ISSN: 3107-6513

Plant Disease Detection Using Deep Learning

Gunasekar K[1], Atchaya AJ[2], Dharika S[3], Lithanya C[4], Dr. K. Priya[5]
[1,2,3,4]Students, Department of Computer Science with Artificial Intelligence and Data Science, KPR College of Arts, Science and Research, Coimbatore, Tamil Nadu, India
[5]Department of Computer Science with Artificial Intelligence and Data Science, KPR College of Arts, Science and Research, Coimbatore, Tamil Nadu, India

Abstract
Plant diseases can significantly affect crop health and agricultural productivity. Early identification of visible symptoms is therefore important for timely crop care and disease management. This paper presents a proposed deep-learning-based image classification system for detecting plant diseases from leaf images. The project uses a dataset containing approximately 35,000 plant and leaf images, stored in Google Drive and accessed through a Google Colab development environment. The workflow includes image collection, organization by class, preprocessing, model training, validation, testing, and prediction for a new input image. Preprocessing may include resizing images to a consistent input dimension and normalizing pixel values. A labelled image dataset is used to help the model learn visual characteristics such as colour, texture, shape, and disease-related patterns. The intended output is a predicted disease category for an uploaded leaf image. Suitable evaluation measures include accuracy, precision, recall, F1-score, and a confusion matrix. As the project is still under development in the supplied project material, final model architecture, training configuration, and measured performance are not yet reported. The proposed system is intended as an educational and preliminary decision-support tool, not a replacement for expert agricultural diagnosis.

Keywords— Plant Disease Detection; Deep Learning; Image Classification; Leaf Disease; Computer Vision; Google Colab; Python.

I. INTRODUCTION

Agriculture plays an important role in food production and rural livelihoods. Plant diseases can reduce crop quality and yield, and visible symptoms may appear as spots, discolouration, patches, curling, or changes in leaf texture. Identifying a disease at an early stage can help growers seek appropriate advice and take timely action. In many situations, however, identification depends on visual inspection and access to people with relevant agricultural expertise. Manual inspection can be time-consuming, especially when many plants must be checked or when expert support is not readily available.

Artificial intelligence and computer vision offer methods for analysing images and recognizing patterns. In image classification, a model is trained using labelled examples and learns features that help distinguish one class from another. Deep learning models can learn multiple levels of visual representation directly from image data. For plant disease detection, these representations may include leaf colour, lesion shape, texture, and the distribution of visible symptoms.

This project proposes an image-based plant disease detection workflow. A user supplies a leaf image, the image is prepared in a consistent format, and a trained classifier predicts the corresponding class. The development environment described in the project material is Python in Google Colab, with the dataset stored in Google Drive. The work is intended as an academic demonstration of image preprocessing, model development, and prediction.

A. Need for image-based identification

A photograph provides a convenient input for a computer-based screening workflow. It can be reviewed by a classifier without requiring the user to manually describe every visible feature. The model learns from labelled examples rather than relying only on a fixed list of manually written visual rules. This does not remove the need for agricultural expertise; it offers a preliminary classification that can help organize further inspection.

II. PROBLEM STATEMENT

Traditional plant disease identification often relies on manual observation. The process may take time and can be difficult when symptoms are subtle, when several diseases have similar appearances, or when expert guidance is unavailable. Image quality, lighting, camera angle, background variation, and the growth stage of a plant can also influence how symptoms appear in a photograph.

The project therefore addresses the need for an accessible preliminary method that can classify plant leaf images using a deep-learning workflow. The system should accept an image, prepare it for model input, and return a predicted disease category. Its predictions must be evaluated on images not used for model training. Because the supplied project is still under development, the paper does not claim a final accuracy or a validated field performance.

III. OBJECTIVES

The main objectives of the proposed system are: (1) to prepare a large collection of plant leaf images for model development; (2) to preprocess images into a consistent format suitable for deep learning; (3) to train an image-classification model using labelled examples; (4) to monitor learning using validation data; (5) to evaluate the trained model using unseen test images and appropriate classification measures; and (6) to provide

ISSN: 3107-6513

a simple prediction workflow that returns the predicted plant disease class for a new image.

