Sohar University Journal of Engineering and Information Technology Innovations (SUJEITI), ISSN: 3078-7513
Volume 2, Issue 2, pp 33-52, June 2026, DOI: 10.69983/SUJEITI/2253
Received: 24/05/2026 Revised: 08/06/2026 Accepted: 09/06/2026

Palm Leaf Disease Detection and Classification Using Deep Learning with Stable Diffusion-Based Data Augmentation

Maryam Abdulhelm Muter[1], Abbas H. Hassin Al-Asadi [2*]
[1] Computer Science and Information Technology College, University of Basrah, Basrah, Iraq (maryam.abdulhalim@uobasrah.edu.iq).
[2] Computer Science and Information Technology College, University of Basrah, Basrah, Iraq (abbas.hassin@uobasrah.edu.iq).

*Corresponding author email: (abbas.hassin@uobasrah.edu.iq).

Abstract

In this study, we are addressing the important issue of leaf disease detection and classification for date palm plants, which is a major crop for the economy of arid and semi-arid countries. The height of the trees and the commonality of disease symptoms often hamper the use of standard inspection methods, resulting in delayed or wrong identification. To address these problems, we present a deep learning-based architecture using DenseNet201 as a teacher model and EfficientNetV2-S as a student model in a knowledge distillation setting. The initial image data was provided by Kaggle. This data was diversified using Stable Diffusion inpainting. Manual and automatic mask construction approaches were used for accurate data augmentation. The full workflow includes systematic preprocessing, robust augmentation, and gradual fine-tuning. Experimental results reveal that the suggested system is quite accurate for the classification of four major types.

***Keywords**:* pest detection, stable diffusion models, classification, date palm, deep learning.

1. Introduction

Date palm trees are considered one of the most important plant resources in the world, especially in the regions of arid and semi-arid climates. These trees are important for rural and agricultural economies, providing a major food source, stable financial returns for farmers, and promoting growth in a number of related industries like food production and agricultural services. The production and quality of these trees are nonetheless under a substantial threat from several leaf-related diseases. It emphasizes timely and accurate identification to sustain a healthy agricultural environment and take suitable preventative measures (Sagheer et al., 2025; Muter and Al-Asadi, 2025).

Diseases of leaves are a severe danger to productivity and quality of output of date palm trees. The threat of these diseases is increased by the fact that the usual inspection procedures based only on naked-eye visual examination cannot give reliable and accurate results. This limitation is due to two major factors: 1. the great height of palm trees, which makes it difficult to closely examine the upper leaves; 2. the great similarity of the symptoms of different diseases, which makes it difficult to accurately identify them. Moreover, the time taken and amount of labor needed

in inspecting the palm leaves by hand through large agricultural areas may cause infections to pass undetected for a long time until any action is taken, which means that damage will have occurred by then. Thus, there is a pressing need for a better technology that will aid in diagnosing these diseases. (Abu-zanona et al., 2022).

2. The Impact of Dubas Bug on Date Palm Trees

Dubas Bug is the most destructive pest causing damage to date palm crops. This insect becomes active mostly during the spring and autumn periods when the nymphs hatch from eggs. During their feeding process, some sticky and sweet substance is released from insects and gets deposited onto different parts of the plant. This sticky substance when mixed with dust particles creates an ideal environment for black sooty molds. Black sooty molds prevent sunlight rays from penetrating into the leaves and also hinder the process of photosynthesis, thus affecting the overall well-being of the plant. Studies have revealed that the crop density accounts for 21% to 31% of the variance among the levels of insect population. It should be noted that this insect thrives well in traditional gardens when there are trees in high density. This is because of the high humidity created due to tree clustering. (Al Shidi et al., 2028).

Date palm leaves go through four successive stages when infected by the Dubas bug:

1. Stage One - Bug Only: This is the initial phase where the insect begins attacking the date palm leaves, marking the starting point of the infestation.
2. Stage Two—Mixed: This stage represents the most severe phase, where the insect combines with the honeydew it secretes, together causing the most significant damage to the date palm's leaves.
3. Stage Three - Honey: At this point, the insect dies off and disappears, leaving behind only the honeydew residue on the date palm leaves as a visible remnant of the infestation.
4. Stage Four—Healthy: This represents the natural and normal state of the date palm leaves, completely free from any signs of infection or disease symptoms (Al-Mahmood et al., 2023).

3. Related work

In the recent research, deep learning techniques are most commonly used for plant leaf disease and there many approaches use pre-trained models with complex optimization methods that show a high accuracy. For instance, some researchers employed the EfficientNet architecture within a transfer learning framework, achieving accuracy rates of 99.91% and 99.97% on the original and augmented datasets, respectively. Similarly, other researchers utilized MobileNetV2 and DenseNet201 models enhanced with the Whale Optimization Algorithm (WOA), obtaining an accuracy of 95.7% in citrus disease classification (Ahmed & Ahmed, 2023).

Due to the large improvement of computational ability, deep learning methods have been proved to be more efficient and have a strong ability to automatically extract features rather than manual feature engineering (2015; 2017). Here, convolutional neural networks (CNNs) have been used to classify 13 plant diseases with 96.3% accuracy (2016). Models like AlexNet and GoogLeNet have also achieved amazing accuracy of 99.35% on the PlantVillage dataset (2016). In addition, the DenseNet architecture has been proven to perform better, with an accuracy of 99.75% and fewer parameters, compared to other models (2019) (Atila et al., 2021).

Limited studies have explored the application of deep learning for detecting and classifying date palm diseases. Some researchers have demonstrated the feasibility of using convolutional neural networks (CNNs) to classify specific date palm diseases such as blight spots and leaf spot infections, achieving an accuracy of 97.9% (Alaa et al., 2020). In another study, the potential of deep learning was investigated for diagnosing Sudden Decline Syndrome in date palms, where the proposed approach included image preprocessing, feature extraction from color and texture information, and a multi-stage CNN-based classification framework (Hessane et al., 2023).

4. Knowledge Distillation

Knowledge Distillation (KD) is a model compression method in deep learning that seeks to create a smaller model with lower computational complexity and space requirements while maintaining most of the original model's performance. that aims to reduce the computational complexity and storage requirements of a large model without significantly sacrificing its performance. This process is called “knowledge transfer” from a large and complex model, called the “Teacher Model,” to a smaller model, called the “Student Model.” The student model is trained to mimic the behaviour and outputs of the teacher model in order to perform efficiently with less computational resources.

With the fast development of deep learning models and their applications in computer vision and natural language processing, the importance of knowledge distillation has been growing. These models are accurate but are often memory intensive and computationally demanding, thus limiting their use in resource-constrained devices such as smartphones, surveillance systems and autonomous vehicles.

Knowledge distillation has three major elements: the transferred knowledge (knowledge), the distillation process (distillation algorithm), and the teacher-student structure (teacher-student architecture). This helps the student model learn important representations learned by the teacher model. This technique helps in decreasing the model size and execution time while keeping good performance as seen in Figure 1 (Gou et al., 2021).

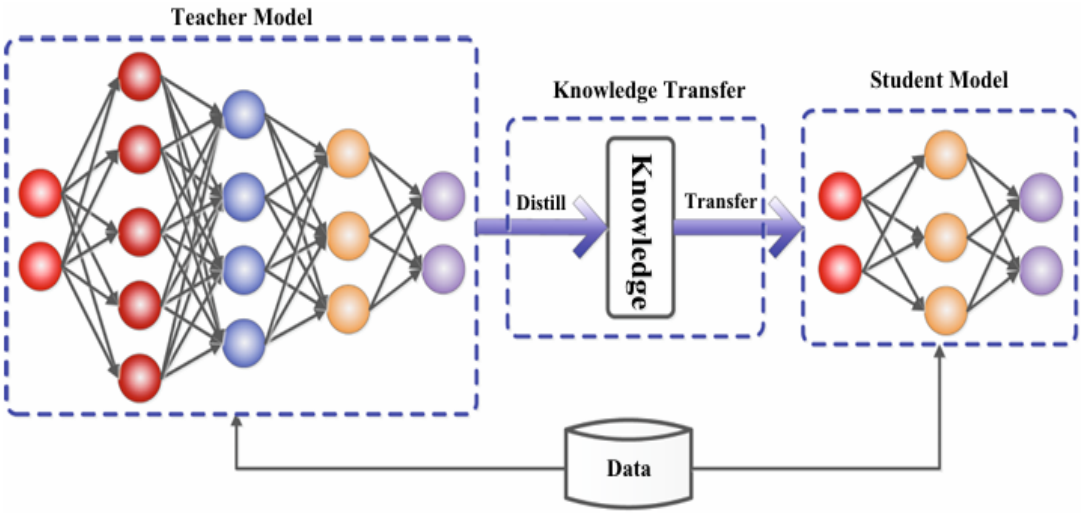


Figure 1: Knowledge Distillation (Gou et al., 2021).

