the input. Thus, extensive experiments of [37] proved that even if the *k* is small, the network obtained remarkable results in image classification tasks.

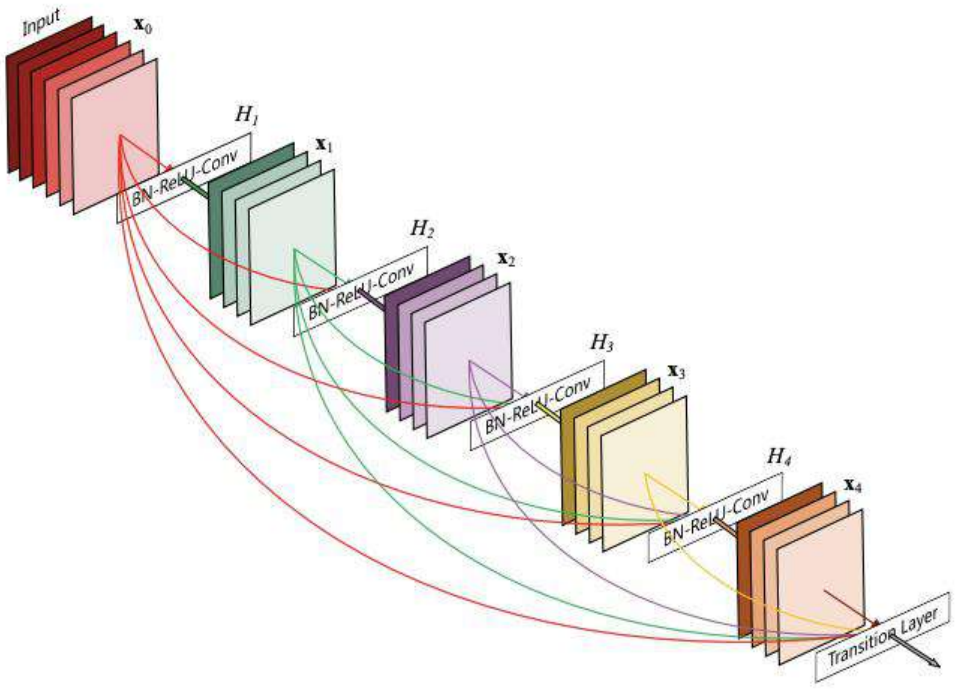


Fig 3.2. DenseNet structure with 4 weight layers (with batch normalization (BN), ReLU and the convolution operation as layers) and a growth rate of $k = 4$. [37]

The use of kernel sizes 3 × 3 in the convolutional layers performed well in the design of DenseNet. However, DenseNet is designed to use the pooling as well as avoid the concatenation of different feature maps size. With the concatenation operation, each layer may have access to all earlier generated feature maps. Thus, every network block can have a dense connectivity structure with an appropriate feature map size. At the end of the network, the transition layer retrieves the information from the dense connections path and the identity path. The authors of [37] illustrated the importance of each individual feature map with different experiments. The presented results showed that the weights of every layer are shared over the different prior feature maps. For transition and fully-connected layers, the authors demonstrated that these layers have the ability to make the feature maps more informative and even may improve the classification results.

3.2 Network parameters reduction:

The DenseNet design motivates the compression concept. Where at a connection between two blocks, the feature maps number is decreased by a compression rate of θ. The loss of information is remarkably low since the next block in the network is extremely focused on the complex feature

maps produced at the end of the preceding block. Consequently, it appears that more reduction of feature maps may have an impact on the network. The authors of [37] employed the convolutional layer with kernel size 1×1 as an extra transition layer to reduce the network parameter, as discussed in [36].

Assume that a prior dense block generated m feature maps. The output channels of the convolutional layer are then set to $\theta \times m$ with $0 < \theta \leq 1$. Following the state-of-the-art [36], [73] approaches, the authors demonstrated that the use of bottleneck layers can effectively reduce the network parameters. That is, the authors [37] adopted the bottleneck layers in the DenseNet design. Before each layer (BN, ReLU, 3×3 convolutional), the bottleneck layer can be adopted. With the use of 1×1 kernels, the BN, ReLU, and convolutional layers would be optimized. In addition, CNNs generally contain pooling layers in order to reduce the size of the feature maps in the network, especially for image classification. Consequently, DenseNet benefits from pooling layers in the reduction of its parameters. The yielded design block consists of output feature maps with just 4k parameters. Thus, the input feature maps are decreased particularly at the end of the network. Where a high number of feature maps have been produced with fewer parameters. The authors successfully improved the computational efficiency as well as reduced the parameters of the network.

3.3 The DenseNet results

In the experiment of [37], the authors proposed to test four configurations, DenseNet-121, DenseNet-169, DenseNet-201, and DenseNet-264. For instance, the number 121 in DenseNet-121 refers to the number of layers with trainable weights. All DenseNet designs achieved a significant improvement in terms of accuracy by increasing the number of parameters (in different designs), in the absence of any signs of overfitting or performance degradation. Under different configurations, the DenseNet structures achieved state-of-the-art results compared to ResNet [36] and other networks through the most challenging datasets.

3. 4 DenseNet-121

The study of [37] proposed multiple network architectures, specifically designed for the ImageNet dataset. The DenseNet-121 has fewer parameters compared to other configurations. DenseNet-121 successfully proved the efficiency of the dense connectivity pattern for image classification and outperformed other common models such as VGG-16 [70] or ResNet-34 [36].

Table 3.1. DenseNet-121 model architecture [37].The growth rate for all the networks is k = 12

Layers	Output size	DenseNet-121 k=32
Convolution	112 × 112	7 × 7 Conv, stride2
Pooling	56 × 56	3×3 Max pooling, stride 2
Dense Block 1	56 × 56	$\begin{pmatrix}1\times1\text{ Conv}\\3\times3\text{ Conv}\end{pmatrix}\times 6$
Transition layer 1	56 × 56 28 × 28	$\begin{pmatrix}1\times1\text{ Conv}\\2\times2\text{ Avg pool, stride2}\end{pmatrix}$
Dense Block 2	28 × 28	$\begin{pmatrix}1\times1\text{ Conv}\\3\times3\text{ Conv}\end{pmatrix}\times 12$
Transition layer2	28 × 28 14 × 14	$\begin{pmatrix}1\times1\text{ Conv}\\2\times2\text{ Avg pool, stride2}\end{pmatrix}$
Dense Block 3	14 × 14	$\begin{pmatrix}1\times1\text{ Conv}\\3\times3\text{ Conv}\end{pmatrix}\times 24$
Transition layer 3	14 × 14 7 × 7	$\begin{pmatrix}1\times1\text{ Conv}\\2\times2\text{ Avg pool, stride2}\end{pmatrix}$
Dense Block 4	7 × 7	$\begin{pmatrix}1\times1\text{ Conv}\\3\times3\text{ Conv}\end{pmatrix}\times 16$
Classification layer	1 × 1	7 × 7 Global Avg pool

The proposed network architecture has four dense blocks. In general, the network is composed of batch normalization, ReLU, and convolutional layers. Table .3.1 illustrates the overview of the DenseNet-121 structure. More formally, a 7×7 convolution and a 3×3 max-pooling layers (both have a stride of 2) are employed to shrink the size of the input images before the first block. Every

transition layer follows the dense connectivity pattern with the mix of a bottleneck and convolutional layer via an identity path. The convolutional layers are configured with a kernel size of 1×1 in bottleneck layers and 3×3 otherwise. The output channels are primarily related to the growth rate of the network ($k = 32$).

For bottleneck layers, the transition layers are composed of a compression (BN, ReLU, 1×1 convolution) and a 2×2 average pooling layer, which are connected with the dense blocks. Every compression layer decreased by half of the input feature maps with a compression factor of $\theta = 0.5$. The feature vector is produced after the global average pooling of the last dense block. The DenseNet-121 has a fully-connected layer of 1000 neurons followed by a Softmax activation for label classification.

In order to fine-tune the DenseNet-121, the following changes are applied. Since the model is already trained, we used ImageNet parameters as pretrained weights. Dense layers are very narrow and the classification is based on knowledge of pretrained weights. The default input dimension of the DenseNet-121 is variable and adaptable and so is the output. The change of dimension required in the input layer concerns the width and height setting. Those parameters are modified to match those of the input images. In the same way, the output classes' number must be the same as the number of neurons in the output layer. In our experiments, we have used a 10-class classification with an image size of 256×256 pixels. Hence, we have added an input layer with a width and height of 256×256. The input layer is followed by a convolution layer of a kernel of 3×3 with 3 filters, then we insert the DenseNet-121 pretrained model. Also, we set the global average-pooling layer just after the pretrained DenseNet-121 model, followed by a fully connected layer of 256 neurons to improve computational efficiency. In addition, we fix the number of neurons in the output layer to 10 with sigmoid activation. Meanwhile, to obtain better classification performance and less computation time, we fine-tuned the DenseNet-121 classification network trained on ImageNet with the Plant Village and PlantDoc datasets.

4. Datasets and training settings

The plant datasets are usually private for some institution or laboratory examination and are not open to the public. In our study, we only used the public datasets such as PlantDoc and PlantVillage datasets.

4.1 PlantDoc dataset

Table 3.2 Categorization of each tomato leaf image in the PlantDoc dataset.

ID	Category	Number of images
0	Bacterial spot	101
1	Late blight	101
2	Mold leaf	85
3	Septoria leaf spot	140
4	Spider mites	2
5	Early blight	79
6	Tomato yellow curl virus	70
7	Tomato mosaic virus	44
8	Healthy	55

The PlantDoc dataset was collected taking into account the real-world problems [7]. The dataset provided 2,598 images of 13 plant types with a total of 28 classes (18 disease and 10 healthy classes) for experimenting on the classification problem. Table 3.2 demonstrates the details of tomato leaves in the PlantDoc dataset. The dataset contains 8 diseases of tomato leaves as well as one healthy class. The size of the images is variable, and it contains some misclassified images presented in the image, as shown in the example of Fig 3.3.

Fig 3.3. An example of tomato leaves. One leaf has late blight, and the others are healthy [7].

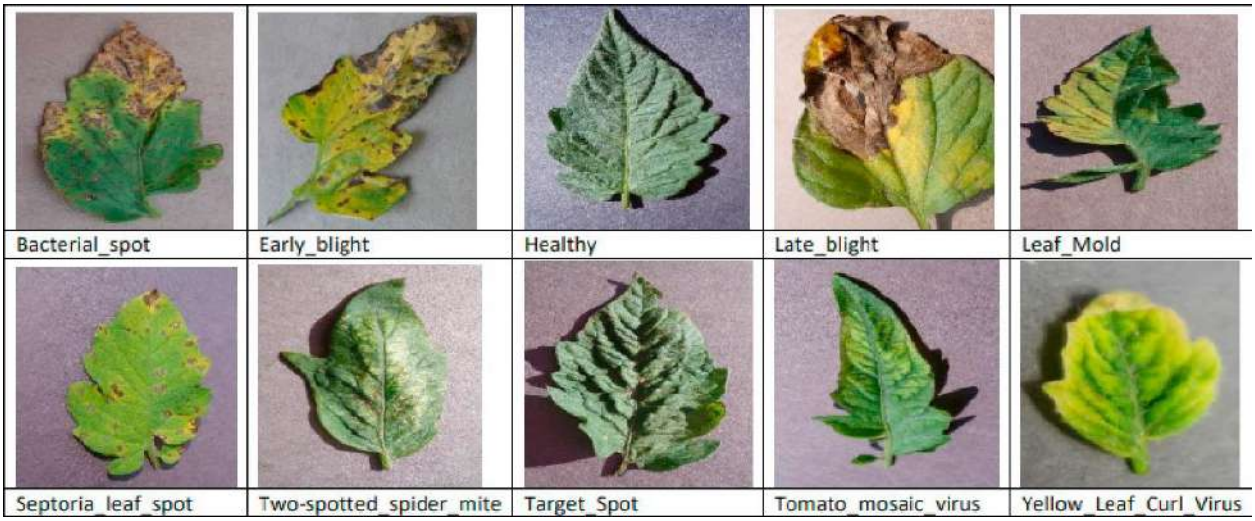


Fig 3.4. Overview of the PlantVillage dataset images. [8]

4.2 PlantVillage dataset

The other crop disease images used in the experiments are obtained from the Plant Village dataset [8], as illustrated in Fig 3.4. The dataset includes over 50,000 images of 14 kinds of crops, such as tomato, corn, grape, apple, and soybean. Considering tomato as the highest-produced crop having the largest number of diseases in the Plant Village dataset, it has been selected as the target crop in this research. Every image in the Plant Village has a dimension of 256×256 pixels (denoted as reference HR images). The tomato images are reorganized in Kaggle for a Plant Disease detection System with pepper and potato crops, available in [103], and the total number of tomato

images is exactly 16,012 in this dataset. Table 3.3 shows the 9 categories of tomato leaf diseases as well as the healthy ones.

Table 3.3 Categorization of each tomato leaf image in the PlantVillage dataset.

ID	Category	Number of images
0	Bacterial spot	2 127
1	Early blight	1 000
2	Late blight	1 909
3	Mold leaf	952
4	Septoria leaf spot	1 771
5	Spider mites	1 676
6	Target spot	1 404
7	Tomato yellow curl virus	3 209
8	Tomato mosaic virus	373
9	Healthy	1 591

4.3 Training setting

To this end, we changed the DenseNet-121 settings of the input layer dimension to 256 × 256 and the number of output classes to 10 of the output layer for PlantVillage and 9 for PlantDoc to meet our image dimensions. Also, we set the Adam optimizer for optimization with parameters of β_1 =0.9, β_2= 0.999, and ε= 0.1, and the learning rate is set to 0.002 and decreases to half after 30 epochs.

Preprocessing: Due to the limitation of dataset images, we went through a data augmentation process to enlarge the number of training data. This stage of experiments needs to be effective and

an important step for the classification of the diseases. This will improve learning and the model can converge quickly.

5. Results and discussion

We fine-tuned the DenseNet-121 on tomato leaf diseases of the PlantVillage [8] and fine-tuned it on the PlantDoc datasets [7]. In addition, we compare all state-of-the-art approaches [12], [61], [78], [92], [93], [99] that focused on the classification of all the diseases types of tomato leaves (9 diseases as well as healthy ones) using PlantVillage as well as PlantDoc datasets.

5.1 Classification using PlantVillage dataset

Table 3.4 Classification results of tomato leaf images using the PlantVillage dataset.

Category	Number of images	Number of parameters
SIFT+SVM [12]	85.1%	Not mentioned
SVM [61]	84%	Not mentioned
Deep CNN [78]	96.34%	Not mentioned
VGG16 [92]	95.24%	119.6M
AlexNet [93]	95.65%	61M
SqueezeNet [93]	94.3%	5M
VGG19 [99]	97%	138M
Xception [99]	95%	23M
ResNet-50 [99]	60%	23.6M
DenseNet-121 (our)	**97.81%**	7.1M

In this study, we assessed the DenseNet-121 in order to compare its results with the state-of-the-art [12], [61], [78], [92], [93], [99] that focused on the diseases classification of tomato leaves task.

First, we tested the DenseNet121 model and compared its performance to other networks, namely SIFT+SVM, SVM, Deep CNN [78], VGG16, AlexNet, SqueezeNet, VGG19, Xception, ResNet-50, and DenseNet-121. The results for the 10-class classification are shown in Table 3.4. All the models were trained first on the original PlantVillage dataset. The DenseNet121 model obtained the best results among all the tested models with an accuracy of 97.81%. The results of all the machine learning models (SIFT+SVM, SVM) show clearly a low performance with a classification accuracy of only 84.1% for SIFT+SVM and 85% for SVM. However, Deep CNN [78] obtained remarkable results with an overall accuracy of 96.34%. For the pretrained models, the ResNet-50 performed poorly with a classification accuracy of 60%. Since the threshold of most pretrained models is around 90%, SqueezeNet got the lowest accuracy among the other models with an overall accuracy of 94.3%. Xception, VGG16, and AlexNet achieved approximately the same classification accuracy with a small difference between the three models, but AlexNet

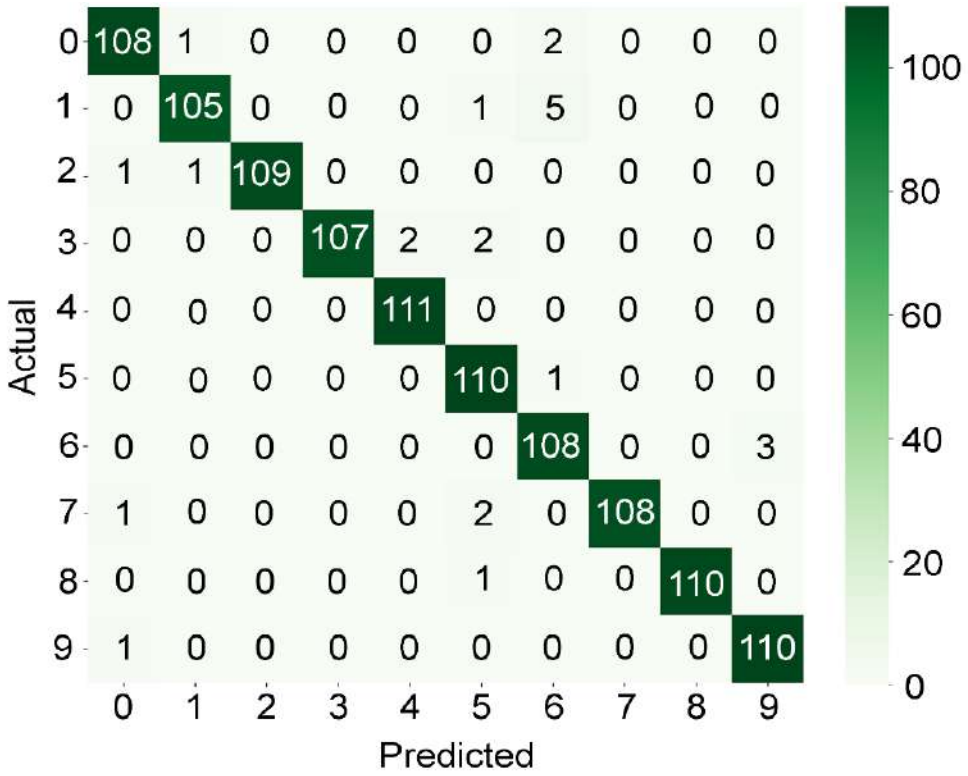


Fig 3.5 Confusion matrix of DenseNet-121 evaluated on PlantVillage dataset. The axes x and y indicate the ID of diseases in Table 2.3.

performed well in this category. As first ranked, we can name VGG19 and DenseNet-121. However, DenseNet-121 outperformed all the architecture in terms of classification accuracy and even in the number of parameters. In addition, although SqueezeNet [93] has the lowest number of parameters, DenseNet-121 beat it in terms of performance and accuracy.

