Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Kasdi Merbah, Ouargla
Faculty of Modern Information and Communication Technologies

MASTER'S THESIS
Presented for the award of the grade of engineer
in: Computer Science

Specialty: Industrial

Theme

Explainable Disease Classification in Tomato Leaves Using Deep Learning

Directed By:
CHAHBI Lyna
AOUACHIR Razane Bia

Supervised By:
Dr. HAMROUNI Basma

Co-Supervised By:
Dr. BOUANANE Khadra

President:
Khadidja AMEUR

Examiner:
Lamis HAMROUNI

Academic Year: 2024-2025

Achnowledgements

In The name of Allah , the Almighty, we would like to express our deep gratitude for Him for the strength , faith and determination He has granted us to overcome the challenges and obstacles encountered throughout our academic journey.

We also wish to sincerely thank our parents and families for their unwavering support. Their presence by our side has been a source of strength and motivation enabling us to move forward on this path with confidence and peace.

We warmly thank our supervisor , Dr.HAMROUNI Basma and Dr BOUANANE Khadra, for her invaluable guidance and tireless efforts, as well as all our teachers for their dedication and contribution to our success.

Finally, Thank you to all those who stood by us, whether from near or far.

May each one find here the expression of our infinite gratitude.

Dedication

This work is dedicated to :

Our Dear Parents, who have been a constant support through their love , their encouragement, and prayers which have been a source of inspiration for us. As well as our teachers , to whom we pay tribute for their dedication , expertise , and patience , which have greatly contributed to our academic and personal growth over these past years.

Abstract

This thesis addresses the challenge of detecting tomato leaf diseases using deep learning combined with Explainable Artificial Intelligence (XAI) techniques, and the development of a mobile application for practical field use. A convolutional neural network (CNN) model is trained on annotated image datasets to accurately classify multiple foliar diseases. To enhance the interpretability of the model's predictions, we implement a modified version of the Score-CAM technique adapted for deployment on lightweight mobile environments, such as TensorFlow Lite. Due to technical constraints of on-device inference, our approach approximates the importance of activation maps by using their maxi- mum activation values as proxies, enabling efficient heatmap generation without the need for intermediate layer access or repeated inference. This pragmatic adaptation balances performance and explainability within the limitations of mobile platforms. The resulting system integrates the trained model into a user-friendly mobile application built with Flutter, offering offline disease detection and visual explanations. This work contributes a novel methodology for practical and interpretable plant disease diagnosis on resource-constrained devices, facilitating early detection and management in agriculture.

Keywords: *Offline Mobile application , Deep learning, CNN, Tomato disease detection, weather conditions, TFLite , lightness , Explainable AI, Compatible Score-CAM.*

Résumé

Ce travail traite du défi de la détection des maladies des feuilles de tomate en utilisant le deep learning combiné à des techniques d'intelligence artificielle explicable (XAI), ainsi que du développement d'une application mobile pour une utilisation pratique sur le terrain. Un modèle de réseau de neurones convolutifs (CNN) est entraîné sur un jeu de données annotées afin de classer avec précision plusieurs maladies foliaires. Pour améliorer l'interprétabilité des prédictions du modèle, nous implémentons une version modifiée de la méthode Score-CAM adaptée au déploiement dans des environnements mobiles légers, tels que TensorFlow Lite. En raison des contraintes techniques de l'inférence sur appareil, notre approche approxime l'importance des cartes d'activation en util- isant leur valeur maximale comme proxy, permettant ainsi une génération effi- cace de cartes de chaleur sans nécessiter l'accès aux couches intermédiaires ni des inférences répétées. Cette adaptation pragmatique équilibre performance et explicabilité dans les limites des plateformes mobiles. Le système résultant intègre le modèle entraîné dans une application mobile conviviale développée avec Flutter, offrant une détection des maladies hors ligne accompagnée d'explications visuelles. Ce travail apporte une méthodologie novatrice pour un diagnostic pratique et interprétable des maladies des plantes sur des dispositifs à ressources limitées, facilitant ainsi la détection précoce et la gestion en agriculture.

Mots-clés: *Application mobile hors ligne , apprentissage profond, CNN, détection des maladies du tomate, conditions météorologiques, TFLite, légèreté, intelligence artificielle explicable, Score-CAM compatible.*

Contents

List of Figures

List of Tables

General Introduction

Identifying plant diseases is essential for early diagnosis and effective agricultural management. Tomatoes (Solanum lycopersicum) are among the most widely cultivated crops globally [1], serving as a valuable income source for farmers due to their rich nutritional content and pharmacological benefits. However, tomato plants are vulnerable to various diseases, particularly those affecting their leaves, which can significantly impact crop quality and yield. Early detection and the use of natural remedies are crucial for effective disease control, requiring a comprehensive understanding of the underlying causes and consequences [2].[3].

Tomato leaves are vulnerable to various diseases and infestations, each presenting distinct symptoms and challenges. Common disease classes include Bacterial Spot, which causes small, water-soaked lesions that may lead to leaf defoliation; Early Blight and Late Blight, both fungal infections marked by dark, concentric spots or rapidly spreading necrosis; and Leaf Mold, which appears as yellow spots on the upper surface and velvety mold underneath. Mosaic Virus and Yellow Leaf Curl Virus are viral diseases that lead to mot- tled, curled, or yellowing leaves, severely affecting plant growth. Septoria Leaf Spot is characterized by numerous small, circular spots with dark borders, re- ducing photosynthetic activity. Target Spot, another fungal disease, creates larger lesions with concentric rings resembling a target. Spider Mites, although insects, cause stippling and bronzing of leaves due to their feeding behavior. Among these, the Healthy class serves as a control, representing disease-free leaves. Accurate classification of these conditions is crucial for timely treat- ment and maintaining tomato crop productivity. Among the widely used al- gorithms for classifying images of leaf diseases in various plants, Convolutional Neural Networks (CNNs) are the most prominent. They are preferred for their ability to extract detailed features from images. These features are essential for identifying disease patterns. A wide range of recent studies has explored

tomato leaf disease detection using deep learning models, highlighting advancements in both accuracy and model efficiency. Jelali Mohieddine (2024) provided a foundational review, emphasizing object detection with Faster R-CNN and YOLO on real-field datasets, though cautioning about overly opti mistic results on PlantVillage. Muzammil Khan (2024) introduced a hybrid CNN-RNN approach, achieving 97.44Currently, artificial intelligence (AI) is attracting significant attention across various domains. Many research fields are either adopting AI or transitioning from traditional rule-based systems to AI-enabled solutions. However, a major limitation of modern AI systems especially those leveraging deep learning and machine learning is their lack of transparency. These systems often fail to explain their internal mechanisms or the variables influencing critical decisions. This opacity can undermine user trust, ultimately discouraging adoption of the resulting applications. While some researchers argue that focusing on explainability is unnecessary or overly complex, others emphasize that providing explanations alongside AI-generated outputs can enhance human understanding and trust. Addressing this gap is crucial not only for fostering confidence in AI technologies but also for unlocking their full potential in practical and commercial applications 48. Therefore, Despite their strong performance, deep learning models often lack interpretability. In sensitive domains such as agriculture, it is essential that users understand the decisions made by these systems especially when they carry significant economic implications. Integrating Explainable Artificial Intelligence (XAI) techniques, such as Grad-CAM, SHAP, or Score-CAM, helps highlight the image regions that contributed to a given prediction. This enhances model transparency and fosters greater user trust. Resource-constrained devices generally have limited processing capabilities, memory, and storage, posing challenges for running traditional, large-scale deep learning models efficiently [4]. To overcome these constraints, researchers have introduced lightweight neural network architectures that aim to balance accuracy with computational efficiency. These models are specifically designed to minimize memory usage and processing demands while still achiev- ing competitive performance, making them well-suited for real-time inference on low-power platforms [5]. Notable examples include MobileNetV3 [6], Efficient- NetV2 [7], which have been widely adopted due to their reduced model sizes, lower latency, and increased efficiency compared to conventional architectures. However, these lightweight models often encounter trade-offs, particularly in

terms of accuracy versus efficiency, when applied to complex classification problems in constrained environments.In this context, and to address the identified challenges, we propose an explainable, lightweight, and deployable system for tomato leaf disease detection using deep learning. Our approach not only ensures accurate and offline diagnosis but also enhances model interpretability. Moreover, we introduce a novel, more efficient variant of the Score-CAM technique, specifically adapted for mobile environments, which constitutes a key contribution of this work. 53.

In this context, and to address the identified challenges, we propose an explainable, lightweight, and deployable system for tomato leaf disease detection using deep learning. Our approach not only ensures accurate and offline diagnosis but also enhances model interpretability. Moreover, we introduce a novel, more efficient variant of the Score-CAM technique, specifically adapted for mobile environments, which constitutes a key contribution of this work.

This thesis is organized into six chapters. Chapter 1 provides an overview of common tomato leaf diseases and their agricultural impact. Chapter 2 ex- plores smart farming technologies and presents the theoretical foundations of artificial intelligence, machine learning, and explainable AI techniques. while also reviewing recent related works in the domains of tomato disease detec- tion, explainability techniques, and mobile applications. Chapter 3 explains the methodology used to build the system, covering data preparation, CNN architecture, model optimization, and the proposed lightweight Score-CAM variant for TFLite. Chapter 4 presents the experimental setup, results, and perfor- mance evaluation of both the model and the mobile application. Chapter 5 discusses the deployment phase, including the basic web prototype phase, the current mobile app deployment strategy and future cloud-based proposed up- dates. Finally, Chapter 6 concludes the work by summarizing the contributions and outlining potential directions for future research.

Contribution

This work offers several key contributions :

- Proposal of an intelligent system for tomato leaf disease detection that combines accuracy, explainability, and lightweight performance.
- Deployment of the model in offline mode, suitable for resource-constrained

agricultural environments, by converting it to TensorFlow Lite format.

- Integration of Explainable AI (XAI) to enhance user trust, using visual explanations that highlight the regions influencing the model's decisions.
- Development of a new, optimized variant of Score-CAM, specifically adapted for mobile environments, reducing computational cost while maintaining explanation quality, a key contribution of this work.
- Implementation of an intuitive and functional mobile application, provid- ing real-time results along with adaptive user guidance (e.g., based on local weather conditions).

Chapter 1

Tomato Leaf Diseases

Tomatoes (Solanum lycopersicum) are among the most widely cultivated and consumed vegetables locally in Algeria and globally, serving as a staple in vari- ous cuisines and contributing significantly to agricultural economies. In Algeria, tomato cultivation holds a prominent position, accounting for 10% of the to- tal vegetable production, with approximately **39,400** hectares dedicated to its cultivation, yielding around 1.2 million tons annually. Highlighting their agri- cultural importance and economic impact [8]. However, tomato production faces numerous challenges, notably illnesses that can drastically reduce yield and quality [9].

1.1 Historical Overview of the traditional tomato disease diagnosis

Historically, the diagnosis of tomato diseases relied heavily on visual, repetitive, and manual inspection as well as the expertise of farmers and agricultural ex- tension workers. Symptoms such as leaf spots, wilting, and fruit discoloration were used to identify diseases, often leading to misidentification due to symptom similarities among different pathogens resulting in misdiagnosis and ineffective treatment which directly causes further plant losses [10]. So this approach is time- consuming and requires a high level of experience.

Advancements in plant pathology introduced laboratory-based diagnostic meth- ods, including microscopic examination, culture techniques, and serological assays like enzyme-linked immunosorbent assay (ELISA). These methods im- proved accuracy but were often inaccessible to smallholder farmers due to cost

and infrastructure requirements.

The advent of molecular techniques, such as Polymerase Chain Reaction (PCR) further enhanced diagnostic precision by detecting specific pathogen Deoxyribonucleic Acid (DNA) [11].

However, these methods still necessitated specialized equipment and trained personnel.

Before the adoption of these scientific techniques, various other traditional principles were made and followed in plant disease management. From 1926 to 1928, H.H. whetzel introduced four fundamental principles of pathogen control:

- **Exclusion:** is a method about restricting the movement of harmful plant pathogens by using measures like seed inspection, quarantine or treatment so they stay out of healthy areas and they don't spread inside them.
- **Eradication:** radication means the complete removal of the disease causing-agents including bacteria, viruses, fungi and other pathogens that are present in the field by killing them.
- **Protection:** it involves using chemicals like coverings or sprays which work as a toxicant barrier to prevent spreading infections by shielding plants from being infected in the first place.
- **Immunization:** also called resistance which means planting various categories of crops that are naturally strong against certain illnesses and less prone to infection. Later on these principles were expanded to six principles by other pathologists like George N.Agrios adding:
- **Avoidance:** means controlling the area or the time of growing crops when diseases are less likely to occur like avoiding known infected places or humid seasons.
- **Therapy:** it involves chemotherapy or thermotherapy (chemical or heat-based treatment) Which is a type of treatment that cures plants that are already infected. [12].

These foundational principles became the basis of what we call today traditional plant disease management practices yet their limitations in speed, precision and accessibility have driven a growing demand for developing an innovative diagnostic tool over time. In recent years, attention has shifted toward the development of rapid, affordable and user friendly diagnostic tools including those

powered by deep learning and mobile technologies.

1.1.1 Geographical Distribution of the world's tomato production

Tomato cultivation is a global enterprise, with production concentrated in regions that offer favorable climatic conditions such as adequate sunlight, temperature, and rainfall. According to the Food and Agriculture Organization **(FAO)**, the leading tomato-producing countries include China, India, the United States, Turkey, and Egypt.[8].

China stands as the largest producer contributing approximately **35%** of the world's tomato output.

India follows, with significant production. The United States particularly California, is a major producer, focusing on both fresh market and processing tomatoes. Italy is the most important producer within The European Union **(EU)** contributing **38%** to the total EU production.

Turkey also play substantial roles in global production, both benefiting from favorable Mediterranean climates.

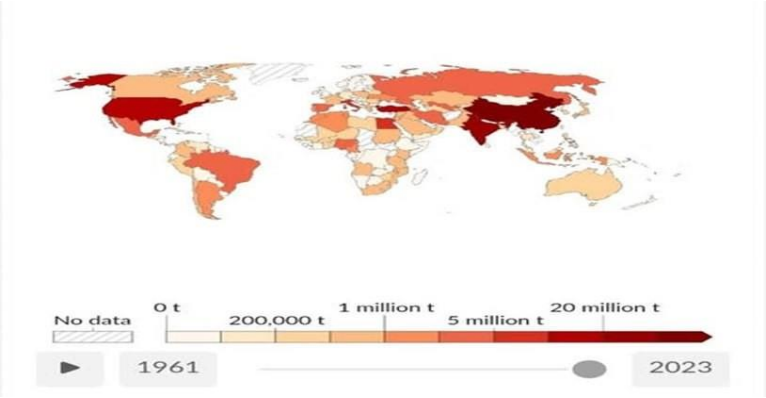


Figure 1.1: Global Tomato Production Map [1].

