Synopsis

on

A Lightweight CNN Framework for Multi-Class Cassava Leaf Disease Detection

For the fulfillment of the Project-I (BCS753) of

Bachelor of Technology

Department of Computer Science and Engineering

Bhagwant Institute of Technology, Muzaffarnagar

Odd Semester 2025

Supervisor:

Assistant Professor

Submitted by:

Saloni

B.Tech-VII Sem

Roll No. 2200840100020

TABLE OF CONTENTS

INTRODUCTION

1.1 Background

Cassava (Manihot esculenta) is a vital staple crop for millions of people, especially in Sub- Saharan Africa, due to its resilience and nutritional value. However, cassava production is severely threatened by various leaf diseases, including Cassava Mosaic Disease, Cassava Brown Streak Disease, Cassava Green Mottle, and Cassava Bacterial Blight. These diseases can cause significant yield losses and threaten food security in regions heavily dependent on cassava.

By leveraging lightweight CNNs, it becomes feasible to provide farmers and agricultural extension workers with accessible tools for rapid cassava disease diagnosis, supporting timely management decisions and contributing to improved crop yields and food security.

Fig. 1: Visual samples of diseased Cassava Leaves used for Classification

1.2 Problem Statement

Cassava leaf diseases significantly threaten crop yields and food security, particularly in developing regions where cassava is a staple food. Traditional disease identification methods are slow, subjective, and often inaccessible to smallholder farmers, leading to delayed or incorrect diagnoses.

While deep learning-based image classification offers a promising solution, most existing Convolutional Neural Network (CNN) models are computationally intensive and unsuitable for deployment on mobile devices or in resource-constrained environments. There is a critical need for a lightweight, accurate, and efficient CNN framework capable of robust multi-class cassava leaf disease detection, enabling rapid, on-site diagnosis and empowering farmers with timely, actionable information for effective disease management and improved agricultural outcomes.

1.3 Objectives

The main objectives of this project are as follows:

- To apply data augmentation techniques to diversify the training set, thereby improving the model's ability to generalize to unseen data.
- To address the issue of class imbalance by implementing class weighting strategies, ensuring fair training for all classes.
- To develop a deep learning-based leaf image classification and detection model for identifying plant diseases.
- Demonstrate that the custom model can outperform standard models in real-world disease classification tasks.

1.4 Significance of the Study

The application of deep learning in agriculture has the potential to revolutionize traditional farming practices. The ability to classify cassava plant diseases accurately and promptly can significantly reduce crop losses and improve agricultural productivity. This project aims to contribute to the growing field of smart agriculture by providing a framework for disease detection using AI. By automating the identification of diseases, farmers and agricultural experts can make informed decisions regarding the management and treatment of crops. This will lead to a reduction in the excessive use of pesticides, thereby promoting sustainable farming practices.

Furthermore, this study emphasizes the use of Custom CNN and data augmentation, showcasing how pre-trained models can be adapted for specialized tasks, even with limited datasets. The use of the Convolutional Neural Network architecture, known for its efficiency in extracting complex features, provides a robust solution to the challenges associated with image classification in agriculture. The outcome of this project is expected to demonstrate that deep learning can be a practical and reliable tool for solving real-world problems in the agricultural sector.

RELATED WORK

Deep learning (DL) has significantly advanced cassava leaf disease detection by enabling automated and highly accurate classification of leaf images. Convolutional Neural Networks (CNNs) play a central role by eliminating the need for manual feature extraction and allowing real-time diagnostic capabilities in agriculture. [2] In addition to custom CNN architectures, pretrained models such as VGG16, ResNet50, and MobileNetV2 [6][5][1] have been widely adopted due to their ability to transfer learned features from large-scale datasets to the cassava disease domain, improving both efficiency and performance. Recent research from 2017 to 2025 has focused on employing CNNs and pretrained deep learning models for identifying cassava leaf diseases. Table I summarizes key studies and approaches proposed during this period.

TABLE I: Comparison of Existing Research on Cassava Leaf Disease Detection

Author (Year)	Dataset Used	No. of Images	Methodology	Accuracy (%)
Ramcharan et al. (2017) [6]	Original cassava	2,756	InceptionNetV3	93.00
Sambasivam et al. (2021) [7]	Cassava 2020	21,397	DenseNet169 + EfficientNetB0	89.94
Metlek (2021) [5]	Cassava 2020	21,397	ResNet50 + SVM	84.40
Anitha et al. (2022) [4]	Cassava 2019	22,031	A custom CNN	90.00
Lilhore et al. (2022) [8]	Cassava 2020	6256	A Custom ECNN	99.47
Kalpana et al. (2024) [1]	Cassava 2020	21,397	EfficientNet	94.27
Sambasivam et al. (2025) [2]	Cassava 2019	10,000	Custom CNN	93.00
Ademir G. et al. (2025) [3]	Cassava 2020	21,397	EfficientNetB3	87.70

Early work by Ramcharan *et al.* [6] demonstrated the effectiveness of deep learning for cassava disease detection using transfer learning with ResNet50, achieving 93% accuracy. Sambasivam and Opiyo [7] evaluated multiple pretrained models, with a hybrid

DenseNet169 + EfficientNetB0 model reaching 89.94% accuracy, and EfficientNetB0 achieving the best F1-score (0.78).

Metlek [5] combined CNN feature extraction with traditional classifiers, with ResNet50 + SVM yielding 84.4% test accuracy. Anitha and Saranya [4] proposed a lightweight CNN achieving 90% validation accuracy. Lilhore *et al.* [8] introduced an Enhanced-CNN with 239 layers, reaching 99.47% accuracy after applying SMOTE and K-fold validation.

Kalpana *et al.* [1] compared models on the Kaggle cassava dataset, with DenseNet121 achieving 94.6%. Sambasivam *et al.* [2] trained a CNN from scratch using SMOTE, class weighting, and focal loss, achieving 93%.

Overall, CNNs and hybrid models show strong potential for accurate, scalable cassava disease detection, suitable for realworld deployment in agricultural settings.

TOOLS AND LIBRARIES

3.1 Python 3.4 Programming Language

Python is a high-level, interpreted programming language that is widely used in data science, machine learning, and artificial intelligence projects due to its simplicity, versatility, and extensive library support. Python 3.4, the latest version of Python, offers improved syntax, better memory management, and new features that make it an ideal choice for complex computations and machine learning applications.