5. Densely Connected Convolutional Network

DenseNet is a deep learning architecture in which all of the previous layers are directly connected. As a result of this dense connectivity, feature reuse is enhanced as is the flow of information and gradients that leads to greater stability while training the network and better performance. In this type of architecture, each layer is fed by all the previous layers, and produces its own feature maps to pass to the next layer. This makes the network more powerful in terms of representation and decreases redundant feature learning. Typically DenseNet uses two layers of convolution, one 1 × 1 and the other 3 × 3, to achieve efficient compression and extraction of features. This results in high accuracy and lower number of parameters as compared to models such as VGG and ResNet. To further boost the efficiency, DenseDSC and Dense2Net were introduced, bringing forth depthwise separable and group convolutions without compromising dense connections and performance on benchmarks such as CIFAR and ImageNet.

The 1 × 1 convolution layer helps to decrease the total number of maps that are processed before reaching the 3 × 3 layer, hence lowering computational costs through compression and keeping only the significant features. Finally, the 3 × 3 layer identifies the spatial features in the compressed maps. (Li et al., 2020).

6. EfficientNet

As of 2012, deep learning models trained on ImageNet have made substantial advances in accuracy and complexity of architecture, but tend to be very compute-intensive. EfficientNet is a family of state-of-the-art Convolutional Neural Networks (CNNs), which are able to attain a performance of about 84.4% on ImageNet with only about 66 million parameters

EfficientNet has eight variants (B0–B7), and the baseline feature of EfficientNet is that as model size increases, accuracy improves while the increase in the number of parameters is not commensurate. The model replaces ReLU with the Swish activation function, which expands optimization and learning performance (Ali et al., 2025).

Instead of applying different scaling coefficients for network depth, width and input resolution like most scaling strategies (such as MSD), EfficientNet implements compound scaling, with which you uniformly scale the depth, width and input resolution of the network. This is achieved by identifying, at a fixed computational budget, the ideal relations between these dimensions and scaling coefficients to scale the baseline model accordingly.

The EfficientNet neural architecture is mainly based on the MBConv (Mobile Inverted Bottleneck Convolution) Modified from MobileNetV2. By using an expand-compress structure with shortcut connections between bottleneck layers, it also makes the more model efficient by using depthwise separable convolutions which reduce computational complexity at least k^2 compared to standard convolutions, where k is the kernel size (Abdulkareem & Abdullah, 2025).

7. Proposed Framework

One of the most important characteristics in applications for smart agriculture and crop health monitoring systems is the accurate identification of the diseases occurring on plants, particularly palm leaves, which directly affects the reliability of the diagnosis of the disease and estimation of plant health. A decrease in classification accuracy may result in wrong diagnoses vs. treatment decisions that are not suitable, thereby leading to potential losses in both agricultural production and quality of crops. This is a clear need that has to be addressed by designing models that can perform with high accuracy, yet well generalize in the presence of new and unseen data. This section provides a brief overview of a proposed framework, that is, stages added on top of each other, that are complementary and interdependent. It starts with collecting data from Kaggle and goes through a generation and augmentation image step using stable diffusion. In addition, the preprocessing is done to enhance the quality of the input data and to make it suitable for training, as depicted in Figure 2.

8. Data Collection and Preprocessing Stage

Data Collection and Preprocessing: This is an important step in the development of an efficient machine learning system because all further steps of analysis and learning depend on it. The key factors that have the largest impact on the accuracy of the performance are the quality of the plant leaf images, the level of imbalance between classes and the range of lighting conditions, viewing angle and background. These are all aspects which are directly related to the usefulness of a model's learning of a representation, its ability to generalize across unseen datasets, and its ability to provide reliable predictions. In this context, a structured and systematic approach was formulated to the data collection process from reliable sources. We created a dataset from real-world sources in this case it is Kaggle site, which will be the main one for this study. After that, a few more samples were generated using this dataset with the Stable Diffusion model. Then, this set of samples was used as a "seed" to create additional samples with the Stable Diffusion model. In addition, the data were separated into a training and a test set to gain an objective assessment of performance and increase reproducibility. Simultaneously, a preprocessing step was performed to normalize the dataset and improve the quality of images used as input for the models. It included classifying the images into folders based on classes and then creating a folder structure to save all processed data.

The images were read and loaded via the OpenCV library, but with validations to avoid corrupted data. Then, all images were rescaled to a common size of 512×512 pixels with an interpolation method that maintains details in pictures. This decrease in variability across the samples helps improve consistency within the data, thus making it suitable for use in a deep learning application. The model can either generate images using Stable Diffusion or perform classification tasks. Ultimately, this process improves the model as it tends to achieve a more stable performance with more accurate results.