In Africa, Algeria, Nigeria and Egypt have seen increased tomato cultivation [8], driven by domestic demand and expert opportunities as well as the ideal climate. Even so challenges such as pests, diseases, and post-harvest losses continue to impact productivity and this research has been made to face these obstacles in Algeria specifically.

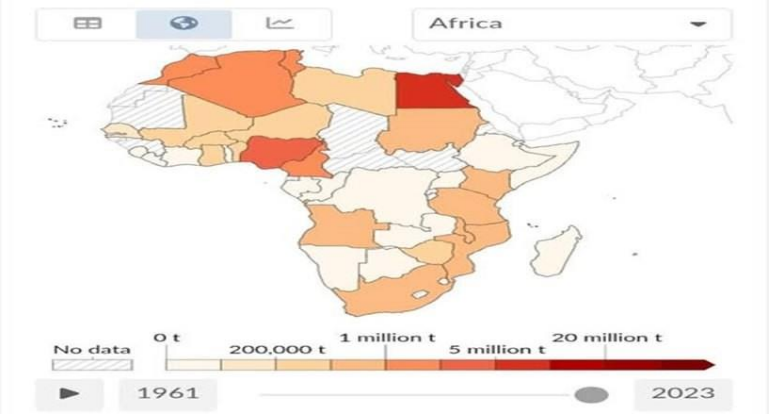


Figure 1.2: African Tomato Production Map [1].

1.1.2 Common Tomato Diseases

Bacterial Spot

is a serious and highly destructive disease that affects tomatoes, causing infections that, in severe cases render the fruit unsuitable for sale and may even lead to plant death. While it can occur in various locations, it is most prevalent in greenhouse, and regions with high temperatures and humidity [13].

Figure 1.3: Tomato leaves infected by bacterial spot disease [2].

Early Blight

Is a common disease in tomatoes that weakens plants by damaging leaves , stems , and fruits leading to reduces yield it appears first as small dark spots on older leaves, later forming large brown lesions with yellow halos.
In severe cases, leaves fall off, stems develop sunken brown spots, and fruit forms black, leathery patches. If the infection spreads, seedlings can wilt and

die [14].

(a)

(b)

Figure 1.4: Tomato ana Tomato leaves infected by early blight [3].

Late Blight

Is a fast-spreading disease in tomato fields that can destroy entire crops if un-treated , it causes large brown patches on leaves, darkens stems, and rots fruit and tubers. While in humid environments, a white fungal layer may form on affected parts.Severely infected plants wilt and die [15].

Figure 1.5: Tomato leaves infected by late blight [4].

Leaf Mold

A disease that thrives in humid environments (above 85%) such as greenhouse conditions primarily affecting tomatoes As it spreads it infects leaves, causing yellow spots on the upper surface and fuzzy olive mold underneath. Over time, leaves wither and die The fungus, Passalora fulva , is the plant responsible for

this disease and can significantly impact yield if not controlled [16].

(a) (b)

Figure 1.6: Tomato leaves infected by leaf mold [5].

Septoria leaf spot

Is a common disease caused by the fungus Septoria lycopersici (a fungal pathogen)that affects tomatoes worldwide, causing severe leaf loss and even total crop failure in extreme cases. It first appears on lower leaves as small water-soaked spots that turn gray-tan with dark margins. As the plant gets in- fected, it leaves yellow, dry out, and fall off, reducing yield. The disease spreads through water, wind, insects, and contaminated tools, thriving in warm, humid environments.[17].

Figure 1.7: Tomato leaves infected by septoria leaf spot [6].

Spider Mites

Tomato plants are commonly affected by the two-spotted spider mite (Tetranychus urticae), especially during periods of hot and dry conditions. These mites damage the plant by feeding on the sap beneath the leaves which leads to a speckled or mottled look on the top of the leaves.
The affected leaves may turn yellow, dry, and eventually fall off reducing the plant's overall health, and in extreme cases, causing the plant's death. Heavy infestations can also result in fine webbing covering the plant.[18].

Figure 1.8: Tomato leaves infected by spider mites [7].

Target spot

Caused by Corynespora cassiicola (pathogen), is a destructive disease that affects tomatoes leading to significant damage to both leaves and fruit.Leaf lesions start as small dark brown spots that expand into light brown or gray areas with

concentric rings and yellowing around the edges. Fruit lesions in the other hand begin as small brown spots, deepening into pitted areas and leaving a character- istic cracking pattern. That disease causes early leaf drop and fruit rot, severely impacting tomato yield and marketability [19].

Figure 1.9: Tomato leaves infected by target spot [8].

Mosaic Virus

Tomato plants are vulnerable to multiple viruses, including Tomato mosaic virus (ToMV), which spreads through contaminated seeds, insects, or human handling. These viruses cause leaf discoloration, yellowing or browning under the skin, deformation, reduced fruit yield, and irregular ripening [20].

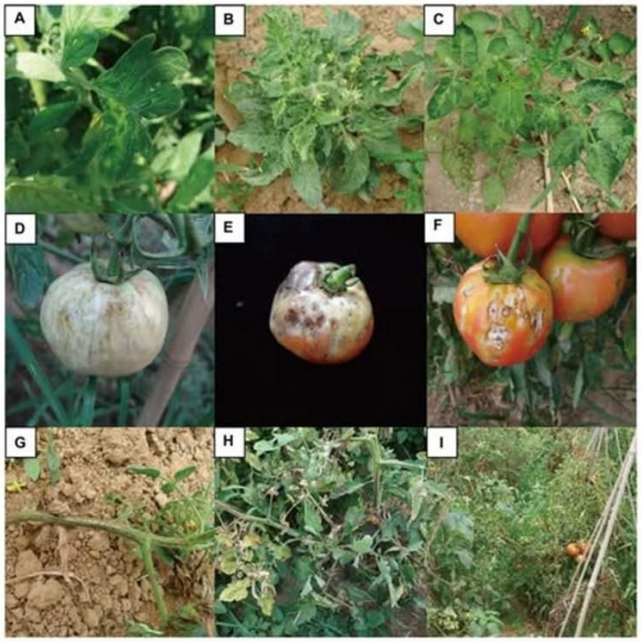


Figure 1.10: Tomato leaves infected by mosaic virus [9].

Yellow leaf curl virus: (TYLCV)

is a viral disease affecting tomatoes in temperate, tropical, and subtropical regions, spread by whiteflies it results in symptoms like yellow and curled leaves , stunted growth, and poor fruit production.[21]

Figure 1.11: Tomato leaves infected by yellow leaf curl virus [10].

1.2 Conclusion

In this chapter, we provided a comprehensive overview of tomato leaf diseases, starting with the traditional methods used for diagnosing plant illnesses in agricultural settings. We also examined the global distribution of tomato production and reviewed the most prevalent diseases impacting tomato crops, such as early blight, late blight, and bacterial spot. While conventional approaches such as visual inspection by farmers or agricultural experts have been instrumental historically, they suffer from critical limitations, including a high dependency on human expertise, limited accuracy under varying field conditions, and poor scalability. These drawbacks underscore the need for more reliable, automated, and scalable solutions, particularly in the face of increasing agricultural demands, climate variability, and the need to ensure food security.

This context lays the groundwork for the next chapter, which explores the emergence of smart farming practices. By integrating advanced technologies such as artificial intelligence (AI), the Internet of Things (IoT), and computer vision, smart farming offers promising avenues for early disease detection, precise crop management, and sustainable agricultural development.

Chapter 2

Smart Farming

2.1 Introduction

This chapter reviews previous research related to tomato leaf disease detec- tion using deep different learning techniques. It highlights key contributions, methodologies, datasets, and limitations of existing approaches. By analyzing these works we aim to identify research gaps and justify the need for the proposed solution in this thesis.

2.2 Smart Farming

2.2.1 Definition and Scope

The concept of smart farming or digital agriculture revolves around the modernization of agricultural practices through the integration of advanced technologies aimed at proving productivity. It commonly includes systems based on the internet of things (IoT), roboticsm sensors, satellite imaging and artificial intelligence (AI) [22]. These technologies work together to enable data-driven decisions, optimize resource use and improve crop management.

2.2.2 Technological components

Internet of things (IoT)

IoT devices, including soil moisture sensors, weather stations, and livestock trackers, enable real-time monitoring of various farm parameters. This continuous data collection allows for precise control over irrigation, fertilization, and

pest management, thereby enhancing crop yields and reducing resource wastage [23].

Robotics and Automation

The deployment of autonomous machines, such as drones and robotic harvesters, has revolutionized tasks like planting, weeding, and harvesting. These technologies not only help solve labor shortages but also increase operational efficiency and accuracy in field activities[24]. Today's farm robots can do more than some simple tasks , they can pick ripe fruits, check soil conditions and even spray pesticides exactly where needed. Crops are being watched over by drones with special sensors and cameras that can spot things like plant stress, pests or low moisture.

Artificial intelligence (AI)

AI algorithms analyze vast datasets to predict crop diseases, optimize planting schedules, and recommend resource allocation strategies. Machine learning models, in particular, have shown promise in identifying patters and anomalies in plants that may not be evident through traditional analysis [25].

Smart farming relies on a variety of advanced technologies, including sensors, cloud platforms, and mobile applications. However, at the core of its intelligence lies the use of data-driven approaches powered by Artificial Intelligence (AI). Among these, Machine Learning (ML) and Deep Learning (DL) have emerged as essential tools for automating complex tasks such as disease detection, yield prediction, and resource optimization.

To fully understand how smart farming systems make accurate and autonomous decisions, it is important to explore the theoretical foundations that support these technologies. The following section delves into the core concepts of Artificial Intelligence, with a particular focus on Machine Learning and Deep Learning, which play a pivotal role in enabling intelligent decision-making in modern agriculture.

2.3 Theoretical Foundations

2.3.1 Artificial Intelligence

Artificial intelligence (AI) is a transformative technology that enables machines to simulate human cognitive functions, including reasoning, learning, problem-solving, decision-making and autonomous operation.
Modern AI systems and AI-powered applications can analyze visual data, recognize images, adapt based on previous experiences, and generate insightful recommendations. Some function independently, reducing or even eliminating the need for human involvement[26]. Among the most impactful branches of AI are Machine Learning (ML) and Deep Learning (DL), which empower systems to automatically learn patterns from data and improve their performance over time without being explicitly programmed.

2.3.2 Machine Learning

Machine learning is a subfield of artificial intelligence that focuses on developing computer models capable of analyzing data, learning from it and making predictions or decisions autonomously. These models continuously improve their accuracy overtime as they process more information. The term ”machine learning” was introduced by Arthur Samuel in 1959, describing it as ”the ability of computers to learn without explicitly programming new skills” by processing datasets, machine learning algorithms apply mathematical models to generate outcomes. The main types of machine learning are supervised and unsupervised learning.[27].

Supervised and Insupervised machine learning

Insupervised machine learning is a branch of machine learning that relies on labeled datasets to train artificial intelligence models. These datasets con- tain input features and their corresponding outputs and by analyzing them the model learns to recognize patterns and relationships. The main goal of this learning process is to develop a model capable of making accurate predictions when exposed to new, real-world data. [28]
In supervised learning, labeled data includes inputs with known correct outputs

guiding the algorithm while processing data to fine-tunes its parameters until it can effectively generalize the relationships between features and labels and finally mapping the inputs into outputs.

Main Learning Tasks in Machine Learning

Machine learning algorithms are generally categorized based on the type of output they generate and the learning process involved. The three commonly used types are classification, and regression in the supervised learning, and clustering in the unsupervised learning [29].

1. **Classification:** is a supervised learning method where the algorithm learns from labeled data to assign input data to predefined categories or classes [29]. For example, in tomato disease detection, convolutional neural networks (CNNs) are trained to classify tomato leaf images into different disease categories [30] such as early blight, late blight, and healthy leaves .This allows automated, reliable diagnosis of the overall plant health status.

2. **Regression:** is another supervised learning technique, used to predict continuous output values based on input features [29]. In the context of tomato disease, regression models can estimate the severity of infection by predicting the percentage of the leaf area affected by disease lesions. This quantitative output helps farmers track disease progression and make precise management decisions [31].

3. **Clustering:** is an unsupervised learning technique that groups similar data points based on feature similarity [29] without the need for labeled data. In tomato disease context, clustering algorithms can identify diseased regions of leaves by grouping pixels with similar color and texture patterns [32]. This helps in detecting disease symptoms even when labeled datasets are unavailable.

2.3.3 Deep learning

Deep learning (DL) is a specialized subset of machine learning that focuses on utilizing neural networks with multiple layers known as deep neural networks to model and understand complex patterns in data. These architectures are designed to automatically learn hierarchical representations, enabling systems

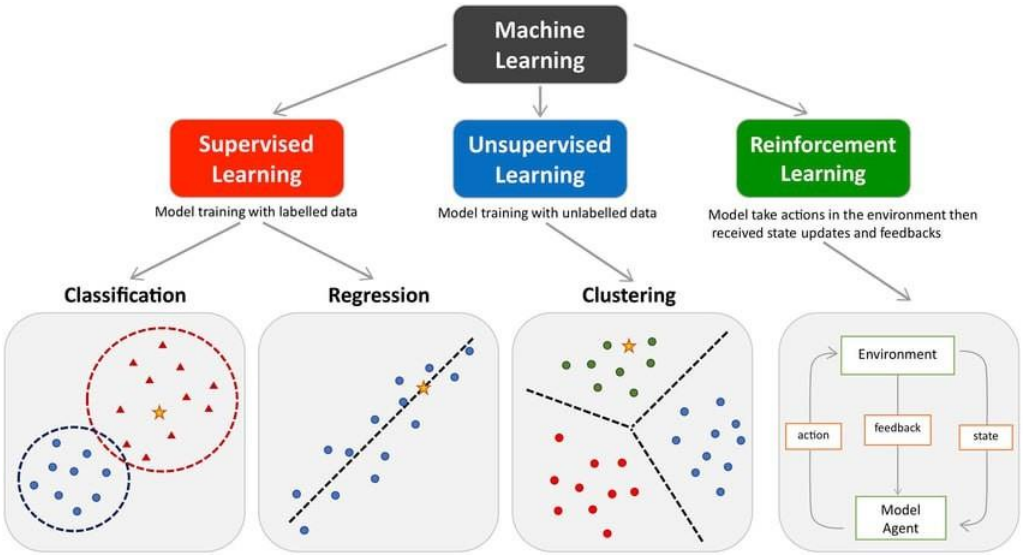


Figure 2.1: Machine Learning types and methods [11].

to process data with multiple levels of abstraction.(26).

2.3.4 Convolutional Neural Network CNN

A Convolutional Neural Network (CNN) is a deep learning architecture widely used for image classification, object detection, and pattern recognition tasks. CNNs are specifically designed to automatically and adaptively learn important features from input images, making them especially effective in visual tasks such as disease detection in plants, including tomato leaf diseases.