DenseNet-121 performance: The confusion matrix for DenseNet-121 identification of the tomato diseases using the PlantVillage dataset is illustrated in Fig 3.5. The DenseNet-121 could recognize most tomato diseases. For instance, the network identified all the samples of class 4 (Septoria leaf spot). In addition, it could recognize most samples of healthy class (9), Spider mites class (5), and mosaic virus class (8) with just one image was miss classified, as well as the Late blight class (2) confusing just two images. However, for the early blight class (1), the network recognized most samples but confused 5 images with the Target spot class (6). In addition, the Target spot class (6) confused 3 samples with the healthy class (9). Generally, the overall values and rates are accurate for the classification of diseases and may help for further analysis and treatment.

Target spot classified as healthy
(a)

Leaf Mold identified as Early blight
(b)

Early blight categorized as Target spot (c)

Early blight categorized as Target spot (d)

Fig 3.6 Misclassified samples using DenseNet-121 from PlantVillage dataset.

Fig 3.6 shows different samples of misclassification. The image Fig 3.6 (a) looks healthy from the first sight, but after zoom-in on the upper part, the target spot looks at the first stage that's why the DenseNet-121 was confused in its consideration of whether is healthy or not. Fig 3.6 (b) illustrates an infected leaf with a Mold leaf. The network failed to correctly classify the sample. The curvy shape at the lower part with a yellowish color of the leaf confused the network to label it as an early blight due to the presence of early blight characteristics.

Early blight (TP)
(a)

Yellow-leaf curl virus (TP)
(b)

Bacterial spot (TP)
(c)

Healthy (TP)
(d)

Fig 3.7 TP examples using DenseNet-121 from PlantVillage dataset.

Fig 3.6 (c and d) demonstrate an example of two early blight images identified as target spots. In general, the difference between early blight and target spot is that the leaf of early blight at the early stage may have the surrounding brown dashes, which are a bit similar to the target spot disease. From the images, we can observe that some characteristics of target spots appear on the leaf that's why the DenseNet-121 cannot distinguish the main difference in such examples.

Fig 3.7 demonstrates a few TP images. Since the leaves' characteristics are well recognizable and even the lesions are clear, the network correctly labeled the images (Early blight, Yellow Leaf curl virus, bacterial spot, and healthy).

5.2 Classification using PlantDoc dataset

Table 3.5 Classification results of tomato leaf images using the PlantDoc dataset.

Category	Number of images	Number of parameters
VGG16 [92]	91.4%	119.6M
AlexNet [93]	91.6%	61M
SqueezeNet [93]	92.9%	5M
VGG19 [99]	96.1%	138M
Xception [99]	93.8%	23M
ResNet-50 [99]	92.1%	23.6M
DenseNet-121 (our)	**96.5%**	7.1M

To assess the DenseNet-121 with a real-world dataset, we fine-tuned DenseNet-121 and other deep networks in order to compare them in terms of accuracy and performance [92], [93], [99] for the diagnosis of tomato leaves.

We added some new images from the internet to the PlantDoc dataset due to the lack of some classes. We compare the DenseNet121 with other fine-tuned networks, namely VGG16, AlexNet, VGG19, Xception, ResNet-50, and DenseNet-121 in terms of classification performance, as shown in Table 3.5. We fine-tuned all the networks on the PlantDoc dataset. The DenseNet121 model obtained the best results among all the models with an accuracy of 96.5%. The VGG19 had the closest rate by 0.4% in terms of accuracy, but in terms of parameters, it is a large network. For the ResNet-50, it performed well behind other models like VGG16 and AlexNet in the overall

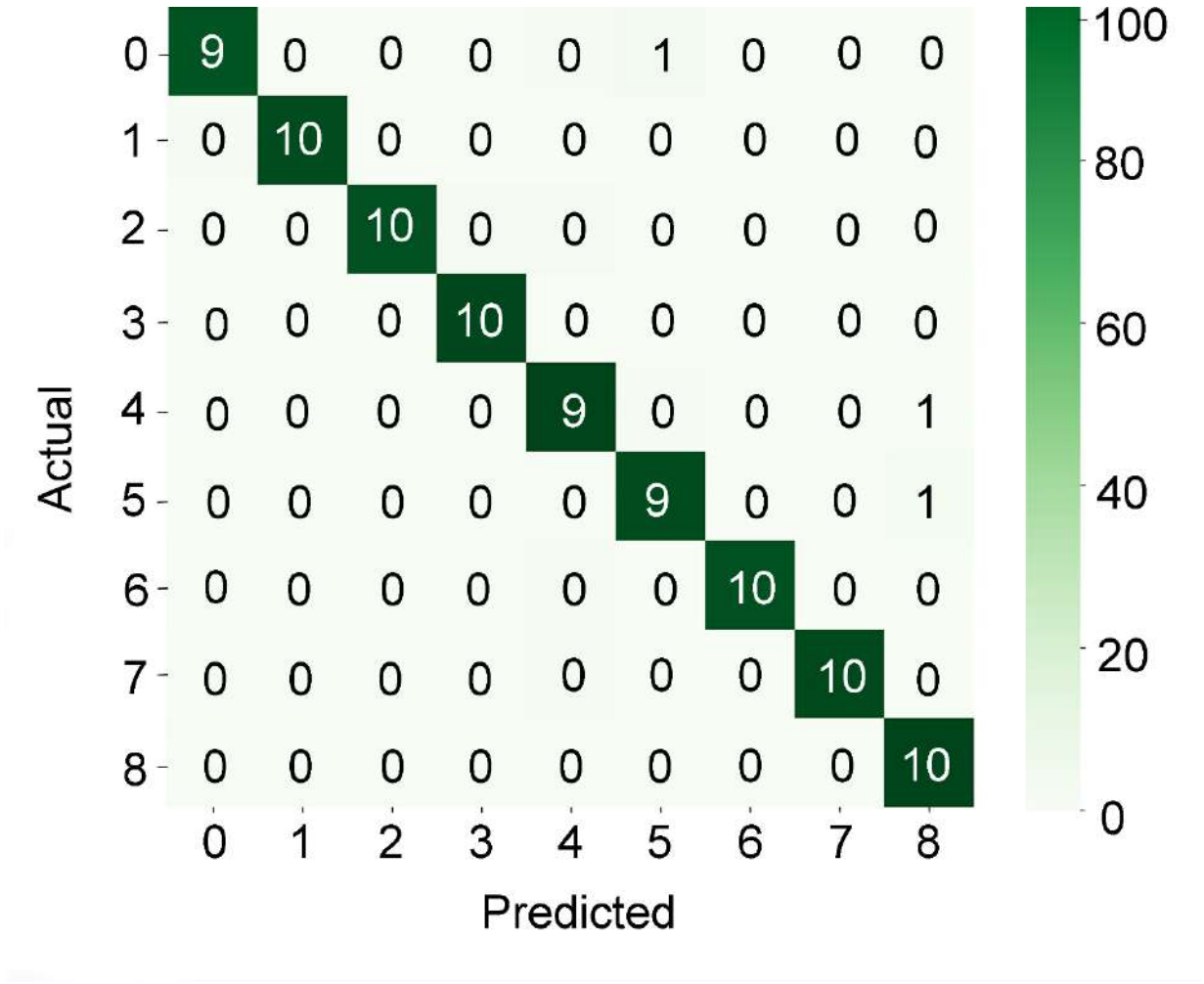


Fig 3.8 Confusion matrix of DenseNet-121 evaluated on PlantDoc dataset. The axes x and y indicate the ID of diseases in Table 3.2.

classification accuracy. Thus, it is clear that the DenseNet-121 has the ability to achieve high accuracy in the diseases classification of tomato leaves with a low number of parameters.

DenseNet-121 performance: Fig 3.8 shows the confusion matrix of DenseNet-121 tomato leaf diseases classification using the PlantDoc dataset. Generally, most samples are correctly recognized by the DenseNet-121, such as the classes 1 (Late blight), 2 (Mold leaf), 3 (Septoria leaf spot), 5 (Early blight), 7 (mosaic virus), and 8 (healthy). However, it could miss classified just 1 sample per class in 0 (Bacterial spot), 4 (spider mites), and 6 (yellow curl virus). Thus, DenseNet-121 has the ability to identify most tomato leaf diseases even with a limited dataset.

To evaluate the robustness of the DenseNet-121, we analyze the misclassification samples for both datasets. Fig 3.9 illustrates the misclassified and the correct images. For PlantDoc, we can

Bacterial spot classfied as Spider mites
(a)

Two-spotted mite classified as healthy
(b)

Fig 3.9 Two misclassified examples using DenseNet-121 from PlantDoc dataset.

observe from Fig. 3.9 (a) that the model classified the bacterial spot image as two-spotted diseases. It's notable on the upper part that there are some characteristics of a speckled appearance on the leaf, which confused the model between the two diseases. The network failed to identify the bacterial spot, which appears in most parts of the leaf. Fig. 3.9 (b) shows a misclassified two-spotted mite disease. The DenseNet-121 recognizes it as a healthy leaf. We can observe that the two-spotted mite appears on the small leaf. The bigger leaf is a healthy one. Moreover, the network was confused about considering the leaf as a healthy or two-spotted mite. The presence of two different classes in one image usually confuses all the image classification examples.

Fig 3.10 demonstrates some TP samples. Due to the obvious clarity of the disease characteristics, the DenseNet-121 has correctly classified the samples (leaf Mold, Mosaic virus, Yellow Leaf curl virus, and late blight).

Leaf mold (TP)

Mosaic virus (TP)

YellowLeaf Curl virus (TP)

Late blight (TP)

Fig 3.10 TP samples of DenseNet-121 from PlantDoc dataset.

These examples proved the remarkable performance of the DenseNet-12, which can correctly identify almost all of the diseases of tomato leaves. In both datasets, DenseNet-121 identified tomato leaf diseases with high accuracy and fewer parameters. Consequently, this network has a better perspective to improve the vigor of self-diagnosis and field scouting.

6. Conclusion

In this chapter, we covered the classification of state-of-the-art approaches in diseases of tomato leaves using ML and DL. We presented the DenseNet and its principle structure in detail, and we applied transfer learning to DenseNet-121 for further analysis. Extensive results showed that the low-complex model (DenseNet-121) beat all the state-of-the-art networks by achieving the best overall classification accuracy as well as with fewer network parameters.

In the next chapter, we use the DenseNet-121 for further analysis of LR images. In addition, we design our SR model in order to improve the LR crop leaves diseases classification.

Chapter 4

Super Resolution for improving the diseases classification of LR plant leaves diseases

1. Introduction

In this second contribution, we supposed that the farmer obtained High-Resolution (HR) images of plant leaves from the field. Because of the small size of plant leaves and other limitations, the obtained HR images can miss some detailed information, which results in blurred LR images of leaves with fewer details. To overcome this issue, we introduced a novel SR algorithm named Wider-activation for Attention-mechanism based on a Generative Adversarial Network (WAGAN) to improve the classification of the tomato diseases LR images. The WAGAN consists of three main parts; the generator network, which has the Wider Activation for Residual Channel Attention Block (WARCAB) as the principal block, the discriminator network, and the adversarial loss. To evaluate the potential of the proposed method in plant diseases recognition, we first recovered SR leaves diseases images from LR images using the WAGAN and then classify them with DenseNet-121. Our contributions are highlighted below.

1) We introduce a new efficient SR network for tomato crop diseases images.
2) We demonstrate the use of linear low-rank convolution with the attention feature in reducing the network parameters and also boosting the classification accuracy in the WAGAN.
3) For the first time, our WAGAN obtains superior qualitative and classification results, and less than the HR by at least 0.18%, compared to the state-of-the-art approaches with a large reduction in the number of parameters (3.6 times less than ft_ESRGAN).

2. Problem formulation

The farmer is supposed to obtain HR images of plant leaves from the field. Due to the tiny size of plant leaves and other limitations, the obtained HR images can miss some detailed information resulting in LR images of leaves with fewer details [104]. Fig 4.1 illustrates some examples of the classification performance of HR images compared to LR ones. Extensive results using LR images demonstrated that DenseNet-121 couldn't identify well the tomato leaves diseases.

To overcome this issue, we can use an SR algorithm to improve the content of the images before classification. To this end, most classification on crop diseases have conducted their studies on training and testing their network using HR images; thus, the LR images remarkably influence the

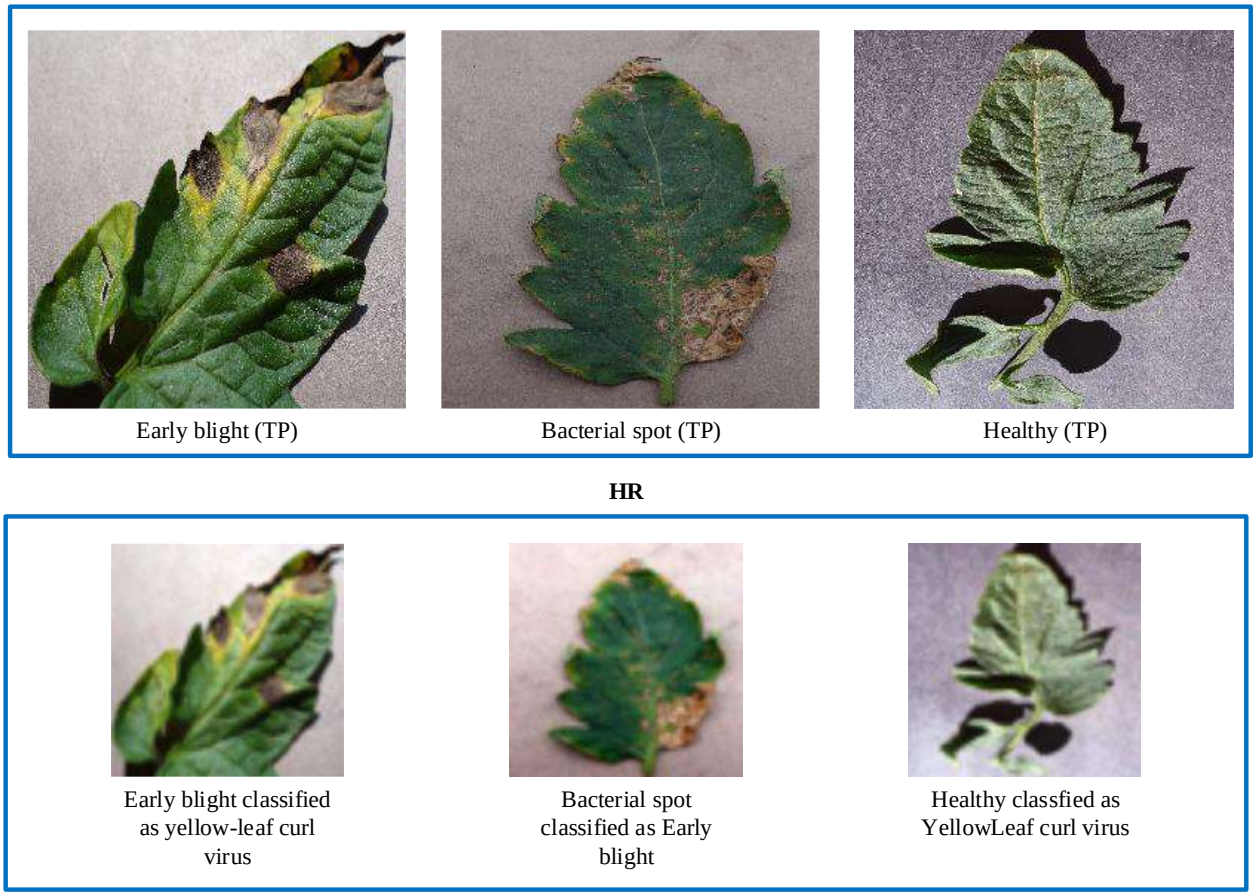


Fig 4.1 Example of images that are correctly classified using HR and erroneously classified using LR.

classification performance [20]. In the same vein, many studies have used SR algorithms to improve the classification performance of LR crop diseases images, but the obtained results were inferior by a large margin to the reference HR images classification accuracy.

In order to overcome the abovementioned limitation, the optimization and the improvement of current SR networks with high performance, to reconstruct the detailed information of HR images, would not only reduce the network complexity but also improve the classification accuracy of SR images of crop diseases.

Proposed network architecture

To this end, we introduce a novel SR architecture named Wider-activation for Attention-mechanism based on Generative Adversarial Network (WAGAN) to improve the visual quality of LR crop diseases images. The WAGAN takes an LR image as an input to generate an SR image at the output. The proposed method is composed of three main parts; a generator network, discriminator network, and adversarial loss. The generator has a core residual block named Wider Activation for Residual Channel Attention Block (WARCAB). This latter contains a linear low-rank convolution block for better feature extraction of the LR components, followed by an attention feature function that essentially concentrates more on valuable high-frequencies. The discriminator network makes the distinction between the real HR and the generated SR, while the adversarial loss helps the generator to focus more on edges, textures, and other specifications that are the main difference between the photo-realistic images and SR images without the adversarial loss. Due to the efficient design of WARCAB and the adversarial loss, the WAGAN focuses more on edges, textures, and other valuable information, which are the key information, needed for the classifier to recognize the disease. As a consequence, our proposed network improves the transformation of the inadequate LR crop images with scale ×4 into better SR images with low complexity. To assess our proposed method on plant leaves, we compare it with LR, SR of the state-of-the-art approaches, and HR images in terms of classification accuracy. The experiments are conducted on tomato crop diseases of the PlantVillage dataset, which contains 14 types of crops, and also tested on the PlantDoc dataset, using an accurate classifier.

The extensive experiments show the efficiency of SR methods in improving the classification accuracy of LR images compared to the state-of-the-art. It shows that the WAGAN outperformed the other approaches and is less than HR images by only 0.18% in terms of classification accuracy.