IV. LITERATURE REVIEW

Deep learning has become an important approach in image recognition because convolutional neural networks can learn spatial patterns from image data. Krizhevsky, Sutskever, and Hinton demonstrated the effectiveness of deep convolutional neural networks for large-scale image classification [1]. Their work helped establish deep neural networks as a strong approach for visual recognition tasks.

He et al. introduced residual learning, in which shortcut connections help train deeper neural networks [2]. Residual architectures are widely used as image feature extractors and provide one possible direction for future experiments in plant image classification. The residual connection helps information move through a deep network and can make the training of deeper architectures more manageable.

Ronneberger et al. proposed U-Net for biomedical image segmentation [3]. Although segmentation differs from whole-image classification, the work illustrates how deep networks can learn spatial structures in images. Segmentation identifies regions within an image, while the present project focuses on assigning an overall category to a leaf photograph.

For the present academic project, the central task is image classification rather than pixel-level segmentation. The supplied project description identifies a dataset of approximately 35,000 images and a Google Colab workflow. The final architecture and experimental settings are to be confirmed after implementation and training.

A. Relevance to the proposed work

The reviewed studies provide a foundation for understanding how neural networks can learn visual features from images. They also show that different network designs serve different tasks. The project will use an image-classification approach because its intended output is a disease category for the complete input image. The final model should be selected after practical training and evaluation rather than assuming that one architecture will perform best for every dataset.

V. DATASET DESCRIPTION AND PREPROCESSING

The project material describes a dataset containing approximately 35,000 images of plants and leaves representing different disease categories. The images are stored and accessed through Google Drive and used in a Google Colab development environment. The supplied description does not provide a complete class-by-class image count, exact dataset source, or final train-validation-test proportions; these details should be added after the dataset structure is finalized.



Fig. 1. Illustrative examples of healthy and symptomatic leaf appearances. These are explanatory visuals, not verified samples from the project dataset.

A. Image loading and organization

Images are loaded from the dataset folders and associated with their class labels. Each image should be checked for readability and correct labelling. Organizing the images into class-wise folders makes it easier to load examples and maintain the relationship between an image and its target category. The number of images in each class should be recorded to identify possible class imbalance.

During inspection, unreadable files, repeated images, unrelated photographs, and images with uncertain labels should be identified. A consistent folder structure also makes it easier to reproduce the same data-loading process during later experiments. The class names used by the program should be recorded so that the model output can be interpreted correctly.

B. Resizing and normalization

Images may have different dimensions and pixel-value ranges. Resizing converts each image to the input dimensions required by the selected model. Pixel normalization scales image values to a consistent numerical range, which can support stable model training. The exact image dimensions and normalization method must match the final model implementation.

The preprocessing operation used for training must also be applied to validation, test, and newly uploaded images. If the training images are resized or normalized differently from prediction images, the model may receive inputs that do not match the representation it learned. Maintaining one consistent preprocessing pipeline is therefore an important part of implementation.

C. Datasetsplitting and augmentation

The dataset should be divided into training, validation, and testing subsets. Training images are used to update model parameters, validation images are used to monitor learning and support model selection, and test images are reserved for final evaluation. The split should be performed carefully to avoid placing duplicate or near-duplicate images across subsets. The final split percentages should be documented after they are selected.

The training subset provides examples from which the model learns the relationship between leaf appearance and class labels. The validation subset

provides an independent check during development and can help identify when a model is no longer improving on data that were not used to update its parameters. The test subset is kept separate until the final evaluation so that it gives a more reliable indication of performance on unseen images.

Image augmentation, such as modest rotation, flipping, or zooming, may be considered during training to improve robustness to ordinary image variation. Augmentation creates modified versions of training examples and can expose the model to a wider range of orientations or framing. It should be applied carefully so that transformations do not remove important disease symptoms or change the meaning of a class label.