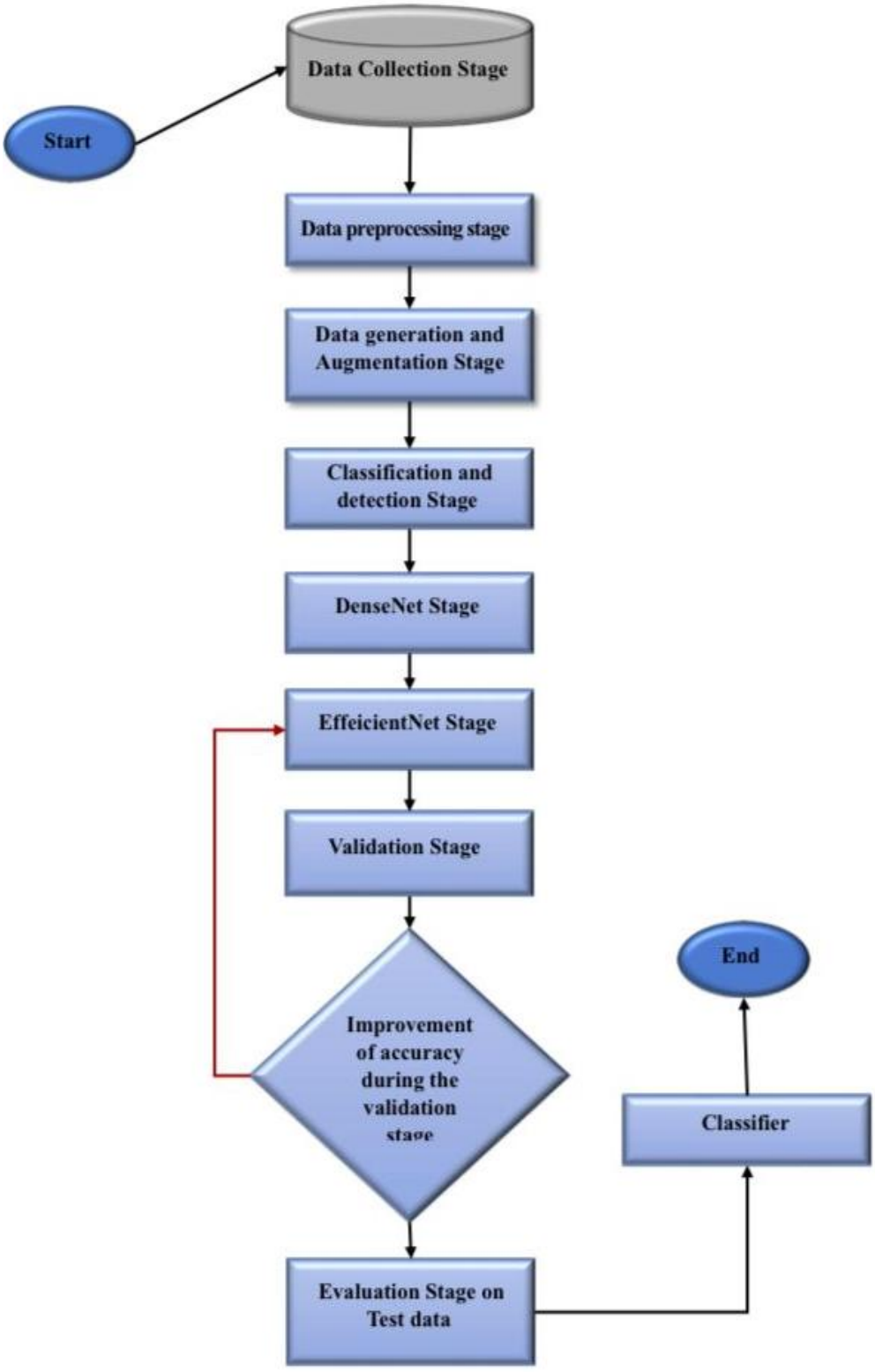


Figure 2: Proposed framework.

9. Data Augmentation Stage

The process of generating palm leaf images is carried out through three main stages as follows:

9.1 Manual Mask (Reference Mask)

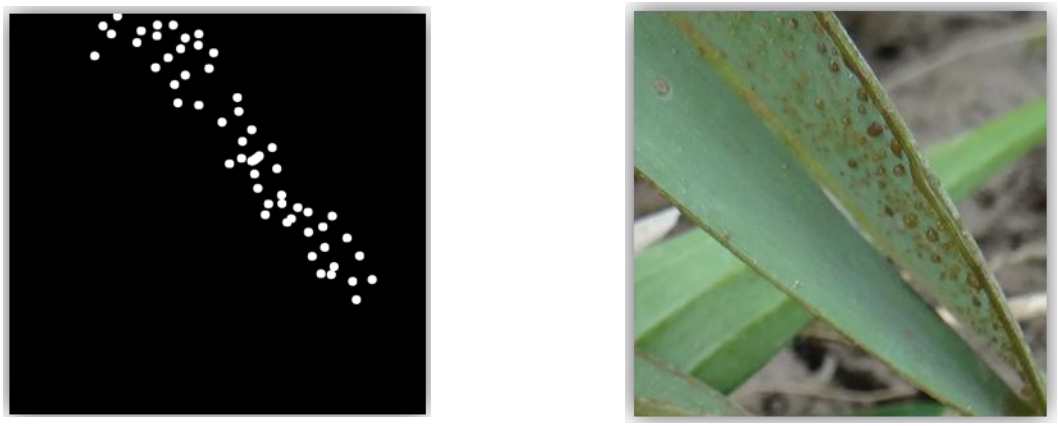
The important regions within the palm leaf image are manually constructed in a reference mask to identify areas in the leaf with disease and/or areas of interest and to accurately separate these areas from the background. This step helps to improve data quality and to minimize visual noise in subsequent processing steps.

This process was applied to these two classes, Bug and Honey, because of the presence of color overlaps and background elements that might impact the accuracy of feature extraction and performance of the identification of infection related features.

Manual masks were not generated for the Mixed and Healthy classes, in contrast. This is because images in these categories showed more distinct visual features, such as the clear visibility of infection symptoms (or the traces of honeydew) and the well-defined boundaries of the images, as well as enough contrast between the image of the leaf and the background. Hence, in this instance manual isolation was deemed not necessary.

The proposed method gives a practical example of generating manual masks of the Honeydew class using programming tools based on libraries of OpenCV and NumPy. The process begins with loading the palm leaf image and checking the validity of the input data. Then an empty mask is created with the same size as the image in RGBA format where the first three channels (RGB) are used for color information and the fourth channel (Alpha) is used for transparency. Values (0,0,0,0) represent the background and values (255,255,255,255) are manually marked to show the infected regions.

Once the annotation process is done, the mask is saved in PNG format, which preserves the RGBA structure. It is then used in image generation techniques such as Stable Diffusion Inpainting to generate or improve new images based on the selected regions, which helps to improve dataset diversity and support deep learning models in classification, as shown in Figure 3.



Manual Mask Honey Image Original

Figure 3: Manual mask and image original.

Then, an interactive tool is used to manually create a mask of the Bug class to precisely detect the insect regions before the next processing steps. First, the image is loaded and checked to see if it is a valid one. Then a single-channel (grayscale) mask is created with a value of (0) being the background and (255) being the selected regions.

Interactive drawing with a fixed brush size to control the precision of the annotation. The mask is updated in real time, and a colored overlay is added on the original image to highlight the selected areas, allowing for a precise separation of the insect areas from the background.

After the annotation process, the mask is stored as a PNG image file to be used later in image generation techniques such as Stable Diffusion inpainting. This enables the generation and/or enhancement of new images from selected regions, leading to higher data quality for deep learning models, as shown in Figure 4.

Manual Mask

Image Original

Figure 4: Manual mask and original image (Bug).

9.2 Automatic Masks (Auto Mask)

The masks are automatically generated based on the manual mask. The manual annotations serve as a guide to help create the masks automatically, without needing any direct input from the user.

These masks are used as a reference to extract the color characteristics of the infected areas and generalize them over the rest of the images in the dataset.

This step uses color segmentation to automatically find the infected areas through the analysis of reference images with a pre-defined mask of the infection's position. The color properties of the infected pixels are extracted and they are used to find similar patterns in other images.

We first define the data paths, meaning the reference image, the manual mask, the input image folder, and the output folder. If the output folder does not exist, it is created automatically.

The reference image is converted to the HSV color space for separating the color information from the illumination information. The manual mask is applied to extract only the infected pixels, and the minimum and maximum values for the H, S, and V channels are calculated. The color range so obtained is then slightly expanded to account for changes in lighting conditions between images, but still within valid channel limits.

The same thing is done in the LAB color space, which is more robust to changes in lighting. The image is converted to LAB format, and infected pixels are extracted according to the manual mask. Then the minimum and maximum values for the L, A, and B channels are calculated. While the range of colors gets slightly increased, the values are maintained in a correct range.

With this process, we will be able to increase the precision in the identification of infection in different images by producing accurate masks for the images..

9.3 Stable Diffusion-based Image Generation

The automatically generated mask is used to create new images or reconstruct parts of images of palm leaves using a Stable Diffusion model. The model generates realistic visual content in the masked regions to increase the data diversity and improve the dataset quality to better enable deep learning model training.

Following the division of the images into training, validation, and test sets, data generation was performed through inpainting using the Stable Diffusion algorithm to expand and diversify the dataset before performing training with deep learning algorithms. The size of the dataset increased from 3,000 images to 12,600 images distributed across four classes.

For the original Bug class dataset, there were 600 images in the dataset. These were expanded to 3,000 images using five new images generated from the original image. This is done by feeding the original image along with a mask marking the affected areas. The model generates additional images only in the masked areas while retaining the unaffected parts.

OS library was employed for managing files, while image processing and neural network computations were done using PIL and PyTorch libraries respectively. Diffusers library was utilized for loading the generative model. The locations of training, validation, and testing sets as well as mask and output images directories were defined.

Once the model was loaded, we defined a text prompt to guide the generation process, as well as a negative prompt to avoid undesired outputs. The parameters for the key generation were also set up, such as the strength to control the extent of modification within the masked region, the guidance scale to control how strictly the model follows the text prompt, and the image size that was set to 512 × 512 pixels.

Finally, five images were generated for each original image, followed by applying several transformations to the generated outputs to increase diversity, including:

- First version: generated directly without any additional transformations.
- Second version: generated and then rotated.
- Third version: generated and then horizontally flipped.
- Fourth version: generated and then spatially translated.
- Fifth version: generated and then zoomed.

This procedure enhances dataset variability and improves the ability of deep learning models to learn and generalize effectively, as shown in Figure 5.