Generally CNN consists of multiple layers that transform the input data into increasingly abstract representations. These layers usually include convolutional layers, activation functions, pooling layers, and fully connected layers. The number of layers and their configurations (the number of layers, or filters in a convolutional layer , filter sizes , types of activation functions etc..) This can vary depending on the application demands and resource availability.

Convolution and Pooling Layers:

- *Convolutional Layers:*
  - The primary function of convolutional layers is to extract features from the input image using learnable filters (kernels).
  - Early layers often detect low-level features such as edges, corners, and textures.

  - As the depth increases, the network begins to identify more complex and abstract patterns, such as shapes or object parts.
  - Each convolution operation is typically followed by a non-linear activation function **ReLU (Rectified Linear Unit)**, which enables the network to learn non-linear patterns.

- *Pooling Layers:*
  - Pooling layers are a type of layers that reduce the spatial dimensions of the feature maps by selecting a specific value within a defined window (for example using Max Pooling the maximum value is selected).This downsampling operation helps in reducing computation, controlling overfitting, and obtaining the most important features. [33]
  - These convolution-pooling blocks can be repeated multiple times, depending on the network's depth and complexity requirements.

Transition to Dense Layers:
After several convolution and pooling layers, the feature maps are:

- **Flattened** into a one-dimensional vector to serve as input for the fully connected (dense) layers.
- Optionally passed through **Dropout layers**, which randomly deactivate a fraction of neurons during training to improve generalization and reduce overfitting.
- Sometimes normalized using **Batch Normalization**, which stabilizes and accelerates training by maintaining consistent data distributions throughout the network.

Fully Connected (Dense) Layers:

- These layers perform high-level reasoning based on the features extracted by the convolutional layers. Each neuron in a dense layer is connected to all neurons in the previous layer, allowing the model to learn global patterns and make decisions based on them.
- The final dense layer uses an activation function (for example Softmax) in multi-class classification tasks, which outputs a probability distribution across all target classes. The class with the highest probability is selected as the model's prediction.

Due to their powerful feature extraction and hierarchical learning capabilities, CNNs are highly effective in image classification tasks like tomato disease detection [34]. Depending on the computational constraints and the application environment (like mobile devices or embedded systems), CNNs can be designed as either shallow (fewer layers) or deep (many layers), balancing between accuracy and efficiency.

2.3.5 Explainable Artificial Inteligence

Explainable Artificial Intelligence (XAI) is a term that refers to a set of methods and techniques that make the decision-making process of artificial intelligence systems understandable to humans. It aims to open the *"Black box"* complex in deep learning models by providing clear explanations of how inputs are transformed into outputs. In practical terms, XAI enables users such as farmers, engineers, or researchers to understand why a model made a specific decision. For example, in tomato disease detection, XAI can highlight which regions of a leaf image were most responsible for classifying it as diseased or healthy using several techniques.

Explainable Artificial Inteligence XAI paradigms

Understanding the internal architecture and the decision making process of Deep Neural Networks (DNNs) is of great importance, because these models often function as black boxes [35]. Therefore, Several techniques aimed at improving interpretability through various visualization and analysis methods which have been proposed to achieve this goal .

a) Feature Visualization: The purpose of feature visualization is to show what an individual neurons or filters have learned. One common approach involves creating artificial inputs . A well-known example is the creation of artificial inputs which will optimally (maximally) activate the specific filters and reveal the type of pattern that these filters detect [35], [36], [37]. These inputs may be created by seeding them either randomly or with real data and optimized to enhance the activations of a targeted feature map. However, the absence of constraints in optimization often produces unrealistic patterns, so regularization methods are applied to maintain natural-lookingvisualizations although this

may slightly reduce their fidelity [37].

b) Saliency Maps: To explain model reasoning why it makes specific prediction from given inputs, attribution techniques examine the relevance of each input feature to the output and how it impacts it. These values are usually visualized in the form of saliency maps, which highlight the important regions of the input that has most influence on the model's decision [38]–[39]. Various methods exist, including simple gradient-basedones [36], like Grad-CAM which does activation weighting by the gradient [40],as well as layer-wise relevance propagation [41]. Saliency maps show great performance and are especially useful for data with visual structure (for instance images, spectrograms) [42]–[43]. However, they only provide explanations for individual predictions and this may be misleading if the computed relevance does not align well with the actual model behavior which can be problematic[44]–[45].

c) Representation Analysis: To gain insight into how a DNN generally processes inputs, researchers study and analyze the representations it generates. This includes evaluating how well different hidden layers encode specific features by training linear classifiers on their outputs [46], [47]. Representational similarity can also be studied using dimensionality reduction techniques like Principal Component Analysis (PCA)[48],Canonical Correlation Analysis (CCA) [49], or clustering algorithms[50], [51].Moreover, a number ofvisual interfaces and tools help explore learned representations and neuron activations [52]–[53],some of which illustrate how these representations change during training [54].
[55]

XAI Techniques for image classification

Several techniques have been developed to explain the predictions of deep learning models,especially, for image classification tasks like disease diagnosis. The following are among the most widely used :

a) Grad-CAM (Gradient-weighted Class Activation Mapping) An XAI technique that produces visual heatmaps showing which parts of the image were most in- fluential in the model's decision. It works by using gradients of a target class flowing into the final convolutional layer to produce a heatmap that highlights

the most relevant regions[56]. This is helpful for example in tomato disease detection to verify whether the model is focusing on actual infected areas rather than irrelevant parts.

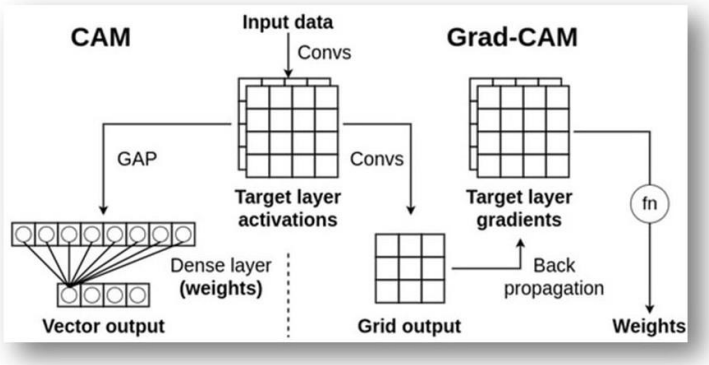


Figure 2.2: Grad-CAM work process [12].

b) LIME (Local Interpretable Model-angostic Explanations) LIME explains a model's decision by perturbing the input image and training a simple, inter- pretable model like linear regression[57]. For example around that prediction .It gives local explanations rather than global model insights.

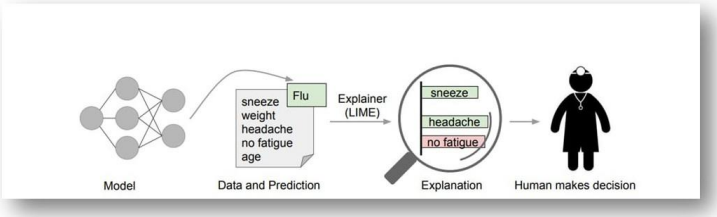


Figure 2.3: Philosophy behind LIME [13].

c) SHAP (SHapley Additive exPlanations) SHAP provides explanation values for each feature by calculating Shapley values from cooperative game theory. In this approach, each feature like a pixel region in an image is treated as a "player" in a game where the model's output is the" reward". This technique then focuses on fairly distribution of the contribution of each feature based on how much it changes the model's prediction when added or removed[58]. This helps understand which parts of an image had the most influence on a decision.

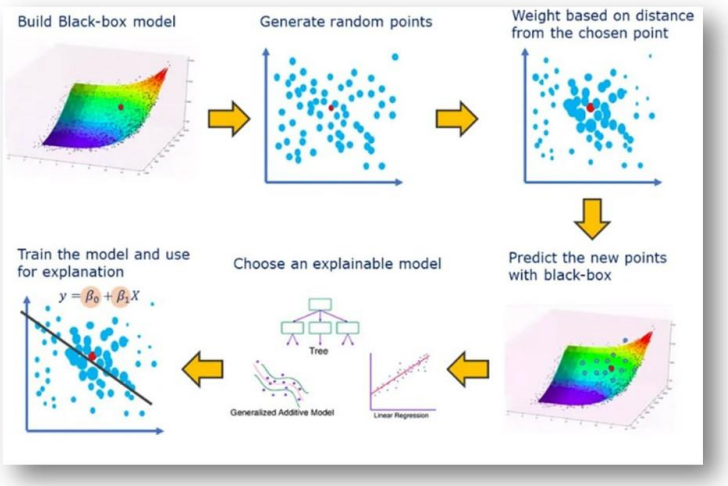


Figure 2.4: SHAP work process [14].

d) Score-CAM (Score-weighted Class Activation Mapping) It's an explainability technique that builds on Grad-CAM, but avoids using gradients, it uses instead the output score of the model to weight the activation maps of the final convolutional layer. By measuring the increase in class confidence when each activation map is passed through the model, it generates clear and sharp visual explanations[59]. The whole process goes in the following order:

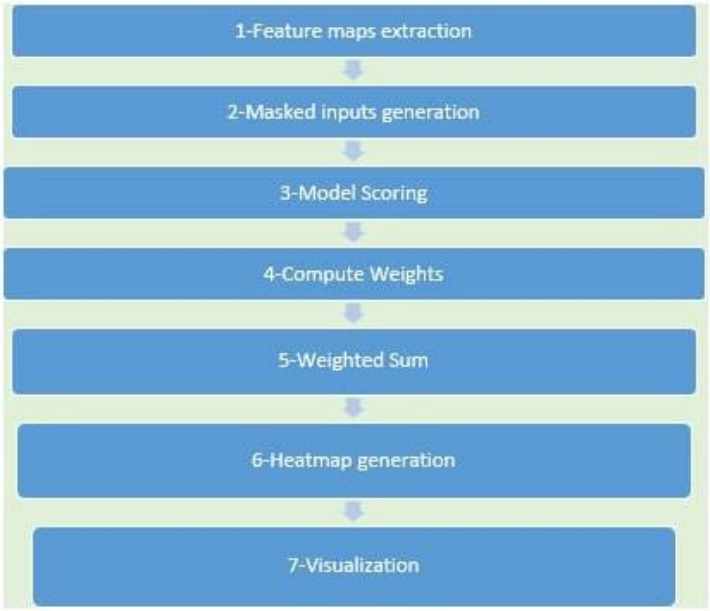


Figure 2.5: Score-CAM work Process[15].

In this thesis, we focus particularly on the Score-CAM technique, which is known for providing more reliable visual explanations without relying on

gradients. Score-CAM uses activation maps weighted by the class scores to identify the most influential regions in an image. To deepen our understanding and assess its effectiveness, we also present in this section the main variants of Score-CAM, which have been proposed to enhance the accuracy, stability, or speed of heatmap generation in image classification tasks.

2.4 Variant of Score-CAM

There are several proposed variations and improvements over the base Score-CAM to address its computational cost, smoothness, or efficiency. Here are the main ones:

2.4.1 Ablation-CAM

Ablation-CAM is a modification of Score-CAM. It estimates the importance of each activation pam by removing them individually and observing their effect on the model's prediction. Specifically, the target activation map is replaced with zeros, and the drop in the class prediction score is measured. This allows us to identify which maps contribute positively or negatively to the model's final decision.
Compared to Score-CAM, Ablation-CAM is more computationally efficient since it does not require additional image forward passes. Moreover, it of- fers deeper insight into the influence of each activation map on the outcome. [60]

2.4.2 Weighted Score-CAM (or Weighted Ablation-CAM)

It is an improved version of Ablation-CAM [60]. It does not completely remove activation maps, but instead assigns each map a smooth, continuous weight that reflects its relative contribution to the prediction. This approach enables the partial impact of each map to be measured rather than relying on an all- or-nothing approach, resulting in smoother and more accurate heat map inter-pretations.
Unlike Ablation cam, which relies on turning off each activation map completely to observe its impact, weighted score-cam assigns each map a partial score based on its influence, allowing for a more detailed understanding of its importance. These weighted maps are then combined to create the final heat map.

This approach produces smoother and more detailed visual interpretations and avoids the coarse effect of the on/off approach, providing a more accurate and precise view of which components actually contribute to the model's decision. [59]

2.4.3 Input-Aware Score-CAM

Input-Aware Score-Cam is an improvement over Score-Cam, providing a more accurate and detailed interpretation of the important parts of the image that influence the model's decision. This approach adds an additional dimension to the analysis of influence by taking into account the sensitivity of the original image, rather than just the activation maps as in traditional Score-Cam.

This approach relies on combining Score-Cam weights with input perturbation (saliency) information, that is, how small changes in the original image affect the model's outcome.

This fusion allows for the detection of fine, specific details, producing clearer, more focused interpretation maps. This method helps reduce noise in the activation maps, improving model accuracy and highlighting the most influential areas of the image more clearly and specifically, especially in cases where the regular score-cam results are fuzzy or out of focus.[61].

2.4.4 FIMF Score-CAM (Faster Score-CAM)

It is an improved and faster version of Input-Aware Score-Cam, aiming to interpret the deep learning model's focus on specific parts of an image more efficiently and in less time. It works on the same basic concept as Input-Aware Score-Cam, using activation maps and image sensitivity analysis. However, instead of using all activation maps, FIMF Score-Cam reduces the number of maps used, reducing computational cost. To achieve this, it uses dimensionality reduction techniques such as principal component analysis (PCA), which helps select the most useful maps while preserving essential information.

The selection is a small set of the most important maps, which are combined with the sensitivity information of the original image to form the final interpre- tive heat map.

This method reduces the number of calculations required (fewer passes), making it suitable for applications with high time requirements or deep and complex checks, while maintaining high quality.[62].

2.4.5 Smooth Score-CAM (SS-CAM)

Score-CAM Ensemble is a method designed to improve the clarity of heatmaps by reducing noise that might appear in standard visual explanations. The main idea is to make several slightly altered versions of the same image by adding a bit of random noise without changing the actual content. Then, Score- CAM is applied to each version, producing different heatmaps. By averaging these results, the method filters out random fluctuations and keeps only the consistent and important areas. This helps create cleaner and more focused visual interpretations, especially for images with fine details or small but critical regions. [63].

2.4.6 Layer-wise Score-CAM

Layer-wise Score-CAM is a version of Score-CAM that uses activation maps from multiple layers in the network, rather than relying solely on the last layer, to generate richer and more detailed interpretive maps.
Score-CAM is applied to each layer (first, middle, and last), and the resulting maps are then combined to produce a multi-level interpretation that combines fine-grained details with general patterns.
This approach allows for a deeper understanding of model decisions by simultaneously capturing contributions from low- and high-level features.[64]

2.4.7 Augmented Score-CAM

Augmented Score-CAM is an enhancement to Score-CAM that uses modified versions of the original image (such as flipping or rotating the image) to produce a clearer and more accurate interpretation map.
Score-CAM is applied to the original image and the modified versions, and the resulting maps are then combined into a final, more accurate and high-quality map.
This approach helps reduce noise and improve interpretation, especially in cases where the original Score-CAM map is inaccurate or unclear enough. [65]

2.4.8 Group CAM

Grouped Score-CAM is an improved version of Score-CAM that partitions activation maps into groups rather than treating them individually [66].