Key Features of Python 3:

- Ease of Use: Simple syntax makes Python easy to read and write, enabling rapid development and prototyping.
- Extensive Libraries: Python has a rich ecosystem of libraries specifically designed for data analysis, visualization, and machine learning.
- Community Support: Strong community support with a plethora of resources, tutorials, and documentation.

In this project, Python 3.4 serves as the primary programming language for building the deep learning model, handling data preprocessing, training, and evaluation.

3.2 Jupyter Notebook

Jupyter Notebook is an open-source web-based interactive computing environment that allows developers to create and share code, equations, visualizations, and text. It is widely used for data analysis, scientific research, and machine learning development.

Key Features of Jupyter Notebook:

- Interactive Coding: Allows for real-time code execution, making it easier to test and debug code step-by-step.
- Rich Text Support: Enables documentation with Markdown, creating welldocumented notebooks that combine code, explanations, and results.
- Visualization: Integrates seamlessly with visualization libraries like matplotlib and seaborn for plotting data.

In this project, Jupyter Notebook is used to develop and experiment with the model in a modular fashion, allowing detailed analysis at each step of the pipeline.

3.3 Google Colab

Google Colab, short for "Colaboratory," is a cloud-based platform that provides a Jupyter Notebook environment. It is particularly popular for machine learning and deep learning projects because of its ability to leverage powerful hardware (GPUs and TPUs) for free, making it accessible to a wide audience.

Key Features of Google Colab:

- Free Access to GPUs/TPUs: Enables faster training of deep learning models without the need for expensive local hardware.
- Collaboration: Allows multiple users to work simultaneously on the same notebook, facilitating teamwork and knowledge sharing.
- Integration with Google Drive: Easy integration with cloud storage for data handling and saving trained models.

For this project, Google Colab is used to train the custom model, perform data augmentation, and evaluate the results using high-performance cloud resources.

3.4 Tensor Flow

TensorFlow is an open-source machine learning framework developed by Google. It is widely used for building, training, and deploying deep learning models. TensorFlow is known for its flexibility and scalability, supporting a range of tasks from research experiments to production-scale applications.

Key Features of TensorFlow:

- Wide Range of APIs: Provides high-level APIs like Keras for rapid prototyping and low-level APIs for advanced customizations.
- Scalability: Can be used for training models on a single machine or distributed across multiple machines with GPUs.
- Model Deployment: TensorFlow offers tools like TensorFlow Serving and TensorFlow Lite for deploying models to production environments.

In this project, TensorFlow is the core framework used for implementing the custom CNN image classification model. It provides the computational backbone for training the neural network on the Cassava Leaves dataset.

3.5 Keras

Keras is a high-level neural network API written in Python, running on top of TensorFlow. It is designed to enable fast experimentation with deep learning models, providing a simple and intuitive interface to define and train neural networks.

Key Features of Keras:

- User-Friendly API: Simplifies the process of creating deep learning models with clean and minimalistic code.
- Modularity: Models are built by combining independent building blocks, making them easy to design and modify.
- Support for Pre-trained Models: Offers access to pre-trained models like MobilenetV2, which are ready to be fine-tuned for specific tasks.

In this project, **Keras** is the high-level API used for implementing the custom CNN image classification model. It simplifies the process of building, training, and evaluating deep learning models on the Cassava Leaves dataset, while running on top of TensorFlow as the backend.

3.6 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used in data analysis to plot graphs, visualize data distributions, and present model performance metrics.

Key Features of Matplotlib:

- Versatile Plotting: Provides tools to create a variety of plots, including line charts, scatter plots, histograms, and heatmaps.
- Customization: Allows detailed customization of plot elements (titles, labels, colors) for publication-quality visuals.
- Integration: Easily integrates with Jupyter Notebook for inline visualization.

In this project, Matplotlib is used to visualize the training and validation loss and accuracy trends over epochs, as well as to generate the confusion matrix for model evaluation.

3.7 NumPy

NumPy is the fundamental package for scientific computing with Python. It provides support for multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these data structures efficiently.

Key Features of NumPy:

- Array Handling: Facilitates fast and efficient manipulation of large arrays and matrices.
- Mathematical Operations: Offers a wide range of mathematical functions for operations on arrays, such as linear algebra and statistical calculations.
- Integration with Other Libraries: Acts as the base for many other scientific libraries like TensorFlow and Pandas.

In this project, NumPy is used for handling numerical operations, such as processing predictions and calculating evaluation metrics like precision and recall.

3.8 Pandas

Pandas is a powerful open-source data analysis and manipulation library for Python. It provides data structures like DataFrames and Series to handle structured data intuitively.

Key Features of Pandas:

- Data Handling: Efficiently manages structured data, enabling operations like filtering, grouping, and merging datasets.
- Data Cleaning: Simplifies data cleaning and preprocessing, making it easier to prepare data for analysis or machine learning.
- File I/O: Supports reading and writing data to various formats (CSV, Excel, SQL), making data management convenient.

In this project, Pandas can be used for data preprocessing tasks if necessary, such as loading and inspecting metadata, calculating statistics, or managing labels for classification.

3.9 Pillow (PIL)

Pillow, also known as PIL (Python Imaging Library), is an image processing library for Python. It is used to open, manipulate, and save image files, making it essential for tasks like data augmentation and pre-processing.

Key Features of Pillow:

- Image Manipulation: Supports operations like resizing, cropping, rotating, and filtering images.
- File Format Compatibility: Can handle multiple image formats (JPEG, PNG, BMP, etc.).
- Integration: Works well with libraries like TensorFlow for processing and displaying images.

PROPOSED METHODOLOGY

The proposed methodology for cassava leaf disease identification adopts a structured and systematic pipeline, as illustrated in Fig. 2.