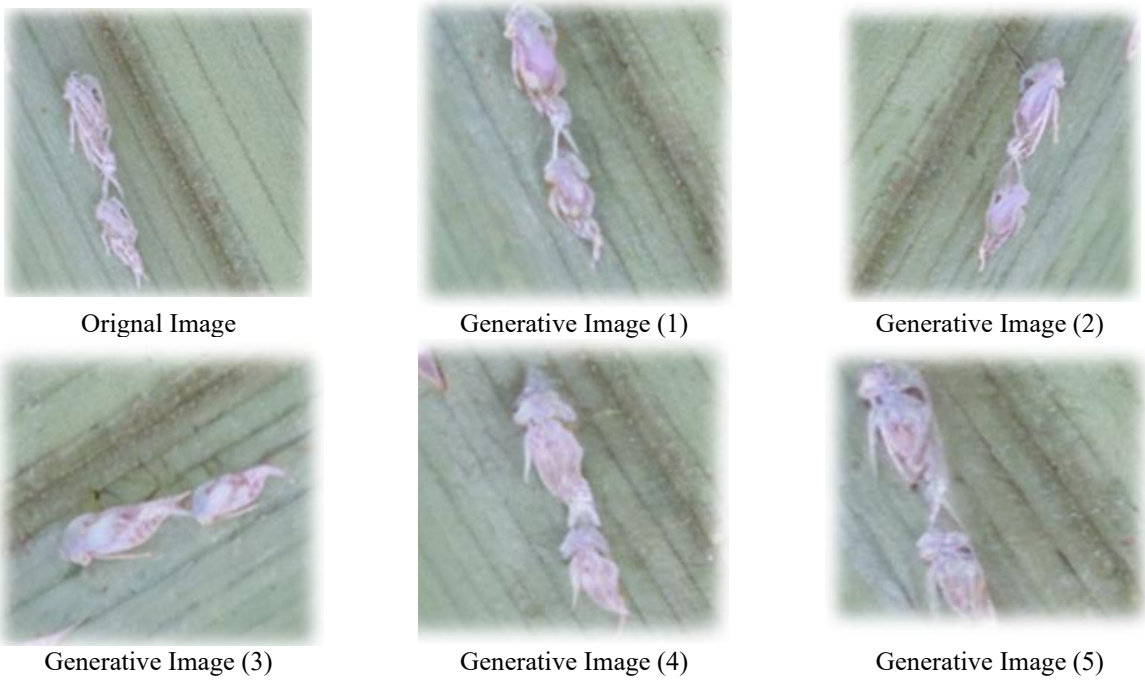


Figure 5: Generative image for Bug.

For the mixed (Dubas class with honey) class, an initial preprocessing step is done after dataset preparation to locate infected regions and automatically generate masks for use in the generation process. In this stage, images are converted to HSV color space to obtain better separation between color and intensity components. Then, a color threshold is applied to select the high-brightness regions corresponding to honeydew secretions, obtaining an initial mask that localizes the infected areas.

Morphological operations are used to improve the mask quality, such as opening for noise removal and dilation for improving the detected regions, providing better coverage of the infection area. The generated mask is then stored and used together with the original image in the image generation stage, specifying the region to be changed or regenerated.

Once the mask has been created, the model uses the original image along with the mask to create new images such that only the infection part gets edited and everything else remains the same .To further enrich the data diversity, several copies of each synthesized image are generated by applying transformations such as rotation, horizontal flipping, and spatial translation. This process augmented the size of the dataset from 800 images to 3200 images. All the transformations are applied after the generation process, as shown in Figure 6. Overall, this method improves data variability and the ability of deep learning models to learn diverse patterns and generalize well.

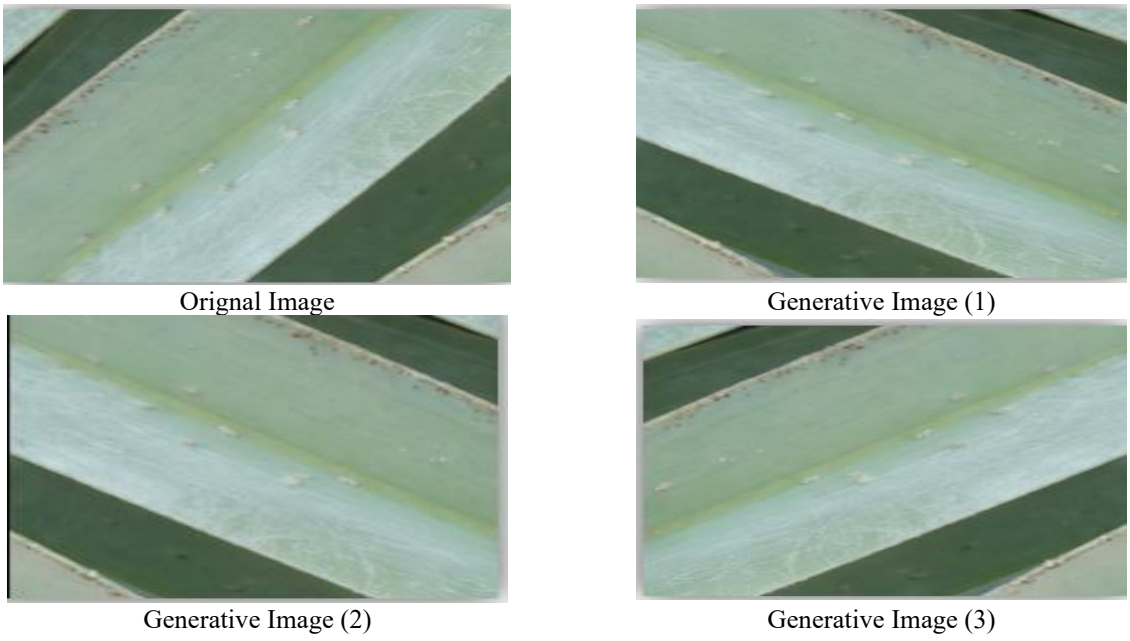


Figure 6: Generative mixed (Dubas Bug and Honey) class.

For the Honeydew class, we used pre-generated masks to guide the data generation process properly. Then, we used the masks to apply the Stable Diffusion Inpainting model on the dataset. The model receives the original image and its mask and generates new content only on the masked region, leaving the rest of the image intact.

To increase the diversity of data, four variants were generated for each original image, and then, some post-generation transformations were applied, such as 90-degree rotation, horizontal flipping, and spatial translation. The generation parameters were also tuned (e.g., strength) to control the degree of the modification in the masked area, and the guidance scale was used to control the degree to which the model follows the text prompt. Furthermore, 50 inference steps were used to generate high-quality images with details. This method increased the diversity of the dataset and improved the generalization capability of the deep learning models, as shown in Figure 7.

Generative Image (2)

Generative Image (3)

Figure 7: Generative Honey.

For the healthy class, since the images are healthy palm leaves with no infections or insects, no predefined masks are needed, as the entire image is suitable for generation. To allow the model to process the entire image without limiting the generation to a particular region, we therefore used a completely black mask (mask_black).

For each original image four versions were generated and then transformed after generation, including the original generated image, a 90-degree rotation, horizontal flipping, and spatial translation with a dark green background. We also set up the same generation parameters, including strength and guidance scale, along with 50 inference steps to generate high-quality and detailed images. The results of this method produced various healthy leaf samples, which help deep learning models to detect the healthy palm leaves from the infected ones more efficiently, as given in Figure 8.

Orignal Image

Generative Image (1)

Generative Image (2)

Generative Image (3)

Figure 8: Generation Healthy**.**

10. Disease Detection and Classification Stage

The process of disease detection and classification has two major stages. In the first stage, we train a deep learning model using the DenseNet201 architecture with a transfer learning approach to extract high-level visual features from the palm leaf images. This model is a teacher model that can learn the distinctive patterns of each disease category.

In the second stage, knowledge distillation is employed to transfer the knowledge learned by the teacher model to a more computationally efficient model, EfficientNetV2-S. Therefore, the Student Model is able to have a high classification accuracy with a lower computational complexity and inference time. Finally, the performance of the models is evaluated on the testing dataset using evaluation metrics including accuracy, confusion matrices, and ROC curves.

