Enhancing Apple Leaf Disease Detection with Deep Learning: From Model Training to Android App Integration

Cahyono Budy Santoso[1,*], Marcello Singadji [2], Denny Ganjar Purnama[3], Saimam Abdel[4], Aqila Kharismawardani[5]

[1,2,3,4,5]*Department of Information System, Universitas Pembangunan Jaya, Blok B7/P, Jl. Cendrawasih Raya Bintaro Jaya, Tangerang Selatan 15413, Banten, Indonesia*

(Received: August 25, 2024; Revised: October 6, 2024; Accepted: November 23, 2024; Available online: December 29, 2024)

Abstract

This study presents an innovative approach to enhance apple leaf disease detection using deep learning by comparing three models: ReXNet-150, EfficientNet, and Conventional CNN (ResNet-18). The objective is to identify the most accurate and efficient model for real-world deployment in resource-constrained environments. Utilizing a dataset of 1,730 high-quality images, the models were trained using transfer learning, achieving significant results. ReXNet-150 outperformed other models with an F1-score of 0.988, precision of 0.989, and recall of 0.989. EfficientNet and ResNet-18 demonstrated competitive performances with F1-scores of 0.966 and 0.977, respectively. The integration of the ReXNet-150 model into a TensorFlow Lite-based Android application ensures real-time detection, enabling farmers and researchers to capture or upload images for immediate classification. The findings highlight ReXNet-150's robustness, achieving a test accuracy of 98.9% and minimal misclassification, making it ideal for practical agricultural applications. The novelty lies in bridging advanced deep learning with mobile deployment, addressing real-world constraints. Future work could extend this framework to multi-crop disease detection and real-time video analysis, providing scalable solutions for precision agriculture.

Keywords: Apple Leaf Disease Detection, Deep Learning, Rexnet-150

1. Introduction

Deep learning has emerged as a revolutionary method in agriculture, especially for the identification and categorization of apple diseases. Deep learning approaches, particularly Convolutional Neural Networks (CNNs), have demonstrated considerable potential in improving the accuracy and efficiency of disease detection in apple leaves. This literature review consolidates recent progress in deep learning techniques for apple disease diagnosis, emphasizing different models and their efficacy. Recent studies have shown the effectiveness of CNNs in detecting apple leaf diseases. Gao et al. introduced BAM-Net, which proficiently detects apple leaf diseases in intricate backdrops, attaining significant accuracy in classification tests [1]. Jiang et al. created a deep CNN model incorporating the Inception module, achieving a mean Average Precision (mAP) of 78.80% for five prevalent apple leaf diseases [1]. This underscores the potential of deep learning models to manage complex visual data, frequently a concern in agricultural environments.

You Only Look Once (YOLO) is a state-of-the-art, real-time object detection system. It analyzes an image in a single forward pass through the network, identifying objects and their locations simultaneously. Unlike traditional methods that process images in multiple stages, YOLO frames object detection as a single regression problem, making it highly efficient for applications requiring both speed and accuracy. Moreover, the integration of advanced architectures such as EfficientNet and YOLO has further enhanced detection capabilities. Alqahtani et al. introduced a method that combines the Sailfish Optimizer with the EfficientNet model, achieving high accuracy in apple leaf disease detection [2]. Additionally,, Wang et al. presented MGA-YOLO, a lightweight one-stage network that streamlines the detection process while maintaining high performance, showcasing the trend towards optimizing models for real-time applications [3]. This is crucial for practical implementations in the field, where timely disease detection can

***Corresponding author: Cahyono Budy Santoso (cahyono.budy@upj.ac.id)**
DOI: https://doi.org/10.47738/jads.v6i1.507

ISSN 2723-6471

significantly impact crop yield and quality. The use of hybrid models has also gained traction, as seen in the work by Di and Li, who proposed a new detection model based on an improved CNN architecture. Their model, DF-Tiny-YOLO, aims to enhance detection speed and accuracy by addressing the complexities of apple leaf images [4]. This aligns with the findings of Liu et al., who emphasized the importance of feature extraction capabilities in CNNs for effective disease classification [5]. The combination of CNNs with other techniques, such as K-means clustering for segmentation, has been shown to improve the identification of infected areas in leaf images [5]. Furthermore, the challenge of data scarcity in training deep learning models has been addressed through innovative approaches like data augmentation. Tian et al. utilized CycleGAN for augmenting image datasets, which enriched the diversity of training data and improved the robustness of their detection model [6]. This is particularly relevant in agricultural contexts where the occurrence of specific diseases may be sporadic, leading to limited training samples. In addition to CNNs, the incorporation of transformer architectures has been explored to enhance model performance. Li and Li's study introduced a hybrid model that combines CNN and transformer structures, improving the model's ability to extract both global and local features from images, thereby enhancing disease identification accuracy [7]. This reflects a broader trend in deep learning research, where hybrid models are increasingly being used to leverage the strengths of different architectures.

In conclusion, the literature indicates a significant advancement in the application of deep learning for apple disease detection. The integration of sophisticated CNN architectures, hybrid models, and data augmentation techniques has led to improved accuracy and efficiency in disease identification. As these technologies continue to evolve, they hold the potential to revolutionize agricultural practices by enabling early detection and management of apple diseases, ultimately contributing to enhanced crop yield and quality.

However, existing research presents several gaps. Many current models struggle with generalization when applied in real-world scenarios, often due to variations in environmental conditions or limited availability of training data. Furthermore, the integration of these models into mobile applications for real-time disease detection remains limited. This research aims to address these gaps by developing a deep learning model that is optimized for real-time apple disease detection and seamlessly integrated into an Android mobile application. This approach allows farmers and agricultural researchers to capture or upload images of apple leaves for immediate disease classification, making the model highly applicable in practical field settings.