Fig. 2. Conceptual dataset workflow. The final split proportions are implementation-dependent and have not been specified in the supplied project description.

D. Dataset quality considerations

The quality of the labels is important because the model learns from the categories assigned to the images. Incorrect labels can teach the model misleading patterns. Images should therefore be checked for basic consistency, and class-wise image counts should be reviewed before training. If some classes contain many more images than others, the evaluation should report class-wise measures so that performance on smaller classes is not hidden by the overall score.

The dataset summary should ultimately include the confirmed source, plant types, disease categories, class-wise counts, image dimensions, and split proportions. The supplied project material confirms an approximate total of 35,000 images but does not yet specify these additional details. They should be inserted only after they have been checked against the actual dataset.

VI. PROPOSED METHODOLOGY

The proposed methodology consists of five main stages: dataset collection, preprocessing, model training, evaluation, and prediction. Each stage contributes to converting a raw leaf photograph into a predicted class. The workflow is designed to be implemented in Python using Google Colab, while Google Drive provides access to the image dataset.



Fig. 3. Proposed system architecture for image-based plant disease classification.

A. Dataset collection

The image dataset is collected and arranged into folders corresponding to the available plant or disease categories. Each image must have a reliable class label. Incorrect labels, unreadable images, and irrelevant photographs should be identified during dataset inspection. The dataset description should record the source, class names, and number of images per class when these details are confirmed.

B. Image preprocessing

The image-loading stage reads each photograph and its corresponding label. Preprocessing then converts images into a consistent input representation. Typical operations include resizing, converting pixel values to numerical arrays, and normalization. If the selected model expects a particular channel order or input shape, the preprocessing code must follow that requirement. The same preprocessing steps must be applied consistently during training, validation, testing, and prediction.

C. Model training

A deep-learning image-classification architecture is selected and trained using the prepared labelled images. During training, the model compares its predicted class with the known label and updates its parameters to reduce classification error. Training data are used for learning, while validation data help monitor whether the model is improving on images that are not used for parameter updates.

The architecture, optimizer, learning rate, batch size, number of epochs, and loss function are implementation choices and should be reported only after they are finalized. These settings influence the training process and should be documented so that the experiment can be repeated. No specific architecture or training configuration is claimed as completed in the supplied project material.

D. Model evaluation

After training, the model is evaluated using the reserved test subset. Accuracy gives the proportion of correctly classified images. Precision indicates how often predictions for a particular class are correct, while recall indicates how many actual examples of that class are identified. The F1-score combines precision and recall. A confusion matrix shows which classes are correctly recognized and which classes are confused with one another.

For a multi-class dataset, these measures should be reported per class as well as using clearly identified macro or weighted averages. Macro averaging gives equal importance to each class, while weighted averaging takes the number of examples in each class into account. Reporting these values together can provide a clearer view than accuracy alone, particularly when class counts are uneven.

E. Prediction

For prediction, a new leaf image is uploaded or selected by the user. The image is resized and normalized using the same preprocessing pipeline used during training. The trained model processes the image and returns class scores or probabilities. The class with the highest model score can be displayed

as the predicted category. Where possible, the interface should also communicate that the result is a model prediction and may require confirmation by an agricultural expert.

F. End-to-end workflow

The complete workflow begins when the dataset is accessed from Google Drive. The images are organized and prepared in Google Colab, after which the selected model is trained and monitored using validation data. The saved model is then evaluated using the reserved test images. For a new input, the same preprocessing steps are applied before the model generates a class prediction. Keeping these stages separate makes it easier to test and update individual parts of the system.

VII. SYSTEM IMPLEMENTATION

The supplied project description identifies Python and Google Colab as the development environment and Google Drive as the dataset storage location. The implementation can be separated into modules so that each stage can be tested independently. A dataset module reads the image folders and labels; a preprocessing module prepares image tensors; a training module defines and trains the classifier; an evaluation module calculates performance measures; and a prediction module loads the saved model and classifies a new image.