3. Related work

3.1 Image Super-Resolution networks

For the last few years, deep learning has become the trending investigation domain in the field of computer vision and image processing [105], [106]. The leading work was done by Dong *et al.* [44] on deep-learning-based Single image Super-Resolution (SISR). They achieved greater results by proposing a Super-Resolution Convolutional Neural Network (SRCNN) framework. Moreover,

the Efficient Sub-Pixel CNN (ESPCN) [107] reduced the computational and memory complexity, by increasing the resolution from LR to HR only at the end of the network, which led to better accuracy compared to SRCNN [44]. These studies were conducted on shallower network design with limited performance in SISR.

Recently, different designs have demonstrated better results with deep networks in SISR. For example, Lim *et al.* [108] developed an Enhanced Deep Super-Resolution (EDSR) and Multi-scale Deep SR (MDSR) networks to improve performance after eliminating redundant blocks in residual networks [36]. By expanding features before the Rectified Linear Unit (ReLU) activation (wider activation), Yu *et al.* [109] showed promising results regarding generated SR images with the use of linear low-rank convolution. Besides, they employed weight normalization, introduced in [110], after every convolution layer which led to superior accuracy with minimal design complexity compared to EDSR [108]. In another powerful design, Zhang *et al.* [111] introduced Residual Channel Attention Networks (RCAN) that utilized a very deep network based on Residual In Residual (RIR) to pass excessive low-frequency components over multiple skip connections and high-frequency through residual blocks. The authors obtained superior results compared to the state-of-the-art approaches. However, the network had a high number of parameters. In our work, we set a new RIR block to form a low-complex RCAN named Wider Activation for RCAN (WARCAN), which is the Generator in WAGAN, by eliminating the high computational 3×3 convolution in RCAN [111] and replacing it with a linear low-rank convolution with a wider activation of WDSR-B [109]. The latter contains a stack of two 1×1 convolutions (ReLU activation after the first layer) and adopts a weight normalization layer after every convolution.

Numerous SR networks have shown a large improvement in terms of the visual quality of SR images. For resembling realistic images, Ledig *et al.* [45] introduced the SR Residual Network (SRResNet) into the GAN (SRGAN) to generate SR images. Experiments demonstrated the potential of the SRGAN that remarkably improved the visual quality of SR images over the compared methods. Based on similar work, Wang *et al.* improved the SRGAN into Enhanced SRGAN (ESRGAN) [46]. The authors enhanced the network generator by introducing the RIR Dense Block (RRDB) in a very deep network with a high number of parameters that resulted in significant visual quality compared to SRGAN. Several works have adopted the attention mechanism, as introduced in [111], into the GAN [112]–[114]. In our design, we used the attention

mechanism based on GAN to force the network to focus more on edges, textures, and other valuable information with a less complex design in order to improve the visual quality.

3.2 Super-Resolution for improving crop diseases classification accuracy

For limited agriculture datasets, many studies have tried to improve the HR classification accuracy with data augmentation [115] using SR algorithms with GAN [116]–[121]. For instance, Yilma *et al.* [122] employed a GAN to augment the tomato leaf diseases dataset (PlantVillage dataset) in order to improve the HR classification accuracy where the proposed approach failed to generate some images. The authors yielded a classification accuracy enhancement of 5% after data augmentation. All the aforementioned studies used the generated SR images with the existed HR images to augment the dataset size in order to use it for the training process. Instead of using data augmentation to improve the HR images accuracy, first, we transform the downscaled HR (LR) into SR images. Then, we directly train the classifier with the available dataset (HR images) and compare the SR images with the LR and HR images in terms of classification accuracy.

Nowadays, several studies have taken advantage of SR methods to improve the LR images classification accuracy in agriculture. For example, Yamamoto *et al*. [20] proposed an algorithm to reconstruct 9 types of SR tomato leaf diseases images. Their studies were conducted on a previous version of the PlantVillage dataset (currently unavailable) using SRCNN [44] with five scale factors (from 2 to 6). They demonstrated that the classification accuracy, using AlexNet [19], achieved on SR images was superior to classical SR methods and those on LR images. Due to the low parameters used in SRCNN, the obtained results were not sufficient (approximately 80% on scale ×4) for the classification of tomato leaf diseases (from scale ×4 to ×6), compared to HR ones (around 94%). In another work, the authors proposed a Generative Adversarial Network with Dual-Attention and Topology-Fusion mechanisms called DATFGAN [114]. To transform the LR images into SR images, their experiments were conducted on the Plant Disease Recognition Competition of the AI Challenger 2018 dataset with 27 different categories of crop diseases. They achieved slightly superior results but they did not specify the downscale factor used in their experiments. In addition, they only compared their results to LR and other state-of-the-art approaches and not the reference HR images. In our study, we focus only on tomato leaf diseases from the PlantVillage dataset on a scale factor of ×4 and compare the classification accuracies results of LR, SR, and HR images. In a similar study, Wen *et al.* [123] proposed to enhance the accuracy of tomato crop

diseases classification with the Plant Village dataset [8] using a fine-tuned large network named Enhanced SR Generative Adversarial Network (ESRGAN), developed in [46], to recover realistic crop information SR images. Instead of upscaling the LR to the original HR size (256×256) presented in the dataset, the authors proposed to upscale the crops images from two different sizes of LR 16×16, 32×32 to SR images of 64×64 and 128×128, respectively. The authors achieved a classification accuracy of 89% for 64×64 and 90% for 128×128, and they outperformed the state-of-the-art approach [20]. The authors used a very large model with 16M of parameters, but they did not recover the original size of the presented images (256×256) in the dataset, and they also used a classic classification model (VGG16) that does not give a significant result compared to the actual classification models. In our work, we compared with [20], [123] studies on the PlantVillage dataset [8]. We proposed a WARCAN (WARCAN with 50 RIR as a generator) to build the WAGAN with only 4.5M of parameters. The experiments are conducted on scale ×4 with respect to the original size (256×256). The proposed method achieves superior results compared to the state-of-the-art [20], [123] and with lower complexity than ft_ESRGAN [120]. Additionally, we used the DenseNet-121 [37] instead of VGG16 and AlexNet in classification for better evaluation of the resulting SR images.

The abovementioned studies have proved the potential of SR methods for agricultural objectives. However, the use of an improved SR design would enhance the crop images quality as well as the classification accuracy. Therefore, we introduce a Wider-activation for Attention-mechanism based on Generative Adversarial Network (WAGAN) with low complexity based on attention mechanism in GAN to obtain a realistic image quality and improve the classification accuracy of SR tomato crop diseases images of the PlantVillage and PlantDoc datasets. Also, we employed the DenseNet-121 for the assessment of the LR, SR, and HR images in terms of classification accuracy.

4. Proposed method

The SR proposed method is introduced with descriptive details in this section. The purpose of SISR is to construct an HR image from an LR image (I_{LR}). The targets are the HR image (I_{HR}) and the estimated image (I_{SR}). The SR impact is based on the similarity between I_{SR} and I_{HR}. Our

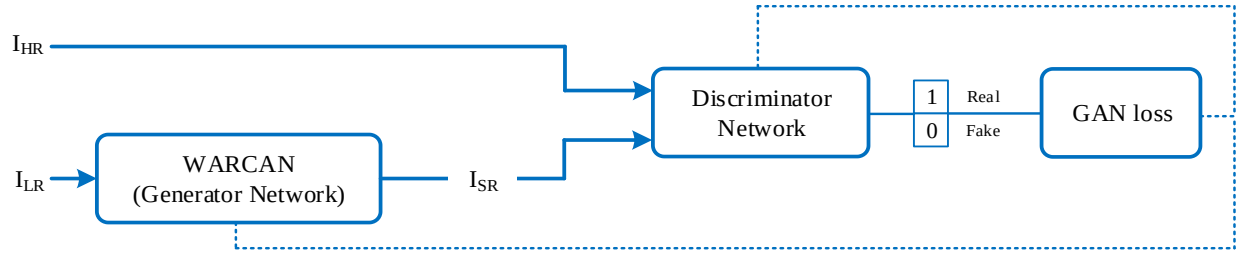


Fig. 4.2 Overview of the WAGAN

proposed method is a representation-learning model that aims to generate a super-resolved image I_{SR} using the WAGAN function (H_{WAGAN}) on the I_{LR} image:

$$I_{SR} = H_{\mathrm{WAGAN}}(I_{LR}, \Theta) \tag{4.1}$$

where, Θ designates the model parameters, learned with an iterative optimization process.

Several literature studies have adopted the GAN for the learning of an SR network [45], [46], [124] where the SR network (i.e., the generator) is associated with a discriminator. Fig.4.2 shows the overall architecture of the WAGAN. However, the WARCAN (as a generator) and the discriminator network are used to design the WAGAN network. The WARCAN represents the I_{SR} generator in our network. This network can efficiently extract high-level features concurrently from the I_{LR} image and built-in visual attention mechanism. Based on the I_{HR} and generated I_{SR}, the discriminator tries to distinguish whether the I_{SR} is real (1) or fake (0) compared to I_{HR}. Additionally, the GAN loss function is used to optimize the WAGAN. First, we introduce the Wider Activation for Residual Channel Attention Block (WARCAB), which is the core of WARCAN. The WARCAB is the improved layout of the Residual Channel Attention Block (RCAB) inspired by Yulun Zhang *et al.* [111] in RCAN. Second, we briefly describe the Discriminator block and the perceptual loss that are already introduced in [45].

4.1 Generator network architecture

Fig. 4.3 shows the WARCAN architecture, which is the generator network of the WAGAN. This network is a combination of three parts. An Identity Branch (IB), the WARCABs to form a residual path, and an Upscale module (UP). The generator network has several parameters; namely *M* the number of WARCABs residual blocks, *S* the scaling factor. The LLRC block parameters are *e* and *l* called, respectively, the expansion and reduction factors. The attention feature function parameter *r* known as the reduction factor. Additionally, *n* is the number of filters common to the LLRC block and the attention feature function. To extract the shallow features from the LR input in the residual path, we use only one convolutional layer (Conv) at the input of the WARCAN. The latter is then passed through *M* residual blocks (WARCABs), and the last output of the residual blocks is gone through a convolutional layer to adapt the depth of the feature maps to $3 \times S^2$, before the UP module, where 3 is the number of channels in the RGB images and *S* the scaling factor. We additionally employed the pixel shuffle function in ESPCN [107] as the up-scale module to improve the speed and the parameter reduction. The IB takes directly the I_{LR} to pass via the same last convolution layer and pixel shuffle function where the output of the latter is joined in the end via an element-wise summation to estimate the I_{SR} image.

Wider Activation for Residual Channel Attention Block

The RCAN [111] exhibited a high performance with attention mechanism in SISR. The WARCAB (b) is designed to focus on significant high-frequency components and thus allowing the redundant low-frequency information to pass through the network via the residual connections. Indeed, the RCAB (a) is composed of a stack of two 3×3 convolution filters and a feature attention sub-module, which is discussed in Section 3.1.1.2. In fact, the output of the convolution operation is then passed via the feature attention sub-module. This latter is weighted up to track the most promising high-frequency components. However, they often do not take advantage of contextual

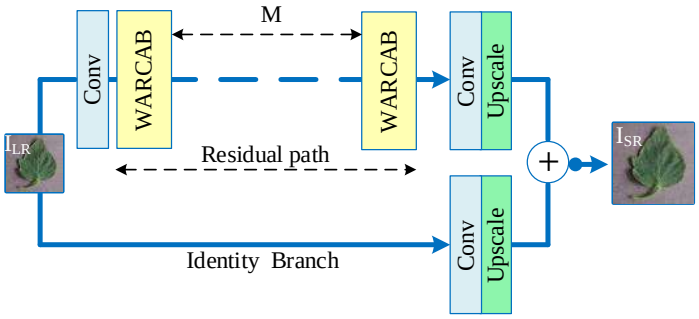


Fig. 4.3 Overview of the WARCAN (generator) architecture: Identity Branch (IB), residual path (WARCABs) and upscale module.

information outside their limited region of view. Therefore, we propose to tackle this problem with the use of the Linear Low-Rank Convolution (LLRC) to improve the design of RCAB. More formally, we focused on exploring more relevant features before passing them through the attention function which leads to better performance on the WARCAB more than the RCAB design, which was not taken into consideration. The LLRC, dashed box with green in Fig 4.4, is explained in detail in Section 3.1.1.1. In addition, the only changes made to the attention feature function of RCAB (dashed box with red in Fig. 4.4) were all about adding the Weight Normalization (WN) layer after every convolution. Finally, the WARCAB output F_i is equal to

$$F_i = F_{i-1} + H_{FA}(X_{i-1})\, X_{i-1} \tag{4.2}$$

where, F_i and H_{FA} are the output of WARCAB and the attention feature function, respectively, and X_{i-1} the WARCAB output of the LLRC. More formally, F_{i-1} is the input of the WARCAB.

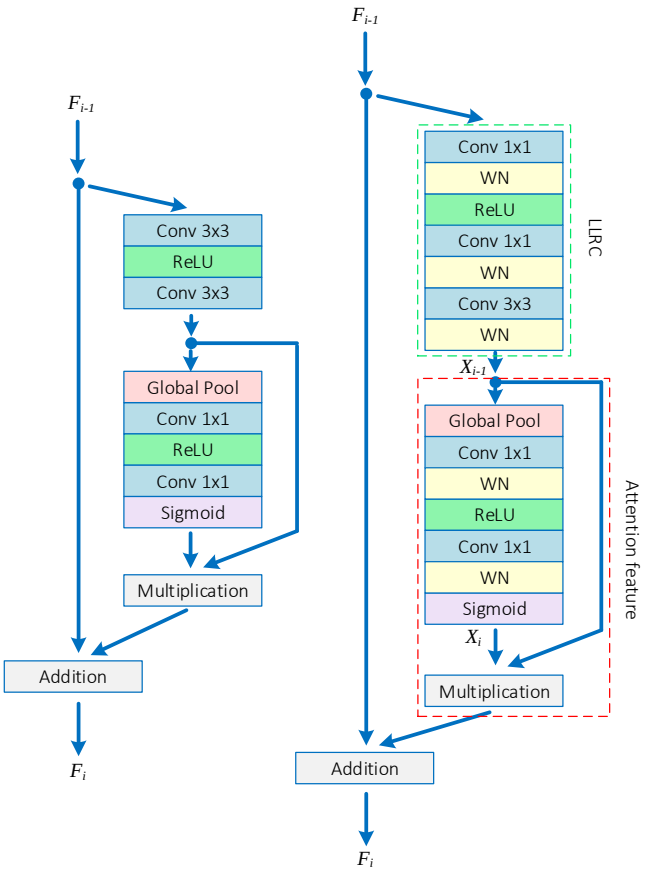


Fig. 4.4 Comparison of Residual Channel Attention Block (RCAB) in original RCAN [111] and ours.

a) Weight Normalization: Weight Normalization (WN) [110] is based on weights re-parameterization that decouples the weight magnitude from its direction. However, this process implies the normalization of weights after the convolution layer F_{conv}:

$$o = w \,.F_{conv} + b, \;\; with \;\; w = \frac{g}{||v||} v \tag{4.3}$$

where o is the output and F_{conv} is the input of the WN layer. w denotes the weight and b the bias. g, v the magnitude and the direction of w, respectively ($||w|| = g$ independent of parameters v). $||v||$ is the Euclidean norm of v. the WN calculates the weight from the magnitude and direction that leads to a better convergence of the optimization process. Hence, the WN layer speeds up the network convergence that in turn improves the training and testing accuracies.

b) Linear Low-Rank Convolution block

The Linear Low-Rank Convolution (LLRC) block is employed to explore more features before the attention feature function [109]. Fig. 4.5 shows the structure of the LLRC block. This block consists of two stack convolutions with 1×1 kernels and one convolution with 3×3 kernels. The two-stack convolutions are separated by the non-linearity ReLU function before the second convolution layer. Moreover, the number of filters (n) in the first convolution is expanded by a factor e, where $e \in \mathbb{N}^*$ ($e > 1$). Then, the number of filters in the second convolution is reduced by a factor of l, and $0 < l < 1$, where $l \in \mathbb{R}^*+$. However, the last convolution keeps the actual size n as the filter number. Besides, we adopt a weight normalization (WN) layer after each convolutional layer to improve training and testing accuracy. We can express this operation as

$$X_{i-1} = F_{WN}(F_{i,conv3}(F_{WN}(F_{i,conv2}(F_{WN}(F_{i,\sigma}(F_{i,conv1}(F_{i-1}))))))) \tag{4.4}$$

where, X_{i-1} and F_{i-1} represent the output and the input of the i^{th} LLRC block, respectively. F_{WN} is the weight normalization layer. $F_{i,conv1}$ and $F_{i,\sigma}$ are the output of the first convolution expanded layer with filter number of $e \times n$ and the output of the ReLU function applied to $F_{i,conv1}$, respectively. $F_{i,conv2}$ is the reduced convolution layer output with a filter number equal to $l \times n$. $F_{i,conv3}$ is the final output of the convolution layer with n number of filters.

The expansion on the first convolution layer aims to have a maximum of feature maps before the nonlinearity function, the second convolution is used to decrease the feature dimensions and the final convolution operates spatial-wise feature extraction. As a result, the LLRC block focuses more on the extraction of valuable components to feed them into the attention feature function.

c) Attention feature function

As discussed in [111], the attention feature function allows the network to further concentrate more on informative features. However, the LLRC block X_{i-1} is then the input of the attention feature (H_{FA}). The latter output X_i is given by

$$X_i = F_{WN}(W_2 * max\ \left(F_{WN}(0, W_1 * X_{i-1} + B_1)\right) + B_2) \tag{4.5}$$

where W_1 and W_2 denote the filters with size $f_h \times f_w$ each, B_1 and B_2 the biases, '*' the convolution function and F_{WN} the weight normalization layer. The number of filters is equal to n. Consequently, the whole low-frequency components in $\boldsymbol{X}_{i-1}$ are able to pass via the residual connection. The attention feature may allow the network attention to emphasize more on valuable high-frequency signals. To this end, the use of global average pooling permits the attention module to take the feature-wise global spatial information into the feature descriptor. Accordingly, from the input $\boldsymbol{X}_{i-1} = [x_1, \ldots., x_C]$ where, C is the features maps with shape $H \times W$, we extract $z_C \epsilon R^C$ the feature statistics using the following formula

$$z_C = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{k=1}^{W} X_{i-1}(i, k) \tag{4.6}$$

where, statistics of the feature z_C are considered as a collection of descriptors, and contribute to expressing the whole image. As shown in Fig. 4.5, the output of the global pooling function is then operated using a stack of two convolutional layers where each convolution is followed by a weight normalization layer. Then, it is passed by a ReLU and a sigmoid activation function, respectively.