The objective of this study is twofold: first, to identify the best-performing model among ReXNet-150, EfficientNet, and conventional CNNs for apple leaf disease detection. ReXNet-150, a CNN model known for its lightweight structure and high performance, offers a balance between computational efficiency and classification accuracy, making it suitable for resource-constrained environments such as mobile devices. Second, this study aims to bridge the gap between model development and practical deployment by integrating the best model into a mobile application. By converting the trained model into TensorFlow Lite format, the mobile application allows real-time disease detection, empowering farmers to make timely interventions.

2. Literature Review

Artificial Neural Networks (ANNs) and CNN have become essential technologies in artificial intelligence and machine learning. ANN are mathematical constructs derived from the biological neural networks of the human brain, engineered to process information via interconnected nodes or neurons, as illustrated in figure 1, which depicts the layered structure of ANN nodes and their interconnections. They possess the ability to learn intricate patterns and generate predictions from input data, resulting in their extensive utilization across diverse fields, including as image recognition, natural language processing, and medical diagnostics [8], [9], [10].

 ISSN 2723-6471

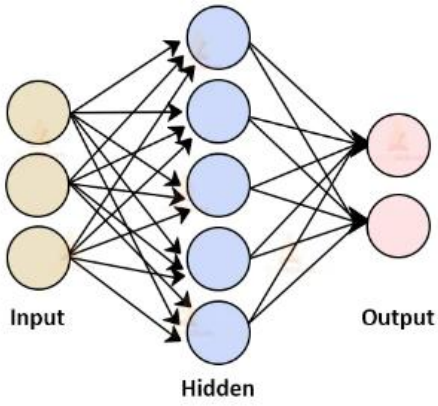

Figure 1. Artificial Neural Networks

This architecture enables CNNs to excel in picture categorization and object detection, frequently surpassing human performance in particular observational tasks [11], [12]. CNNs have been effectively utilized in the detection of lesions in mammography, showcasing their efficacy in medical imaging applications [13]. The CNNs architecture is shown in Figure 2.

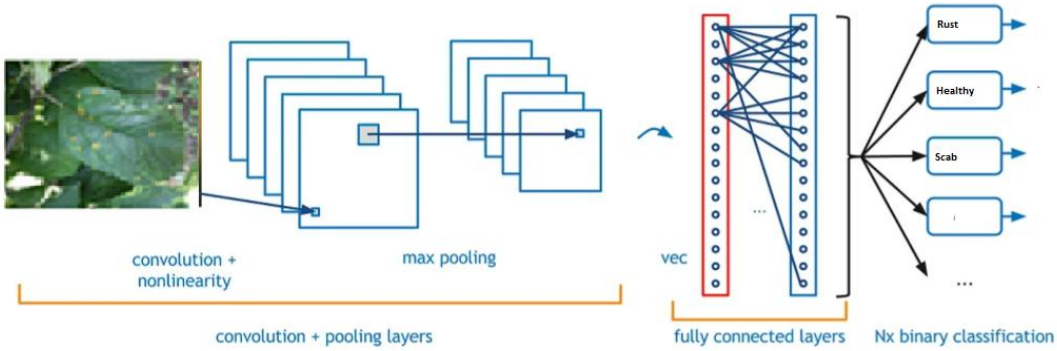

Figure 2. Convolutional Neural Networks

The capacity of CNNs to manage extensive datasets and extract pertinent characteristics has established them as a fundamental element of deep learning, resulting in considerable progress across many applications. The architecture of convolutional neural networks is especially significant. They employ convolutional operations that allow the network to discern local patterns in the data, essential for applications involving spatial data like as pictures [11], [12]. The hierarchical feature learning capabilities of CNNs enables them to identify both low-level characteristics (e.g., edges and textures) and high-level features (e.g., forms and objects), rendering them exceptionally successful for intricate picture recognition tasks [14]. Moreover, the implementation of new methodologies like the Faster R-CNN has augmented the efficacy of CNNs in object detection tasks by enhancing both the speed and precision of the detection process [13].

In addition to CNNs, the development of architectures like ReXNet has introduced new methodologies in designing neural networks. ReXNet, which stands for "Rectified Linear Unit (ReLU) eXtended Network," focuses on optimizing the efficiency of CNNs by employing a lightweight architecture that maintains high performance while reducing computational costs. This is particularly relevant in applications where resources are limited, such as mobile devices and embedded systems [11]. The evolution of these architectures reflects the ongoing innovation in neural network design, aimed at improving performance while addressing the challenges of computational efficiency and scalability. In summary, the advancements in ANN and CNN technologies, including the emergence of specialized architectures like ReXNet, highlight the transformative impact of these models in artificial intelligence. Their ability to learn from data and perform complex tasks has revolutionized various fields, from healthcare to autonomous systems, underscoring the importance of continued research and development in this domain.

 ISSN 2723-6471

3. Methodology

The research steps used are in figure 3 and the details are as follows:

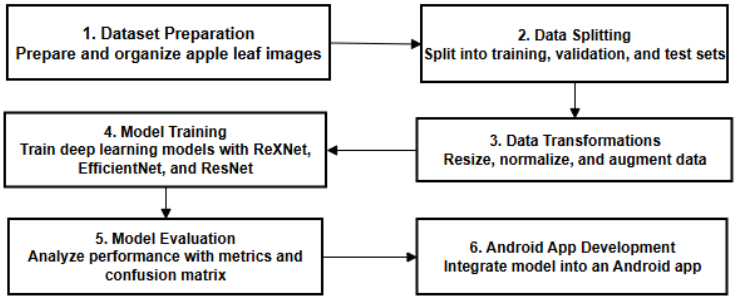

Figure 3. Method Flowchart

3.1. Dataset Preparation

The first step in the research methodology is preparing the dataset. The apple leaf disease images are organized in a folder structure, where each sub-folder represents a different class of disease. The dataset is loaded using Python's glob library, which scans the directories for image files, and the PIL.Image module is used to open the images in RGB format. This structured organization of data is essential as it allows the model to understand the class labels based on the directory structure. In this step, each image is associated with a corresponding label representing its class (e.g., healthy, diseased). The dataset is then passed to the CustomDataset class, which extends PyTorch's Dataset class and defines how to retrieve an image and its label from the dataset.