A. Development environment

Python is used for data handling and model development. Google Colab provides a notebook-based environment for executing code and conducting experiments. Google Drive is used to store the image dataset and, where required, saved model files. Libraries should be selected according to the final implementation, and their versions should be recorded to support reproducibility.

The notebook environment supports a step-by-step workflow in which data loading, preprocessing, model definition, training, and evaluation can be documented in separate sections. Saving the trained model and the class-label mapping will help ensure that a later prediction session uses the same learned parameters and output categories.

B. Main implementation modules

1) Dataset loading: access the image folders and assign labels. The loader should check that the image path and class label correspond correctly and should report files that cannot be opened.

2) Preprocessing: resize and normalize images and prepare batches. This stage should apply the same input format throughout training and prediction.

3) Model creation: define the selected deep-learning architecture and its output classes. The output layer should correspond to the confirmed set of dataset labels.

4) Training: fit the model using the training subset and monitor validation performance. The training configuration should be recorded after it is finalized.

5) Model saving: store the trained weights or complete model. The class-label mapping and preprocessing configuration should also be retained for later use.

6) Prediction: load the saved model, process a new leaf image, and display the predicted category. The prediction stage should handle unsupported or unreadable image files with a clear message.

C. Suggested user interaction

A simple user interface may provide an image-upload control and a prediction button. After the image is submitted, the system can display the uploaded image, predicted class, and an explanatory note that the output is preliminary. A clear interface can make the system easier to demonstrate because the user does not need to interact directly with the model code.

The supplied project material does not confirm that a completed web or mobile interface has been implemented; this is therefore described as a possible extension rather than a completed feature. In a future interface,

the user should be able to select a supported image, review the selected file, request a prediction, and read the result together with a short caution about its limitations.

D. Implementation summary

Item	Details available in project material
Dataset size	Approximately 35,000 plant and leaf images
Data storage	Google Drive
Development environment	Python in Google Colab
Task	Deep-learning image classification
Input	Plant or leaf image
Output	Predicted disease category
Final performance	Pending completion of training and evaluation

VIII. RESULTS AND DISCUSSION

The supplied project document states that the project is under development and that final model performance will be reported after complete training and evaluation. Accordingly, no numerical accuracy, precision, recall, F1-score, or confusion-matrix values are claimed in this paper. Reporting unmeasured values would not provide a reliable assessment of the proposed system.

Once training is complete, the results section should include the final model name, training configuration, dataset split, number of classes, and the number of images in each subset. Training and validation curves can be used to inspect learning behaviour. A confusion matrix can help identify disease categories that have similar visual characteristics. Misclassified examples

should be reviewed to understand whether errors are associated with poor image quality, overlapping symptoms, background clutter, class imbalance, or incorrect labels.
The final evaluation should use the untouched test subset. If multiple classes are present, a single overall accuracy value may hide poor performance on a smaller class. Per-class precision and recall, macro F1-score, weighted F1-score, and a confusion matrix provide a more complete view. Any performance result should be accompanied by the test-set size and the evaluation procedure.

A. Interpretation of evaluation measures

Accuracy describes the proportion of all test images that receive the correct class. Precision focuses on the reliability of predictions made for a particular disease category. Recall focuses on how many of the actual images belonging to that category are found by the model. The F1-score combines precision and recall and can be useful when both missed detections and incorrect alerts matter.
The confusion matrix gives additional detail by showing the actual class alongside the predicted class. Large values along its main diagonal indicate correct classifications, while values outside the diagonal indicate cases where one category was mistaken for another. Reviewing those errors can guide improvements to image quality checks, class balance, preprocessing, and model selection.

B. Discussion

The proposed system combines image preparation and deep-learning classification in a single workflow. Its usefulness will depend on whether the training images represent the range of leaf appearances that the system is expected to process. Performance should therefore be interpreted in relation to the dataset, class distribution, image conditions, and evaluation procedure rather than being treated as a universal measure of plant disease recognition.
Because the final model and measured results are not yet available in the supplied project material, the present paper describes the planned evaluation rather than reporting a completed experiment. After implementation, the results can be added without changing the overall workflow: report the test-set composition, provide the evaluation metrics, show the confusion matrix, and discuss the most frequent classification errors.