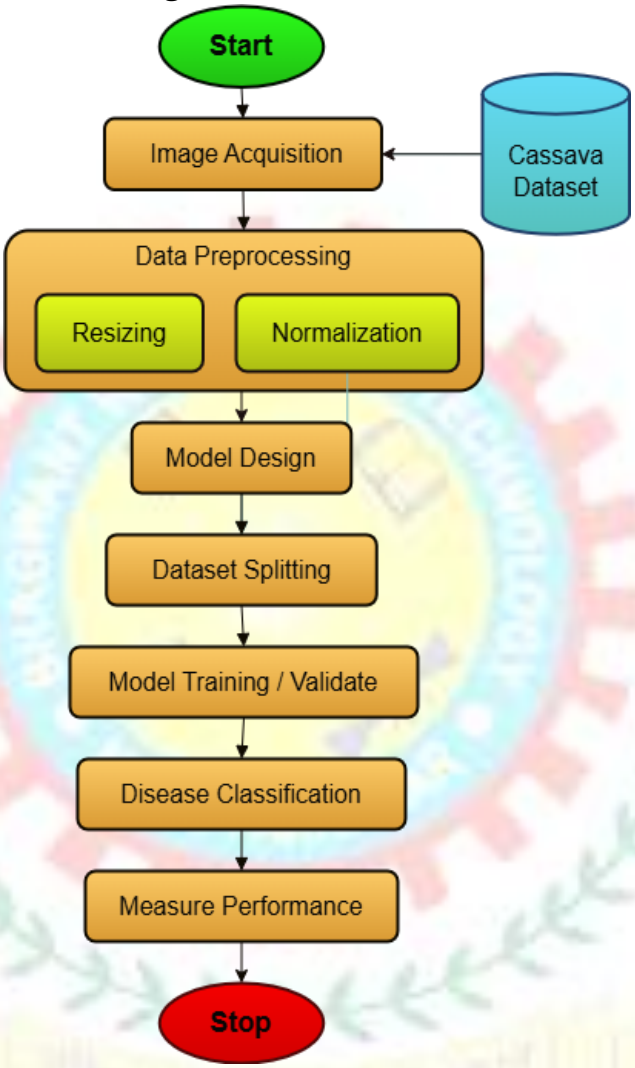


Fig. 2: Flowchart of the Proposed Methodology

The process commences with the selection of an appropriate dataset, specifically the publicly available Cassava Leaf Disease dataset from the Kaggle Competition 2020, which is widely recognized in plant pathology research. The raw images are subsequently subjected to standard preprocessing techniques, including image resizing and normalization, to ensure uniformity and enhance model performance. Following preprocessing, the dataset is partitioned into training and validation subsets to facilitate effective model learning and evaluation. The dataset comprises five distinct classes—Cassava Mosaic Disease (CMD),

Cassava Green Mottle (CGM), Cassava Brown Streak Disease (CBSD), Cassava Bacterial Blight (CBB), and Healthy leaves—offering a diverse set of samples essential for developing a robust and generalizable disease classification model.

4.1 Image Acquisition

This study utilizes the Cassava Leaf Disease Dataset from the 2020 Kaggle competition titled *"Cassava Leaf Disease Classification."* The dataset comprises 21,397 RGB images (800×600 resolution) of cassava leaves, manually annotated by agricultural experts. It captures diverse conditions in terms of lighting, orientation, and background, making it ideal for training robust deep learning models. The dataset was split into 80% training and 20% testing sets, with 20% of the training portion reserved for validation.

*1) Dataset Description:*Cassava Leaf Disease Dataset:

This dataset contains five classes representing major cassava diseases and healthy leaves. The class-wise distribution, high resolution format, and expert annotation contribute to its effectiveness for plant disease classification tasks. Table II summarizes the dataset, and Fig. 3 visualizes the number of images per class.

TABLE II: Cassava Dataset Overview

Attribute	Details
Name of Dataset	Cassava Leaf Disease Dataset
Total Images	21,397
Image Type	RGB (800×600 resolution)
Total Classes	5
Class Names	Cassava Bacterial Blight (CBB), Cassava Mosaic Disease (CMD), Cassava Green Mottle (CGM), Healthy, Cassava Brown Streak Disease (CBSD)

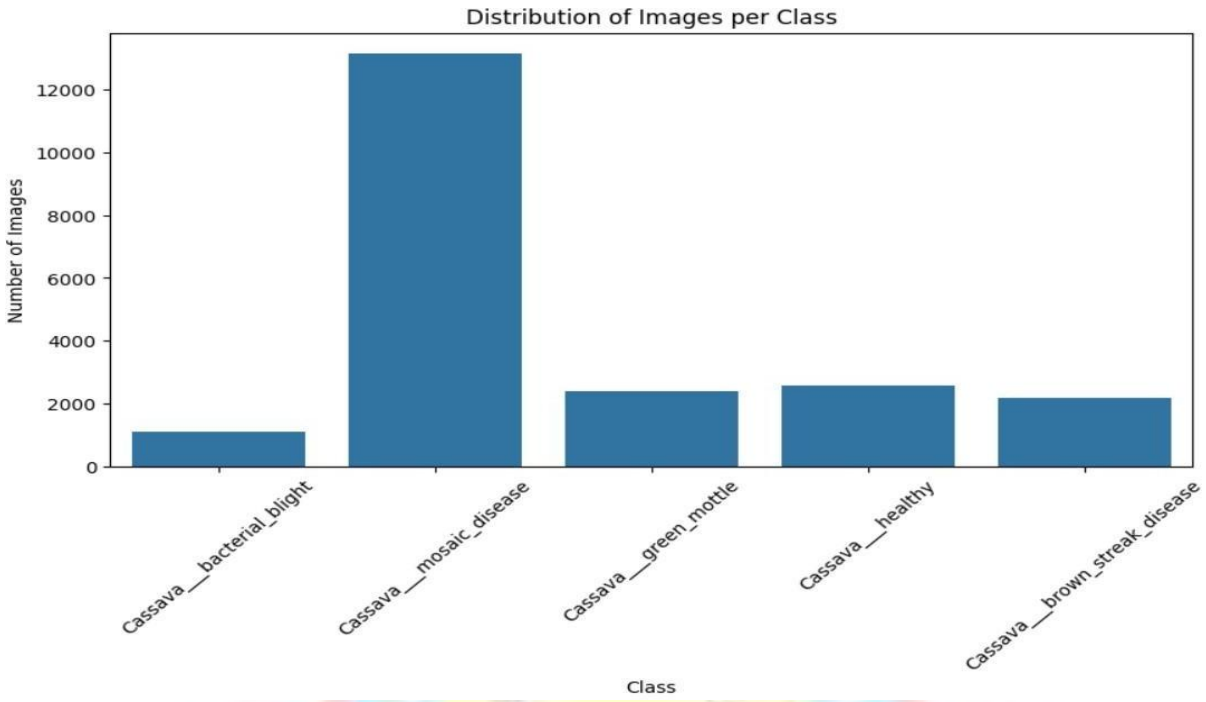


Fig. 3: Class-wise distribution of cassava leaf images.