Apple orchards in the U.S. face persistent threats from numerous pathogens and insects, making early and accurate disease detection critical for effective disease management. Delayed or incorrect diagnoses can lead to either overuse or underuse of chemicals, resulting in higher production costs, environmental harm, and health risks. To address this, a dataset comprising 1,730 high-quality images of apple foliar diseases was manually captured, reflecting diverse conditions such as variable illumination, angles, surfaces, and noise. A subset of this data, annotated by experts for apple scab, cedar apple rust, and healthy leaves, was shared with the Kaggle community. However, the dataset's geographical and ecological diversity is not explicitly detailed, raising concerns about its representativeness.

3.2. Data Splitting

Once the dataset was prepared, it was divided into training (90%), validation (5%), and test sets (5%) using PyTorch's random_split function. This splitting approach ensures that the model is trained on one set of data while its performance is continuously evaluated on unseen validation data. The test set, entirely separate from the training process, was reserved to assess the model's generalization ability after training. This method of splitting helps prevent overfitting and allows for robust performance evaluation on unseen data, a practice commonly used in machine learning studies [15].

3.3. Data Transformations

Before the images were fed into the deep learning model, several transformations were applied to ensure data uniformity and improve model performance. Each image was resized to 224x224 pixels, a standard resolution used in many CNN-based image classification tasks [16]. Images were then normalized using the mean and standard deviation values calculated from the ImageNet dataset, as the model was initialized with pretrained weights from ImageNet [17]. This preprocessing step helps the model generalize better across different image samples, as normalization reduces the impact of lighting variations and other visual inconsistencies.

3.4. Model Training

The ReXNet-150 model was chosen for this study because of its efficacy and robust performance in classification tests. ReXNet-150 is a convolutional neural network (CNN) architecture optimized for computational efficiency, particularly in resource-limited settings. The model was refined utilizing a pretrained variant from the timm library. Training

utilized cross-entropy loss, a prevalent selection for multi-class classification tasks [18], and the Adam optimizer was employed to optimize model parameters.

EfficientNet is a family of convolutional neural network architectures designed to achieve a balance between high accuracy and computational efficiency. By leveraging a compound scaling method that uniformly scales depth, width, and resolution, EfficientNet achieves state-of-the-art performance with significantly fewer parameters and reduced computational costs compared to traditional CNNs. This makes it particularly suitable for resource-constrained environments, such as mobile or edge devices, where computational efficiency is critical.

Conventional CNNs, on the other hand, refer to standard deep learning architectures that typically consist of sequential layers of convolutions, pooling, and activations to extract hierarchical features from images. While these models, such as ResNet or VGG, have been widely used for image classification tasks due to their simplicity and robustness, they often lack the optimized scaling and parameter efficiency found in modern architectures like EfficientNet [19].

3.5. Model Evaluation

Upon completion of the training phase, the model was evaluated on the test set to measure its performance on novel data. Multiple essential measures were computed to offer an in-depth comprehension of the model's classification capabilities, including accuracy, precision, recall, and F1-score. The ratings illustrate the model's proficiency in properly differentiating among healthy, rust, and scab leaves, exhibiting minimal false positives and false negatives. Nonetheless, there was a minor misclassification of healthy leaves as diseased, indicating that additional refinement may be necessary to enhance the recall for the healthy category.

3.6. Android Application Development for Leaf Disease Detection

To ensure the trained model could be deployed in real-world agricultural settings, an Android application was developed. The goal of the app is to allow users, particularly farmers and agricultural researchers, to capture or upload images of apple leaves and classify the leaf condition in real-time. The app was built using Android Studio with the frontend designed in Java and Kotlin. To integrate the machine learning model, the trained model was converted to TensorFlow Lite format, allowing for efficient execution on mobile devices [20].

Converting a model to TensorFlow Lite typically involves saving it in a compatible format (e.g., SavedModel), applying the TensorFlow Lite Converter, and optionally optimizing it through techniques like quantization or pruning to reduce model size and inference time. However, challenges can arise, including unsupported operations in TensorFlow Lite, potential accuracy loss from quantization, and constraints posed by mobile hardware. These issues can be mitigated by replacing unsupported layers, using custom operators, selectively applying quantization, and testing the converted model's performance on target devices using the TensorFlow Lite Interpreter.

The key functionalities of this application include the ability to take photos, upload images, and display predictions based on the pre-trained model. The Android application was developed using Android Studio with the integration of TensorFlow Lite (TFLite) to ensure smooth execution of the deep learning model on mobile devices. The trained model from the previous stage) was converted into a TensorFlow Lite format to reduce the computational resources and optimize it for mobile environments. The app was designed using Java and Kotlin for frontend development, ensuring a user-friendly interface where users can either take photos of apple leaves directly or select images from their phone gallery.

The application follows a structured process flow to facilitate real-time leaf disease detection. Initially, users can input images by either capturing a photo using the device's camera or selecting an existing image from the gallery. Once the image is input, preprocessing steps are applied, which include resizing the image to a resolution of 224x224 pixels and normalizing it to meet the model's requirements, such as RGB format and mean-standard deviation normalization. After preprocessing, the TensorFlow Lite-based pre-trained model is invoked for inference. This step classifies the image into predefined categories, including healthy leaves and various disease types, such as apple scab, black rot, and cedar apple rust. Finally, the prediction results are displayed on the application interface, providing users with an immediate and accurate assessment of the leaf's condition. This streamlined process ensures that the app delivers user-friendly and efficient disease detection for agricultural applications.