The maps are grouped based on:

- Spatial pattern similarity (visually similar regions).
- Channel similarity (maps that interact with the same features).
- Or predefined groups based on network layers (raw versus deep layers).

After partitioning, Score-CAM is applied to each group and combines the results into a final interpretive map that balances local detail with global visibility. This approach helps reduce noise and improve interpretation efficiency, especially for networks with a large number of activation maps.

Variation	Main Trick	Compared to TFLite Version
Ablation-CAM	Turns off each feature map at a time to see how much prediction drops	This method requires full model access and many re-runs which is not possible in TFLite
Weighted Score-CAM	Softly reduces influence of feature maps instead of turning them off completely	More precise but needs more model control which is not possible in TFLite
Input-Aware Score-CAM	Adds sensitivity to input changes	Uses extra saliency info (it needs to test how the output changes when the input changes) which is too complex for TFLite
FIMF Score-CAM	Uses PCA to reduce number of feature maps, speeds up Input-Aware Score-CAM	Similar goal (efficiency), but TFLite version avoids PCA and saliency
Smooth Score-CAM	Averages heatmaps from multiple noisy versions of the image	Requires rerunning model many times not suitable for TFLite
Layer-wise Score-CAM	Uses multiple layers, not just the last one	TFLite version only uses first layer due to architecture limits
Augmented Score-CAM	Uses flipped/rotated images to improve map quality	Not compatible TFLite can't rerun model with many variations
Group-CAM	Groups similar feature maps to analyze in batches	Too complex TFLite version processes maps individually

Table 2.1: Comparing Score-CAM variations with Adapted Score CAM with TFlite.

2.5 Related Works

In the following section, we explore three key aspects related to tomato disease detection. First, we examine recent research efforts that apply Deep Learning techniques for identifying and classifying tomato leaf diseases with high accuracy. Second, we review the integration of Explainable Artificial Intelligence (XAI) methods to enhance the transparency and interpretability of these mod- els, ensuring that the predictions are understandable and trustworthy. Finally, we highlight the development of mobile applications in this domain, showcasing how these AI-powered systems are being deployed in practical.

2.5.1 Deep Learning for Agricultural Image Classification

Deep learning has emerged as a powerful tool in agricultural image classification, enabling accurate and automated identification of plant diseases, pest infestations, and crop health conditions. In recent years, numerous studies have explored the use of Convolutional Neural Networks (CNNs) to classify images of leaves, fruits, and entire plants with high precision. Researchers have applied various deep learning architectures—including AlexNet, VGG, ResNet, and MobileNet, on agricultural datasets such as PlantVillage and real-world farm images. These works have demonstrated the superiority of deep learning models over traditional machine learning approaches in handling the complexity and variability of agricultural images. However, despite these advancements, many challenges remain, such as limited dataset quality, model interpretability, and deployment feasibility in resource-constrained environments, highlighting the need for further research and practical solutions.The following table 3.2 summarizes the key contributions and characteristics of these works.

2.5.2 Summary of Recent Deep Learning-Based Tomato Disease Detection Studies

Author	Model	Dataset	Contrib-ution	Perform-ance	Limit-ation	Year	Privte /Public
Jelali Mohieddine [67]	Faster R-CNN, YOLO (object detection and classification)	Real field datasets (PlantDoc, self-collected)	First comprehensive review of tomato disease detection using DL	Accuracy: Above 90% with improved CNN architectures	Overoptim-istic results on PlantVillage dataset; need for large real-world datasets	2024	Public
Muzam-mil Khan [68]	Inception-V3, Faster R-CNN (object detection and classification)	Plant-Village, PlantDisease (6000 images)	Hybrid CNN-RNN method for tomato leaf disease classification	Accuracy: 97.44%	Requires powerful GPU, focus only on EB & LB diseases	2024	Public

Selvara-jah Thusee-than [5]	Siamese Network-based Lightwe- ight Framework (Classifica-tion)	Plant-Village, Taiwan tomato leaf disease dataset	Novel lightweight framework for tomato leaf disease recognition using small and imbalanced data	Accuracy: 96.97% (PlantVil-lage), 95.48% (Taiwan)	Struggles with real-world samples; limited performance without augmentation	2024	Public
Mohit Agarwal et al.[69]	CNN (custom, 3 Conv + 3 Pool)	Plant- Village	Proposed a lightweight CNN model for classifying 9 tomato diseases and healthy leaves	Accuracy: 91.2% (Avg); Class-wise: 76% –100%	Lower performance on small class distinctions;	2020	Public

Abdelmalik Ouamane [70]	CNN with HOWSVD-MDA (object detection and classification)	Plant-Village, Taiwan tomato leaf dataset	Novel CNN-based approach with tensor subspace learning for improved classification	Accuracy: 98.36% (PlantVillage), 98.39% (Taiwan)	Struggles with real-world applicability; requires complex preprocessing steps	2024	Public

Diye Xin & Tianqi Li [71]	FCHF-DETR (Faster-Cascaded-attention-High-feature-fusion-Focaler Detection-Trans-former) based on RT-DETR-R18 (object detection and classification)	Tomato Leaf Diseases Detection CV & Tomato Diseases Multiple Sources (Kaggel) +512 custom images	Light- weight high-precision model for real-time detection	performa- nce with 96.4% precision, 96.7% recall, and 97.2% accuracy	Trained on only 3147 images, affecting gen-eralization. Doesn't adapt to different climate conditions.	2024	Public

Xuewei Wang & amp; Jun Liu [72]	TomatoDet (Transformer + YOLOv8n) with Swin-DDETR, Meta-ACON, and IBiFPN (object detection and classification)	Selfco-llected dataset	Combined Transformer with YOLOv8n for effective real-time detection. -Improved feature extraction using Swin-DDETR. - Applied Meta-ACON to enhance global information capture.	- mAP: 92.3% - FPS: 46.6 on Tesla T4	Not yet tested in open-field agricultural environ-ments.	2024	Public

Abdus Sobur,Md Humayun Kabir, Md. Zakir Hossain, Anwar Hossain & Imran Chowdhury Rana [73]	Dense-Net121, Res-Net50V2, AlexNet, Hybrid Model (Clas-sification)	Tomato leaf disease dataset from Bangladesh	Study on tomato leaf disease detection using ML and DL models under varying environmenta l conditions in Bangladesh	Hybrid Model: 97.80% accuracy	Dense-Net121 had higher loss (0.2932), AlexNet is older and simpler with less power	2025	Public
Yong Li, Changxia Sun, Zhengdao Song, Qian Liu, Haiping Si, Yingjie Yang [74]	EMA-DeiT (Classifica-tion)	Dataset of Tomato Leaves, PlantDoc, Tomato-Village	Improved tomato disease recognition system using EMA-DeiT	99.6% on PlantVillag	Limited data complexity in real- world scenarios, challenges	2025	Public

Table 2.2: Table summarizing the related works.

2.5.3 XAI Methods Applied to leaf tomato Image Classification

Author	Model	Dataset	Contrib-ution	Perform-ance	Limit-ation	Year	Privte /Public
Saurav Sagar, Mohammed Javed, David S Doermann [4]	Various CNN (efficient net dense net and google net) and Transformer models + Explainable AI (XAI) (Grad CAM /++ LIME and SHAPE)	Various datasets including PlantVillage, Tomato Leaf Disease Dataset	Introduced Explainable AI (XAI) to enhance interpretability in plant disease detection	Accuracy: 98.8% with deep learning models; improved trans-parency with XAI	XAI methods like Grad-CAM/ LIME are still developing and may not always provide accurate or stable explanations.	2023	Public

M. Shoaib, T. Hussain, B. Shah, I. Ullah, S. M. Shah, F. Ali, and S. H. Park [75]	CNN with Score-CAM (Classification + Visualization)	Custom tomato leaf dataset	Used Score-CAM for interpretable CNN-based disease detection	Accuracy: 97.25%	Limited real-world dataset scale	2022	Public
Gookyi, D.A.N. , Wulnye, F.A. , Wilson, M. , Danquah, P. , Danso, S.A. , Gariba, A.A. [76]	TensorFlow Lite deployment on Edge Impulse	Tomato leaf datasets	Deployed DL models for tomato disease detection on edge devices with TensorFlow Lite	Accuracy: 96.4%	Model performance may degrade without proper quantization and pruning before TFLite conversion	2023	Public

Aritra Das, Fahad Pathan, Jamin Rahman Jim, Momotaz Rahman Ouishy, Md Mohsin Kabir, M.F. Mridha [77]	XLTLDisNet (Lightweight Convolutional Neural Network (CNN)), Integrated Grad-CAM and LIME	PlantVillage subset (Tomato leaves only)	Proposed a lightweight CNN model (XLTLDisNet) for tomato leaf disease detection, Integrated Grad-CAM and LIME for model interpretability, Achieved high accuracy with fewer trainable parameters, Web-based deployment for farmer use	Accuracy: 97.24%	Class imbalance (few samples for mosaic virus, many for yellow leaf curl virus), Misclas-sification due to shad-ows/background noise,Hosting on free server (Vercel), limited scalability	2025	Public

Table 2.3: Table summarizing the XAI related works.

2.5.4 Mobile Applications in Agricultural Diagnosis

Author	App	Model	Dataset	Contrib-ution	perfor-mance	Limit-ation	Year
Asani, Emmanuel Oluwatobi and Osadeyi [78]	mPD-APP	Convolutional Neural Network (CNN)	14 plant disease categories; Sub-Saharan datasets	Mobile-enabled app for early diagnosis, aiming to improve food security in Africa	Accuracy: 93.91%	Dataset size is moderate; geographic focus may limit general-izability	2023
Rimon, Saiful Islam and Islam, Md Rakibul and Dey, Ashim and Das, Annesha [79]	PlantBuddy	Deep Convo-lutional Neural Network (DCNN)	54,305 images covering 26 plant diseases	Android mobile app for broad plant disease detection; user-friendly design for farmers	Accuracy 97%	Performance metrics not fully detailed; potential challenges in processing diverse disease symptoms	2021

A. Elhassouny, F. Smaran-dache [80]	Smart Tomato App	CNN-based architecture	Tomato leaf images; 10 disease classes (Plant Villag)	Mobile app specifically recognizing tomato leaf diseases using CNN	Accuracy up to 90.3%	Limited to tomato leaf diseases; precision details not fully disclosed	2019
A. Debnath, M. M. Hasan, M. Raihan, N. Samrat, M. M. Alsulami, M. Masud, A. K. Bairagi [7]	EfficientNet Tomato App	Efficient-NetV2B2 + Explainable AI (LIME, Grad-CAM)	Tomato leaf dataset (Kaggle) + PlantVillage	Smartphone + web app with multilingual support; integrates XAI for transparency	Accuracy: up to 100%	complexity of explainability techniques may impact app speed	2023

Table 2.4: Table summarizing the XAI related works.

2.6 Conclusion

As noted in the studies, there is remarkable improvement in the use of deep learning models like CNN and YOLO for monitoring tomato disease. Many researchers achieved above 95% accuracy using the PlantVillage dataset or custom images with DenseNet, Inception-V3, and hybrid models showing commendable results.

Regardless of the advancements made, there remains a number of issues in these works. The majority of the models are trained and tested in closed setups lacking consideration for real world limitations like offline accessibility and the variable weather conditions where user specific instructions are necessary. Also, while a few recent works using Explainable AI techniques frame XAI with Grad-CAM or LIME, the sponsorship of Score-CAM is heavily lacking in com- pact and TFLite mobile-ready formats.

Differently from prior studies, this work seeks to understand how the mobile application with offline access works by embedding a lightweight deep learn- ing model optimized with TensorFlow Lite. Moreover, this work employs a lightweight variant of Score-CAM, making it well-suited for deployment on mobile devices. This represents a key contribution of our research, as it ensures the system remains both accurate and interpretable, even in resource-constrained agricultural environments. In addition, the application offers adaptive user guidance based on real-time local weather conditions.

Chapter 3

Methods

This chapter presents the methods and techniques employed in the development of our mobile application. We begin by outlining the overall system workflow using a flowchart, followed by a detailed description of the convolutional neural network (CNN) architecture used for classification. Next, we explain the process of converting the trained model into the TensorFlow Lite (TFLite) format to enable deployment on mobile and embedded devices. Finally, to enhance the interpretability of the model's predictions, we integrate our contribution, a lightweight explainable AI (XAI) technique, Lightweight CAM, an optimized variant of Score-CAM, which generates heatmaps that highlight the most influential regions contributing to each decision.

3.1 The process of the proposed system

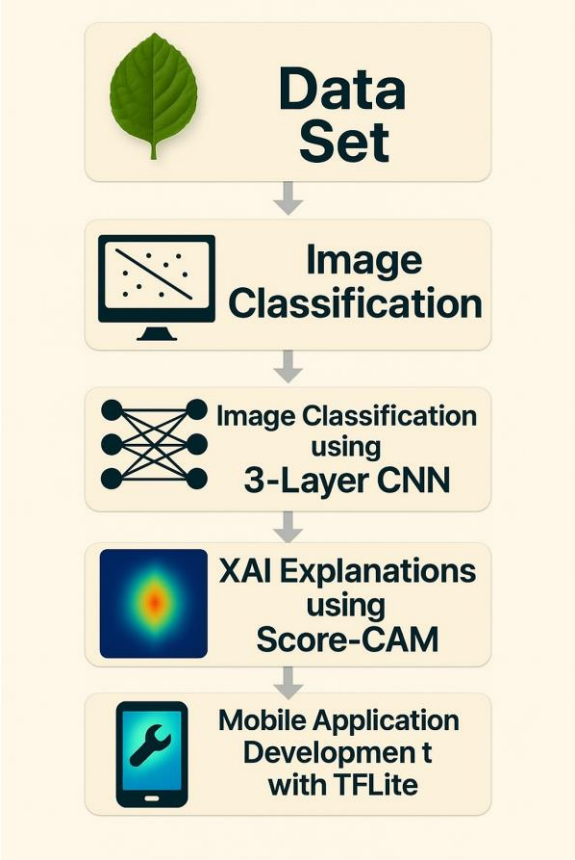


Figure 3.1: FlowChart of the proposed system

3.2 Dataset and Preparation

Dataset and preparation are critical steps in building an effective deep learn- ing model for agricultural image classification. In this study, images of tomato leaves were gathered from publicly available datasets, the PlantVillage. The collected data included various classes representing healthy and diseased leaves affected by different types of foliar diseases. To ensure the quality and consis- tency of the dataset, several preprocessing steps were applied, including resizing, normalization, and noise removal. In addition, data augmentation techniques such as rotation, flipping, zooming, and contrast adjustment were employed to increase dataset variability and prevent model overfitting. This prepara- tion stage was essential to enhance the model's generalization capability and performance in real-world conditions.