10.1 DenseNet Stage

In the first stage, the directory paths of the datasets used in the experiment are defined for the DenseNet201 model. Separate folders are allocated to training, validation, and testing datasets. This separation forms the training part of the data where you can fit a model on, and test its generalisation ability through unseen samples which we do through validation and testing.

Then, we specify the target classes of various patient conditions on the palm leaves, i.e., Bug, Mixed, Healthy and Honeydew. To achieve consistency throughout the labeling process, a mapping dictionary, class_map, is created between original folder names and predefined labels based on classification. Mapping your data helps to organize your dataset and remove inconsistencies that may arise due to different folder names.

Moreover, a custom function named relabel_dataset is used to relabel the dataset by reading each image's folder name and assigning it to the relevant target class based on the predefined labels. This step ensures label consistency and enhances the reliability of the training process.

Then a number of image preprocessing operations are performed. All images are resized to a fixed size of 299 × 299 pixels and converted to a tensor format to be used as inputs for deep learning models. In this stage, we do not use any data augmentation or normalization methods to keep the initial experimental setup simple and to focus on the core learning capability of the model.

And lastly, the datasets are loaded with ImageFolder function reading images directly from their directories while applying the defined transformations. Then, the relabel_dataset function is used to standardize class names and gives us three tidy datasets for training, validation, and test.

Now that, we have done the data preparation step, let us load the DenseNet201 model trained in dataset ImageNet using transfer learning. This method enables the model to utilize previously learned knowledge to extract important visual features from palm leaf images and reduce the training time and computation cost.

In the first stage of training, we freeze the feature extraction layers by setting requires_grad = False and only train the last classification layer. Furthermore, the CrossEntropyLoss function is used for the multi-class classification task, and the label smoothing of 0.1 is adopted to improve the generalization ability and avoid overfitting.

We only update the classification layer parameters using the AdamW optimizer for weight optimization. We set the learning rate to 1e-4, which is a good trade-off between the learning speed and the training stability. The weight decay is set to 1e-3 to regularize the model weights and minimize the risk of overfitting.

Additionally, the ReduceLROnPlateau learning rate scheduler was used to monitor the model performance on the validation set. The scheduler automatically decreased the learning rate by a factor of 0.5 when the validation loss stopped improving for a certain amount of time, which helped to improve convergence during training.

Training was run for 25 epochs with an early stopping criterion of 7 epochs. The training was stopped when the validation loss did not improve for several consecutive epochs, which reduced the possibility of over-fitting and saved training time.

Each era involved two primary steps. Firstly, there was a training step. During this stage, the training data were fed into the algorithm. Here, the loss values and accuracy values were computed as well, after which the backpropagation method was utilized to update the weights of the classification layer alone, while the feature extractor layers remained unchanged or frozen. Secondly, there was a validation step. This entailed evaluation of the algorithm using the validation data and without any weight updates. Validation loss and accuracy were computed to determine how well the model had been generalized.

In the subsequent training stage, the DenseNet201 algorithm was thoroughly fine-tuned in order to improve its discrimination capabilities for various classes of palm leaf diseases.

In the initial stage, all layers of the model template are frozen and prevent modifying the learned knowledge from ImageNet dataset. Only the deeper layers denseblock3, denseblock4 and norm5original were unfrozen (or requires_grad = True) This allows the model to use flexible high-level feature representations that are specific to the target task while preserving low-level features that remain stable and general.

During the training, we used CrossEntropyLoss with a label smoothing factor of 0.1, which is appropriate for the multi-class classification task. This technique helps to reduce overconfidence in predictions, increase generalization, and reduce overfitting.

For optimization, the AdamW optimizer was used to update the trainable parameters only with a very low learning rate of 5e-6 for stable updates on deep layers and a weight decay of 1e-4 to reduce overfitting further.

Furthermore, we used a learning rate scheduler with cosine annealing with warm restarts, which periodically reduces the learning rate according to a cosine decay with warm restarts. In this scheduler, T_0 defines the length of the initial cycle before the first restart, and T_mult is a multiplication factor that progressively increases the length of subsequent cycles, helping the model converge more effectively and avoid poor local minima during fine-tuning.

The training stage process was set to 30 epochs with an early stopping mechanism with a patience value of 7. We stop training when the validation loss does not improve for 7 epochs in a row, which reduces overfitting and increases computational efficiency. Furthermore, the best-performing model weights (best_model_wts), corresponding to the lowest validation loss, are saved and reloaded after training completion.

During each epoch, we have two main steps: training and validation. The input images go through the model to compute the predictions and loss values, which are then backpropagated updating just the trainable parameters: pre-selected deep layers and last classification layer. To prevent exploding gradients, we apply gradient clipping with max norm 2.0 to provide training stability. We calculate training accuracy in order to observe the learning.

In the validation phase, the model is tested on the validation dataset, and no weights are updated. For the assessment of generalization performance and the monitoring of possible overfitting during epochs, validation loss and accuracy are computed.

Once the training is complete, the final model is saved locally, and a backup copy is also stored on Google Drive for accessibility and data preservation.

In the last evaluation phase, the model is set to evaluation mode (model.eval()) and tested on the unseen test dataset. The overall accuracy is computed by comparing the predicted labels with the ground-truth labels, which provides an indication of the performance of the model on unseen data.

A comprehensive analysis is performed with a confusion matrix and a classification report also. Confusion matrix allows us to take a closer look at the true and false predictions across all classes, while classification report presents the precision, recall and F1-score of every class.

In general, the proposed pipeline is based on a two-stage fine-tuning approach on DenseNet201. The classification layer is first trained, followed by a deep fine-tuning of selected internal layers to obtain better results in the classification of palm leaf images into four categories: Bug, Mixed, Healthy, and Honey.

The model was trained on an augmented dataset of palm leaves, which has four classes: Bug, Dubas (Mixed), Healthy, and Honey, and training in a two-stage approach, first training the classifier and then deep fine-tuning.

The first step was to freeze the whole DenseNet201 model except the last classification layer. This allowed the model to learn the basic feature representations for each class without modifying the deep network structure. In this stage, the model reached a training accuracy of 86.83% and a validation accuracy of 85.48%, as shown in Table 1.

Table 1: Classifier performance on testing data.

	Training accuracy	Validation accuracy
Classifier only	86.83	85.48

In the second stage, a deep fine-tuning process was applied by unfreezing the denseblock3, denseblock4, and norm5 layers in addition to the classification layer, allowing the model to update weights and refine deep feature

representations. This has led to a significant improvement in all performance in both training and validation stages, as shown in Table 2.

Table 2: Fine-tuning performance.

Deep fine-tuning	Training accuracy	Validation accuracy
	99.66%	93.41

The accuracy reached by DenseNet201 was 94.96% after fine-tuning was done, which indicates that the model has been successful in differentiating between the four palm leaf types.

Table 3 provides the training and validation accuracy of the model after training had been completed, and it shows the improvement in performance due to deep fine-tuning. Table 3 only shows the accuracy of the training and validation datasets.

Table 3: Fine-tuning training and validation accuracy for each class.

Deep fine-tuning	Training accuracy	Validation accuracy
Bug class	100%	86.67%
Dubas's class	99.87%	90.00%
Honey class	100%	98.12%
Healthy class	100%	100%

As for the final assessment, testing of the model was performed using unseen test data. Table 4 shows the classification results of the model in more detail using additional metrics such as precision, recall, and F1-score. **Table 4:** Fine-tuning performance on testing data

Data on palm leaf diseases	Accuracy	precision	Recall	F1-score
Bug class	90.00%	0.94	0.90	0.92
Dubas's class	91.00%	0.92	0.91	0.91
Honey class	99.22%	0.97	0.99	0.98
Healthy class	99.22%	0.97	0.99	0.98

Additionally, as shown in Figure 9 below, the confusion matrix represents the accuracy and inaccuracy for both classifications (infection and non-infection), indicating the ability of the model to distinguish between the two

classifications.