 ISSN 2723-6471

The user interface was designed to be simple and intuitive. Upon launching the app, users are presented with two main options: to capture a new image using the camera or to select an existing image from their device's gallery. After choosing an image, the application processes the image, runs the classification model, and displays the results. The result page includes the predicted disease (or healthy condition)

4. Results and Discussion

4.1. Dataset Preparation

Figure 4 shown in the image represents a grid of apple leaf images with corresponding ground truth (GT) labels, indicating whether the leaf is classified as "healthy," "rust," or "scab." This visualization is an essential part of the data exploration and analysis phase during dataset preparation, which precedes the training of machine learning models. By randomly selecting and visualizing samples from the dataset, researchers can ensure that the dataset contains representative images for each class and that the distribution of categories (healthy, rust, scab) is well-balanced or properly addressed through augmentation if necessary.

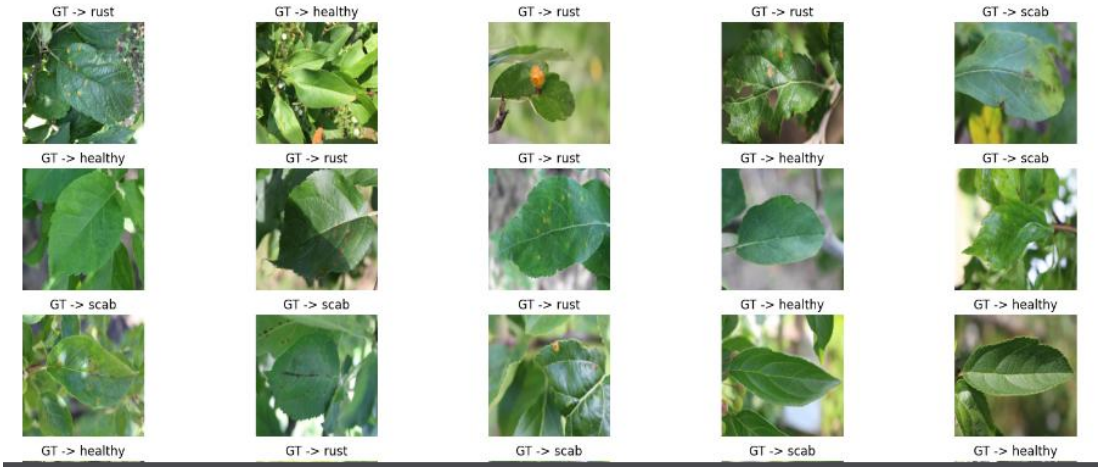

Figure 4. Grid of apple leaf images

Figure 5 represents a class imbalance analysis of the dataset, visualized using a bar chart. Each bar corresponds to a specific class of apple leaf condition: "healthy," "rust," and "scab," with the number of instances in each class shown at the top of the bars. The heights of the bars reflect the distribution of the dataset, with 510 samples for the "healthy" class, 572 for "rust," and 552 for "scab." This kind of analysis is crucial during the dataset preparation phase to assess the distribution of classes and determine whether class imbalance exists. Class imbalance, where one class significantly outnumbers others, can adversely affect model performance, particularly in classification tasks, as the model might become biased toward the majority class.

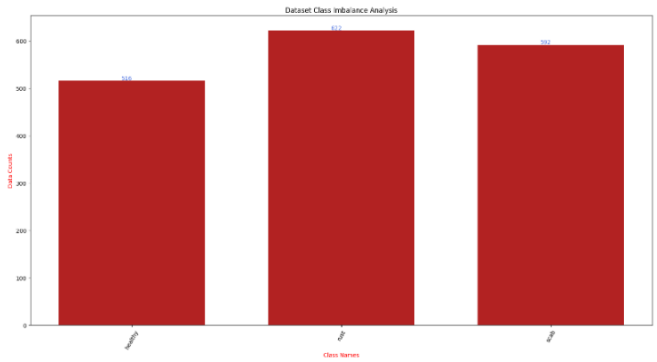

Figure 5. The class distribution of the apple leaf dataset

 ISSN 2723-6471

Understanding class distribution allows to decide whether techniques such as data augmentation, resampling, or applying class weights during training are needed to address imbalance. If the imbalance is significant, it might skew the model's predictions, leading to poor generalization, especially for underrepresented classes. In this specific case, the class distribution appears relatively balanced, minimizing the risk of bias, although some adjustment may still be necessary. Thus, this visualization serves as a critical component of the dataset preparation process, guiding data preprocessing decisions to ensure the machine learning model receives a representative and balanced dataset for training.

The data augmentation techniques implemented for the training dataset enhance model robustness by simulating various real-world scenarios that the model might encounter. Specifically, random horizontal flipping with a probability of 0.5 allows the model to learn features invariant to orientation changes, while random rotations up to 20 degrees further diversify orientation perspectives. Additionally, the use of color jittering with variations in brightness, contrast, saturation, and hue (up to 0.2 and 0.1, respectively) enables the model to adapt to different lighting conditions, making it more resilient to variations in image appearance due to environmental factors. Together, these augmentations—applied before normalization and tensor conversion—strengthen the model's ability to generalize effectively, improving its performance on unseen data.