3.3 Classification model with CNN

Our model is based on a Convolutional Neural Network (CNN) architecture that we built with TensorFlow and Keras libraries, composed of three layers (with a kernel size of 3×3) using ReLU as the activation function. ReLU (Rectified Linear Unit) introduces non-linearity into the network and helps prevent the vanishing gradient problem by setting all negative values to zero, which speeds up training and improves model performance.

- The first convolutional layer contains 32 filters.
- The second layer has 64 filters.
- The third layer includes 128 filters.

Each convolutional layer is followed by a Max-Pooling layer (with a kernel size of 2×2), which reduces the spatial dimensions of the feature maps, thereby decreasing the number of parameters and computational cost while retaining the most important features. After the convolutional blocks, a Flatten layer is applied to convert the 3D feature maps into a 1D vector, making the data compatible with fully connected layers. The purpose of this fully connected layer is to learn complex, high-level representations by combining the features extracted by previous layers. Then a dropout layer is applied to prevent over-fitting by randomly disabling 50% of the neurons during training, which forces the model to learn more robust patterns. Finally, the output is passed through a dense layer with softmax activation, which produces a probability distribution over the target classes, allowing for multiclass classification.

3.4 Hyperparameters Configuration

The hyperparameters used for training the model are summarized in Table(3.1):

Element	Value
Optimizer	Adam
Learning Rate	0.001
Loss Function	Categorical Crossentropy
Epochs	50
Batch Size	32
Early Stopping	Enabled

Table 3.1: Hyperparameters used in the model training

3.5 Optimization for Mobile Deployment: Lightweight Models

A lightweight model is an artificial intelligence model specifically designed to be efficient in term of memory usage, speed, and resource consumption, mak- ing it suitable for implementation on divices with limited capabilities such as smartphones, embedded devicesn and the Internet of Things (IoT).
These models aim to deliver acceptable performance (high prediction accuracy) while minimizing the number of parameters and computations, thereby reduc- ing execution time and model size.
To achieve this goal, of techniques is relied upon, such as:

- **Depthwise Separable Convolutions:** It reduces the computational size compared to traditional convolutions and is used in networks such as Mo- bileNet.
- **Model Compression:** Compress the model through techniques such as quantization to convert weights from float32 to int8, reducing size and execution speed.
- **Network pruning:** To remove weak or unimportant weights.
- **Neural Architecture Search:** To choose the best network structure in terms of performance and efficiency.
- **Knowledge Distillation:** To transfer the knowledge of a large model to a smaller, lighter model.

These models play a pivotal role in achieving on-device AI, which allows processing to be performed locally on the device without the need to connect to the cloud, enhancing privacy and reducing response time. [6]

3.5.1 Model deployment with TensorFlow Lite (TFLite)

TensorFlow Lite (TFLite), a framework developed by Google, is designed to execute deep learning models on mobile and embedded systems. TFLite allows for on-device inference by transforming trained models into smaller, optimized versions through quantization, a process that diminishes size and increases speed while preserving accuracy. This capability is crucial for real-time applications in areas with limited or no internet access like farms or far fields.

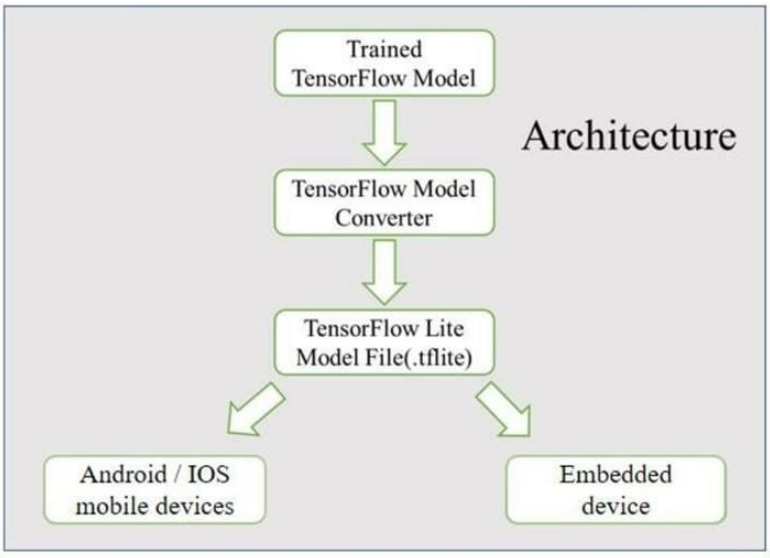


Figure 3.2: Architecture and use of TfLite. [16]

In order to practically implement the lightweight strategies discussed above, we proceeded to convert our trained model into a mobile-optimized format.
After training and evaluating the model, it was saved in the HDF5 (.h5) format. This format enables storing both the model's architecture and the learned weights in a single file, facilitating future reuse or deployment. To integrate the training model into a mobile application, we converted the model from (.h5) to (.tflite) with TensorFlow Lite (TFlite converter). This conversion reduces the model size and improves inference speed, making the application more efficient and capable of running offline without using internet.

3.6 Explainable Artificial Intelligence (XAI) with Enhanced Score-CAM

Score-CAM takes the weights of each activation map that supports the predicated class by following a specific process:

- First, each activation map from the selected convolutional layer is resized to the same dimensions of the input image,
- Than normalized to the range [0, 1] to serve as a mask,
- This mask is than multiplied with the original image, that producing a masked-image

$$K.M = M_k \tag{3.1}$$

- The masked image is passed through the model to measure the influence of the corresponding activation map on the model's prediction, This process is repeated for all activation maps, and the resulting class scores are stored.
- Finally, the weighted combination of all activation maps is computed to produce the final heatmap, using the formula:

$$ReLU\left(\sum_k S_k.A_k\right) = H \tag{3.2}$$

where S_k is the score for the masked image and A_k is the corresponding activation map.[63]

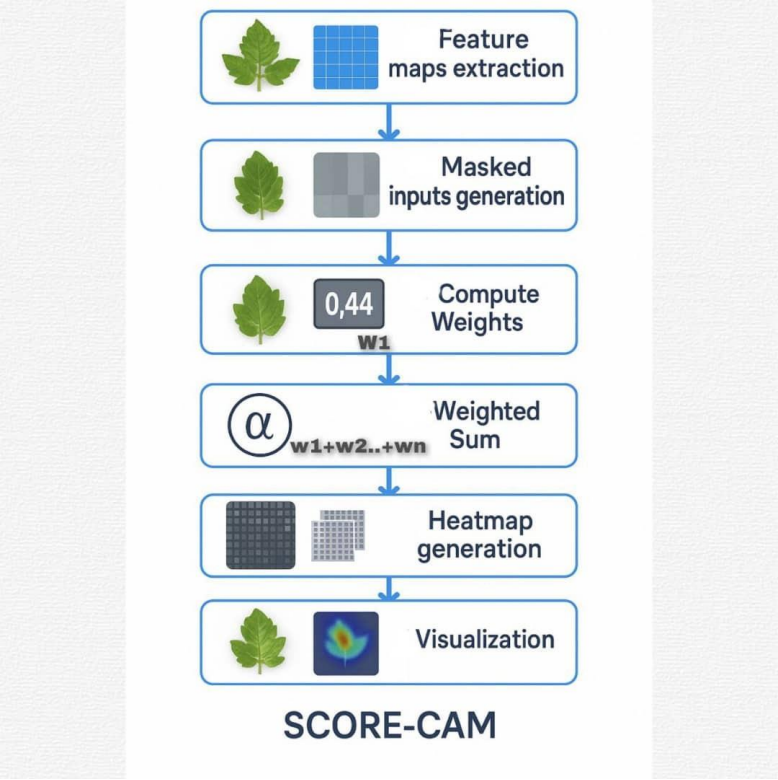


Figure 3.3: Standar SCORE CAM

Due to the limitations of our model after conversion, applying the original Score-CAM technique becomes challenging, because it does not allow intermediate layer access or dynamic re-inference with masked inputs, which are essential steps for recomputing class scores.
To overcome this, we adopted an alternative strategy based on two separate models:

- one for classification.
- and an other for heatmap extraction.

3.7 The Proposed Lightweight CAM

In this work, we propose Lightweight CAM, a novel variant of the popular Score-CAM explainability technique. Our method is specifically optimized for deployment alongside lightweight convolutional neural networks (CNNs) on resource-constrained devices such as smartphones and embedded platforms.

The main advantageous of this method is the Fast Weighting Mechanism

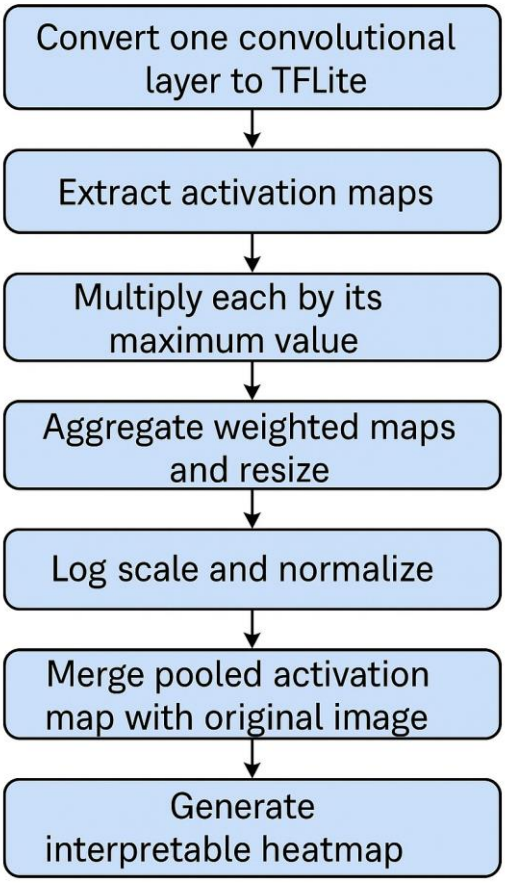


Figure 3.4: Proposed Lightweight CAM

3.7.1 Fast Weighting Mechanism

Rather than relying on the model's confidence scores for each activation (as in the original Score-CAM), we introduce a statistical weighting scheme. This method uses max values to assign weights, eliminating the need for repeated forward passes through the CNN MODEL and further accelerating the process. Specifically, we converted the first convolutional layer of the original model to a TFLite to extract all the activation maps. Then we extract the maximum value of each activation map.

$$max(Ai) = V\,i \tag{3.3}$$

We multiplying each activation map A_i by its maximum value V_i, enhances the influence of the most important activation maps.

$$A_i \cdot V_i = M_{weighted} \tag{3.4}$$

After that, we aggregated the weighted activation maps and resized the result to 150×150 pixels.

$$\sum_{i=1}^{N} A_i(x, y) \cdot V_i = pooledMap \tag{3.5}$$

To enhance visual contrast in the resulting heatmap, we applied logarithmic scaling (log scaling) (especially useful when values are closely distributed) followed by normalization to the [0, 1] range.

Finally, we overlaid the normalized map on the original image to generate an interpretable heatmap, visually indicating the regions that most influenced the model's prediction.

3.8 Technologies Used (Flutter Dart)

3.8.1 Flutter Dart

Flutter is an open-source UI toolkit developed by Google, that allows for the creation of high-performance, cross-platform applications from a single code. It supports Android, Iso, Web and even desktop platform. In Flutter every application is written with the help of Dart, which is also developed by Google.[81]

Figure 3.5: Flutter Apps [17].

3.9 Conclusion

In this chapter, we presented the methods and techniques used to develop our mobile application. Our main contribution lies in proposing a lightweight and mobile-optimized deep learning solution for tomato leaf disease diagnosis. We first described the CNN model used for training and classification. Then, a key part of our contribution is the conversion of this model into the Tensor- Flow Lite (TFLite) format, enabling efficient, low-resource, and offline deploy- ment on mobile devices, a critical need in agricultural settings with limited infrastructure. Furthermore, we integrated an explainable AI (XAI) method by proposing a novel lightweight version of the Score-CAM technique, tailored specifically for mobile compatibility. This addition improves the interpretability of model predictions while maintaining system efficiency, making our approach both innovative and practical in real-world farming environments.

Chapter 4

Experiment and results

4.1 Introduction

This chapter, we present the results of our experiment that we obtained while developing our mobile application, starting with the results of the CNN model, then the heatmap obtained using one of the XAI techniques (Score-CAM compatible withe TFLite model), in addition to the weather conditions, and finally a comprehensive description of the application using the TFLite.

4.2 Dataset

To build our application, we used a dataset of tomato leaf images from the PlantVillage dataset accessibles from Kaggle platform. This dataset contains 38 classes diseases and healthy leaves, representing 14 different plant species. In this study, we selected a subset of the PlantVillage dataset, consisting of 14529 tomato leaf images divided into ten classes: nine representing different diseases and one representing healthy leaves. These classes include: *Bacterial Spot, Early Blight, Late Blight, Mold Leaf, Mosaic Virus, Septoria Leaf Spot, Spider Mites, Target Spot, Yellow Leaf Curl Virus,* and *Healthy*, as shown in Figure 1. Each image in this dataset has an RGB color model (three channels) with dimensions of 256 X 256 pixels.

4.3 Preprocessing

Before training, we resized all tomato leaf images from PlantVillage dataset to 150 x 150 pixels and normalized them by rescaling their pixel values to the [0, 1] range. This preprocessing standardizes input dimensions and improves

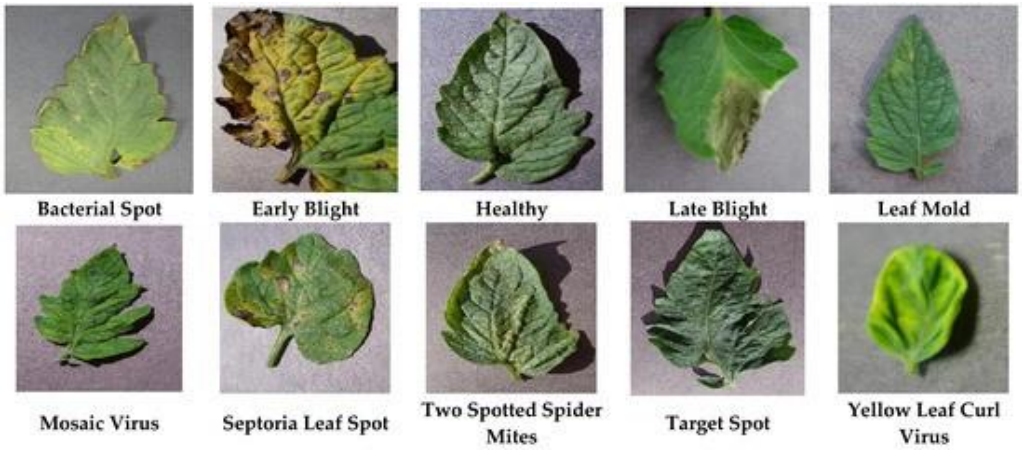


Figure 4.1: Simple image of the dataset [18].

model convergence by avoiding large gradient updates. To ensure proper model evaluation, we split dataset into 60% for training, 30% for validation and 10% for testing, while preserving class distribution across all subsets.