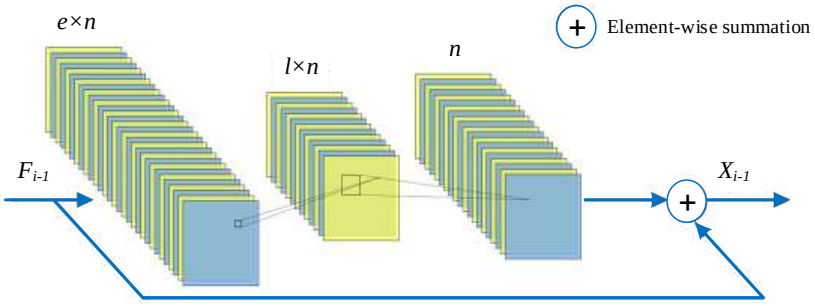


Fig. 4.5 Structure of the Linear Low-Rank Convolution (LLRC).

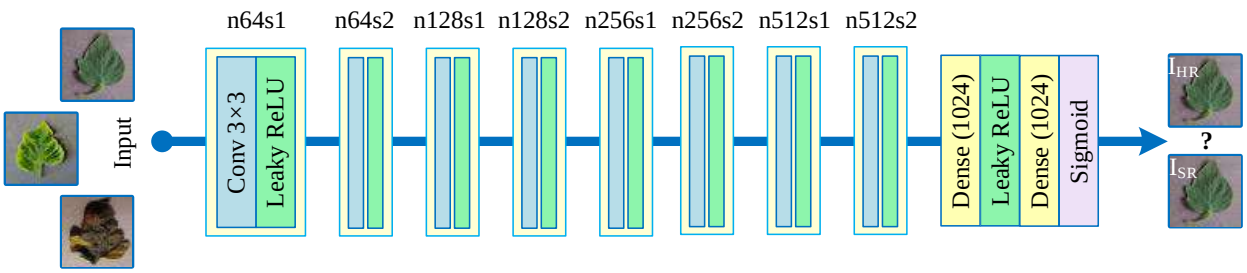


Fig. 4.6 Discriminator network.

The stack of two convolutional layers permits the production of a non-linear mapping function that may deeply capture feature-wise dependencies from the aggregated information extracted using the global pooling operation. The initial convolutional layer decreases the feature dimension with a factor of *r*, and subsequently, the second layer recovers the original size and rescales its values from 0 to 1 with a sigmoid function in a non-mutually exclusive relationship.

Then, the WARCAN is optimized with a loss function. Several loss functions have been introduced, such as L2 [44] and L1 [125]. To show the potential of the WARCAN, we optimize with L1 loss function as investigated in previous works [125] for better convergence. Given a training set $\{I_{LR}^{i}, I_{HR}^{i}\}_{i=1}^{K}$ that contains (K) LR inputs and their HR counterparts. The goal of training the WARCAN is to minimize the loss function L1 as

$$L_1(\Theta) = \frac{1}{K}\sum_{i=1}^{K}\left\|H_{WARCAN}(I_{LR}^{i}) - I_{HR}^{i}\right\|_1 \quad (4.7)$$

where Θ denotes the parameters set of the proposed model. The loss function is optimized using Adam [126]. More details of the training are shown in Section 4.1.

4.2 Discriminator network architecture

The discriminator network is trained for the distinction between the real I_{HR} and the generated I_{SR}. Fig. 4.6 shows the discriminator network architecture. The discriminator has two main parameters; *n* and *s* are the number of filters and the stride, respectively. Referring to the architecture presented in Ledig *et al.* [45] and the VGG16 network [70], the discriminator consists of eight stack convolution layers followed by a LeakyReLU. Additionally, the number of filters (denoted by *n* in Fig. 4.6) is incremented by a factor of 2; i.e., starting from n = 64 and ending with n = 512. Note that, we eliminated the batch normalization layer as discussed in [127]. To reduce the image resolution each time, the convolution stride (noted by *s* in Fig. 4.6) is employed. Finally, the output

of the last convolution layer is then fed into two dense layers to obtain the probability for the sample classification function using a sigmoid function.

The discriminator and the generator WAGAN work competitively, the discriminator tries to distinguish between I_{SR} and I_{HR} while the generator tries to fool the discriminator. Both networks get improved by each other after calculating the perceptual loss for every iteration. Calculating the perceptual loss function depends on the generator and the discriminator outputs. The WAGAN is optimized using the perceptual loss L, proposed in [45], which can be formulated as

$$L = L_{content} + 10^{-3} L_{GAN} \tag{4.8}$$

where, $L_{content}$ and L_{GAN} denote the content loss and the adversarial loss, respectively.

5. Experimental results and analysis

5.1 Training Settings

We used the Peak Signal-to-Noise Ratio (PSNR) in dB and the Structural SIMilarity (SSIM) [47] index as criteria to measure the performance of the proposed model.

(1) *Data preparation:* The tomato crop disease images from the Plant Village dataset were used for training and to compare the proposed model with the state-of-the-art methods; also the PlantDoc was used only for testing [7], [8]. Images were randomly divided into training samples (80%), validation samples (10%), and test samples (10%). All experiments were applied with an upscale factor ×4 between LR, SR, and HR images. The HR images were downscaled using MATLAB *imresize ( )* function with Bicubic kernel. The reference HR images dimension are 256×256 pixels. Since a larger patch size required more computing resources and training time, we used the Tensorflow tf.data API [128] to cache and prefetch the dataset as well as to optimize and speed up the training process. Furthermore, the batch size was set to 16, and HR images were flipped and rotated for data augmentation.

(2) *WARCAB implementation:* Following the RCAN settings [111], all the number of filters n = 64. For the WARCAN parameters, experiments showed that the WARCAB improves the accuracy when the reduction factor l is equal to 0.6, 0.7 or 0.8 in the LLRC block. The couple $l = 0.8$ and expansion factor $e = 8$ yields the most efficient accuracy (e and l are fixed in further experiments for the three proposed algorithms) that assesses the work given in [100] for $6 \leq e \leq 9$. The first

layer has 512 filters ($e \times n$), and the second layer has 51 filters ($l \times n \in \mathbb{N}^*$). The convolution layer of the attention feature in WARCAB channel-downscaling had n/r filters where r is the reduction ratio. However, we followed the same design, as fixed in [111], where $n/r = 4$ and $r = 16$.

(3) *Network design:* We designed the RCAN [111] with 7 RCAB. Table 4.1 demonstrates the three versions of the proposed networks. The WARCAN consists of 5 WARCAB blocks ($M = 5$) in order to compare with the RCAN. On the other hand, The WAGAN- has the WARCAN ($M = 5$) as a generator. The WAGAN used the WARCAN with 50 WARCAB ($M = 50$) as a generator.

(4) *Training process:* The WARCAN ($M = 5$) is first trained on the PlantVillage dataset and then pre-trained as a generator in WAGAN-. For the WAGAN training process, the WARCAN with 50 WARCAB ($M = 50$) is first trained on DIV2K [129] dataset, then pre-trained as generator in the WAGAN. After the pre-training, the discriminator receives quite good SR images to concentrate more on texture discrimination. During the training of the WAGAN (and also for WAGAN-), we alternate updates of the generator and the discriminator networks after every iteration until the model converges. Note that, the parameters of the discriminator are held fixed as mentioned in Section 3.2. Finally, the proposed model was optimized with Adam [126] by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-7}$. We set the initial learning rate for both WARCAN and WARCAN with 50 WARCAB to 1×10^{-4} that was halved at every 2×10^5 iterations. For WAGAN- and WAGAN, the first 10^5 update iterations were trained with a learning rate of 10^{-4}, and the second 10^5 iterations at a learning rate of 10^{-5}.

Table 4.1 The three versions of the proposed network.

Name of the model	Number of WARCAB block (M)	Based on GAN?
WARCAN	5	No
WAGAN-	5	Yes
WAGAN	50	Yes

Computations were conducted using Python 3.8 and Cuda 10.1 on Windows 10 system. We implemented the models with the Tensorflow framework (version tensorflow_gpu-2.3.0). The total number of parameters of every network is presented in Table 4.1. The training process was conducted with an NVIDIA GeForce GTX 1060 GPU.

5.2 Qualitative and Quantitative evaluation of the Generated SR images compared to the state-of-the-art

Due to the absence of an effective standard metric for perceptual quality, we show some examples of qualitative results of LR, the reconstructed SR images of our and the state-of-the-art networks, and HR with the available standard metrics (PSNR and SSIM) with an upscale factor of 4 in Fig. 4.7, 4.8 using PlantVillage and PlantDoc datasets (PlantDoc dataset is only used for testing). We compared our final networks to several state-of-the-art networks including Bicubic, SRCNN [20], ESPCN [107], EDSR [108], WDSR-B [109], RCAN [111], and ft_ESRGAN [123]. However, we retrained these approaches under our experimental datasets for an equal and convincing comparison. We measured the PSNR as well as the SSIM of a test image from tomato early blight disease. In terms of PSNR and SSIM values, the WARCAN outperformed all the compared networks, only in Fig. 4.7 (first figure) the WAGAN obtained the highest PSNR value. In terms of visually pleasing, the WAGAN was able to recover the edge details of the early blight disease from the LR image, and the other parts of the leaf that looks realistic compared to the reference HR image. However, the ft_ESRGAN [123] could recover the details of the disease but the realistic details around the disease were not realistic as the WAGAN. On the other hand, the WAGAN- reconstructed the details with slight noise. However, the WARCAN and RCAN [111] networks could recover the edges of the leaf better than the EDSR [108] and WDSR-B [109]. For the ESPCN [107] and SRCNN [20], it was hard to identify the details of the global disease characteristics. Due to the shallow network design and the limited performance of the SRCNN in SISR, the WAGAN yields more visually realistic textures and edges than those produced by SRCNN [20]. The Bicubic could remove just a few blurring effects from the LR image. This illustrates that the WAGAN has a more natural view and powerful representational ability and can extract more complicated features from the LR space.

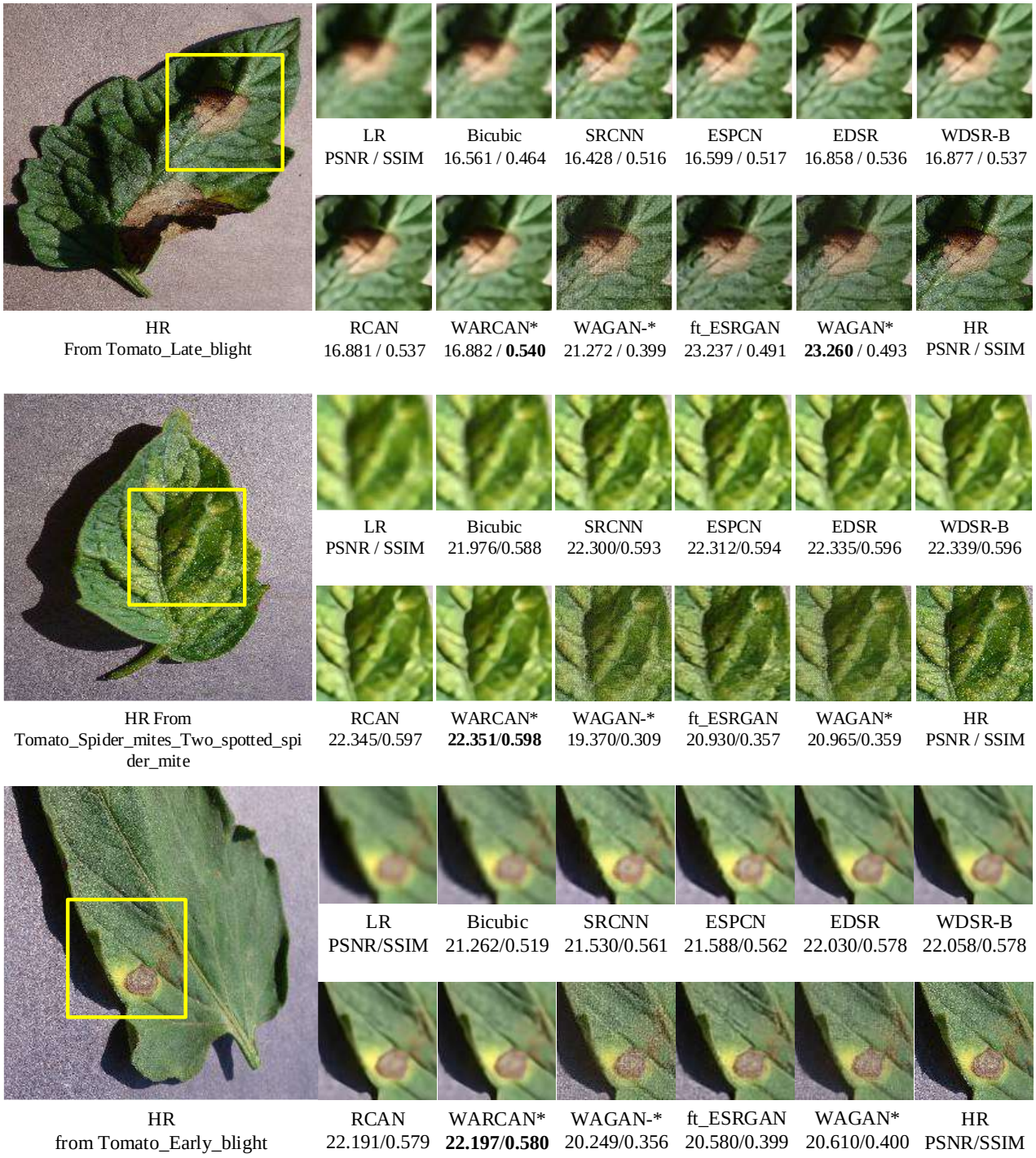


Fig. 4.7 ×4 reconstruction results of our proposed network compared with LR, state-of-the-art SR images, and reference HR with PlantVillage. (*) refers to our proposed network

Fig. 4.8 ×4 reconstruction results of our proposed network compared with LR, state-of-the-art SR images, and reference HR with PlantDoc. (*) refers to our proposed network

Table 4.2. Test results (PSNR (dB) and SSIM) of our and state-of-the-art SR methods on the Plant Village as well as PlantDoc datasets.

	PlantVillage		PlantDoc	
Method	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
WAGAN (ours)	24.877	0.488	26.798	0.689
ft_ESRGAN [123]	24.311	0.467	26.512	0.677
WAGAN- (ours)	24.151	0.459	26.409	0.672
WARCAN (ours)	**25.935**	**0.549**	**28.579**	**0.759**
RCAN [111]	25.913	0.547	28.545	0.758
WDSR-B [109]	25.891	0.546	28.467	0.755
EDSR [108]	25.869	0.545	28.465	0.755
ESPCN [107]	25.610	0.532	27.615	0.733
SRCNN [20]	25.543	0.530	27.599	0.732
Bicubic	25.164	0.497	27.046	0.695

For a more general comparison, we calculated the PSNR and SSIM for 1000 image samples, taken randomly from the test set for all SR approaches. Quantitative results are summarized in Table 4.2, the results showed that the WARCAN yielded the best results in terms of PSNR and SSIM. The WARCAN outperformed the other methods, with a difference of at least 0.022 (PSNR) and 0.002 (SSIM) in PlantVillage, and 0.034 (PSNR) and 0.001 (SSIM) in PlantDoc on a scale of ×4. Thus, we can conclude that the WARCAN obtained superior values compared to the other SR methods instead of achieving the best visual performance, which is the key feature for classification so that it will be investigated in the next section.

5.3 Diseases Classification

Table 4.3 Accuracy of tomato leaves diseases classification for LR, HR, and SR images with magnification scales 4.

	PlantVillage	PlantDoc
Method/Images	**Accuracy**	**Accuracy**
HR (images)	97.81%	96.50%
WAGAN (ours)	97.63%	96.30%
ft_ESRGAN [123]	97.36%	96.00%
WAGAN- (ours)	94.26%	93.10%
WARCAN (ours)	88.53%	87.40%
RCAN [111]	88.35%	87.10%
WDSR-B [109]	88.08%	86.80%
EDSR [108]	87.80%	86.60%
ESPCN [107]	85.89%	84.70%
SRCNN [20]	85.45%	84.20%
Bicubic	77.89%	76.60%
LR (images)	63.69%	62.40%

In this section, we performed crop diseases classification experiments on tomato leaves to evaluate whether the reconstructed SR images, using the WARCAN, include valuable information for classification. Table 4.3 shows the classification accuracies results of the LR, SR, and HR images on the tomato leaves diseases test samples. Meanwhile, to obtain better classification performance and less computation time, we fine-tuned the DenseNet-121 [37] classification network, trained on ImageNet, with tomato leaves images of the Plant Village and PlantDoc datasets [7], [8]. Therefore, we made a comparison, in terms of classification accuracy, of the proposed network with the LR, HR, and SR images obtained through the use of Bicubic, SRCNN [20], ESPCN [107], EDSR [108], WDSR-B [109], RCAN [111], and ft_ESRGAN [123]. In this

experiment, the total number of the test samples is 1110 for PlantVillage and 90 images for PlantDoc, where 111 and 90 images, respectively, are randomly selected from every class. The experiments are conducted with a downscaled factor of 4. Thus, we obtained LR images with 64×64 for a factor scale of ×4. Then, we generated the SR images (256×256) using Bicubic, SRCNN [20], ESPCN [107], EDSR [108], WDSR-B [109], RCAN [111], and ft_ESRGAN [123] and our proposed networks. Finally, we classified them with the DenseNet-121 [37] fine-tuned model. To do this, we changed the setting of the input layer dimension of DenseNet-121 to 256×256 and the number of output classes to 10 in order to meet the dataset image dimensions. Furthermore, we set Adam optimizer for optimization with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 0.1$. The learning rate was set to 0.002 and decreased to half after 30 epochs.

From Table 4.3, it can be seen that the classification accuracy of the WAGAN (highlighted in blue) is higher than those of the LR and other SR methods, and too close to the reference HR (highlighted in red). In the opposition to the best performance of the WARCAN in terms of PSNR and SSIM, the classification achieved by upscaling the images with WAGAN is superior to ft_ESRGAN [123], WAGAN-, WARCAN, RCAN [111], WDSR-B [109], EDSR [108], ESPCN [107], SRCNN [20], and Bicubic methods as well as LR. That is, the WAGAN achieved a classification accuracy of 97.63% (in PlantVillage) and 96.30% (in PlantDoc) that are smaller than the original HR images by at 0.18%, and higher than the ft_ESRGAN network by at least 0.27%, and by a minimum of 9% with respect to the other SR approaches and the LR (33.94%). Based on this, we can empirically conclude that the WAGAN was able to reconstruct SR images, closely similar to the reference HR images, with effective and detailed information for classification application. Particularly, the low accuracies obtained from the LR images reveal that these images lack valuable information required for disease recognition by DenseNet-121.