4.2. Data Splitting

This study utilized PyTorch's random_split function to partition the dataset into training, validation, and test sets, allocating 90% for training, 5% for validation, and 5% for testing. This partitioning technique proved helpful in guaranteeing a balanced and resilient training procedure. Throughout model training, performance was continuously assessed using the validation set, facilitating early termination when the validation loss stabilized, so averting overfitting. The distinct test set was utilized to assess the model's generalization capability, confirming that the model was not simply memorizing the training data but was able to make precise predictions on novel samples. The model attained an accuracy of 98.9% on the test set, indicating robust generalization skills, attributable to the efficient data partitioning strategy that facilitated ongoing performance assessment and mitigated overfitting, a significant obstacle in deep learning endeavors. This data partitioning strategy, commonly utilized in machine learning, instills confidence in the model's efficacy in real-world applications.

4.3. Data Transformation

The use of data transformations, such as resizing and normalization, was essential for achieving the model's superior performance in this study. All images were scaled to 224x224 pixels, a common dimension for CNN-based models, ensuring consistent input sizes irrespective of the original image dimensions. The scaling process is crucial for ensuring consistency and compliance with the model architecture. Normalization was implemented utilizing the mean and standard deviation values derived from the ImageNet dataset. This phase was essential for enhancing the model's capacity to generalize across diverse situations in the dataset, as normalization mitigates the impact of elements like lighting fluctuations and other visual inconsistencies that could otherwise inject noise into the model's learning process. The integration of scaling and normalization markedly enhanced the model's accuracy to 98.9% on the test set, since these preprocessing techniques ensured optimal preparation of the input images for training, hence promoting more efficient learning from the data.

4.4. Model Training

The training process involved a comparative evaluation of three models: ReXNet-150, EfficientNet, and a conventional CNN aiming to identify the best-performing model in terms of both accuracy and computational efficiency. The ReXNet-150 model, selected for its lightweight structure and suitability for resource-constrained environments, was trained using a pretrained version from the timm library, leveraging transfer learning to accelerate convergence. Similarly, EfficientNet was trained as a benchmark model known for its optimized scaling across depth, width, and resolution, while the conventional CNN served as a baseline for performance comparison. All models were trained using cross-entropy loss, a standard loss function for multi-class classification, in conjunction with the Adam optimizer to ensure effective parameter updates. Over 20 epochs, early stopping was employed to prevent overfitting by halting training when the validation loss did not improve for five consecutive epochs. Continuous evaluation on the validation

 ISSN 2723-6471

set provided consistent feedback, ensuring the robustness of each model. The final trained models were evaluated on the test set.

The training results for ReXNet-150 (figure 6) demonstrate consistent improvements in both training and validation performance during the early epochs, followed by stable accuracy and fluctuations in validation loss in later epochs. In the first epoch, the model achieved a training accuracy of 82.2% and a validation accuracy of 88.4%, with corresponding losses of 0.709 and 0.398. By the fifth epoch, the training accuracy had improved significantly to 99.0%, and the validation accuracy reached 95.3%, accompanied by a minimal validation loss of 0.079. Peak performance was observed during the 9th and 10th epochs, where the model achieved a validation accuracy of 100% with reduced validation losses of 0.009 and 0.004, respectively, while maintaining a training accuracy of over 99%.

```
Starting training for ReXNet-150...
1-epoch train loss          -> 0.709
1-epoch train accuracy      -> 0.822
1-epoch validation loss     -> 0.398
1-epoch validation accuracy -> 0.884
2-epoch train loss          -> 0.121
2-epoch train accuracy      -> 0.970
2-epoch validation loss     -> 0.180
2-epoch validation accuracy -> 0.953
3-epoch train loss          -> 0.055
3-epoch train accuracy      -> 0.983
3-epoch validation loss     -> 0.132
3-epoch validation accuracy -> 0.953
16-epoch train loss          -> 0.051
16-epoch train accuracy      -> 0.988
16-epoch validation loss     -> 0.104
16-epoch validation accuracy -> 0.977
17-epoch train loss          -> 0.086
17-epoch train accuracy      -> 0.980
17-epoch validation loss     -> 0.130
17-epoch validation accuracy -> 0.977
Loss value did not decrease for 4 epochs
18-epoch train loss          -> 0.017
18-epoch train accuracy      -> 0.995
18-epoch validation loss     -> 0.011
18-epoch validation accuracy -> 0.988
Loss value did not decrease for 5 epochs
Stop training since loss value did not decrease for 5 epochs.
Training complete for ReXNet-150!
```

Figure 6. The Training Process Result of RexNet-150

Despite achieving excellent accuracy, validation loss began fluctuating after the 11th epoch, suggesting diminishing returns in further training. Early stopping was applied after the 18th epoch when the validation loss did not improve for five consecutive epochs, ensuring the model avoided overfitting. The final results underscore ReXNet-150's ability to achieve high accuracy (100% validation accuracy) and low loss (0.004), demonstrating its effectiveness and robustness as a lightweight model suitable for deployment in real-world scenarios. This performance highlights its potential for applications in resource-constrained environments, such as mobile platforms.

The training processes for EfficientNet and Conventional CNN (ResNet-18) demonstrate distinct performance trends, highlighting their capabilities in apple leaf disease detection. EfficientNet, known for its optimized scaling architecture, began with a training accuracy of 79.9% and a validation accuracy of 91.9%, accompanied by a validation loss of 0.174 during the first epoch. By the fourth epoch, EfficientNet improved to a training accuracy of 98.2% and a validation accuracy of 97.7%, with a reduced validation loss of 0.076. Despite these gains, validation loss began to fluctuate after the fifth epoch, while validation accuracy plateaued. The highest validation accuracy of 98.8% was achieved in the 14th epoch, but training was halted after the 14th epoch using early stopping, as validation loss failed to decrease consistently for five consecutive epochs. This training strategy demonstrated EfficientNet's ability to achieve high accuracy, though fluctuations in loss suggested sensitivity to validation data variability.