4.2 Data Preprocessing

Data preprocessing plays a critical role in ensuring dataset consistency and enhancing the effectiveness of model training by preparing the input images in a standardized manner.

1) ***Resizing:*** All cassava leaf images were initially loaded using the OpenCV library and uniformly resized to a spatial resolution of 32 × 32 pixels. This dimensionality reduction step is instrumental in decreasing computational overhead while retaining key visual features necessary for accurate classification.

2) ***Normalization:*** To facilitate efficient convergence during the training phase, pixel intensity values for each image were normalized to the range [0,1] by dividing all pixel values by 255.0. Subsequently, each image was assigned a unique numeric label corresponding to its respective disease category. The distribution of labeled data across various classes in the original dataset is depicted in Table III.

The preprocessed image data was structured into a NumPy array of shape (21397,32,32,3), while the associated labels were maintained in a separate one-dimensional array.

3) ***Addressing Class Imbalance using SMOTE:*** The dataset exhibited significant class imbalance, with the Cassava Mosaic Disease category being disproportionately represented. To mitigate this issue, the Synthetic Minority Over-sampling Technique (SMOTE) was employed to synthetically augment the data as shown in Table III.

TABLE III: Class Distribution Before and after SMOTE minority classes

Class Name	Original Count	After SMOTE
0 (Cassava Bacterial Blight)	1,087	13,158
1 (Cassava Mosaic Disease)	13,158	13,158
2 (Cassava Green Mottle)	2,386	13,158
3 (Healthy)	2,577	13,158
4 (Cassava Brown Streak Disease)	2,189	13,158
Total	21,397	65,790

Given that SMOTE operates on tabular data, the four-dimensional image array was flattened into a 2D representation with shape (21397,3072), treating each image as a 3072-dimensional feature vector. The SMOTE algorithm was then applied to generate synthetic samples for the under-represented classes, balancing the dataset such that each class contained 13,158 samples.

This oversampling process helped in producing a more uniform distribution across all categories, addressing the class imbalance. Upon completion, the balanced dataset enabled better model training, improving classification performance by reducing bias and enhancing the model's ability to generalize across all disease categories.

Upon completion of oversampling, the dataset was reshaped back to its original four-dimensional format, resulting in a balanced dataset of shape (65790,32,32,3). This preprocessing strategy effectively alleviates class imbalance, thereby reducing model bias and enhancing classification performance across all disease categories. Moreover, this approach improves the model's generalization capabilities, enabling it to better detect and classify rare disease categories.

4.3 Model Design

The proposed Convolutional Neural Network (CNN) architecture has been rigorously engineered to enable the extraction of hierarchical spatial features, perform dimensionality reduction, and achieve high-fidelity classification of cassava leaf diseases. As depicted in Fig. 4, the network comprises a sequence of convolutional blocks, batch normalization layers, max pooling operations, dropout layers, a flattening stage, and fully connected layers. Each constituent component is strategically incorporated to enhance representational capacity, curb overfitting, and preserve computational efficiency.

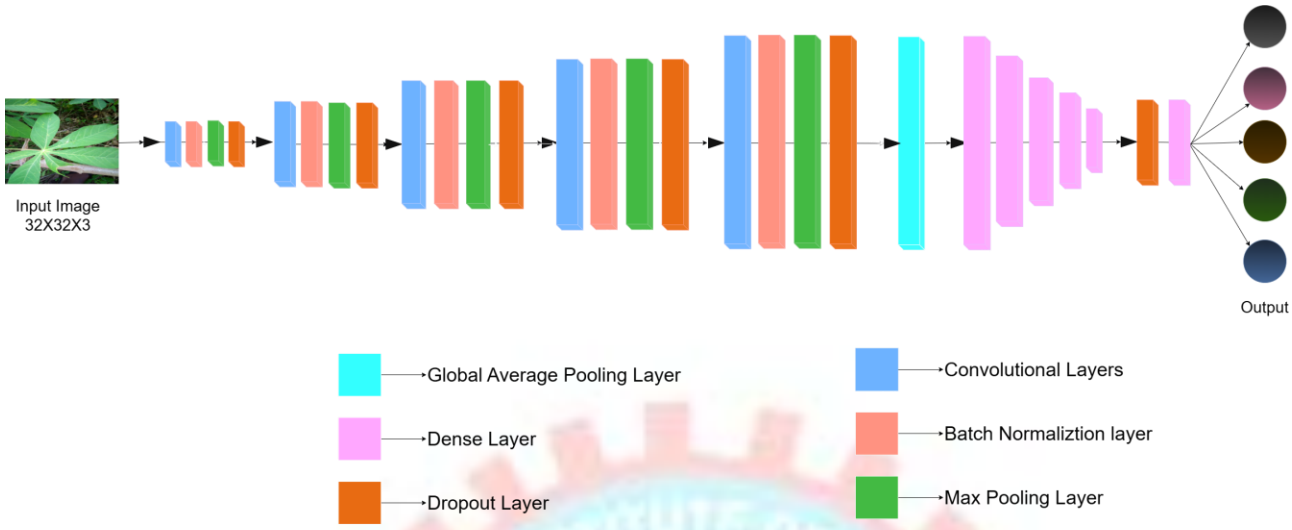


Fig. 4: Architecture of proposed CNN model

The mathematical operations underpinning each component are detailed as follows:

- Convolutional Layers: Serve as the principal feature extractors by convolving input feature maps with learnable kernels [11]. Given the input image y, the output feature map $z_p[u,v]$ at location (u,v) using kernel dimensions H,W, kernel k and bias term c_p is given by:

$$z_p[u,v] = \sum_{a=1}^{H}\sum_{b=1}^{W} y_p[u+a-1, v+b-1] \cdot k[a,b] + c_p$$

- Batch Normalization: Applied immediately after each convolutional layer to normalize activations and stabilize training [12]. Given activation I, batch mean μ, variance σ^2, and learnable parameters γ,β, the normalized output is

$$\hat{I} = \frac{I-\mu}{\sqrt{\sigma^2+\epsilon}}, \quad y = \gamma\hat{I} + \beta.$$