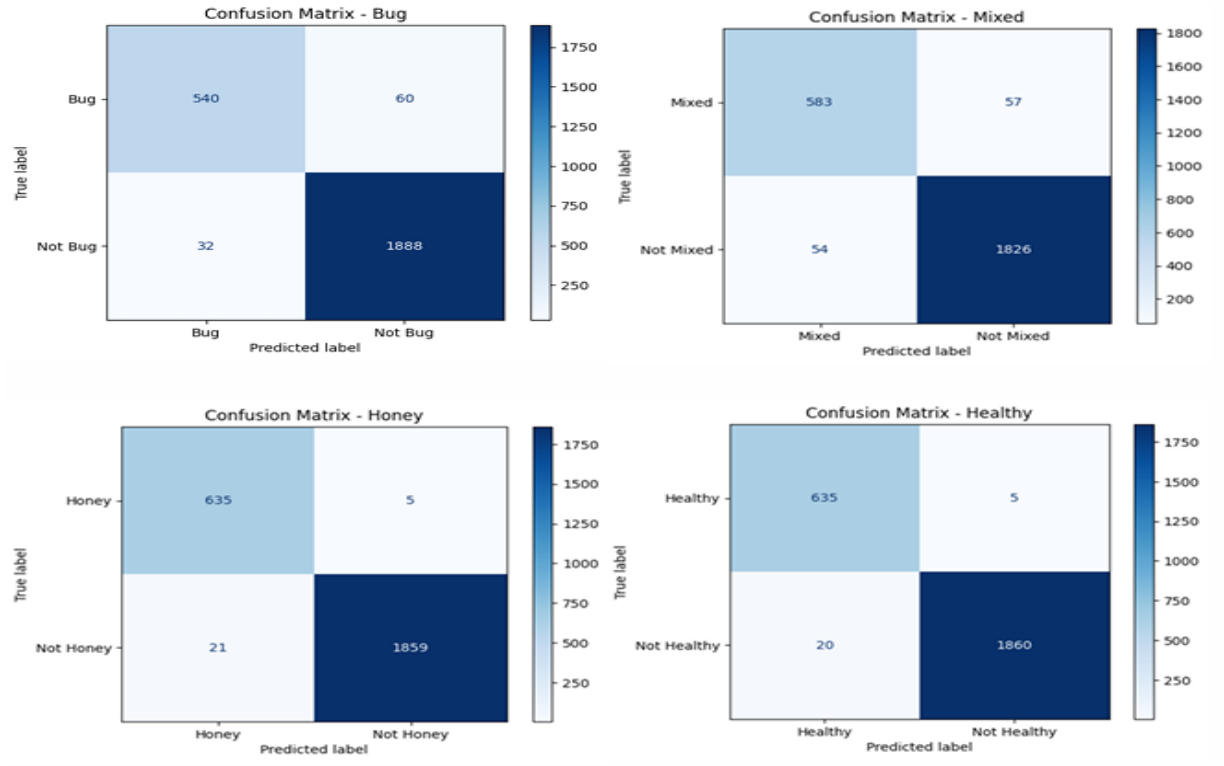


Figure 9: The confusion matrix for each class on testing data (DenseNet).

10.2 EfficientNet Stage

This stage is the second part of the project. It aims to apply the knowledge distillation technique using the pre-trained EfficientNetV2-S model. The objective is to transfer knowledge from a larger, more complex model to a more efficient architecture, thus reducing the computational cost and enhancing the overall performance.

We used Automatic Mixed Precision (AMP) to improve training efficiency, which allows some calculations to be done in FP16 precision instead of FP32, leading to memory savings and training speed-up. Moreover, GradScaler is used to stabilize the gradient update to avoid the gradient underflow or overflow problem.

Next, the paths of the training, validation, and testing datasets were defined while the target labels were unified into four main classes: Bug, Mixed, Healthy, and Honey. A dictionary class_map was also used to map the original folder names to the standardized classes so that label consistency was maintained during the training process.

During data preparation, several image augmentations and transformations were conducted to enhance the model's generalization ability and robustness to image variability. The input image size was fixed to 380×380 pixels to meet the requirements of the EfficientNetV2-S architecture.

The training transformations were random-size cropping, horizontal and vertical flipping, rotation, color adjustments, perspective distortion, and random erasing to simulate partial occlusions. The images were converted to tensors and normalized with the ImageNet mean and standard deviation values.

The validation and test datasets were resized and center-cropped only and converted to tensor and normalized, without any random augmentations, so as to ensure stable and accurate model evaluation.

To overcome the class imbalance problem, the WeightedRandomSampler was used to assign higher sampling weights to the underrepresented classes in the training dataset so that the model can learn all categories more uniformly. This was done by calculating the number of samples in each class and assigning weights inversely proportional to the class frequency.

Finally, a WeightedRandomSampler was created to use the computed class weights and total number of samples to ensure balanced sample selection during training.

The dataset was prepared, and data loaders were configured. The knowledge distillation approach was implemented by defining the teacher and student models and configuring the loss functions and optimization strategy used during training.

We trained the DenseNet201 model as the teacher model, which is the same model as the one trained using the two-stage fine-tuning approach before. The last classification layer was adapted to the number of classes by replacing the original classifier with a new classification head composed of a dropout layer at rate 0.4 followed by a linear layer with a number of outputs equal to the number of classes. Then we loaded the pre-trained weights from the saved checkpoint.

To ensure that the teacher model is only used as a knowledge source during distillation, the model was set to evaluation mode by model.eval(). This configuration also disables stochasticity in layers such as dropout and fixes batch norm statistics during execution. Moreover, all the parameters of the models were frozen so that the weights were not updated, and the teacher model was only used as a source of prediction knowledge for the student model.

The student model was built upon the EfficientNetV2-S architecture, initialized with ImageNet pre-trained weights to benefit from transfer learning. Its last classification layer was also modified accordingly to fit the number of target classes.

The student model aims to build a lighter and computationally efficient network with strong classification performance by learning from both the ground truth labels and the soft predictions given by the teacher model during the knowledge distillation process.

We used a custom loss function, distillation loss, which is a sum of cross-entropy loss and KL divergence. In this setting, the student model can learn from the true labels in the data but also from the soft predictions made by the teacher model. In this way it will not only learn the correct class but also how one class connects with another thus improving its whole understanding.

The approach is done on the raw model outputs (logits) before Softmax. These values are scaled with a temperature parameter, creating a softer and better-balanced distribution over the classes. This is helpful so the model can learn from the knowledge of the teacher more effectively. Additionally, a weighting factor (alpha = 0.6) is used to balance the two parts of the loss and lets it know that true labels are more important but still receive guidance from teacher. To optimize we employed AdamW with low learning rate and weight decay for stability and control of overfitting.

We also employed a cosine annealing scheduler to gradually decay the learning rate during training, which helps the model converge more smoothly.

Early stopping was used to make training more efficient. It automatically stops the training process if there is no improvement in the validation set for some number of epochs. The weights of the best models are saved, i.e., the final model is the one with the best validation performance and not over-trained results.

At each epoch of training the model, the current performance is compared with the best achieved until now. If an improvement is found, then the new best value is stored and the counter is reset. If there is no improvement, the counter increases until it reaches the predefined patience value, and then the training is stopped automatically. Once training is finished, load_best_weights can be used to retrieve the best model.

Then, a knowledge distillation stage is used to transfer the knowledge from the teacher model to the student model. At this point, the student learns from two sources, the outputs of the teacher model and the true labels of the dataset. This allows them to learn deeper relationships instead of just counting on the correct class. Training is also stopped when there are no improvements on the validation set (early stopping).

The student model is used to make predictions in the training phase. The teacher model runs in parallel and outputs some values but does not update its parameters. So the error is obtained from distillation loss, which learns from both dataset and teacher model.

We use the AdamW optimizer to update the model weights and Automatic Mixed Precision (AMP) to speed up the training and reduce the memory consumption.

The model is evaluated on the validation set at the end of each epoch to measure performance, and the learning rate is gradually reduced with a cosine annealing scheduler to improve training stability and convergence.