In comparison, the Conventional CNN (ResNet-18) exhibited rapid and stable performance improvements, starting with a training accuracy of 90.0% and a validation accuracy of 97.7% in the first epoch. By the sixth epoch, it reached peak validation accuracy of 100%, with a validation loss of 0.017 and a training accuracy of 98.8%. Unlike EfficientNet, ResNet-18 maintained more consistent validation loss and accuracy throughout the training process, though some minor fluctuations occurred after the seventh epoch. Early stopping was applied after the 13th epoch when validation loss ceased to improve for five consecutive epochs. Overall, both models achieved competitive accuracy, with ResNet-18 showing greater stability, while EfficientNet demonstrated high initial performance gains. These results provide valuable insights into the comparative strengths of these architectures in achieving robust and efficient disease classification.

 ISSN 2723-6471

4.5. Model Evaluation

Figure 7's confusion matrix provides an in-depth analysis of the model's classification performance, highlighting its precision and reliability in detecting apple leaf conditions. The model successfully classified 24 instances as healthy, 27 as rust, and 35 as scab, demonstrating a high level of accuracy in distinguishing both healthy and diseased leaves. However, one misclassification was observed, where a healthy leaf was incorrectly labeled as scab. This error suggests that while the model performs exceptionally well in differentiating between diseased categories such as rust and scab, it occasionally misidentifies healthy leaves, potentially due to overlapping visual characteristics or environmental factors, such as lighting variations. This detailed evaluation underscores the model's robust classification capability while identifying areas for potential refinement to further minimize errors.

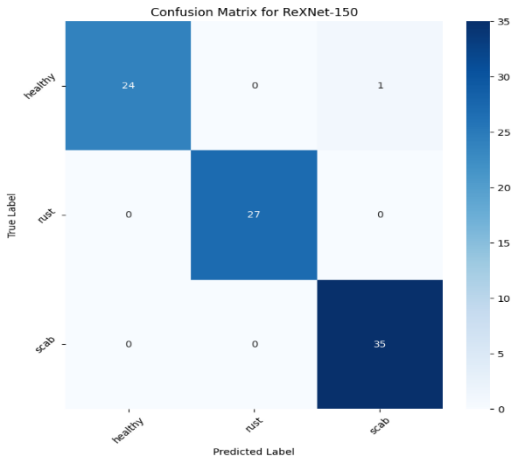

Figure 7. The Confusion Matrix for RexNet

The confusion matrix highlights the model's strong performance in identifying diseased categories, particularly with zero misclassifications for "rust" and "scab." However, future work should aim to enhance the differentiation of healthy leaves from diseased ones to minimize the likelihood of false positives, thereby ensuring more accurate and actionable recommendations for disease management in real-world applications.

Based on the table 1, ReXNet-150 outperforms the other models in this task due to its lower misclassification rates and consistent high performance in identifying all three categories ("Healthy," "Rust," and "Scab"). This makes ReXNet-150 the most reliable and robust model for deployment in real-world applications requiring high accuracy in disease detection.

Table 1. The Comparison of Confusion Matrix

Model	True Positive (Healthy)	True Positive (Rust)	True Positive (Scab)	Misclassified (Healthy as Rust/Scab)	Misclassified (Rust as Healthy/Scab)	Misclassified (Scab as Healthy/Rust)
ReXNet-150	24	27	35	1	0	0
EfficientNet	24	26	34	1	1	1
Conv CNN	25	26	34	0	1	1

Figure 8 depicts the training and validation loss for ReXNet-150 over 18 epochs, showcasing the model's learning progression and generalization capability. During the initial epochs, there is a sharp decline in both training and validation loss, indicating rapid learning and convergence. For instance, the validation loss drops significantly from approximately 0.4 in the first epoch to below 0.1 by the fourth epoch. This reduction demonstrates that the model effectively minimizes errors on both training and validation datasets early in the process.

From the 5th epoch onward, the training loss remains consistently low, reflecting the model's capacity to capture patterns in the training data. However, slight fluctuations in validation loss are observed from epochs 6 to 16, which

 ISSN 2723-6471

could indicate minor overfitting tendencies or sensitivity to specific validation samples. By the 18th epoch, both training and validation losses are minimal and well-aligned, suggesting robust performance and a well-generalized model. This trend validates the use of early stopping after the 18th epoch, preventing further overfitting and ensuring optimal model performance.

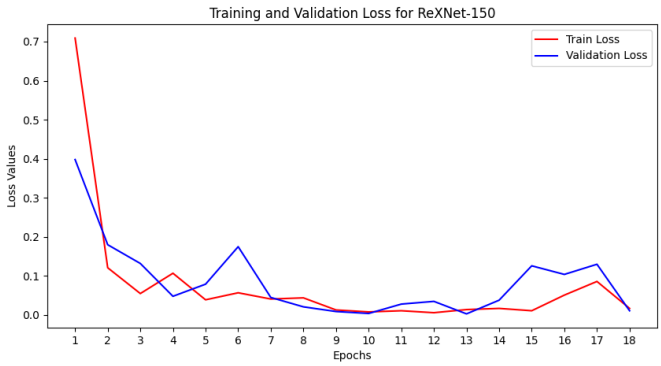

Figure 8. Training and Validation Loss

Figure 9 illustrates the training and validation accuracy for ReXNet-150 over 18 epochs, providing insight into the model's performance and generalization ability. Initially, both training and validation accuracies show rapid improvement, with training accuracy increasing from approximately 82.2% in the first epoch to nearly 99.0% by the fifth epoch. Similarly, validation accuracy shows a consistent rise, reaching around 95.3% by the fifth epoch.