- Max Pooling: Reduces spatial dimensions by retaining maximal activations over non-overlapping windows of size $F \times F$ [13]:

$$Y_{i,j} = \max\{X_{p,q} \mid p \in [i, i+F-1],\ q \in [j, j+F-1]\}$$

- Dropout Layers: Introduce stochastic regularization by deactivating neurons with probability p during training [14]. For activation vector $x = [x_1,...,x_n]$ and mask $r_i \sim$ Bernoulli(p):

$$y_i = r_i \cdot x_i, \text{and at inference:} y = p \cdot x.$$

- Global Average Pooling (GAP): Perform dimensionality reduction by computing the average value over the spatial dimensions of each feature map [15]. Given a feature map $x_k[m,n]$ with spatial size $I_h \times I_w$ for the k^{th} feature map.:

$$y_k = \frac{1}{I_h \times I_w} \sum_{m=1}^{I_h} \sum_{n=1}^{I_w} x_k[m,n]$$

- Dense (Fully Connected) Layers: Execute global feature integration [16]. For input vector I, weight matrix W, bias c, and activation $g(\cdot)$:

$$O = g(WI + c).$$

- Output Layer: Consists of five neurons corresponding to target disease classes [17]. Utilizing the softmax function, the class probabilities are computed as

$$\text{softmax}(x)_k = \frac{e^{x_k}}{\sum_{l=1}^{m} e^{x_l}},$$

where x_k denotes the logit for class k.

These interconnected layers establish a fully differentiable pipeline, facilitating end-to-end training and yielding a scalable framework for accurate cassava disease classification. The complete structural configuration of the model is illustrated in Table IV.

TABLE IV: Overview of Proposed CNN Architecture

Layer Name	Kernel Size	Kernels	Activation Function	Output Shape	Parameters
Input	-	-	-	64x64x3	0
Conv2D 1	3x3	64	ReLU	64x64x64	1,792
BatchNorm 1	-	-	-	64x64x64	256
MaxPooling 1	2x2	-	-	32x32x64	0
Dropout 1	-	-	-	32x32x64	0
Conv2D 2	3x3	128	ReLU	32x32x128	73,856
BatchNorm 2	-	-	-	32x32x128	512
MaxPooling 2	2x2	-	-	16x16x128	0
Dropout 2	-	-	-	16x16x128	0
Conv2D 3	3x3	256	ReLU	16x16x256	295,168
BatchNorm 3	-	-	-	16x16x256	1,024
MaxPooling 3	2x2	-	-	8x8x256	0
Dropout 3	-	-	-	8x8x256	0
Conv2D 4	3x3	512	ReLU	8x8x512	1,180,160
BatchNorm 4	-	-	-	8x8x512	2,048
MaxPooling 4	2x2	-	-	4x4x512	0
Dropout 4	-	-	-	4x4x512	0
Conv2D 5	3x3	1024	ReLU	4x4x1024	4,719,616
BatchNorm 5	-	-	-	4x4x1024	4,096
MaxPooling 5	2x2	-	-	2x2x1024	0
Dropout 5	-	-	-	2x2x256	0
GlobalAvgPooling	-	-	-	1024	0
Dense 1	-	1024	ReLU	1024	1,049,600
Dense 2	-	512	ReLU	512	524,800
Dense 3	-	256	ReLU	256	131,328
Dense 4	-	128	ReLU	128	32,896
Dense 5	-	64	ReLU	64	8,256
Dropout 5	-	-	-	64	0
Dense (Output)	-	5	Softmax	5	325
				Total Parameters	8,026,713

RESEARCH AND DISCUSSION

5.1 Experimental Setup

The model was developed and trained on Google Colab using a virtual machine equipped with an NVIDIA T4 GPU. Implementation was done in Python with Keras and TensorFlow backend. To handle class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was applied, expanding the dataset to 52,632 training and 13,158 validation images. Training was conducted for 40 epochs with a batch size of 36, using the Adam optimizer (learning rate = 0.001) and categorical cross-entropy loss. The total training time was approximately 15 minutes.

The detailed hyperparameters used are summarized in Table V.

TABLE V: Hyperparameter Values Used in the Experimental Setup

Hyperparameter	Value
Input Size	32×32
Epochs	40
Optimizer	Adam
Learning Rate	0.001
Batch Size	36
Dropout Rate	0.5
Verbose	1
Loss Function	Categorical Cross-Entropy

5.2 Evaluation Metrics

Model performance was assessed using the following metrics:

- Accuracy: The overall proportion of correctly predicted samples.
- Precision: The ratio of true positives to total predicted positives, measuring model reliability.
- Recall (Sensitivity): The ratio of true positives to actual positives, measuring completeness.
- F1-score: Harmonic mean of precision and recall, balancing both measures.

The mathematical formulations for Precision, Recall, Accuracy, and F1 Score are summarized in Table VI, with key terms such as True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) used in the calculations to evaluate the model's performance.

5.3 Performance Evaluation and Comparative Analysis

This section conducts an in-depth evaluation of the proposed model using a subset of the Cassava Leaf dataset [9]. It encompasses the presentation of training and validation performance through visual graphs, alongside the utilization of the ROC curve, Precision-Recall curve, confusion matrix, and an extensive classification report.

1) Confusion Matrix Analysis: The confusion matrix, illustrated in Figure 5, provides a detailed breakdown of the model's classification performance across all classes, highlighting both correct predictions and misclassifications.