6. Discussion

(1) Tomato leaves disease classification performance: We confirmed the superior visual quality performance of WAGAN. For the classification, the study in [20] is based on the SRCNN to generate SR crop tomato images, while the AlexNet [19] is used to classify the LR and the resulting SR images. According to Table 4.3, the WAGAN leads to higher accuracy performance by a large margin (12.1%) than SRCNN in terms of the generated SR images classification. That is, meanly to the better SR quality of WAGAN. As a consequence, the combination of WAGAN with

DenseNet-121 showed a high performance compared to the one of SRCNN. Additionally, it can be efficient for crop tomato disease recognition and diagnosis.

The classification results (90.78%) obtained in [123] using VGG16 [70] classifier with the size of 128×128 were not sufficient compared to the HR (95.14%) with size of 128×128 (which was not the original size (256×256) presented in the PlantVillage dataset [8]). Therefore, our study demonstrated that the use of DenseNet-121 [37] instead of VGG16 [70], as investigated in [101], led to superior performance with the reference HR (97.81%) with the size of 256×256 even using ft_ESRGAN (97.36%) that was not introduced by Wen *et al.* [123].

(2) Network complexity Analysis: To investigate the network complexity, we made a comparison in terms of classification results with respect to the network number of parameters as shown in Table 4.4. We prefer to perform the comparison based on PlantVillage because PlantDoc has a limited dataset compared to PlantVillage. The network RCAN [111] contains 7 RCAB with a number of parameters of 0.6M whereas the WARCAN consists of 5 WARCAB with a total number of parameters of 0.4M. In terms of classification, the WARCAN outperformed the RCAN by at least 0.18% and even in terms of PSNR (0.022) and SSIM (0.002) values (as demonstrated in Table 3.5). Compared to the RCAB design, the LLRC design with the attention feature would not only improve the learning but also enhanced the attention feature performance by extracting more features before feeding them into the attention feature, and WARCAN has less number of residual blocks than the RCAN that led to a reduction in computational cost.

Table 4.4 Comparison of the proposed networks with the state-of-the-art networks in terms of classification results with respect to the network number of parameters.

Model	RCAN [111]	WARCAN (ours)	WAGAN- (ours)	Ft_ESRGAN [123]	WAGAN (ours)
Number of parameters (M)	0.6	0.4	0.4	16.6	4.5
Accuracy	88.35%	88.53%	94.26%	97.36%	97.63%

To illustrate the efficacy of the LLRC design, the first convolution layer produced the maximum of the feature maps before the ReLU function while, the second decreased the feature dimensions. The Last convolution operated the spatial-wise feature extraction that led to better extraction of the key information before feeding it into the attention feature function. Compared to the 3×3 convolution filters used in RCAN [111], the use of the 1×1 convolution filters remarkably reduced the filter space dimensionality with less information loss and faster computation [130]. In addition, the 1×1 convolution filters improved the network non-linearity to learn more complex mapping at finer levels in SISR. Moreover, the re-parameterization of weights with the WN layer considerably leads to better convergence and accuracy compared to RCAN architecture; which complies with the study for deep SR models [109]. As a result, the LLRC operated a better feature extraction helping the attention feature function to concentrate more on the valuable information, needed by the DenseNet-121 to recognize the disease. Consequently, with fewer parameters, the WARCAN design obtained a better classification accuracy (88.53%) compared to RCAN (88.35%).

Based on this low parameter count network, we proposed the WAGAN- and WAGAN for better visual performance. Besides, the WAGAN- obtained remarkable classification results (94.26%) but is less accurate than the deeper WAGAN design. Fig. 4.9 shows the accuracy and number of parameters of the WAGAN and state-of-the-art methods compared to the reference HR. The WAGAN has 4.5M number of parameters and is 3.6 times less complex than the ft_ESRGAN [123] (16.6M). Specifically, the WAGAN generator takes benefit of the WARCAB (as described in detail in the previous paragraph) with deeper architecture based on GAN to essentially concentrate on the

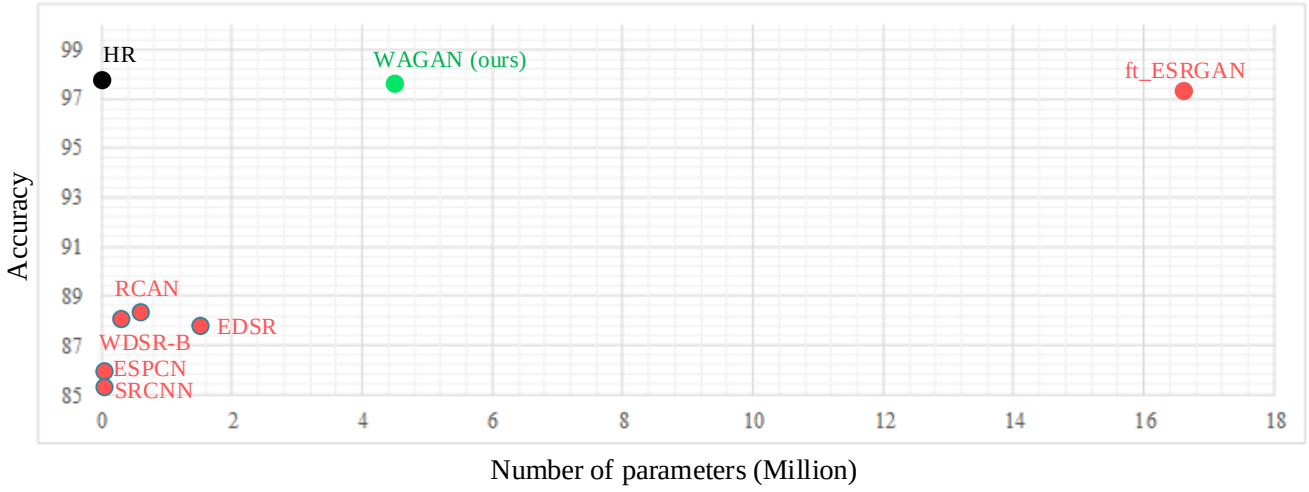


Fig. 4.9 Accuracy and number of parameters of the WAGAN (labeled with green) and state-of-the-art methods (labeled with red) compared to the reference HR. Results were evaluated on tomato leaves of the PlantVillage dataset.

content. On the other side, the adversarial loss focuses on helping the WAGAN generator to make the images look like natural ones. That is, it addressed the edges, textures, and other details that are the key difference between the photo-realistic images and SR images without the adversarial loss that is missing in the ft_ESRGAN design (based on 3×3 convolution filters). According to the above discussion, our study confirms the efficiency of the use of the 1×1 convolution and WN (in LLRC design) along with the attention feature function in the WAGAN; yielding an improved visual image quality and therefore a better classification accuracy. In effect, the WAGAN recovered more natural images than the ft_ESRGAN as well as achieving a closer classification accuracy to the reference HR images by only 0.18%. Therefore, the proposed network has a better perspective to improve the crop images diagnosis and analysis.

(3) HR, LR, and WAGAN classification analysis: To assess the WAGAN improvement of LR images, we compare the super-resolved images using WAGAN with some examples of images that are correctly classified using HR and erroneously classified using LR. The extensive results showed that the classification accuracy decreased by 34.12% (where the accuracy of LR is equal to 63.69%, and HR is 97.81%). Fig. 4.10 illustrates the HR, LR, and the resulted WAGAN images. The LR images look blurred and smooth compared to HR and SR images. The general shape of the leaf is distinguishable, but the image details are blurred. The lesions lost their specific visual characteristics and become hard to distinguish from other disease types, especially for mold leaf and target spots. However, the WAGAN recovered the details of the lesions and made the leaf edges more pronounced. For example, the infected region of the bacterial spot in the LR image becomes similar to early blight. Whereas, the WAGAN recovered all the lesion details and the DenseNet-121 recognized it as a bacterial spot disease. Fig 4.11 shows an example of corrected

Mosaic virus (TP)
HR

Mosaic virus (TP)
LR

Mosaic virus (TP)
WAGAN

Fig 4.11 Samples of images that are correctly classified using HR, LR and WAGAN.

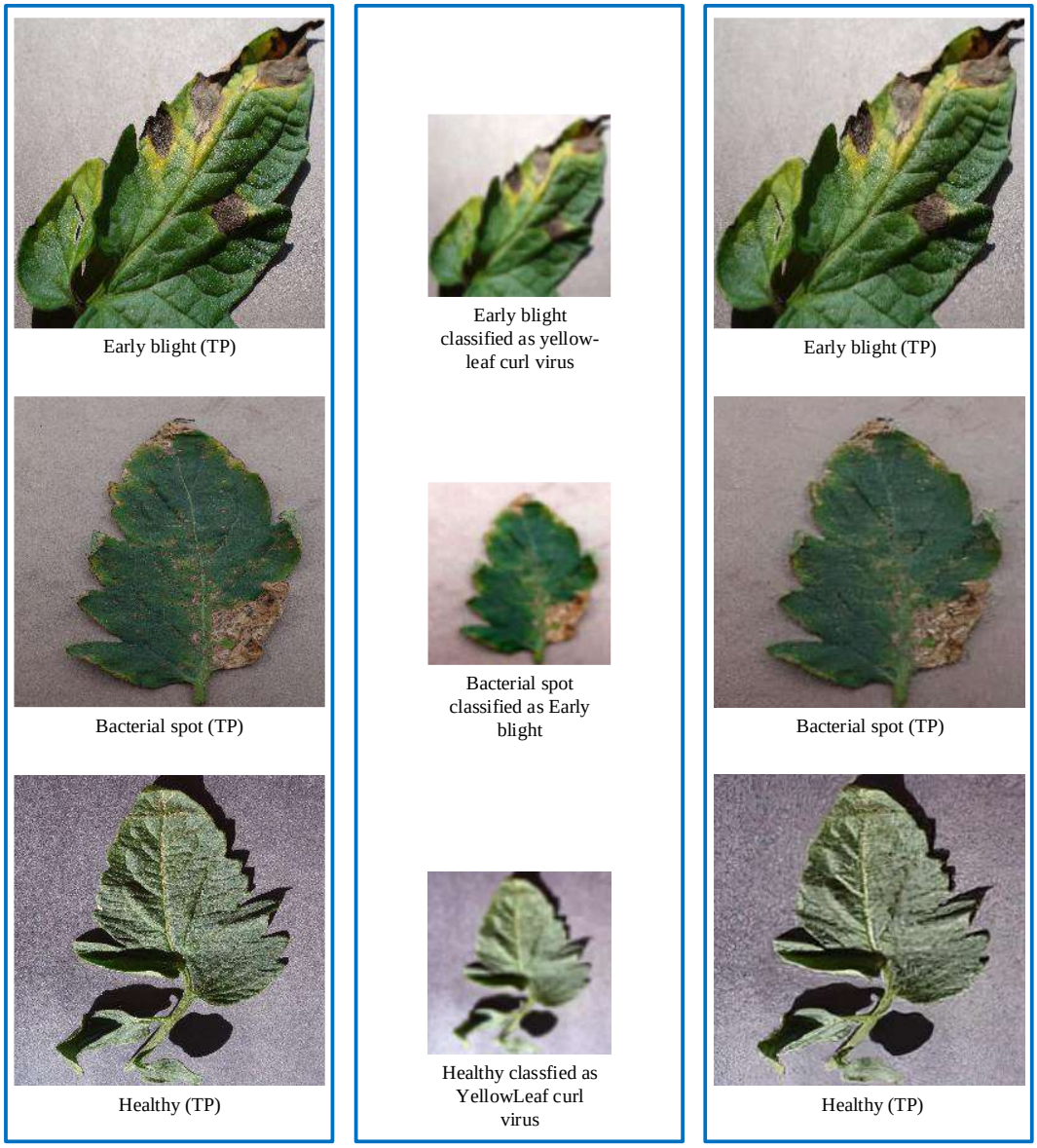


Fig 4.10 Samples of images that are correctly classified using HR, WAGAN and erroneously classified using LR.

classified HR, LR, and SR images. Concerning the LR, the DenseNet-12 can identify only the diseases that occur on the overall shape of the leaf such as mosaic (Fig. 4.11). Nevertheless, the model performed poorly with lesions that can occur in a small portion of the leaf. Thus, the SR images of WAGAN look realistic (similar to HR), which helps the DenseNet-121 recognize the diseases correctly.

(4) Classification performance: In order to investigate the classification accuracy in each class, we provide in Figs. 4.12-15. Quantitative evaluation using the confusion matrix for factor scale 4, The axes *x* and *y* indicate the ID of diseases in Table 3.3. We can see that the classification accuracy increases, progressively, from LR to deep learning SR and finally to HR. It is also clear that the WAGAN achieved the highest performance among the selected LR and SR approaches. More particularly, all the classes are easily identified for ft_ESRGAN and the WAGAN had a closer similarity to HR identification. However, the WAGAN- had less rate of identification.

Generally, WARCAN, RCAN [111], WDSR-B [109], and EDSR [108] networks are less recognized class 0 (Bacterial spot), class 5 (Spider mites), class 6 (Target spot), class 9 (healthy). However, ESPCN [107], SRCNN [20], Bicubic, and the LR confused the class 0 (Bacterial spot) with class 7 (Tomato yellow curl virus), class 4 (Septoria leaf spot) with class 3 (Mold leaf), class 5 (Spider mites) with class 7 (Tomato yellow curl virus), class 6 (Target spot) with class 2 (Late blight), and class 9 (healthy) with class 2 (Late blight).

The confusion matrix for DenseNet-121 identification of the LR tomato diseases is illustrated in Fig 4.12. The overall classification of the DenseNet-121 is not accurate. For Late blight (2), Mold leaf (3), and Tomato yellow curl virus (7) classes, the model recognized most samples. Moreover, the DenseNet-121 confuses the early blight (1) with the Late blight (2), Septoria leaf spot (4) with Mold leaf (3), and mosaic virus (8) with Tomato yellow curl virus (7) classes. The

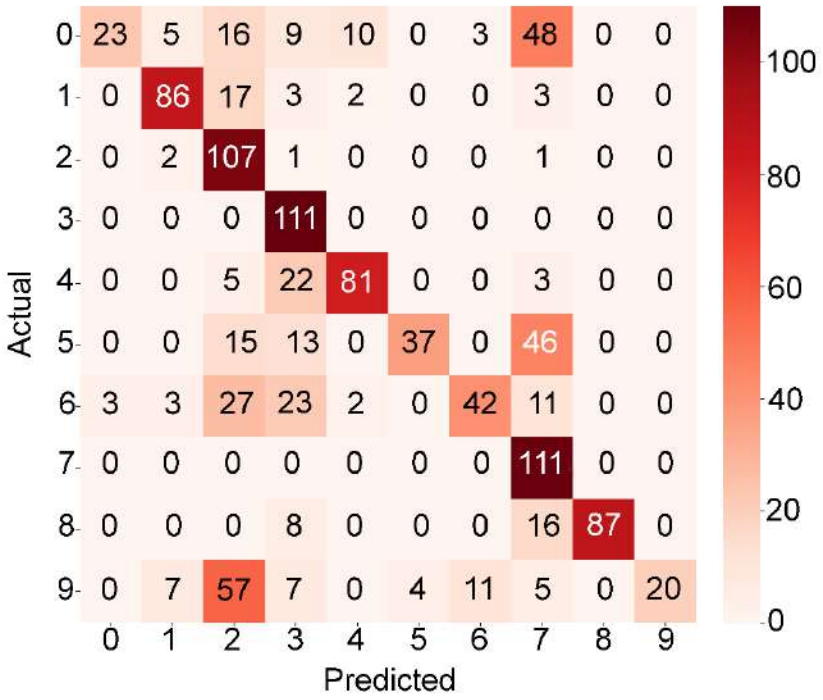


Fig 4.12 Confusion matrix of the DenseNet-121 diseases classification using LR.

model misclassifies four classes namely; Bacterial spot (0), Spider mites (5), Target spot (6), and healthy (9) classes. Generally, it confused these classes with Late blight (2) and Tomato yellow

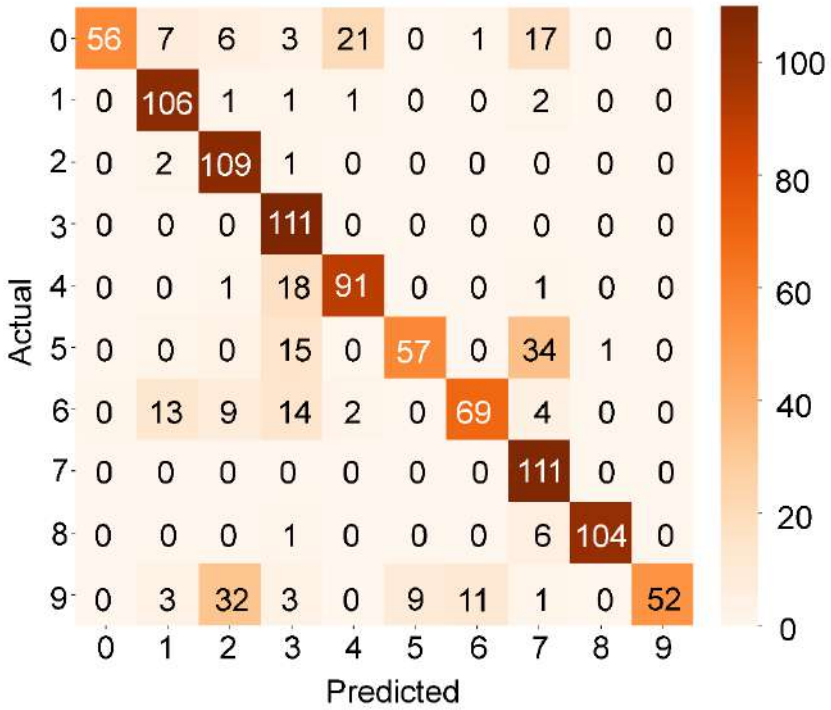


Fig 4.13 Confusion matrix of the DenseNet-121 diseases classification using Bicubic.