From the sixth epoch onward, both training and validation accuracies stabilize, with the training accuracy nearing 100% by the 10th epoch. The validation accuracy achieves a perfect score of 100% during epochs 9 and 10, demonstrating the model's ability to generalize effectively. Minor fluctuations in validation accuracy between epochs 11 and 16 suggest sensitivity to validation data variability, but the overall trend remains high. By the 18th epoch, both training and validation accuracies converge at nearly 99%, indicating that the model maintains robust performance while avoiding overfitting. This steady alignment of accuracy metrics highlights the effectiveness of the training strategy and early stopping criteria in ensuring optimal model performance.

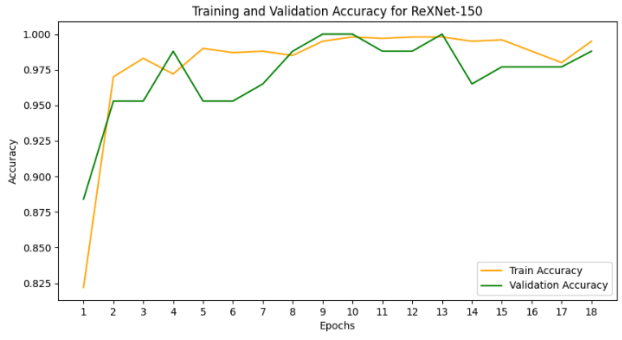

Figure 9. Training and Validation Accuracy

The ReXNet-150 model shown robust efficacy in categorizing apple leaf illnesses into three classifications: healthy, rust, and scab. Following the training of the model for 20 epochs, we assessed its performance utilizing various critical measures. The classification accuracy on the test set was 98.9%, indicating the model's proficiency in generalizing to novel data.

The performance metrics are detailed as follows: F1-Score: 0.989, Precision: 0.989 and Recall: 0.988. These scores indicate a well-balanced model with high precision and recall, suggesting that the model makes few false positive and false negative errors. This is crucial in the context of disease detection, where misclassifying a diseased leaf as healthy could lead to crop loss. In particular, the model exhibited the highest precision in identifying the "rust" and "scab"

 ISSN 2723-6471

categories, while slightly lower recall for the "healthy" class indicates some tendency to misclassify healthy leaves as diseased.

Based on table 2, ReXNet-150 is the most effective model for this task, followed by the Conventional CNN and then EfficientNet. This comparative analysis reinforces the robustness of ReXNet-150 for applications requiring high precision and recall, such as apple leaf disease detection.

Table 2. The Comparison of Metric

Model	Precision	Recall	F1-score
ReXNet-150	0.989	0.989	0.988
EfficientNet	0.966	0.966	0.966
Conventional CNN	0.979	0.977	0.977

4.6. Android Application Development for Leaf Disease Detection

Figure 10 depicts a mobile interface of an application for scanning apple leaves to predict their health condition. The user interface (UI) presents a preview of the uploaded or captured image under "Pratinjau Gambar" (Image Preview), allowing users to either choose a photo from their device or take a photo directly with the "Ambil Foto" (Take Photo) button. After the image is uploaded, users can initiate the scan by clicking the "Scan Sekarang" (Scan Now) button. The result of the prediction is displayed in the "Hasil Prediksi" (Prediction Result) section, where, in this example, the leaf has been classified as "healthy." This interface demonstrates a straightforward design that supports easy navigation for users, making it suitable for field use in agriculture, where quick and accurate predictions of leaf health are essential. The application integrates the machine learning model into a user-friendly environment, making it accessible to non-experts for disease detection in real-time. In the context of the Android application for apple leaf disease detection, an automatic resizing feature can be implemented. This involves applying a resizing function to transform the images captured or uploaded by users into the standardized 224x224 pixel size while maintaining their quality.

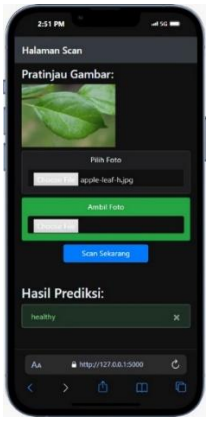

Figure 10. Interface of Application for Scanning Apple Leaves

This research would benefit from additional insights into the specific challenges encountered during the model integration into the Android app, as well as an analysis of potential differences between the real-world dataset and the training data. Integrating a machine learning model into a mobile application often involves technical challenges, such as optimizing model size for limited device storage, managing memory usage during inference, and ensuring efficient performance under varying hardware specifications. Furthermore, real-world datasets may differ significantly from training data in terms of environmental conditions, lighting variations, and image quality, which can impact model accuracy and robustness. A discussion of these challenges and potential mitigation strategies would provide valuable guidance for practitioners seeking to deploy similar models in mobile applications.

5. Conclusion

This study provides a comprehensive evaluation of three deep learning models—ReXNet-150, EfficientNet-B0, and Conventional CNN (ResNet-18)—to address the critical challenge of apple leaf disease detection. Among the evaluated models, ReXNet-150 emerged as the most effective, achieving the highest metrics with a precision of 98.9%, recall of 98.9%, and F1-score of 98.8%. The confusion matrix analysis for ReXNet-150 also revealed minimal misclassifications, with only one instance of a healthy leaf misclassified as diseased, demonstrating its superior ability to distinguish between healthy and diseased leaves. Additionally, the integration of the trained model into an Android application provides a practical, real-time solution for disease detection. The mobile application enables users, such as farmers and agricultural researchers, to easily capture or upload images of apple leaves and receive instant classification results, making it a user-friendly tool for fieldwork.

This study faces several limitations that warrant attention in future research. The mobile application lacks mechanisms to address misclassified errors, which may impact its reliability in critical use cases. The paper does not evaluate model performance under varying environmental conditions, such as changes in lighting or leaf orientation, nor does it assess app functionality in low-resource settings, limiting its generalizability and practicality. Additionally, the dataset of 1,730 images from Kaggle may not capture sufficient diversity in apple leaf diseases across regions. Furthermore, the research script does not include explicit hyperparameter tuning, potentially leaving room for optimization in model performance. Addressing these constraints could enhance the robustness and applicability of the proposed system.