The model was continuously monitored on the validation set, and early stopping was applied. The best model weights were saved when there was an improvement in validation accuracy. During the knowledge distillation stage, if no improvement was observed in 20 consecutive epochs, the training process was stopped automatically and the best weights were restored. These weights were then used for the final model for the evaluation phase.

Post the knowledge distillation phase, a gradual unfreezing strategy was employed to further enhance the performance of the EfficientNet student model. The approach starts by freezing all layers of the network and then progressively unfreezing them, enabling the model to adapt to the dataset while retaining the knowledge acquired during distillation.

The unfreezing process was carried out in three stages:

- In the first stage, only the upper layers were trained (about 30% of the network).
- In the second stage, the training portion was increased to around 60%.
- In the final stage, the entire model was unfrozen (100%) for full fine-tuning.

Each stage was trained for a fixed number of epochs with a gradually decreasing learning rate to ensure stable and controlled updates.

We applied Cross-Entropy Loss with Label Smoothing (0.1) to reduce overconfidence and improve generalization. We used the AdamW optimizer for the update of only trainable parameters and a learning rate scheduling strategy to help the convergence be smoother.

In addition, Early Stopping (patience = 15) was used to monitor the validation performance and stop the training when no improvement was observed, keeping the best model weights throughout the process.

And in the end, the best weights from all phases would be restored and be used as a final model for test and evaluation.

The model was saved, and the test dataset is used for final evaluation. To further improve the robustness of results, we utilized Test Time Augmentation (TTA). It also produces three versions of the original test image: the image flipped along both the X and Y axes, as well as rotated by a small angle. Each version is input to the model independently, and the final prediction is an average of the Softmax probabilities. This results in producing stable and stronger outputs.

The model was evaluated using standard classification metrics after the final predictions were generated. Accuracy was measured by the overall proportion of correctly classified samples, and a classification report was produced, including precision, recall, and F1-score for each class.

In addition, a confusion matrix was used to show the correct and incorrect predictions for all classes to get better insights into the classification errors of the model.

To allow a more fine-grained assessment, ROC curves were computed for each class individually along with their respective AUC scores. AUC is the measure of the model's ability to distinguish one class from the other. Finally, a macro-AUC was computed using a One-vs-rest approach in order to get a general measure for the multi-class discrimination performance of the model.

To ensure full trainability of all layers of the EfficientNetV2-S student model, the first knowledge distillation stage was performed. This enabled the model to fully adapt to the data by learning from the ground truth labels (challenging labels) and the probabilistic outputs provided by the teacher model (soft targets).

Table 5 displays the results of this stage's performance.

Table 5: Knowledge Distillation Performance: Best Training and Validation Accuracy.

	Best training accuracy	Best validation accuracy
Knowledge distillation	88.92%	95.63%

In the second Progressive Fine-Tuning Stage, a progressive unfreezing strategy was applied to the EfficientNetV2-S student model over three consecutive steps. In each stage, the number of trainable layers was gradually increased,

allowing the model to adapt to the new data while preserving previously learned knowledge. The results showed a steady performance improvement, confirming the effectiveness of this approach.

The first stage was conducted by unfreezing about 30 percent of the model's parameters mostly in the later layers while the remaining network was frozen. This enabled the model to undergo initial adaptation to the training and validation sets, but still keep the features that had been learned through pre-training. These results are given in Table 6.

Table 6: Fine-tuning 30% performance, best training, and validation accuracy.

First Stage of Progressive Fine-Tuning (30%)	Best training accuracy	Best validation accuracy
	99.78%	96.03%

In the second stage, the percentage of trainable parameters was increased to 60%, which included most of the layers at the top of the model, with other layers being kept frozen. With this, the model was able to adapt even further to the data set and its learning capabilities were further improved. These findings have been provided in Table 7 below.

Table 7: 60% fine-tuning performance, best training, and validation accuracy.

Second Stage of Progressive Fine-Tuning (60%)	Best training accuracy	Best validation accuracy
	99.83%	96.03%

All the model parameters were then unfrozen, which permitted full weight updating during training in the last step. The model could thus adapt completely to the dataset and perform better. Results for the last stage are presented in Table 8.

Table 8: Fine-tuning 100% performance, best training and validation accuracy.

Third Stage of Progressive Fine-Tuning (100%)	Best training accuracy	Best validation accuracy
	99.92%	95.24%

Thirdly, the Test-Time Augmentation stage was performed in order to increase the stability and improve the accuracy of the results generated. The test-time augmentation method creates different variations of the image by simply flipping or rotating them by angles of +5° and –5° horizontally and vertically.

These images are then fed into the model, and the output values are averaged to determine the final result. It was observed that this process resulted in greater stability and accuracy, as can be seen in Table 9

Table 9: Fine-tuning performance accuracy for each class on testing data (EfficientNet).

Data on palm leaf diseases	Accuracy	Precision	Recall	F1-score
Bug class	97.86%	0.96	0.95	0.95
Dubas's class	97.46%	0.95	0.95	0.95
Honey class	99.84%	1.00	1.00	1.00
Healthy class	99.44%	0.98	0.99	0.99

The detailed performance for each class is presented in Table 9**,** while the corresponding visual results are illustrated in Figures 10 **and** 11.

The evaluation results show that the EfficientNet model, after applying the three main stages (knowledge distillation, fine-tuning, and test-time augmentation), achieved strong performance in the classification task.

The macro average values for precision, recall, and F1-score were approximately 0.97, indicating a well-balanced performance across all classes. Similarly, the weighted average scores were nearly the same, reflecting the model's stability and its ability to handle class distribution differences within the dataset.

For the ROC-AUC analysis, the model demonstrated excellent discriminative ability between classes. The AUC values were 0.9953 for Bug, 0.9946 for Mixed, 0.9990 for Healthy, and 1.0000 for Honey. The overall macro AUC reached 0.9972, showing very high consistency in distinguishing between different categories.

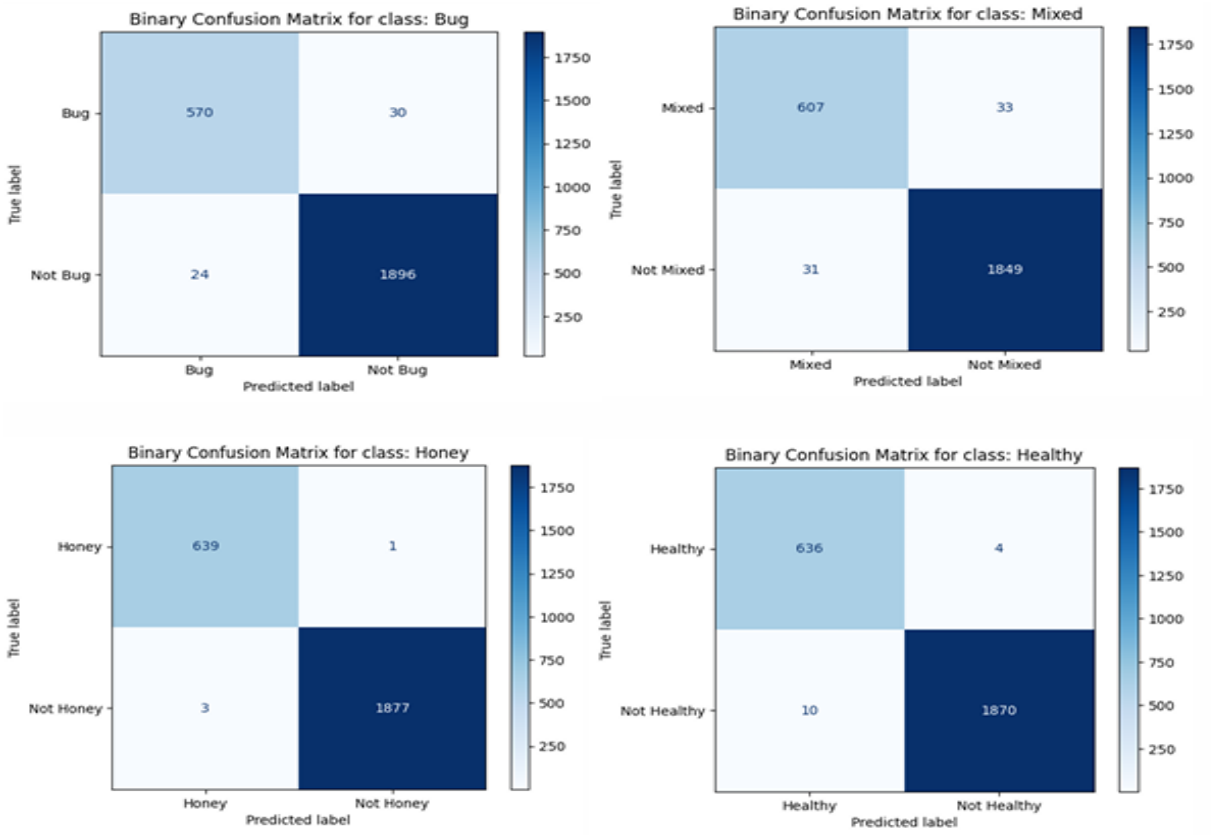


Figure 10: The confusion matrix for each class in the testing data of EfficientNet**.**