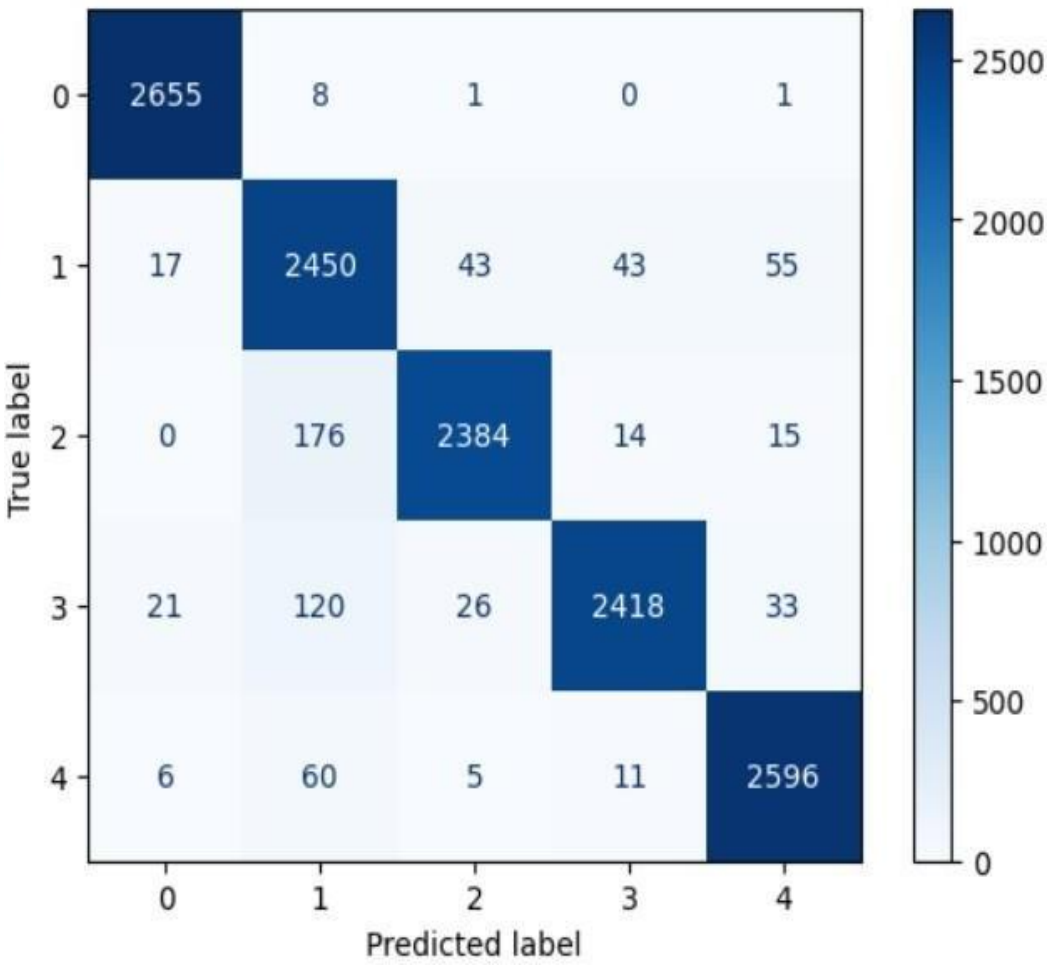


Fig. 5: Confusion Matrix

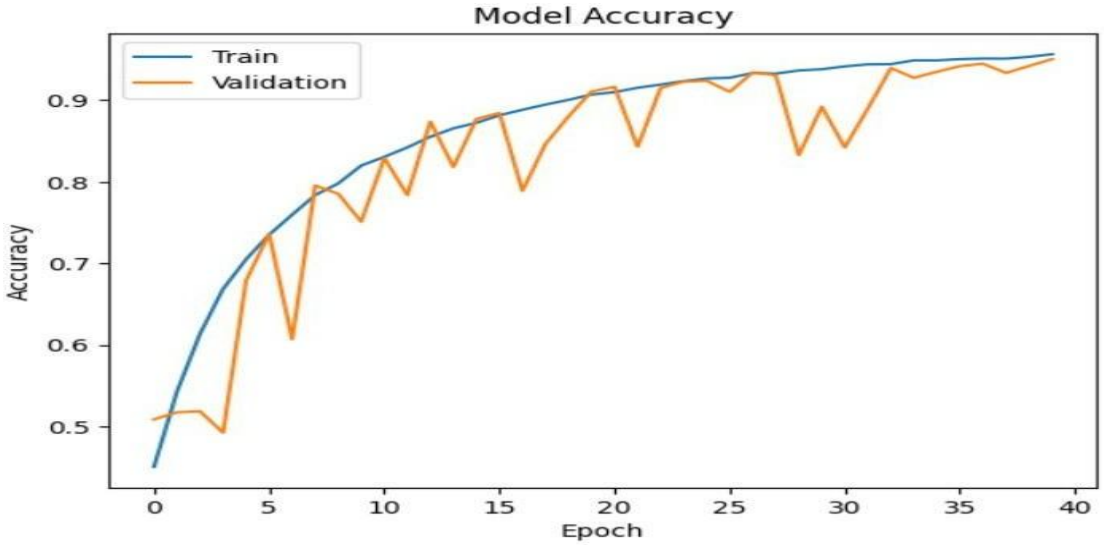
2) Classification Report: The model demonstrates high classification accuracy across all categories, with notable performance in the CBB class (Class 0), where precision, recall, and F1-score values are exceptionally high. In contrast, disease classes such as CMD (Class 1) and CGM (Class 2) exhibit slightly lower precision and recall values respectively, reflecting the inherent challenges of differentiating these diseases with high precision. Nevertheless, the F1-scores for these classes remain competitive, indicating a good balance between precision and recall.

Table VI delineates the model's performance across individual classes, detailing precision, recall, and F1-score metrics.

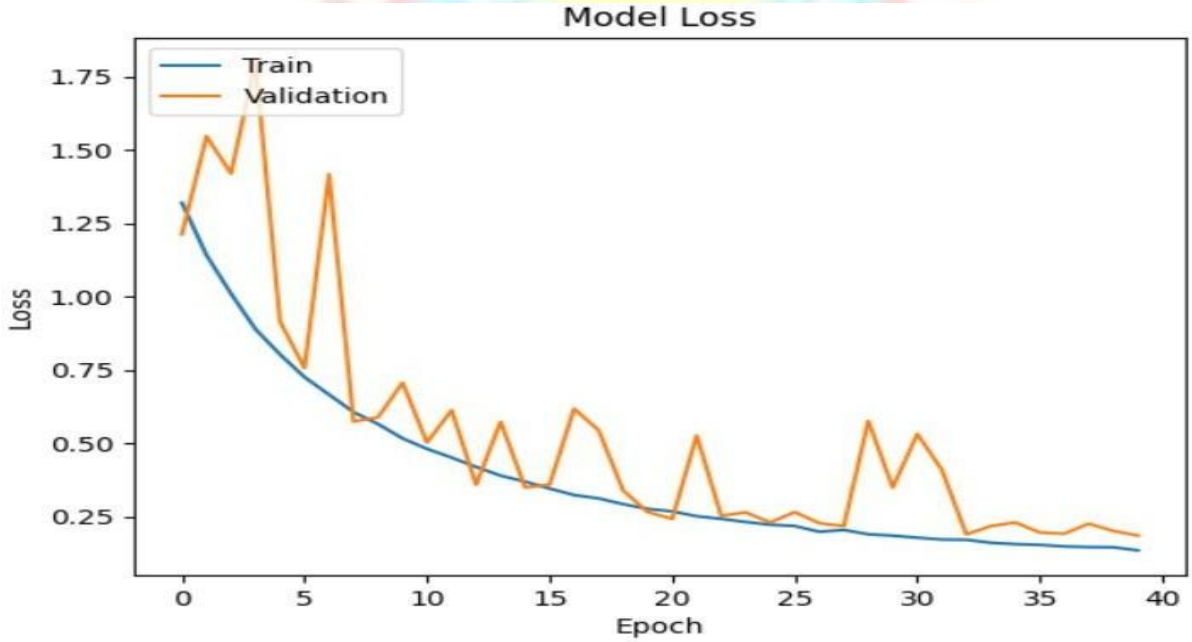
TABLE VI: Precision, Recall, F1-Score, and Support for Each Class