To improve the model's generalization and reduce overfitting, we applied real-time data augmentation exclusively to the training set. The augmentation techniques included random rotation (+20 degrees and -20 degrees), width and height shifts (20%), zooming and horizontal flipping. These transformations helped simulate real-word variability leaf orientation and appearance, enabling the model to learn more robust and invariant features.

4.4 Model Training, Validation, and Testing

The dataset of tomato leaf images was split into three subsets: training, validation, and testing. The training set was used to teach the CNN model to recog- nize patterns associated with different leaf diseases. The validation set helped monitor the model's performance during training and tune hyperparameters to prevent overfitting. Finally, the test set provided an unbiased evaluation of the model's generalization ability. Performance metrics such as accuracy, precision, recall, and F1-score were used to assess the effectiveness of the trained model in correctly identifying healthy and diseased leaves.

4.5 Model Performance Evaluation

Our model achieved a training accuracy of 95% , indicating effective learning. On the validation group, the model gets 93% accuracy .
The final evaluation on the test set achieved 95.11% accuracy , confirming that the model generalizes well and maintains stable performance on unseen data. as shown in Table (4.1).

	Accuracy (%)
Training	95
Validation	93
Test (Evaluation set)	95.11

Table 4.1: Performance results of the model on different data splits

Classification Report:

	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.96	0.90	0.93	166
Tomato___Early_blight	0.94	0.90	0.92	69
Tomato___Late_blight	1.00	0.95	0.97	147
Tomato___Leaf_Mold	0.97	0.96	0.97	79
Tomato___Septoria_leaf_spot	0.91	0.99	0.95	139
Tomato___Spider_mites Two-spotted_spider_mite	0.90	0.91	0.90	130
Tomato___Target_Spot	0.80	0.80	0.80	112
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	0.99	0.99	442
Tomato___Tomato_mosaic_virus	1.00	0.75	0.86	20
Tomato___healthy	0.90	1.00	0.95	149

Figure 4.2: Classification report showing the precision, recall, F1-score, and support for each tomato disease class produced by the CNN model..

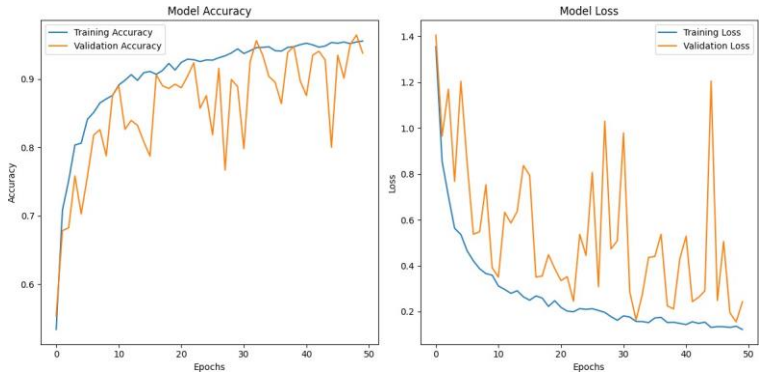


Figure 4.3: Plot illustrating training and validation accuracy and loss.

By comparing our work with that of Agrawal [69], we observe the following:

	Agrawal Results	Our Results
Model	CNN	**CNN**
Model size (.h5)	˜100MB	**217MB**
Convert to TFLite	No conversion mentioned	**Yes with TFLite converter**
Model size(.TFLite)	No TFLite model mentioned	**370KB**
Test Accuracy	91.2%	**95.11%**
XAI Integration	No XAI mentioned	**Yes, with heatmap generation**
Mobile APP	Not mentioned	**Yes, (Flutter + Offline Mode + Weather + Plant Care Advice)**

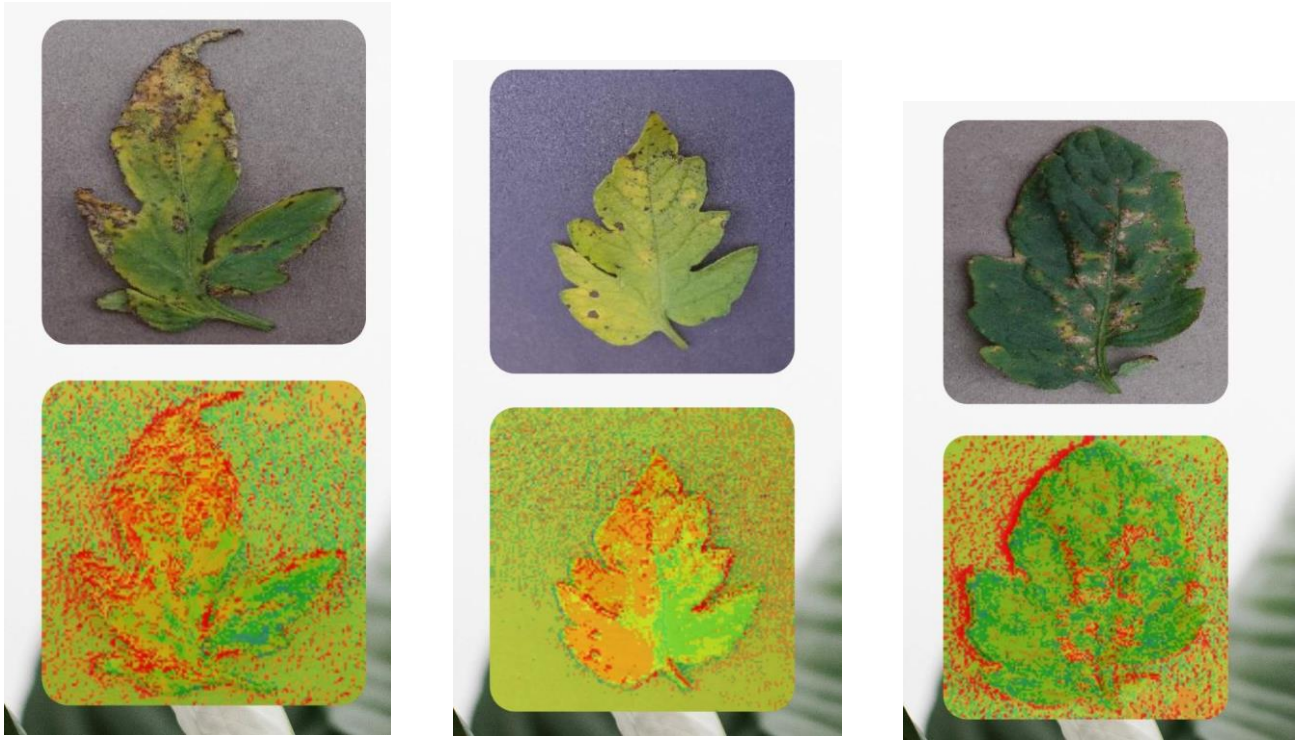
Table 4.2: Performance results of the model on different data splits

4.5.1 XAI Results and Heatmap Generation

In order to better understand the decision-making process of our classification model, we generated heatmaps that highlight the most influential regions of the tomato leaf image, as shown in Figure (4.3). Each color has a specific meaning, as illustrated in the following table:

Color	Meaning	Interpretation on leaf image
Red	Very high activation	Strong disease indication
Orange	High activation	Likely diseased zones
Yellow	Moderate activation	Potentially relevant areas
Green	Low activation	Healthy regions with little impact on prediction

Table 4.3: Table showing the heatmap colors.



(a) (b) (c)

Figure 4.4: Heatmaps of tomato leaf images.

4.6 Score-CAM Evaluation Metric

4.6.1 Intersection over Union (IoU)

Intersection over Union (IoU) is a evaluation metric used in computer vision tasks, such as object detection and image segmentation.
It measures the degree of overlap between the predicted region and the ground truth region by dividing the area of their intersection by the area of their union. The metric is defined with the following formula:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}$$

Where: A is the ground truth region, B is predicted region.
The IoU value ranges from 0 to 1. The closer to 1, the better model's performance.[82]

4.6.2 Time

In addition to the accuracy indicators, in this work we measured the execution time of Score-CAM for each image As shown in Table 4.4.

Images	Traditional Score-CAM	Enhanced Score-CAM
	IoU: 33% Time: 4.16 seconds	IoU: 26% Time: 0,13 seconds
	IoU: 21% Time: 3,7 seconds	IoU: 22% Time: 0,16 seconds
	IoU: 18% Time: 3,5 seconds	IoU: 22% Time: 0,11 seconds

Table 4.4: Comparison between Traditional and Enhanced Score-CAM in terms of Interpretation Accuracy (IoU) and Execution Time.

The result show that the enhanced Score-CAM achieved higher interpreta- tion accuracy (IoU) while significantly reducing the execution time, making it more suitable for real-time mobile applications.

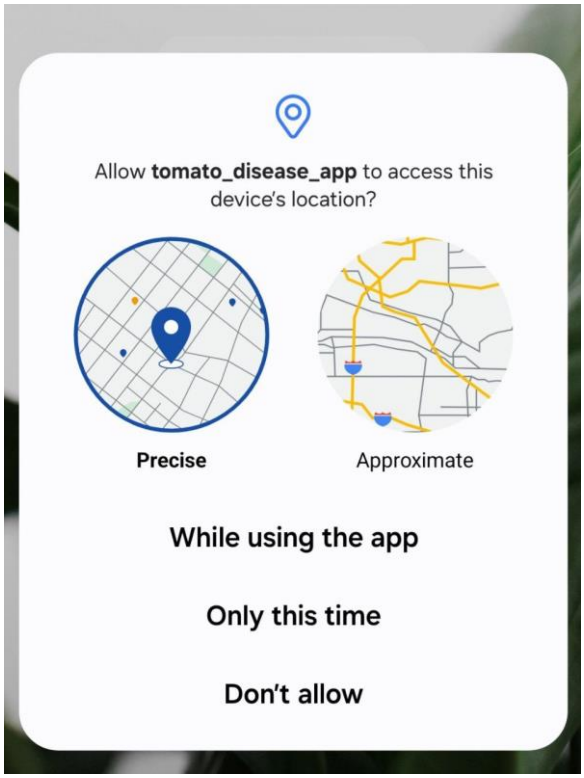
4.7 Mobile Application Development and Testing

4.7.1 Model Integration

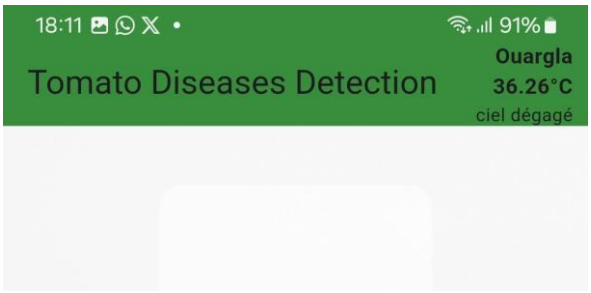
The trained CNN model was converted into a mobile-compatible format using TensorFlow Lite (.tflite). This conversion allows the model to be efficiently embedded into the mobile environment. The integration pipeline involves importing the .tflite model into the mobile application, loading it into memory at runtime, and using it to perform real-time predictions on input images.
For input processing, the application captures or uploads an image and applies the necessary preprocessing steps (resizing, normalization) to match the model's input requirements. For output handling, the model's prediction re- sults are retrieved, interpreted, and displayed to the user through a simple and intuitive interface.

4.7.2 Weather Integration Output

Upon requesting the user to enable location services as shown in Figure (a.4.4), the interface displays the city name, temperature, and weather conditions, Figure (a.4.4). At the same time this button reveals a suggestion box containing advices based on the current weather condition Figurer (4.5).

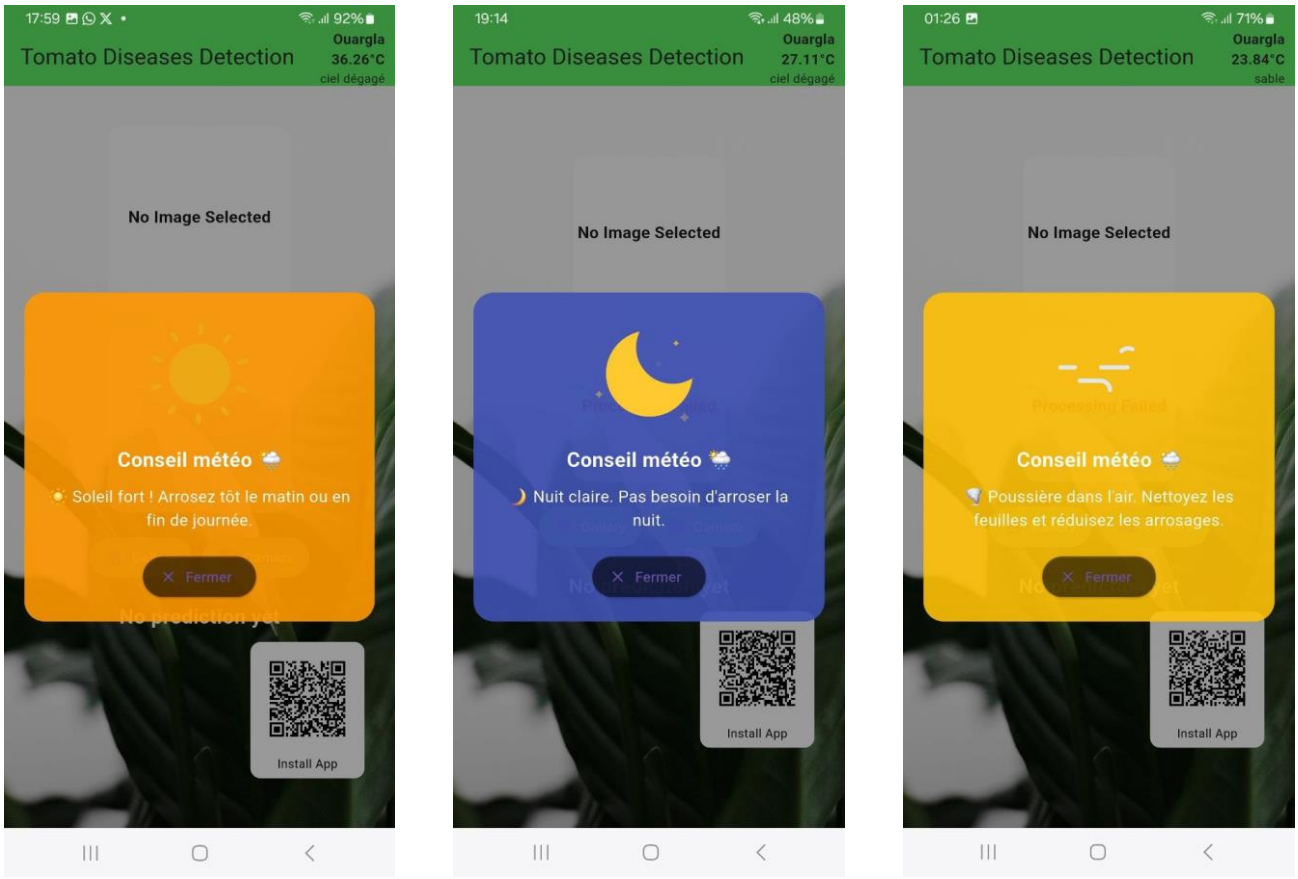


(a) Location request



(b) Weather information

Figure 4.5: Condition for displaying weather information on the interface.