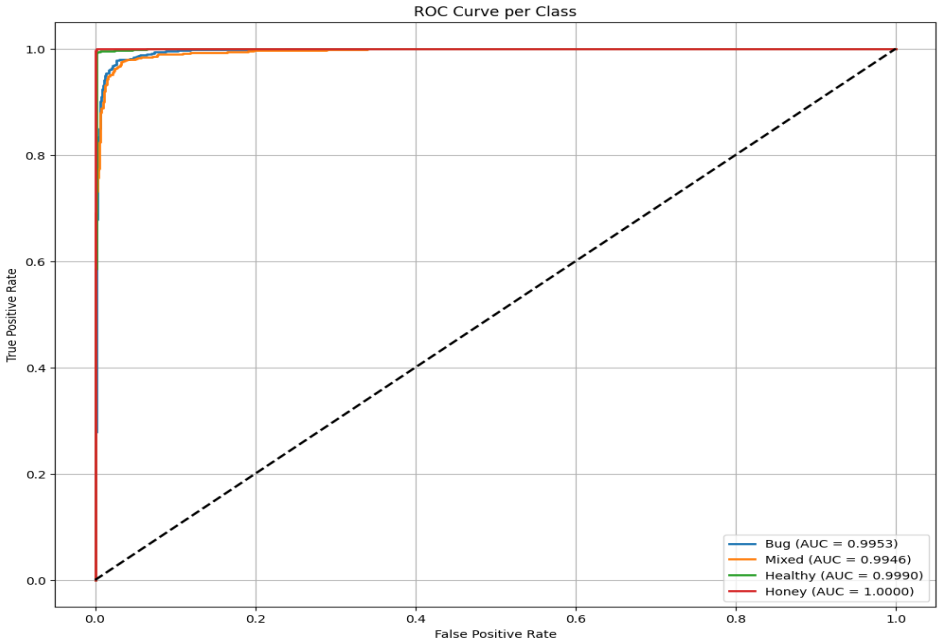


Figure 11: ROC Curve per Class Based on Testing Data.

11. Conclusion

The article introduces an efficient deep learning framework for identifying and classifying diseases on date palm leaves at early stages. By incorporating sophisticated deep neural networks including DenseNet201 and EfficientNetV2-S, and by using the concept of knowledge distillation, the presented framework provides highly accurate disease identification yet maintaining computational efficiency for practical applications. Manual and automated image data augmentation techniques with stable diffusion inpainting technique enhance significantly the quality and robustness of the datasets used. As a result, the multi-stage approach (transfer learning, fine-tuning, and test-time augmentation) shows strong performance with high macro-averaged precision, recall, F1 scores close to 0.97 and macro-AUC above 0.99. These findings imply that the presented framework will efficiently discriminate between healthy and diseased palm leaves, hence aiding in taking appropriate preventive measures. Overall, this paper contributes to the advancement of smart agriculture through introducing a solution to monitor diseases in valuable crops like date palm.

Acknowledgment

The research leading to these results has received funding for a research project from the xxxxxx, Research Grant Agreement No. GRANT AA E2 / 01010. The authors would like to acknowledge support from the xxxxxx.

Alternatively, the research that led to these results did not receive any grant funding.

Author contribution: All authors have contributed, read, and agreed to the published version of the manuscript results.

Conflict of interest: The authors declare no conflict of interest.

References

[1] Abdulkareem, M. H., & Abdullah, H. A. (2025, December). EfficientNet-B0-based image classification: architecture, training, evaluation, and visualization. In IET Conference Proceedings CP959 (Vol. 2025, No. 41, pp. 37-44). Stevenage, UK: The Institution of Engineering and Technology.. https://doi.org/10.1049/icp.2025.4377

[2] Abu-zanona, M., Elaiwat, S., Younis, S., Innab, N., & Kamruzzaman, M. M. (2022). Classification of palm tree diseases using a convolutional neural network. *International Journal of Advanced Computer Science and*

Applications, *13*(6). https://doi.org/10.14569/IJACSA.2022.0130670 (Note: DOI constructed from journal pattern; verify original)

[3] Ahmed, M., & Ahmed, A. (2023). Palm tree disease detection and classification using a residual network and transfer learning of Inception ResNet. *PLoS ONE*, *18*(3), e0282250. https://doi.org/10.1371/journal.pone.0282250

[4] Al Shidi, R. H., Kumar, L., Al-Khatri, S. A. H., Albahri, M. M., & Alaufi, M. S. (2018). Relationship of date palm tree density to Dubas bug *Ommatissus lybicus* infestation in Omani orchards. *Agriculture*, *8*(5), 64. https://doi.org/10.3390/agriculture8050064

[5] Alaa, H., Waleed, K., Samir, M., Tarek, M., Sobeah, H., & Salam, M. A. (2020). An intelligent approach for detecting palm tree diseases using image processing and machine learning. *International Journal of Advanced Computer Science and Applications*, *11*(7), 434–441. https://doi.org/10.14569/IJACSA.2020.0110757

[6] Ali, H., Shifa, N., Benlamri, R., Farooque, A. A., & Yaqub, R. (2025). A fine-tuned EfficientNet-B0 convolutional neural network for accurate and efficient classification of apple leaf diseases. Scientific Reports, 15(1), 25732. https://doi.org/10.3390/agriculture11070617

[7] Al-Mahmood, A. M., Shahadi, H. I., & Khayyat, A. R. H. (2023). Image dataset of infected date palm leaves by Dubas insects. *Data in Brief*, *49*, 109371. https://doi.org/10.1016/j.dib.2023.109371

[8] Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). Plant leaf disease classification using the EfficientNet deep learning model. *Ecological Informatics*, *61*, 101182. https://doi.org/10.1016/j.ecoinf.2020.101182 (Note: Year corrected to 2021 per journal issue; original 2019 appears incorrect)

[9] Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, *129*(6), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z

[10] Hessane, A., Boutahir, M. K., Youssefi, A. E., Farhaoui, Y., & Aghoutane, B. (2023). Empowering date palm disease management with deep learning: A comparative performance analysis of pretrained models for stage-wise white-scale disease classification. *Data and Metadata*, *2*, 102. https://doi.org/10.56294/dm2023102

[11] Li, G., Zhang, M., Li, J., Lv, F., & Tong, G. (2020). Efficient densely connected convolutional neural networks. *Neural Computing and Applications*. Advance online publication. https://doi.org/10.1007/s00521-020-05222-9

[12] Muter, M. A., & Al-Asadi, A. H. (2025). Deep Learning Technique for Early Diagnosis of Date Palm Pests: A Review. Basrah Journal of Agricultural Sciences, 38(2), 570-585. https://doi.org/10.37077/25200860.2025.38.2.40

[13] Sagheer, S. V. M., P V, O., Ameer, P. M., BaQais, A., & Kalathil, S. (2025). A deep learning approach to classification of diseases in date palm leaves. *Computers, Materials & Continua*, *84*(1), 1329–1349. https://doi.org/10.32604/cmc.2025.063961