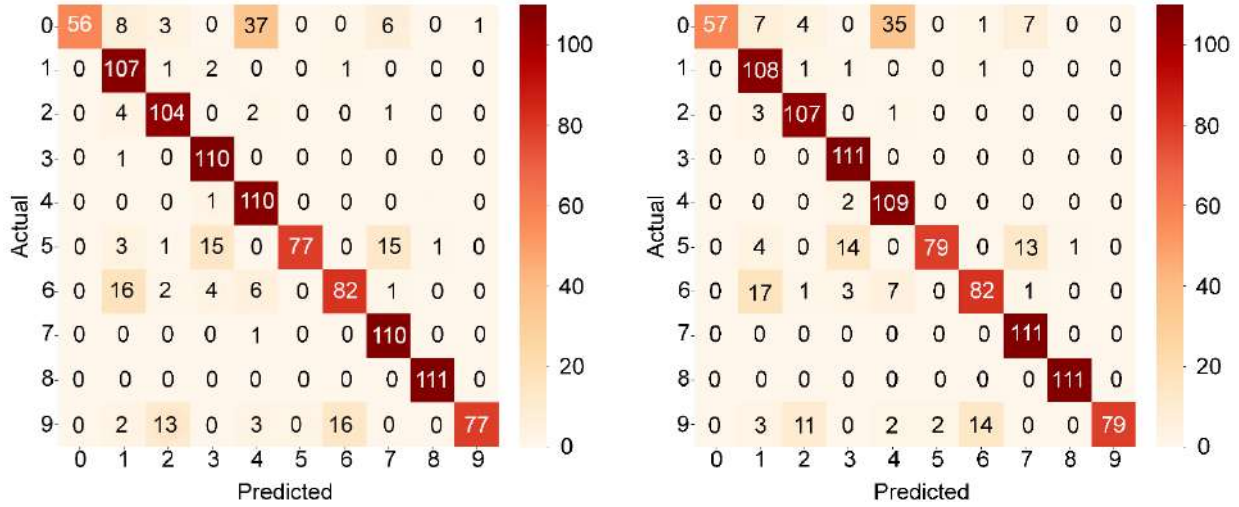


Fig 4.14 Confusion matrix of the DenseNet-121 diseases classification using SRCNN (left) and ESPCN.

curl virus (7).

For the classification performance of DenseNet-121 using Bicubic images (Fig 4.13), the model generally improved the performance of some classes compared to LR performance, especially for early blight (1) and mosaic virus (8) classes.

The recovered images using ESPCN and SRCNN (Fig 4.14) contain more details than Bicubic ones, which help the DenseNet-121 to recognize and differentiate among several classes like Septoria leaf spot (4) and mosaic virus (8) classes.

EDSR, WDSR (Fig 4.15), RCAN, and WARCAN (Fig 4.16) confusion matrices have a slight difference. Contrary to the performance of ESPCN and SRCNN, these models improved the

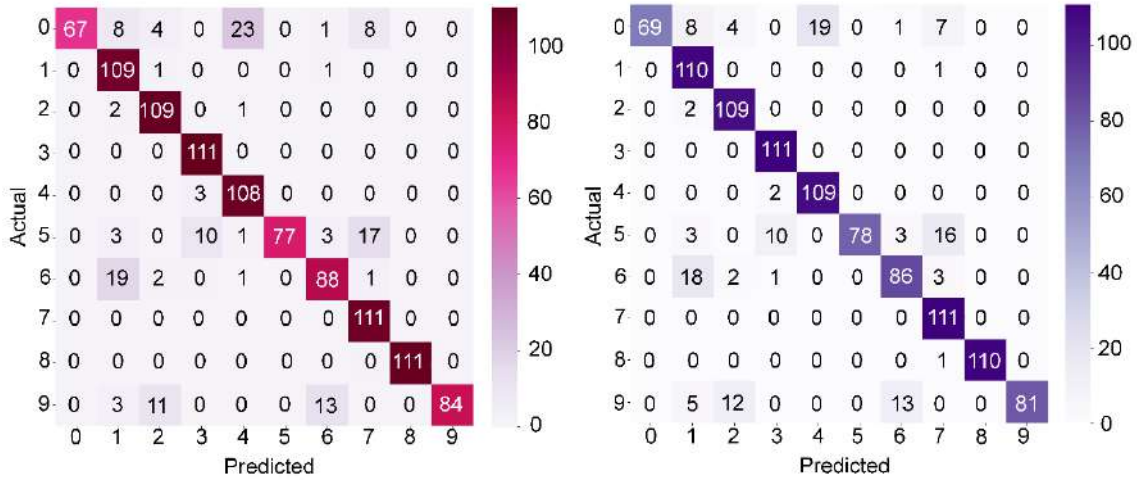


Fig 4.15 Confusion matrix of the DenseNet-121 diseases classification using EDSR (left) and WDSR.

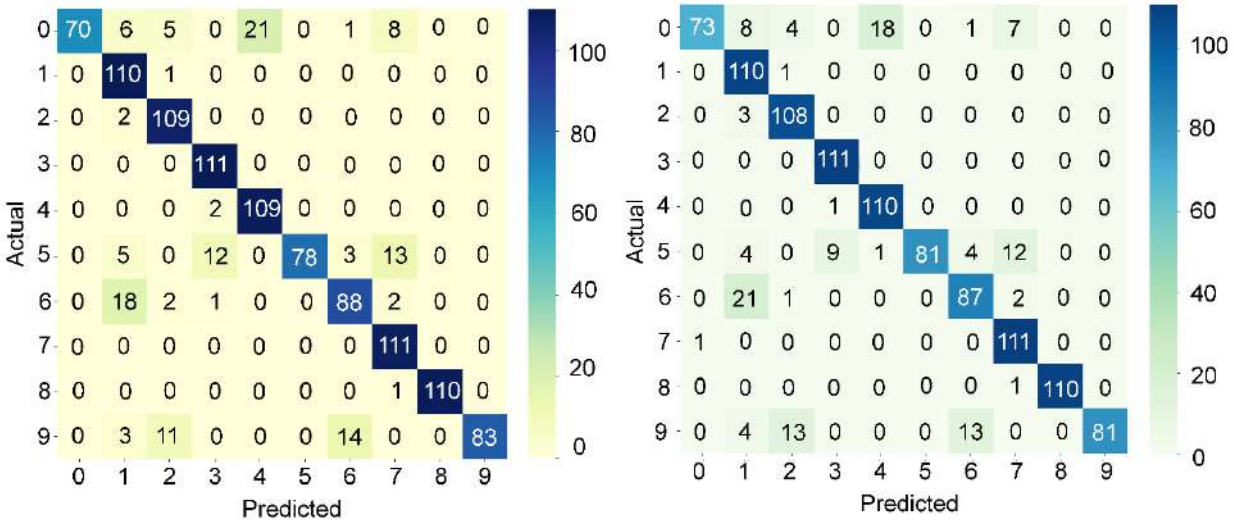


Fig 4.16 Confusion matrix of the DenseNet-121 diseases classification using RCAN (left) and WARCAN.

classification performance for Bacterial spot (0) and a slight difference for Spider mites (5), Target spot (6), and healthy (9) classes.

The WAGAN- (Fig 4.17) achieved a remarkable improvement, where the DenseNet-121 recognized most classes compared to previously mentioned SR algorithms, especially for Bacterial spot (0), Spider mites (5), Target spot (6), healthy (9) classes. Moreover, we can observe a slight decrease in the early blight (1) class.

Compared to LR, the classification performance of ft_ESRGAN (Fig 4.17) and WAGAN (Fig4.18) significantly achieved state-of-the-art results. The DenseNet-121 identifies most classes

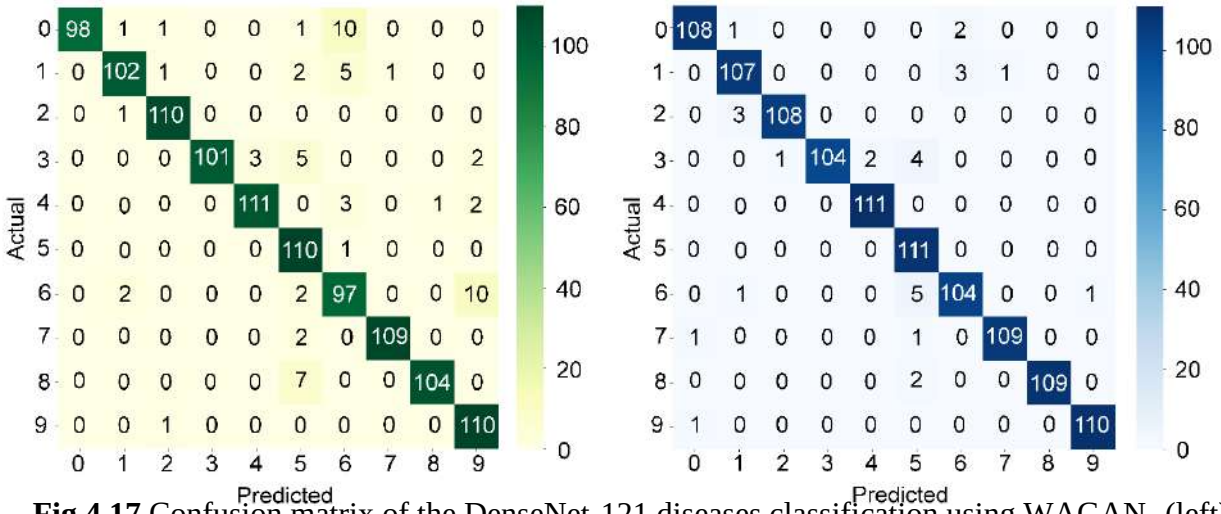


Fig 4.17 Confusion matrix of the DenseNet-121 diseases classification using WAGAN- (left) and ft_ESRGAN.

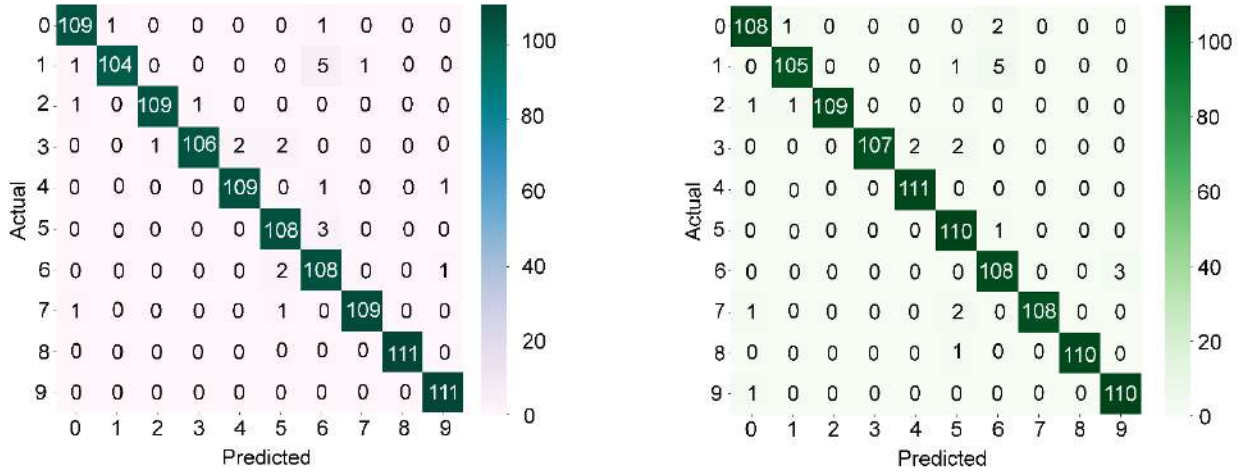


Fig 4.18 Confusion matrix of the DenseNet-121 diseases classification using WAGAN (left) and HR.

with a slight difference to HR (Fig 4.18) results. However, the model could identify all the WAGAN images of mosaic virus (8) and healthy (9), but the HR images were all recognized just in the Septoria leaf spot (4) class. The overall rates of HR are higher than the WAGAN ones, but the WAGAN could even outperform the HR in some cases like Bacterial spot (0), Tomato yellow curl virus (7), mosaic virus (8), and healthy (9).

(5) HR and WAGAN classification analysis: Fig 4.19 shows some examples where images are misclassified using HR and correctly classified with WAGAN. The HR Bacterial spot image looks blurry on the left side of the image. However, the WAGAN deblurred that region. The bare eye cannot detect the differences between the two images, even for the healthy images. Thus, the classifier is the only way to detect deep details. In these examples, the WAGAN could recover more details than the real image already have.

Fig 4.20 illustrates some examples of misclassified WAGAN images and correctly classified HR ones. The HR leaf mold and target spot image look clear, but we can observe that the WAGAN image looks smooth, which may have impacted the classifier decision. Thus, some deep details were slightly missed in the WAGAN images. Generally, the WAGAN achieved a state-of-the-art performance and can find potential applications to improve the identification of the disease. More improvement of the WAGAN may boost the SR performance to recover deep details.

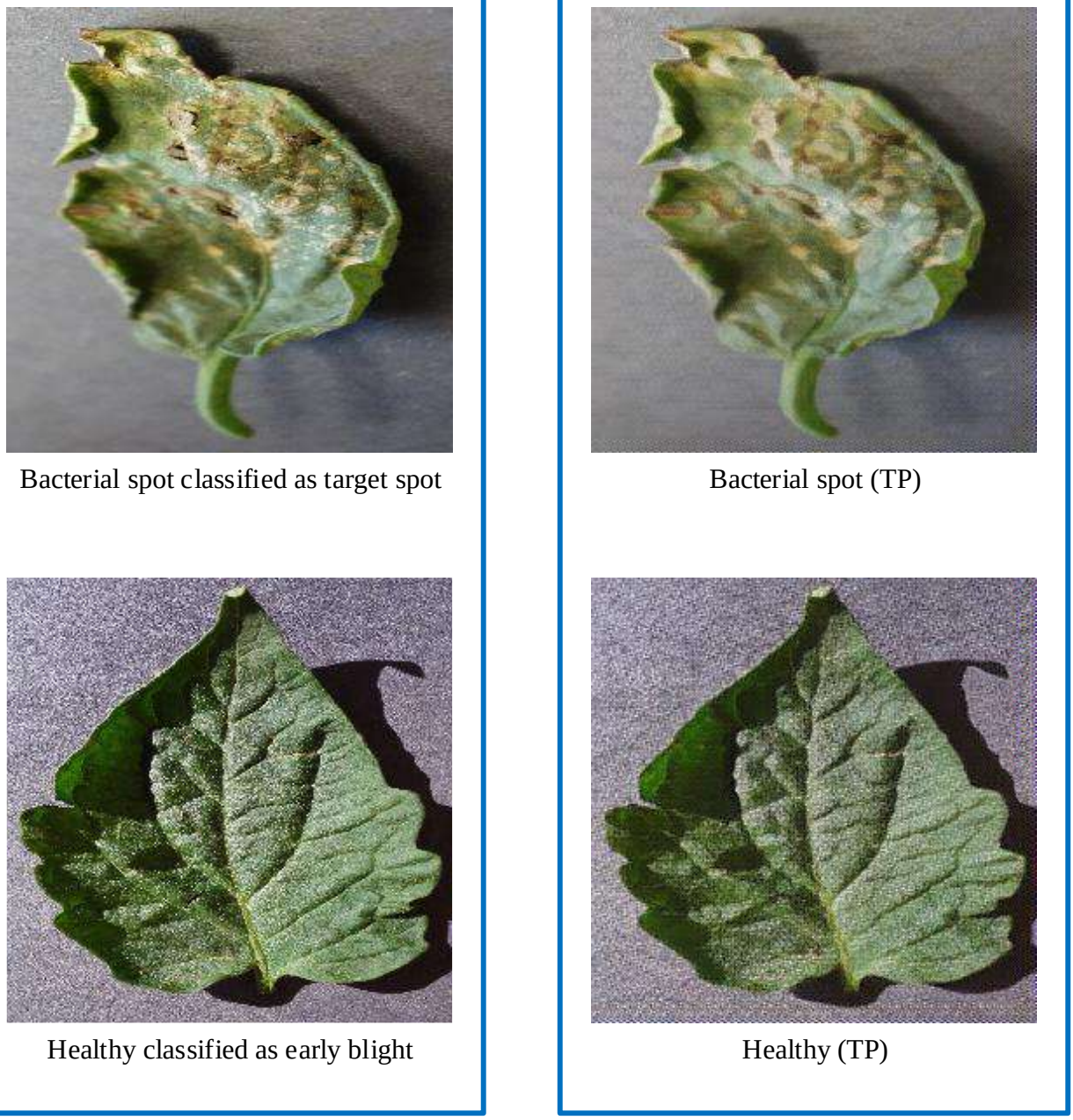


Fig 4.19 Misclassified examples of HR and classified correctly with SR.