IX. ADVANTAGES AND APPLICATIONS

The proposed workflow can provide a faster preliminary review of leaf images than checking every image manually. It offers a practical academic example of how image preprocessing, deep learning, and computer vision can be combined in one application. It may support learning activities in artificial intelligence and data science and may provide a starting point for agricultural decision-support applications.
Potential applications include educational demonstrations, preliminary crop monitoring, plant-health awareness tools, and future mobile or web-based assistance systems. The model should be used as a supporting aid. A prediction from an image alone should not be treated as a definitive diagnosis or as a substitute for advice from qualified agricultural professionals.

A. Educational value

The project provides an opportunity to understand a complete machine-learning image workflow, from organizing labelled data to preparing images, training a classifier, evaluating predictions, and presenting an output. It also demonstrates why data quality, consistent preprocessing, validation, and careful reporting are important parts of an artificial-intelligence project.

X. LIMITATIONS AND FUTURE ENHANCEMENT

Image-based disease classification has several limitations. The quality of predictions can be affected by lighting, focus, image resolution, camera angle, background variation, and the stage of disease development. Different diseases may produce similar visible symptoms, while the same disease may look different across plant varieties or environmental conditions. Class imbalance can also cause the model to perform better on common classes than on less frequent classes.
The current project material does not provide the final model architecture, class distribution, dataset split, training settings, or measured test results. These details must be finalized and reported before the system's performance can be assessed. The dataset's coverage should also be checked to determine whether the images represent the plant types, disease categories, and real-world conditions relevant to the intended users.
Future enhancements may include testing transfer-learning architectures, collecting additional images from varied field conditions, balancing the class distribution, tuning training parameters, and improving the prediction interface. The system could be extended into a mobile or web application after the model is validated. Testing with real-world photographs and feedback from agricultural experts would be important steps before practical deployment.

A. Further development

A future version can compare more than one suitable image-classification architecture under the same data split and evaluation procedure. The comparison should consider per-class precision and recall as well as overall scores. Additional images captured under different lighting, backgrounds, and camera conditions may help assess whether the model generalizes beyond the images sed during development.
A user-facing application may also include clear instructions for taking a useful leaf photograph, a display of the predicted category, and a reminder that the output is preliminary. Before practical use, the system should be tested with representative field images and reviewed with qualified agricultural experts.

XI. CONCLUSION

This paper presents a proposed deep-learning-based approach for plant disease detection from leaf images.

The project uses a dataset of approximately 35,000 plant and leaf images and describes a development workflow based on Python, Google Colab, and Google Drive. The methodology covers dataset organization, image preprocessing, model training, validation, testing, and prediction of a disease category from a new image.

The approach demonstrates how artificial intelligence and image classification can be applied to preliminary plant-health assessment. However, the project is still under development in the supplied material, and final model performance has not yet been established. The model architecture, training configuration, dataset split, and evaluation results should be added after implementation is completed. With careful testing and suitable validation, the system may serve as an educational prototype and a foundation for future agricultural assistance tools.

ACKNOWLEDGMENT

The authors express their sincere gratitude to KPR College of Arts, Science and Research for providing the academic environment and support required to carry out this project. The authors also acknowledge the guidance and academic support provided by Dr. K. Priya and the researchers whose published work on deep learning and image recognition informed the background of this study.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[2] 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, pp. 770–778, 2016.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241, 2015.

[4] The project dataset description supplied by the authors: approximately 35,000 plant and leaf images, stored in Google Drive and used in a Google Colab development environment. Dataset source and class-wise counts should be added when confirmed.

Author note: This manuscript is an academic project research-paper draft based on the supplied Plant Disease Detection project document. Illustrative leaf visuals and workflow diagrams are included for explanation. They are not presented as measured experimental outputs or verified samples from the project dataset.