Class	Precision	Recall	F1-Score	Support
0 (CBB)	0.9837	0.9962	0.9899	2665
1 (CMD)	0.8706	0.9394	0.9037	2608
2 (CGM)	0.9695	0.9208	0.9445	2589
3 (Healthy)	0.9726	0.9236	0.9475	2618
4 (CBSD)	0.9615	0.9694	0.9654	2678
Accuracy	0.9502	0.9502	0.9502	–
Macro Average	0.9516	0.9499	0.9502	13158
Weighted Average	0.9518	0.9502	0.9505	13158

3) Training and Validation Performance: The model was trained for 40 epochs, and its performance was evaluated using accuracy, loss, and F1-score metrics. In Fig.6a, the training accuracy exhibits a steady upward trend, whereas the validation accuracy shows initial spikes before gradually stabilizing, indicating improving generalization over time. Similarly, Fig.6b presents a smooth decrease in training loss, while the validation loss experiences early fluctuations but eventually settles, reflecting effective convergence after some instability in the earlier stages.



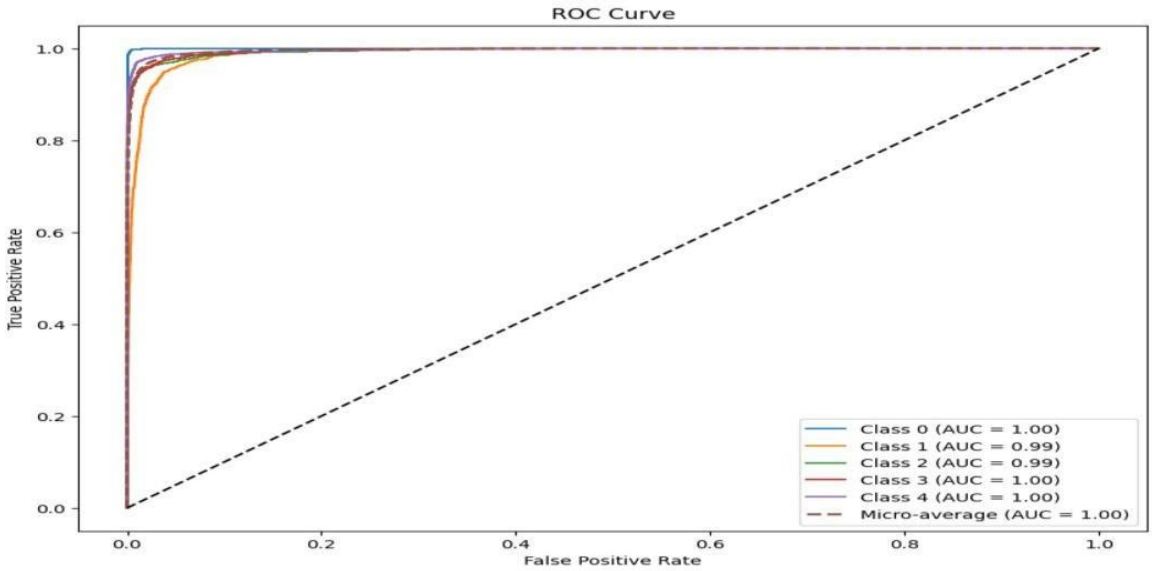
a) *Training and Validation Accuracy*



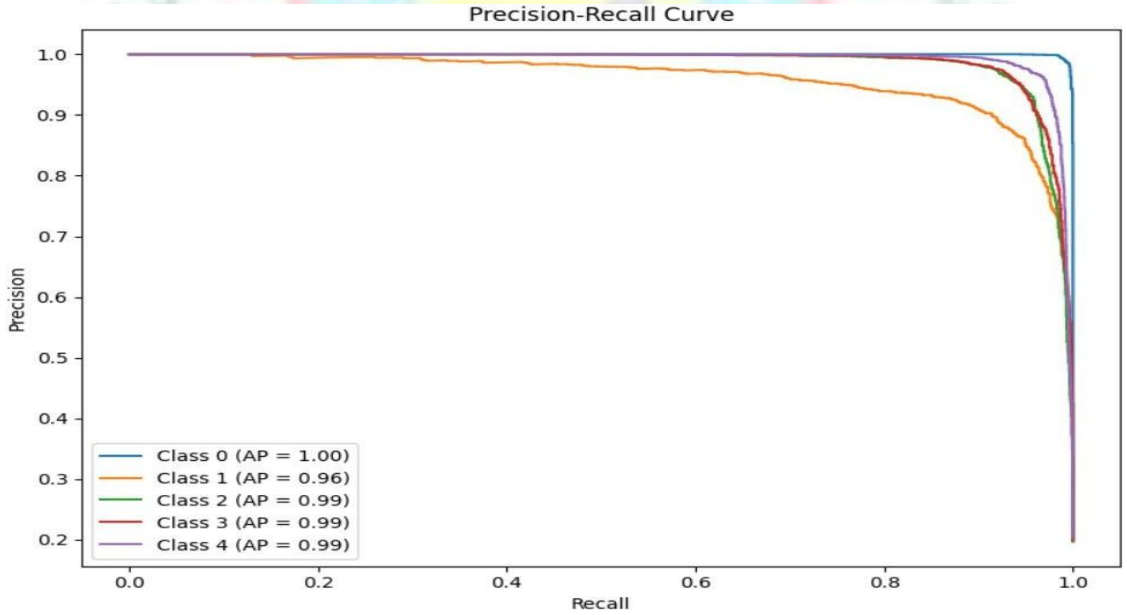
b) *Training and Validation Loss*

Fig. 6: Graphs illustrating the model's training and validation performance: (a) Accuracy and (b) Loss.