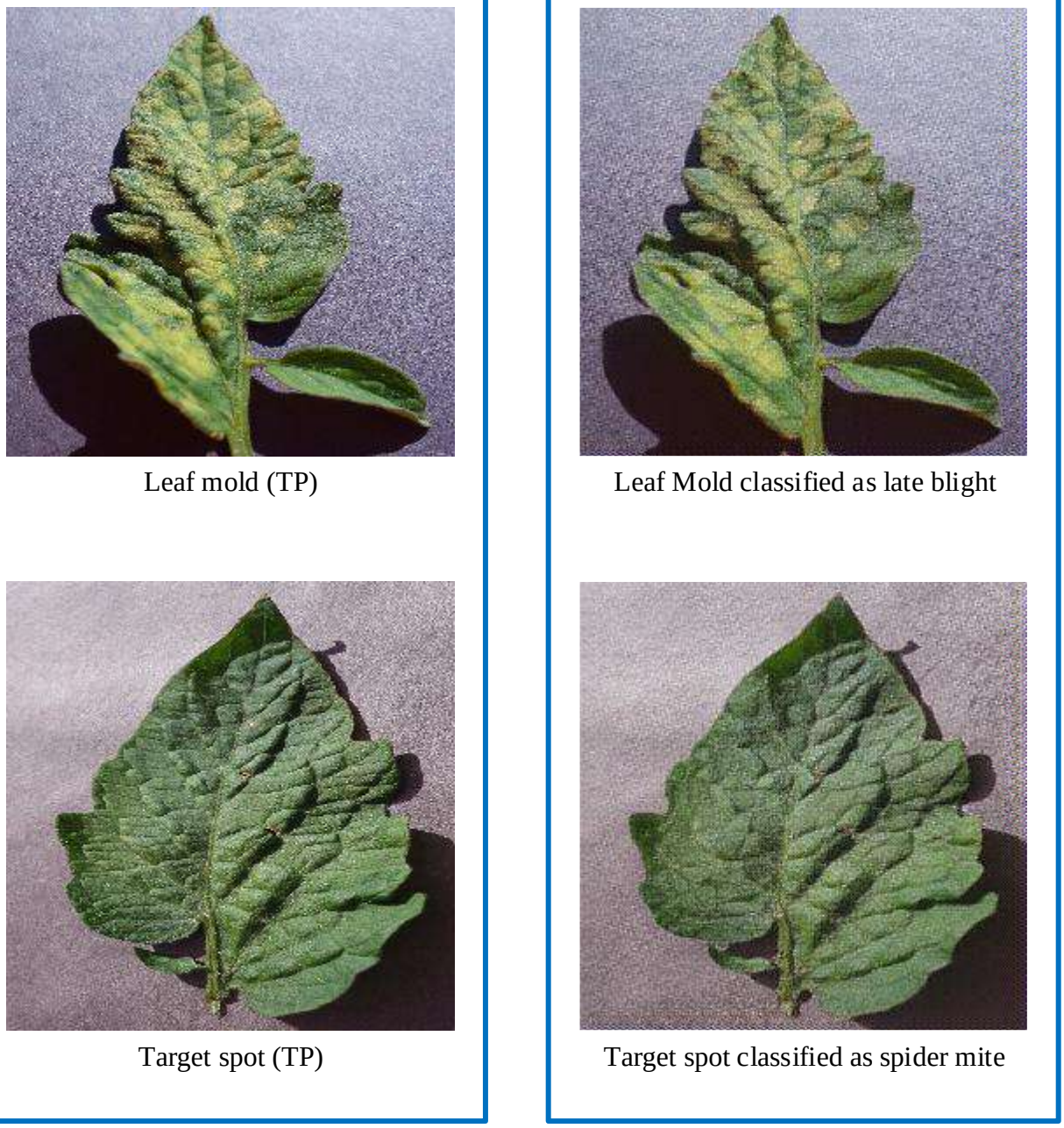


Fig 4.20 Correctly classified samples of HR and Misclassified with SR.

7. Conclusion

In this chapter, we proposed a new enhanced SR algorithm called WAGAN to improve LR images crop disease classification. First, we introduced a novel Residual In Residual (RIR) block to form a low-complexity model than the RCAN [111] named WARCAN. The WARCAN with 50 RIR blocks was used as a generator in the WAGAN design. First, we reconstructed SR images from LR images using the WAGAN and the state-of-the-art methods. In terms of visual quality, the result proved the efficiency of WAGAN compared to the state-of-the-art images approaches with a scale factor of ×4. Then, we evaluated the LR, the resulting SR, and HR images with deep CNNs of 10 types of tomato diseases. The achieved SR images classification accuracy of the WAGAN outperformed the LR and existing SR approaches, and nearby the reference HR accuracy (0.18%), and also with lower complexity (3.6 times) than the ft_ESRGAN [123]. Not only does the proposed method help in the agriculture diagnosis process but also enhances the visual quality of the crop disease images. Therefore, this algorithm may find a potential application to improve the recognition in smart agriculture diagnosis as well as to solve some of the current problems encountered in tomato production.

8. General Conclusion and Future work

In this thesis, we focused on two contributions. We first selected the accurate and low-complex classifier for the HR disease classification images of tomato leaves. Experimental results showed that the classification performance of DenseNet-121 was largely decreased using LR images. Thus, we proposed a new SR network to improve the classification of LR images.

In the first contribution, we proposed to fine-tune many pretrained classifiers on HR images. Extensive results showed that the DenseNet-121 can identify the tomato leaves diseases with high accuracy and low-complex model size. On PlantVillage, the DenseNet obtained a classification accuracy of 97.81% and 96.5% on the PlantDoc dataset. This helped us to select DenseNet-121 as the low-complexity classifier for further analysis of LR images.

In the second contribution, we supposed that the farmer obtained High-Resolution (HR) images of plant leaves from the field. Because of the small size of plant leaves and other limitations, the obtained HR images can miss some detailed information resulting in blurred LR images of leaves with fewer details. Extensive results using LR images demonstrated that DenseNet-121 performed poorly on the tomato leaves diseases. Whereas, the DenseNet-121 obtained an overall accuracy of 63.69% on PlantVillage and 62.40% on PlantDoc. To overcome this issue, we introduced a novel Super-Resolution (SR) algorithm named Wider-activation for Attention-mechanism based on a Generative Adversarial Network (WAGAN) to improve the classification of the tomato diseases LR images. To evaluate the potential of the proposed method in plant diseases recognition, we first recovered SR plant diseases images from LR images using the WAGAN. In terms of visually pleasing, the WAGAN was able to recover more details than the other SR methods, and the recovered SR images look realistic compared to the reference HR image. Next, we tested the classification performance of DenseNet-121 using LR and SR to compare them with HR images. The recovered images using WAGAN outperformed the classification accuracy of LR images with a large margin. On PlantVillage, the DenseNet-121 (using recovered WAGAN images) achieved a classification accuracy of 97.63% and 96.30% on the PlantDoc dataset. The results proved the efficacy of the proposed method in recovering the SR details with ×3.6 lower complexity than the state-of-the-art SR method and very close to the reference HR accuracy.

Focusing on real-world aspects of field scouting is relevant in Algeria to increase the production of crops. The field scouting using UAVs and robots needs an accurate recognition algorithm to

help the farmer in the diagnosis phase. Due to the robustness of WAGAN, we aim to implement our proposed system in field scouting robots to evaluate its performance in a real-world application. Another challenge that was not addressed in this thesis is to implement an SR algorithm with MISR, and with different upscale factors. This idea can be implemented in the UAV monitoring system to prevent the problem of airflow generated by the UAV while flying at a lower altitude. The use of an adaptive upscaling algorithm (MISR) during the UAV capturing, allows the UAV to fly at a higher altitude. Whereas, the SR algorithm upscale every captured image in real-time to improve the detection of diseases. Future studies may focus on collecting more data on the field scouting of many crops. Thus, it may help researchers to build a robust identification system that can recognize many crops disease types to adopt it in different farming machines.

9. List of publications

International publication

1. Salmi, A., Benierbah, S., & Ghazi, M. (2022). Low complexity image enhancement GAN-based algorithm for improving low-resolution image crop disease recognition and diagnosis. Multimedia Tools and Applications, 81(6), 8519-8538. https://doi.org/10.1007/s11042-022-12256-w

Bibliography

Bibliography

[1] Atlas big. World's Leading Tomato Producing Countries. https://www.atlasbig.com/en-cn/countries-by-tomato-production. Accessed 14 Apr 2022.

[2] J. Bai and R. Chen, "Context-aware residual module for image classification,". In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 3388-3395).2020, doi: 10.1109/ICPR48806.2021.9412503.

[3] J. R. Riley, "Remote sensing in entomology," *Annu. Rev. Entomol. Vol. 34*, 1989, doi: 10.1146/annurev.en.34.010189.001335.

[4] A. Merzouki, H. McNairn, J. Powers, and M. Friesen, "Synthetic Aperture Radar (SAR) Compact Polarimetry for Soil Moisture Retrieval," *Remote Sensing* , vol. 11, no. 19. 2019, doi: 10.3390/rs11192227.

[5] U. Shafi, R. Mumtaz, J. García-Nieto, S. A. Hassan, S. A. Zaidi, and N. Iqbal, "Precision Agriculture Techniques and Practices: From Considerations to Applications," *Sensors* , vol. 19, no. 17. 2019, doi: 10.3390/s19173796.

[6] M. Pathan, N. Patel, H. Yagnik, and M. Shah, "Artificial cognition for applications in smart agriculture: A comprehensive review," *Artif. Intell. Agric.*, vol. 4, pp. 81–95, 2020, doi: https://doi.org/10.1016/j.aiia.2020.06.001.

[7] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, "PlantDoc: A dataset for visual plant disease detection," *In Proceedings of the 7th ACM IKDD CoDS and 25th COMAD* (pp. 249-253), 2020, doi: 10.1145/3371158.3371196.

[8] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing," *CoRR*, vol. abs/1511.0, 2015, [Online]. Available: http://arxiv.org/abs/1511.08060.

[9] O. R. Indriani, E. J. Kusuma, C. A. Sari, E. H. Rachmawanto, and D. R. I. M. Setiadi, "Tomatoes classification using K-NN based on GLCM and HSV color space," in *Proceedings - 2017 International Conference on Innovative and Creative Information Technology: Computational Intelligence and IoT, ICITech 2017*, (pp. 1-6), 2018, vol.

2018-January, doi: 10.1109/INNOCIT.2017.8319133.

[10] C. Xie, C. Yang, and Y. He, “Hyperspectral imaging for classification of healthy and gray mold diseased tomato leaves with different infection severities,” *Comput. Electron. Agric.*, vol. 135, p. 154-162, 2017, doi: 10.1016/j.compag.2016.12.015.

[11] J. Basavaiah and A. Arlene Anthony, “Tomato Leaf Disease Classification using Multiple Feature Extraction Techniques,” *Wirel. Pers. Commun.*, vol. 115, no 1, p. 633-651, 2020, doi: 10.1007/s11277-020-07590-x.

[12] C. S. Hlaing and S. M. Maung Zaw, “Tomato Plant Diseases Classification Using Statistical Texture Feature and Color Feature,” *In : 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS). IEEE*, p. 439-444, 2018, doi: 10.1109/ICIS.2018.8466483.

[13] S. Kendler *et al.*, “Detection of crop diseases using enhanced variability imagery data and convolutional neural networks,” *Comput. Electron. Agric.*, vol. 193, p. 106732, 2022, doi: https://doi.org/10.1016/j.compag.2022.106732.

[14] Y. Zhao, C. Sun, X. Xu, and J. Chen, “RIC-Net: A plant disease classification model based on the fusion of Inception and residual structure and embedded attention mechanism,” *Comput. Electron. Agric.*, vol. 193, p. 106644, 2022, doi: https://doi.org/10.1016/j.compag.2021.106644.

[15] S. I. Rimon, M. Islam, A. Dey, A. Das, and others, “PlantBuddy: An Android-Based Mobile Application for Plant Disease Detection Using Deep Convolutional Neural Network,” in *Artificial Intelligence and Technologies*, Springer, 2022, pp. 275–285.

[16] A. Yadav, U. Thakur, R. Saxena, V. Pal, V. Bhateja, and J. C.-W. Lin, “AFD-Net: Apple Foliar Disease Multi Classification using Deep Learning on Plant Pathology Dataset,” *Plant and Soil*: p1-17. 2022.

[17] B. A. M. Ashqar and S. S. Abu-Naser, “Image-Based Tomato Leaves Diseases Detection Using Deep Learning,” *Int. J. Acad. Eng. Res.*, vol. 2, no. 12, 2018.

[18] R. G. De Luna, E. P. Dadios, and A. A. Bandala, “Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition,” in *IEEE*

Region 10 Annual International Conference, Proceedings/TENCON, 2019, p. 1414-1419. 2018-October, doi: 10.1109/TENCON.2018.8650088.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 2.

[20] K. Yamamoto, T. Togami, and N. Yamaguchi, "Super-Resolution of Plant Disease Images for the Acceleration of Image-based Phenotyping and Vigor Diagnosis in Agriculture," *Sensors*, vol. 17, no. 11, p. 2557, Nov. 2017, doi: 10.3390/s17112557.

[21] Z. Wang, J. Chen, and S. C. H. Hoi, "Deep Learning for Image Super-Resolution: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10. 2021, doi: 10.1109/TPAMI.2020.2982166.

[22] C. Gambella, B. Ghaddar, and J. Naoum-Sawaya, "Optimization problems for machine learning: A survey," *European Journal of Operational Research*, vol. 290, no. 3. 2021, doi: 10.1016/j.ejor.2020.08.045.

[23] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *Journal of Artificial Intelligence Research*, vol. 70. 2021, doi: 10.1613/JAIR.1.12228.

[24] A. I. Károly, R. Fullér, and P. Galambos, "Unsupervised clustering for deep learning: A tutorial survey," *Acta Polytech. Hungarica*, vol. 15, no. 8, 2018, doi: 10.12700/APH.15.8.2018.8.2.

[25] D. J. Joshi, I. Kale, S. Gandewar, O. Korate, D. Patwari, and S. Patil, "Reinforcement Learning: A Survey," in *Advances in Intelligent Systems and Computing*, 2021, vol. 1311 AISC, doi: 10.1007/978-981-33-4859-2_29.

[26] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11. 2018, doi: 10.1016/j.heliyon.2018.e00938.

[27] Z. Zhang, P. Cui, and W. Zhu, "Deep Learning on Graphs: A Survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, 2022, doi: 10.1109/TKDE.2020.2981333.

[28] W. H. Cheng, S. Song, C. Y. Chen, S. C. Hidayati, and J. Liu, “Fashion meets computer vision: A survey,” *ACM Computing Surveys*, vol. 54, no. 4. 2021, doi: 10.1145/3447239.

[29] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, “Generative Adversarial Networks: A Survey Toward Private and Secure Applications,” *ACM Computing Surveys*, vol. 54, no. 6. 2021, doi: 10.1145/3459992.

[30] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *J. Big Data*, vol. 3, no. 1, p. 9, 2016, doi: 10.1186/s40537-016-0043-6.

[31] A. Kirchner and C. S. Signorino, “Using Support Vector Machines for Survey Research,” *Surv. Pract.*, vol. 11, no. 1, 2018, doi: 10.29115/sp-2018-0001.

[32] B. Sun and H. Chen, “A Survey of k Nearest Neighbor Algorithms for Solving the Class Imbalanced Problem,” *Wireless Communications and Mobile Computing*, vol. 2021. 2021, doi: 10.1155/2021/5520990.

[33] M. Zakariah, “Classification of large datasets using Random Forest Algorithm in various applications : Survey,” *Int. J. Eng. Innov. Technol.*, vol. 4, no. 3, 2014.

[34] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artif. Intell. Rev.*, vol. 53, no. 8, 2020, doi: 10.1007/s10462-020-09825-6.

[35] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” 2015.

[36] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December, doi: 10.1109/CVPR.2016.90.

[37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-January, doi: 10.1109/CVPR.2017.243.

[38] V. Bolón-Canedo and B. Remeseiro, “Feature selection in image analysis: a survey,” *Artif. Intell. Rev.*, vol. 53, no. 4, 2020, doi: 10.1007/s10462-019-09750-3.

[39] T. M. Lehmann, C. Gönner, and K. Spitzer, “Survey: Interpolation methods in medical image processing,” *IEEE Trans. Med. Imaging*, vol. 18, no. 11, 1999, doi: 10.1109/42.816070.

[40] C. Jiang, Q. Zhang, R. Fan, and Z. Hu, “Super-resolution CT Image Reconstruction Based on Dictionary Learning and Sparse Representation,” *Sci. Rep.*, vol. 8, no. 1, 2018, doi: 10.1038/s41598-018-27261-z.

[41] S. Ayas and M. Ekinci, “Single image super resolution using dictionary learning and sparse coding with multi-scale and multi-directional Gabor feature representation,” *Inf. Sci. (Ny).*, vol. 512, 2020, doi: 10.1016/j.ins.2019.10.040.

[42] M. Yin, J. Gao, D. Tien, and S. Cai, “Blind image deblurring via coupled sparse representation,” *J. Vis. Commun. Image Represent.*, vol. 25, no. 5, 2014, doi: 10.1016/j.jvcir.2014.02.003.

[43] S. Mandal and A. N. Rajagopalan, “Single noisy image super resolution by minimizing nuclear norm in virtual sparse domain,” in *Communications in Computer and Information Science*, 2018, vol. 841, doi: 10.1007/978-981-13-0020-2_15.

[44] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a Deep Convolutional Network for Image Super-Resolution BT - Computer Vision – ECCV 2014,” 2014, pp. 184–199.

[45] C. Ledig *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-January, doi: 10.1109/CVPR.2017.19.

[46] X. Wang *et al.*, “ESRGAN: Enhanced super-resolution generative adversarial networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11133 LNCS, doi: 10.1007/978-3-030-11021-5_5.

[47] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004, doi: 10.1109/TIP.2003.819861.

[48] J. Chen, D. Zhang, and Y. A. Nanehkaran, “Identifying plant diseases using deep transfer

learning and enhanced lightweight network," *Multimed. Tools Appl.*, vol. 79, no. 41, pp. 31497–31515, 2020, doi: 10.1007/s11042-020-09669-w.

[49] D. Bisen, "Deep convolutional neural network based plant species recognition through features of leaf," *Multimed. Tools Appl.*, vol. 80, no. 4, pp. 6443–6456, 2021, doi: 10.1007/s11042-020-10038-w.

[50] M. M. Ozguven and K. Adem, "Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms," *Phys. A Stat. Mech. its Appl.*, vol. 535, p. 122537, 2019, doi: https://doi.org/10.1016/j.physa.2019.122537.

[51] M. A. Khan, T. Akram, M. Sharif, K. Javed, M. Raza, and T. Saba, "An automated system for cucumber leaf diseased spot detection and classification using improved saliency method and deep features selection," *Multimed. Tools Appl.*, vol. 79, no. 25, pp. 18627–18656, 2020, doi: 10.1007/s11042-020-08726-8.

[52] K. Trang, L. TonThat, N. G. M. Thao, and N. T. T. Thi, "Mango Diseases Identification by a Deep Residual Network with Contrast Enhancement and Transfer Learning," in *2019 IEEE Conference on Sustainable Utilization and Development in Engineering and Technologies (CSUDET)*, 2019, pp. 138–142, doi: 10.1109/CSUDET47057.2019.9214620.

[53] B. Espejo-Garcia, N. Mylonas, L. Athanasakos, E. Vali, and S. Fountas, "Combining generative adversarial networks and agricultural transfer learning for weeds identification," *Biosyst. Eng.*, vol. 204, pp. 79–89, 2021, doi: https://doi.org/10.1016/j.biosystemseng.2021.01.014.

[54] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network," *Procedia Comput. Sci.*, vol. 167, pp. 293–301, 2020, doi: https://doi.org/10.1016/j.procs.2020.03.225.