Future research could focus on expanding the application of this model to other crops and plant diseases, potentially broadening its impact across the agricultural sector. Furthermore, integrating real-time video scanning capabilities could enhance the practicality of the tool, enabling continuous disease monitoring without the need for individual image capture. Cloud-based computing and AI-driven recommendations for disease treatment based on predictions are additional improvements that could increase the tool's utility, particularly for users in resource-constrained environments. The inclusion of features such as multi-language support and regional disease detection could further enhance the app's accessibility and effectiveness for a wider range of users.

Extending this system to accommodate larger datasets and multi-crop disease detection would require several strategic modifications. Firstly, expanding data storage and processing capabilities is essential to manage the increased volume and complexity inherent to broader datasets. Implementing scalable data pipelines, such as those provided by Apache Spark or TensorFlow Data, would enable efficient data loading and preprocessing in distributed environments, a critical feature for handling extensive agricultural data. Additionally, adjustments to the model architecture could be made to address the diversity in disease symptoms across various crops. Employing a multi-task learning approach or designing a model with separate branches dedicated to each crop would enhance performance by allowing the system to capture shared characteristics while also specializing in crop-specific features.

6. Declarations

6.1. Author Contributions

Conceptualization: C.B.S. ; Methodology: C.B.S.; Software: C.B.S.; Validation: C.B.S.; Formal Analysis: C.B.S.; Investigation: C.B.S.; Resources: C.B.S. , M.S., D.G.P. ; Data Curation: C.B.S., S.A., A.K; Writing – Original Draft Preparation: C.B.S.; Writing – Review and Editing: C.B.S., M.S., D.G.P., S.A., and A.K.; Visualization: C.B.S. All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

The authors received financial support for the research, authorship, and publication of this article from The Lembaga Penelitian dan Pengabdian kepada Masyarakat Universitas Pembangunan Jaya under contract number 014/PKS-LP2M/UPJ/09.24.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Gao, Z. Cao, W. Cai, G. Gong, G. Zhou, and L. Li, "Apple leaf disease identification in complex background based on bam-net", *Agronomy*, vol. 13, no. 5, pp. 1240-1253, 2023

[2] M. Alqahtani, A. Dutta, S. Almotairi, M. Ilayaraja, A. Albraikan, and F. Al‐Wesabi, "Sailfish optimizer with efficientnet model for apple leaf disease detection", *Computers Materials and Continua*, vol. 74, no. 1, pp. 217-233, 2023.

[3] Y. Wang, Y. Wang, and J. Zhao, "Mga-yolo: a lightweight one-stage network for apple leaf disease detection", Frontiers in Plant Science, vol. 13, Article 927424, no. 8, pp 1-12, 2022.

[4] J. Di and Q. Li, "A method of detecting apple leaf diseases based on improved convolutional neural network", *Plos One*, vol. 17, no. 2, Article 0262629, pp. 1-12, 2022.

[5] X. Liu, H. Liu, S. Yu, and Z. Zhong, "A control system for fine farming of apple trees", *Third International Conference on Image Processing and Intelligent Control*, vol 12782 , no. 5, pp. 230-236, 2023

[6] Y. Tian, G. Yang, Z. Wang, E. Li, & Z. Liang, "Detection of apple lesions in orchards based on deep learning methods of cyclegan and yolov3-dense", *Journal of Sensors*, vol. 2019, Article ID 7630926, no. 1, pp. 1-13, 2019.

[7] X. Li and S. Li, "Transformer help cnn see better: a lightweight hybrid apple disease identification model based on transformers", *Agriculture*, vol. 12, no. 6, Article. 884, pp. 1-16, 2022.

[8] K. Atanassov, S. Sotirov , and T. Pencheva, T. “Intuitionistic fuzzy deep neural network”. *Mathematics*, vol. 11, no. 3, pp. 716-728, 2023.

[9] M. Prayugo, "Human voice recognition system with backpropagation neural network method", Proceedings of the 12th International Conference on Green Technology, vol. 43, no. 5, pp. 432-442, 2023.

[10] L. Ahmed, "Comparison of artificial neural network and box- jenkins models to predict the number of patients with hypertension in kalar", *Ibn Al- Haitham Journal for Pure and Applied Science*, vol. 33, no. 4, pp. 110-121, 2020.

[11] R. Yamashita, M. Nishio, R. Gian, and K. Togashi, "Convolutional neural networks: an overview and application in radiology", *Insights Into Imaging*, vol. 9, no. 4, pp. 611-629, 2018.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks", *Communications of the Acm*, vol. 60, no. 6, pp. 84-90, 2017.

[13] R. Reiazi, R. Paydar, A. Ardakani, & M. Etedadialiabadi, "Mammography lesion detection using faster r-cnn detector", *Computer Science & Information Technology (CS & IT) Conference Proceedings*, vol. 8, no. 2, pp. 111-115, 2018.

[14] K. He, X. Zhang, S. Ren, & J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification", In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, vol. 2015, no. 12, pp. 1026-1034, 2015.

[15] A. Paszke, S. Gross, F. Massa, et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, no. 1, pp 8024-8035, 2019.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 2016, no. 6, pp. 770-778, 2016.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", *2009 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2009, no. 6 pp. 248-255, 2009.

 ISSN 2723-6471

[18] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107-115, 2021.

[19] L. Prechelt, "Early stopping—But when?" in *Neural Networks: Tricks of the Trade"*, Springer vol. 53, no. 1, pp. 53-67, 2012.

[20] P. Warden, , and D. Situnayake, "*TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers"*. O'Reilly Media, Inc, 2019