(a) (b) (c)

Figure 4.6: Advices on weather conditions.

4.8 Offline Inference Using TFLite:

Integrating the model using TensorFlow Lite enables on-device inference di- rectly on the mobile application without requiring an internet connection. This approach is especially beneficial in agricultural settings where network connec- tivity may be limited or unavailable. Offline inference ensures faster response times and preserves user data privacy. Thanks to TFLite optimizations—such as quantization and reduced model size, the model maintains good performance while being well suited for the limited resources of mobile devices.

Chapter 5

Deployment Phase

5.1 Introduction

Deployment represents the final necessary stage when a trained and validated deep learning model is integrated into a usable application that is ready be delivered to its intended users. While training and evaluation usually take place in controlled environments with powerful computing resources, deployment focuses on practical considerations such as device limitations, offline access, latency, and usability in the field. For agriculture-based systems, especially in remote or resource-constrained regions, deployment must be lightweight, reliable, and customized especially to users with limited connectivity or technical expertise to ensure not only efficiency in solving the problem the application was made to solve but also to offer a user friendly experience while doing so.
In this project, our deployment of our mobile application that is designed for tomato disease detection has been approached in two parts: (1) the current offline deployment, which ensures accessibility in low-connectivity environments, and (2) are some proposed model update mechanisms for future enhancement without user-side complexity.

5.2 Basic Prototype phase

Before reaching its current state as a deployable mobile application, the tomato disease detection system underwent several key development stages:

- **Python Based Model Testing:** The initial phase focused on building and validating the core deep learning model using Python. Key libraries

included TensorFlow and Keras for model development, NumPy for numerical operations, ImageDataGenerator for data augmentation, and scikit-learn (via train test split) for dataset handling.

- **Web Based Interface Prototype:** A basic web interface was created using Flask as the backend framework to serve predictions and visualize results. Supporting modules included Werkzeug for secure file handling and render template for HTML-based interaction. This web application enabled users to upload leaf images, receive classification outcomes, and test the system's functionalities in a browser environment. It served as a bridge between early model testing and mobile deployment, offering a preliminary UI experience.
- **Transition to Mobile App:** The finalized version of the system was converted into a mobile application using Flutter for cross-platform UI development and TensorFlow Lite for on-device inference, enabling fast and offline tomato leaf disease detection on Android devices.

5.3 Current Deployment Strategy

To ensure immediate usability in real-world farming environments, our mobile application has been designed to operate entirely offline, without requiring an internet connection after installation.
Two distribution channels are currently in use:

5.3.1 QR Code Request System

- Users can request the application by scanning a QR code distributed by the development team,
- The scan generates a request that is automatically sent to the development team,
- Upon review the team's approval is confirmed through email,
- The application package then **(.apk)** is shared with the user allowed to install it manually on their Android device.

5.3.2 Direct Link via Google Drive

- Alternatively, the application may be accessed through a secure Google Drive link,
- This link is provided upon request and allows users to directly download and install the **.apk** file.

These methods serve several important purposes:

- They allow controlled distribution during the testing and optimization phase.
- They ensure compatibility with low-end mobile devices.
- They preserve offline functionality, which is essential for agricultural regions with unreliable connectivity.

Once installed, the application provides full functionality including: Tomato disease detection explainability of the obtained results of diseased and healthy tomato leaves through heatmap visualization, and weather-based advice using GPS.

The final version of our mobile application with a friendly interface that the user through early and fast detection of tomato plant diseases, without requiring an internet connection:

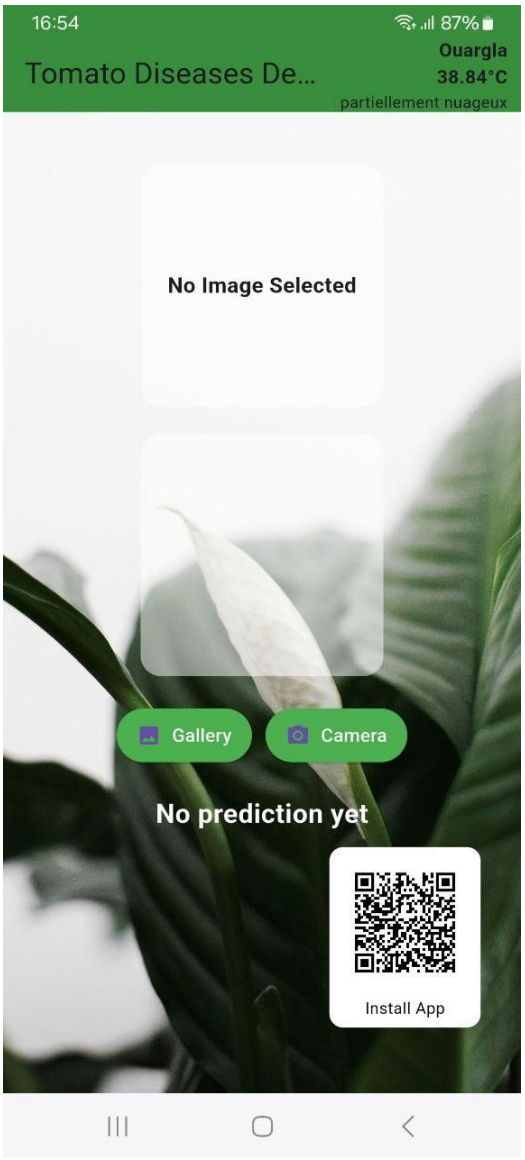
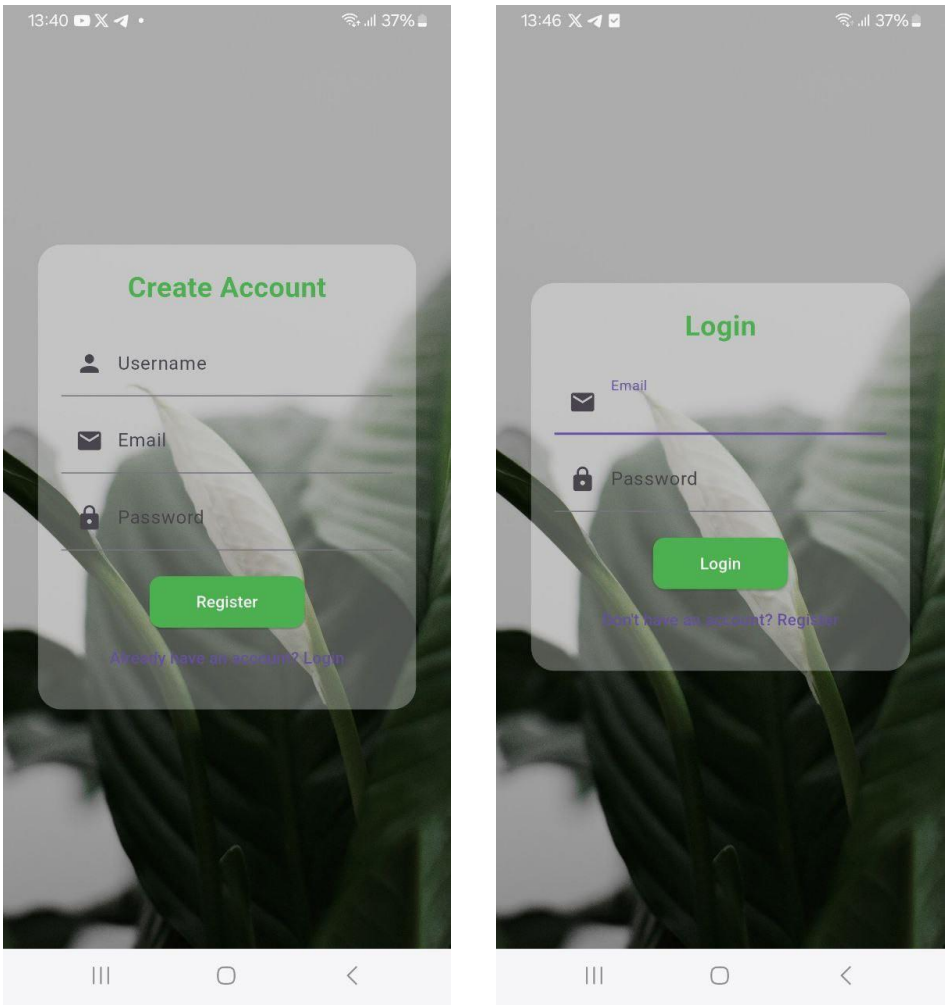


Figure 5.1: Simple image of user interface.

- The user starts by creating a new account or logging in using existing credentials through a simple and intuitive interface. This ensures secure and personalized access to the application features.



(a) (b)

Figure 5.2: Register and Login Screens of the Mobile Application.

- Captured or selected image from camera or gallery with its own heatmap.
- The predicted disease name is shown below.
- Treatment suggestions provided based on the diagnosis.
- Prevention advices to help avoid the disease in future cases.
- Weather information including temperature, description, and location.

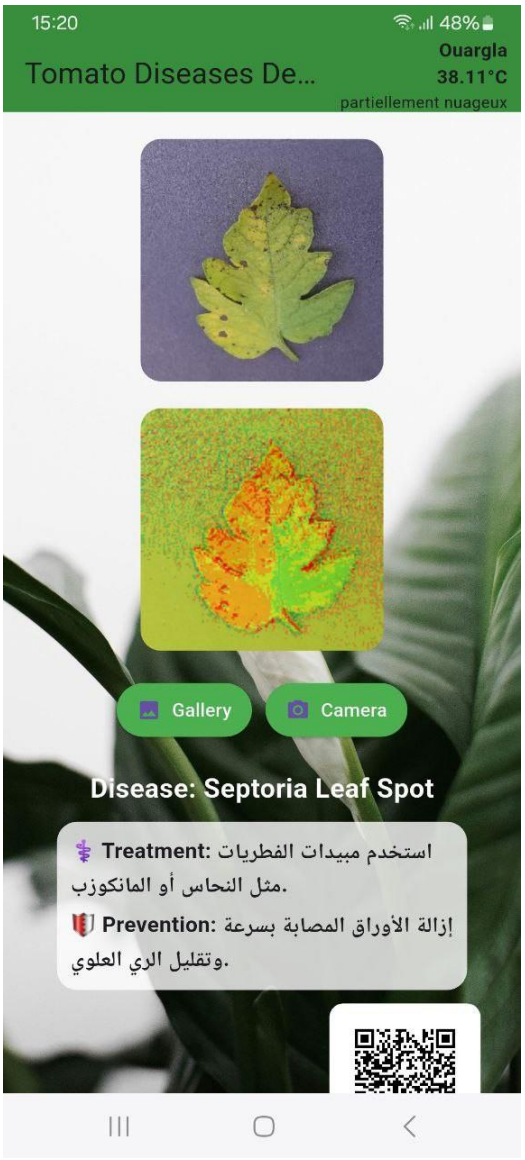


Figure 5.3: Simple image of user interface.

5.4 Proposed Future Strategy: Cloud-Based Model Syncing

While the current version operates fully offline, future iterations of the system will incorporate a cloud-based model synchronization mechanism to enable dynamic model updates. This hybrid deployment approach ensures that users can benefit from the latest model improvements while retaining offline capability.

5.5 Conclusion

The deployment strategy adopted in this project reflects the practical needs of agricultural users, particularly those seeking an effective, intelligent diagnostic tool for tomato diseases. By progressing from a simple script and web interface to a fully offline mobile application, the system ensures accessibility, ease of use, and reliability in environments where they are most needed.

The current version enables robust tomato disease detection, visual explainability through heatmaps, and weather-based advice all without requiring internet connectivity. However, the design does not stop here. To ensure the system continues to evolve and meet growing user needs, a future integration of cloud-based model synchronization is planned. This enhancement will allow dynamic model updates while preserving offline functionality.

Ultimately, this hybrid deployment strategy will strengthen the application's long-term usability and effectiveness, offering an optimal balance between of- fline reliability and online adaptability.

Chapter 6

General Conclusion

In this thesis, we successfully addressed the critical challenge of early and accurate detection of tomato leaf diseases by proposing and implementing a complete deep learning-based mobile solution. Recognizing the limitations of traditional methods that depended on manual inspections, expert resources which are not always readily available around. Our work introduces an innovative and responsive system that integrates modern artificial intelligence technologies, optimized model deployment, and explainability mechanisms to support smarter, more sustainable agricultural practices. We began by constructing a robust image classification model using a custom Convolutional Neural Net- work (CNN), chosen to detect and identify ten tomato leaf classes including nine common diseases including: Bacterial spot, Early blight, Late blight, Leaf mold, Septoria leaf spot, Spider mites, Target spot , Mosaic virus, and Yellow leaf curl virus and one healthy class based on 14,529 annotated images from the PlantVillage dataset. The preprocessing phase standardized the images to 150×150 pixels and included real-time data augmentation techniques to improve generalization. The CNN model showed a strong performance across all evaluation stages: Training Accuracy: 95%, Validation Accuracy: 93%, Test- ing Accuracy: 95.11%, with a .This confirms that the model is both effective in learning patterns and in generalizing to unseen data. To ensure real-world applicability.

The trained model was optimized and converted to the TensorFlow Lite (.tflite) format. This reduced the model size from 217 MB (.h5) to only 370 KB, en- abling offline, on-device inference without the need for internet connectivity es- pecially for remote agricultural contexts. This makes our solution not only fast and efficient but also practical for areas with limited digital infrastructure. A

major contribution of this thesis lies in the integration of Explainable Artificial Intelligence (XAI). Per our knowledge we implemented a version of Score-CAM technique that is compatible with TFLite's constraints and works under their limitations. By generating heatmaps that highlight areas of high relevance in each leaf image, the system benefits from interpretability, user trust and understandability enhancements. Color-coded regions in the obtained heatmaps ranging from red (very high activation) to green (low activation) provide visual explanations of the prediction results. The application was developed using Flutter and Dart, offering a cross-platform and user-friendly interface. It allows users to capture or upload tomato leaf images, instantly receive predictions regarding the presence/ absence of diseases, their types and access correspond- ing treatment and prevention advice. Further, we worked on improving the utility of the app by offering weather information and context-specific guidance based on the user's location, an important step toward adaptive and personal- ized agricultural decision support. In comparison with related works, our sys- tem distinguishes itself in multiple dimensions: Higher test accuracy (95.11%) surpassing the previous CNN-based models such as Agrawal's (91.2%) Inclusion of both XAI and weather-based guidance Lightweight deployment using TFLite, enabling offline inference. A fully functional mobile application supporting disease detection, explainability, and user guidance. Through this work, we contribute a scalable, interpretable, accessible, and easy-to-use diagnostic tool aimed at empowering farmers especially in under-resourced settings. By permitting early detection and actionable insight our application, makes sustainable crop management and improved food security feasible.