4) ROC and Precision-Recall Analysis: Figure 7 illustrates both the ROC curve and Precision-Recall curve. Figure 7a illustrates the ROC curve, showcasing the model's strong capability to distinguish between classes. Figure 7b presents the Precision-Recall curve, which highlights the model's robust balance between precision and recall. However, a minor disparity is observed in Class 1. Despite this, the overall Precision-Recall results confirm the model's effectiveness, further validating its exceptional discriminative ability.



a) ROC Curve



b) Precision-Recall Curve

Fig. 7: Performance evaluation curves of the proposed CNN model: (a) Receiver Operating Characteristic (ROC) curve and (b) Precision-Recall (PR) curve.

5) Comparative Analysis: This study introduces a lightweight, efficient, and straightforward CNN-based framework for multi-class classification of cassava leaf diseases, achieving a high accuracy of 95.02% while maintaining computational efficiency. Due to its simplicity and low resource requirements, the proposed model is particularly well-suited for real-time applications, such as deployment on mobile and IoT devices. Although certain existing models exhibit slightly higher accuracies, they often come at the cost of increased model complexity and computational demands. In contrast, the proposed approach strikes an optimal balance between performance and efficiency. A detailed comparison with recent studies is presented in Table VIII, highlighting the competitiveness of the proposed model against various CNN-based and transfer learning methodologies.

TABLE VIII: Comparative Analysis of the Proposed Method and Recent Advances in the Field.

Author Year	Methodology	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Ademir G. et al. (2025) [3]	CNN	87.70	87.80	87.80	87.70
Sambasivam et al. (2025) [2]	CNN	93.00	-	-	-
Kalpana et al. (2024) [1]	Transfer Learning	94.27	-	-	-
Lilhore et al. (2022) [8]	ECNN	99.47	-	-	-
Anitha & Sarany (2022) [4]	CNN	90.00	-	-	-
Metlek (2021) [5]	CNN	84.40	-	-	-
Sambasivam & Opiyo (2021) [7]	CNN	89.94	78.00	78.00	78.00
Ramcharan et al. (2017) [6]	Transfer Learning	93.00	-	-	-

Proposed Model	CNN	95.02	95.18	95.02	95.05

Conclusion

6.1 Impact on Agriculture

This research introduces a lightweight, efficient, and scalable Convolutional Neural Network (CNN)-based framework for the multi-class classification of cassava leaf diseases. Achieving a high validation accuracy of 95% while maintaining low computational complexity, the model is particularly well-suited for deployment in resource-constrained agricultural environments.

Overall, the study presents a robust and accessible deep learning solution tailored to real-world agricultural challenges. It not only contributes to the growing body of research in AI driven crop health monitoring but also lays a strong foundation for the development of intelligent, field-deployable diagnostic systems that can support sustainable farming practices and global food security initiatives.

6.2 Challenges

A primary challenge was dealing with dataset imbalance, where certain plant diseases had far fewer images compared to healthy plants or other disease categories. Dataset imbalance led to biased predictions, favoring more frequent classes, which required techniques like class weighting and oversampling to mitigate.

Further exploration into methods like synthetic data generation could improve the model's ability to handle class imbalance. Data quality was another challenge, as some plant leaves were partially obscured or photographed under poor lighting conditions, affecting model generalization. Improved data preprocessing and augmentation techniques could address limitations related to data quality. vi. The model's performance was sensitive to hyperparameter settings, such as learning rate and batch size.

6.3 Future Work and Improvements

Future research will aim to improve the model's robustness and generalization by expanding the dataset to include a wider variety of cassava leaf images taken under different environmental conditions and showing various stages of disease. Adding more disease categories, especially rare and newly emerging ones, is expected to increase the model's diagnostic range. The framework could also be adapted for use with other important crops such as maize and rice, allowing for broader application in agriculture. In addition, integrating the model into mobile or edge devices would support real-time, onsite disease detection in areas with limited resources

References

[1] T. Kalpana, R. Thamilselvan, K. Chitra, S. Kaviya, M. Kiruthiga, and C. Mahesh, ”Deep Learning based Approach for Cassava Leaf Disease Detection,” in *Proc. 2024 8th Int. Conf. Electron., Commun. and Aerosp. Technol. (ICECA)*, Nov. 2024, pp. 775–778.

[2] G. Sambasivam and G. D. Opiyo, ”A predictive machine learning application in agriculture: Cassava disease detection and classification with imbalanced dataset using convolutional neural networks,” *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 27–34, 2021.

[3] A. G. C. Junior, F. S. da Silva, and R. Rios, ”Deep Learning-Based Transfer Learning for Classification of Cassava Disease,” *arXiv preprint arXiv:2502.19351*, 2025.

[4] J. Anitha and N. Saranya, ”Cassava leaf disease identification and detection using deep learning approach,” *Int. J. Comput. Commun. Control*, vol. 17, no. 2, 2022.

[5] S. Metlek, ”Disease detection from cassava leaf images with deep learning methods in web environment,” *Int. J. 3D Print. Technol. Digit. Ind.*, vol. 5, no. 3, pp. 625–644, 2021.

[6] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, ”Deep learning for image-based cassava disease detection,” *Front. Plant Sci.*, vol. 8, p. 1852, 2017.

[7] G. Sambasivam, G. P. Kanna, M. S. Chauhan, P. Raja, and Y. Kumar, ”A hybrid deep learning model approach for automated detection and classification of cassava leaf diseases,” *Sci. Rep.*, vol. 15, no. 1, p. 7009, 2025.

[8] U. K. Lilhore et al., ”Enhanced convolutional neural network model for cassava leaf disease identification and classification,” *Mathematics*, vol. 10, no. 4, p. 580, 2022.

[9] E. Mwebaze, J. Mostipak, Joyce, J. Elliott, and S. Dane, ”Cassava Leaf Disease Classification,” Kaggle, 2020. [Online]. Available: https: //kaggle.com/competitions/cassava-leaf-disease-classification.

[10] E. Mwebaze and T. Gebru, ”Cassava Disease Classification,” Kaggle, 2019. [Online]. Available: https://kaggle.com/competitions/ cassava-disease

[11] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[12] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). pmlr.

[13] Scherer, D., Muller, A., & Behnke, S. (2010, September). Evaluation of¨ pooling operations in convolutional architectures for object recognition. In International conference on artificial neural networks (pp. 92-101). Berlin, Heidelberg: Springer Berlin Heidelberg.

[14] Chen, Q., & Yan, S. (2014). Network in network. In International Conference on Learning Representations (ICLR).

[15] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.
[16] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.
[17] Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.