[55] M. Sardogan, A. Tuncer, and Y. Ozen, "Plant Leaf Disease Detection and Classification Based on CNN with LVQ Algorithm," *In : 2018 3rd international conference on computer science and engineering (UBMK). IEEE, 2018. p. 382-385,* 2018, doi: 10.1109/UBMK.2018.8566635.

[56] X. Chen, G. Zhou, A. Chen, J. Yi, W. Zhang, and Y. Hu, "Identification of tomato leaf

diseases based on combination of ABCK-BWTR and B-ARNet," *Comput. Electron. Agric.*, vol. 178, p. 105730, 2020, doi: https://doi.org/10.1016/j.compag.2020.105730.

[57] P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi, and S. G. Koolagudi, "Tomato Leaf Disease Detection Using Convolutional Neural Networks," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*, 2018, pp. 1–5, doi: 10.1109/IC3.2018.8530532.

[58] J. Lu, R. Ehsani, Y. Shi, A. I. de Castro, and S. Wang, "Detection of multi-tomato leaf diseases (late blight, target and bacterial spots) in different stages by using a spectral-based sensor," *Sci. Rep.*, vol. 8, no. 1, p. 2793, 2018, doi: 10.1038/s41598-018-21191-6.

[59] U. Mokhtar, M. A. S. Ali, A. E. Hassenian, and H. Hefny, "Tomato leaves diseases detection approach based on Support Vector Machines," *In : 2015 11th International computer engineering conference (ICENCO). IEEE*, p. 246-250 2016, doi: 10.1109/ICENCO.2015.7416356.

[60] A. E. Hassanien, T. Gaber, U. Mokhtar, and H. Hefny, "An improved moth flame optimization algorithm based on rough sets for tomato diseases detection," *Comput. Electron. Agric.*, vol. 136, 2017, doi: 10.1016/j.compag.2017.02.026.

[61] C. S. Hlaing and S. M. M. Zaw, "Model-based statistical features for mobile phone image of tomato plant disease classification," in *Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings*, 2018, vol. 2017-December, doi: 10.1109/PDCAT.2017.00044.

[62] F. Jakjoud, A. Hatim, and A. Bouaddi, "Detection of diseases on tomato leaves based on sub-classifiers fuzzy combination," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 5, 2019.

[63] K. Tian, J. Li, J. Zeng, A. Evans, and L. Zhang, "Segmentation of tomato leaf images based on adaptive clustering number of K-means algorithm," *Comput. Electron. Agric.*, vol. 165, 2019, doi: 10.1016/j.compag.2019.104962.

[64] Y. W. Tian, P. H. Zheng, and R. Y. Shi, "The Detection System for Greenhouse Tomato Disease Degree Based on Android Platform," *In : 2016 3rd International Conference on*

Information Science and Control Engineering (ICISCE). IEEE, p. 706-710. 2016, doi: 10.1109/ICISCE.2016.156.

[65] J.-H. Park and S.-K. Lee, "An Image Processing Mechanism for Disease Detection in Tomato Leaf," *J. Korea Inst. Electron. Commun. Sci.*, vol. 14, no. 5, pp. 959–968, 2019.

[66] D. Wang *et al.*, "Early Detection of Tomato Spotted Wilt Virus by Hyperspectral Imaging and Outlier Removal Auxiliary Classifier Generative Adversarial Nets (OR-AC-GAN)," *Sci. Rep.*, vol. 9, no. 1, 2019, doi: 10.1038/s41598-019-40066-y.

[67] H. R. Xu, Y. B. Ying, X. P. Fu, and S. P. Zhu, "Near-infrared Spectroscopy in detecting Leaf Miner Damage on Tomato Leaf," *Biosyst. Eng.*, vol. 96, no. 4, 2007, doi: 10.1016/j.biosystemseng.2007.01.008.

[68] J. Lu, M. Zhou, Y. Gao, and H. Jiang, "Using hyperspectral imaging to discriminate yellow leaf curl disease in tomato leaves," *Precis. Agric.*, vol. 19, no. 3, 2018, doi: 10.1007/s11119-017-9524-7.

[69] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, 1998, doi: 10.1109/5.726791.

[70] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition. *BT - 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings." 2015, [Online]. Available: http://arxiv.org/abs/1409.1556.

[71] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07-12-June-2015, doi: 10.1109/CVPR.2015.7298594.

[72] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning, ICML 2015*, 2015, vol. 1.

[73] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December, doi:

10.1109/CVPR.2016.308.

[74] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-January, doi: 10.1109/CVPR.2017.195.

[75] C. Cevallos, H. Ponce, E. Moya-Albor, and J. Brieva, “Vision-Based Analysis on Leaves of Tomato Crops for Classifying Nutrient Deficiency using Convolutional Neural Networks,” *In : 2020 International Joint Conference on Neural Networks (IJCNN).* IEEE, p. 1-7. 2020, doi: 10.1109/IJCNN48605.2020.9207615.

[76] M. Brahimi, K. Boukhalfa, and A. Moussaoui, “Deep Learning for Tomato Diseases: Classification and Symptoms Visualization,” *Appl. Artif. Intell.*, vol. 31, no. 4, 2017, doi: 10.1080/08839514.2017.1315516.

[77] J. Sun, X. He, M. Wu, X. Wu, J. Shen, and B. Lu, “Detection of tomato organs based on convolutional neural network under the overlap and occlusion backgrounds,” *Mach. Vis. Appl.*, vol. 31, no. 5, 2020, doi: 10.1007/s00138-020-01081-6.

[78] T. A. Salih, A. J. Ali, and M. N. Ahmed, “Deep Learning Convolution Neural Network to Detect and Classify Tomato Plant Leaf Diseases,” *OALib*, vol. 07, no. 05, 2020, doi: 10.4236/oalib.1106296.

[79] A. Elhassouny and F. Smarandache, “Smart mobile application to recognize tomato leaf diseases using Convolutional Neural Networks,” *In : 2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*, p. 1-4. 2019, doi: 10.1109/ICCSRE.2019.8807737.

[80] Q. Wu, Y. Chen, and J. Meng, “Dcgan-based data augmentation for tomato leaf disease identification,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2997001.

[81] O. Emebo, B. Fori, G. Victor, and T. Zannu, “Development of Tomato Septoria Leaf Spot and Tomato Mosaic Diseases Detection Device Using Raspberry Pi and Deep Convolutional Neural Networks,” in *Journal of Physics: Conference Series*, 2019, vol. 1299, no. 1, doi: 10.1088/1742-6596/1299/1/012118.

[82] K. R., H. M., S. Anand, P. Mathikshara, A. Johnson, and M. R., “Attention embedded

residual CNN for disease detection in tomato leaves," *Appl. Soft Comput.*, vol. 86, p. 105933, 2020, doi: https://doi.org/10.1016/j.asoc.2019.105933.

[83] A. S. Paymode and V. B. Malode, "Transfer Learning for Multi-Crop Leaf Disease Image Classification using Convolutional Neural Network VGG," *Artif. Intell. Agric.*, vol. 6, pp. 23–33, 2022, doi: https://doi.org/10.1016/j.aiia.2021.12.002.

[84] M. S. A. M. Al-gaashani, F. Shang, M. S. A. Muthanna, M. Khayyat, and A. A. Abd El-Latif, "Tomato leaf disease classification by exploiting transfer learning and feature concatenation," *IET Image Process.*, vol. 16, no. 3, pp. 913–925, 2022, doi: https://doi.org/10.1049/ipr2.12397.

[85] A. Abbas, S. Jain, M. Gour, and S. Vankudothu, "Tomato plant disease detection using transfer learning with C-GAN synthetic images," *Comput. Electron. Agric.*, vol. 187, 2021, doi: 10.1016/j.compag.2021.106279.

[86] L. Aversano, M. L. Bernardi, M. Cimitile, M. Iammarino, and S. Rondinella, "Tomato diseases Classification Based on VGG and Transfer Learning," *In : 2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, p. 129-133, doi: 10.1109/MetroAgriFor50201.2020.9277626.

[87] J. Liu and X. Wang, "Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model," *Plant Methods*, vol. 16, no. 1, 2020, doi: 10.1186/s13007-020-00624-2.

[88] I. Ahmad, M. Hamid, S. Yousaf, S. T. Shah, and M. O. Ahmad, "Optimizing Pretrained Convolutional Neural Networks for Tomato Leaf Disease Detection," *Complexity*, vol. 2020, p. 8812019, 2020, doi: 10.1155/2020/8812019.

[89] V. Venkatesh, N. Yallappa, S. U. Hegde, and S. R. Stalin, "Fine-Tuned MobileNet Classifier for Classification of Strawberry and Cherry Fruit Types," *J. Comput. Sci.*, vol. 17, no. 1, 2021, doi: 10.3844/jcssp.2021.44.54.

[90] S. Vallabhajosyula, V. Sistla, and V. K. K. Kolli, "Transfer learning-based deep ensemble neural network for plant leaf disease detection," *J. Plant Dis. Prot.*, 2021, doi: 10.1007/s41348-021-00465-8.

[91] P. Sahu, A. Chug, A. P. Singh, D. Singh, and R. P. Singh, "Classification and Activation Map Visualization of Banana Diseases Using Deep Learning Models," *In : International Conference on Innovative Computing and Communications*. Springer, Singapore. p. 751-767. 2022.

[92] E. Suryawati, R. Sustika, R. S. Yuwana, A. Subekti, and H. F. Pardede, "Deep structured convolutional neural network for tomato diseases detection," *In : 2018 international conference on advanced computer science and information systems (ICACSIS)*. p. 385-390. 2019, doi: 10.1109/ICACSIS.2018.8618169.

[93] H. Durmus, E. O. Gunes, and M. Kirci, "Disease detection on the leaves of the tomato plants by using deep learning," *In : 2017 6th International conference on agro-geoinformatics.* p. 1-5. 2017, doi: 10.1109/Agro-Geoinformatics.2017.8047016.

[94] L. Mkonyi *et al.*, "Early identification of Tuta absoluta in tomato plants using deep learning," *Sci. African*, vol. 10, 2020, doi: 10.1016/j.sciaf.2020.e00590.

[95] J. Shijie, J. Peiyi, H. Siping, and Sl. Haibo, "Automatic detection of tomato diseases and pests based on leaf images," in *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, 2017, vol. 2017-January, doi: 10.1109/CAC.2017.8243388.

[96] Di. Jiang, F. Li, Y. Yang, and S. Yu, "A Tomato Leaf Diseases Classification Method Based on Deep Learning," *In : 2020 chinese control and decision conference (CCDC). IEEE*. p. 1446-1450. 2020, doi: 10.1109/CCDC49329.2020.9164457.

[97] S. Z. M. Zaki, M. A. Zulkifley, M. M. Stofa, N. A. M. Kamari, and N. A. Mohamed, "Classification of tomato leaf diseases using MobileNet v2," *IAES Int. J. Artif. Intell.*, vol. 9, no. 2, p. 290, 2020.

[98] A. Hidayatuloh, M. Nursalman, and E. Nugraha, "Identification of Tomato Plant Diseases by Leaf Image Using Squeezenet Model," *In : 2018 International Conference on Information Technology Systems and Innovation (ICITSI)*. p. 199-204. 2018, doi: 10.1109/ICITSI.2018.8696087.

[99] E. K. Nithish, M. Kaushik, P. Prakash, R. Ajay, and S. Veni, "Tomato leaf disease detection using convolutional neural network with data augmentation," *In : 2020 5th*

International Conference on Communication and Electronics Systems (ICCES). p. 1125-1132. 2020, doi: 10.1109/ICCES48766.2020.09138030.

[100] M. Ouhami, Y. Es-Saady, M. El Hajji, A. Hafiane, R. Canals, and M. El Yassa, "Deep transfer learning models for tomato disease detection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, vol. 12119 LNCS, doi: 10.1007/978-3-030-51935-3_7.

[101] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput. Electron. Agric.*, vol. 161, pp. 272–279, 2019, doi: https://doi.org/10.1016/j.compag.2018.03.032.

[102] A. Chakraborty, D. Kumer, and K. Deeba, "Plant Leaf Disease Recognition Using Fastai Image Classification," *In : 2021 5th International Conference on Computing Methodologies and Communication (ICCMC).* IEEE. p. 1624-1630 2021, doi: 10.1109/ICCMC51019.2021.9418042.

[103] T. O. Emmanuel, "PlantVillage Dataset," *Kaggle.com*, 2018. https://www.kaggle.com/emmarex/plantdisease (accessed Dec. 12, 2020).

[104] J. Torres-Sánchez, F. López-Granados, A. I. De Castro, and J. M. Peña-Barragán, "Configuration and Specifications of an Unmanned Aerial Vehicle (UAV) for Early Site Specific Weed Management," *PLoS One*, vol. 8, no. 3, 2013, doi: 10.1371/journal.pone.0058210.

[105] Y. Zhang, Y. Li, W. Wen, Y. Wu, and J. Chen, "Deciphering an image cipher based on 3-cell chaotic map and biological operations," *Nonlinear Dyn.*, vol. 82, no. 4, pp. 1831–1837, 2015, doi: 10.1007/s11071-015-2280-1.

[106] Y. Li, B. Zou, S. Deng, and G. Zhou, "Using Feature Fusion Strategies in Continuous Authentication on Smartphones," *IEEE Internet Comput.*, vol. 24, no. 2, 2020, doi: 10.1109/MIC.2020.2971447.

[107] W. Shi *et al.*, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December, doi:

10.1109/CVPR.2016.207.

[108] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced Deep Residual Networks for Single Image Super-Resolution,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017, vol. 2017-July, doi: 10.1109/CVPRW.2017.151.

[109] J. Yu *et al.*, “Wide Activation for Efficient and Accurate Image Super-Resolution.,” *CoRR*, vol. abs/1808.0. 2018, [Online]. Available: http://arxiv.org/abs/1808.08718.

[110] T. Salimans and D. P. Kingma, “Weight normalization: A simple reparameterization to accelerate training of deep neural networks,” *Advances in neural information processing systems*, vol. 29. 2016.

[111] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11211 LNCS, doi: 10.1007/978-3-030-01234-2_18.

[112] P. Wu, Z. Cui, Z. Gan, and F. Liu, “Residual group channel and space attention network for hyperspectral image classification,” *Remote Sens.*, vol. 12, no. 12, 2020, doi: 10.3390/rs12122035.

[113] H. Tang, D. Xu, N. Sebe, Y. Wang, J. J. Corso, and Y. Yan, “Multi-channel attention selection gan with cascaded semantic guidance for cross-view image translation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, vol. 2019-June, doi: 10.1109/CVPR.2019.00252.

[114] Q. Dai, X. Cheng, Y. Qiao, and Y. Zhang, “Crop Leaf Disease Image Super-Resolution and Identification With Dual Attention and Topology Fusion Generative Adversarial Network,” *IEEE Access*, vol. 8, pp. 55724–55735, 2020, doi: 10.1109/ACCESS.2020.2982055.

[115] J. G. Arnal Barbedo, “Plant disease identification from individual lesions and spots using deep learning,” *Biosyst. Eng.*, vol. 180, pp. 96–107, 2019, doi: https://doi.org/10.1016/j.biosystemseng.2019.02.002.

[116] L. Zhang, J. Jia, Y. Li, W. Gao, and M. Wang, “Deep learning based rapid diagnosis system for identifying tomato nutrition disorders,” *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 4, 2019, doi: 10.3837/tiis.2019.04.015.

[117] Q. H. Cap, H. Tani, H. Uga, S. Kagiwada, and H. Iyatomi, “Super-Resolution for Practical Automated Plant Disease Diagnosis System,” in *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, 2019, pp. 1–6, doi: 10.1109/CISS.2019.8692855.

[118] R. Sun, M. Zhang, K. Yang, and J. Liu, “Data enhancement for plant disease classification using generated lesions,” *Appl. Sci.*, vol. 10, no. 2, 2020, doi: 10.3390/app10020466.

[119] M. A. B. Mahmoud, P. Guo, and K. Wang, “Pseudoinverse learning autoencoder with DCGAN for plant diseases classification,” *Multimed. Tools Appl.*, vol. 79, no. 35, pp. 26245–26263, 2020, doi: 10.1007/s11042-020-09239-0.

[120] M. Arsenovic, M. Karanovic, S. Sladojevic, A. Anderla, and D. Stefanovic, “Solving current limitations of deep learning based approaches for plant disease detection,” *Symmetry (Basel).*, vol. 11, no. 7, 2019, doi: 10.3390/sym11070939.

[121] H. Nazki, S. Yoon, A. Fuentes, and D. S. Park, “Unsupervised image translation using adversarial networks for improved plant disease recognition,” *Comput. Electron. Agric.*, vol. 168, p. 105117, 2020, doi: https://doi.org/10.1016/j.compag.2019.105117.

[122] G. Yilma, S. Belay, Z. Qin, K. Gedamu, and M. Ayalew, “Plant Disease Classification Using Two Pathway Encoder GAN Data Generation,” in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2020, pp. 67–72, doi: 10.1109/ICCWAMTIP51612.2020.9317494.

[123] J. Wen, Y. Shi, X. Zhou, and Y. Xue, “Crop Disease Classification on Inadequate Low-Resolution Target Images,” *Sensors (Basel).*, vol. 20, no. 16, 2020, doi: 10.3390/s20164601.

[124] H. Qin, M. A. El-Yacoubi, Y. Li, and C. Liu, “Multi-Scale and Multi-Direction GAN for CNN-Based Single Palm-Vein Identification,” *IEEE Trans. Inf. Forensics Secur.*, vol. 16, 2021, doi: 10.1109/TIFS.2021.3059340.

[125] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss Functions for Image Restoration With Neural Networks,” *IEEE Trans. Comput. Imaging*, vol. 3, no. 1, pp. 47–57, 2017, doi: 10.1109/TCI.2016.2644865.

[126] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:*1412.6980, 2015.

[127] S. Nah, T. H. Kim, and K. M. Lee, “Deep multi-scale convolutional neural network for dynamic scene deblurring,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-January, doi: 10.1109/CVPR.2017.35.

[128] M. Q. Khairuzzaman, “Tf.data,” 2016. https://www.tensorflow.org/guide/data (accessed Dec. 12, 2020).

[129] E. Agustsson and R. Timofte, “NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1122–1131, doi: 10.1109/CVPRW.2017.150.

[130] M. Lin, Q. Chen, and S. Yan, “Network In Network,” *CoRR*, vol. abs/1312.4, 2014.