Future work will focus on expanding the dataset to include real-field images, extending the model to multi-crop diagnosis, and incorporating edge comput- ing for even more efficient performance. Nevertheless, this research lays the groundwork for the use of a strong foundation for AI-powered precision agriculture and marks a step forward in democratizing smart farming and agricultural technology.

Bibliography

[1] X. Chen et al. Identification of tomato leaf diseases based on combination of abck-bwtr and b-arnet. *Computers and Electronics in Agriculture*, 2025. In press or volume/page info missing.

[2] M. Dutot et al. Predicting the spread of postharvest disease in stored fruit, with application to apples. *Postharvest Biology and Technology*, 2013.

[3] Changxia Sun et al. Research on tomato disease image recognition method based on deit. *European Journal of Agronomy*, 162:127400, 2025.

[4] Saurav Sagar, Mohammed Javed, and David S Doermann. Leaf-based plant disease detection and explainable ai. *arXiv preprint arXiv:2404.16833*, 2023.

[5] Selvarajah Thuseethan, Palanisamy Vigneshwaran, Joseph Charles, and Chathrie Wimalasooriya. Siamese network-based lightweight framework for tomato leaf disease recognition. *Computers*, 13(12):323, 2024.

[6] Tasnim Shahriar. Comparative analysis of lightweight deep learning models for memory-constrained devices. *arXiv preprint arXiv:2505.03303*, 2025.

[7] Anjan Debnath, Md Mahedi Hasan, M Raihan, Nadim Samrat, Mashael M Alsulami, Mehedi Masud, and Anupam Kumar Bairagi. A smartphone-based detection system for tomato leaf disease using efficientnetv2b2 and its explainability with artificial intelligence (ai). *Sensors*, 23(21):8685, 2023.

[8] Food and Agriculture Organization of the United Nations. Home — food and agriculture organization of the united nations, 2025. Retrieved on 8 juin 2025.

[9] Zhen Zhang, Zhijie Wang, Lijie Huang, Jinhao Sun, Yujie Sun, and Xiang-dong Li. Tomato leaf disease detection using improved yolov5 model based on attention mechanism. *Remote Sensing*, 14(21):5465, 2022.

[10] Y. T. Demissie. Diagnosis of fungal and bacterial diseases based on symptom & sign. *American Journal of Plant Biology*, 4(4):57–66, 2019.

[11] R. R. Martin, D. James, and C. A. Lévesque. Impacts of molecular di- agnostic technologies on plant disease management. *Annual Review of Phytopathology*, 38:207–239, 2000.

[12] S. Farwah, M. Siddiqui, R. Mujeeb, N. Tabassum, and M. H. Rizvi. Introduction to disease and integrated disease management in solanaceous vegetables. *International Journal of Current Microbiology and Applied Sciences*, 9(11):1–10, 2020.

[13] University of Wisconsin-Madison Division of Extension. Bacterial spot of tomato — wisconsin horticulture, 2025. Retrieved on February 16, 2025.

[14] University of Minnesota Extension. Early blight of tomato and potato, 2025. Retrieved on February 17, 2025.

[15] University of Minnesota Extension. Late blight of tomato and potato, 2025. Retrieved on February 17, 2025.

[16] University of Minnesota Extension. Tomato leaf mold, 2025. Retrieved on February 17, 2025.

[17] S. M. Douglas. Septoria leaf spot of tomato, 2003. Retrieved on February 17, 2025.

[18] University of Kentucky. Two-spotted spider mite, 2025. Entomology Extension. Retrieved on February 17, 2025.

[19] K. MacKenzie, J. Chitwood, G. Vallad, and S. Hutton. Target spot of tomato in florida. 2018.

[20] University of Florida IFAS Extension. Tomato mosaic virus (tomv) and its management, 2023. Retrieved on February 17, 2025.

[21] North Carolina State University. Tomato yellow leaf curl virus, 2025. Retrieved on February 17, 2025.

[22] A. Gomstyn and A. Jonker. Smart farming, 2025. IBM Think. Retrieved on February 15, 2025.

[23] N. Jadhav, S. V, S. Premalatha, S. S, and R. B. Enhancing crop growth efficiency through iot-enabled smart farming system. *EAI Endorsed Transactions on Internet of Things*, 10, December 2023.

[24] Avinash Kumar Rai, N Kumar, D Katiyar, O Singh, G Sreekumar, and P Verma. Unlocking productivity potential: The promising role of agricultural robots in enhancing farming efficiency. *Int. J. Plant Soil Sci*, 35(18):624–633, 2023.

[25] Ersin Elbasi, Nour Mostafa, Chamseddine Zaki, Zakwan AlArnaout, Ahmet E Topcu, and Louai Saker. Optimizing agricultural data analysis techniques through ai-powered decision-making processes. *Applied Sciences*, 14(17):8018, 2024.

[26] IBM. Supervised learning — ibm think, 2025. Retrieved on February 21, 2025.

[27] Jakub Kufel, Katarzyna Bargie-l-Łaczek, Szymon Kocot, Maciej Koźlik, Wiktoria Bartnikowska, Micha-l Janik, Łukasz Czogalik, Piotr Dudek, Miko-laj Magiera, Anna Lis, et al. What is machine learning, artificial neural networks and deep learning?—examples of practical applications in medicine. *Diagnostics*, 13(15):2582, 2023.

[28] Staff writer Ivan Belcic and AI Models Cole Stryker, Editorial Lead. What is supervised learning? *IBM*, 2024.

[29] Solveig Badillo, Balazs Banfai, Fabian Birzele, Iakov I Davydov, Lucy Hutchinson, Tony Kam-Thong, Juliane Siebourg-Polster, Bernhard Steiert, and Jitao David Zhang. An introduction to machine learning. *Clinical pharmacology & therapeutics*, 107(4):871–885, 2020.

[30] Naresh K Trivedi, Vinay Gautam, Abhineet Anand, Hani Moaiteq Al- jahdali, Santos Gracia Villar, Divya Anand, Nitin Goyal, and Seifedine Kadry. Early detection and classification of tomato leaf disease using high- performance deep neural network. *Sensors*, 21(23):7987, 2021.

[31] Md Shofiqul Islam, Sunjida Sultana, Fahmid Al Farid, Md Nahidul Islam, Mamunur Rashid, Bifta Sama Bari, Noramiza Hashim, and Mohd Nizam Husen. Multimodal hybrid deep learning approach to detect tomato leaf

disease using attention based dilated convolution feature extractor with logistic regression classification. *Sensors*, 22(16):6079, 2022.

[32] Amir Saleh, Achmad Ridwan, and M Khalil Gibran. Machine learning and fuzzy c-means clustering for the identification of tomato diseases. *The Indonesian Journal of Computer Science*, 12(5), 2023.

[33] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[34] Zamir Osmenaj, Evgenia-Maria Tseliki, Sofia H Kapellaki, George Tselikis, and Nikolaos D Tselikas. From pixels to diagnosis: Implementing and evaluating a cnn model for tomato leaf disease detection. *Information*, 16(3):231, 2025.

[35] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[36] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[37] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google research blog*, 20(14):5, 2015.

[38] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[39] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[40] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations

from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[41] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise ex- planations for non-linear classifier decisions by layer-wise relevance propa- gation. *PloS one*, 10(7):e0130140, 2015.

[42] Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Interpreting and explaining deep neural net- works for classification of audio signals. *arXiv preprint arXiv:1807.03418*, 1, 2018.

[43] Lauréline Perotin, Romain Serizel, Emmanuel Vincent, and Alexandre Guérin. Crnn-based multiple doa estimation using acoustic intensity fea- tures for ambisonics recordings. *IEEE Journal of Selected Topics in Signal Processing*, 13(1):22–33, 2019.

[44] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[45] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise ex- planations for non-linear classifier decisions by layer-wise relevance propa- gation. *PloS one*, 10(7):e0130140, 2015.

[46] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[47] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quan- titative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[48] James Fiacco, Samridhi Choudhary, and Carolyn Rose. Deep neural model inspection and comparison via functional neuron pathways. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguis- tics*, pages 5754–5764, 2019.

[49] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in neural information processing systems*, 31, 2018.

[50] Tasha Nagamine, Michael L Seltzer, and Nima Mesgarani. Exploring how deep neural networks form phonemic categories. In *Interspeech*, pages 1912–1916, 2015.

[51] Andreas Krug, Maral Ebrahimzadeh, Jost Alemann, Jens Johannsmeier, and Sebastian Stober. Analyzing and visualizing deep neural networks for speech recognition with saliency-adjusted neuron activation profiles. *Electronics*, 10(11):1350, 2021.

[52] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e15, 2019.

[53] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.

[54] Mingwei Li, Zhenge Zhao, and Carlos Scheidegger. Visualizing neural networks with the grand tour. *Distill*, 5(3):e25, 2020.

[55] Valerie Krug, Raihan Kabir Ratul, Christopher Olson, and Sebastian Stober. Visualizing deep neural networks with topographic activation maps. In *HHAI 2023: Augmenting Human Intellect*, pages 138–152. IOS Press, 2023.

[56] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[57] Aarshay Kumar. Everything you need to know about lime, 2022.

[58] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[59] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual expla- nations for convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 24–25, 2020.

[60] Harish Guruprasad Ramaswamy et al. Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In *proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 983–991, 2020.

[61] Rakshit Naidu, Ankita Ghosh, Yash Maurya, Soumya Snigdha Kundu, et al. Is-cam: Integrated score-cam for axiomatic-based explanations. *arXiv preprint arXiv:2010.03023*, 2020.

[62] Jing Li, Dongbo Zhang, Bumin Meng, Yongxing Li, and Lufeng Luo. Fimf score-cam: fast score-cam based on local multi-feature integration for visual interpretation of cnns. *IET Image Processing*, 17(3):761–772, 2023.

[63] Haofan Wang, R Naidu, J Michael, and SS Kundu. Ss-cam: Smoothed score-cam for sharper visual feature localization (2020). *arXiv preprint arXiv:2006.14255*, 2006.

[64] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Transactions on Image Processing*, 30:5875–5888, 2021.

[65] Rami Ibrahim and M Omair Shafiq. Augmented score-cam: High resolution visual interpretations for deep neural networks. *Knowledge-Based Systems*, 252:109287, 2022.

[66] Q Zhang, L Rao, and Y Yang. Group-cam: Group score-weighted visual explanations for deep convolutional networks. arxiv 2021. *arXiv preprint arXiv:2103.13859*.

[67] Mohieddine Jelali. Deep learning networks-based tomato disease and pest detection: a first review of research studies using real field datasets. *Frontiers in Plant Science*, 15:1493322, 2024.

[68] Muzammil Khan, Fahmida Gulan, Muhammad Arshad, Abnash Zaman, and Ammara Riaz. Early and late blight disease identification in tomato plants using a neural network-based model to augmenting agricultural productivity. *Science Progress*, 107(3):00368504241275371, 2024.

[69] Mohit Agarwal, Abhishek Singh, Siddhartha Arjaria, Amit Sinha, and Suneet Gupta. Toled: Tomato leaf disease detection using convolution neural network. *Procedia Computer Science*, 167:293–301, 2020.

[70] Abdelmalik Ouamane, Ammar Chouchane, Yassine Himeur, Abderrazak Debilou, Slimane Nadji, Nabil Boubakeur, and Abbes Amira. Enhancing plant disease detection: A novel cnn-based approach with tensor subspace learning and howsvd-mda. *Neural Computing and Applications*, 36(36):22957–22981, 2024.

[71] Diye Xin and Tianqi Li. Revolutionizing tomato disease detection in com- plex environments. *Frontiers in Plant Science*, 15:1409544, 2024.

[72] Xuewei Wang and Jun Liu. An efficient deep learning model for tomato disease detection. *Plant Methods*, 20(1):61, 2024.

[73] Abdus Sobur, Md Humayun Kabir, Md Zakir Hossain, Anwar Hossain, and Imran Chowdhury Rana. Enhancing tomato leaf disease detection in varied climates: A comparative study of advanced deep learning models with a novel hybrid approach. *International Journal of Creative Research Thoughts— IJCRT*, 12(2), 2024.

[74] Changxia Sun et al. Research on tomato disease image recognition method based on deit. *European Journal of Agronomy*, 162:127400, 2025.

[75] Muhammad Shoaib, Tariq Hussain, Babar Shah, Ihsan Ullah, Sayyed Mudassar Shah, Farman Ali, and Sang Hyun Park. Deep learning-based segmentation and classification of leaf images for detection of tomato plant disease. *Frontiers in plant science*, 13:1031748, 2022.

[76] Dennis Agyemanh Nana Gookyi, Fortunatus Aabangbio Wulnye, Michael Wilson, Paul Danquah, Samuel Akwasi Danso, and Awudu Amadu Gariba. Enabling intelligence on the edge: Leveraging edge impulse to deploy multiple deep learning models on edge devices for tomato leaf disease detection. *AgriEngineering*, 6(4):3563–3585, 2024.

[77] Aritra Das, Fahad Pathan, Jamin Rahman Jim, Momotaz Rahman Ouishy, Md Mohsin Kabir, and MF Mridha. Xltldisnet: A novel and lightweight approach to identify tomato leaf diseases with transparency. *Heliyon*, 2025.

[78] Emmanuel Oluwatobi Asani, Yomi Phineas Osadeyi, Adekanmi A Ade- gun, Serestina Viriri, Joyce A Ayoola, and Ebenezer Ayorinde Kolawole. mpd-app: a mobile-enabled plant diseases diagnosis application using con- volutional neural network toward the attainment of a food secure world. *Frontiers in Artificial Intelligence*, 6:1227950, 2023.

[79] Saiful Islam Rimon, Md Rakibul Islam, Ashim Dey, and Annesha Das. Plantbuddy: an android-based mobile application for plant disease detec- tion using deep convolutional neural network. In *Artificial Intelligence and Technologies: Select Proceedings of ICRTAC-AIT 2020*, pages 275–285. Springer, 2021.

[80] Azeddine Elhassouny and Florentin Smarandache. Smart mobile applica- tion to recognize tomato leaf diseases using convolutional neural networks. In *2019 International Conference of Computer Science and Renewable En- ergies (ICCSRE)*, pages 1–4. IEEE, 2019.

[81] Aakanksha Tashildar, Nisha Shah, Rushabh Gala, Trishul Giri, and Pranali Chavhan. Application development using flutter. *International Re- search Journal of Modernization in Engineering Technology and Science*, 2(8):1262–1266, 2020.

[82] Rafael Padilla, Wesley L Passos, Thadeu LB Dias, Sergio L Netto, and Eduardo AB Da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3):279, 2021.